The Optimisation of the Long Term Parking of Aircraft

Victor O'Callaghan

Technische Universiteit Delft

TU Delft

The Optimisation of the Long Term Parking of Aircraft

by

Victor O'Callaghan

to obtain the degree of Master of Science,
at the Delft University of Technology,
to be defended publicly on Friday October 29, 2021 at 13:00.

Student number: 4286138

Thesis committee:	Prof. dr. ir. J. M. Hoekstra,	TU Delft, chair
	Ir. P. Roling,	TU Delft, supervisor
	Dr. ir. A. Bombelli,	TU Delft, member

Cover image source: https://commons.wikimedia.org/wiki/File: KLM_aircraft_parked_on_Schiphol_runway_during_corona_crisis_(cropped).jpg

Preface

This thesis marks the end of my time as a student at the TU Delft Faculty of Aerospace Engineering. I would like to thank my supervisor Paul Roling for his guidance during the frequent progress meetings. In addition, I would like to thank Alessandro Bombelli for his feedback during various milestone meetings.

Victor O'Callaghan
Delft, October 2021

Contents

Introduction

Due to the coronavirus pandemic, air travel passengers decreased rapidly between March and April 2020 as travel bans and stay at home orders were imposed. As a result, airlines were forced to heavily reduce their flight schedule, or even temporarily cease flight operations completely for several weeks. This rises the question of how to park these aircraft efficiently, minimising valuable airport space used.

This problem is different from the well researched gate/stand allocation problem, as for the purpose of long term parking aircraft can be parked anywhere on a given surface at any orientation without the need for using fixed aircraft infrastructure (e.g. passenger boarding bridge) or requiring large margins for service personnel and equipment (e.g. baggage handling or galley service trucks). While the research presented in this document is focused on aircraft parking for the purpose of long term parking, the concepts could also be applied to aircraft parking optimisation within in a (maintenance) hangar, or in the broader scope for general cutting and packing problems.

This report consists of three parts and is structured as follows. The first part contains the scientific paper, which is the main deliverable of the master thesis. The second part consists of supporting work, where some parts of the thesis are explained in more detail and supporting material is presented. Finally, in part three the literature study that was carried out for this thesis is included.

I

Scientific Paper

A Tabu Search Algorithm for the Optimisation of the Long Term Parking of Aircraft

Victor O'Callaghan

Air Transport and Operations, Aerospace Engineering
Delft University of Technology
Delft, the Netherlands

Abstract

The 2020 coronavirus pandemic lead to a virtual standstill of air passenger traffic in the spring of that same year. While some travel restrictions have since been lifted, passenger air travel is not expected to return to pre-coronavirus levels for several years. Then the question arises of how to park the large amounts of grounded aircraft efficiently, minimising valuable airport space used. While aircraft parking for this purpose is a largely unexplored area in academic literature, the problem shows similarities with cutting and packing problems which have been researched for many years. Hence, the proposed model in the paper is modelled similar to that of the irregular strip packing model, where a fixed width is used and the length of the parking layout is to be minimised. Aircraft are represented as non-convex polygons and are allowed to rotate in discrete intervals. The concept of the no-fit polygon (NFP) is used in order to prevent overlap between aircraft. A tabu search algorithm with an adaptive tabu list is proposed in order to optimise the sequence and orientations in which the aircraft are placed onto the placement area using a bottom-left (BL) placement strategy. In order to evaluate the effectiveness of the proposed algorithm, several instances are created and tested using computational experiments.

Keywords: aircraft long term parking, tabu search, no-fit polygon

1 Introduction

During the 2020 global crisis due to the outbreak of the COVID-19 coronavirus, governments throughout the world imposed lock downs or stay at home orders, banning all non-essential travel. As a result, air travel demand decreased rapidly and airlines had to significantly reduce capacity, with some airlines even grounding their entire fleet for several weeks. According to IATA (2020), in April 2020 the amount of traffic (in terms of revenue passenger kilometres) decreased by 94% and capacity (in terms of available seat kilometres) decreased by 87% compared to the same period in the previous year. In order to accommodate all the grounded aircraft, airlines and airports creatively parked the aircraft on runways, taxiways, and/or unused gates and stands. Although travel restrictions are gradually being lifted, it is expected that air travel demand will not return to pre-coronavirus levels for several years (REUTERS, 2020). Consequently, aircraft will need to remain parked for quite some time and the challenge for airlines and airports then becomes to how to park these aircraft efficiently.

During normal airport operations aircraft are parked at fixed airport infrastructure such as gates and stands, where large margins are necessary to allow for aircraft servicing (such as the passenger boarding bridge, cargo/luggage loading, galley service trucks, etc). However, for the purpose of long term parking, such margins are not needed and aircraft can be parked anywhere on a given surface at any arbitrary orientation and close to each other. Aircraft could even be parked in non-conventional ways, with their wings overlapping.

To the best of the author's knowledge, no literature exists on the topic of optimising the long term parking layout of aircraft, or any instance for that matter where aircraft can be parked on a large scale at any arbitrary position and orientation within the given placement area. Although the gate/stand allocation problem has been researched extensively (see e.g. Guépeta et al. (2015)), it is not relevant for the reasons mentioned earlier. Limited research has been carried out recently on the somewhat related problem of optimising the parking of aircraft within a maintenance hangar (e.g. Qin et al. (2018); Li et al. (2019)), however the problem size is naturally limited due to limited space available within a hangar. Additionally, due to manoeuvrability constraints within a hangar, aircraft are usually parked tail or nose first into the hangar and are thus limited to a maximum of two orientations. Due to these limitations, such problems do not relate well to the long term parking of aircraft. Another similar but more general problem of generating tight layouts is found in the field of cutting and packing problems, where e.g. shapes must be cut from a sheet of metal or items must be packed close together in order to minimise wasted material or space. When irregular shapes such as aircraft are involved, this problem is also referred to as the nesting problem. While many papers have been published on cutting and packing of regular shapes without or only a limited amount of possible orientations, the nesting problem that allows many or even continuous rotations in combination with large problem instances has not been researched extensively. The nesting problem is known to be NP-hard (Fowler et al., 1981) and therefore solutions approaches are mainly based on

heuristic methods in order to obtain good solutions in a timely manner.

As airports usually have long rectangular paved areas such as runways, taxiways and remote stands where aircraft could be parked, the objective in this paper is to minimise the total length of such a rectangular area with fixed width for a given set of aircraft, which is similar to the irregular strip packing problem. A tabu search algorithm for optimising the long term parking layout of aircraft is proposed, using a bottom-left placement strategy based on the no-fit polygon (NFP). Aircraft are allowed to rotate in discrete intervals, although more orientations are considered than only the 2 or 4 orientations commonly used for the nesting problem in literature. While this paper focuses on the long term parking of aircraft (on outdoor surfaces), the results could possibly also be beneficial for the purpose of maintenance hangar space optimisation or in the broader scope for nesting problems in general.

The structure of the remainder of the paper is as follows. First, relevant literature is discussed in section 2. In section 3 the formal problem description is given, followed by the solution approach methodology in section 4. The results are presented in section 5. Finally, the conclusion and recommendations for future research are discussed in section 6.

2 Literature Review

The purpose of this section is to provide an overview of the existing literature relevant to the optimisation of the long term parking of aircraft. As was mentioned in section 1, to the best of the author's knowledge, no literature exists on the topic of the optimisation of the long term parking of aircraft. Therefore, the literature review focuses on general geometric modelling techniques commonly used for similar problems (subsection 2.1), a brief overview of common metaheuristics (subsection 2.2), the related problem of aircraft parking within a hangar (subsection 2.3), and finally the more general case of cutting and packing problems, where the focus will be on irregularly shaped items, also known as nesting problems (subsection 2.4).

2.1 Geometric tools

The first obstacle in the aircraft parking problem, and more generally in cutting and packing problems, is related to the geometry: how does one determine if two shapes are overlapping? This question becomes more complex when irregular shapes, such as representations of aircraft, are involved (Leao et al., 2020). Bennell and Oliveira (2008) noted that this geometric problem is not a trivial task and provided a tutorial for the geometry of the nesting problem. They found that the four most common modelling approaches applied to the nesting problem are the raster method, direct trigonometry, the no-fit polygon (NFP), and the phi-functions.

Using a raster representation the continuous placement area is represented by a discrete grid. However, due to the discretisation, objects cannot be represented precisely. Using direct trigonometry, objects can be represented accurately although the computational cost of determining overlap in such approaches is expensive. The no-fit polygon (NFP) represents the relative position of two objects overlapping. Since the NFP is constructed from the original edges of the two objects, overlap can be determined accurately while being more efficient than direct trigonometry approaches. For each orientation a new NFP must be created, hence only a discrete set of rotations ca be considered. Stoyan et al. (2001) introduced the concept of phi-functions, which are used to define the relation between two objects. Although phi-functions allow for continuous rotations, there is no algorithmic approach to obtain the phi-functions and they must be derived by hand.

2.2 Metaheuristics

Some problems can be so complex that it might not be realistically possible to solve for an optimal solution. When exact methods are not able to solve a problem to an optimal solution (within a reasonable amount of time), heuristics are commonly applied to find a good feasible solution, approximating an optimal solution. A heuristic algorithm is based on common sense ideas tailored to a specific problem and is usually an iterative process, searching for a new, possibly better, solution each iteration. Such improvement procedures can, however, get stuck in a local optimum when a problem has multiple local optima. Metaheuristic are general solution methods that have the ability to escape local optima, and thereby directing the search towards the global optimal solution.

The three most common metaheuristics according to Hillier and Lieberman (2015) are the tabu search, simulated annealing, and the genetic algorithm. The key characteristic of a tabu search algorithm is that it continues the search when a local optimum is found by temporarily accepting solutions which are worse than the current solution, and a tabu list temporarily forbids moves in the direction of a previous solution. The simulated annealing approach moves in random directions initially. At the beginning of the search, downwards moves have higher probabilities of being accepted, while this probability gradually decreases throughout the search and as a result mostly only upward moves are accepted in later stages. The genetic algorithm is based on the survival of the fittest principle, where each iteration a pool of solutions is considered. Features of "fit" (i.e. good) solutions are combined in order to create a new pool of solutions and resulting in better solutions each iteration.

2.3 Aircraft hangar parking

A problem related to the long term parking problem is that of the aircraft placement in a maintenance hangar. By optimising the parking positions of aircraft within the hangar, a larger number of aircraft could be serviced simultaneously, benefiting the maintenance service providers. The problem is similar in the sense that a certain number of aircraft have to be parked within a limited

space as efficiently as possible. However, the problem differs from the long term parking problem in certain key aspects: the problem size is automatically constraint due to the maximum amount of aircraft that can be placed in the hangar, and due to manoeuvrability constraints within a hangar aircraft are parked nose or tail first and therefore limiting the amount of orientations.

The existing literature on hangar space optimisation is rather limited, and two main approaches have been proposed. Qin et al. (2018) proposed a two-stage MIP (mixed integer programming) method, based on the concept of horizontal partitioning (Alvarez-Valdes et al. (2013)) of the NFP to ensure aircraft do not overlap. No rotations of aircraft were considered, and this exact approach was able to solve small instances (up to 11 aircraft) to optimality. The work was improved by Qin et al. (2019) by allowing the wing of larger aircraft to extend over the wing of smaller aircraft if the height difference is sufficient, introducing a revised NFP in such cases.

Li et al. (2019) developed a genetic algorithm for aircraft parking optimisation within a hangar. In contrast to Qin et al. (2018), the aircraft are allowed to be placed tail-in or head-in, i.e. allowing two orientations. An aircraft is placed along a reference line selected from a set of predetermined lines superimposed onto the placement area, and moved iteratively along that line until no overlap is present. Such a placement strategy effectively discretises the placement area.

Although these approaches have their limitations, for each method it was concluded that it was an improvement over manual parking planning practices.

2.4 Cutting and packing of irregular shapes

In this subsection the existing approaches for solving cutting and packing problems are discussed. More specifically, problems that involve irregular shapes (such as aircraft) are considered, which are also known as nesting problems. Irregular problems are much harder to solve than problems where e.g. only rectangles are considered, due to the geometric complexities it involves (Leao et al., 2020). Allowing the shapes to rotate adds to this complexity even further. The problem is known to be NP-hard (Błażewicz et al., 1993; Stoyan et al., 2016) which results in the solution approaches mainly using heuristic methods in literature, though some exact approaches have been researched. The main focus in this section is on 2D nesting problems, since for the purpose of long term parking of aircraft, the aircraft are parked on the same surface and then the aircraft's projected 2D shape can be used. Nevertheless, 3D elements could be incorporated, such as the wing of larger aircraft extending over the wing of smaller aircraft, as was already discussed in subsection 2.3.

Several exacts methods were proposed for solving the nesting problem. Alvarez-Valdes et al. (2013) proposed a MIP model where the NFP is divided into horizontal partitions, however no rotations were considered and only instances with at most 10 pieces could be solved to optimality. Toledo et al. (2013) introduced the dotted-board model which discretises the placement area into grid of potential placement points and the corresponding MIP model is solved. The discretatistion allows instances up to 56 pieces to be solved to optimality, although no rotations were considered. Cherri et al. (2016) included (discrete, predetermined) rotations in their MIP model by duplicating the piece as many times as there are allowed orientations, rotated with the respective orientation amount. At most two different orientations were considered and instances up to 12 pieces could be solved to optimality. Although allowing more orientations results in better solutions, it comes at the expense of computational time. Larger instances, or instances where more orientations are allowed, can therefore not be realistically solved to optimality. Stoyan et al. (2016) proposed a non-linear programming (NLP) model which considers continuous rotations using the phi-functions representation, however the model cannot be solved to optimality and therefore heuristic methods were applied.

Therefore, it is concluded that exact methods are not suited for long term parking problem, as larger problem instances and more orientations are allowed for the long term parking problem. The remainder of this section is therefore focused on (meta)heuristic approaches.

Metaheuristic approaches usually represent the problem either as a sequence of pieces to be placed where the sequence is altered in order to find improved solutions, or as a complete physical layout where pieces are moved within the actual layout in order to find improved solutions. The sequence representation is decoded through a placement algorithm in order to obtain the actual layout. A placement algorithm is also used in order to generate a initial layout in the physical layout representation. The bottom-left placement heuristic has been a widely used placement algorithm in literature, which places a piece at the most bottom left feasible position on the placement area. The algorithm can be implemented by superimposing a grid onto the placement area and moving the piece in a stepwise manner along the grid until it cannot be moved anymore and reaches its final position (Mundim et al., 2017). Gomes and Oliveira (2002) derived candidate placement points from the NFPs, where the feasible placement area is outside all NFPs and then the bottom left point of the feasible area is chosen. They observed that this results in placing the piece on one of the vertices of the NFPs, or on the edge intersections of two NFPs. Alternative placement algorithms include TOPOS (Oliveira et al., 2000), which uses a floating origin and pieces can be added from any side, and Liu and He (2006) proposed a placement algorithm based on the principle of lowest potential energy. Rotations can either be considered by the sequence manipulation algorithm, or the placement algorithm can determine the best (local) orientation during placement.

Typical moves in the sequence representation in the search for improved solutions are insert moves, swap moves, or orientation changes. The metaheuristics tabu search (Burke et al., 2006; Ramakrishnan et al., 2008) and genetic algorithms (Liu and He, 2006; Mundim et al., 2017) have been applied to guide the search for problems using the sequence representation. The bottom left

placement algorithm was used to decode the sequence into a layout in all approaches except (Liu and He, 2006), who used the lowest gravity principle. Abeysooriya et al. (2018) proposed an innovative jostle heuristic, which alternates between packing from left to right and right to left and where the x-coordinate of the previous iteration is used to determine the packing sequence of the current iteration.

When the search is over the actual physical layout, overlap is usually allowed during the search process. The algorithm then attempts to resolve the overlap and once no overlap is present (i.e. a local optimum is found), the length of the placement area is reduced by a certain margin or percentage. The reduction is likely to introduce overlap again, and the process is repeated until the algorithm is unable to find a feasible solution and the best solution found is reported as final solution. Ramakrishnan et al. (2008) proposed a tabu search method, selecting the piece with the most overlap and placing it in a position with least amount of overlap, where the NFP approach is used to generate candidate placement points. Egeblad et al. (2007) finds the position of minimal overlap of a piece by translating the piece either horizontally or vertically using the intersection area theorem and the guided local search is used to guide the search. A similar approach was taken by Umetani et al. (2009), but used the penetration depth as measure of overlap. Sato et al. (2019) proposed a guided local search approach based on a discrete raster placement area and introduced the novel concept of the raster penetration map, which is derived from the NFP and represents the raster penetration depth between two pieces. Although the solution space is discretised, the method matched or improved 9 out of the 15 benchmark data sets tested.

There is no consensus in literature as to which representation is superior, as with both representations researchers continue to publish good results. However, Ramakrishnan et al. (2008) compared both representations and observed that the layout representation could get stuck resolving overlap. The sequence representation does not suffer from this problem, since the sequence has to be decoded through a placement algorithm and therefore the final layout will always be a feasible solution.

3 Problem description

The long term aircraft parking problem is modelled similar to that of the two dimensional irregular strip packing problem. The rectangular placement area considered (representing e.g. (part of) a runway) has a fixed width W and the objective is to minimise the total length L of the parking layout, as can be seen in Figure 1.

Aircraft are allowed to extend over the area outside the paved surface area which is not suitable for bearing the load of an aircraft (e.g. grass next to the runway), while the landing gears must remain within the load-bearing area. In addition, the maximum allowed length of the wings extending over the non load-bearing placement surface can be specified for each side of the placement area individually (δ_{upper}, δ_{lower}, δ_{side}), which can be useful if

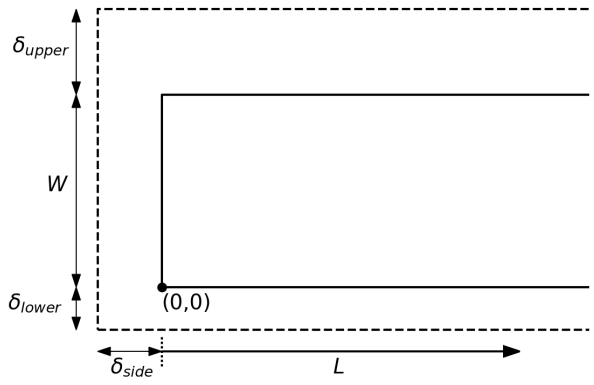


Figure 1: Problem layout.

one side must remain clear because of e.g. an adjacent active taxiway. The solid line in Figure 1 thus represents the loading-bearing placement area and the gears of all aircraft must remain within that area, while the dashed line represents the allowed overhang area and all aircraft must remain within that area.

Aircraft are represented as non convex polygons enclosing the aircraft as can be seen in Figure 2 (dotted line). For tail mounted engines, the polygon points for the engine are omitted as the engines will be included in the tail section. Conversely, additional polygon points are introduced for quad engine aircraft. The reference position of the aircraft is pointing upwards, with its reference point at its left trailing edge tail, and a positive rotation results in a counter clockwise aircraft rotation. The aircraft are allowed to rotate in discrete intervals. The wing of a larger aircraft can be allowed to extend over the wing of smaller aircraft, if the difference in wing height allows for such a placement. In order to ensure the gear remains on the load bearing area, a polygon around the gear is defined (dashed line in Figure 2).

A set of aircraft parameters are given as input to generate the aircraft and gear polygons, which are derived from aircraft airport planning manuals provided by aircraft manufactures (e.g. Airbus (2005))

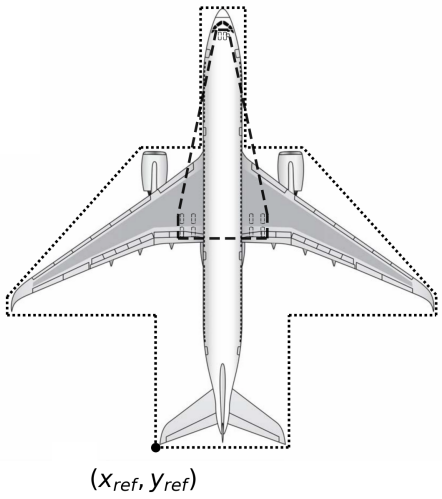


Figure 2: Aircraft (dotted) and gear (dashed) representation and its reference point (adapted from (Airbus, 2005))

4 Methodology

In this section the theoretical concepts for the construction of the optimisation algorithm are explained. A common strategy for solving similar problems, such as cutting and packing problems, is to search over the sequence (and rotations) of items to be placed. This sequence with the corresponding orientations is then decoded through a placement algorithm to obtain an actual layout, and new sequences with corresponding (new) orientations are found through a heuristic algorithm. The advantage of such an approach is that layout is guaranteed to be feasible because the placement rules of the placement algorithm ensure items are placed at a feasible location, i.e. within the placement area and with no overlap. In contrast, solution approaches searching over the actual physical layout allow for items to overlap during the search and hence the algorithm could potentially get stuck in resolving overlap, resulting in an infeasible solution (Ramakrishnan et al., 2008). For that reason, the algorithm proposed in this paper follows the former approach as the general solution strategy.

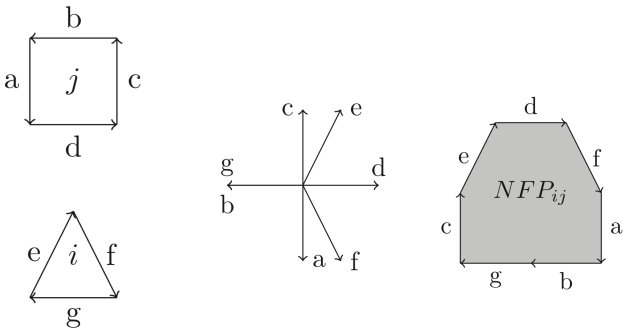
First, the concept of the no-fit polygon (NFP) as a geometric tool is introduced in subsection 4.1. Next, the placement algorithm is described in subsection 4.2. Finally, the tabu search algorithm is described in subsection 4.3.

4.1 No-fit Polygon (NFP)

The no-fit polygon (NFP) is a geometric tool commonly used for similar problems such as the cutting and packing problems, and has also been applied to aircraft parking optimisation within a maintenance hangar. The boundary of the NFP represents the relative position of the reference points of the two items where the two items touch each other. If the reference point of the item placed within the NFP, the two items overlap; if it is placed outside the NFP no overlap is present (and the two items do not touch). The NFP can be understood as sliding an item (i.e. the orbiting piece) around the other item (i.e. the fixed piece) and tracing its reference point, where the resulting line is the NFP.

