Oude Waalsdorperweg 63
2597 AK Den Haag
Postbus 96864
2509 JG Den Haag

www.tno.nl

T +31 88 866 10 00
F +31 70 328 09 61
infodesk@tno.nl

The Online UAV Mission Planning Problem

Master's Thesis
Amarins van de Voorde

September 28, 2012

Supervisors TNO
L. Evers MSc
Dr. A.I. Martins Botto de Barros

Supervisor UU
Prof. Dr. Ir. E.J. Balder

Report number: TNO 2012 S10674

Abstract

In the Online UAV[1] Mission Planning Problem a reconnaissance mission has to be planned in a given area that contains both targets that are given beforehand and new targets that arise during the flight. The goal of the mission is to gather information from a subset of the targets, in such a way that the gathered amount of information is maximal, whilst keeping the fuel required for the mission within predefined limits.

We developed five strategies by which the planned tour can be reoptimized during the flight, either for the general problem where the new targets appear in the entire target area, or for the special case where all new targets appear in some prespecified zone. All strategies were composed of two steps: finding an optimal initial tour first and adjusting the tour later on during the flight, when information about new targets was released.

Theoretical as well as empirical performance bounds were found for the strategies that were developed for the general problem. Finally, the performance of the strategy for the special case was compared to the performance of the other strategies, when applied to this case.

[1]Unmanned Aerial Vehicle

Acknowledgements

First and foremost, I would like to thank my supervisors at TNO, Lanah Evers and Ana Barros. You were always there for me when I needed it, took your time in helping me and taught me a lot about myself. Thank you again for giving me the opportunity to come to TNO.
Secondly, I would like to give my gratitude to Erik Balder, my supervisor at Utrecht University, for making sure that I kept going in the right direction.

A heartfelt thanks goes out to all my colleagues at TNO. You made me feel very welcome from the start and I really enjoyed my stay, which was of course livened up by all the coffee breaks and '3k-borrels'. In particular, I would like to mention my fellow interns and officemates Corrinne Luteyn and Arnold Bakker, and thank you for 'putting up' with me all this time. In addition, I would like to thank Paul Eigeman, Nick van der Poel and Eelco Kuipers for rescuing me when Java was getting the better of me.

Lots of thanks to my sister Imelda: I really enjoyed spending more time with you. Thank you for all your advice and for listening to my problems. I am truly going to miss our daily bike-trips together!
And last, but definitely not least, I would hereby like to thank my parents, Gelein and Rixt van de Voorde, for supporting me in so many ways during my education. Without you I really would not have come this far.

Contents

1 Introduction

Unmanned Aerial Vehicles (UAVs) provide a valuable information source for both civilian and military operations, as they can be used to gather imagery about territories that are hard to scout otherwise; in particular they can be used to gather imagery from the military theater. The goal of such reconnaissance missions is to gather as much information as possible about certain locations in a given area. Those target locations will provide information of some value, but that value may vary per location. Since it is unlikely that a UAV will be able to visit all target locations in the area of interest - mainly due to fuel capacity - it is paramount to find a feasible tour with maximal total information value.

When all target locations, all information values of those locations, and the fuel requirements to go from any target to another are known, this problem reduces to a known optimization problem. But in real-life situations, not all data might be certain or even known beforehand. We will mention four characteristics that extend the standard model, which might occur in a real-life UAV mission planning problem. First of all, weather conditions may influence the fuel consumption, both positively and negatively, which in turn can influence the optimality or even the feasibility of a chosen tour. Secondly, targets may turn out to have decreased information value, they may disappear all together, or new targets might arise, i.e. there might not be a fixed set of target locations. A problem with this extension is generally referred to as an *online problem*. Thirdly, it might not be possible to collect information from a target all the time. For such targets a time-window has to be taken into account. Finally, there is the case when multiple vehicles are available. This may enable a set of flight plans that covers all targets, but for these flights the same additional factors could be taken into account. And for multiple vehicles there is another factor that can be considered. It might occur that one of the vehicles cannot visit all its scheduled targets, due to one or more of the dynamic factors. Then the flight plan of one of the other vehicles could be altered, if possible, to visit such a missed target. In short, there are four extensions to the basic problem:

- uncertain fuel consumption
- time windows
- changing target set
- multiple vehicles

There are several problems that are related to the basic UAV Mission Planning Problem (UAV-MPP). By the UAV-MPP we refer to the problem with deterministic fuel-consumption, no time-sensitive targets, a fixed target set and a single vehicle. As mentioned before, the basic UAV-MPP reduces to a known optimization problem: the Orienteering Problem (OP), see [11].This problem is based on the game of 'Orienteering', where cross country runners have to navigate through unfamiliar terrain in which a number of control posts are located. A value is assigned to each control post and the runners collect

the value of a control post by visiting it. The runners have to return to the starting point within a given time limit. As they will not be able to visit all control posts, their goal is to construct a tour that passes a subset of the control posts that maximizes the total collected value.
There even are variants of the OP that can be adapted for the basic problem with some of the extensions, as is discussed in [43]. For instance, the Team Orienteering Problem (TOP) could be used for a UAV mission planning problem with multiple vehicles. The UAV mission planning problem with time-sensitive targets could be modeled using the Orienteering Problem with Time Windows (OPTW). And in [12] it becomes clear that the OP can be adapted to model the subproblem with uncertain fuel usage.

Besides the OP, there are more routing problems that can be adapted to model extensions of the UAV-MPP. There is the widely known Traveling Salesman Problem (TSP), where, given a set of cities and the distances between all those cities, a single tour of shortest length has to be planned, that visits each city exactly once. Note that this is a specific case of a larger problem, the TSP with multiple (m) vehicles, where a set of m tours have to be found, such that every location (except the depot) is visited by exactly one vehicle. This problem is denoted by the m-TSP; the well known TSP is therefore just a version of the m-TSP, with $m = 1$. Another example is the Vehicle Routing Problem (VRP), [26], which is similar to the m-TSP, but with an extra constraint: the vehicles now have load capacities that cannot be exceeded. Goal for the VRP is also to minimize the total length of all tours.
A disadvantage of both m-TSP and VRP is that they differ from the UAV-MPP on two important properties. First of all, each city has to be visited, while in the UAV-MPP it is very likely that not all targets can be visited. Secondly, the goal of both problems is to minimize the total traveled distance, whereas the UAV-MPP has as objective to maximize the amount or value of gathered information.
But on the other hand, the VRP can be adapted with real-time elements or time-windows, as can be seen in [15]. So problems like the VRP could be used to gather insight in how to include one or more of the extensions into the associated variant of the UAV-MPP.
A question that now arises is whether or not there may be more problems that can be adapted to model the UAV-MPP with one or more extensions. Another question is which model should be adapted for which subproblem. Is the OP the best model, or can the VRP or the TSP be adapted in such a way, that the UAV-MPP can be modeled by it? We would like to know as well whether the OP can be adapted for any of the extensions, whether solution methods exists for those extensions, and whether something can be said about their performance. We also would like to know whether some of the extensions have been researched already.
Another aspect of the UAV-MPP is that it clearly has practical applications. So while it is very interesting to find theoretical bounds for the solution methods, it is also important to determine how well the methods will work in practice.

In order to determine which subproblems will be most interesting, the research started with a literature analysis. During this analysis a number of questions will be answered:

1. Literature analysis
   - What research has been done on Orienteering Problems, and possibly on related problems, that can be of use for the subproblems of the UAV Mission Planning problem?
   - Based on the findings of the literature analysis, which subproblem(s) will be most interesting for further investigation?

When subproblems have been chosen, the following questions will be answered:

2. How can the chosen subproblems be modeled best? What solution methods can be used?
3. Can theoretical bounds be found for the performance of the solution methods?
4. How well do the solution methods perform in practice? Are the theoretical bounds correct, or can better empirical bounds be found?

2 Literature analysis

A lot of research has been done on the OP and on related problems. The amount of publications on this topic is vast, and its range wide - from simple definitions of the problem and its history to detailed solution algorithms or heuristics and from the basic problem to versions with multiple extensions. It is therefore important to first get an overview of the current research, in order to be able to determine which aspects of the problem are most interesting to study.
As mentioned in the introduction, the basic UAV-MPP can probably be modeled best by the OP or a similar problem, but it has several extensions that are not necessarily embedded in those problems. These characteristics are uncertain fuel usage, time-sensitive targets, a set of targets that may change during the flight and use of multiple vehicles. For a number of papers, we have determined which of those extensions have been taken into account. Apart from the extensions, the problem type has been noted and for each article we have checked whether some form of a bound on the performance of the solution has been found.
Most of these results are summarized in table 1. Some of the papers found only discuss heuristics for solving the OP; the greater part of those are not listed in the table.

As can be seen in table 1, many of the papers found on the OP focus either on the extension with time windows, known as the Orienteering Problem with Time Windows (OPTW), or on the extension with multiple vehicles, known as the Team Orienteering Problem (TOP). Part of those papers even focus on the combined subproblem in the Team Orienteering Problem with Time Windows (TOPTW). Those papers only provide heuristics, not exact solution methods.
A few papers take non-deterministic travel times into account: [9], [39], [12] and [13]. In the former two papers the travel times are stochastic, that is, the travel times are distributed according to some specified distribution and the actual travel times only become known during the flight. This implies that there is a chance that the actual length of a planned tour exceeds its time limit, so not all planned tours turn out to be feasible. In [12] the Robust Orienteering Problem (ROP) is introduced. In this problem the travel and service times are also approximated with random variables, but no assumptions have been made on the distributions of those variables. Additional variables are introduced instead, which are used to integrate the uncertainty into the travel times, absorbing certain deviations from the actual travel times. In [13] the ROP is compared to a stochastic programming approach of the same problem.

To the best of our knowledge there are very few papers that discuss the third extension, a changing set of targets. The only paper that we did find is by [28]. In this paper a combination of 3 of the extensions is made: time windows, multiple vehicles and a changing set of targets. There is also other research on problems with this extension, but we have only seen it in combination with either the TSP, as can be seen in [3], [7] and [19] or the VRP, as in [17] and [30].

For the OP other combinations of the four characteristics have not been found, but they do exists for the other problems. For instance, the combination of non-deterministic

travel times and time-sensitive targets has been researched for both the TSP, as in [7], and the VRP, as in [17] and [30]. The combination of non-deterministic travel times and multiple vehicles for the VRP has been researched in [17], [30] and [35]. The former two papers even take all four characteristics into account, but as they investigate the VRP, their approaches can not necessarily be applied to the OP.

Whilst most of the found research is concerned with finding heuristics to solve problems of any size and usually compare their results with benchmark instances, some of the papers present exact algorithms. In a few cases a performance bound is found. Exact algorithms for the OP have been found by [27] and [33]; in [8], [18], [24], [39] and [41] algorithms were found for small instances.
When the problem is an online problem, not all information is known beforehand, so it is not possible to find an exact algorithm or an exact solution. In order to be able to say something about the performance of algorithms for online problems, their outcome, determined by the value of the objective function, is compared to the optimal so-called *offline solution*. Note that in the offline problem all targets that are to appear are known beforehand, and to each target a release date is assigned; some of those release dates are very likely to be 0. The fraction of online outcome and optimal (offline) outcome is the *performance ratio* for the problem. When the performance ratios for all instances of a problem are compared, a lower bound for all these bounds can be found. This bound is called the *competitive ratio*. The notion of a competitive ratio for online algorithms was developed by [23]. For the online TSP, competitive ratios are presented by [3], [7], [19], [20] and [21].

Based on these results, we have drawn some conclusions. First of all, neither the OP with time windows, the Team OP nor the Team OP with Time Windows will be interesting to investigate for this thesis, as a lot of research has been done on those subproblems already. The same holds for the subproblem with uncertain fuel consumption.
The OP in combination with a changing set of targets will be very interesting for further study, as - as mentioned before - very little research exists on this subproblem. And while several theoretical bounds have been found for solution methods for the online TSP, to the best of our knowledge no research has been done yet on finding theoretical bounds for solution methods for the online problem. Therefore, the UAV-MPP extended with a changing set of targets is the subproblem that will be investigated further. Whilst there are many other combinations of subproblems of the UAV-MPP that might be interesting to investigate, the second part of the research for this thesis will also concern itself with the UAV-MPP extended with a changing set of targets. In that case, we will investigate the cases where the targets appear according to different arrival rates.

Table 1: Overview of characteristics in found literature
(*a cross signifies that that specific characteristic has been researched in the corresponding paper. 'Fuel' means non-deterministic travel times, 'TW' means time windows, 'online' means a changing set of targets and 'team' means the use of multiple vehicles*)

Article	Fuel	TW	Online	Team	Type	Remarks
[2]				x	OP	
[6]				x	OP	
[8]				x	OP	exact algorithm for some cases
[9]	x				OP	
[10]				x	OP	
[12]	x				OP	
[13]	x				OP	
[16]					OP	
[18]					OP	exact algorithm for some cases
[22]		x			OP	
[24]					OP	exact algorithm for some cases
[25]		x		x	OP	
[27]					OP	exact
[29]		x		x	OP	
[32]					OP	
[33]		x			OP	exact
[36]					OP	
[37]				x	OP	
[38]				x	OP	
[40]				x	OP	
[39]	x				OP	exact algorithm for some cases
[41]		x		x	OP	exact algorithm for some cases
[44]		x		x	OP	
[45]				x	OP	
[46]				x	OP	
[28]		x	x	x	OP	
[5]					OP/TSP	
[3]			x		TSP	competitive ratio
[7]	x	x			TSP	competitive ratio
[14]				x	TSP	exact for single vehicles
[21]				x	TSP	competitive ratio
[1]			x		TSP	
[19]		x	x		TSP	competitive ratio
[20]			x		TSP	competitive ratio
[17]	x	x	x	x	VRP	
[30]	x	x	x	x	VRP	
[35]	x			x	VRP	

3 Problem description

3.1 Modeling the UAV-MPP

The basic UAV-MPP will be modeled as the OP. First of all there is a set of targets $V = \{1, \ldots, n\}$ and a depot 0, so the total set of locations is $V^+ = V \cup \{0\}$. There is a set of arcs $A = \{(i,j)|i \neq j \in V^+\}$ between all pairs of locations. This implies that we can construct a graph $G = (V^+, A)$ that consists of all locations and all arcs in between the locations. We assume that this graph is directed, i.e. that for any pair $i, j \in V^+$ $(i \neq j)$ of locations both arc (i,j) and arc (j,i) exist. Each of these arcs (i,j) has a length $d(i,j)$. We assume that the lengths of the two arcs between two vertices are not necessarily equal, i.e. $d(i,j) \neq d(j,i)$. Therefore, $|A| = n(n+1)$ and graph G is a complete directed graph on $m+1$ vertices.
The length of arc (i,j) corresponds to the amount of fuel that is required to get from i to j. For each mission there is a fuel capacity C. Note that the vehicle travels at unit speed, so the concepts of time and distance can be interchanged.
Just as for the OP, the solution of the UAV-MPP will be a tour along some of the vertices from set V^+. This tour will be represented as an ordered set of vertices, and is denoted by T. In order to define the ordering of T, we first need to look at a different representation of the tour. The tour can be expressed as a sequence A_T of arcs from A by which the vehicle travels when following tour T. Now let $(i,j) \in A_T$. Then we can define an ordering $i < j$, where the binary relation $<$ stands for "is visited earlier than". Note that this ordering holds for any vertex k that is visited after i.
For example, let a tour T consist of vertices 7, 1 and 9. Then we can express the tour as $T = \{0, 7, 1, 9\}$ and the set of subsequent arcs is $A_T = \{(0,7), (7,1), (1,9), (9,0)\}$. Note that the arc $(9,0)$ is added: the vehicle always returns to the depot after visiting the last target on the tour.
Each tour has a length L_T, which is determined by the sum of distances between the consecutive pairs of vertices on the tour. In case of the example, the length of the tour will be determined as follows: $L_T = d(0,7) + d(7,1) + d(1,9) + d(9,0)$.
Set x_{ij} as a decision variable for whether arc (i,j) is on tour T. Then

$$x_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in A_T \\ 0 & \text{otherwise} \end{cases}$$

Similarly, we can define a decision variable for whether vertex i is in the tour:

$$x_i = \begin{cases} 1 & \text{if } i \in T \\ 0 & \text{otherwise} \end{cases}$$

Based on all these definitions, we can now formulate the optimization problem:

$$\text{(OP)} \qquad \max \sum_{i \in V^+} x_i v(i) \tag{1}$$

$$\text{s.t.} \sum_{(i,j) \in A} x_{i,j} d(i,j) \leq C \tag{2}$$

$$\sum_{i \in V} x_{0,i} = \sum_{i \in V} x_{i,0} = 1 \tag{3}$$

$$x_j = \sum_{i \in V^+} x_{i,j} = \sum_{k \in V^+} x_{j,k} \leq 1 \qquad \forall j \in V \tag{4}$$

$$1 \leq u_i \leq n \qquad \forall i \in T \tag{5}$$

$$u_i - u_j + 1 \leq (1 - x_{i,j})n \qquad \forall i, j \in V \tag{6}$$

$$x_i \in \{0,1\} \qquad \forall i \in V^+ \tag{7}$$

$$x_{i,j} \in \{0,1\} \qquad \forall (i,j) \in A. \tag{8}$$

In this problem, (2) is the fuel constraint, (3) makes sure that the tour start and ends at the depot. Constraints (4) are the flow conservaton constraints, but they also make sure that each location gets visited at most once and assign the correct values to all decision variables x_i. Constraints (5) and (6) prohibit subtours.