The method used to construct the NFP between two items used in this paper is based on the approach originally proposed by Cuninghame-Green (1989) for the case of two convex polygons and works as follows. First, the edges of polygons i and j are oriented in a clockwise and counterclockwise manner respectively, where i is the fixed polygon and j is orbiting polygon (Figure 3a). Next, the edges are translated to start at a single point (Figure 3b). In the final step the translated edges from Figure 3b are linked together in a clockwise manner, resulting in the NFP_{ij} (Figure 3c).

Although the aircraft representation introduced in section 3 is not a convex polygon, it can easily be decomposed into several smaller, convex sections (namely a nose section, mid/wing section, and tail section). The NFP is then generated for each of the sections individually, and by recombining the separate section NFPs, the full NFP for two aircraft can be obtained. An example of the NFP between two aircraft is shown in Figure 4, where AC_i is the fixed polygon and AC_j is the orbiting polygon.



(a) Pieces i and j clockwise and counterclockwise oriented respectively

(b) Edge translation to single point

(c) Concatenate edges clockwise to create NFP_{ij}

Figure 3: Cuninghame-Green (1989) method for no-fit polygon generation (Cherri et al., 2016).

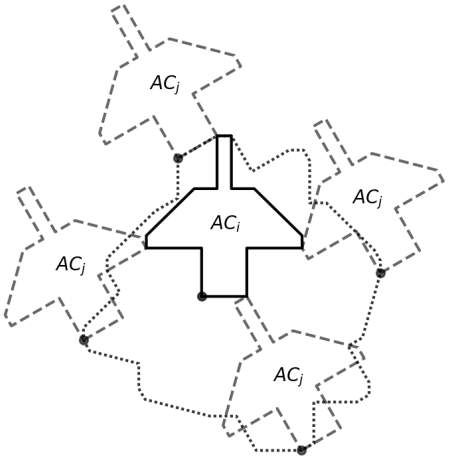
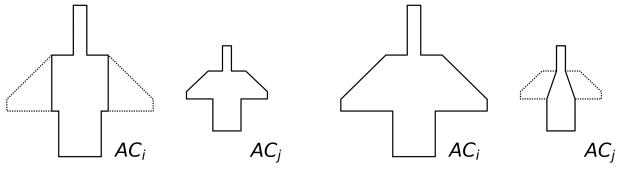


Figure 4: Example of the NFP (dotted line) for aircraft AC_i and aircraft AC_j. Aircraft AC_j is plotted multiple times at various arbitrary locations to show its relative position with respect to AC_i.

In instances were the wing height difference between aircraft types allows for overlap, the NFP is modified. The wing of the higher wing aircraft can extend over the lower winged aircraft, but cannot protrude its fuselage. In addition, the position of the engines of the higher wing aircraft should be considered such that the lower wing aircraft's wing is not protruding the engines. Therefore, in order to take into account those two conditions, two NFPs are generated: one for the the full lower wing aircraft and the higher wing aircraft where the outer wing is cut-off after the (outer) engines, and a second for the lower wing aircraft's fuselage (and including its tail) and the full higher wing aircraft. The aircraft representations used to generate these two NFPs are shown in Figure 5a and Figure 5b, respectively. The two resulting NFPs are combined to obtain the complete modified NFP and an example is shown in Figure 6. The hatched area of the wing of AC_j can be placed underneath the part of the wing of AC_i extending beyond its engines, marked by the grey area.



(a) (b)

Figure 5: Modified aircraft representations for NFP generation such that lower wing aircraft AC_j can be placed underneath the higher wing aircraft AC_i without overlapping the fuselage or engines. Figure 5a prevents overlap between AC_j and the fuselage and engines of AC_i, Figure 5b prevents overlap between AC_i and the fuselage of AC_j.

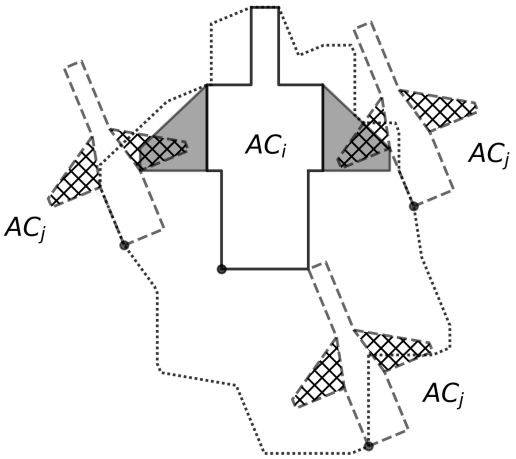


Figure 6: Example of the modified NFP (dotted line) for aircraft AC_i and aircraft AC_j, where the wing of AC_j can be placed underneath the wing of AC_i.

In reality, aircraft are not placed touching each other and therefore a safety margin is desired between aircraft. A safety margin can be imposed by extending the boundary of the aircraft with the desired margin (Figure 8a). By definition of the NFP, when the reference point of the orbiting aircraft AC_j is placed on the boundary of the NFP, the two aircraft touch. Hence moving the edges of the NFP outwards is equivalent to adding a safety margin around the aircraft (Qin et al., 2019) (blue region in Figure 8b). Therefore, in order to add a safety margin sf between aircraft, the edges of the NFP are moved outwards by distance sf resulting in the revised NFP (Figure 8c).

Using concept similar to the NFP, the inner-fit polygon, or IFP, can be created. Instead of sliding around a fixed item as is done with the NFP, the item slides within the other item (Figure 7). The NFP is used to verify an item remains within the placement area. Since the wing is allowed to extend over non-load bearing surfaces, one IFP is generated for the landing gear and the load bearing placement area, and a second IFP is generated for the entire aircraft and the overhang area. The final IFP is then the limiting side of each individual IFP, such that the landing gear and the entire aircraft remain within their respective boundaries.

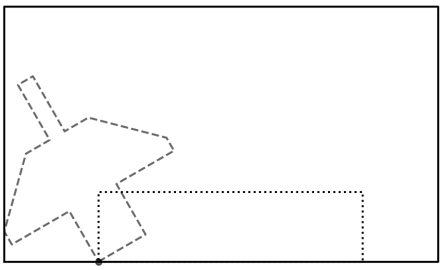


Figure 7: Example of the IFP (dotted line) for an aircraft and the placement area (solid line).

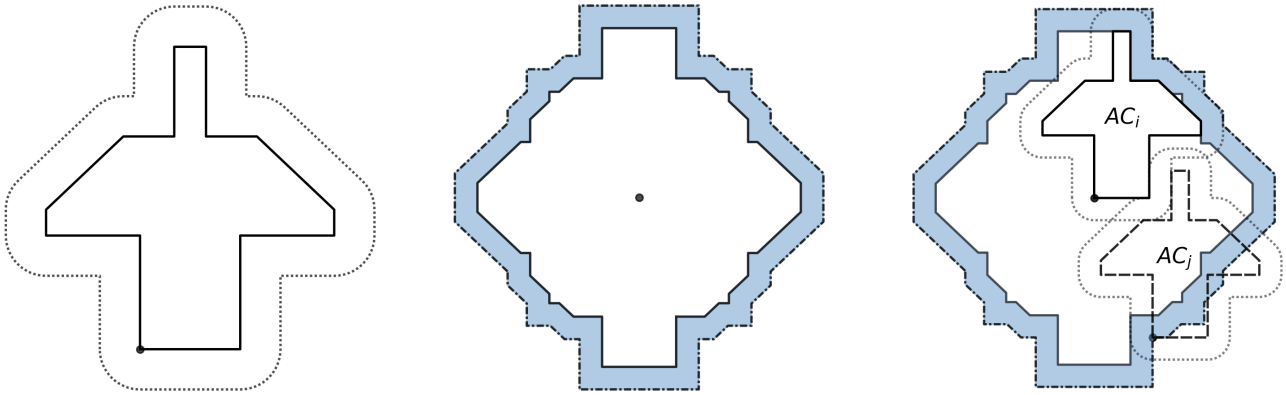
The NFPs and IFPs are generated in a preprocessing phase, where for each aircraft type and for each orientation combination a NFP (IFP) is generated and stored in the computer's memory. The placement algorithm is then able to retrieve the NFPs (IFPs) during the placement process in the optimisation phase.

4.2 Placement Algorithm

In order to place a sequence of aircraft with their corresponding orientations onto the placement area, the single pass bottom-left (BL) placement algorithm is used.While originally proposed by Art (1966), it has become a widely used placement strategy and is still relevant today (see e.g. Abeysooriya et al. (2018); Mundim et al. (2017)). In the context of the current problem of parking aircraft, illustrated in Figure 1, the BL algorithm is equivalent to placing an aircraft as far to the left as possible (i.e. lowest feasible x-coordinate) and in case there are multiple such positions possible, the aircraft is placed at the bottom most position (i.e. of the positions with lowest x-coordinate, the position with the lowest y-coordinate is chosen).

As the placement area is a continuous space, the NFPs and IFPs are used to obtain a feasible placement area for an aircraft, also known as the collision free region (CFR), from which a set of placement points can be derived and the bottom-left position is then easily selected from those points. While the CFR still is a continuous region, when looking to place an item at its BL position on the placement area, only the vertices of the boundary of the CFR can be considered (Gomes and Oliveira, 2002). The CFR for the current aircraft to be placed is obtained by subtracting the NFPs for each aircraft already placed from the IFP of the current aircraft.

The process of placing an aircraft sequence and their corresponding orientations is illustrated in Figure 10. For the first piece, only the corresponding IFP is retrieved, since no other aircraft have been placed onto the layout yet. The CFR is then simply equal to the IFP and hence the aircraft is placed at the BL coordinate of the IFP (Figure 10a). For the second aircraft to be placed, in addition to its IFP, the NFP between the aircraft already placed and the aircraft to be placed is retrieved (Figure 10b). By subtracting the NFP from the IFP, the CFR is obtained from which the BL coordinate of its vertices is selected to place the aircraft (Figure 10c). This process is repeated until all aircraft for a given sequence are placed onto the placement area. Note that this approach can place smaller aircraft into empty spaces in the partial layout when placing a sequence, as can be seen in Figure 10c, which otherwise would not be filled using e.g.



(a) (b) (c)

Figure 8: Safety margin around aircraft (a) and revised NFP (b,c)

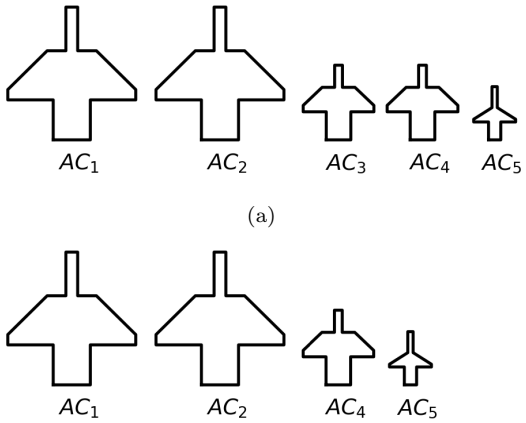
a sliding or translate approach.

4.3 Tabu Search

In order to search over the sequence and aircraft orientations, the tabu search (TS) metaheuristic is used. A tabu search includes a local search procedure which aims to improve the current solution. A key feature of the tabu search is that when no improving moves can be found (i.e. a local optimum is reached), the search is allowed to continue by accepting non-improving moves and is therefore able to escape local optima. To avoid immediately circling back to the same local optimum, a tabu list temporarily stores recent moves. Moves on the tabu list are not allowed, unless that move would result in a better solution than the best (global) solution found in any of the previous iterations (also known as the aspiration criterion).

Problem encoding

For the tabu search the aircraft sequence and orientations are encoded as a binary string. The first part of the bitstring represents the sequence, and the second part represents the orientation for each aircraft. The sequence part of the bitstring, bits are used to point to each aircraft in their initial sorting, which is the sequence in which aircraft are sorted by non-increasing area and all bits are set to zero. For the orientations part of the bitstring the possible orientations for each aircraft is given by the variable *orient_bit_size* resulting in $\frac{360^\circ}{2^{orient\_bit\_size}}$ orientations. Their zero position is pointing upwards and orientations are defined counterclockwise as was described in section 3.



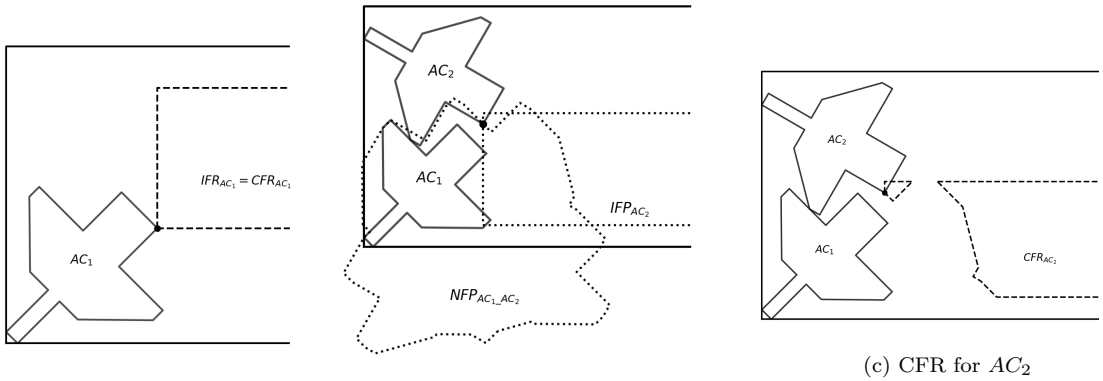
(a)

(b)

Figure 9: Aircraft sequence

The concept is best illustrated using an example. Consider the case for five aircraft, which are sorted by non-increasing area as described above and are given the following numbering: AC_1, AC_2, AC_3, AC_4, AC_5 (Figure 9a). Consider the sequence bitstring 010 11 10 1; in order to point to any of the 5 aircraft in the initial list, 3 bits are needed ($\log_2(5)$, rounded up). Hence, in this case the first three bits in the bitstring (010) are used to select the first aircraft to be placed. The binary number 010 represents the number 2 ($0*2^2+1*2^1+0*2^0$), add 1 to account for bits numbering starting at zero, and thus points to the third aircraft in the initial list, AC_3. Aircraft AC_3 is therefore selected to be placed first, and the remaining aircraft are AC_1, AC_2, AC_4, AC_5 (Figure 9b). In order to point to any of the 4 aircraft in the remaining list of aircraft (Figure 9b), now only two ($\log_2(4)$) bits are needed. Therefore, the next two bits in the bitstring (11) represent the number 3, add 1 to account for bits numbering starting at zero, and thus point to the fourth aircraft in the remaining aircraft list, AC_5. The process is repeated until all aircraft are placed. The full sequence bitstring 010 11 10 1 for example therefore represents the sequence of aircraft to be placed in the order $AC_3, AC_5, AC_4, AC_2, AC_1$.

Note that in order to select the first aircraft, 3 bits were needed. However, 3 bits gives $2^3 = 8$ possibilities, while there are only 5 aircraft. Bits pointing to an aircraft number greater than the remaining amount of aircraft (5 in this case) would not result in a meaningful aircraft selection (e.g. if the first 3 bits were 111, it would refer to the (nonexistent) eighth aircraft in the initial list). In order to obtain a meaningful result in such cases, the total amount of remaining aircraft is subtracted from the binary number. Hence, 111 would point to the eighth aircraft, subtracting the total remaining aircraft (i.e. 5 in this case), results in the third aircraft AC_3 being selected



(a) IFP=CFR for AC_1

(b) NFP and IFP for AC_2

(c) CFR for AC_2

Figure 10: Placement process

for placement.

The orientation for each aircraft is described by a certain amount of bits ($orient\_bit\_size$). The amount of bits dictate the amount of discrete orientations possible and their increments. E.g. a 5 bit orientation representation results in $2^5 = 32$ possible orientations in $\frac{360^\circ}{2^5} = 11.25^\circ$ increments. The total amount of bits needed to encode the orientations of all aircraft is equal to the amount of aircraft times number of bits per orientation.

Basic tabu tools

Using the binary sequence and orientation encoding, the local search method used for the tabu search is done by flipping bits. A move is defined as flipping one bit in the current bitstring, and each iteration each bit in the current bitstring is flipped to obtain new candidate bitstrings. The neighbourhood size is therefore automatically dependent on the length of the bitstring, which in turn is dependent on the amount of aircraft to be placed and the bitsize used for representing the orientations. The bitstring is decoded into a layout through the placement algorithm described in subsection 4.2 to obtain corresponding length layout L. The bitstring corresponding to the layout with the lowest length where none of the bits are tabu is chosen as the new current bitstring. If two bitstrings result in the same objective value, a bitstring is chosen by evaluating the bitstring's binary numerical value and the binary string with the lowest numerical value is selected as the current bitstring.

The tabu list size in an ordinary tabu search is set at a fixed value. As will become clear in subsection 5.1, a fixed tabu list size might not always yield good results. Therefore a reactive tabu search (RTS) is proposed, based on the method described by Battiti and Tecchiolli (1994), which can automatically adapt the tabu list size to the problem and the progression of the search. While a simple adapting tabu list could avoid cycling of the search, it can still become trapped in a region of the solution. Therefore, a diversifying escape mechanism is included when the former mechanism is inadequate.

Reactive Tabu Search (RTS)

Each bitstring bit_str that is encountered during the search is saved, together with the iteration number it was last encountered $\Pi(bit\_str)$ and how many times it has been encountered ($\phi(bit\_str)$). If a bitstring is encountered a value greater than REP times, the counter $chaotic$ is increased by 1 ($chaotic = chaotic + 1$) which counts the number of often repeated bitstrings. If the value of chaotic is greater than the threshold $CHAOS$, a diversifying mechanism is triggered. INC and DEC determine by how much the ts_list_size is increased or decreased and are constant throughout the search. A moving average of the detected $cycle\_length$ is stored in $moving\_avg$ and $steps\_since\_last\_ts\_change$ records when the ts_list_size was last changed. If the detected $cycle\_length$ ($=iter\_count - \Pi(bit\_str)$) is lower than $cycle\_max$, the ts_list_size is increased ($ts\_list\_size = INC \times ts\_list\_size$). The search is stopped after $max\_non\_improve$ iterations and the bitstring corresponding to the $best\_so\_far$ solution is returned.

While the algorithm limits cycling between the same bitstrings, the algorithm can still be trapped in a region of the solution space. Therefore, the diversification procedure is also triggered in the following cases. When the of_rep_count exceeds 100 consecutive repeating objective value functions, and when there has not been an improvement in the $best\_so\_far$ in the past 500 iterations.

The diversification strategy adds a penalty to the candidate objective value and is based on the flip frequency of the bits. The (candidate) bitstrings are generated by a bit flipping procedure; if the bit flipped corresponding to the (candidate) bitstring is flipped frequently, a higher penalty is associated with that bitstring. The penalty for a given bitstring is defined as $penalty = \frac{bit\_flip\_count}{max\_bit\_flip\_count} \times div\_param \times best\_so\_far$, where $max\_bit\_flip\_count$ is the value of the bit flip count of the bit which is flipped most often. The exception is in the case where the objective value associated with a candidate bitstring is lower than $best\_so\_far$. In that case, the penalty is set to 0 and the value $best\_so\_far$ is updated. The main RTS algorithm is shown in pseudo code in algorithm 1 and the reaction procedure is shown

in algorithm 2

The initial bitstring can be a zero bitstring (all bits are set to zero when aircraft are sorted by non-increasing area and pointing upwards), a random bitstring, or can be generated by a greedy best fit heuristic. The greedy heuristic used in this paper first sorts all aircraft by non-increasing area. Then for each aircraft each admissible orientation is tested and the orientation resulting in the lowest partial layout length is chosen. The process is repeated until all aircraft in the sorted listed are placed. The corresponding orientations can then be translated into bits to obtain the initial bitstring (the sequence parts is set to zero since the aircraft are sorted by non-increasing area, as was mentioned earlier).

Algorithm 1 Reactive Tabu Search pseudo code

```
Sort aircraft by non-increasing area

iter_count = 0
non_improve_count = 0
steps_since_last_size_change = 0
ts_list_size = 1

while non_improve_count < max_non_improve do
    Find and evaluate candidate bitstrings
    DIV = Check_for_repeating_bitstrings(bitstring)

    if DIV is False then
        Choose best non tabu bitstring
        Update count, last observed, bit flip count
    else Diversify:
        for i in moving_avg do
            for i in Candidate bitstrings do
                if obj_val(candidate bitstring) < best_so_far then
                    obj_val = obj_val(candidate bitstring)
                else
                    obj_val = obj_val(candidate bitstring)
× bit_flip_count / max_bit_flip_count × div_param × best_so_far
                end if
            end for
            Choose best non tabu bitstring
            Update count, last observed, bit flip count
        end for
    end if

    if obj_val < best_so_far then
        non_improve_count = 0
        best_so_far = obj_val
    else
        non_improve_count += 1
    end if
end while
Return best solution found best_so_far
```

Algorithm 2 Reaction and diversification mechanism

```
procedure CHECK_FOR_REPEATING_BITSTRINGS(bit_str)
    DIV = False
    steps_since_last_size_change += 1
    Check for bitstring in memory

    if bitstring found previously then
        cycle_length = iter_count - Π(bit_str)
        Π(bit_str) = iter_count
        φ(bit_str) += 1
        if φ(bit_str) > REP then
            chaotic += 1
            if chaotic > CHAOS then
                DIV = True
                chaotic =0
            end if
        end if
        if cycle_length < cycle_max then
            moving_avg = 0.1 × cycle_length + 0.9 × moving_avg
            ts_list_size = ts_list_size × INC
            steps_since_last_size_change = 0
        end if
    else
        Store bit_str and Π(bit_str) in memory
    end if

    if steps_since_last_size_change > moving_avg then
        ts_list_size = ts_list_size × DEC
        steps_since_last_size_change = 0
    end if

    if non_improve_count > max_non_improve or of_rep_count >
max_of_rep then
        DIV = True
        non_improve_count = 0
        of_rep_count = 0
    end if
end procedure
return DIV
```

Table 1: Default input values and algorithm settings

Parameter	Value
$init\_bit\_str$	zero bitstring
$orient\_bit\_size$	5
W	60
δ_{upper}	20
δ_{lower}	20
δ_{side}	10
$wing\_overlap$	True
$max\_non\_improve$	1000
INC	1.3
DEC	0.9
$CHAOS$	3
REP	3
$cycle\_max$	250
max_of_rep	100
div_param	1

5 Results

In this section the results of computational experiments are presented. The methods described in the previous sections are coded in Python, relying heavily on the *Shapely* library for the geometric calculations. Parallel processing is used in order to evaluate multiple moves simultaneously. The experiments are run on a computer with a 3 GHz 6-Core Intel Core i5-8500B processor with 32GB of RAM. Unless explicitly mentioned otherwise, the algorithm settings shown in Table 1 are used.

5.1 Tabu List Size

A key element of a tabu search is of course the tabu list. The length of the tabu list (also known as tabu tenure) is usually determined early on through experimental analysis. In order to find an appropriate tabu list length for the aircraft parking problem, several instances with different problem sizes and varying tabu list lengths are tested (while all other parameters remain the same). 4 different instances, with 4, 8, 12, and 16 Airbus A350-900s, are tested for this purpose. A 5 bit representation was used for the orientations, the width of the placement area was $W = 60$, overhang values of $\delta_{upper} = 20$, $\delta_{lower} = 20$, $\delta_{side} = 10$ were used, and no diversification strategies were applied during the search at this stage. The results are shown in Figure 11, and are expressed as relative percentages in order to compare each problem instance.

As is expected, at the extreme ends of the different

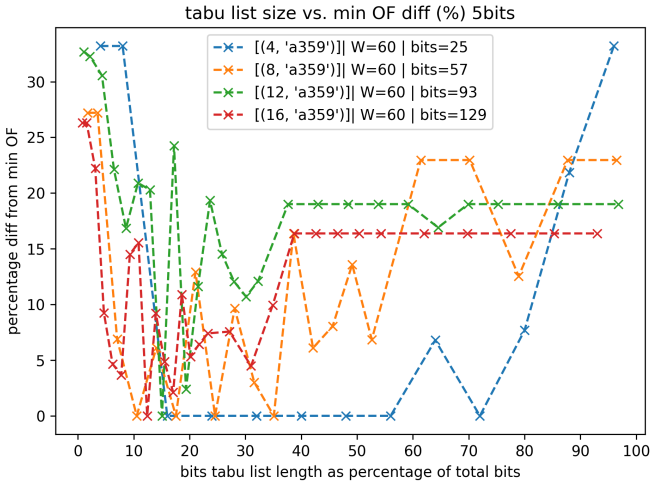


Figure 11: Relative layout length for varying tabu list size expressed as percentage of the total bits for that problem instance

tabu list lengths tested the worst results are observed. If the tabu list is too short, the search will simply cycle back and forth between the same solutions. When the list is too long, it is likely the search does not reach an area with good solutions. However, no clear optimum is observed in the middle tabu list length values and there is a high degree of variability between adjacent data points. Most notably, in the instance of 12 A359 aircraft, the length of the final layout increases 25% from its minimum value when the tabu list sizes increases from 16 to 18 (Figure 11). This indicates a hilly solution space which is difficult to navigate and the algorithm is sensitive to its parameter settings. Hence, using a fixed tabu list may result sub optimal results for specific problem instances, while other instances may yield good results. Therefore, the reactive tabu search with a variable tabu list size is used in the remainder of this paper.

Using this strategy, the best solutions found for each problem instance with fixed tabu list length from Figure 11 were matched in initial experimental analysis. As this removes the need to define a fixed tabu list length, it is concluded an adaptive tabu list is likely the better strategy to be adopted for the aircraft parking problem.

5.2 Orientation bits

The orientation bits define the discrete set of orientations an aircraft is allowed to rotate. More orientations could lead to better solutions, however, more orientations could also increase the complexity of the solution space and lead to longer run times (when not using a fixed time limit) or to search can become trapped. In this subsection, the effect of varying the orientation bit representation is investigated. All other parameters equal, the orientation bits are varied from two bits (i.e. $2^2 = 4$ orientations) up to six bits (i.e. $2^6 = 64$ orientations). Two instances were tested, one consists of 12 Airbus A350-900 aircraft (Figure 12b), the second problem includes a mix of aircraft types namely 3 Airbus 350-900, 4 Airbus A320neo, and 6 Bombardier CRJ200 (Figure 12a).

From Figure 12 the general trend is that the layout length decreases as the orientation bits resolution increases, while computational time increases.

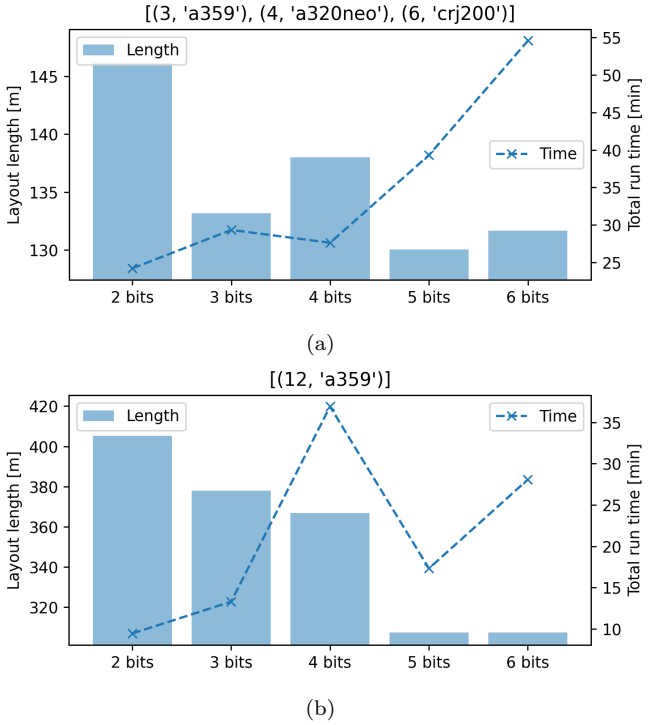


Figure 12: Variation in orientation bits and resulting layout lengths and algorithm run time.

There are some exceptions though, the run with 4bits for the instance with multiple aircraft types in Figure 12a results in a longer layout compared to 3bit run. Since the computational time is also longer, it appears that the search did not reach a favourable region of the search space when the stopping criterion was met. The 12 A359s instances in Figure 12b show a peak in run time at 4bits. This indicates the search only reached favourable solutions later in the search. The corresponding length however is in line with expectations (i.e. lower than 3 bits, higher than 5 bits).

The layouts resulting from the test runs in Figure 12b are plotted in Figure 13 in order to visually understand how the orientation bit resolution affects the layout. The benefits of using more orientations than typically found in literature (i.e. none or two orientations) is clear, as the worst results are consistently found at lower bits resolutions. It is up to the user to determine the trade-off between orientation bits resolution and computational complexity.

5.3 Safety Margin

The safety margin is used in order to maintain a safe distance between aircraft and gives some margin when manoeuvring aircraft into their position. In this section it is investigated what the effect on the final layout is when the safety margin is increased. In this instance 2 CRJ200s, 2 A320neos, 2 A359s are considered with safety margins of 0, 1, 3, 5, 10 meters. The resulting layouts are shown in Figure 14. As is expected, the length of the resulting layouts increase as the safety margin is increased.

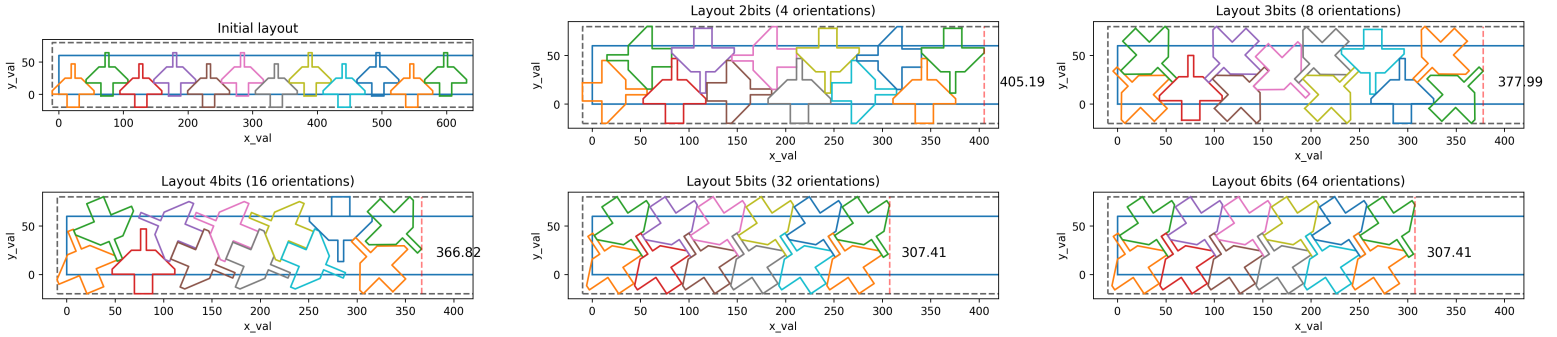


Figure 13: Resulting layout for varying orientation bits, starting from the same initial layout

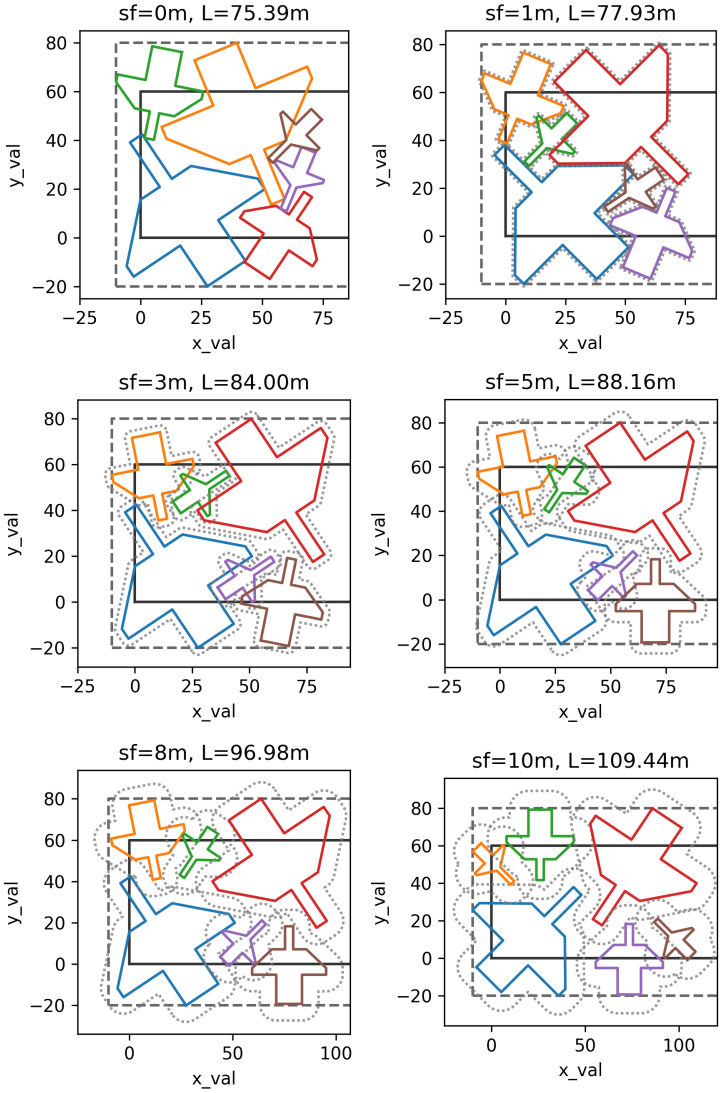


Figure 14: Results for increasing safety margins

5.4 Initial Solutions

A tabu search can be sensitive to the initial solution used for starting the search. In the previous experiments the zero bitstring was used for all problems, in this subsection the effect of using other starting solutions is investigated. The results from selected instances from the previous sections are compared to the results when started from an initial solution generated by the greedy best fit heuristic (see subsection 4.3 for a description of the heuristic).

It is not immediately clear which initial solution is better. Although the results are within roughly $< 1\%$ of each other, the notable exception here is the [12 $A359$] instance, where the greedy initial solution results in a 10% longer layout. One possible explanation for this behaviour when starting from a seemingly promising initial solution is that the search becomes trapped in that region of the solution space (and the diversification strategies are not powerful enough to escape). These results show that the model is indeed sensitive to its input solution.

Table 2: Resulting layout lengths for starting from zero bitstring or greedy initial solution

Instance	Zero bitstring	Greedy
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $5bit$	130.03	128.82
[12 $A359$], $5bit$	307.41	336.90
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 0$	75.39	75.72
[8 $A380$]	239.97	239.87

5.5 Case study

In this subsection the model is applied to a real-world scenario. Here, 8 Airbus A380-800 (among others) are parked at the Southern California Logistics Airport, near the beautifully named city Victorville, USA (Figure 15). The aircraft can be enclosed by a rectangle measuring approximately $345m \times 160m$. For the purpose of this analysis, the shorter edge $160m$ is assumed as the fixed width of the placement area, no overhang is allowed and a safety margin of $5m$ is chosen in order to resemble similar conditions.

The result from the RTS algorithm can be seen in Figure 16. The length of the layout is 239.97m, which is a reduction of roughly 30% compared to Figure 15.

Figure 15: A380s parked at the Southern California Logistics Airport (Google, 2021)

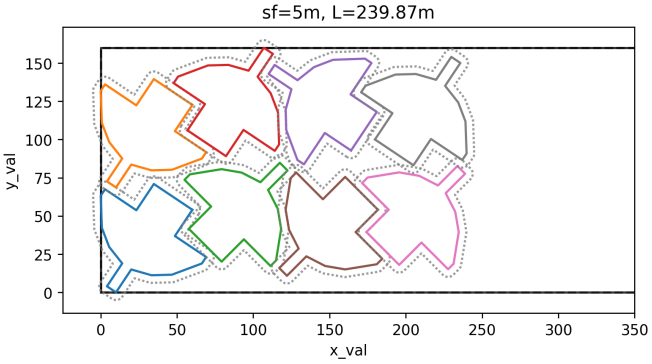


Figure 16: Resulting layout for A380 parking

5.6 Results Summarisation

In this section the results are summarised in the form of a table (Table 3), in the same order as they appeared in the text. The instance is described in the first column together with the variable of interest for that specific experiment, all other settings are equal to the values given in Table 1. In addition, the result of the greedy initial placement strategy described in subsection 4.3 for each instance is given in the last columns as a reference benchmark for the results obtained by the tabu search. In all cases, the final length from the tabu search were better than the greedy heuristic. Note that the total time and greedy time columns are excluding the pre-processing stage for creating the NFPs(and IFPs), which is shown separately in the column NFP time.

6 Conclusion & Recommendations

In this paper a tabu search algorithm was proposed, where the search was over the sequence and orientations of aircraft to be placed using a bottom-left placement strategy. The no-fit polygon was used as a geometric tool to ensure aircraft do not overlap. An adaptive tabu list was adopted in order to automatically adjust the tabu size to the problem and progression of the search. The results of computational experiments showed that the algorithm is highly sensitive to input variables and algorithm settings, indicating a hilly solution landscape. Nevertheless, the layouts generated show a tight placement of aircraft and are guaranteed to be feasible. The results show an improvement over greedy single-pass placement heuristics. Furthermore, a case study showed a decrease in layout length compared to the real-world scenario.

Recommendations

As the long term aircraft parking problem is an unexplored research area (expect for the tangential but different problem of aircraft hangar parking optimisation), there are many avenues for future research. In the proposed tabu search the search was over the sequence and orientations of aircraft to be placed, using a bottom-left placement heuristic. However, different local search procedures, metaheuristics or matheuristics, and placement strategies could be explored or developed for the purpose of long term aircraft parking. Since the model has been shown to be sensitive to its input and settings, more

Table 3: Results table

Instance	Length bitstring [-]	NFP time [s]	Best found [m]	Iterations [-]	Total time [s]	Avg. iter time [s]	Greedy sol length [m]	Greedy time [s]	Length decrease from greedy algorithm [%]
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $2bit$	63	0.54	146.14	2240	1451.35	0.65	160.48	0.01	-8.94
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $3bit$	76	1.88	133.16	2096	1757.49	0.84	169.21	0.01	-21.30
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $4bit$	89	6.93	138.02	1854	1649.74	0.89	166.57	0.02	-17.14
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $5bit$	102	27.89	130.03	2218	2331.25	1.05	141.55	0.04	-8.14
[3 $A359$, 4 $A320neo$, 6 $CRJ200$], $6bit$	115	110.91	131.68	2760	3164.11	1.15	140.51	0.07	-6.28
[12 $A359$], $2bit$	57	0.13	405.19	1058	567.06	0.54	409.07	0.00	-0.95
[12 $A359$], $3bit$	69	0.24	377.99	1362	798.16	0.59	437.31	0.00	-13.56
[12 $A359$], $4bit$	81	0.56	366.82	3086	2213.80	0.72	437.31	0.01	-16.11
[12 $A359$], $5bit$	93	2.28	307.41	1320	1037.00	0.78	336.90	0.01	-8.75
[12 $A359$], $6bit$	105	8.45	307.41	1931	1675.52	0.87	336.90	0.02	-8.75
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 0m$	41	26.53	75.39	2278	396.86	0.17	88.46	0.04	-14.76
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 1m$	41	28.58	77.93	1767	318.52	0.18	90.17	0.04	-13.57
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 3m$	41	28.99	84.00	2872	483.28	0.17	112.43	0.04	-25.29
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 5m$	41	27.38	88.16	2983	478.65	0.16	117.24	0.04	-24.80
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 8m$	41	27.63	96.98	1568	251.05	0.16	132.79	0.04	-26.97
[3 $A359$, 2 $A320neo$, 2 $CRJ200$], $sf = 10m$	41	27.43	109.44	1791	299.44	0.17	143.66	0.04	-23.82
[8 $A380$]	57	3.01	239.97	2971	933.05	0.31	253.00	0.01	-5.51

robust procedures should be investigated. Furthermore, the proposed algorithm assumes a placement area with a fixed width and infinite length. The algorithm could be extended by considering a placement area with fixed dimensions, and selecting which aircraft of a given set should be parked and which aircraft remain unparked to obtain the highest space utilisation. Or even multiple different areas with fixed dimensions can be considered (similar to bin packing problems).

From a practical perspective, some layouts generated by the proposed algorithm might not be possible in practice due to aircraft tugs unable to move aircraft into the required position. In addition, maintenance requirements may require aircraft to be moved to avoid tyre deformation, or engines to be run periodically such that aircraft cannot be placed in the neighbourhood of the exhaust area (and which could require stairs access to the aircraft). Such requirements are not taken into account in the presented algorithm. Within the broader scope of cutting and packing problems, the irregular shape cutting/packing (also known as nesting) problem with many (or continuous) rotations remains a challenging and under-researched topic.

References

Abeysooriya, R. P., Bennell, J. A., and Martinez-Sykora, A. (2018). Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation. *International Journal of Production Economics*, 195:12–26.

Airbus (2005). A320 aircraft characteristics airport and maintenance planning. Technical report no. 03/02, AIRBUS S.A.S., Customer Services, Technical Data Support and Services, 31707 Blagnac Cedex, FRANCE. Available at `https://www.airbus.com/content/dam/corporate-topics/publications/backgrounders/techdata/aircraft_characteristics/Airbus-Commercial-Aircraft-AC-A320.pdf`. Retrieved: 27 July 2020.

Alvarez-Valdes, R., Martinez, A., and Tamarit, J. M. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, 145(2):463–477.

Art, R. C. (1966). An approach to the two dimensional irregular cutting stock problem. (Bachelor's Thesis, Massachusetts Institute of Technology, Cambridge (Massachusetts), United States). Retrieved from `https://dspace.mit.edu/bitstream/handle/1721.1/97782/25841973-MIT.pdf`.

Battiti, R. and Tecchiolli, G. (1994). The reactive tabu search. *ORSA Journal on Computing*, 6(2):107–220.

Bennell, J. A. and Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, 184(2):397–415.

Burke, E., Hellier, R., Kendall, G., and Whitwell, G. (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, 54(3):587–601.

Błażewicz, J., Hawryluk, P., and Walkowiak, R. (1993). Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, 41:313–325.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M. B., Oliveira, J. F., and Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, 253(3):570–583.

Cuninghame-Green, R. (1989). Geometry, shoemaking and the milk tray problem. *New Scientist 12 August 1989 No 1677*, pages 50–53.

Egeblad, J., Nielsen, B. K., and Odgaard, A. (2007). Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, 183(3):1249–1266.

Fowler, R. J., Paterson, M. S., and Tanimoto, S. L. (1981). Optimal packing and covering in the plane are np-complete. *Information Processing Letters*, 12:133–137.

Gomes, A. M. and Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, 141(2):359–370.

Google (2021). Google maps. `https://www.google.com/maps/place/Southern+California+Logistics+Airport/@34.5906162,-117.381907,108m/data=!3m1!1e3!4m5!3m4!1s0x80c3706663e2cc39:0xc8cb7a684dce9138!8m2!3d34.5839839!4d-117.3787064`. Retrieved: 02 June 2020.

Guépeta, J., Acuna-Agost, R., Briant, O., and J.P.Gayon (2015). Exact and heuristic approaches to the airport stand allocation problem. *European Journal of Operational Research*, 246(2):597–608.

Hillier, F. S. and Lieberman, G. J. (2015). *Introduction to Operations Research*. Mc Graw Hill, New York, NY, 10 edition.

IATA (2020). Covid-19 air travel reaching a turning point. `https://www.iata.org/en/iata-repository/publications/economic-reports/air-travel-reaching-a-turning-point/`. Retrieved: 25 June 2020.

Leao, A. A. S., Toledo, F. M. B., Oliveira, J. F., Carravilla, M. A., and Alvarez-Valdés, R. (2020). Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, 282(3):803–822.

Li, X., Wang, Z. X., Chan, F. T. S., and Chung, S. H. (2019). A genetic algorithm for optimizing space utilization in aircraft hangar shop. *International Transactions in Operations Research*, 26(5):1655–1675.

Liu, H. and He, Y. (2006). Algorithm for 2D irregular-shaped nesting problem based on the NFP algorithm and lowest-gravity-center principle. *Journal of Zhejiang University SCIENCE A*, 7(4):570–576.

Mundim, L. R., Andretta, M., and Queiroz, T. A. (2017). A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. *Expert Systems with Applications*, 81:358–371.