3.2 UAV-MPP with uncertain fuel usage

For each of the extensions, ways have been found to adapt the nominal OP. As was mentioned in the introduction, [12] found a formulation for the OP with uncertain fuel usage: the ROP. This is modeled like the nominal OP, but a few constraints have been added, such that the following problem has been constructed:

$$\text{(ROP)} \qquad \max \sum_{i \in V} x_i v(i),$$

$$\text{s.t.} \sum_{(i,j) \in A} x_{ij} \bar{d_{ij}} + \sum_{s \in S} \rho_s \|y^s\|_s^* \leq C, \tag{9}$$

$$\sum_{s \in S} y_{ij}^s = \sigma_{ij} x_{ij} = (\sigma \otimes x)_{ij}, \qquad \forall (i,j) \in A, \tag{10}$$

$$\sum_{i \in V} x_{0,i} = \sum_{i \in V} x_{i,0} = 1,$$

$$x_j = \sum_{i \in V^+ \setminus \{j\}} x_{ij} = \sum_{k \in V^+ \setminus \{j\}} x_{jk} \leq 1, \qquad \forall j \in V,$$

$$u_i - u_j + 1 \leq (1 - x_{ij})n, \qquad \forall i, j \in V,$$

$$1 \leq u_i \leq n, \qquad \forall i \in V,$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in A,$$

$$y_{ij}^s \in \mathbb{R}, \qquad \forall s \in S, (i,j) \in A, \tag{11}$$

Note that in this case, the fuel usage on arc (i, j) is denoted by $\bar{d}_{ij}$: this is the expected fuel usage from location i to j. Also, (9) replaces (2) in the nominal OP, together with (10). For more details on how these constrants were obtained, see [12].

3.3 UAV-MPP with time-windows

The time-window extension can be easily included in the nominal OP, as can be seen in [43]:

$$
\begin{aligned}
\text{(OPTW)} \qquad & \max \sum_{i \in V} x_i v(i), \\
& \text{s.t.} \sum_{(i,j) \in A} d(i,j) x_{ij} \leq C, \\
& \sum_{i \in V} x_{0i} = \sum_{i \in V} x_{i0} = 1, \\
x_j = \sum_{i \in V^+ \setminus \{j\}} x_{ij} = & \sum_{i \in V^+ \setminus \{j\}} x_{jk} \leq 1, \qquad \forall j \in V \\
t_i + d_{ij} - t_j \leq & M(1 - x_{ij}), \qquad \forall i, j \in V^+ \qquad (12) \\
r_i \leq & t_i, \qquad \forall i \in V^+ \qquad (13) \\
t_i \leq & d_i, \qquad \forall i \in V^+ \qquad (14) \\
x_{ij} \in & \{0, 1\}, \qquad \forall (i,j) \in A
\end{aligned}
$$

In this problem, Constraints (12), (13) and (14) have been added. In these constraints, a few new variables have been introduced: t_i is the moment the vehicle arrives at target i, r_i is the first moment target i can be visited and d_i the last, so $[r_i, d_i]$ is the time-window for target i. This implies that Constraints (13) and (14) make sure that target i is visited within its time-window. Note that when $r_i = 0$ and $d_i = C$, target i can always be visited, i.e. it has no time-window. For $i = 0$ we set t_0 as the time that the vehicle returns at the depot.
The third new constraint, (12), is a variant of the subtour-elimination constraint; it makes sure that the order of the targets, timewise, corresponds to the order of the targets in the tour. Variable M is set to a large constant.

3.4 UAV-MPP with multiple vehicles

The OP can be adapted to model the problem with the multiple-vehicle extension as well:

$$\text{(TOP)} \qquad \max \sum_{i \in V} v(i) \sum_{p=1}^{P} y_{ip}, \tag{15}$$

$$\text{s.t.} \sum_{p=1}^{P} \sum_{(i,j) \in A} d(i,j) x_{ijp} \leq C, \tag{16}$$

$$\sum_{i \in V} \sum_{p=1}^{P} x_{0ip} = \sum_{i \in V} \sum_{p=1}^{P} x_{i0p} = P, \tag{17}$$

$$\sum_{p=1}^{P} \sum_{i \in V^+ \setminus \{j\}} x_{ijp} = \sum_{p=1}^{P} \sum_{k \in V^+ \setminus \{j\}} x_{jkp} \leq 1, \qquad \forall j \in V \tag{18}$$

$$\sum_{p=1}^{P} y_{ip} \leq 1, \qquad \forall i \in V \tag{19}$$

$$1 \leq u_{ip} \leq n, \qquad \forall i \in V, \forall p \leq P \tag{20}$$

$$u_{ip} - u_{jp} + 1 \leq (2 - x_{ijp})n, \quad \forall (i,j) \in A, \forall p \in P \tag{21}$$

$$x_{ijp}, y_{ip} \in \{0,1\}, \quad \forall i \in V, \forall (i,j) \in A. \tag{22}$$

In this problem most of the constraints and variables have been replaced. Say there are $P \in$ tours, then those tours have to be nearly disjoint: they must have vertex 0 in common, but can't have any other shared vertices or arcs. The variable x_{ijp} denotes whether arc (i,j) lies on path p, so for each arc (i,j) there can be only one p for which this variable takes value 1, as can be seen in Constraint (18). A similar constraint is constructed for the depot in (17). Additional variables y_{ip} are introduced to make sure that each vertex is on at most one tour, in Constraint (19).

For the version with a changing set of targets, a different approach has to be used. This will be discussed in the next chapter.

4 A changing set of targets

In the UAV-MPP with a changing set of targets, also known as the *online* UAV-MPP, we assume that there are two types of targets. Targets of the first type, 'fixed targets', are known in advance, and can be visited at any time. Targets of the second type, 'new targets', are not known at first, but appear at some moment during the flight and become available the moment they appear. We assume that all new targets appear at time C at the latest. We also assume that targets do not disappear.

It is not obvious how this online problem should be modeled. Due to the fact that knowledge about some of the targets is only gained when those targets appear, it is very difficult to find a programming formulation like the (OP). This means that for the online problem, a different type of solution algorithm has to be found. The performance of algorithms for online problems is generally measured by comparing the outcomes of its objective function to the outcome of the objective function of the optimal solution in the so-called *offline* case. As mentioned in Chapter 2, these performance ratios are bounded by their competitive ratio [23], which is a worst-case ratio over all instances of the problem. For a maximization problem, this implies that we want to find a lower bound on the performance ratios of all instances. It can be determined as follows: let I be an instance of some problem. Let I^* be the optimal offline solution for this instance and $|I^*|$ its value of the objective function. Let I_A be a solution for this instance by some algorithm, with objective function-value $|I_A|$. Then $r_c = \sup\left\{r \,\middle|\, \frac{|I_A|}{|I^*|} \geq r, \forall \text{ instances } I_A\right\}$ is the competitive ratio for a maximization problem. Note that in case of a minimization problem, the competitive ratio is an upper bound for the performance ratios. In that case, we express the competitive ratio as $r_c = \inf\left\{r \,\middle|\, \frac{|I_A|}{|I^*|} \leq r, \forall \text{ instances } I_A\right\}$.

The offline case differs from the online case in that the release dates of all targets are given beforehand. But when all required knowledge is known beforehand, a programming formulation can be used. We can therefore model the offline case of the online UAV-MPP as an OP with time-windows (OPTW): the n fixed targets have release date 0 and end date C, the new targets $n+i,\ i \geq 1$ have a release date $r_i > 0$ and end date $d_i = C$. This problem can be solved with an exact algorithm, as was shown in [33].

In case of the online UAV-MPP, there is a set of fixed locations $V^+ = \{0, 1, \ldots, n\}$. We assume that the number of targets that appears before time C is equal to some random variable $\mu \in$ and that the value of both fixed and new targets is equal to 1, i.e. $v(i) = 1,\ \forall i \in \{1, \ldots, n+\mu\}$.[2] This implies that in both the online and the offline version of the problem we want to maximize the number of targets in the tour. Also, we assume that no additional fuel is required at any target.

Let's first take a look at the problem *without* new targets. This is just a regular OP-like problem, so we can assume that an optimal solution can be found, as exact algorithms exist for the OP, see for instance [27]. Let's denote this tour, that contains m targets, by T_m and its value of the objective function, i.e. the number of targets it contains, by $|T_m|$. Therefore, $|T_m| = m$. Similarly, when considering the situation where new targets may appear, the tour found by an online algorithm is denoted by T_A and its objective function-value by $|T_A|$. The optimal offline tour is denoted by T^* and its value by $|T^*|$.

In this chapter we will present a few algorithms that generate online tours. Each of these

[2]Note that at the depot no information can be gathered, so $v(0) = 0$.

algorithms only work when the ratio of existing and new targets is not too small: when the number of new targets is excessively large in comparison to the number of fixed targets, then these algorithms will probably have suboptimal results. We have not developed algorithms for such cases, as it is very unlikely that the number of new targets exceeds the number of existing targets.
For each of the algorithms it is not only important to determine the competitive ratio, but also to look at the computational time. An algorithm that has a high competitive ratio, that also takes a lot of computational time is less interesting to use, since some of the reoptimization steps should take place during the flight. Therefore, for each of the algorithms we will look at the computational time as well.

4.1 Insert algorithm

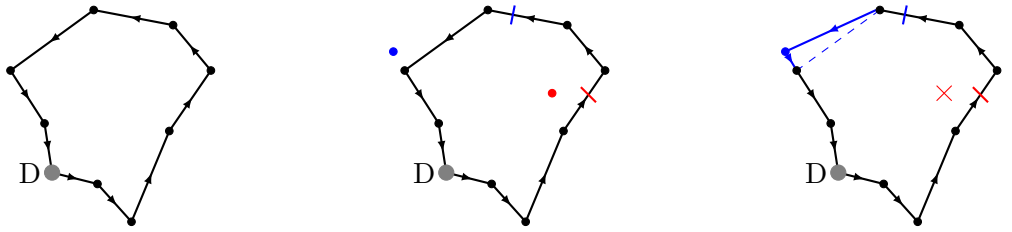
The first algorithm is a greedy algorithm, based on the extended model for online vehicle routing, as presented in [30]. The idea behind it is to insert new targets in the tour at the 'best' location, that is, with the least additional fuel consumption, and only when the fuel required to finish this adapted tour does not exceed the current remaining amount of fuel. This algorithm is called the 'Insert algorithm' (IA) and is defined below.
For each new target that appears, there is an arc in the tour to which it lies nearest. There are three decision rules for this nearest arc and the new target. Exactly one of the rules will be applied. The rules will therefore be checked in the following order, until the rule is reached that fits the situation:

(i) The tour has already passed the arc completely, so the target will not be visited.

(ii) The new target lies closest to the arc that the vehicle is currently traveling. Let (i, j) be this arc, k the new target and l the current location. If $d(i, l) \geq d(i, k)$ or if $d(i, k) \geq d(i, j)$, then the target will be ignored. Otherwise, the vehicle will deviate from the planned route and visit new target k directly, before continuing to target j and resuming the planned route, provided that the fuel capacities are not exceeded due to the additional fuel usage.

(iii) The arc has not been passed yet, so the new target will be added to the tour between targets i and j if the remaining fuel allows the additional fuel usage.

The IA bears most similarities to the extended model in [30] in the rules for adding targets. The greatest difference between the IA and the extended model, is that the extended model allows the disappearance of any of the targets, whereas the UAV-MPP does not.
An example of the algorithm can be seen in figure 1.
In order to determine whether such a new target k can be added to the tour, the 'remaining fuel' C_r has to be updated, by subtracting the additional amount fuel that is required for visiting k from the current remaining fuel C_r. For instance: in case of situation (ii), target k is visited instead of going directly from target i to target j. Let us assume that the current location of the vehicle is l, which lies somewhere on arc (i, j). Then $C_r' := C_r + d(l, j) - d(l, k) - d(k, j)$. If $C_r' \geq 0$, new target k can be visited. Note that at the start of the tour, $C_r = C - L_T$.



(a) Original tour T

(b) Appearance of 2 new targets (*red/blue dot: new target; red/blue line: moment of appearance*)

(c) Adding feasible targets (*red target: infeasible, blue target: feasible; dashed line: removed arc*)

Figure 1: Example of the Insert algorithm

For each new target v that appears, a certain computation time is needed to determine if, and, if so, where this target is added to the tour. Let T' be the current tour, i.e. the tour that includes all new targets that have been added to original tour T. Note that $|T'| \geq |T|$.
First of all, the Euclidean distances from point v to each line $(i,j), \forall (i,j) \in A_{T'}$ (the set of all arcs on the current tour) are calculated, in order to determine which arc the new target lies closest to. This takes $m \cdot O(1) + O(m \log(m)) = O(m \log(m))$ time. Let $(i',j') \in A_{T'}$ be the closest arc for point v, then we have to verify whether the tour has passed (i',j') already: this is done by looking at the last passed vertex v_{last} of the current tour and comparing it to (i',j').

Case (i): $v_{\text{last}} \geq j'$ so new target v will be ignored. Note that in comparisons between vertices, like $i < j$, the inequality symbols $<, >, \leq, \geq$ stand for the binary relations as defined on ordered set T.

Case (ii): $i' = v_{\text{last}}$, then $d(i',v)$ and $d(i',l)$, the distance from vertex i' to current location l, are compared. If $d(i',l) < d(i',v)$ and $d(i',v) < d(i',j')$ and $d(l,v) + d(v,j') - d(l,j') \leq C_r$, the target will be added to the tour. The comparisons are all simple computations; this step therefore takes $O(1)$ time in total.

Case (iii): $v_{\text{last}} < i'$, then we have to determine whether $d(i',v)+d(v,j')-d(i',j') \leq C_r$. This takes $O(1)$ time.
Determining whether a new target v should be added to the tour takes therefore at most $O(m \log(m) + O(1) = O(m \log(m))$ time.

Let a set of n fixed targets be given and let m be the maximal number of those targets that can be visited for a given fuel capacity C.

Theorem 1. *The competitive ratio for the Insert algorithm is* $\frac{m}{m+\mu}$ *and this bound is tight.*

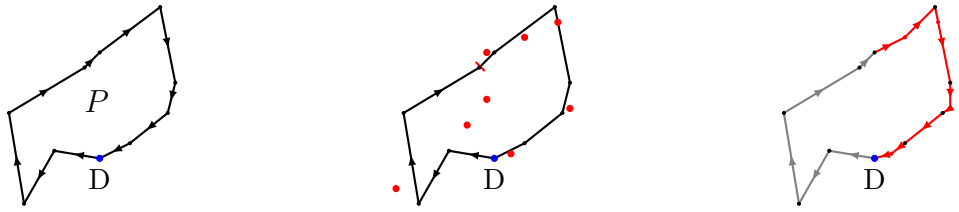
Proof. The original tour on the set of fixed targets contains m targets. During the flight μ new targets will appear, so at most $m+\mu$ targets can be visited within a fuel capacity C, no matter where the targets are located. The argument for this statement is quite clear: if $m+\mu+1$ targets can be visited, then, since only μ targets have been added, this implies that a tour along $m+1$ targets was possible in the case where the new targets were not taken into account. But the found tour with m targets was the optimal tour, so this is a contradiction. Hence, the optimal offline tour consists of at most $m+\mu$ targets,

and $|T^*| \leq m + \mu$, so $\frac{1}{|T^*|} \geq \frac{1}{m+\mu}$.
Also, since the online tour will be based on the route of the tour along the fixed targets, it is clear that the online tour will consist of at least m targets: in the worst case, only the m (fixed) targets will be visited. This implies that $|T_A| \geq m$. Therefore, $\frac{|T_A|}{|T^*|} \geq \frac{m}{m+\mu}$, so for any instance I of this problem, the performance ratio is at least $\frac{m}{m+\mu}$, so $r_c = \sup\left\{r \,\middle|\, \frac{|T_A|}{|T^*|} \geq r, \;\; \forall \text{ instances } I\right\} = \frac{m}{m+\mu}$.
By definition, such a bound is tight when there is an instance for which that bound is attained. Let T_m be the tour along the set of fixed targets that were known beforehand and L_{T_m} the fuel required for this tour. Let i be the first fixed target and j the last fixed target of tour T_m (so $T_m = (0, i, \ldots, j, 0)$). Based on this tour, 'reversed' tour T'_m can be constructed, in which the targets of T_m are visited in opposite direction, so $T'_m = (0, j, \ldots, i, 0)$. Then the length of this tour is $L_{T'_m}$. Since T_m was an optimal tour, $L_{T_m} \leq L_{T'_m}$. We assume that both tours are feasible, i.e. $L_{T_m} \leq L_{T'_m} \leq C$.
Now let each of the μ new targets appear at the time that the vehicle passes one of the fixed targets of the original tour T_m, at one location l on arc $(0, i)$ of the tour. When $\mu > m$, let the 'remaining' $\mu - m$ new targets appear at some time t: $d(0, l) < t \leq L_{T'_m} - d(l, 0)$. As mentioned before, the vehicle travels at unit speed, so time can be measured as distances. Since all new targets appear at location l, at or after time l, the online tour will not add any of them to the tour, either by rule (i) or rule (ii) of the Insert algorithm, and the vehicle visits only the m fixed targets, so $|T_A| = m$.
In case of the offline tour it is known that all μ targets arrive at location l before time $L_{T'_m} - d(l, 0)$. The offline tour will therefore travel backwards along the route of T_m, so the initial offline tour becomes tour T'_m. Then when the vehicle arrives at location l, all μ new targets have appeared there, so it can visit all of them. Hence, the number of targets in the offline tour is $|T^*| = m + \mu$ and the performance ratio of the Insert algorithm is $\frac{|T_A|}{|T^*|} = \frac{m}{m+\mu}$ for this instance.
Therefore, the bound $\frac{m}{m+\mu}$ on the performance ratios of the instances of the Insert algorithm is tight. □

4.1.1 Example of an instance with improved bounds

While the bound of $\frac{m}{m+\mu}$ is tight for the Insert algorithm, we will now show that there are instances of the problem for which a better performance ratio can be found. Let there again be a set of n fixed targets and a set of μ new targets, for some unknown μ. On the set of n targets a feasible optimal tour T_m of m targets can be constructed, so $L_{T_m} \leq C$. The set of arcs on tour T_m are again denoted by A_T. Let T'_m be again the 'reversed tour', and $L_{T'_m}$ its length. We assume that $C > L_{T'_m} \geq L_{T_m} > \frac{1}{2}C$ and that the remaining fuel is enough to add $\min\{\mu, m\}$ new targets to either tour, using the Insert algorithm.