Oliveira, J. F., Gomes, A. M., and Ferreira, J. S. (2000). TOPOS – A new constructive algorithm for nesting problems. *OR-Spektrum*, 22:263–284.

Qin, Y., Chan, F. T. S., Chung, S. H., Qu, T., and Niu, B. (2018). Aircraft parking stand allocation problem with safety consideration for independent hangar maintenance service providers. *Computer and Operations Research*, 91:225–236.

Qin, Y., Wang, Z., Chan, F. T., Chung, S., and Qu, T. (2019). A mathematical model and algorithms for the aircraft hangar maintenance scheduling problem. *Applied Mathematical Modelling*, 67:491–509.

Ramakrishnan, K., Bennell, J. A., and Omar, M. K. (2008). Solving two dimensional layout optimization problems with irregular shapes by using metaheuristic. In *2008 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore*, pages 178–182.

REUTERS (2020). Airline industry braces for lengthy recovery from coronavirus crisis. `https://www.reuters.com/article/us-health-coronavirus-airlines/airline-industry-braces-for-prolonged-recovery-from-coronavirus-crisis-idUSKBN21K3KL`. Retrieved: 25 June 2020.

Sato, A. K., Martins, T. C., Gomes, A. M., and Tsuzuki, M. S. G. (2019). Raster penetration map applied to the irregular packing problem. *European Journal of Operational Research*, 279:657–671.

Stoyan, Y., Pankratov, A., and Romanova, T. (2016). Cutting and packing problems for irregular objects with continuous rotations: mathematical modelling and non-linear optimization. *Journal of the Operational Research Society*, 67(5):786–800.

Stoyan, Y., Terno, J., Scheithauer, G., Gil, N., and Romanova, T. (2001). Phi-functions for primary 2D-objects. *Studia Informatica Universalis*, 2(1):1–32.

Toledo, F. M. B., Carravilla, M. A., Ribeiro, C., Oliveira, J. F., and Gomes, A. M. (2013). The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics*, 145(2):478–487.

Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., and Ibaraki, T. (2009). Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research*, 16:661–683.

II

Supporting Work

1

Aircraft representation data

In order to obtain the measurements necessary for representing an aircraft in that way, airport planning documents were consulted. Such documents are issued by aircraft manufactures for each aircraft (sub)type for airport planning purposes and provide the general aircraft dimension parameters (see e.g. Airbus, 2005).

Not all dimensions necessary are given in airport planning documents; in such cases the dimensions are derived from a similar dimension given or derived from the drawings using the scale bar. In addition, the exact wing height varies depending on both the actual aircraft weight and the aircraft's centre of gravity and therefore aircraft manufactures provide a range of values. For the purpose of this thesis, a middle value is selected from those provided ranges.

The dimensions are used to generate a 2D aircraft representation for each aircraft (sub)type, described in a local x-y coordinate system. As depending on the aircraft type the amount of points used to describe an aircraft and its landing gear (2 or 4 engine aircraft, wing or tail mounted engine, landing gear configuration), the coordinates refer to the aircraft points as can be seen in their corresponding figures.

Airbus A320neo

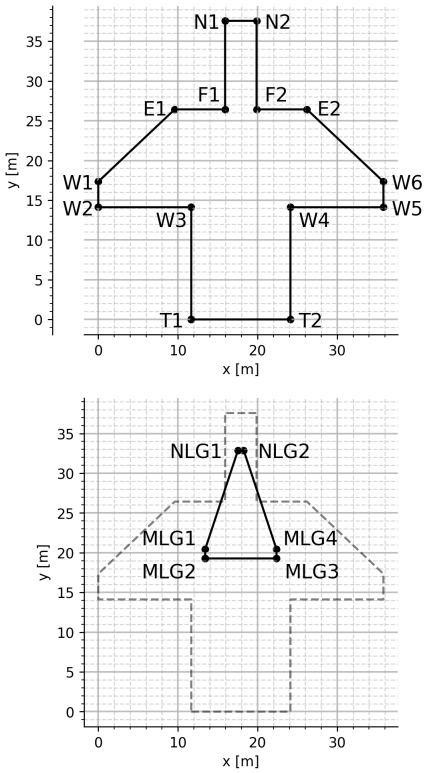


Figure 1.1: Airbus A320neo representation

Table 1.1: Airbus A320neo representation coordinates

Point	Coordinates (x,y) [m]
N1	(15.925, 37.57)
N2	(19.875, 37.57)
F1	(15.925, 26.43)
F2	(19.875, 26.43)
E1	(9.6, 26.43)
E2	(26.2, 26.43)
W1	(0.0, 17.34)
W2	(0.0, 14.12)
W3	(11.675, 14.12)
W4	(24.125, 14.12)
W5	(35.8, 14.12)
W6	(35.8, 17.34)
T1	(11.675, 0.0)
T2	(24.125, 0.0)
NLG1	(17.54, 32.881)
NLG2	(18.26, 32.881)
MLG1	(13.425, 20.444)
MLG2	(13.425, 19.276)
MLG3	(22.375, 19.276)
MLG4	(22.375, 20.444)

Airbus A350-900

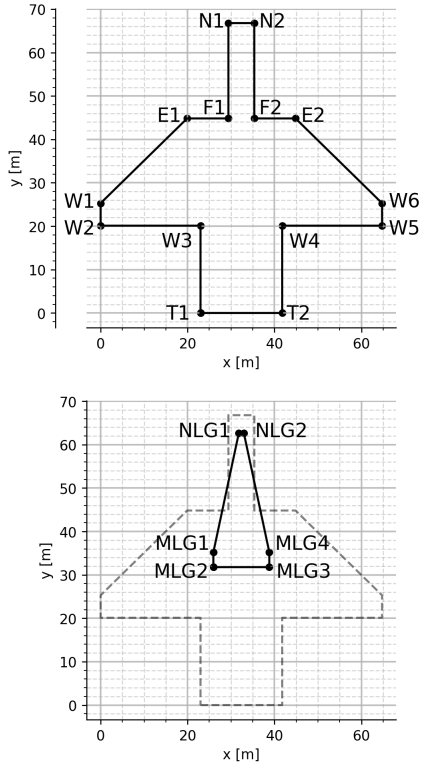


Figure 1.2: Airbus A350-900 representation

Table 1.2: Airbus A350-900 representation coordinates

Point	Coordinates (x,y) [m]
N1	(29.395, 66.8)
N2	(35.355, 66.8)
F1	(29.395, 44.83)
F2	(35.355, 44.83)
E1	(19.905, 44.83)
E2	(44.845, 44.83)
W1	(0.0, 25.3)
W2	(0.0, 20.12)
W3	(22.98, 20.12)
W4	(41.77, 20.12)
W5	(64.75, 20.12)
W6	(64.75, 25.3)
T1	(22.98, 0.0)
T2	(41.77, 0.0)
NLG1	(31.8, 62.695)
NLG2	(32.95, 62.695)
MLG1	(25.94, 35.23)
MLG2	(25.94, 31.79)
MLG3	(38.81, 31.79)
MLG4	(38.81, 35.23)

Bombardier CRJ200

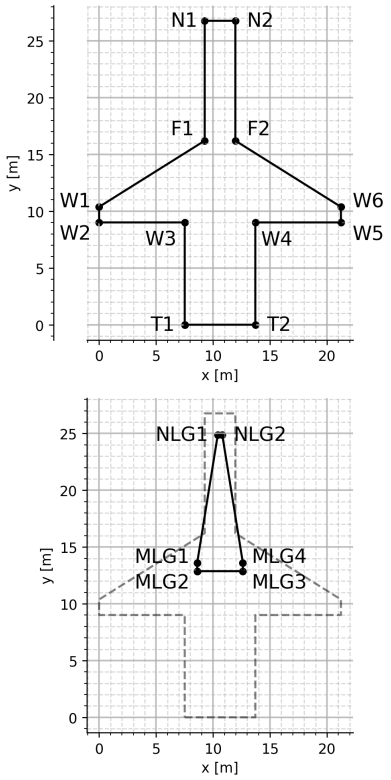


Figure 1.3: Bombardier CRJ200 representation

Table 1.3: Bombardier CRJ200 representation coordinates

Point	Coordinates (x,y) [m]
N1	(9.27, 26.77)
N2	(11.96, 26.77)
F1	(9.27, 16.21)
F2	(11.96, 16.21)
W1	(0.0, 10.38)
W2	(0.0, 9.01)
W3	(7.515, 9.01)
W4	(13.715, 9.01)
W5	(21.23, 9.01)
W6	(21.23, 10.38)
T1	(7.515, 0.0)
T2	(13.715, 0.0)
NLG1	(10.415, 24.8686)
NLG2	(10.815, 24.8686)
MLG1	(8.61, 13.6083)
MLG2	(8.61, 12.8717)
MLG3	(12.62, 12.8717)
MLG4	(12.62, 13.6083)

Airbus A380

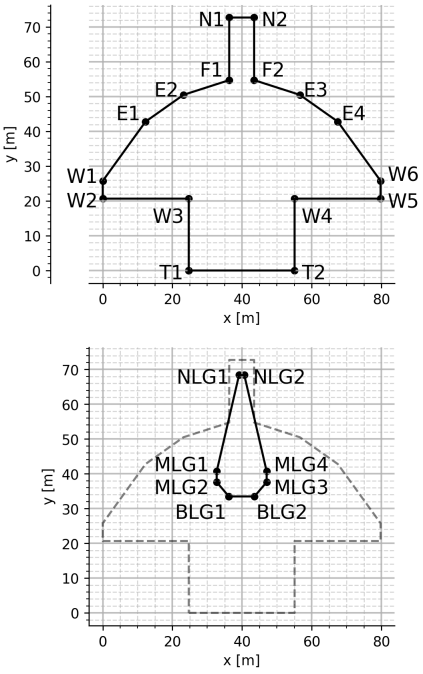


Figure 1.4: Airbus A380 representation

Table 1.4: Airbus A380 representation coordinates

Point	Coordinates (x,y) [m]
N1	(36.305, 72.73)
N2	(43.445, 72.73)
F1	(36.305, 54.73)
F2	(43.445, 54.73)
E1	(12.275, 42.79)
E2	(23.175, 50.5)
E3	(56.575, 50.5)
E4	(67.475, 42.79)
W1	(0.0, 25.76)
W2	(0.0, 20.66)
W3	(24.69, 20.66)
W4	(55.06, 20.66)
W5	(79.75, 20.66)
W6	(79.75, 25.76)
T1	(24.69, 0.0)
T2	(55.06, 0.0)
NLG1	(39.1225, 68.395)
NLG2	(40.6275, 68.395)
MLG1	(32.707, 40.7)
MLG2	(32.707, 37.6)
MLG3	(47.043, 37.6)
MLG4	(47.043, 40.7)
BLG1	(36.203, 33.48)
BLG2	(43.547, 33.48)

2

Placement Strategies

In this chapter the placement strategies explored are discussed. Initially, another placement strategy was explored than was explained in Part 1 (scientific paper) in order to decode the sequence of aircraft into a actual physical layout. For completeness this alternative method, referred to as the “translate-rotate” method, is explained here in section 2.1. Furthermore, some components of the no-fit polygon (NFP) method used in the final algorithm presented in Part 1 (scientific paper) are discussed in further detail in section 2.2

2.1. Translate-rotate method

The translate-rotate strategy is based on a direct geometry method, while using a similar sequence

The main principle is that an aircraft is introduced at some distance far away form the current partial layout at an angle relative to the last placed aircraft, is then translated towards the partial layout until (at least) one point of contact, and is subsequently rotated towards the partial layout until (at least) two points of contact with the aircraft already placed. The angle at which an aircraft is placed relative to the previous aircraft is referred to as the placement angle and is shown in Figure 2.1. The placement angle is obtained from the tabu search and therefore the length of the bitstring increases by the amount of bits representing an angle times the amount of aircraft - 1, since the first aircraft will be fixed to the corner of the placement area. This placement process is illustrated in Figure 2.2. The rotation direction is determined by the moment resulting from applying an imaginary force from the centroid of the aircraft to be placed in the direction of the placement angle + $180\,\mathrm{deg}$ (Figure 2.2b).

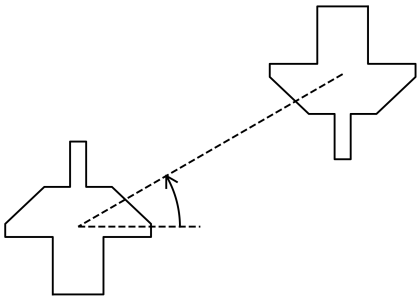


Figure 2.1: Placement angle

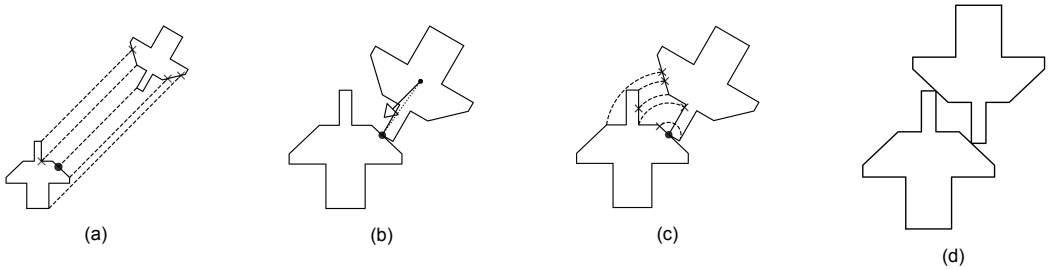


Figure 2.2: (a) translate step, (b) determine rotate direction, (c) rotate step, (d) final position of the aircraft.

The layouts obtained through this method of decoding the bitstring are not guaranteed to be within the placement area. Therefore, soft constraints were introduced by assigning penalties proportional to the distance the final parking layout is exceeding the boundary of the placement area.

The geometric calculations are handled by the Python library *shapely*. The amount of calculations necessary increases with the amount of aircraft placed and the amount of points used to represent the aircraft's geometry. In addition, during the tabu search itself more solutions must be evaluated each iteration due to the increased amount of bits needed for representing the placement angles.

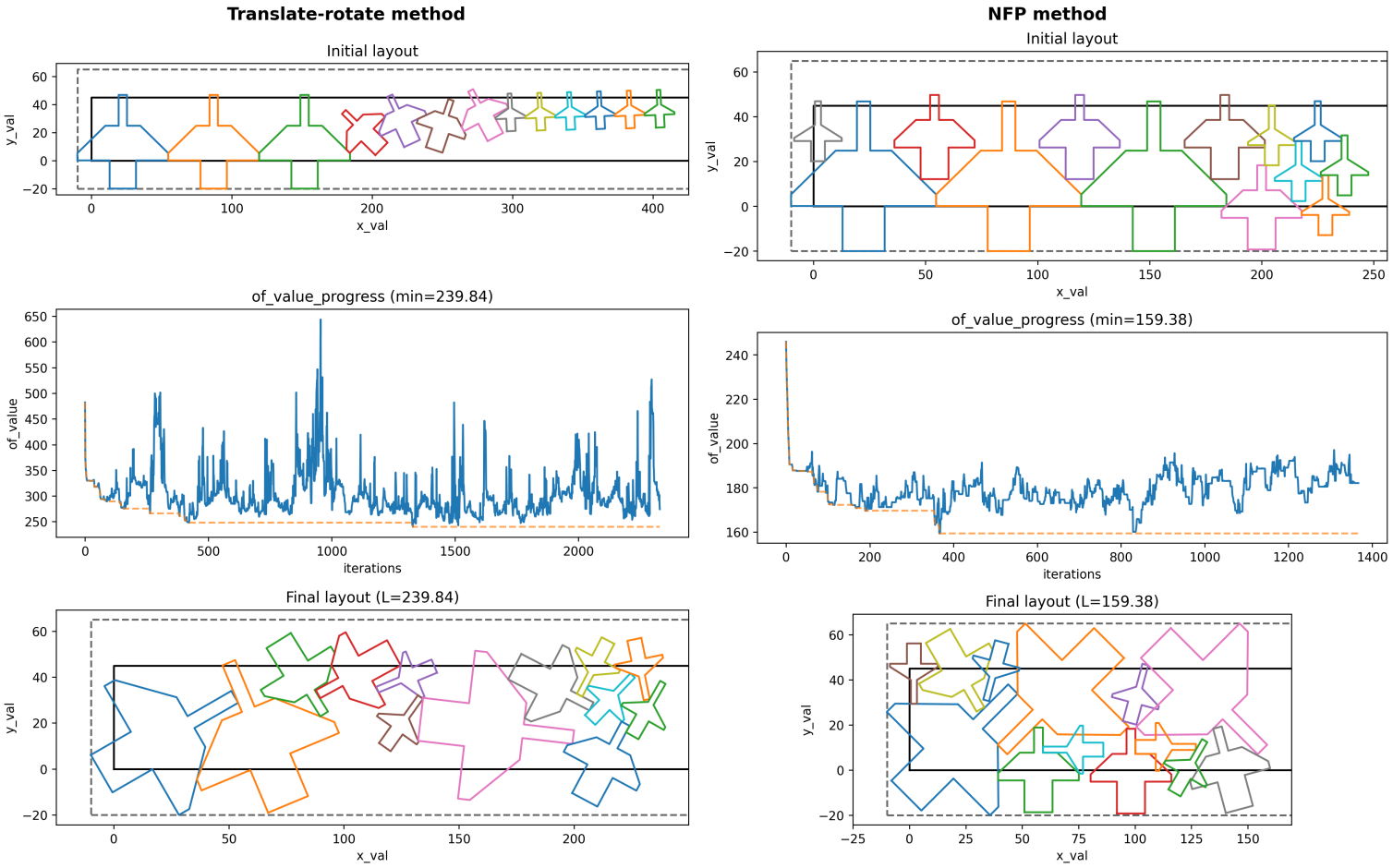
In contrast to the NFP method, the geometric calculations cannot be performed in a pre-processing phase and must be done online during the tabu search. In order to decrease the amount of geometric calculations, several optimisations were implemented. During the translation step, only aircraft directly along the translation path are considered in order to calculate the maximum translation distance in order to place the aircraft as close as possible to the current partial layout. Similarly, in the rotate step only aircraft in the immediate neighbourhood of the rotating aircraft are considered for calculating the maximum rotation before touching a neighbouring aircraft. Furthermore, partial and complete layouts are stored and can be retrieved later in the search; the placement algorithm can then start from that partial solution instead of decoding the sequence (and the corresponding geometric calculations) from scratch.

Initial experiments showed this method to be slower and struggling to converge to a tight parking layout. An example is shown in Figure 2.3 and corresponding key algorithm performance metrics are shown in Table 2.1. The initial bitstring for both methods is the same (zero bitstring), however, due to the difference in placement algorithm the resulting initial layout is different for each method. All other settings are the same, the tabu list length was set at 62 and the search is stopped after 1000 non-improving iterations.

Even without model parameter tuning, the NFP method is able to generate a tighter layout, in less iterations and less time. There are several explanations as to why this is the case. The longer average iteration time for the translate-rotate method is due to a combination of 1) the geometric calculations necessary for placing aircraft and 2) the placement angle bits simply introducing more solutions to evaluate each iteration. In addition, the placement angle bits introduces more complexity in the search and might lead to the tabu search unable to find regions in the solutions space with good solutions. Furthermore, the soft constraints used for the boundaries of the placement area introduce humps in the solution space making it more difficult for the tabu search to navigate the solution space.

Table 2.1: Translate-rotate (TR) vs. NFP method comparison results

Result	**TR**	**NFP**
Layout length [m]	239.84	159.38
Average iteration time [s]	7.55	0.60
Total iterations	2330	1367
Total optimisation time	04h:53m:02s	00h:13m:43s



(a) Translate-rotate method

(b) NFP method

Figure 2.3: Placement algorithm comparison

2.2. NFP method

As was mentioned in Part 1 (scientific paper), the NFP is constructed by using the Cuninghame-Green (1989) method by decomposing the aircraft in three convex sections. In this section the NFP generation and recombination step is explained in more detail.

While the method generates the NFP for two objects, it does not give the position of the NFP in relation to the fixed polygon. The NFP constructed using this method adds the edges (translated to the origin) of the two polygons in their ascending slope order, but the starting vertex is arbitrary. Hence, the NFP must be translated in order to obtain a meaningful result. As explained by Bennell and olive 2008, when the bottom left corner of the enclosing rectangles of both polygons are taken as reference point, the reference point of the NFP is found by placing the orbiting polygon at the bottom left corner of the enclosing rectangle of the NFP. Then, the reference point of the NFP is the top right corner of the enclosing rectangle of that translated orbiting polygon. The (reference point of) the NFP is subsequently translated to the origin and the NFP is stored at that position. During the optimisation stage, the NFP can then by retrieved and

An example of this process is shown in Figure 2.4. Consider fixed polygon A and orbiting polygon B, where their bottom left corners of their enclosing rectangles are taken as reference points: $(3, 3)$ and $(1.5, 2)$ respectively (Figure 2.4a). The NFP construction algorithm, presented in Part 1 (scientific paper), results in NFP_{AB} where the bottom left corner of its enclosing rectangle is found at $(0, -2)$. (Figure 2.4b). Translating the orbiting polygon B from $(1.5, 2)$ to the NFP's bottom left corner $(0, -2)$, the reference point of NFP_{AB} is found at the top right corner of the (translated) orbiting polygon B at $(1, -1)$ marked by the cross in Figure 2.4c. During the optimisation stage, the reference point of the NFP is translated to match the reference point of the fixed polygon A (Figure 2.4d). The boundary of the NFP then represents the actual possible positions where (the reference point of) the orbiting polygon B can be placed with respect to fixed polygon A in order for the two polygons to touch.