Let the new targets appear in such a way that when half the available time, i.e. $\frac{1}{2}C$, has passed, half of the new targets have appeared near arcs of the second part of the tour. Let h be the location of the vehicle on initial tour T_m at time $\frac{1}{2}C$. Then we can define the second part of the tour: the shortest part of the tour that starts at location h and continues to depot 0. Note that location h is not necessarily one of the vertices of tour T_m, it can also lie on one of the arcs in A_T. This second part of the tour contains at



(a) Original tour T

(b) Arrival of new targets before $\frac{1}{2}C$ (*red dots: new targets; red line:* $\frac{1}{2}C$)

(c) Adding new targets to the tour

Figure 2: Example of the Insert algorithm with a better performance ratio

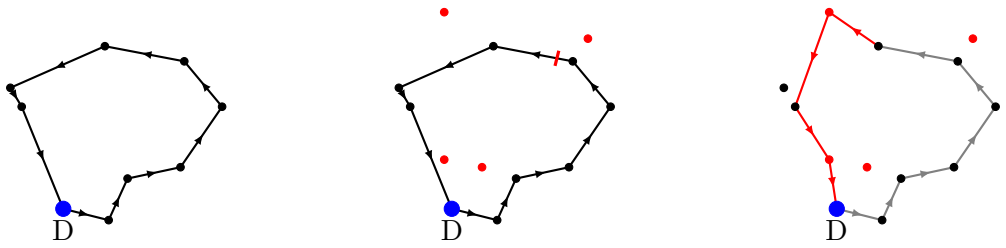
least $b = \lfloor (L_{T_m} - \frac{1}{2}C)/(\max_{(i,j)\in A_T} d(i,j)) \rfloor$ complete arcs of A_T; we assume that this is at least 1 complete arc, so $b \geq 1$. As at least $\min\{\mu, m\}$ new targets can be added to the tour (fuelwise), this implies that either a new target can be added to each arc of the tour, or that there are less new targets than existing targets. Then at least $\min\{b, \lceil \frac{1}{2}\mu \rceil\}$ new targets can be added to the online tour, so $|T_A| \geq \min\{b, \lceil \frac{1}{2}\mu \rceil\}$. As at most $\min\{\mu, m\}$ targets can be added in the offline tour, the performance ratio for this special case is: $\frac{|I_A|}{|I^*|} = \frac{m+\min\{b, \lceil \frac{1}{2}\mu \rceil\}}{m+\min\{m,\mu\}}$.
This is a better performance ratio than the general competitive ratio for the Insert algorithm. First of all: $m + \min\{m, \mu\} \leq m + \mu$, so $\frac{1}{m+\min\{m,\mu\}} \geq \frac{1}{m+\mu}$. Secondly, as $\min\{b, \lceil \frac{1}{2}\mu \rceil\} > 0$, $m + \min\{b, \lceil \frac{1}{2}\mu \rceil\} > m$. Therefore, $\frac{m+\min\{b, \lceil \frac{1}{2}\mu \rceil\}}{m+\min\{m,\mu\}} > \frac{m}{m+\mu}$.
Hence, instances of the Insert algorithm with a better performance ratio than the competitive ratio of $\frac{m}{m+\mu}$ do exist.
See figure 2 for an example.

4.2 Halfway-reoptimization algorithm

The second algorithm is, just like before, based on the original optimal tour on m targets, T_m. We also assume that a total of μ new targets will appear over time period C. The vehicle will travel exactly half of the original tour, and then a reoptimization takes place. The choice for waiting for half the tour length before reoptimization is a slightly arbitrary one, but can be explained. If we would wait a longer period, for instance until $\frac{3}{4}$ of the tour has passed, there may be more targets to add to the tour, but less possibilities to actually add the targets to the tour, as the remaining fuel capacity is smaller. If the reoptimization would take place after a shorter time period, for instance at $\frac{1}{3}$ of the tourlength, the remaining fuel capacity would be larger, so there would be more possibilities to add new targets to the tour, but there will probably be less new targets. Waiting for half the tourlength therefore is a good compromise between remaining fuel capacity and number of new targets.
Let L_{T_m} be the duration of the original tour T_m, then at time $t = \frac{1}{2}L_{T_m}$, μ' new targets may have appeared, for some $0 \leq \mu' \leq \mu$ (and $\mu' \in$). Then at time $\frac{1}{2}L_{T_m}$ we have a set $M(t)$ of new targets (with $|M| = \mu'$), a set of visited fixed targets $S(t)$ and a set of unvisited fixed targets $U(t)$. Note that at any time t, $|U(t)| + |S(t)| = n$.
In the reoptimization that will take place at time t, a new route will be determined for the vehicle that starts at the current location on the tour, ends at 0, that visits at least

$m - |S|$ targets from the set $M \cup U$ and that requires at most $C - \frac{1}{2}L_{T_m}$ fuel.
Due to the fact that only one reoptimization step takes place, approximately halfway during the tour, we call this the 'Halfway reoptimization algorithm' (HRA).For this algorithm, we cannot find a better competitive ratio than $\frac{m}{m+\mu}$: as neither the locations nor the release times of the new targets are known, we are unable to say anything about the performance of the halfway-reoptimization algorithm, except that it is guaranteed that the vehicle will visit at least m targets in the online tour. A similar argument can be found for the performance of the offline algorithm: by Theorem 1 we know that the vehicle cannot visit more than $m + \mu$ targets in the offline tour. Therefore, the competitive ratio[3] for this algorithm is: $\frac{m}{m+\mu}$.

This algorithm is based on reoptimizing half of the tour, on a set of targets that can turn out to be as large as the original set of targets. This new problem could be solved like the regular OP, but as the OP is NP-hard, it may take up to exponential time. The computation time for the reoptimization step may therefore be exponentially large. This implies that it might be necessary to use a heuristic method instead of an exact solution method and that the found solution may be suboptimal.



(a) Original tour T of length L_T

(b) New targets that appear before $\frac{1}{2}L_T$ (*red dots: new targets. Red line:* $\frac{1}{2}L_T$)

(c) Reoptimized tour T'

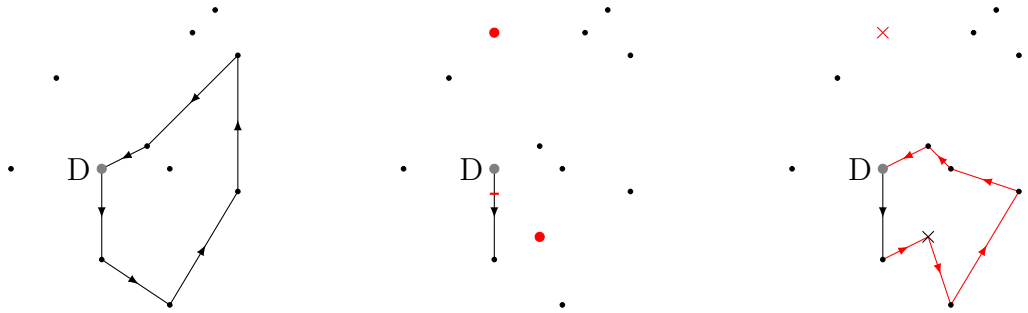
Figure 3: Reoptimization of a tour according to the HRA

4.3 Repeated-reoptimization algorithm

The third algorithm is called 'Repeated-reoptimization algorithm' (RRA) and is again based on the a-priori tour on the n fixed targets. The idea behind the algorithm is that when some new targets have appeared, the tour is reoptimized, in order to add new targets to the tour when possible. This reoptimization step may be repeated several times.
Let's first define the algorithm:
We are given an a-priori optimal tour T_m on the set of n fixed targets, that consists of m targets. We define U as the set of unvisited targets, S as the set of visited targets and T as the current tour. Note that at the start of the tour, set U consists of all fixed targets, so $U = \{1, \ldots, n\}$, set S is empty, so $S = \emptyset$, and $T = T_m$.
The vehicle will start with tour T_m and will visit at least the first original target. The

[3]Note that this does not imply that this bound is attained by one of the instances.

moment that a new target v appears it will be added to set U and a reoptimization step will be set in motion: tours will be found that start at the current next target i, end at the depot 0 and that visit as many of the targets from set $U\backslash\{i\}$ as possible. Such a new tour T' is called feasible if it meets two requirements. First of all, the number of targets on the new tour must be at least as large as the number of targets on the current tour, so $|T'| + |S| \geq |T| \geq m$. Secondly, the fuel required for the current tour up to target i plus the fuel required for the new tour does not exceed fuel capacity C.



(a) Initial tour

(b) Two new targets appear (*red dots: new targets. Red line: moment of appearance*)

(c) Updated tour

Figure 4: Example of the Repeated-reoptimization algorithm

The reoptimization step continues until either the optimal solution is found or the vehicle reaches target i. In the former case, the vehicle continues with the optimal tour if it is feasible. In the latter case, the vehicle continues with the best found tour, if feasible. This best tour is defined as the tour with the highest objective value, and when there are multiple tours with the same objective value, the tour that also has the lowest fuel requirements. In both cases: if the chosen tour is not feasible, the current tour is resumed. This process is repeated for each new target, until the vehicle reaches the depot (0).
Note that it might happen that a new target appears whilst a reoptimization step is taking place, then that target will be ignored for the time being. As soon as the vehicle reaches the first planned target i and either continues with a new tour or resumes the current tour, all new 'ignored' targets will be added to set U and the next reoptimization step will start. An example of the algorithm is shown in figure 4.

The offline problem can be modeled as an OPTW and solved with an exact algorithm, as we mentioned earlier in this chapter. This will yield optimal solution T^* that visits $|T^*|$ of the $n + \mu$ targets in L_{T^*} time.
It's easy to see that the online tour will contain at least m targets, up to $m + \mu$ targets. The offline tour can contain at most $m + \mu$ targets, by the same argument that was presented in the proof of Theorem 1. This implies that for this algorithm the same competitive ratio holds: $r_c = \sup\left\{r \,\middle|\, r \leq \frac{|T_A|}{|T^*|},\ \forall \text{ instances } I\right\} = \frac{m}{m+\mu}$. Note that this bound is not necessarily attained by an instance.

In case of this algorithm, it may occur that the tour will be reoptimized for every new target. The target set available for reoptimization may contain up to $m - 1 + \mu$ targets, so there may be μ steps of finding a new optimal tour along at most $m - 1 + \mu$ targets.

Finding such a tour may take exponential time, so the computational time for this algorithm is likely to be very large. Since reoptimization takes place during the flight and there is a limited fuel capacity, this might imply that in some of the optimization steps a suboptimal tour is chosen; that is, a tour that has the best objective value so far.

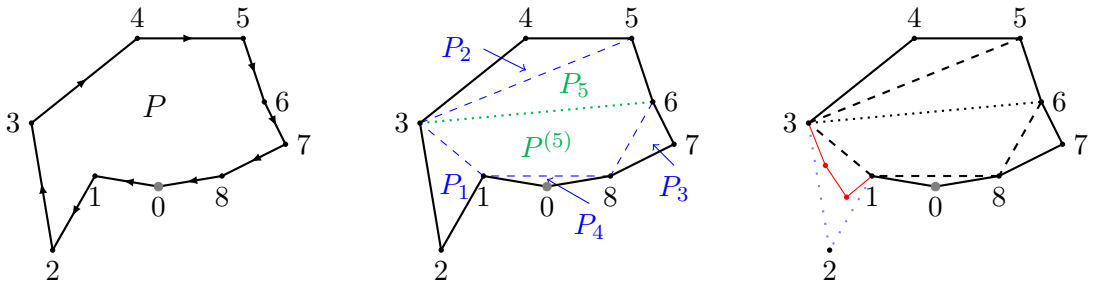
4.4 Internal target algorithm

Let T_m again be the tour along m targets that can be found when solving the UAV-MPP on the n fixed targets to optimality, and A_T the set of arcs on this tour. When we take these m targets and the depot, and take the $m+1$ arcs that the tour travels along, we get a simple[4] polygon with $m+1$ vertices and $m+1$ edges. When this polytope is subdivided into non-overlapping triangles on sets of three vertices, we create target sets in which the targets lie relatively close to each other, compared to the whole target set. Note that those triangles are allowed to have vertices or edges in common. Also, a number of new targets may appear in each triangle. One such triangle, together with the new targets that will appear inside it, can be seen as a subproblem of the UAV-MPP, on which the tour can be reoptimized. Since such target sets are quite small, each reoptimization step will take little computation time. The algorithm is called the 'Internal target algorithm' (ITA) as is only concerns itself with new targets that appear inside the fore mentioned triangles.
Let P be the polytope on the $m+1$ vertices of the tour. Before polytope P is subdivided into triangles, the vertices on the boundary of P are renumbered, such that 1 is the first target on tour T_m, 2 is the second target on the tour, etc. Then the last renumbered target is m. The set of edges of the polytope A_P can now be defined as the set of edges between adjacent pairs of vertices of P: $A_P = \{(0,1),(1,2),\ldots,(m-1,m),(m,0)\}$. The tour will be updated to $T'_m = (0,1,2,\ldots,m)$.

The actual subdivision of P is based on the fact that P is very likely to be a concave polygon. The process of subdividing the polygon is based on cutting off triangles that 'cause' non-convexity of P. By causing non-convexity we mean for instance that there is a triangle of three vertices $(i, i+1, i+2)$ of the polygon, such that either line $(i-1, i+1)$ or line $(i+1, i+3)$ does not lie inside P.
The process is as follows. Starting at vertex 0, the vertex with the lowest possible index, let $i, i+2$ be the first pair of vertices such that line $(i, i+2)$ lies inside P. Then the first triangle is based on vertices $i, i+1, i+2$, creating remaining polygon $P' = P \backslash \{i+1\}$. This process is repeated for k times, until vertex 0 is reached again. Then there is a remaining polygon $P^{(k)}$. If $P^{(k)}$ still has edges in common with the P, then the process will be continued for this polygon, again starting at the vertex with the lowest index possible. Note that triangles that are now created will not have two adjacent pairs of adjacent vertices as edges. After s steps, for some $\lceil \frac{m}{2} \rceil \leq s \leq m$, all edges of P will be an edge of exactly one of the s triangles, creating remaining polygon $P^{(s)}$ that has no edges in common with original polygon P. We will not divide this polygon further, as the subsequent triangles will have no edges in common with P. Tours on those triangles will either involve cutting off part of the original tour or returning to a target that has already been visited, and are therefore less interesting.

[4] A polytope is called simple when it has no self-intersecting edges

An example of how this subdivision works is shown in figure 5. The original polygon is given in figure 5a. Then the first possible triangle is $P_1 = \{1, 2, 3\}$, creating remaining polygon $P' = \{0, 1, 3, 4, 5, 6, 7, 8\}$. The subsequent triangles are $P_2 = \{3, 4, 5\}$, $P_3 = \{6, 7, 8\}$ and $P_4 = \{8, 0, 1\}$. As $P^{(4)} = \{1, 3, 5, 6, 8\}$ still has an edge in common with P, namely edge $(5, 6)$, the process of creating triangles is repeated. Starting from the vertex with the lowest index, vertex 1, triangle $\{3, 5, 6\}$ is the first possible triangle on three adjacent vertices from set $P^{(4)}$ that has an edge in common with P, so $P_5 = \{3, 5, 6\}$. Then as $P^{(5)} = \{1, 3, 6, 8\}$ has no more edges in common with P, all necessary triangles have been found. Figure 5b shows all constructed triangles.



(a) Initial polytope P of tour T_m.

(b) Subdivision of P (*into triangles* $P_1, \ldots, P_5$)

(c) Reoptimized tour on P_1 (*red dots: new targets, appeared early enough for reoptimization on* P_1. *Red lines: new tour arcs*)

Figure 5: Division of polygon P into triangles and first reoptimizion of the tour

If P is convex, then the subdivision will be quite straightforward, and mostly based on pairs of adjacent edges of P. The majority of the triangles will have a set $\{i,\ i+1,\ i+2\}$ as vertices and set $\{(i, i+1),\ (i+1, i+2),\ (i, i+2)\}$ as edges. If m is even, the penultimate triangle will be based on vertices $m-2,\ m-1,\ m$. In that case, the last triangle will be based on vertices $m,\ 0,\ 2$ and will only have edge $(m, 0)$ in common with the edge set of P. The remaining polygon will then be $P^{((m+2)/2)} = \{0, m+2, m+4, \ldots, m-1\}$, so there will be $\frac{m+2}{2}$ triangles. In case m is odd, the last triangle will be based on vertices $m-1,\ m, 0$ and all $\frac{m+1}{2}$ triangles will have exactly two edges in common with the edge set of P. Note that the remaining polygon $P^{((m+1)/2)} = \{1, 3, \ldots, m-1\}$ has no edges in common with P.

Before the tour can be reoptimized, the triangles have to be sorted, based on the order of their appearance on the original tour. In case of the example, the visiting order of the triangles is: P_1, P_2, P_5, P_3, P_4. The reason for visiting P_4 last, instead of first - even though it contains edge $(0, 1)$, which is the first edge of the tour - will be given later on in this section.
As before, an unknown number μ of targets will appear in the feasible region, over a time-period of C; it is unknown where and when the targets will appear. This implies either all new targets appear inside of P, or that some of the new targets appear outside of P. Let $\mu' \leq \mu$ be the number of targets that appear inside P. Let there be k subdivisions of P, then each of the μ' targets will appear in either one of the k triangles $P_1, \ldots, P_k$ or in remaining polygon $P^{(k)}$.

The algorithm will be as follows: the tour will start as the planned tour T'_m with length L_{T_m}. When the tour is near the first vertex of some triangle P_i and new targets have appeared in that triangle, a reoptimization will take place on that triangle. The triangle has either a single edge or two adjacent edges in common with A_P (the set of edges of polytope P). When following the direction of the tour T'_m, let t be the first vertex on the (first) edge and u the last vertex on the (last) edge. Let the total length of these edge(s) of P_i be denoted by L_{P_i}.
In the reoptimization step, a path is found from the vertex t to vertex u that visits at least as many targets within P_i as the original tour on this triangle, and that uses at most $L_{P_i} + C - L_{T_m}$ fuel. An example of this can be seen in figure 5c, where 2 new targets appear in the first subpolytope, and a new tour is constructed that visits both new targets, but cuts off target 2. This step is repeated for each triangle.
It might occur that the vehicle reaches the first vertex of a triangle, but no new targets have appeared inside it yet. Then the vehicle will continue the original tour on that triangle. When new targets appear while the vehicle is travelling the triangle, and the vehicle is not on the last fixed edge of that triangle, the tour on that triangle will be reoptimized before the vehicle reaches the second vertex.
A special rule applies for the triangle containing vertex 0, on which 0 is either the first or the second vertex. In that case, this triangle will very likely contain the first arc of the tour. For this arc, no reoptimization can take place, as reoptimization for any arc has to be finished before the vehicle reaches that arc. If the triangle has two edges in common with P, then some reoptimization is possible. This is the case, if either 0 is the first vertex of the first triangle, and new targets have appeared inside the triangle whilst the vehicle is still traveling the first arc, or if 0 is the second vertex of the last triangle and new targets have appeared inside the triangle before the vehicle has reached its first vertex. Note that 'first triangle' and 'last triangle' do not necessarily refer to triangles P_1 and P_k, but to the order of the triangles along the original tour.

For this algorithm a performance ratio has been found as well, in a similar way as for the previous algorithms. The original tour contains m targets. When a reoptimization step takes place on one of the subpolytopes, some of the original targets may be left out of the tour, but the algorithm requires that during that step at least as many new targets are added to the tour. Hence, the number of targets in the reoptimized tour on each subpolytope is as least as large as the number of targets in the original tour on that subpolytope, so the number of visited targets in the total reoptimized tour is at least m.
For the offline case, the same argument holds as in the case of the previous algorithms: when it is modeled as an OPTW, it can be solved to optimality with an exact algorithm. Therefore, by Theorem 1, at most $m + \mu'$ targets can be visited in the offline case. This implies that for the internal point algorithm, the competitive ratio is $c_r = \sup\left\{ r \,\middle|\, \frac{|T_A|}{|T^*|} \geq r,\ \forall \text{ instances } I \right\} = \frac{m}{m+\mu'}$.
Note that this bound is not necessarily tight, as there may not be an instance in which all μ' new targets appear inside polytope P in such a way that they can be added to the offline tour, but not to the online tour.
The computational time for this algorithm is relatively small. For each triangle a small version of the OP has to be solved; this can be done by an exact algorithm in polynomial time.

5 Performance of the algorithms

5.1 Description of the simulation settings

In the previous chapter we described theoretical results for each of the four algorithms. While such results may be adequate in some cases, it has become clear that the found results for our algorithms seem to be quite unspecific, as we found similar bounds for each of the algorithms. This can be explained by the fact that there were too many uncertainties in the general instance. In order to provide more insight into the performance of the algorithms, measured by both the objective value and the computational time, we expanded the research by applying the algorithms to testcase instances, by which empirical data have been obtained. The used testcase instance was based on benchmark instances for the OP that were presented in [42], as real data were unavailable. This instance consists of a depot and 19 targets, located in a 15×15-area, that has $(0, 5)$ as the lower left corner and $(15, 20)$ as the upper right corner. All locations i, $0 \leq i \leq 19$, are further specified with a set of coordinates $(x(i), y(i))$ in the Euclidean plane and a score $s(i)$. Except for the depot, which has score 0, these scores are multiples of five between 10 and 50. The exact specifications of the instance can be found in table 2 and a graphical representation of the target locations is given in figure 6.

Table 2: Target instance, as used for all simulations *(i denotes the index of a target, $(x(i), y(i))$ denote the coordinates of target i and $s(i)$ denotes its score)*

node i	$x(i)$	$y(i)$	$s(i)$	node i	$x(i)$	$y(i)$	$s(i)$
0	4.6	7.1	0	10	6.3	7.9	15
1	5.7	11.4	20	11	5.4	8.2	10
2	4.4	12.3	20	12	5.8	6.8	10
3	2.8	14.3	30	13	6.7	5.8	25
4	3.2	10.3	15	14	13.8	13.1	40
5	3.5	9.8	15	15	14.1	14.2	40
6	4.4	8.4	10	16	11.2	13.6	30
7	7.8	11.0	20	17	9.7	16.4	30
8	8.8	9.8	20	18	9.5	18.8	50
9	7.7	8.2	20	19	4.7	16.8	30

In addition to the set of known targets, sets of new targets were necessary and have been generated according to the following requirements. First of all, all new targets will appear inside the fore mentioned 15×15-area. Secondly, all new targets will appear during the flight of the vehicle, that is, target that appear either before or after the flight of the UAV are disregarded. As all the specifications of the new targets were unknown, we used models to find these specifications. First of all, it is easy to see that the arrival of the targets can be described as a counting process, as all of the four following requirements are met with: the number of appeared targets is at least 0 at each time $t \geq 0$; the number of targets is an integer; targets do not disappear, i.e. $N(s) \leq N(t)$ for any pair $s, t \geq 0$ such that $s < t$; and finally for any pair of moments $0 \leq s < t$, $N(t) - N(s)$ equals the number of targets that have appeared in the interval $(s, t]$. Furthermore, we assumed that the time between any two consecutive appearances is distributed according to an

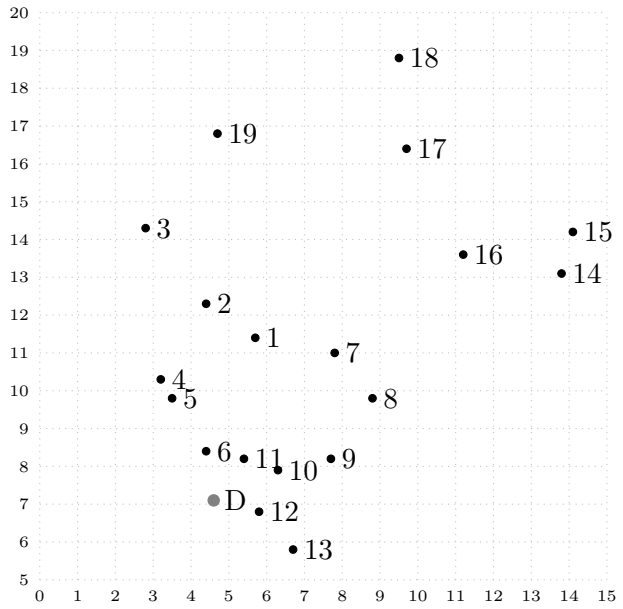


Figure 6: Location (in the Euclidean plane) of the nodes of the Tsiligirides-instance

exponential distribution with some given mean $\frac{1}{\lambda}$ and that these interarrival times are independent of each other. The arrival of new targets can therefore be described by a renewal process. See [34] for more elaborate description of these stochastic processes.
Let, for some $i > 0$, X_i be the interarrival time between the $(i-1)^{st}$ and the i^{th} new target, and $X_i \sim exp(\lambda)$. The arrival times of the new targets can be computed when the interarrival times are known, but for the interarrival times we need to know how many new targets will arrive. The total number of new targets at some time $t > 0$ can be described by Poisson random variable $N(t)$, which has rate per time unit λ. In this case, the time unit is defined to be equal to the fuel capacity, as we are only interested in targets that can be added to the tour, i.e. targets that appear before the time limit of the flight has elapsed.
We generated independent $(0,1)$-uniform random variables $U_1, U_2, \ldots$, such that

$$N + 1 = \min \left\{ n \left| \prod_{i=1}^{n} U_i < e^{-\lambda} \right. \right\}.$$

Then N is the number of targets that have arrived before t. Using this value N, we determined the arrival times of the targets, by generating a new set of random numbers $U_1, \ldots, U_N \sim U(0,1)$ and multiplying them by t. The values $tU_1, \ldots, tU_N$ are the interarrival times of the N new targets. The arrival times $t_1, \ldots, t_N$ can then be determined by adding the interarrival times of all targets that have appeared at that moment: $t_i = \sum_{j=0}^{i} tU_j$, for $i \in \{1, \ldots, N\}$.
In addition to the arrival times, coordinates and scores have been determined for each of the N new targets. These are all based on $(0,1)$-uniform distributions as well. Let $i \leq N$ be the index of a new target, then its coordinates $(x(i), y(i))$ are determined as follows: let $(X_i, Y_i) \sim U(0,1) \times U(0,1)$, then $x(i) = 15X_i$ and $y(i) = 5 + 15Y_i$. The coordinates were rounded to one decimal point, just like the coordinates of the given targets.
For each of the new targets we have generated scores, similar to those of the given targets.

Since the new targets are expected to have slightly higher information values, the scores ranged from 20 to 60 and were determined as follows: let $Z_i \sim U(0,1)$, $\forall i \leq N$, then $s(i) = 20 + 40Z_i$. By rounding the $s(i)$ to the nearest 5-tuple, we obtained the score for new target i.

All used distances between pairs of locations were based on, but not necessarily equal to, the Euclidean metric. We assumed that at each target location some time is required for locating the actual target and gathering imagery, and have therefore used a so-called recording time equal to 2 fuel units. Note that this recording time is only necessary at targets and not at the depot. In order to make the computations easier, we have added this recording time to the distance between two locations. We assumed that the recording takes place after the vehicle has flown from one location to the next, so for any pair $i \neq j$, $0 \leq i, j \leq N + 19$ with $j \neq 0$, the distance $d(i, j)$ is computed as follows: $d(i,j) = \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2} + 2$. When $j = 0$, recording time is not necessary, so in those cases $d(i,j) = \sqrt{(x(i) - x(j))^2 + (y(i) - y(j))^2}$. Note that $N + 19$ refers to the indices of all locations: there were 20 given locations, with indices $0, \ldots, 19$, and N new targets, so $N + 19$ is the index of the new target that appears last.
The fuel capacity was set to 65.0 in each of the simulations. We have considered multiple rates for the arrival of the targets: in any testcase we set the rate to a value of either 3, 5, 10 or 20.

In addition to the testcases mentioned so far, we had a second, similar set of testcases, in which all scores, both those of the given targets and those of the new targets, were set to 1. This means that there were two types of scores, four options for the arrival rate of the new targets and four algorithms. This results in 32 different testcases; an overview of the cases is given in table 3. We generated four sets of 10.000 new target instances, i.e. one set of 10.000 instances for each of the four arrival rates, and used these as new target instances for the testcases, as to be able to compare the results of similar testcases.
The models have all been implemented in Eclipse 3.7.2, combined with Cplex 12.2 for the optimization steps.

Table 3: Overview of the settings of all 32 testcases (*per algorithm there are 2 scoretypes; per scoretype there are 4 different rates*)

Algorithm	IA / HRA / RRA / ITA							
Score	1				varying			
Rate	3	5	10	20	3	5	10	20

5.2 Evaluation of the results

As mentioned in the previous paragraph, we had two sets of 16 testcases. For both testcases a solution to the basic problem with only the given targets was found first; these basic solutions, combined with the original data, were used as input for any of the instances in any of the testcases. The basic solution for the target set with so-called 'full scores', i.e. the scores ranging from 10 to 50, was a tour with an objective value of 360, that required 64,94 fuel units and visited 13 targets along the way. The basic solution

for the target set with score 1 had an objective value of 15 and had a length of 63,44 fuel units. See also figure 7 for a graphical representation. Note that these tours are quite different: in the case where all scores are set to 1, the resulting tour visits two targets more than the tour for the target set with full scores. Such a result is not quite unexpected when looking at the data: targets 3 and 14 through 19 may have higher scores, but also lie relatively far from the depot. When only the number of visited targets, instead of the values of those targets, is important - as it is in this case - the fore mentioned targets are less likely to be visited.

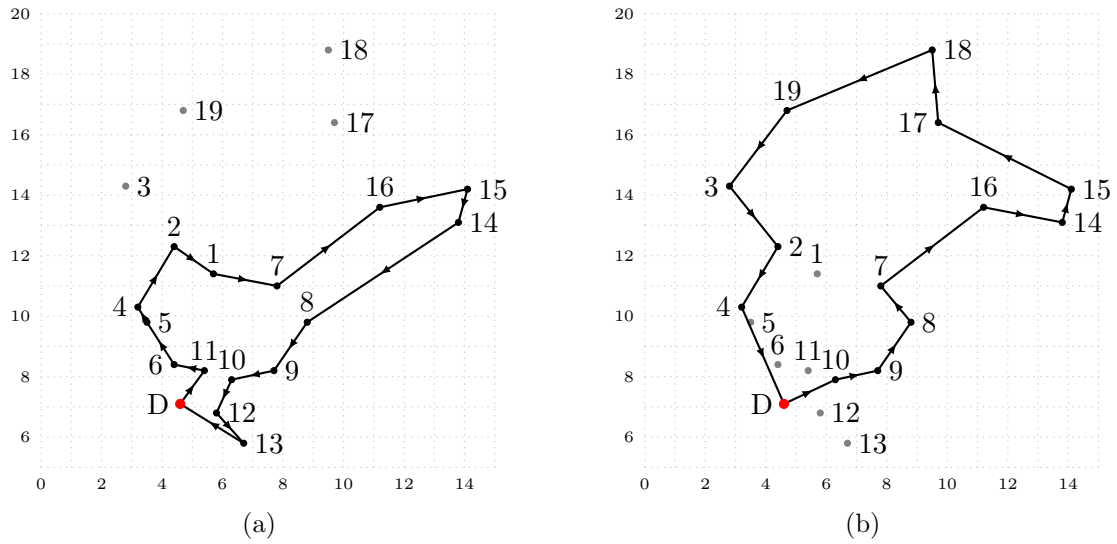


Figure 7: Basic solutions for the cases with score 1 (a) and varying scores (b), shown in the Euclidean plane

5.2.1 First case: equal scores

The testcases in which all scores were set to 1 were meant first and foremost for the determination of performance ratios of the algorithms. They can also be used for different analyses, however. First of all, the results show for each of the algorithms whether it is possible to add targets to the existing tour under the given conditions, and if so, how many. As all four methods require a solution to be at least as good as the basic solution, each of the found tours contains at least 15 targets. The methods result in addition of 0 up to 5 targets to the tour. The results of the simulations can be seen in table 4 in more detail. Note that these results are determined by taking the averages of all 10000 outcomes. The average fuel consumptions, that are mentioned in the fourth column, are based on the fuel consumptions of the updated tours. The last column contains average computation times in milliseconds. More detailed results can be found in appendices A and B.