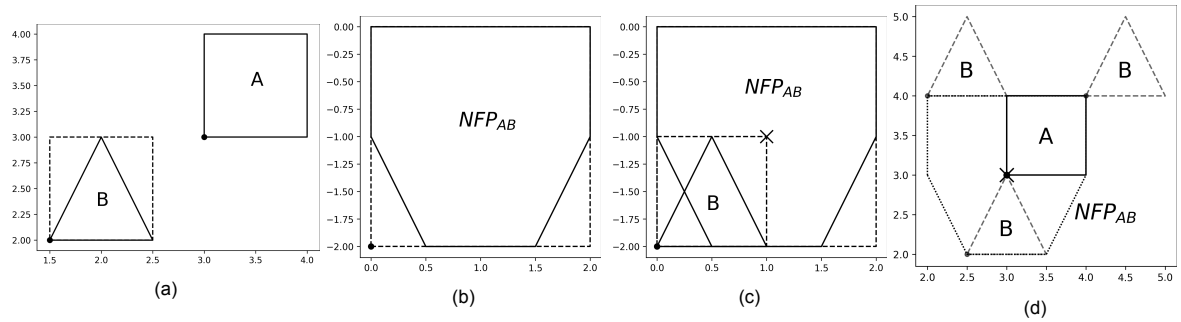


Figure 2.4: (a) fixed polygon A and orbiting polygon B and their respective enclosing rectangles, (b) the resulting NFP_{AB} and its enclosing rectangle, (c) translating orbiting polygon B to match the bottom-left corner of its enclosing rectangle with that of the NFP to find the reference point of the NFP, (d) the NFP translated to match the fixed polygon A.

III

Literature study (AE4020)

3

Geometric representations and tools

The first obstacle in the aircraft parking problem, and more generally the cutting and packing problem, is related to the geometry: how does one determine if two shapes are overlapping? This question becomes much more complex when irregular shapes, such as representations of aircraft, are involved (Leao et al., 2020). Bennell and Oliveira (2008) noted that this geometric problem is not a trivial task and provided a tutorial for the geometry of the nesting problem. They found that the four most common modelling approaches applied to the nesting problem are the raster method, direct trigonometry, the no-fit polygon (NFP), and the phi function. Each one of these approaches will be discussed in the next sections, respectively.

3.1. Raster method

When a raster representation is used, the continuous placement area is represented by a discrete grid. An example in its most basic form can be seen in Figure 3.1. Here, the piece is represented by 1s in the grid, and empty cells are assigned 0. Checking overlap between pieces then simply becomes a matter of adding the values for each cell in the grid; whenever the cell's value is greater than one, overlap is present in that cell. An advantage of the raster approach is that it is easy to implement and no complex calculations are required. A disadvantage is that pieces cannot be represented precisely (as is clear from Figure 3.1). Though accuracy can somewhat be improved by increasing the grid resolution, it comes at the expense of an increased computational time (Bennell and Oliveira, 2008).

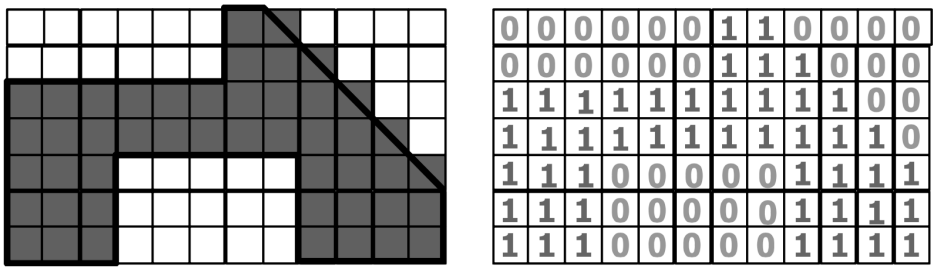


Figure 3.1: Example of raster representation (Bennell and Oliveira, 2008).

3.2. Direct trigonometry

Direct trigonometry methods can be used to more accurately represent the pieces compared to the raster method. Though the methods are more precise, the run times for feasibility checks will be slower (Bennell and Oliveira, 2008). Leao et al. (2020) identified three direct trigonometry approaches, namely the D-function, circle covering method, and the separation lines approach.

The D-function originally proposed by Konopasek (1981), where the pieces are represented as polygons, has been a popular direct trigonometry tool applied to the nesting problem (Leao et al., 2020). The method is based on the distance from a point to a straight line and efficiently determines

the relative position between two edges of different pieces and can be used to check if the two edges intersect. Ferreira et al. (1998) improved the method by first determining whether the bounding boxes of pieces intersect, and if true subsequently identifies whether overlap is present between the bounding boxes of the edges of those pieces. The principle is shown in Figure 3.2. The procedure showed a significant reduction in the amount of direct edge intersection tests between two polygons (Bennell and Oliveira, 2009).

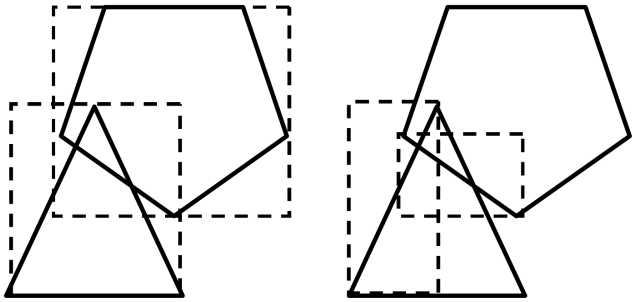


Figure 3.2: Bounding box overlap for complete polygon (left) and individual edges (right) (Bennell and Oliveira, 2008).

Rocha et al. (2014) proposed a circle covering approach to represent a piece by a set of circles covering the piece in a nonlinear model. The main idea behind this approach is that operations with circles are simple, and therefore the approach allows for straightforward overlap calculations and it can take into account continuous rotations of pieces. They compared three types of circle covering representations. In the inner covering representation all the circles lie within the piece (Figure 3.3a). In the complete circle covering representation, the circles cover the piece entirely (Figure 3.3b). In the partial circle covering, the circles do not completely contain the piece, but also do not lie entirely within in the piece.

A benefit of the circle covering approach is that continuous rotations of pieces can easily be considered, whereas usually in literature the orientation is fixed or only a discrete set of orientations is allowed. A drawback of the approach is that the pieces are not precisely represented, and a balance between the amount of circles representing the piece and the precision must be found. As a result, the imprecision could lead to infeasible solutions when the inner circle covering representation is used, and in complete circle covering representations it could result in gaps between the actual pieces in the final layout (Rocha et al., 2014).

(a) Inner circle representation

(b) Complete circle representation

Figure 3.3: Circle covering representation (Leao et al., 2020).

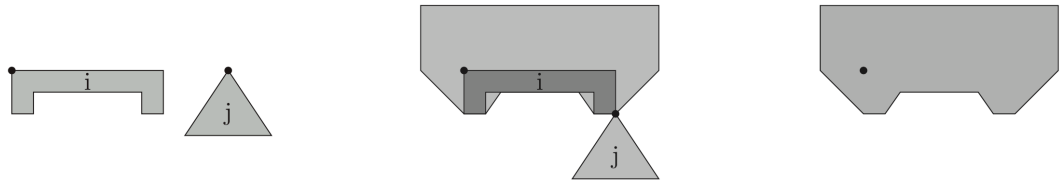
Recently Peralta et al. (2018) used the concept of separation lines for the packing of irregular pieces problem in their nonlinear model. Separation lines are used to ensure that two polygons do not overlap and a straight line is a separation line when all vertices of one polygon are on or on one side of that line, and all vertices of the other polygon are on or on the other side of that line. Separation lines can rotate and translate, as long as the aforementioned definition remains valid. Non-convex polygons are decomposed into a set of convex polygons, and each polygon in the convex set has its own separation line with other polygons not belonging the set.

The benefit of the method is that it allows for continuous rotations and in contrast to the circle covering method proposed by Rocha et al. (2014) discussed earlier, the polygons are used directly and therefore does not suffer from the imprecision caused by the circle covering approach. The results were found to be slightly worse than the approach by Stoyan et al. (2016), who used the phi function (described in section 3.4) to determine overlap and they too allowed for continuous rotation of pieces.

3.3. No-fit polygon

The concept of the no-fit polygon (NFP) was first introduced by Art (1966) and has become a widely used tool to handle the geometry of nesting problems. Since the NFP is constructed from the original edges of the polygons, it represents the pieces accurately while the method is more efficient than the direct trigonometry approach, especially benefiting iterative search methods for finding solutions to the nesting problem (Bennell and Song, 2008).

The boundary of the NFP can be viewed as the relative position of the two polygons that represents the two pieces touching. Any point within the NFP represents overlap, and any point outside the NFP represents that the pieces do not overlap (or touch). An example of the NFP is shown in Figure 3.4. Here, piece i is fixed and piece j slides around piece i in such a way that piece j always touches piece i, and its reference point (top of the triangle piece j) is traced during the procedure. The resulting polygon NFP_{ij} is shown in Figure 3.4c. The inner-fit polygon (IFP) is a concept related to the NFP. Here, the piece slides within (instead of around) the fixed polygon and remains enclosed by it. The IFP is used to verify that a piece remains within the physical boundaries of the problem (e.g. aircraft placements do not exceed the allocated taxiway dimensions) (Bennell and Oliveira, 2009).

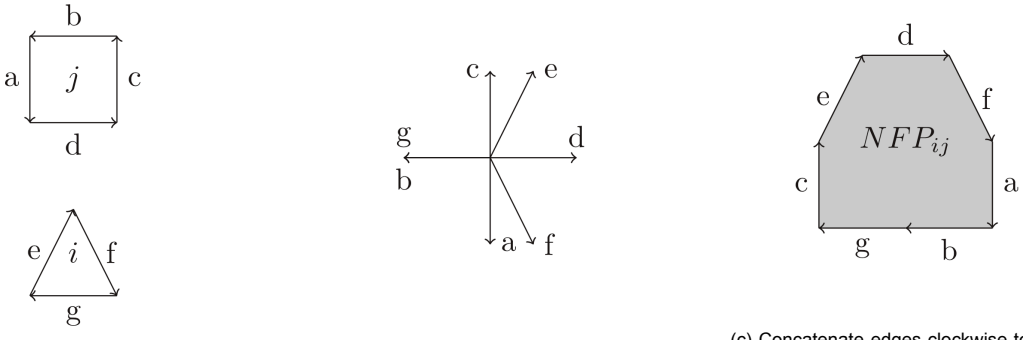


(a) Pieces i and j (b) NFP construction (c) NFP_{ij}

Figure 3.4: No-fit polygon example (Leao et al., 2020).

The benefit of the NFP approach is that it reduces the overlap test to a simpler point in polygon test and the NFPs for all pairs of pieces can be calculated in a pre-processing phase, both reducing the computational cost in the optimisation phase. However, for each orientation of a piece a new NFP must be calculated and therefore only a discrete set of orientations can be used, not allowing for continuous rotations. In addition, NFP generation for non-convex shapes is not a straightforward task (Bennell and Oliveira, 2009).

The method proposed by Cuninghame-Green (1989) generates a NFP for the simplest case of two convex polygons. It provides a simple, easy to understand algorithm illustrating the NFP generation and the principle is shown in Figure 3.5. First, the edges of polygons i and j are oriented in a clockwise and counterclockwise manner respectively, where i is the fixed polygon and j is moving polygon (Figure 3.5a). Next, the edges are translated to start at a single point (Figure 3.5b). In the final step the translated edges from Figure 3.5b are linked together in a clockwise manner, resulting in the NFP_{ij} (Figure 3.5c).



(a) Pieces i and j clockwise and counterclockwise oriented respectively

(b) Edge translation to single point

(c) Concatenate edges clockwise to create NFP_{ij}

Figure 3.5: Cuninghame-Green (1989) method for no-fit polygon generation (Cherri et al., 2016).

As mentioned earlier, constructing an algorithm for NFP generation for non-convex polygons, such

as aircraft representations, is not trivial. Bennell and Oliveira (2008) identified three main approaches, namely the sliding approach, the method using Minkowski sums, and the decomposition of polygons into simpler convex polygons.

The sliding approach, also known as the orbital approach, was first introduced by Mahadevan (1984). The method is based on physically sliding the orbiting piece around the fixed polygon by finding the feasible translation vector. Each translation creates an edge of the NFP and the process is repeated until the starting point is reached again. However, the approach cannot deal with special cases such as concavities within a polygon. More recently, Burke et al. (2007) improved the method to correctly handle degenerate cases (e.g. holes, concavities, exact fits), by finding additional starting points along the edges of the polygon, from where the sliding approach is used once again to generate the remaining NFP sections.

The second approach is based on a form of vector addition, namely the Minkowski sums. Bennell et al. (2001) and Bennell and Song (2008) provide a detailed method to obtain the NFP using Minkowski sums. The Minkowski sum is defined as follows in Equation 3.1, where A and B are two sets of points:

$$A \oplus B = \{a + b : a \in A, b \in B\} \tag{3.1}$$

It can be shown that the Minkowski difference, defined as $A \oplus -B$, is equal to the NFP of A and B (NFP_{AB}) (Bennell et al., 2001). This is equivalent to orienting the edges of the pieces in opposing directions (Figure 3.5a), as is done in the Cuninghame-Green (1989) method discussed earlier. In fact, the Cuninghame-Green (1989) method is identical to the Minkowski sum in its simplest form where both polygons are convex. The detailed approach for constructing the NFP using Minkowski sums by Bennell and Song (2008) improves the work by Bennell et al. (2001), by properly handling degenerate cases, and provides a robust procedure for generating the NFP for irregular and non-convex polygons using Minkowski sums.

The third method is to decompose the non-convex polygons into simpler convex polygons for which the NFP can be generated easily by using for example the Cuninghame-Green (1989) approach discussed earlier. Bennell and Oliveira (2008) noted however that decomposing and recombining the several sub NFPs into one NFP is a complex process. Although an optimal set of sub NFPs reduces the Minkowski sum computation times, the computation times for generating an optimal set of sub NFPs are costly and do not justify the obtained benefit, hence heuristic methods to create a set of (sub optimal) sub NFPs are preferred. Furthermore, some degenerate cases cannot be handled and additional direct tests must be applied to find those cases.

3.4. Phi-function

Stoyan et al. (2001) introduced the concept of phi-functions, which are used to define the relation between two objects. If the value of the phi-functions is smaller than zero, the two objects overlap, if the value is greater than zero, the two objects do not overlap. If the value is exactly zero, the two objects touch and the phi-function is equal to the boundary of the NFP. When the phi-function is normalised, the Euclidean distance between the objects is equal to the value of the phi-function. Stoyan et al. (2001) derived the phi-functions for so called primary objects, which are circles, rectangles, regular polygons, and convex polygons. More complex objects can be represented by combining primary objects, using their unions and/or intersections.

A simple example with two circles is shown in Figure 3.6, where r_1 is the radius of circle 1 (positioned at (x_1, y_1)) and r_2 the radius of circle 2 (positioned at (x_2, y_2)). In order for the circles not to overlap, the distance between their centres must be at least the sum of their radii. Hence, the equation describing all the positions of circle 2 where it would be touching circle 1 is given by $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} = r_1 + r_2$. As mentioned earlier, when the objects touch the value of the phi-function is zero. Hence the phi-function, when $\phi(x_1, y_1; x_2, y_2) = 0$, is then defined as (Equation 3.2):

$$\phi(x_1, y_1; x_2, y_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} - (r_1 + r_2) \tag{3.2}$$

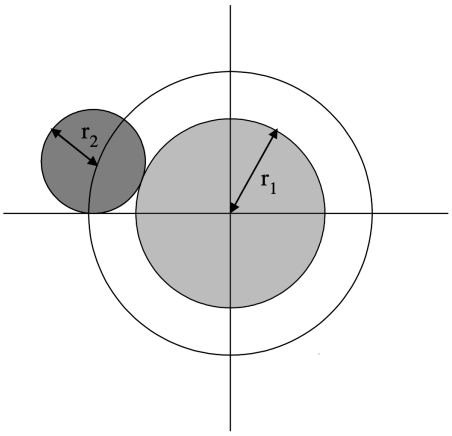


Figure 3.6: Two circles touching (Bennell and Oliveira, 2008).

The example in Figure 3.6 with two circles is the simplest case, where the phi-function can be defined as a single equation. Other objects require multiple functions, and the dominating function is determined by whether the function is either the maximum or minimum, depending on the shape and position. For example, in Figure 3.7, consider the case where $\phi(x, y) = 0$ (i.e. the NFP), then four phi-functions must be defined, one for each edge. If the orbiting square is placed e.g. on the bottom edge of the NFP, and thus $\phi(x, y) = 0$ is for the equation describing the bottom edge of the NFP. Consequently, the equations describing the other three edges will then result in a negative value ($\phi(x, y) < 0$) for those edges. Therefore, selecting the maximum of the four functions will give the dominating function, correctly describing the NFP.

When the orbiting square is moved further away, additional functions must be defined, taking into account the curved corners for instances where $\phi(x, y) > 0$. The reader is referred to Stoyan et al. (2001) for the derivation of such instances, and their paper also includes phi-function derivations of other primary objects.

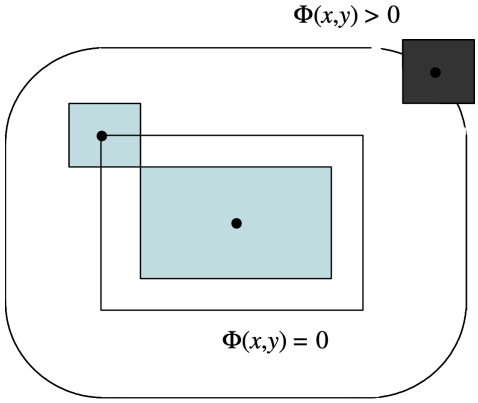


Figure 3.7: Two cases of phi-functions (Bennell and Oliveira, 2008).

A benefit of the of the phi function method is that continuous rotations can be considered such as presented by Stoyan et al. (2016). A drawback of the phi-function method is that there is no algorithmic approach to obtain functions for arbitrary shapes and the functions must be derived by hand. Bennell and Oliveira (2008) suggest that that might be the reason why the phi-functions have not been widely adopted by researchers, despite it being a potentially powerful tool. Indeed, a recent review of nesting problems by Leao et al. (2020) showed that phi-function approach is still mostly used in work (co-)authored by Stoyan.

4

Metaheuristics

Some problems can be so complex that it might not be realistically possible to solve for an optimal solution. When exact methods are not able to solve a problem to an optimal solution (within a reasonable amount of time), heuristics are commonly applied to find a good feasible solution, approximating an optimal solution. A heuristic algorithm is based on common sense ideas tailored to a specific problem and is usually an iterative process, searching for a new, possibly better, solution each iteration. Such improvement procedures can, however, get stuck in a local optimum when a problem has multiple local optima. Metaheuristic are general solution methods that have the ability to escape local optima, and thereby directing the search towards the global optimal solution. In this section the main principles of the most common metaheuristics according to Hillier and Lieberman (2015) are briefly discussed, namely the tabu search in section 4.1, simulated annealing (section 4.2), and the genetic algorithm in section 4.3.

4.1. Tabu search

A tabu search algorithm uses a local search procedure to find a local optimal solution, iteratively improving the solution during the search. When a local optimum is reached, the search then continues by allowing moves which do not improve the solution in order to escape the local optimum. To avoid circling back to the same local optimum, a tabu list temporarily stores moves which would otherwise lead back to the same local optimum. Moves on the tabu list are not allowed, unless the move would result in a better feasible solution than the best solution found in any of the previous iterations. Longer term memory is used in diversification strategies, which force the search process into a new unseen part of the solution space, and intensification strategies, which search a particular part of the solution space in more detail (Hillier and Lieberman, 2015).

4.2. Simulated annealing

Using the analogy of hill climbing, the tabu search climbs to the top of a hill and then descends in search for another hill to climb, whereas the simulated annealing approach tries to find the tallest hill by initially moving in random directions. This is done by selecting one of the neighbouring solutions randomly, and comparing it to the current solution. If the randomly selected solution is better or equal, it will always become the new solution in the next iteration. If it is worse, than the solution is only accepted according to the following probability (in case the objective is maximisation), where Z_c is the objective function value of the current solution, and Z_n is the objective function value of the potential next solution under consideration (Equation 4.1, Hillier and Lieberman (2015)):

$$Prob\{acceptance\} = e^x \text{ where } x = \frac{Z_n - Z_c}{T} \tag{4.1}$$

Slight downward moves therefore have higher probabilities of becoming the next solution, while moves producing solutions which are much worse have lower probabilities of becoming the next solution. Obviously, the probability also depends on the value of T. At the start of the search, the value of T is (relatively) high, resulting in higher acceptance probabilities of downward moves. Throughout the

search the value of T is lowered, decreasing the probability of accepting a downwards move and consequently mostly upwards moves will be accepted in the later stages of the search process. Hence the name simulated annealing, which is based on the physical annealing process, where a melted metal with a high temperature is slowly cooled during the process. The values of T throughout the search must be chosen carefully for a successful implementation of the algorithm (Hillier and Lieberman, 2015).

4.3. Genetic algorithm

The genetic algorithm is based on the analogy with the evolutionary principle of the survival of the fittest in nature. Instead of improving one solution each iteration, a set of solutions (the population) is considered each iteration. The fitness is determined to select the best solutions, and the fittest members (i.e. best solutions) have better chances to become parents, which are paired randomly. Children are generated, i.e. new solutions which have features of both parents. During the process of generating children, mutations can occur so that children obtain features that do not belong to any two of the parents. Mutations allow for diversification of the search into unexplored areas of the solution space, possibly generating better solutions. If a generated child with potential mutations results in an infeasible solution, a new solution is created until a feasible solution is obtained. The principle of survival of the fittest results in improving populations each iteration, and tends to lead to a near optimal solution (Hillier and Lieberman, 2015).

5

Hangar aircraft placement optimisation

A problem related to the long term parking problem is that of the aircraft placement in a maintenance hangar. By optimising the parking positions of aircraft within the hangar, a greater number of aircraft could potentially be serviced simultaneously, benefiting the maintenance service providers. Although the problem is different in certain aspects (which will become clear at the end of this section), it is similar in that a certain number of aircraft have to be parked within a limited amount of space as efficiently as possible.

The hangar aircraft parking problem in existing literature can essentially be seen as a variation of the cutting and packing or nesting problem, discussed in chapter 6. Because of their relevance to and similarity with the long term parking problem, the papers incorporating the hangar aircraft parking problem are discussed here separately in detail.

The existing literature on this topic is rather limited, and the papers on this topic will therefore be discussed in chronological order in this chapter. Qin et al. (2018) proposed a two-stage MIP strategy for the hangar parking problem, minimising unused hangar space in the first stage and maximising the safety spacing between aircraft in the second stage. Their work is discussed in section 5.1. In section 5.2 the quasi 3D parking arrangement of Qin et al. (2019) is discussed briefly, who incorporated the hangar layout planning problem into the maintenance scheduling problem of aircraft. Li et al. (2019) developed a genetic algorithm for optimising space utilisation in an irregular hangar which is discussed in section 5.3. Finally, some differences with regard to the long term parking problem are highlighted in section 5.4.

5.1. A two-stage MIP approach (Qin et al., 2018)

Qin et al. (2018) studied the aircraft hangar parking problem from the perspective of an independent maintenance service provider. In contrast to a maintenance facility owned and used by an airline itself, where usually fixed parking stands are designed because of the limited amount of aircraft types within the airline, an independent maintenance provider may receive maintenance requests from different customers with different aircraft types and sizes, requiring a flexible solution of parking the aircraft in the hangar. In this particular problem it is assumed that the aircraft selected for maintenance are rolled into the hangar at the same time as a batch and maintenance starts when all aircraft are in their respective position. During maintenance, the aircraft are not moved and are rolled out only when maintenance of the complete batch is finished. Furthermore, aircraft are moved in and out of the hangar in a straight line and no rotations are allowed; all aircraft face in the same forward direction. The hangar space considered is a rectangular area which can be fully utilised for aircraft placement, there is no space reserved for other facilities (e.g. offices, equipment storage, maintenance vehicles, etc). As noted by the authors, although the scheduling of maintenance has been a topic of research in the literature for many years, no research was available for the optimisation of hangar space utilisation.

Methodology

The aircraft are modelled as non-convex polygons, roughly following their 2D outline in the horizontal plane. In order to detect overlap between two aircraft, the no-fit polygon (NFP) approach is used to find

the region where an aircraft cannot be placed. More precisely, the Minkowski sum method (Bennell et al., 2001; Bennell and Song, 2008) was applied to construct the NFP between the two non-convex polygons representing the aircraft.

In order to incorporate the NFP into the MIP model, a horizontal partitioning approach is adopted as proposed by Alvarez-Valdes et al. (2013). For each edge of the NFP, two horizontal lines, one from each vertex, are drawn outwards away from the NFP, representing the horizontal slice. Each horizontal slice is associated with a binary decision variable, taking a value of 1 when the reference point of the aircraft to be positioned is placed within that particular slice and 0 otherwise. The horizontal slice is a polygon and defined by a set of linear inequalities, ensuring that the exact position of the reference point of the aircraft to be positioned is placed within that particular slice. This principle can be seen in Figure 5.1 for the NFP between two aircraft.

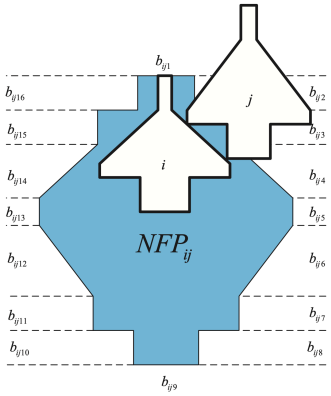


Figure 5.1: Concept of horizon slices for the NFP between two aircraft (Qin et al., 2018).

A two-stage MIP model was proposed. In the first stage the objective is to maximise overall profit by selecting the aircraft maintenance requests which can be parked within the hangar without overlapping each other. Under the assumption that larger aircraft (measured by their area) yield higher profits, the problem is equivalent to minimising the unused space and the latter is used as objective function for the MIP. However, the solution from the first stage might result in aircraft being placed closed to each other, even while part of the hangar remains empty and all non-overlapping constraints are satisfied, as can be seen in the left side of Figure 5.2.

In order to spread the aircraft more evenly across the hangar (right side of Figure 5.2), in the second stage, the aircraft selection found from the first stage is taken and the safety margins between the aircraft are maximised according to a specified lower and upper bound for each aircraft type. A higher risk of collision is associated with larger aircraft as they are less manoeuvrable, and hence larger margins are assigned to larger aircraft. The margin is incorporated into the model by enlarging the NFP outwards by the safety margin, creating a buffer area around the aircraft. Imposing the lower bound of the safety margin in the second stage might result in an infeasible solution and implies that not all aircraft from the first stage can be placed inside the hangar. In that case, the first stage is rerun but with revised NFPs, including the lower bound safety margin, resulting in a feasible solution satisfying the minimal (lower bound) safety margin.

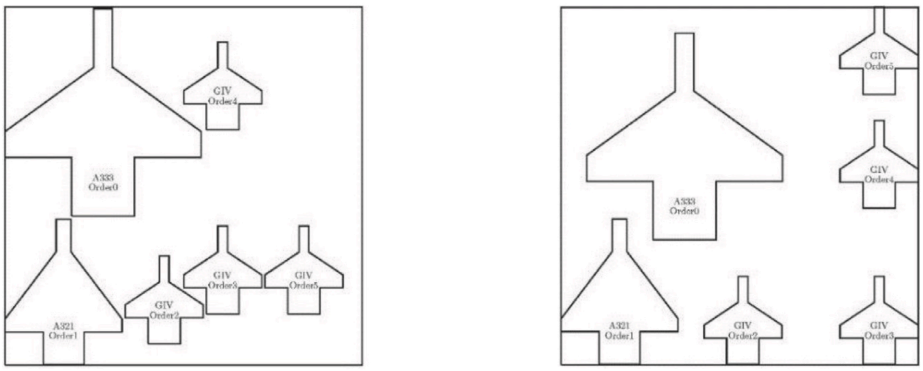


Figure 5.2: Layout of first stage (left) and layout after second stage (right) (Qin et al., 2018).

In the second stage the MIP is adapted to maximise the safety margin between aircraft. For improved efficiency, a heuristic algorithm is used to tighten the upper bound and the principle of the algorithm is as follows. First, the safety margin of all aircraft are augmented until no feasible solution can be produced anymore. Then the next step is to decrease the safety margin of individual aircraft, starting with the aircraft having the lowest area. This process is repeated until a feasible solution is found. If the lower bound of the safety margin of an individual aircraft is reached, then the next smallest aircraft is taken. Subsequently, when a feasible solution is found, the safety margin of the aircraft with the largest area is increased. If this results in an infeasible solution, the aircraft is removed from future safety margin augmentation consideration and the next largest aircraft is selected.

Results

40 instances were tested with different combinations of aircraft types and the MIPs were solved using CPLEX. Of the 40 instances, 6 could not be solved optimally in the first stage. All of these instances included larger number of aircraft and after inspection of the best known solution for each instance it was found that the hangar space was almost fully utilised and provided a satisfactory solution. It was concluded that because of the non convex shape of the aircraft, the partitioning of the NFP between two aircraft into horizontal slices results in many binary variables. Therefore, visiting all relative positions of aircraft before updating the bounds and thus the optimality gap is a difficult process.

In the second stage the number of binary variables depends on the number of aircraft from stage one and the range of the safety margin. Since this leads to an even greater amount of binary variables, this problem becomes much more difficult than stage one and large optimality gaps are observed in certain instances, although the solutions given are considered satisfactory from a practical perspective, as was the case in the first stage. Nevertheless, in many instances the solution from the initial heuristic solution was demonstrated by the exact branch and bound algorithm (CPLEX) to be the optimal solution. Though in certain cases a better solution was found by the branch and bound algorithm, the computational time was much longer in comparison to the heuristic solution. In addition, the heuristic solution showed significantly improved layout plans regarding safety margins compared to manual planning. Therefore, it was concluded by the authors that the heuristic solution gives a good solution for all practical purposes, balancing computational time and solution quality.

5.2. 3D parking arrangement (Qin et al., 2019)

Qin et al. (2019) incorporated the hangar layout problem into the scheduling of the maintenance, generating a series of parking plans at different times which are aligned with the maintenance schedule planning horizon. Although their research encompasses more than the hangar parking problem, for the purpose of this literature study, the focus is on the hangar parking layout problem part only.

The research by Qin et al. (2019) with regard to the parking layout is an extension of their earlier work (Qin et al., 2018). With the aim of more efficiently using the available space, the wing of a smaller aircraft could be placed under the wing of a larger aircraft. In order to incorporate this idea into the models, when the wing height difference between the two aircraft allows placement under the wing, the NFP for the aircraft is split into two parts; the main body of the aircraft, and the wings. The main aircraft

body NFP ensures separation between two aircraft bodies, while the wing NFP allows for overlap within a certain safety margin. This concept is illustrated in Figure 5.3.

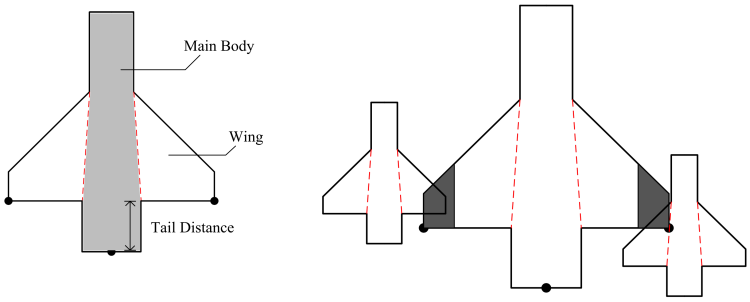


Figure 5.3: 3D parking problem (Qin et al., 2019).

5.3. Genetic algorithm for optimising space utilisation (Li et al., 2019)

Li et al. (2019) proposed a genetic algorithm to solve the (2D) hangar parking problem. Similar to Qin et al. (2018), it was assumed that aircraft are rolled in (and out) in a straight line and aircraft of different types and sizes are considered. Differences are that Li et al. (2019) allowed the aircraft to be placed tail-in or head-in, and an irregularly shaped hangar was considered instead of a rectangular hangar area (Figure 5.4). A safety margin of one metre was used between aircraft and the hangar walls.

Methodology

The aircraft are represented by non-convex polygons as can be seen in Figure 5.4, which is similar to Qin et al. (2018), and the hangar space is represented as a 2D Cartesian coordinate system (origin in the left bottom corner). The reference point for placement is taken as the intersection of the centre line of the aircraft and the edge of the polygon representing the tail. In order to place the aircraft, a set of main references lines are drawn as shown in Figure 5.4 indicated by the square brackets. Each main set consists of a further amount of equidistant lines, which coincide with the x-value in the Cartesian coordinate system (the dotted lines around and including reference line [6] in Figure 5.4).

Each chromosome consists of the same amount of parts as there are main sets of reference lines, where each part of the chromosome corresponds directly to that set. In each part also the position of the reference line (i.e. the x-value), its orientation, and the aircraft type is recorded. For example, in Figure 5.4, for the L900LX aircraft, it will be in part 6 of the chromosome (i.e. the set of references lines indexed [6]) and is placed on the 4th reference line within that set. It is directed upward (i.e. tail-in) and the type is a F900LX aircraft.

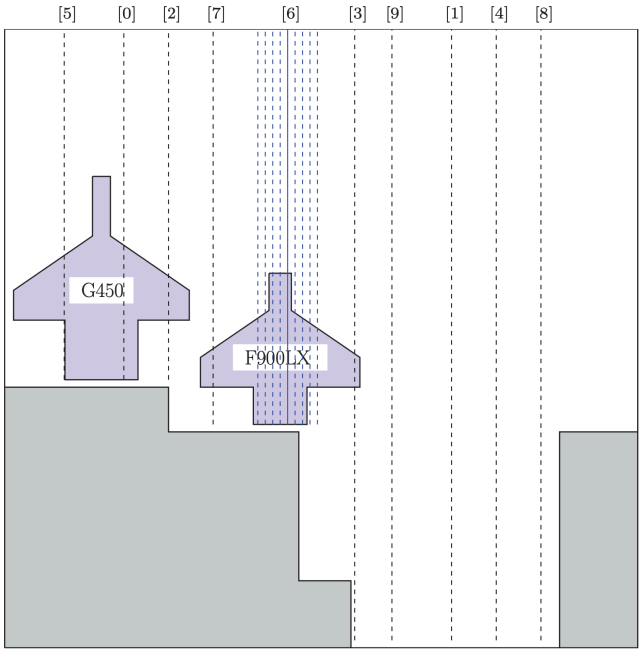


Figure 5.4: Reference lines for aircraft placement (Li et al., 2019).

In order to place an aircraft, the algorithm starts by placing the aircraft at the lowest y-value possible. If conflicts occur, the aircraft is moved upwards along the chosen reference line until no conflicts are present. This process is shown in Figure 5.5. If the aircraft cannot be placed along the reference line, the chromosome is deemed not fit and a new chromosome is generated.

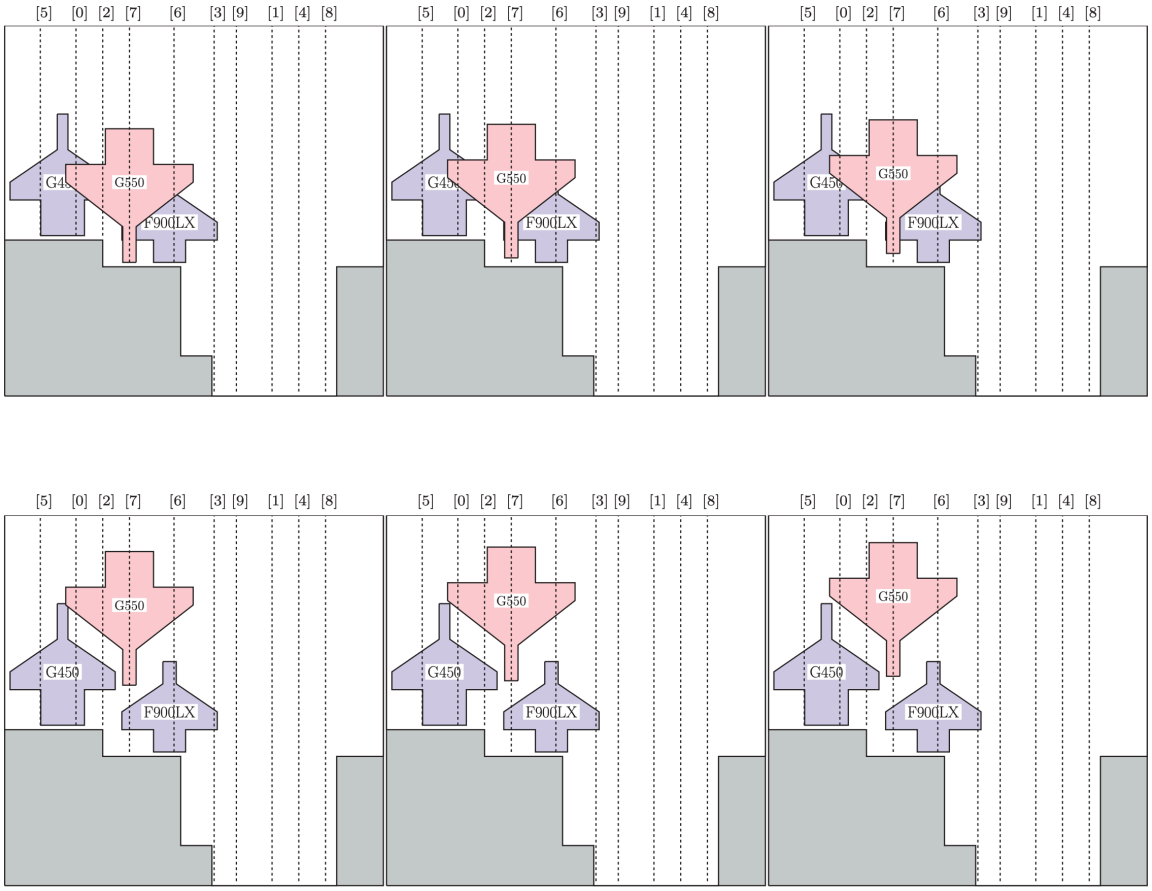


Figure 5.5: Vertical positioning along the reference line (Li et al., 2019).

To create an initial set of solutions, chromosomes are generated randomly. At initialisation, only a small number of aircraft are placed in each chromosome and therefore feasible (fit) solutions are easily obtained. A crossover operation randomly pairs two chromosomes and the purpose is to place an aircraft from one chromosome to the same position in the other chromosome in order to obtain new chromosomes, provided that that position in the other chromosome is empty and the aircraft is not already placed in that chromosome.

Three mutation operators, which are executed sequentially, are defined: swapping, moving line, and turning head. First, a swapping operation randomly selects an occupied line set and is swapped with another randomly select line set. The moving line operator then randomly selects an occupied line set, and the occupied reference line within that set is moved to another (empty) reference line within that line set. Finally, in the turning head operator an aircraft already placed is randomly selected and its orientation is changed from tail-in to head-in or vice versa. Furthermore, after each mutation operation, an insertion operation is executed attempting to place the aircraft not yet placed on one of the empty reference lines.

In order to select the better chromosomes for the next generation, a roulette wheel approach is used. In such an approach, better chromosomes have a higher chance proportional to their value of being selected.

Results

The experiment was run 10 times with an average computational time of 50 minutes and a standard deviation of 12 minutes. For the case studied, it was found that by using the proposed algorithm one additional aircraft could be park when compared to manual planning.

However, the research does not incorporate other hangar layouts and does not discuss the placing of the reference lines. The placement of the (set of) references lines could possibly influence the outcome of the solution. In addition, each line can only be used once, therefore placing (the reference

point of) two aircraft on the same line immediately after each other is not possible; an offset will always be present. Furthermore, using such reference lines effectively discretises the solution space and potential feasible solutions are therefore missed. Finally, aircraft are limited to two orientations (i.e. head-in or tail-on), allowing more rotation angles could possibly result in better solutions.

5.4. Differences with the long term parking problem

The hangar aircraft parking problem is the problem closest related to that of the long term parking problem. As noted by Qin et al. (2018) and Li et al. (2019), the literature on the topic is rather limited but the hangar parking problem could be a starting point for the long term parking problem. However, some clear differences between the hangar parking problem and the long term parking problem can be identified.

First, aircraft are rolled in and out of the hangar in straight lines due to manoeuvrability constraints within the hangar, leaving only two orientations possible (i.e. tail-in or head-in). In contrast, for the long term parking problem, the aircraft are placed in an open space such as a runway or taxiway where such restrictions are not present and therefore any orientation is possible. A second difference is that aircraft placed on a runway are not confined to the dimensions of the runway itself, as for example the wing or tail could be allowed to overhang the grass next to the runway. Furthermore, the aircraft do not allow for any overlap in the hangar problem researched by Qin et al. (2018) and Li et al. (2019). As certain maintenance tasks perhaps do not allow for such overlap, the layout with wing overhang proposed by Qin et al. (2019) could be used more aggressively in the long term parking problem. In addition, several maintenance tasks must be performed regularly during the long term parking to keep the aircraft in good condition, such as idle engine runs, tire rotation, and operational system checks (Ackert, 2010). Finally, the problem size for long term parking is expected to be larger because more aircraft can be parked on an outside surface than within a hangar and therefore also more types of aircraft could be present compared to the hangar problem, possibly influencing the results. These additional requirements should be taken into account when positioning aircraft for long term parking.

6

Cutting and packing of irregular shapes problems

In this section the existing approaches for solving cutting and problems are discussed. Since little research exists on the optimisation of aircraft parking positions outside the context of using fixed airport infrastructure, commonly known as the gate or stand allocation problem, see e.g. Guépeta et al. (2015), approaches for solving cutting and packing problems could be a source of inspiration for the optimisation of the long term parking of aircraft. More specifically, problems that involve irregular shapes (such as aircraft) are considered, which are also known as nesting problems. Irregular problems are much harder to solve than problems where e.g. only rectangles are considered, due to the geometric complexities it involves (Leao et al., 2020). Allowing the shapes to rotate adds to this complexity even further. The problem is known to be NP-hard (Błażewicz et al., 1993; Stoyan et al., 2016) which results in the solution approaches mainly using heuristic methods in literature, though some exact approaches have been researched. The main focus in this section is on 2D nesting problems problems, since for the purpose of long term parking of aircraft, the aircraft are parked on the same surface and then the aircraft's projected 2D shape can be used. Some 3D elements could be incorporated, such as wing overhang, as was already discussed in section 5.2. There are several variants of the nesting problem, the most common ones discussed in this section are: packing items into a single container with fixed dimensions, packing items into a rectangular piece with fixed width and infinite length (strip packing), or bin packing where items are divided and packed into several containers. The focus in this section is not on either of the variants, rather the main interest is to review the typical packing strategies used.

First, exact approaches that have been proposed in literature will discussed in section 6.1. Next, in section 6.2, heuristic approaches will be discussed. The section is concluded in section 6.3 with a short summary of the main approaches discussed.

6.1. Exact methods

Although some exact methods have been proposed, such approaches can only realistically solve to optimality for small instances, and with no or only a very limited amount of rotations. For example, Alvarez-Valdes et al. (2013) proposed a branch and bound algorithm and used a horizontal partitioning approach dividing the NFP into horizontal slices in order to incorporate the NFP into their MIP model. No rotations were considered, and only the instances with 10 pieces could all be solved to optimality within one hour. Some, but not all, instances up to 16 pieces could be solved to optimality when the time limit was increased to ten hours. They concluded that the complexity of the problem increases with the amount of pieces, and the amount of edges each piece has (i.e. the complexity of the piece itself). Some of the complexity can be eliminated by introducing the Dotted-Board Model (Toledo et al., 2013), which discretises the continuous placement area into a grid of points, and the corresponding MIP is solved using a commercial solver. The discretisation allowed them to solve problem instances up to 56 pieces to optimality (in the discrete solution space for a certain discretisation step), however no rotations were considered. In order to solve instances with more pieces or where rotations are allowed using the dotted board model, pieces can be pairwise clustered by joining them into a single piece

(Sato et al., 2018). Although clustering will not improve the quality of the solution, computations will be faster. Cherri et al. (2016) proposed a MIP model where the pieces are decomposed and represented as convex polygons, allowing for easy NFP generation, and the D-function is used to impose the non-overlapping constraints between a piece and the corresponding NFP. Discrete rotations can be included by reproducing the piece with the corresponding amount rotations and adding additional constraints so that only one of the reproduced pieces is used in the layout of the model. Two orientations were considered for instances up to 12 pieces, and although better results were found when rotations were allowed, it comes at the expensive of increased computational time and it is not possible to solve to optimality for larger problems.

Several non-linear programming (NLP) approaches were proposed (Peralta et al., 2018; Rocha et al., 2014; Stoyan et al., 2016), allowing for continuous rotations in the mathematical model (this in contrast to MIP formulations, which only allow discrete rotations). However, Peralta et al. (2018) and Rocha et al. (2014) their solution methods using existing solvers only solve to local optima, and Stoyan et al. (2016) stated that it is not realistic to solve the NLP problem to optimality and proposed an algorithm to find good local optimal solutions.