The fact that the average objective values are equal to the average number of visited targets is easy to explain: the value of each of the targets is 1, so the objective value is equal to the number of visited targets for each tour. Hence, the average objective value is equal to the average number of visited targets.

Table 4: Results of the testcases with score 1. (*Results in columns 3 up to 7 are averages over 10000 instances. Fuel consumption of denotes the fuel consumption of a reoptimized tour; computation time is given in ms*)

Algorithm	Rate	Objective value	Fuel consumption	Number of visited targets	Computation time
IA	3	15	63.448	15	0.0
IA	5	15	63.448	15	0.0
IA	10	15	63.448	15	0.1
IA	20	15	63.448	15	0.2
HRA	3	15.035	59.218	15.035	18.9
HRA	5	15.085	58.448	15.085	24.4
HRA	10	15.253	58.116	15.253	33.9
HRA	20	15.591	58.326	15.591	56.0
RRA	3	15.098	63.611	15.098	710.6
RRA	5	15.227	63.690	15.227	1153.3
RRA	10	15.617	63.802	15.617	2587.1
RRA	20	16.291	63.963	16.291	7633.6
ITA	3	15.000	63.437	15.000	0.3
ITA	5	15.000	63.433	15.000	0.5
ITA	10	15.000	63.418	15.000	0.9
ITA	20	15.000	63.393	15.000	2.1

Table 5: Results of the testcases with varying scores (*Results in columns 3 up to 7 are averages over 10000 instances. Fuel consumption denotes the fuel consumption of reoptimized tour; computation time is given in ms*)

Algorithm	Rate	Objective value	Fuel consumption	Number of visited targets	Computation time
IA	3	360	64.937	13	0.0
IA	5	360	64.937	13	0.0
IA	10	360	64.937	13	0.0
IA	20	360	64.937	13	0.1
HRA	3	372.950	59.213	12.856	29.2
HRA	5	380.782	58.179	12.813	41.1
HRA	10	400.301	57.585	12.880	70.7
HRA	20	434.425	57.534	13.231	187.2
RRA	3	383.677	64.397	12.821	1458.6
RRA	5	397.172	64.074	12.817	1709.1
RRA	10	427.196	64.434	12.946	4118.5
RRA	20	475.109	64.438	13.400	14251.9
ITA	3	361.245	64.907	13.001	0.5
ITA	5	362.107	64.885	13.001	1.0
ITA	10	364.101	64.838	13.002	2.1
ITA	20	368.312	64.745	13.011	5.7

5.2.2 Second case: varying scores

An overview of the results of the second set of 16 testcases, with scores ranging from 10 to 50 for the given targets and ranging from 20 to 60 for the new targets, can be found in table 5. Note that the fuel consumptions in this table are based on the amounts of fuel required for the reoptimized tours, just as in table 4.
A first impression is that the results of either case are quite similar, that is, when the two sets of results of an algorithm are compared to each other.
The third algorithm (RRA), that reoptimizes the tour whenever a new target appears, seems to yield the best results overall: its objective values range from 510 to 675. This was to be expected, as the remaining part[5] of the tour is reoptimized each time a new target arrives and/or as soon as the current reoptimization is finished, and only solutions are allowed that have equal or better objective values. The algorithm therefore allows tours that are quite different from the basic solution. In instances where the arrival rate is high, the new targets lie relatively close to the depot and their scores are relatively high, when compared to the scores of the targets in the remaining part of the tour, the algorithm might produce tours that have relatively high objective values.
The second algorithm (HRA) has relatively good results for the same reasons as the third algorithm. The average objective values for the instances with either rate 5, rate 10 or rate 20 are relatively high in case of the third algorithm, when compared to the results of the second algorithm. This is easy to explain: in case of the third algorithms, many reoptimizations may take place, since many new targets arrive, whilst the second algorithm allows only one reoptimization step. When looking at the average results for instances with rate 3, the difference in results is a lot smaller. This could be explained by the fact that there were only a few new targets, so also only a few reoptimization steps were possible.
The Internal Target Algorithm also finds better solutions than the original solution, though those solutions are not as good as those found by either the HRA or the RRA. This is due to the structure of the algorithm and the fact that the remaining capacity of the basic solution does not allow addition of new targets to the tour at first. This means that, in any of the instances, no more than two extra targets have been added to a tour.
The Insert algorithm seems to be performing the worst in either case: it does not produce any improved solutions at all. This was to be expected: the IA only allows addition of new targets to the current tour, but for each additional target 2 extra fuel units are necessary for recording. Since the length of original tour is more than 63 fuel units already, this is not possible, even if the remaining fuel capacity would allow visiting a new target. The current settings of the testcases are therefore unsuitable for the IA.

With the possible application of the algorithms to real life situations in mind, there are two more factors that should be taken into account. First of all, the computation time of the algorithms is important. An algorithm that finds tours with very high objective values[6], may not be so interesting for further research when it also takes a relatively long time before the tours are found. We therefore have to evaluate the durations of the

[5]By 'remaining part' we imply the part of the current tour that starts at the next target that is to be visited, i.e. the target that the vehicle is currently traveling to.

[6]I.e. when the results of specific instances are compared with each other

simulations. For this, we measured the time required by the reoptimization process in each of the algorithms.
As expected, based on the complexity analysis in Chapter 3, the simulations for both the first (IA) and the fourth (ITA) algorithm took very little time, even for a relatively large set of new targets. The largest computation time was 121 ms; the average computation time was less than 6 ms.
For both the second (HRA) and the third (RRA) algorithm, more computation time was required, as can be seen in tables B.1 and B.2 in appendix B. For both algorithms we found that the simulations with so-called full scores took more time than the simulations were all targets had score 1. For the HRA with score 1, the average computation time ranged from 19 ms, in case of the instances with rate 3, to 56 ms, in case of the instances with rate 20. For the HRA with full scores, the average computation time ranged from 29 ms for rate 3 to 187 ms for rate 20.
Simulations of the RRA with score 1 took 710 ms to 7633 ms on average, for rate 3 and rate 20 respectively; simulations for the RRA with full scores took 1459 ms to 14252 ms on average, for rate 3 and rate 20, respectively.
Secondly, the RRA may yield tours that differ greatly from the basic solution, or even from tours found in previous reoptimization steps. In real life, for instance in military situations, this may be an undesirable result and it might make the RRA less interesting.
When comparing all results, the repeated reoptimization algorithm (RRA) seems to yield the best tours; the theoretical performance of the algorithm is clearly the best. But as not only the theoretical performance is important, we need to take the computation times into account as well. As can be seen in appendix B, the maximal measured computation time was about 143 seconds, for an instance of RRA with full scores and arrival rate 20. Note that this is the computation time for the entire instance. That particular instance contained 21 new targets, so the computation time per reoptimization step was about 7 seconds on average. While a reoptimization time of 7 seconds may seem quite acceptable in real-life situations, we would like to remark that it could also occur that in a worst-case scenario just one of the steps took 143 seconds. In real life situations there may not be much time available for reoptimization, so such processes would have to be stopped prematurely, which could result in less-than-optimal solutions. Both this fact, and the fact that the optimal tours found by the reoptimization steps in the RRA might differ greatly from the original tour, lead to the conclusion that the RRA does not seem to perform well in practice.
The HRA, on the other hand, does. Its computation times are relatively small, the reoptimized tour cannot differ too greatly from the original one, as the first half overlaps, and the reoptimized tours have better objective values in more than half the cases. The HRA therefore seems more a suitable algorithm.

5.2.3 Offline solutions and competitive ratios

For determining the performance ratios, we needed to determine the offline solutions first. In this case, only four sets of 10000 solutions were necessary: one set per arrival rate. Finding such a solution is similar to finding online solutions, but in this case, all data on the new targets were known in advance. The offline variant of the UAV-MPP with a changing set of targets can be therefore modeled as an OPTW. We used the programming

formulation as mentioned in Section 3.3, but made some minor changes to it to improve its efficiency.
Let set W be the set of new targets. We added a dummy depot to set V^+, that coincides with the original depot, for programming reasons. The dummy depot has index $l = |V| + |W|+1$. Then $V^{++} = V \cup \{0\} \cup \{l\}$ denotes the set of all given locations, including both the actual depot and the dummy depot. Note that not all arcs (i, j) with $i, j \in V^{++} \cup W,\ i \neq j$ are used. The set of arcs is constructed as follows: $A = \{(0, i) | i \in V \cup W\} \cup \{(i, j) | i, j \in V \cup W\ \ i \neq j\} \cup \{(i, l) | i \in V \cup W\}$. As the set of locations has been expanded, we have adapted most of the constraints:

$$\begin{aligned}
\text{(OPTW')} \qquad & \max \sum_{i \in V} x_i v(i), & & \\
& \text{s.t.} \sum_{(i,j) \in A} d_{ij} x_{ij} \leq C, & & (23)\\
& x_0 = \sum_{i \in V} x_{0i} = 1, & & (24)\\
& x_l = \sum_{i \in V} x_{il} = 1, & & (25)\\
x_j = \sum_{i \in V^+ \setminus \{j\}} x_{ij} = & \sum_{i \in V \cup \{l\} \setminus \{j\}} x_{ji} \leq 1, & \forall j \in V \cup W & (26)\\
& t_i + d_{ij} - t_j \leq M(1 - x_{ij}), & \forall i, j \in V \cup W & (27)\\
& x_i r_i \leq t_i, & \forall i \in V^{++} \cup W & (28)\\
& t_i \leq x_i d_i, & \forall i \in V^{++} \cup W & (29)\\
& x_{ij} \in \{0, 1\}, & \forall (i, j) \in E &
\end{aligned}$$

Note that variables d_{ij}, $v(i)$, r_i and d_i, constants C and M and decision variables x_i and x_{ij} are similar to the ones used in the original OPTW-model. Constraint (3) have been split up into two new ones: (24) for the original depot and (25) for the dummy depot. Both constraints now take both sets of targets into account, instead of only set V. This last change has been made to all constraints of the original OPTW that concerned itself with set V.
The changes to Constraints (28) and (29) were made mostly for computational reasons. This way, when the tour does not include location i, t_i is automatically set to 0. Note that both sets of constraints still contain only linear constraints, as all values of r_i and d_i are input parameters of the OPTW'-model. Also, we defined t_i as the time that the vehicle leaves location i and adapted the values of most of the r_i and d_i. Since the vehicle cannot leave a target earlier than the Euclidean distance between the depot and the target plus the recording time, $r_i = d_{0i} + 2.0,\ \forall i \in V$. The vehicle cannot leave a new target earlier than the moment of arrival of the new target plus recording time, so $r_i = t(i) + 2.0,\ \forall i \in W$, where $t(i)$ is the arrival time of new target i. The vehicle cannot leave the last target in the tour later than the fuel capacity minus the distance from that target back to the depot, so $d_i = C - d_{il},\ \forall i \in V \cup W$. The vehicle has to start at the depot at $t = 0$, so $r_0 = d_0 = 0.0$, and the vehicle cannot go from the depot directly to the depot, so $r_l = 2.0$ and $d_l = C$. The value of 2.0 refers to recording time. Note that

we deviated from the values $r_i,\ d_i, \forall i \in \{1, \ldots, 19\}$, as they were mentioned in Section 3.3. The reason for this is mainly a practical one: smaller time windows $[r_i, d_i]$ implies that there are less possibilities for t_i and therefore results in less computing time.

The value of constant M has some influence on the efficiency of the solution process as well. It is therefore important to pick a value that is not too high, but does allow any feasible combination of values for t_i and t_j. As $t_i - t_j \leq C,\ \forall (i,j) \in A$ and $d_{ij} \leq 15\sqrt{2} + 2, \forall i, j \in V^{++} \cup W, i \neq j$, M can be set to $C + 2 + 15\sqrt{2}$. Note that a slightly smaller M would suffice as well, as a combination of $t_i = 65.0,\ t_j = 0.0$ and $d_{ij} = 15\sqrt{2} + 2$ will not occur.

We solved the offline problem with 500 new target instances for each of the arrival rates, in order to determine the empirical performance ratios. In some cases, finding an exact solution proved to be rather difficult, which can be explained by the fact that all scores were equal, some of the sets of new targets were quite large and some the new targets were located close to each other or close to the given targets. In such a case, the solution of that particular instance was ignored if an exact solution had not been found after 10 minutes. Also, when the set of new targets is empty, no reoptimization takes place and the corresponding solution is trivial. Such instances were ignored as well. We therefore only used the solutions for the first 500 non-empty instances that could be solved in 10 minutes.
The performance ratios were determined per algorithm, per arrival rate, by comparing the objective value of the online solution and the objective value of the offline solution for each of the target instances. This means that, for instance, the solution of the HRA for the fifth set of new targets, generated according to an arrival rate of 10.0, was compared to the offline solution on the same set of new targets.
For each of the 16 combinations of type of algorithm and arrival rate, we determined the minimal performance ratio, the maximal performance ratio and the average performance ratio. The results can be found in tables 6 and 7.

Table 6: Empirical performance ratios for rate 3.0 (left) and rate 5.0 (right) (*Results refer to minimal found performance ratio, maximal found performance ratio and average of the found performance ratios*)

Algorithm	Min	Max	Average
IA	$\frac{15}{18}$	1	0.979
HRA	$\frac{15}{18}$	1	0.982
RRA	$\frac{15}{18}$	1	0.987
ITA	$\frac{15}{18}$	1	0.979

Algorithm	Min	Max	average
IA	$\frac{15}{18}$	1	0.957
HRA	$\frac{15}{18}$	1	0.962
RRA	$\frac{15}{17}$	1	0.972
ITA	$\frac{15}{18}$	1	0.957

For each of the algorithms and each of the rates, the maximum performance ratio was 1. While this seems like a good result, we would like to remark that there are cases where the offline solution does not contain more targets than the original solution of 15 targets. In these cases the online solution will have an equal objective value, as we allow only online solutions that have an objective value that is as least as large as the original objective value. This results in a performance ratio of 1, but does not say much about

Table 7: Empirical performance ratios for rate 10.0 (left) and rate 20.0 (right) (*Results refer to minimal found performance ratio, maximal found performance ratio and average of the found performance ratios*)

Algorithm	Min	Max	Average
IA	$\frac{15}{19}$	1	0.917
HRA	$\frac{15}{19}$	1	0.932
RRA	$\frac{15}{18}$	1	0.954
ITA	$\frac{15}{19}$	1	0.917

Algorithm	Min	Max	Average
IA	$\frac{15}{20}$	1	0.865
HRA	$\frac{15}{19}$	1	0.897
RRA	$\frac{15}{18}$	1	0.935
ITA	$\frac{15}{20}$	1	0.865

the performance of the online algorithm. In some cases, however, both the offline and the online solution have an equal objective value higher than 15; this has occurred for each of the rates for both the RRA and the HRA, so we may conclude that both algorithms perform optimal for some instances.

First of all, there may be circumstances under which the Internal Algorithm (IA) performs better than it has for the given circumstances. Two simple adaptations of the given settings, that may lead to better performance, are shorter recording times and adding some slack to the fuel capacity. By adding slack we mean that part of the fuel capacity will remain reserved for reoptimization purposes, and cannot be used for the initial tour. Another adaptation that may result in better performance involves an adaptation of the objective function. While the optimal solutions for either basic problem are tours with the highest objective value, their tours may not be of 'minimal' length, as the length of the tours are not taken into account in the objective function. There may be tours with equal objective values of shorter length; such a tour might prove to be more suitable as a basic solution, since it might create enough slack for adding one or more targets to the existing tour. When a penalty for fuel consumption is added to the objective function, the optimal tour will be optimal on both aspects. A slight disadvantage of such objective functions is that it will take more time to optimize the corresponding problems.

Another part of the research that might be interesting to investigate further, is an adaptation of the Internal Target algorithm. When a new target appears close to one of the arcs of the tour before the vehicle has reached that arc, the remaining fuel is such that the target could be added to the tour, but the target lies outside of the tour, instead of inside, this target will be ignored, even though it would lead to an improvement of the objective value. A way to correct this problem is not to look at triangles that lie 'inside' the tour, but find other areas in the vicinity of the arcs, in which we look for new targets.

6 A changing set of rates

In the previous chapters we described which solution strategies might be useful for solving the UAV-MPP with a changing set of targets. For these strategies we assumed that all new targets appear according to the same arrival rate. But in real-life situations this might not necessarily be true. The arrival rate might be higher in some reasons, due to presence or absence of habitation, for instance. For such situations, the found strategies might not be optimal.
In this part of the thesis we will determine whether the four found strategies can be successfully applied to instances with both a changing set of targets and varying arrival rates. We will start by developing a new strategy, designed for the instances where the new targets appear in one prespecified zone only.