Cherri et al. (2018) proposed a novel mixed integer quadratically constrained programming model with continuous rotations and used off-the-shelf global optimisation solvers to find the optimal solution. However, most instances with 4 or more pieces could not be solved to proven optimality as either the computational time limit of 7 hours was reached or the computer system with 16 gigabytes of memory ran out of memory.

Cherri et al. (2016) concluded that approaches using mathematical models are only able to solve small instances with no or very limited rotations to proven optimality. The lower bounds were found to be of poor quality, and therefore feasible solutions generated during the search are not better than the solutions (meta)heuristic approaches can achieve in the same amount of time.

It is therefore concluded that exact methods are not suitable for the purpose of long term aircraft parking optimisation, as aircraft can be placed at any arbitrary orientation and instances with larger amounts of aircraft should be considered than the amount of pieces that were considered in the exact methods described above. Hence, the remainder of this section will be focused on heuristic methods to solve the irregular cutting and packing problem.

6.2. Heuristic methods

Improvement heuristics improve complete solutions by e.g. moving and swapping pieces, and thereby creating neighbouring solutions. As was mentioned in chapter 4, heuristics tend to converge to a local optimum and hence are combined with a higher-level metaheuristic in order to escape local optima. Bennell and Oliveira (2009) divided the improvement heuristics into two main categories, based on their representation. In the first representation the layout is a sequence of pieces, which is decoded into the final layout through a set of placement rules, also known as a constructive algorithm or placement heuristic. The search for a new solution is therefore over the sequence in which the pieces are placed. The second approach is working with the physical layout directly, and moving/swapping/rotating pieces within the actual layout itself instead of the sequence. In this approach overlap is usually permitted in order the search the solution space, and the overlap is then penalised in the objective function. In both cases a constructive algorithm is used; in the sequence representation the sequence is decoded through the constructive algorithm as a subroutine to generate and evaluate the actual layout, while with the physical layout a constructive algorithm is used to generate an initial feasible solution.

In this section the same division between improvements heuristics will be used. First, common constructive algorithms will be discussed in subsection 6.2.1. This is then followed by solution approaches searching over the sequence, and approaches searching over the physical layout in subsection 6.2.2 and 6.2.3, respectively.

6.2.1. Constructive algorithms

Two main aspects must be considered when developing a constructive heuristic (also known as placement heuristic): the placement rules must be established (i.e. where to place the next piece) and the initial placement sequence of the pieces must be determined (i.e. which piece to place next). A popular placement rule is the bottom left placement rule (Bennell and Oliveira, 2009). An intuitive example is shown in Figure 6.1. Here the piece is iteratively moved horizontally as much as possible, then

continues down vertically until its position allows it to move horizontally again. When the piece is not able to move left or down anymore, the procedure is stopped. The concept was first proposed by Art (1966) for the nesting problem, and the principle is still used in more recent solution methods. For example, Abeysooriya et al. (2018), Burke et al. (2006), Gomes and Oliveira (2002), and Mundim et al. (2017) all proposed solution methods which searched over the sequence and decoded their generated sequences through a bottom left algorithm, and Egeblad et al. (2007) and Elkeran (2013) used the bottom left principle to create an initial solution which is then subsequently improved by searching over the physical layout directly.

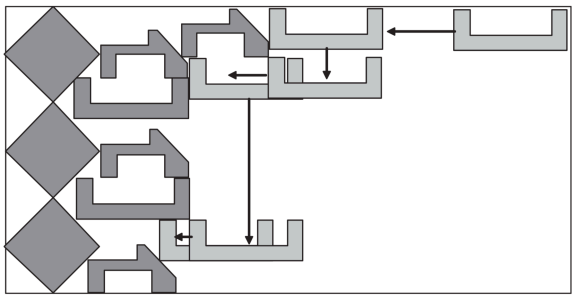


Figure 6.1: Bottom left placement strategy principle (Bennell and Oliveira, 2009).

The exact implementation of the bottom left placement rule is dependent on the geometric representation used. If a discrete raster representation is used for both the piece and the placement area, the piece is moved horizontally from raster point to raster point until an unfeasible location is reached. The piece is then returned to the previous feasible location and moves one grid unit downwards, if possible. The algorithm then repeats and tries moving horizontally again. If both options would result in an infeasible solution, i.e. overlap is detected or the piece moves outside the boundaries of the placement area, the piece remains at its current grid position and becomes its final position. When the pieces are represented by polygons and the placement area is continuous, the infinite amount of feasible placement points must be reduced to a discrete, finite set. Several methods have been proposed to achieve this reduction. For example, Mundim et al. (2017) superimposed a grid onto the placement area and moved the pieces stepwise along the grid, similar to the raster representation but the pieces remain represented as polygons. Burke et al. (2006) proposed a method where the x-axis is discretised similar to the previous approaches, but the y-axis is continuous, therefore producing more compact layouts. Geometric functions are then used in order to determine the vertical displacement needed for a feasible placement along the continuous y-axis. The concept of the NFP was used by Gomes and Oliveira (2002) to determine a feasible placement set. The (reference point of the) piece that is placed next must be outside all NFPs and within the IFP (inner-fit polygon), which represents the feasible placement region. By selecting the left most (lowest) x-coordinate of that region, and then, given that left most x-coordinate, selecting the bottom most (lowest) y-coordinate, the placement position is found. The authors observed that this procedure would always result in placing the piece on a vertex of the feasible region; namely, a vertex of a NFP, a vertex of the IFP, the intersection of two edges of two NFPs, or the intersection of a NFP edge and an IFP edge.

Holes might be present in the layout of the pieces already placed, and a pure bottom left algorithm will not be able to fill these holes since the pieces are introduced from the right (relative to Figure 6.1). Burke et al. (2006) therefore proposed the new bottom left fill algorithm, which starts searching from the left side, through the infeasible regions first. In this manner holes will be encountered first and fill up first before reaching the empty right side of the placement area. The constructive algorithm proposed by Abeysooriya et al. (2018) overcomes the hole filling issue by first comparing the area of the hole to the area of the piece to be placed next. If the hole has a smaller area than the piece, the piece will certainly not fit in the hole. Otherwise, if the area of the piece is smaller than the hole, the piece could potentially fit and a procedure is started to generate an IFP of the hole and the piece. The piece is then placed at the most bottom left position of this IFP. If the IFP does not exist, the piece will not fit and then the algorithm will continue with the regular bottom left rule. The bottom left rule based on the NFP method by Gomes and Oliveira (2002) discussed earlier automatically takes into account holes, since the holes will be part of the feasible placement set.

Oliveira et al. (2000) introduced a new constructive algorithm named TOPOS and the core principle is illustrated in Figure 6.2. Pieces are added to the layout one by one, growing the partial solution with each piece placement. The pieces already placed (i.e. the partial solution) are (is) represented by the external contour of the pieces and the contour can be treated as one piece. Hence, any enclosed gaps cannot be used anymore for future placement. The origin is not fixed and hence pieces can be added at any side of the partial solution, as long the generated new solution does not exceed the maximum dimensions of the actual placement area. The concept of the NFP is used to determine the feasible placement points when placing the next piece. Three placement rules were proposed for adding a piece to the partial layout: 1) minimising the area of the rectangular enclosure of the generated new partial solution; 2) minimising the length of the rectangular enclosure of the generated new partial solution; or 3) maximising the overlap between the rectangular enclosures of the current partial solution and the piece to be placed, while the pieces themselves do not overlap. The TOPOS algorithm was revised by Bennell and Song (2010) to allow for filling holes by merging the NFPs of the original individual polygons rather than representing the layout by its external contour and generating the NFP for the external contour. Abeysooriya et al. (2018) used another variant of TOPOS by fixing the origin and closing any gaps in the partial layout smaller than the smallest still to be placed piece.

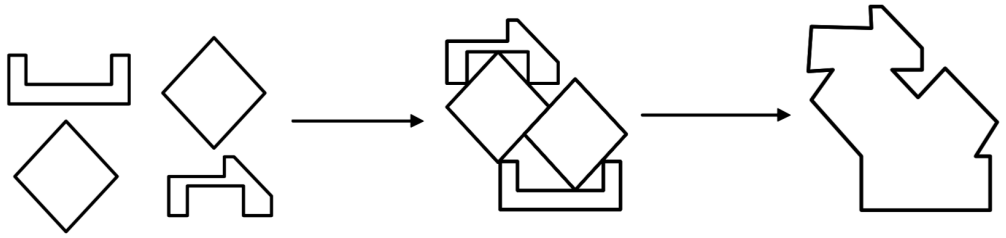


Figure 6.2: TOPOS principle. The pieces to be placed (left) are placed according to the placement rules into a layout (middle) and then pieces are merged and represented by their external contour (right) (Bennell and Song, 2010).

A novel placement principle based on the lowest-gravity-centre principle was introduced by Liu and He (2006). First, the gravity centre of the piece to be placed next is calculated and is used as the piece's reference point for constructing the IFP. The lowest point on the IFP will than become the placement point with the lowest-gravity-centre since the gravity centre is used as reference point for the IFP. The placed piece is subtracted from the placement area in order to generate an artificial new boundary of the placement area. Therefore only the IFP has to be calculated when placing a piece. This way, the method aims at pushing the piece to be placed as close as possible to the pieces already placed. In order to reduce the amount of holes, when a piece is placed the biggest hole between the piece and the boundary is tested to fit other pieces. If another piece fits, that piece must be placed first. A somewhat similar approach was proposed by Liu and Ye (2011), who introduced the HAPE algorithm. It aims to minimise the total potential energy by keeping the centre of gravity as low as possible. However, the method avoids the use of the NFP by discretising the placement area by superimposing a grid. The piece is moved over all the grid points and at each grid point its gravity centre is calculated provided there is no infeasibility (i.e. the piece does not overlap with other pieces and remains within the boundary of the placement area). The grid point where the piece's gravity centre is the lowest is chosen as the placement point for the piece. Because calculating NFPs is a resource intensive task, the HAPE is much faster than traditional NFP methods. However, the increase in computation times comes at the expensive of solution quality as the solution space is discretised.

Besides the placement rules, the second aspect of a constructive algorithm is the placement sequence. For both solution approaches an initial placement sequence must be determined to generate an initial solution. Gomes and Oliveira (2002) proposed several sorting criteria, namely: random order, decreasing area, decreasing length, decreasing width, decreasing irregularity, increasing rectangularity. Irregularity is measured as the difference between the area of the piece and the area of its convex hull, and rectangularity is measured as the difference between the area of the piece and the area of its enclosing rectangle. Except random sorting, all sorting criteria aim to place larger or more complicated piece first, since its easier to place them first and the smaller pieces can then be used to fill any holes present in the layout. The choice of initial sorting method can influence the final layout, and the best initial sorting is dependant on the problem's data set. (Gomes and Oliveira, 2002). Hence, the

initial sorting strategy for a specific data set should be chosen based on experimental analysis. When the constructive algorithm is used as a subroutine, decoding the placement sequence in approaches searching over the sequence, the subsequent sequences generated during the search are used.

A constructive algorithm is capable of quickly generating a feasible solution. However, in order to achieve higher quality solutions, an improvement heuristic must be used, by either searching over the sequence of placement (subsection 6.2.2), or searching over the actual physical layout (subsection 6.2.3). The choice for either option is not trivial, since researchers continue to present competitive solutions of good quality and improve results using benchmark data sets for both approaches. There does not seem to be a consensus in literature as to which method is preferred. For example, Bennell and Dowsland (2001) stated that the method allowing overlap in the physical layout is more efficient than searching over the sequence, while Ramakrishnan et al. (2008) concluded that searching over the sequence has several key benefits. Ramakrishnan et al. (2008) argued that researchers might be improving algorithms to suit the benchmark data sets specifically, adding problem specific features into the algorithm, which makes determining which solution approach performs best difficult. They therefore ran a set of computational experiments comparing the core principles of both solutions approaches, without any problem specific features, and used the metaheuristic tabu search to guide the search over the solution space in both cases. It was found that searching over the sequence has several benefits compared to searching over the physical layout. Good solutions were found in short time spans consistently and due to the constructive algorithm decoding the sequence a feasible solution was guaranteed each iteration. In contrast, approaches searching over the layout cannot guarantee a feasible solution as overlap is allowed when moving between solutions and the algorithm can potentially get stuck resolving the overlap. Nevertheless, excellent results have been achieved by approaches searching over the physical layout (Egeblad et al., 2007; Sato et al., 2019) and are therefore worth considering. It must also be noted though that Ramakrishnan et al. (2008) only considered 2 admissible orientations for the pieces. Allowing more rotations, as required for the aircraft parking problem, could possibly influence the results as for each rotation the sequence must be decoded again for solution evaluation, which is a computational expensive process (Bennell and Oliveira, 2009) and therefore potentially leading to greater run times.

6.2.2. Searching over the sequence

When the problem is represented as a sequence of pieces which are then decoded through a constructive algorithm to arrive at the actual layout, the search for finding better solutions is over the sequence (such an algorithm is sometimes also known as an iterative constructive heuristic). A key advantage of this approach is that feasibility is guaranteed by the placement rules of the constructive algorithm.

Common strategies to move from one solution to a neighbouring solution during the search are swap moves, insert moves, and orientation changes (Bennell and Oliveira, 2009). A swap moves swaps the position of two pieces in the sequence, an insert move moves the piece from its current position to another position in the sequence, and an orientation change changes the piece's orientation at its present position. The three moves are illustrated in Figure 6.3, where the bottom left algorithm is used to decode the sequence into a layout. Here it can be clearly seen that each moves results in a vastly different layout. Generally, an insert move tends to result in bigger changes to the layout than a swap moves because all pieces after the removed piece will now have changed position in the sequence. Solution approaches can use a combination of these moves during the search, or simply only move can be applied.

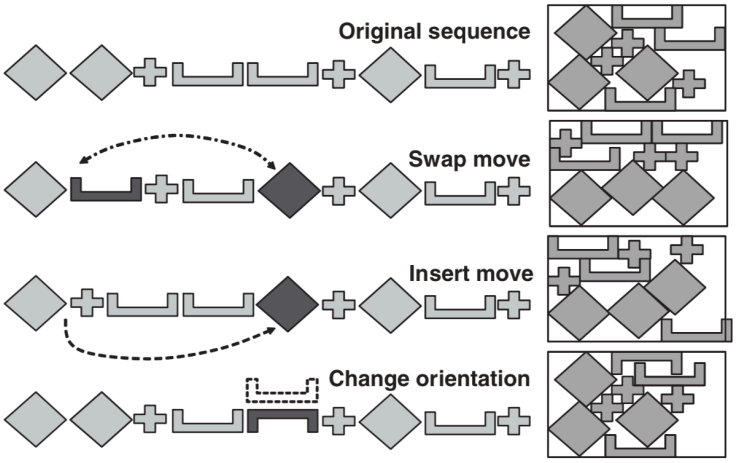


Figure 6.3: Neighbourhood moves and their respective resulting layouts using the bottom left algorithm (Bennell and Oliveira, 2009).

Because of the amount of pieces to be placed in a typical nesting problem, the size of the neighbourhood must be limited. This can for example by done by selecting one piece and include all possible moves as the neighbourhood, or include all pieces but restrict the amount of allowed moves (Bennell and Oliveira, 2009).

Gomes and Oliveira (2002) proposed an algorithm which exchanges two pieces in the sequence (i.e. a swap move) and restricted the neighbourhood size by setting a maximum swap distance between the pieces. A bottom left algorithm is used to decode the sequence into a layout and the predefined admissible orientations (0 or 180 degrees) are tested during the placement to find the most bottom left placement of the piece. Several swap distances were evaluated, ranging from one to three, and three strategies were considered to select the next solution. The first strategy is to select the first solution that is better than the current solution, with the purpose of changing solutions quickly. The second is to select the best solution (within the restricted neighbourhood) by an exhaustive search of the neighbourhood. The third proposed strategy randomly selects a solution among all better solutions in the neighbourhood, each with equal probability. The algorithm is stopped when no better solution is found and thus has reached a local optimum. Each combination of swap distance and solution selection strategy results in a total of nine algorithm variants, and the best results were found with the probabilistic solution selection strategy in combination with a maximum swap distance of 3. In four of the five benchmark data sets tested, their work produced the best known solution published at that time.

Burke et al. (2006) proposed a tabu search based method and restricts the neighbourhood to five solutions. One of the following five operators are used to generate a neighbouring solution: an insert move, swap move, swapping three pieces, swapping four pieces, and swapping a random number of pieces. In each case the pieces to be swapped are chosen randomly, however the pieces cannot be of the same type as otherwise the resulting layout would be the same. In the case of the insert move the piece and its new location are chosen randomly. The operator to generate a neighbouring solution is chosen randomly with a bias towards the less disruptive operators (i.e. an insert move has a higher chance of being selected while the operator swapping a random number of pieces has a lower chance of being selected). The best solution within the neighbourhood is chosen and the tabu list avoids visiting recent solutions. The tabu list length was set at 200 after initial experimentation, and the bottom-left-fill algorithm, which uses a discrete x-axis but allows for continuous vertical placement, was applied to decode the sequence into a layout. The orientation of a piece is determined by the placement algorithm, choosing the orientation resulting in the most bottom left piece placement for all allowed orientations. Out of the 26 data sets tested, the authors produced the best known results for 25 of them with the proposed algorithm.

In the computational experiments by Ramakrishnan et al. (2008) a tabu search strategy was adopted as well. Five different neighbourhood types were tested. The first type swaps each piece in the sequence with each of the five next items in the sequence. The second type is similar to first type,

however, only each of the first five pieces (instead of the whole sequence) are swapped with each of the next five items. The third follows the same principle as the second, however now the first 10 pieces of the sequence are swapped, and the maximum swap distance is 10, instead of 5. The fourth type is similar to first type, but restricts its size by randomly sampling 10% of the possible moves. The fifth type is similar to the fourth, however the swap distance here is increased to 10. Hence, the second type results in the smallest neighbourhood size, and the first type in the largest. Experiments showed that the fifth type consistently produced good results. The bottom left algorithm is used to decode the sequence into a layout. Several tabu list lengths were considered ranging from 5 to 100, though it was concluded the best tabu list length is problem dependant. Using this approach, solutions were close to or matched the best known results of the data sets tested and it was concluded that the tabu search is an excellent search method for the nesting problem.

Several approaches are based on the genetic algorithm. Jakobs (1996) proposed a genetic algorithm, where the chromosome directly represents the sequence. A bottom left strategy was used to translate the sequence into a layout which physically slides the pieces into their most bottom left position. The fitness function is defined as the contiguous remainder, i.e. the area where no items have been placed yet but excluding holes. Solutions are selected to become parents with a probability proportional to their fitness value. A random place in the sequence of one parent is chosen and a random number of pieces starting from that place are copied to be the beginning of the new sequence. The remainder of the pieces are then added to the back of the new solution in the same order as in the other parent. A mutation operator rotates the piece 90 degrees, which has been set at a low probability of being selected. The approach is based on rectangular shapes, though an algorithm variant is proposed taking into account irregular shapes by using the pieces' their respective bounding rectangular boxes and by applying a shrinking algorithm. When the bottom left algorithm based on the no-fit polygon is used, the bounding box restriction for irregular shapes can be removed (Junior et al., 2013).

The genetic algorithm was combined with the lowest gravity centre principle placement algorithm by Liu and He (2006). The chromosome represents the placement sequence, and the initial population is generated by sorting the pieces by decreasing area and creating several mutations thereof. The fitness function is defined as the remaining space in the container. Children are generated by randomly selecting part of the sequence of the first parent, and the remainder of the sequence is filled with the remaining pieces based on their positions in the second parent. A mutation operator with a low probability is added to randomly swap two pieces in the sequence. The orientation in the layout is determined by the placement algorithm, without a predefined set of fixed orientations to try. Because each orientation requires a new NFP, first several orientations with larger, fixed intervals are evaluated. Then, the rotation resulting in the lowest gravity centre and its two neighbouring orientations are divided into smaller fixed steps to obtain a more precise orientation. Although not fully continuous, this approach allows many more orientations to be explored than the traditional fixed set of orientations. By allowing this many orientations, better results can be obtained than the best known results that used a fixed, small number of orientations.

Mundim et al. (2017) proposed a newer variant of the genetic algorithm for the nesting problem, namely the biased random-key genetic algorithm. In the algorithm the sequence of pieces is represented by a sequence of random-keys in the [0,1) interval. One of the parents is always chosen from an elite group of members (i.e. the most fit), and there is a bias towards selecting characteristics from the fitter parent during the crossover operation. The resulting child is sorted on the random keys in increasing order, which then becomes the placement sequence for the pieces corresponding to the keys. The dotted-board model (discrete raster) representation was adopted and the sequence is decoded into a layout using the bottom left principle where overlap is avoided using the no-fit polygon concept. The algorithm was able to match or outperform (in terms of solution quality) the exact model proposed by Toledo et al. (2013) and the bottom left fill algorithm by Burke et al. (2006). Although the authors mention that the algorithm can be adapted to include piece rotations, it is not clear how it affects the solution or computational time.

Abeysooriya et al. (2018) proposed a new solution approach for the bin packing problem based on the jostle algorithm. The principle is based on the idea of shaking a container with pieces, a process where pieces naturally tend to pack closely together. This is implemented by iteratively alternating packing the pieces from left to right and then right to left, where the sequence of the pieces to be packed next is sorted on the pieces' current x-positions. The concept is illustrated in Figure 6.4. A variant of the TOPOS algorithm is used as placement algorithm, where the origin is fixed and holes within the

partial remain available for placement of subsequent pieces. For problems that allow pieces to rotate at a fixed set of angles, the NFP is simply generated for each orientation and the best is selected. When arbitrary rotations are considered, the piece is first placed similar to the discrete orientation approach to obtain a first orientation. Then the piece is rotated based on the angles the edges of the polygons make with each other at the touching point, a procedure which aims to align the edges of the two polygons.

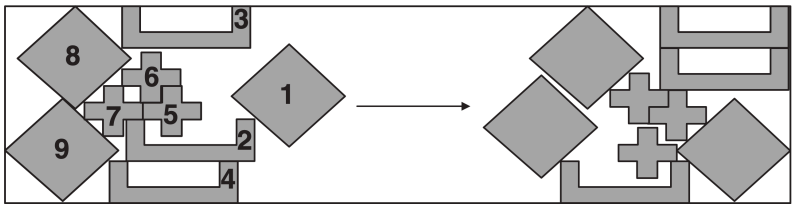


Figure 6.4: Jostle principle (Bennell and Oliveira, 2009).