6.1 Delay algorithm

For the new strategy, we assumed that all new targets appear in some zone that is located inside the area of interest, and that both the fact that all new targets appear in a certain area and the specific location of this area are known beforehand. The strategy is based on the idea of delaying the arrival of the vehicle at the specified new target zone, so that as many new targets can be visited as possible. Our way to model this is by use of the variables u_i, that were used for instance in Constraints (6) of the nominal OP.
These constraints were introduced in the paper on Integer Programming for the TSP, by Miller et al. [31] and can be seen as an alternative to the subtour elimination constraints that are more commonly known, i.e. those that were mentioned for instance in [4]. The commonly known constraints consist of two sets of constraints:

$$\sum_{e \in \delta(\{i\})} x_e = 2,\ i \in N \tag{30}$$

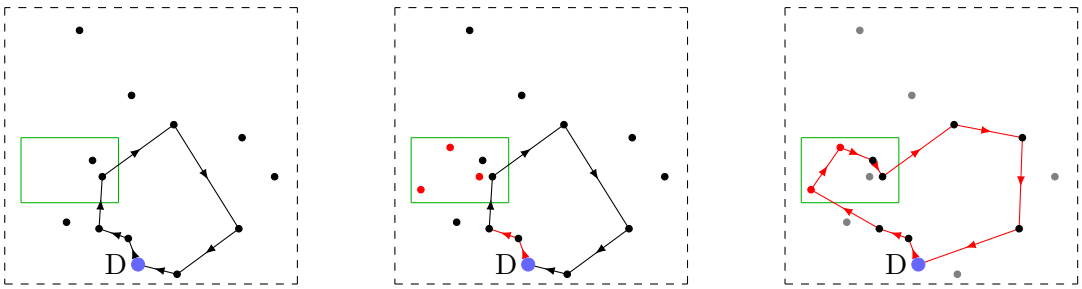
$$\sum_{e \in E(S)} x_e \leq |S| - 1,\ S \subset N,\ s \neq \emptyset, N \tag{31}$$

In these constraints, $x_e \in \{0, 1\}$ is a decision variable for each edge $e \in E$. E is the set of edges of some graph $G = (N, E)$, N is the set of nodes, S is a subset of nodes, $E(S) = \{(i, j) \in E | i, j \in S\}$ and $\delta(\{i\})$ denotes the set of edges incident to i. Constraints (30) ensure that exactly two edges are incident to each node; Constraints (31) ensure that for each subset of nodes S, there are at most $|S| - 1$ edges, so no so-called subtours are possible. Note that the first set of constraints are useless in case of the OP, as it is very likely in the OP that not all nodes are visited. While both these constraints are quite straightforward, they are not very efficient: let $|N| = n$, then (31) consists of $\sum_{k=1}^{n-1} \binom{n}{k} = -2 + \sum_{k=0}^{n} \binom{n}{k} = 2^n - 2 \sim O(2^n)$ constraints. The constraints introduced by Miller et al., here on after denoted as the MTZ-constraints, are based on the set of arcs:

$$u_i - u_j + 1 \leq |V|(1 - x_{ij}),\ \forall (i, j) \in A. \tag{32}$$

In these constraints V is the set of nodes, A is the set of arcs, the $x_{ij} \in \{0, 1\}$ are decision variables for the arcs and $1 \leq u_i \leq |V|,\ i \in \{1, 2, \ldots, |V|\}$. As there is such a constraint for each of the arcs in A, set (32) consists $n(n-1) \sim O(n^2)$ constraints. This is a

significantly smaller set of constraints for most of the problems, as it is very likely that $n >> 4$.
The MTZ-constraints ensure the formation of subtour-free tours as follows: to each of the u_i an integer between 1 and $|V|$ is assigned.
Let's assume that all MTZ-constraints are satisfied. If some arc $(i, j) \in A$ is part of the tour, the $x_{ij} = 1$, so $u_i - u_j + 1 \leq 0$. Then $u_j \geq u_i + 1$, for each of the arcs (i, j) in the tour. This implies that when the u_i are ordered based on the position of their corresponding vertex i in the tour, their values form an increasing sequence of integers. Let $\{i_1, \ldots, i_s\} \subset V$ $(s \in \mathbb{N}_{\geq 3})$, be a subtour on a subset of vertices of V, then $u_{i_1} + 1 \leq u_{i_2}$, $u_{i_2} + 1 \leq u_{i_3}$, $\ldots$, $u_{i_{s-1}} + 1 \leq u_{i_s}$. But, as this is a tour, the successor of vertex i_s in this tour is i_1, so $u_{i_s} + 1 \leq u_{i_1}$. This implies that $u_{i_s} + 1 \leq u_{i_1} \leq u_{i_s} - 1$. ↯
Therefore, if the MTZ-constraints are satisfied, subtours cannot exist.
This notion of assigning integer values to the u_i forms the basis of the fifth algorithm. Let the depot be denoted by vertex 0 and let the indices of the targets be $1, \ldots, n$. Let $N' \subset N$ be the set of targets which lie inside the new target zone, or if no targets lie within this zone, a prespecified set of targets which lie close to the zone of new targets. Finally, let delay factor $k > 0$ be some prespecified integer, that denotes the delay of the visit to the zone of new targets. Note that this integer k indicates how many targets $i \in N \backslash N'$ have to be visited before any of the targets $j \in N'$, located in or near the zone of new targets, can be visited. The 'Delay algorithm' (DA) consists of 2 parts. First an initial solution is found, that consists of a tour on the set N of vertices of optimal value and of optimal duration, which requires at most $C' < C$ fuel units, for some $C' > 0$. Let C be the total available amount of fuel. Then the so-called slack in fuel, $C - C'$, is reserved for the reoptimization step. If multiple optimal tours with similar objective values can be found, then a tour is chosen that has both the optimal value and minimal length. An additional requirement for the initial solution is that in this tour at least one of the $j \in N'$ are visited, in such a way that all of the $j \in N'$ are either visited as the k^{th} target or later, or not at all. Finally, the values of the u_i, $\forall i \in V$ have to be chosen in such a way that for any target $i \in V$, its corresponding value u_i reflects its position in the tour. This implies that if target i is the m^{th} target in the tour, $u_i = m$. Note that, as not all vertices are part of the tour, the reverse is not necessarily true and that there may be $i \neq j \in V$, such that $u_i = u_j$. In that case, at least one of the vertices i and j is



(a) Initial tour (*black nodes: fixed targets; zone of new targets: green rectangle*)

(b) New targets (*red nodes*) appear before the vehicle arrives at the second node

(c) Reoptimizing tour, using slack fuel capacity (*red arcs: new tour; grey nodes: unvisited targets*)

Figure 8: Example of the Delay algorithm

not part of the tour.
The second part of the algorithm takes place during the flight: let j be the first target of the tour that lies in the zone of new targets, and let $k' \geq k$ be the position of this target on the tour. Then a reoptimization step will take place just before the vehicle reaches the $(k'-1)^{st}$ target. Let C_r be the amount of fuel that was required for the part of the initial tour that started at this target and ended at the depot.
During the reoptimization step a path will be found that starts at the $(k'-1)^{st}$ vertex of the tour and ends at the depot, and that visits a subset of the set of targets that are available at that moment. This set consists of all new targets that have appeared before the start of the reoptimization process and all fixed targets that have not been visited yet and which can be visited without intersecting the first part of the tour. The length of this path is restricted by the remaining fuel plus the predetermined amount of slack: $C - C' + C_r$. The objective of the reoptimization process is again to obtain a path of maximal objective value.
For the first part of the algorithm we used the following model:

$$\text{(OP')} \qquad \max \sum_{i \in V} x_i v(i) - 0.1 \sum_{(i,j) \in A} y_{ij} d(i,j) \tag{33}$$

$$\text{s.t.} \sum_{(i,j) \in A} y_{ij} d(i,j) \leq C' \tag{34}$$

$$x_0 = \sum_{i \in V} y_{0i} = 1 \tag{35}$$

$$x_l = \sum_{i \in V} y_{il} = 1 \tag{36}$$

$$x_j = \sum_{i \in V \cup \{0\} \setminus \{j\}} y_{ij} = \sum_{i \in V \cup \{l\} \setminus \{j\}} y_{ji} \leq 1, \quad \forall j \in V \tag{37}$$

$$\sum_{i \in V'} x_i \geq i \tag{38}$$

$$u_i - u_j + 1 \leq |V|(1 - y_{ij}), \quad \forall (i,j) \in A \tag{39}$$

$$0 \leq u_i \leq |V|, \quad \forall i \in V^{++} \tag{40}$$

$$u_i \geq k, \quad \forall i \in V' \tag{41}$$

$$u_0 = 0 \tag{42}$$

$$u_l = \sum_{i \in V \cup \{l\}} x_i \tag{43}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) in A \tag{44}$$

$$x_i \in \{0,1\}, \quad \forall i \in V^{++} \tag{45}$$

In this model, V is the set of fixed targets, 0 is the depot and l is a dummy depot, similar to the one mentioned in Section 5.2.3; $V^{++} = V \cup \{0\} \cup \{l\}$. Then A is the set of all arcs. Note that A contains neither $(0,l)$, nor arcs of the type $(i,0),\ \forall i \in V$ or $(l,i),\ \forall i \in V$. As mentioned before, the set V' consists of either the fixed targets that lie inside the new target zone, or, if such targets do not exits, of targets that lie close to the new target zone. Note that Constraint (38) forces the initial tour to visit at least one of these targets. Other additional constraints are the constraints (41), which makes sure that if a target

$i \in V'$ is visited, it has at least k predecessors. To the objective value (33), a penalty function has been added in order to find tours with maximal objective value that have a minimal fuel consumption. The importance of minimizing the fuel consumption lies in the fact that when an initial tour requires less fuel, the amount of remaining fuel is larger, which may result in possibly better reoptimized tours.
The last new Constraint, (43), corrects a problem that arises when not all targets are visited, as is the case with most instances of the OP. In the TSP all vertices are visited, so when $u_i \in \{1, \ldots, |V|\}$, $\forall i \in V$, all values between 1 and $|V|$ are assigned to exactly one of the u_i. In the OP it is very likely that not all vertices are visited, and consequently that not all values between 1 and $|V|$ are used. This implies that there may be 'gaps' larger than 1 between the values of u_i of two vertices that are adjacent in the tour, as the MTZ-constraints only ensure sure that $u_j - u_i \geq 1$, for any arc (i, j) on the tour. In such cases, the values of the u_i do not reflect the positions of the corresponding targets in the tour. A way to solve this is to make sure that the value of u_l, where l is the dummy depot, is equal to its position in the tour, i.e. the number of targets in the tour, including l itself. As the number of targets in the tour can be determined by adding the values of the x_i, $u_l = x_l + \sum_{i \in V} x_i$.
The model for the second part of the algorithm is similar to the one for the first part. Some small changes have to be made: in this case, the set of available targets is not V, but $V'' \cup W'$, where V'' is the set of targets that have not been visited in the first part of the tour and which can be reached without intersecting the first part of the tour. Note that as before, $l \notin V''$. $W' \subseteq W$ is the subset of new targets that have appeared prior to the arrival of the vehicle at the $(k-1)^{st}$ target of the tour. As the vehicle now departs from the $(k-1)^{st}$ target, and not the depot, Constraints (35) and (42) have to be replaced by $x_{k-1} = \sum_{i \in V''} y_{k-1\ i} = 1$ and $u_{k-1} = 0$, respectively. Constraint (41) is superfluous, as the delay factor was only necessary for determining the initial tour. Furthermore, the set of arcs A has to be expanded with arcs leading to and coming from all new targets.

6.2 Description of the simulation settings

For determining the performance of the Delay algorithm, the same target set has been used as for the other algorithms; see Section 5.1 for more details. The set of fixed targets was again expanded by a set of new targets, but these were generated differently, as the new targets only appear in a prespecified zone. In this case, the zone of new targets is a rectangle located between $(8.8, 15.9)$ and $(14.2, 19.3)$; see figure 9. Arrival times and scores were generated the same way as before; coordinates were generated slightly different, as the zone in which the new targets appeared was different. Let the number of new targets be N and $1 \leq i \leq N$ the index of one of the new targets, then its coordinates $(x(i), y(i))$ are determined as follows: let $(X_i, Y_i) \sim U(0,1) \times U(0,1)$, then $x(i) = 5.4X_i + 8.8$ and $y(i) = 3.4Y_i + 15.9$. The distances between any pair of targets were computed the same way as before. Let N be the number of new targets and let the number of fixed targets be 19. Then for any pair $0 \leq i, j \leq N + 19$, such that $i \neq j$ and $j \neq 0$, the distance from target i to target j is $d(i,j) = \sqrt{(x(i) - x(j))^2 + (y(i) + y(j))^2} + 2$. If $j = 0$, the distance from target i back to the depot is $d(i,0) = \sqrt{(x(i) - x(0))^2 + (y(i) - y(0))^2}$.

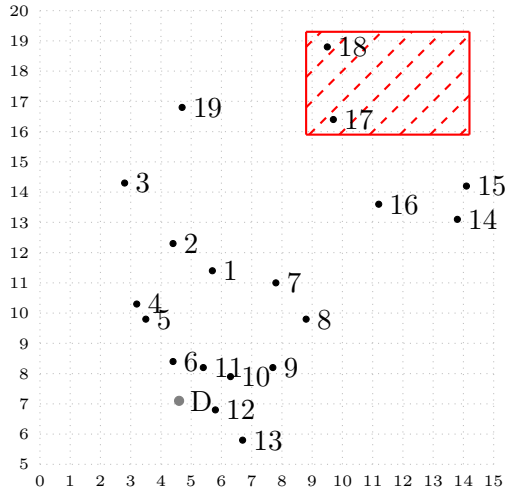


Figure 9: Location in the Euclidean plane of the nodes of the Tsiligirides-instance and the zone of new targets (red rectangle).

Just as for the first four algorithms, we determined the performance of the DA on 10000 simulated instances. It is clear that for each of these instances the same initial solution should be used. There were two additional parameters that had to be taken into account: the amount of fuel that is reserved for the reoptimization step, here after denoted as the 'slack value', and the number of targets that have to be visited before the vehicle is allowed to visit a target in the new target zone, here on after denoted as the 'delay factor'. It was not obvious which value to choose for either parameter, or whether some combinations of slack value and delay factor may lead to better initial solutions or better reoptimized solutions, so for both parameters a number of values was chosen. For the slack value we choose 10.0, 15.0 or 20.0 fuel units; for the delay factor we chose 5, 6 or 7. This leads to 9 combinations of parameter settings. Each of these settings were combined with an arrival rate of 3.0, 10.0 or 20.0, in order to determine how well the DA would perform under different circumstances. An overview of the settings can be seen in table 8.

Table 8: Overview of all 27 simulation settings for the Delay algorithm (*per slack value there are 3 options for the delay factor, per delay factor there are 3 options for the arrival rate*)

Slack value	3.0 // 10.0 // 20.0								
Delay factor	5			6			7		
Arrival rate	3	10	20	3	10	20	3	10	20

The total fuel capacity in each case was 65.0, as before, so for the initial tours there were either 55.0, 50.0 or 45.0 fuel units available. For each of these simulation settings, we determined the optimal initial solution. These solution tours were optimal on two accounts, as mentioned before: their objective value was maximal and their fuel consumption relatively minimal. The optimal solutions for all of the settings are depicted in figures C.1, C.2 and C.3 in appendix C.

Table 9: Initial solutions for all simulation settings

Slack value	Delay factor	Objective value	Fuel consumption	Number of targets
10	5	290	54.546	9
10	6	290	54.603	9
10	7	285	51.290	9
15	5	250	49.196	8
15	6	250	49.196	8
15	7	245	49.055	8
20	5	210	44.633	6
20	6	205	44.820	7
20	7	205	44.820	7

Apart from these simulation settings, we also determined how well the first four algorithms perform under these new circumstances, and whether the DA performs better or not. For this we only chose some combinations of slack value and delay factor, as the reoptimization process took quite some time in case of the HRA and the RRA. The chosen settings were a slack value of 15.0 combined with a delay factor of 5 and a slack value of 20.0 combined with a delay factor of 6. In all cases the arrival rate was set to 20.0. Due to limitations on the time available for the simulations, we only applied the algorithm to the first 1000 sets of new targets that arrive according to an arrival rate of 20.0. In addition to the lower number of simulations, the computation time for some of the algorithms had to be restricted, in order to be able to obtain results within acceptable time periods. For the HRA, the computation time was restricted to 900 seconds for the entire reoptimization process of one instance. The time limit for each of the reoptimization steps of the RRA was set to 15 seconds. Setting a time limit implies that if an algorithm cannot find the optimal solution within the given time frame, the best found solution so far will be used as the optimal solution.
In addition to setting a time limit, the computation time of the RRA was reduced further by removing the fuel-penalty from the objective function.

6.3 Evaluation of the results

6.3.1 Results of the Delay algorithm

An overview of the results of all 27 simulations can be found in table 10. In this table, only the averages of the results are mentioned; the complete set of results are shown in appendices D and E, where the data are sorted by slack value and by arrival rate, respectively. The found results can be analysed from several points of view.
In the table in appendix D we can see that for any combination of slack value and delay factor, when comparing the average results, the DA seems to find tours with higher objective values in instances with a higher rate. When all results for one of the slack values are compared, i.e. when the slack value is fixed, but both delay factor and arrival rate vary, we obtain sets of results that are quite similar; there does not seem to be a delay factor for which significantly better results are found.

Table 10: Results of the Delay algorithm (*'Targets': number of visited targets in total, reoptimized tours. Objective value and fuel consumption refer to the total, reoptimized tours*)

Slack value	Delay factor	Rate	Objective value Average	Fuel consumption Average	Targets Average	Computation time Average (ms)
10	5	3	368.864	64.253	13.396	76.4
10	5	10	420.903	64.382	13.788	93.7
10	5	20	489.676	64.347	14.533	318.7
10	6	3	377.473	64.162	13.819	75.0
10	6	10	432.294	64.370	14.161	130.4
10	6	20	503.542	64.365	14.975	1070.0
10	7	3	374.933	64.651	13.715	57.1
10	7	10	429.655	64.364	13.838	204.1
10	7	20	486.912	64.441	14.277	2576.2
15	5	3	377.473	64.162	13.819	77.3
15	5	10	432.921	64.381	14.216	300.5
15	5	20	503.542	64.365	14.975	1071.6
15	6	3	377.473	64.162	13.819	74.3
15	6	10	432.921	64.381	14.216	336.8
15	6	20	503.542	64.365	14.975	1065.9
15	7	3	383.749	64.545	14.002	63.8
15	7	10	439.251	64.323	14.649	193.1
15	7	20	507.770	64.379	15.350	1288.4
20	5	3	368.864	64.253	13.396	135.3
20	5	10	420.903	64.382	13.788	127.7
20	5	20	489.676	64.347	14.533	335.1
20	6	3	382.105	64.162	13.435	83.9
20	6	10	441.224	64.367	14.260	323.6
20	6	20	505.577	64.443	14.807	2567.8
20	7	3	382.105	64.162	13.435	84.9
20	7	10	441.224	64.367	14.260	314.4
20	7	20	505.577	64.443	14.807	2371.1

Table 11: Overview of the simulation results of all reoptimization algorithms, based on the settings for the Delay algorithm (*'Fuel': average fuel consumption of complete, reoptimized tours; 'Targets': number of visited targets in reoptimized tours*)

Algorithm	Slack value	Delay factor	Objective value Min	Objective value Max	Objective value Average	Fuel Average	Targets Average	Computation time Average (ms)
IA	15	5	295	500	387.705	63.977	13.459	0.1
	20	6	230	475	329.015	63.833	12.104	0.2
HRA	15	5	350	610	466.965	55.890	12.725	1160.2
	20	6	205	615	486.336	59.753	14.646	20976.6
RRA	15	5	405	670	543.020	64.533	14.014	7882.2
	20	6	385	660	530.155	64.573	13.792	25427.3
ITA	15	5	250	540	308.675	53.245	10.447	8.6
	20	6	205	290	208.589	45.005	8.091	0.7
DA	15	5	360	660	503.080	64.364	14.962	810.7
	20	6	380	625	505.576	64.451	14.779	3809.1

When looking at the computation times of each of the simulation sets, we see that most of the reoptimizations take very little time on average. For small sets of new targets, i.e. in case of arrival rate 3.0, the maximal computation time is 2.5 seconds. When the arrival rate was set to 10.0, the maximal computation time was approximately 405 seconds, but for each of the simulation sets with rate 10.0, there were not more than 8 instances with a computation time higher than 10 seconds. For the instances with arrival rate 20.0, the maximal computation time was about 3126 seconds.

Similar observation can be made about the results sorted by arrival rate, shown in appendix E. As previously stated, the results for an arrival rate of 20.0 seem most promising, but when all results of the simulations with arrival rate 20.0 are compared, it not clear which settings yield the best results. We can only observe that the results for a combination of a slack value of 15 and a delay factor of 5, and a slack value of 20 and a delay factor of 6 seem to the most 'stable', that is, the intervals in which the objective values lie are smallest for these two combinations of settings.

From these results we may conclude that the DA does not perform relatively better or relatively worse under any of the combinations of settings.

6.3.2 Comparison of all five algorithms

An overview of the averages of the results can be seen in table 11; a more extensive set of results can be found in tables F.1 and F.2 in appendix F. It is easy to see that all four algorithms yield tours with objective values that are higher than the initial tours with objective values 250 and 205, respectively. The IA and the ITA do not perform very well, when compared to the other three algorithms, but they do find better tours than the initial solution. The IA finds better tours for all instances; the ITA finds better tours for about 76% of the instances with slack value 15.0 and delay factor 5, but for the instances with slack value 20.0 and delay factor 6 it finds a better solution in only 2.5% of the cases. The RRA seems to yield tours with the highest objective values, on average. When looking at the computation times for the HRA, RRA and DA, we see that the DA requires, on average, far less time for its reoptimization step than the other two algorithms. The maximal computation time for the DA seems quite large, but we must remark that this concerns a single instance, and that among the used test instances, the second largest computation time for the DA is about 59 seconds.
Note that, as the computation times were restricted for both the HRA and the RRA, it is very likely that for some of the instances these algorithms were not able to find an optimal solution. For the HRA less than 1% of the reoptimizations were cut short in case of the instances with slack value 20 and delay factor 6. It could reoptimize all of the instances with slack value 15 and delay factor 5 within the time limit, however. The RRA finished the entire reoptimization process within 15 seconds for 81.0% of the instances with slack value 15 and delay factor 5, so at least 81% of the found solutions were optimal. By a similar argument, in case of the instances with slack value 20.0 and delay factor 6, at least 31.3% of the instances were solved to optimality.
The fact that the reoptimization by either the HRA or the RRA takes a lot of time for most instances with slack value 20.0 and delay factor 6 can be easily explained. As

mentioned before, during the reoptimization steps a set of both fixed and new targets are considered, but those targets have to meet some prespecified requirements. For the instances that were used in this case, the only requirement for the fixed targets is that they have not been visited yet and that they can be reached without intersecting any of the arcs that the vehicle has to travel before arriving at the zone of new targets. When we look at the initial solution for the instances with slack value 20.0 and delay factor 6, we see that upon arrival at target 16, all unvisited fixed targets meet this requirement, so the set of targets that will be used for reoptimization consist of all fixed targets, except 8, 9, 13 and 16. This may result in very large target sets for the reoptimization steps.
In table 12 we can see the number of solutions that were found by the DA, HRA or RRA within certain time periods.

Table 12: Overview of the number of reoptimization instances, out of 1000 instances, that were solved by the DA, HRA or RRA within time periods of 3, 5, 10 or 15 seconds

Algorithm	Slack value	Delay factor	Time limit (s) 3	5	10	15
DA	15	5	968	979	989	992
	20	6	928	958	978	987
HRA	15	5	941	966	980	986
	20	6	752	812	859	879
RRA	15	5	291	546	763	810
	20	6	70	121	241	313

From these results we may conclude that the IA and the ITA perform not so well as the DA. The performances of the HRA and RRA may be similar to or slightly better than the performance of the DA, but their computation times are a lot longer on average. A second factor that has to be taken into account is the fact that the tours found by the RRA may differ greatly from the initial tour, which may be unacceptable in real-life situations. When taking all the important factors into account, the Delay algorithm seems to perform best.

7 Conclusions and future research

7.1 Conclusion

In this thesis we investigated the Online UAV mission planning problem (UAV-MPP). The research started with a literature analysis, in which we discovered that the Online UAV-MPP was one of the extensions of the basic UAV-MPP that was most interesting to investigate, and that the model for the UAV-MPP could best be based on the 'Orienteering problem'.
We found models for each of the other three extensions of the basic UAV-MPP, based on the Orienteering Problem and determined that for the Online UAV-MPP a similar model does not exist. We then developed four strategies for solving this problem: the Insert Algorithm (IA), partly based on a solution approach for the Vehicle Routing Problem with dynamic travel times [30], the Halfway Reoptimization Algorithm (HRA), the Repeated Reoptimization Algorithm (RRA) and the Internal Target Algorithm (ITA). Each of the algorithms was based on first determining an optimal solution for the problem without new targets, and successively reoptimizing it by either adding new targets to the predetermined tour (IA) or replacing part of the predetermined tour (HRA, RRA and ITA) according to certain rules.
By counting the total number of visited targets in the reoptimized tour, which was achieved by setting the scores of both given and new targets to 1, and comparing the scores of the resulting solutions to the scores of the solutions for the same instances by an offline solution method, we were able to determine theoretical performance bounds for all four algorithms. These bounds, the so-called competitive ratio, were $\frac{m}{m+\mu}$ for the IA, HRA and RRA, where m is the number of targets in the original tour on only given targets, and μ is the number of new targets that have appeared. We proved that the bound found for the IA is tight. For the ITA a similar bound was found: in that case, we only looked at the $\mu' \leq \mu$ new targets that would appear 'inside' the original tour, and the competitive ratio was set to $\frac{m}{m+\mu'}$.
We performed simulations in order to gain insight into the actual performance of all four algorithms, both in comparison to the found competitive ratios and in real-life situations, based on a benchmark instance and generated sets of new targets. These simulations showed us that the RRA yields tours with the highest average objective value, i.e. the highest sum of target scores, and the HRA yields tours with the second best average objective value. However, the time required for one of the reoptimization steps of the RRA could be as much as 140 seconds. Another issue for the RRA is that the reoptimized tours may differ greatly from the original tour. Both issues might be unacceptable in real-life situations; the HRA therefore seems more suitable. The IA did not yield any improved solutions, due to the fact that there was not enough sufficient fuel remaining to allow addition of new targets to the original tour. A similar argument can be applied to the ITA, which only yielded improved solutions for a few of the instances.
We performed a second series of simulations in which all scores - both for the fixed targets and the new ones - were equal, in order to find empirical performance bounds. We found that the RRA and the HRA yield the best tours, that is, tours that contain the most targets. Both algorithms found optimal solutions for some of the generated sets of targets.

In the last part of the thesis we presented a solution strategy developed specifically for cases where all new targets appear in a prespecified, smaller zone inside the target area, the Delay Algorithm (DA). The strategy was based on the idea of both forcing the tour to pass through this zone and delaying its arrival there, and reserving part of the fuel for the reoptimization step. We performed simulations in order to determine the performance of this algorithm, in which we varied 3 factors: the number of targets that had to be visited before arriving at the zone of new targets, the amount of fuel that was reserved for the reoptimization step, and the number of new targets. From these simulations we concluded that only the last factor had significant influence on the objective value, i.e. that in instances with more new targets, the objective value is relatively higher.
Finally, we compared the performance of the DA to the performance of the first four algorithms, based on the initial solutions and simulation settings for the DA. The results for the DA, RRA and HRA were similar and better than those for the IA or ITA. Whilst the results for the DA were not significantly higher than those for the HRA or the RRA, we may conclude, due to secondary factors like computing times and overlap between the reoptimized tour and initial tour, that the DA performs best.

7.2 Future research

First of all, there may be circumstances for which the IA performs better than it has for the given circumstances. Two simple adaptations of the given settings, that may lead to better performance, are shorter recording times and incorporating some slack in the fuel capacity. By this we mean that part of the fuel capacity will remain reserved for reoptimization purposes, and cannot be used for the initial tour. A second adaptation of the IA that may result in improved performance involves an alteration of the objective function. As mentioned before, tours of optimal value for either basic problem are not necessarily of minimal length. When a penalty for fuel consumption is added to the objective function, the optimal tour will be optimal on both aspects, and more fuel remains for adding new targets to the tour. A slight disadvantage of such objective functions is that it will take more time to optimize the corresponding problems.
Another part of the research that might be interesting to investigate further, is an adaptation of the Internal Target algorithm. When the algorithm is not based on looking for targets only in triangles 'inside' the tour, but on looking for targets in larger regions in the vicinity of the tour, it will allow the vehicle to visit a possibly larger subset of the new targets, and thud increase the possibility of yielding tours of higher objective value. An example of such a larger region is a circle that encloses multiple arcs of the tour.
The Delay algorithm may be adapted to situations where the new targets appear in multiple zones within the target area. A suggestion for such an adaptation is to visit at least one given target inside or near each of the zones of new targets, and appointing a different delay factor to each of the zones. The delay factors could, for instance, be based on importance of a certain zone or arrival rate of the targets in that zone.

And finally, an aspect of the Online UAV-MPP that was not researched in this thesis, but will be very interesting to study, is the case where targets can either appear or disappear.

References

[1] L. Allulli, G. Ausiello, V. Bonifaci, and L. Laura. On the power of lookahead in on-line server routing problems. *Theoretical Computer Science*, 408(2/3):116–128, 2008.

[2] C. Archetti, A. Hertz, and M.G. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1):49–76, 2007.

[3] G. Ausiello, V. Bonifaci, and L. Laura. The online price-collecting travelling salesman problem. *Information Processing Letters*, 107:199–204, 2008.

[4] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific and Dynamic Ideas, 1 edition, 1997.

[5] A. Blum, S. Chawla, D.R. Karger, T. Lane, A. Meyerson, and M. Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. *SIAM Journal on Computing*, 37(2):653–670, 2007.

[6] H. Bouly, D.-C. Dang, and A. Moukrim. A memetic algorithm for the team orienteering problem. *Quarterly Journal of the French, Belgian and Italian Operations Research Society*, 8(1):49–70, 2010.

[7] B. Brodén, M. Hammar, and B.J. Nilsson. Online and offline algorithms for the time-dependent tsp with time zones. *Algorithmica*, 39(4):299–319, 2004.

[8] S.E. Butt and T.M. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101–111, 1994.

[9] A.M. Campbell, M. Gendreau, and B.W. Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, 2011.

[10] I.-M. Chao, B.L. Golden, and E.A. Wasil. Theory and methodology: The team orienteering problem. *European Journal of Operational Research*, 88(3):464–474, 1996.

[11] I.-M. Chao, B.L. Golden, and E.A. Wasil. Theory and methodology: A fast and effective heuristic for the orienteering problem. *European journal of Operational Research*, 88(3):475–489, 1996b.

[12] L. Evers, T. Dollevoet, A.I. Barros, and H. Monsuur. Robust UAV mission planning. Submitted to *Annals of Operations Research.*

[13] L. Evers, K. Glorie, and S. van der Ster. The orienteering problem under uncertainty, stochastic programming and robust optimization compared. To be published.

[14] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39(2):188–205, 2005.

[15] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151(1):1–11, 2003.

[16] A. Gupta, R. Krishnaswamy, V. Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In *Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1522–1538. SIAM, Philadelphia, PA, USA, 2012.

[17] S. Ichoua, M. Gendreau, and J.-Y. Potvin. Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2):379–396, 2003.

[18] T. Ilhan, S.M.R. Iravani, and M.S. Daskin. The orienteering problem with stochastic profits. *IIE Transactions*, 40(4):406–421, 2008.

[19] S. Irani, X. Lu, and A. Regan. On-line algorithms for the dynamic travelling repair problem. *Journal of Scheduling*, 7(3):243–258, 2004.

[20] P. Jaillet and M.R. Wagner. Online routing problems: Value of advanced information as improved competitive ratios. *Transportation Science*, 40(2):200–210, 2006.

[21] B. Kalyanasundaram and K.R. Pruhs. The online transportation problem. *SIAM Journal on Dirscrete Mathematics*, 13(3):370–383, 2000.

[22] M.G. Kantor and M.B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, 43(6):629–635, 1992.

[23] A.R. Karlin, M.S. Manasse, L. Rudolph, and D.D. Sleator. Competitive snoopy caching. *Algorithmica*, 28(2):79–119, 1988.

[24] S. Kataoka and S. Morito. An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31(4):515–531, 1988.

[25] N. Labadie, J. Melechvský, and R. Wolfler Calvo. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. *Journal of Heuristics*, 17(6):729–753, 2011.

[26] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[27] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2-3):193–207, 1990.

[28] J.A. Larco, R. Dekker, and U. Kaymak. Coverage consideration for emergencies in service vehicle routing. To be published.

[29] S.W. Lin and V.F. Yu. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, 217(1):94–107, 2012.

[30] S. Lorini, J.-Y. Potvin, and N. Zufferey. Online vehicle routing and scheduling with dynamic travel times. *Computers & Operations Research*, 38(7):1086–1090, 2011.

[31] C.E. Miller, A.W. Tucker, and R.A. Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the Association for Computing Machinery*, 7(4):326–329, 1960.

[32] V. Nagarajan and R. Ravi. The directed orienteering problem. *Algorithmica*, 60(4):1017–1030, 2011.

[33] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, 36(4):1191–1203, 2009.

[34] S.M. Ross. *Introduction to Probability Models*. Academic Press, 10 edition, 2010.

[35] M.A. Russel and G.B. Lamont. A genetic algorithm for unmanned aerial vehicle routing. In H.G. Beyer, editor, *Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 1523–1530. Association for Computing Machinery, New York, NY, USA, 2005.

[36] J. Silberholz and B. Golden. The effective application of a new approach to the generalized orienteering problem. *Journal of Heuristics*, 16(3):393–415, 2010.

[37] W. Souffriau, P. Vansteenwegen, G. vanden Berghe, and D. van Oudheusden. A path relinking approach for the team orienteering problem. *Computers & Operations Research*, 37(11):1853–1859, 2010.

[38] W. Souffriau, P. Vansteenwegen, G. vanden Berghe, and D. van Oudheusden. The planning of cycle trips in the province of East Flanders. *Omega*, 39(2):209–213, 2011.