6.2.3. Searching over the physical layout

When the search is over the layout rather than the sequence, pieces are moved within the actual physical layout. A common characteristic of such approaches is that usually overlap is allowed during the search, which is then penalised in the objective function. The benefit is that this results in a smoother solution landscape, improving search algorithm efficiency (Bennell and Oliveira, 2009). Another advantage is that it is not required to decode the sequence each iteration, which is a computationally expensive step (Egeblad et al., 2007). However, resolving overlap can be problematic and the algorithm can get stuck in the overlap minimisation stage, resulting in an infeasible solution (Ramakrishnan et al., 2008). Two main approaches are considered in the literature when searching over the layout. In the first approach, known as local search approach, a single piece or small number of pieces can moved within the layout, by either removing it from its current position and inserting it in a new position on the placement area, or swapping the positions of two (or more) pieces. Rotations can also be considered as a move. The second approach, known as compaction and separation, moves all pieces simultaneously although the search space is heavily restricted.

Local search approaches

The same problem of reducing the continuous search space to a set of discrete points discussed subsection 6.2.1 with regard to constructive algorithms arises in local search approaches when moving pieces. Methods commonly use the raster representation (and therefore possibly eliminate good solutions), or the concept of the no-fit polygon is used to obtain a set of promising placement points (Elkeran, 2013). The method proposed by Błażewicz et al. (1993) first generates a initial using the bottom left algorithm where the pieces are sorted by decreasing area. The method then attempts to move fill holes in the layout by moving the right most pieces into the holes, or if that is not possible the other (internal) pieces are considered. A feasible placement region within the hole is determined, and the most bottom left position of that region is chosen as placement point. If none of the pieces fits into a hole, the piece is placed to the right of the current layout in the unused area, again according to the bottom left principle. A tabu list prevents pieces that have recently been moved to move again during the tabu search process. This method does not allow for overlap, although many other approaches searching over the layout do allow overlap in the search process.

For example, Ramakrishnan et al. (2008) proposed a tabu search method where the piece which has the most amount of overlap is selected to be moved and that piece is then placed in a position that results in the least amount of overlap. The overlap is measured as the sum of the maximum horizontal and vertical penetration depth between pieces. The possible placement points are found based on the vertices of the no-fit and inner-fit polygon and their intersections, a method originally proposed by Gomes and Oliveira (2002) (discussed in subsection 6.2.1). Again, the piece that was moved is added to the tabu list. Rotation can be considered by generating the NFPs for each admissible orientation of the pieces. An initial solution is generated by the bottom left algorithm where the pieces are sorted by decreasing area. The length of the initial solution is then taken and the length of the boundary is reduced by one unit. Pieces which are not within the reduced boundary anymore, are translated back into the reduced boundary, resulting in overlap in the layout. The reduction step is

repeated every time a feasible solution is found during the search, until the algorithm is unable to find a feasible solution within the set time limit and the best solution is reported as final solution. Such an approach therefore separates the strip minimisation problem and the overlap minimisation problem into two separate parts. This approach of separating the two problems was also applied by Egeblad et al. (2007), although they proposed a different method to resolve overlap. Each iteration the selected piece is restricted to a horizontal or vertical translation in order to minimise overlap, and the intersection area theorem proposed by Nielsen and Odgaard (2003) is used to calculate the overlap. The main idea of the theorem is to express the overlap of a piece with all other pieces as a function of the horizontal (or vertical) position of the piece. The obtained overlap function is piecewise quadratic, and the placement position of the piece being moved is then moved to the same position as where the overlap function is minimal. The principle can be extended to included piece rotations (Nielsen and Odgaard, 2003). In order to avoid the overlap minimisation algorithm getting trapped in a local optimum, the metaheuristic guided local search is applied. In a guided local search penalties are applied to unwanted features and the objective function for the overlap minimisation process is adjust to incorporate those penalties. The approach was proven to be remarkably efficient compared to other approaches in literature, obtaining better solutions within 10 minutes of run time for certain instances and low standard deviations of the solutions between runs of the same data set.

In fact, the approach of separating the problem into two separate parts, namely length minimisation and overlap minimisation where the length is reduced and overlap is subsequently resolved, is a popular method in literature and all of the following methods adopted this principle. Umetani et al. (2009) also proposed a guided local search method, however the overlap was measured using the directional penetration depth and the piece is translated alternately in the vertical and horizontal directions until no better placement can be found. Leung et al. (2012) proposed a method which resolves overlap based on the penetration depths by solving a nonlinear model (to a local optimum). An initial bottom left layout is generated and two pieces within the layout are then swapped to generate neighbouring solutions. Resulting overlap is resolved by the nonlinear model. The metaheuristic tabu search is used to guide the search and the pieces recently move are stored in the tabu list. In addition, a second tabu list is created to record the piece type in order to forbid the next moves to be made with a piece of the same shape. This is important to note for the aircraft parking problem since aircraft fleet usually consists of only several (sub)types.

Elkeran (2013) proposed an approach based on the cuckoo search in combination with the guided local search algorithm. The cuckoo search is a nature inspired algorithm based on cuckoos, which are known to lay their eggs in nests of other birds. The algorithm firsts generates an initial pool of solutions randomly. One solution is chosen randomly and the algorithm then randomly alters this solution by Lévy flights (random walks with the steps drawn from a Lévy distribution). Another solution is chosen from the pool of solutions, and if the overlap of the newly generated solution is less, the new solution is adopted. In addition, pieces can be pairwise clustered in order to improve algorithm run time.

Recently, Sato et al. (2019) proposed an overlap minimisation approach where the placement area was reduced to a discrete set of placement points by rasterisation. The novel concept of the raster penetration map was introduced, which is constructed from the no-fit polygon and represents the raster penetration depth (in grid space) between two pieces. For a given piece, the penetration maps with all other pieces can be added to obtain the obstruction map, which can be thought of as some sort of heat map. Large values indicate that a lot overlap is present, while zero values indicate no overlap. Each admissible orientation results in a different NFP and thus a different penetration map and obstruction map. For a given piece, the whole space is searched and the predetermined fixed discrete orientations are tested to obtain the position with minimum overlap given by the obstruction map. The guided local search method is used to avoid local optima. A two stage multi-resolution is proposed. First, a lower resolution grid is used the search the whole space (within in the grid) and the position with minimum overlap is selected. In the second stage this position then becomes the centre of a new local grid with a higher resolution, restricted to a small space of the original resolution. Despite the discretisation, the approach was shown to be very competitive, matching or improving best known results for 9 out of the 15 benchmark data sets tested with 20 minutes of run time.

Compaction and separation

Another class of approaches for searching over the physical layout is known as compaction and separation approaches. However, these approaches appear to have lost popularity in recent years in favour

of the local search methods described above. Nevertheless, the principles of the approach are briefly discussed here. In such approaches, the problem is solved as a linear program which moves all pieces simultaneously. The objective function minimises length and the constraints ensure that the pieces remain with the boundaries of the placement area and do not overlap. Overlap constraints however can be relaxed and penalised in the objective function to allow for a more efficient search. The linear program model is highly constrained and restricts the solution space to only a small area. Bennell and Dowsland (2001) combined the linear program with a tabu search. The non-overlapping constraints are derived from the no-fit polygon concept, where the closest edge, or series of edges, of the NFP is (are) used as constraints in the LP model and restricts the piece's movement to a small, feasible area. It might therefore be necessary to run the LP model multiple times to completely resolve overlap as polygon are only allowed to move in small steps. Gomes and Oliveira (2006) proposed a similar approach, but combined linear programming with the metaheuristic simulated annealing.

6.3. Summarising cutting and packing problems

In short, this section can be summarised as follows. Exact approaches are not suitable for the purpose of long term parking of aircraft as only small instances with no or very limited rotations can be solved within a realistic amount of time. The problem is NP-hard and therefore heuristic approaches are commonly applied to the nesting problem. Heuristic approaches can be divided into two methods: searching over the sequence, and searching over the physical layout. In the former, the algorithm changes the sequence of the pieces in search for a better layout and the sequence is decoded into a layout by a constructive algorithm (also known as placement algorithm), following a fixed set of placement rules. The search procedure is often combined with the metaheuristics tabu search and genetic algorithm to guide the search over the solution space. Approaches searching over the physical layout move pieces within the physical layout during the search for better solutions, where a placement algorithm is used to generate an initial solution. Usually overlap is permitted in the search process when searching over the layout and the sub problem of overlap minimisation is solved separately from the layout length minimisation problem. Tabu search and guided local search are popular metaheuristics when searching over the layout. The bottom left algorithm is a common placement algorithm for both approaches. Both approaches achieve excellent results and provide promising starting points for the long term parking optimisation problem. An overview of the different heuristic based approaches is given in Table 6.1 in the same order as they were discussed in this section.

Table 6.1: Overview heuristic methods for the nesting problem. BL = bottom left, GC = gravity centre, TS = tabu search, GA = genetic algorithm, GLS = guided local search, SA = simulated annealing.

Author (Year)	Sequence	Layout	Placement Algorithm	Metaheuristic
Gomes and Oliveira (2002)	X		BL	-
Burke et al. (2006)	X		BL	TS
Ramakrishnan et al. (2008)	X		BL	TS
Jakobs (1996)	X		BL	GA
Liu and He (2006)	X		GC	GA
Mundim et al. (2017)	X		BL	GA
Abeysooriya et al. (2018)	X		TOPOS	JOSTLE
Błażewicz et al. (1993)		X	BL	TS
Ramakrishnan et al. (2008)		X	BL	TS
Egeblad et al. (2007)		X	BL	GLS
Umetani et al. (2009)		X	BL	GLS
Leung et al. (2012)		X	BL	TS
Elkeran (2013)		X	BL	GLS
Sato et al. (2019)		X	BL	TS
Bennell and Dowsland (2001)		X	BL	TS
Gomes and Oliveira (2006)		X	BL	SA

7

Conclusion literature study & research questions

The purpose of the is this report was to provide an overview of the existing literature relevant to the optimisation of the long term parking of aircraft, and to identify the research gap which presents a research opportunity. First, geometric tools used to solve the problem were reviewed. In order to represent the placement area and the pieces, and to facilitate overlap calculations, four geometric modelling techniques are commonly used in literature: the raster representation, direct geometry, the no-fit polygon (NFP), and the phi-functions. Furthermore, the main principles of several metaheuristics were explained. Next, approaches to a similar problem, namely that of aircraft parking in a maintenance hangar, were discussed. Several key differences were identified: in contrast to the hangar parking problem, for the purpose of long term parking, aircraft can be parked at any orientation, the amount of aircraft to be parked is larger, and wing and tail could be allowed to overhang other aircraft or unpaved surfaces. Finally, methods to solve the irregular cutting and packing problems, also know as nesting problems, were discussed. Exact approaches were found to be unsuitable as they are not capable of solving large data sets. Heuristic approaches can be divided into methods that search over the sequence, and methods that search over the physical layout. Excellent results have been produced by both heuristic based approaches and there is no consensus in literature as to which approach is superior. Hence, both heuristic approaches provide promising starting points for the optimisation of long term parking of aircraft.

To the best of the author's knowledge, the existing literature does not address the issue of the optimisation of the long term parking of aircraft. Although the gate assignment problem has been studied extensively in literature, in such problems aircraft are placed at fixed airport infrastructure gates (or stands) with large margins to allow for access of personnel and equipment during service. The maintenance hangar aircraft parking problem is related to the long term parking, although it differs in several key aspects (mentioned above).

Hence, the research gap is clear, which leads to the following research objective and research questions presented next.

7.1. Research questions

The proposed research objective of the proposed research is to optimise the long term parking of aircraft by using metaheuristic approaches. The research (sub)questions to be answered are defined as follows:

How to optimise the long term parking of aircraft by using metaheuristic algorithms?

1. *Which solution algorithm provides the best results?*
   (a) *How to evaluate the performance of the algorithm?*
   (b) *How does the optimisation perform compared to current (manual) practices?*
2. *Which geometric modelling technique provides the best result?*
3. *How does the fleet composition influence the performance of the model?*
   (a) *What is the effect of varying the fleet size?*
   (b) *What is the effect of diversifying the aircraft types?*

Bibliography

Abeysooriya, R. P., Bennell, J. A., & Martinez-Sykora, A. (2018). Jostle heuristics for the 2D-irregular shapes bin packing problems with free rotation. *International Journal of Production Economics*, *195*, 12–26.

Ackert, S. P. (2010). *Basics of aircraft maintenance programs for financiers* [Retrieved: 14 July 2020]. http://www.aircraftmonitor.com/uploads/1/5/9/9/15993320/basics_of_aircraft_maintenance_programs_for_financiers___v1.pdf

Airbus. (2005). *A320 aircraft characteristics airport and maintenance planning* (Technical report no. 03/02) [Available at https://www.airbus.com/content/dam/corporate-topics/publications/backgrounders/techdata/aircraft_characteristics/Airbus-Commercial-Aircraft-AC-A320.pdf. Retrieved: 27 July 2020.]. AIRBUS S.A.S., Customer Services, Technical Data Support, Services, 31707 Blagnac Cedex, FRANCE.

Alvarez-Valdes, R., Martinez, A., & Tamarit, J. M. (2013). A branch & bound algorithm for cutting and packing irregularly shaped pieces. *International Journal of Production Economics*, *145*(2), 463–477.

Art, R. C. (1966). An approach to the two dimensional irregular cutting stock problem. [(Bachelor's Thesis, Massachusetts Institute of Technology, Cambridge (Massachusetts), United States). Retrieved from https://dspace.mit.edu/bitstream/handle/1721.1/97782/25841973-MIT.pdf].

Bennell, J. A., & Dowsland, K. A. (2001). Hybridising tabu search with optimisation techniques for irregular stock cutting. *Management Science*, *47*(8), 1160–1172.

Bennell, J. A., Dowsland, K. A., & Dowsland, W. B. (2001). The irregular cutting-stock problem — a new procedure for deriving the no-fit polygon. *Computer & Operations Research*, *28*(3), 271–287.

Bennell, J. A., & Oliveira, J. F. (2008). The geometry of nesting problems: A tutorial. *European Journal of Operational Research*, *184*(2), 397–415.

Bennell, J. A., & Oliveira, J. F. (2009). A tutorial in irregular shape packing problems. *Journal of the Operational Research Society*, *60*, S93–S105.

Bennell, J. A., & Song, X. (2008). A comprehensive and robust procedure for obtaining the nofit polygon using minkowski sums. *Computer & Operations Research*, *35*(1), 267–281.

Bennell, J. A., & Song, X. (2010). A beam search implementation for the irregular shape packing problem. *Journal of Heuristics*, *16*, 167–188.

Błażewicz, J., Hawryluk, P., & Walkowiak, R. (1993). Using a tabu search approach for solving the two-dimensional irregular cutting problem. *Annals of Operations Research*, *41*, 313–325.

Burke, E. K., Hellier, R. S. R., Kendall, G., & Whitwell, G. (2007). Complete and robust no-fit polygon generation for the irregular cutting problem. *European Journal of Operational Research*, *179*(1), 27–49.

Burke, E., Hellier, R., Kendall, G., & Whitwell, G. (2006). A new bottom-left-fill heuristic algorithm for the two-dimensional irregular packing problem. *Operations Research*, *54*(3), 587–601.

Cherri, L. H., Cherri, A. C., & Soler, E. M. (2018). Mixed integer quadratically-constrained programming model to solve the irregular strip packing problem with continuous rotations. *Journal of Global Optimization*, *72*, 89–107.

Cherri, L. H., Mundim, L. R., Andretta, M., Toledo, F. M. B., Oliveira, J. F., & Carravilla, M. A. (2016). Robust mixed-integer linear programming models for the irregular strip packing problem. *European Journal of Operational Research*, *253*(3), 570–583.

Cuninghame-Green, R. (1989). Geometry, shoemaking and the milk tray problem. *New Scientist 12 August 1989 No 1677*, 50–53.

Egeblad, J., Nielsen, B. K., & Odgaard, A. (2007). Fast neighborhood search for two- and three-dimensional nesting problems. *European Journal of Operational Research*, *183*(3), 1249–1266.

Elkeran, A. (2013). A new approach for sheet nesting problem using guided cuckoo search and pairwise clustering. *European Journal of Operational Research*, *231*(3), 757–769.

Ferreira, J. C., Alves, J. C., Albuquerque, C., Oliveira, J. F., Ferreira, J. S., & Matos, J. S. (1998). A flexible custom computing machine for nesting problems. *Proceedings of the XIII Design of Circuits and Integrated Systems conference, Madrid, Spain.*, 686–691.

Gomes, A. M., & Oliveira, J. F. (2002). A 2-exchange heuristic for nesting problems. *European Journal of Operational Research*, *141*(2), 359–370.

Gomes, A. M., & Oliveira, J. F. (2006). Solving irregular strip packing problems by hybridising simulated annealing and linear programming. *European Journal of Operational Research*, *171*, 811–829.

Guépeta, J., Acuna-Agost, R., Briant, O., & J.P.Gayon. (2015). Exact and heuristic approaches to the airport stand allocation problem. *European Journal of Operational Research*, *246*(2), 597–608.

Hillier, F. S., & Lieberman, G. J. (2015). *Introduction to operations research* (10th ed.). Mc Graw Hill.

Jakobs, S. (1996). On genetic algorithms for the packing of polygons. *European Journal of Operational Research*, *88*(1), 165–181.

Junior, B. A., Pinheiro, P. R., & Saraiva, R. D. (2013). A hybrid methodology for nesting irregular shapes: Case study on a textile industry. *IFAC Proceedings Volumes*, *46*(24), 15–20.

Konopasek, M. (1981). Mathematical treatments of some apparel marking and cutting problems [99-90857-10]. *U.S. Department of Commerce Report*.

Leao, A. A. S., Toledo, F. M. B., Oliveira, J. F., Carravilla, M. A., & Alvarez-Valdés, R. (2020). Irregular packing problems: A review of mathematical models. *European Journal of Operational Research*, *282*(3), 803–822.

Leung, S. C. H., Lin, Y., & Zhang, D. (2012). Extended local search algorithm based on nonlinear programming for two-dimensional irregular strip packing problem. *Computers & Operations Research*, *39*, 678–686.

Li, X., Wang, Z. X., Chan, F. T. S., & Chung, S. H. (2019). A genetic algorithm for optimizing space utilization in aircraft hangar shop. *International Transactions in Operations Research*, *26*(5), 1655–1675.

Liu, H., & He, Y. (2006). Algorithm for 2D irregular-shaped nesting problem based on the NFP algorithm and lowest-gravity-center principle. *Journal of Zhejiang University SCIENCE A*, *7*(4), 570–576.

Liu, X., & Ye, J. (2011). Heuristic algorithm based on the principle of minimum total potential energy (HAPE): A new algorithm for nesting problems. *Journal of Zhejiang University-SCIENCE A (Applied Physics & Engineering)*, *12*(11), 860–872.

Mahadevan, A. (1984). Optimization in computer-aided pattern packing [(PhD Thesis, University of North Carolina, United States)].

Mundim, L. R., Andretta, M., & Queiroz, T. A. (2017). A biased random key genetic algorithm for open dimension nesting problems using no-fit raster. *Expert Systems with Applications*, *81*, 358–371.

Nielsen, B. K., & Odgaard, A. (2003). *Fast neighborhood search for the nesting problem* (Technical report no. 03/02) [Available at https://di.ku.dk/forskning/Publikationer/tekniske_rapporter/2003/03-03.pdf. Retrieved: 27 July 2020.]. University of Copenhagen. DIKU, Department of Computer Science, Copenhagen, Denmark.

Oliveira, J. F., Gomes, A. M., & Ferreira, J. S. (2000). TOPOS – A new constructive algorithm for nesting problems. *OR-Spektrum*, *22*, 263–284.

Peralta, J., Andretta, M., & Oliveira, J. F. (2018). Solving irregular strip packing problems with free rotations using separation lines. *Pesquisa Operacional*, *38*(2), 195–214.

Qin, Y., Chan, F. T. S., Chung, S. H., Qu, T., & Niu, B. (2018). Aircraft parking stand allocation problem with safety consideration for independent hangar maintenance service providers. *Computer and Operations Research*, *91*, 225–236.

Qin, Y., Wang, Z., Chan, F. T., Chung, S., & Qu, T. (2019). A mathematical model and algorithms for the aircraft hangar maintenance scheduling problem. *Applied Mathematical Modelling*, *67*, 491–509.

Ramakrishnan, K., Bennell, J. A., & Omar, M. K. (2008). Solving two dimensional layout optimization problems with irregular shapes by using meta-heuristic. *2008 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore*, 178–182.

Rocha, P., Rodrigues, R., Gomes, A. M., Toledo, F. M. B., & Andretta, M. (2014). Circle covering representation for nesting problems with continuous rotations. *Proceedings of the 19th World Congress. The International Federation of Automatic Control. Cape Town, South Africa.*, 5235–5240.

Sato, A. K., Bauab, G. E. S., de Castro Martins, T., de Sales Guerra Tsuzuki, M., & Gomes, A. M. (2018). A study in pairwise clustering for bi-dimensional irregular strip packing using the dotted board model. *IFAC-PapersOnLine*, *51*(11), 284–289.

Sato, A. K., Martins, T. C., Gomes, A. M., & Tsuzuki, M. S. G. (2019). Raster penetration map applied to the irregular packing problem. *European Journal of Operational Research*, *279*, 657–671.

Stoyan, Y., Terno, J., Scheithauer, G., Gil, N., & Romanova, T. (2001). Phi-functions for primary 2D-objects. *Studia Informatica Universalis*, *2*(1), 1–32.

Stoyan, Y., Pankratov, A., & Romanova, T. (2016). Cutting and packing problems for irregular objects with continuous rotations: Mathematical modelling and non-linear optimization. *Journal of the Operational Research Society*, *67*(5), 786–800.

Toledo, F. M. B., Carravilla, M. A., Ribeiro, C., Oliveira, J. F., & Gomes, A. M. (2013). The dotted-board model: A new MIP model for nesting irregular shapes. *International Journal of Production Economics*, *145*(2), 478–487.

Umetani, S., Yagiura, M., Imahori, S., Imamichi, T., Nonobe, K., & Ibaraki, T. (2009). Solving the irregular strip packing problem via guided local search for overlap minimization. *International Transactions in Operational Research*, *16*, 661–683.