[39] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, 2005a.

[40] H. Tang and E. Miller-Hooks. Algorithms for a stochastic selective travelling salesperson problem. *The Journal of the Operational Research Society*, 56(4):439–452, 2005b.

[41] F. Tricoire, M. Romauch, K.F. Doerner, and R.F. Hartl. Heuristics for the multi-period orienteering problem with multiple time-windows. *Computers & Operations Research*, 37(2):351–367, 2010.

[42] T. Tsiligirides. Heuristic methods applied to orienteering. *The Journal of the Operational Research Society*, 35(9):797–809, 1984.

[43] P. Vansteenwegen, W. Souffriau, and D. van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209(1):1–10, 2011.

[44] P. Vansteenwegen, W. Souffriau, G. vanden Berghe, and D. van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–2390, 2009a.

[45] P. Vansteenwegen, W. Souffriau, G. vanden Berghe, and D. van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, 196(1):118–127, 2009b.

[46] C. Zhu, J.Q. Hu, F. Wang, Y. Xu, and R. Cao. On the tour planning problem. *Annals of Operations Research*, 192(1):67–86, 2012.

A Detailed results of the first four online algorithms

Table A.1: Overview of all data for the instances with varying scores
(*Results in columns 5 up to 7 are averages over 10000 instances. Fuel consumption denotes the fuel consumption of reoptimized tour; computation time is given in ms*)

Algorithm	Rate	Objective value			Fuel consumption	Computing time
		Min	Max	Average		(ms)
IA	3	15	15	15	63.448	0.0
	5	15	15	15	63.448	0.0
	10	15	15	15	63.448	0.1
	20	15	15	15	63.448	0.2
HRA	3	15	16	15.035	59.218	18.9
	5	15	17	15.085	58.448	24.4
	10	15	17	15.253	58.116	33.9
	20	15	17	15.591	58.326	56.0
RRA	3	15	17	15.098	63.611	710.6
	5	15	18	15.227	63.690	1153.3
	10	15	19	15.617	63.802	2587.1
	20	15	20	16.2910	63.963	7633.6
ITA	3	15	15	15	63.437	0.3
	5	15	16	15.000	63.433	0.5
	10	15	16	15.000	63.418	0.9
	20	15	16	15.000	63.393	2.1

Table A.2: Overview of all data for the instances with equal scores
(*Results in columns 5 up to 7 are averages over 10000 instances. Fuel consumption denotes the fuel consumption of reoptimized tour; computation time is given in ms*)

Algorithm	Rate	Objective value			Fuel consumption	Computing time
		Min	Max	Average		(ms)
IA	3	360	360	360	63.937	0.0
	5	360	360	360	63.937	0.0
	10	360	360	360	63.937	0.0
	20	360	360	360	63.937	0.1
HRA	3	360	500	372.950	59.213	29.2
	5	360	510	380.782	58.179	41.1
	10	360	520	400.301	57.585	70.7
	20	360	580	434.425	57.534	187.2
RRA	3	360	510	383.677	64.397	1458.6
	5	360	535	397.172	64.074	1709.1
	10	360	595	427.196	64.434	4118.5
	20	360	675	475.109	64.438	14251.9
ITA	3	360	420	361.245	64.907	0.5
	5	360	420	362.107	64.885	1.0
	10	360	430	364.101	64.838	2.1
	20	360	460	368.312	64.745	5.7

B Computation times

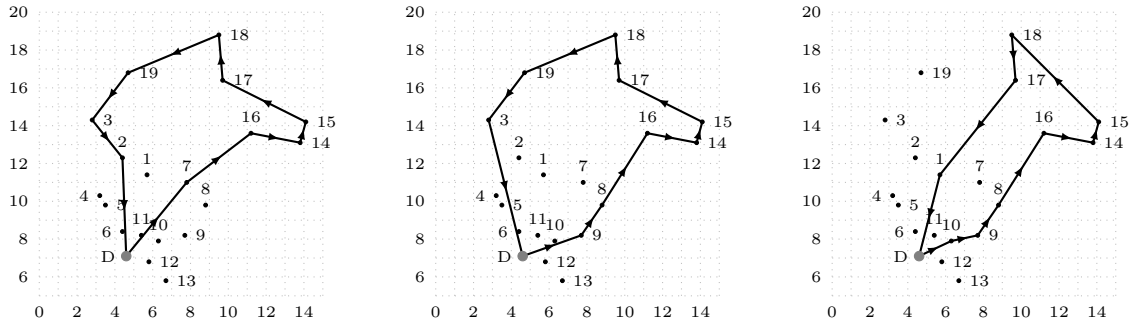
Table B.1: Overview of the computation times for the instances with equal scores (*The results in columns 3, 4 and 5 are: minimal found value, maximal found value and average value over 10000 instances, respectively, all given in ms*)

Algorithm	Rate	Computation time		
		Min	Max	Average
IA	3	0	15	0.0
	5	0	16	0.0
	10	0	16	0.1
	20	0	16	0.2
HRA	3	0	157	18.9
	5	0	157	24.4
	10	0	173	33.9
	20	15	407	56.0
RRA	3	0	4529	710.6
	5	0	4528	1153.3
	10	78	13319	2587.1
	20	687	38454	7633.6
ITA	3	0	117	0.3
	5	0	119	0.5
	10	0	121	0.9
	20	0	120	2.1

Table B.2: Overview of the computation times for the instances with varying scores (*The results in columns 3, 4 and 5 are: minimal found value, maximal found value and average value over 10000 instances, respectively, all given in ms*)

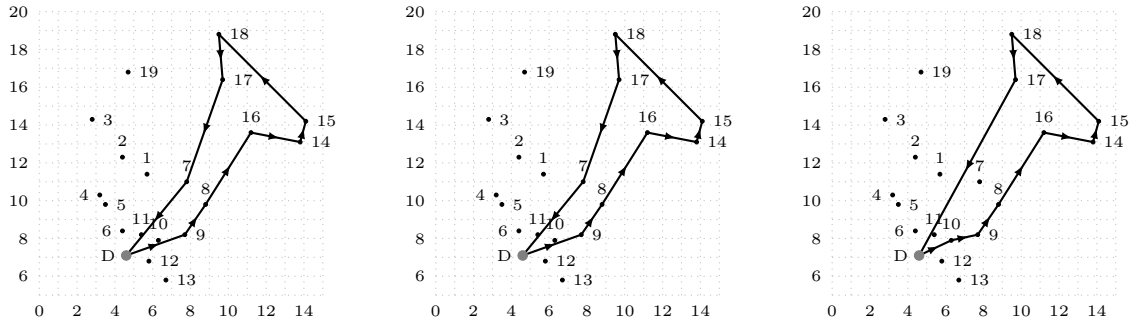
Algorithm	Rate	Computation time		
		Min	Max	Average
IA	3	0	15	0.0
	5	0	1	0.0
	10	0	16	0.0
	20	0	16	0.1
HRA	3	0	249	29.2
	5	0	298	41.1
	10	0	609	70.7
	20	16	3028	187.2
RRA	3	0	21790	1458.6
	5	0	6854	1709.1
	10	63	27129	4118.5
	20	422	142740	14251.9
ITA	3	0	98	0.5
	5	0	100	1.0
	10	0	86	2.1
	20	0	82	5.7

C Initial solutions for the Delay algorithm



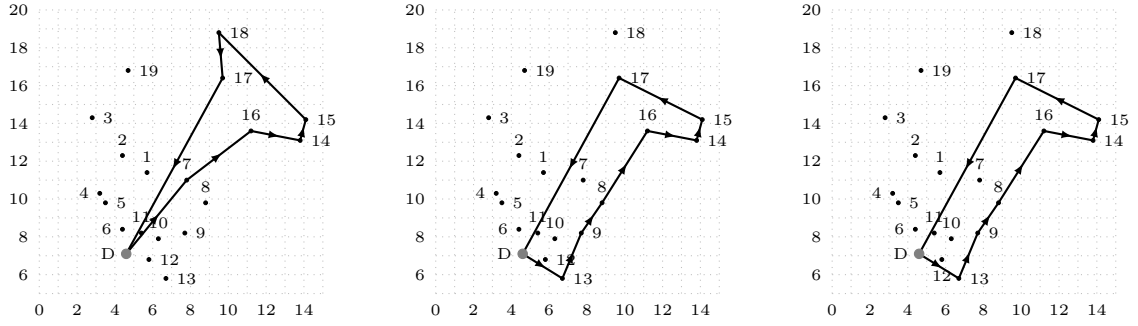
(a) Delay factor 5 (b) Delay factor 6 (c) Delay factor 7

Figure C.1: Initial solutions for the Delay algorithm, with slack value 10.0 and delay factor 5, 6 or 7.



(a) Delay factor 5 (b) Delay factor 6 (c) Delay factor 7

Figure C.2: Initial solutions for the Delay algorithm, with slack value 15.0 and delay factor 5, 6 or 7.



(a) Delay factor 5 (b) Delay factor 6 (c) Delay factor 7

Figure C.3: Initial solutions for the Delay algorithm, with slack value 20.0 and delay factor 5, 6 or 7.

D Detailed results of the Delay algorithm

Overview of all simulation results of the Delay algorithm, sorted by slack value.
min: minimal value; max: maximal value; av: average value

Slack	Delay	Rate	Objective value			Fuel consumption			Number of visited targets			Computation time (ms)		
			Min	Max	Av	Min	Max	Av	Min	Max	Av	Min	Max	Av
10	5	3	340	505	368.864	63.153	64.999	64.253	12	15	13.396	15	686	76.4
10	5	10	340	620	420.903	62.866	65.000	64.382	12	16	13.788	15	18606	93.7
10	5	20	345	705	489.676	62.864	65.000	64.347	12	17	14.533	15	179194	318.7
10	6	3	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1250	75.0
10	6	10	350	640	432.294	62.865	65.000	64.370	12	17	14.161	15	11318	130.4
10	6	20	350	685	503.542	62.881	65.000	64.365	13	17	14.975	15	1337311	1067.0
10	7	3	355	520	374.933	62.867	65.000	64.651	12	16	13.715	15	1233	57.1
10	7	10	355	600	429.655	62.868	65.000	64.364	12	16	13.838	15	39682	204.1
10	7	20	355	620	486.912	62.875	65.000	64.441	12	16	14.277	15	1644976	2576.2
15	5	3	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1264	77.3
15	5	10	350	640	432.921	62.965	65.000	64.381	12	17	14.216	15	63972	300.5
15	5	20	350	685	503.542	62.881	65.000	64.365	13	17	14.975	17	1340865	1071.6
15	6	3	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1218	74.3
15	6	10	350	640	432.921	62.965	65.000	64.381	12	17	14.216	15	73184	336.8
15	6	20	350	685	503.542	62.881	65.000	64.365	13	17	14.975	16	1344782	1065.9
15	7	3	360	525	383.749	62.915	65.000	64.545	13	16	14.002	15	1138	63.8
15	7	10	360	635	439.251	62.922	65.000	64.323	13	17	14.649	15	71761	193.1
15	7	20	360	665	507.770	62.890	65.000	64.379	13	18	15.350	15	1476317	1288.4
20	5	3	340	505	368.864	63.153	64.999	64.253	12	15	13.396	15	1785	135.3
20	5	10	340	620	420.903	62.866	65.000	64.382	12	16	13.788	15	28170	127.7
20	5	20	345	705	489.676	62.865	65.000	64.347	12	17	14.533	17	188465	335.1
20	6	3	355	520	382.105	62.929	65.000	64.162	13	15	13.435	15	2472	83.9
20	6	10	355	615	441.224	62.897	65.000	64.367	13	17	14.260	15	405412	323.6
20	6	20	355	655	505.577	62.911	65.000	64.443	13	17	14.807	31	3126091	2567.8
20	7	3	355	520	382.105	62.929	65.000	64.162	13	15	13.435	15	1718	84.9
20	7	10	355	615	441.224	62.897	65.000	64.367	13	17	14.260	17	392399	314.4
20	7	20	355	655	505.577	62.911	65.000	64.443	13	17	14.807	31	2484651	2371.1

E Detailed results of the Delay algorithm (2)

Overview of all simulation results of the Delay algorithm, sorted by arrival rate.
min: minimal value; max: maximal value; av: average value

Rate	Slack	Delay	Objective value			Fuel consumption			Number of visited targets			Computation time (ms)		
			Min	Max	Av	Min	Max	Av	Min	Max	Av	Min	Max	Av
3	10	5	340	505	368.864	63.153	64.999	64.253	12	15	13.396	15	686	76.4
3	10	6	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1250	75.0
3	10	7	355	520	374.933	62.866	65.000	64.651	12	16	13.715	15	1233	57.1
3	15	5	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1264	77.3
3	15	6	350	530	377.473	63.005	65.000	64.162	12	16	13.819	15	1218	74.3
3	15	7	360	525	383.749	62.915	65.000	64.545	13	16	14.002	15	1138	63.8
3	20	5	340	505	368.864	63.153	64.999	64.253	12	15	13.396	15	1785	135.3
3	20	6	355	520	382.105	62.929	65.000	64.162	13	15	13.435	15	2472	83.9
3	20	7	355	520	382.105	62.929	65.000	64.162	13	15	13.435	15	1718	84.9
10	10	5	340	620	420.903	62.866	65.000	64.382	12	16	13.788	15	18606	93.7
10	10	6	350	640	432.294	62.865	65.000	64.370	12	17	14.161	15	11318	130.4
10	10	7	355	600	429.655	62.868	65.000	64.364	12	16	13.838	15	39682	204.1
10	15	5	350	640	432.921	62.965	65.000	64.381	12	17	14.216	15	63972	300.5
10	15	6	350	640	432.921	62.965	65.000	64.381	12	17	14.216	15	73184	336.8
10	15	7	360	635	439.251	62.922	65.000	64.323	13	17	14.649	15	71761	193.1
10	20	5	340	620	420.903	62.866	65.000	64.382	12	16	13.788	15	28170	127.7
10	20	6	355	615	441.224	62.897	65.000	64.367	13	17	14.260	15	405412	323.6
10	20	7	355	615	441.224	62.897	65.000	64.367	13	17	14.260	17	392399	314.4
20	10	5	345	705	489.676	62.865	65.000	64.347	12	17	14.533	15	179194	318.7
20	10	6	350	685	503.542	62.881	65.000	64.365	13	17	14.975	15	1337311	1070.0
20	10	7	355	620	486.912	62.875	65.000	64.441	12	16	14.277	15	1644976	2576.2
20	15	5	350	685	503.542	62.881	65.000	64.365	13	17	14.975	17	1340865	1071.6
20	15	6	350	685	503.542	62.881	65.000	64.365	13	17	14.975	16	1344782	1065.9
20	15	7	360	665	507.770	62.89	65.000	64.379	13	18	15.350	15	1476317	1288.4
20	20	5	345	705	489.676	62.865	65.000	64.347	12	17	14.533	17	188465	335.1
20	20	6	355	655	505.577	62.911	65.000	64.443	13	17	14.807	31	3126091	2567.8
20	20	7	355	655	505.577	62.911	65.000	64.443	13	17	14.807	31	2484651	2371.1

F Detailed results of all algorithms

Table F.1: Detailed results of objective values and fuel consumptions for all algorithms, applied to the settings and new target instances for the DA, with arrival rate 20.0.
(*Min: minimal value; Max: maximal value; Av: average value; Objective value and fuel consumption refer to the total reoptimized tour*)

Algorithm	Slack value	Delay factor	Objective value			Fuel consumption		
			Min	Max	Av	Min	Max	Av
IA	15	5	295	500	387.705	56.308	64.997	63.977
	20	6	230	475	329.015	50.961	64.997	63.833
HRA	15	5	350	610	466.965	54.448	56.494	55.890
	20	6	205	615	486.336	44.820	60.352	59.753
RRA	15	5	405	670	543.020	62.999	65.000	64.533
	20	6	385	660	530.155	63.059	65.000	64.573
ITA	15	5	250	540	308.675	49.196	64.937	53.245
	20	6	205	290	208.589	44.820	49.990	45.005
DA	15	5	360	660	503.080	62.881	64.998	64.364
	20	6	380	625	505.576	63.006	65.000	64.451

Table F.2: Detailed results of the number of visited targets and computing times for all algorithms, applied to the settings and new target instances for the DA, with arrival rate 20.0.
(*Min: minimal value; Max: maximal value; Av: average value; number of targets: all visited targets in the total reoptimized tour; computation time of the entire reoptimization process*)

Algorithm	Slack value	Delay factor	Number of targets			Computation time (ms)		
			Min	Max	Av	Min	Max	Av
IA	15	5	12	16	13.459	0	16	0.1
	20	6	10	15	12.104	0	16	0.2
HRA	15	5	10	15	12.725	15	67231	1160.2
	20	6	9	17	14.646	0	900140	20976.6
RRA	15	5	12	16	14.014	187	60861	7882.2
	20	6	12	16	13.792	359	91610	25427.3
ITA	15	5	9	16	10.447	0	375	8.6
	20	6	8	10	8.091	0	171	0.7
DA	15	5	13	17	14.962	31	41766	810.7
	20	6	13	17	14.779	31	2549001	3809.1