AI-Based Mission Planning for High-Altitude Pseudo-Satellites in Time-Varying Environments

Jane Jean Kiam

Vollständiger Abdruck der von der Fakultät für Luft- und Raumfahrttechnik der Universität der Bundeswehr München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Gutachter/Gutachterin:

1. Univ. Prof. Dr.-Ing. Axel Schulte
2. Prof. Dr. Eva Besada-Portas

Diese Dissertation wurde am 24.06.2019 bei der Universität der Bundeswehr München, 85577 Neubiberg eingereicht und durch die Fakultät für Luft- und Raumfahrttechnik am 01.11.2019 angenommen. Die mündliche Prüfung fand am 12.12.2019 statt.

Abstract

The success stories of High-Altitude Pseudo-Satellites (HAPS) from the previous test flights have led to a serial production of the platform, ready to be deployed as a flexible alternative to satellites with fixed orbits. With the current flight endurance record of almost 26 days, the platform is suitable for long-term missions, which are to monitor ground activities continuously from the stratosphere.

The challenges HAPS have to face are numerous, mainly due to the dynamic environment and the time-dependent mission requirements that the light-weight platform with energy-efficient electro-motors can hardly cope with. For the deployment to be commercially viable (i.e. reducing manpower in operation, increasing mission success rate etc.) and for safety purposes, increasing the degree of automation is essential. This is applicable for the mission planning aspect as well.

This work proposes an automated mission planner, of which the goal is to advice the HAPS operator to make fast and right decisions during planning prior to mission execution, working thus towards a “single-operator, multiple-HAPS” setup.

Some insights of the platform and the ongoing development of airspace regulations for stratospheric unmanned aircraft are first provided. Deriving from the trend, a realistic mission scenario is conceived, in which HAPS are commanded to monitor ground activities at different areas. Time-varying environment (i.e. weather, airspace availability etc.) and time-dependent mission requirements are realistically stated as well.

Given the complexity of the mission planning problem, it is dealt with module-wise. The most insidious problem in the estimation (during planning) of the spatial and temporal states of the HAPS is foremost tackled using a flight path planner, which can solve a kinodynamic path planning problem in a vector field. A hierarchical task planner guided by a genetic algorithm is integrated as a high-level planner to decompose the problem into smaller ones that the flight path planner can solve. To avoid frequent need for a replanning, a plan repair via reactive avoidance can be triggered in the presence of unforeseen but scarce obstacles. The planning methods are implemented; the functionalities and feasibility are validated using a six degree-of-freedom HAPS simulator and real historical weather data.

Zusammenfassung

Die Erfolgsgeschichten der High-Altitude Pseudo-Satellites (HAPS) aus den bisherigen Testflügen haben zu einer Serienproduktion der Plattform geführt, die als flexible Alternative zu Satelliten mit festen Umlaufbahnen eingesetzt werden kann. Mit dem aktuellen Flugdauerrekord von fast 26 Tagen eignet sich die Plattform für Langzeitmissionen, die die Bodenaktivitäten von der Stratosphäre aus kontinuierlich überwachen sollen.

Die Herausforderungen, denen sich die HAPS stellen müssen, sind vielfältig, vor allem aufgrund des dynamischen Umfelds und der zeitabhängigen Missionsanforderungen, die die leichtgewichtige Plattform mit energiesparsamen Elektromotoren kaum bewältigen kann. Für einen wirtschaftlich sinnvollen Einsatz (d.h. Reduzierung des Personaleinsatzes im Betrieb, Erhöhung der Missionserfolgsrate etc.) und aus Sicherheitsgründen ist eine Erhöhung des Automatisierungsgrades unerlässlich. Dies gilt auch für den Aspekt der Missionsplanung.

In dieser Arbeit wird ein automatisierter Missionsplaner konzipiert und entwickelt, dessen Ziel es ist, den HAPS-Betreiber zu beraten, um vor der Missionsdurchführung schnelle und richtige Entscheidungen bei der Planung zu treffen und so auf ein "Single-Operator, Multiple-HAPS"-Setup hinzuarbeiten.

Zunächst werden einige Einblicke in die Plattform und die Weiterentwicklung der Luftraumregelung für stratosphärische unbemannte Flugzeuge gegeben. Aus dem Trend abgeleitet wird ein realistisches Missionsszenario konzipiert, in dem HAPS zur Überwachung der Bodenaktivitäten in verschiedenen Gebieten befohlen werden. Dabei werden auch sich ändernde Umgebungsbedingungen (d.h. Wetter, Luftraumverfügbarkeit etc.) und zeitabhängige Missionsanforderungen realistisch angegeben.

Aufgrund der Komplexität des Missionsplanungsproblems wird es modulweise behandelt. Das tückischste Problem bei der Abschätzung (während der Planung) der räumlichen und zeitlichen Zustände des HAPS wird vor allem mit einem Flugbahnplaner angegangen, der ein kinodynamisches Pfadplanungsproblem in einem Vektorfeld lösen kann. Ein hierarchischer Aufgabenplaner, der auf einem genetischen Algorithmus basiert, wird als High-Level-Planer integriert, um das Problem in Kleinere zu zerlegen, die der Flugwegplaner lösen kann. Um eine häufige Neuplanung zu vermeiden, kann bei unvorhergesehenen, aber seltenen Hindernissen eine Planreparatur durch reaktive Vermeidung ausgelöst werden. Die Planungsmethoden werden implementiert; die Funktionalitäten und die Machbarkeit werden mit Hilfe eines Sechs-Grad-Freiraum-HAPS-Simulators und realen historischen Wetterdaten validiert.

Acknowledgements

Many thanks to Prof. Axel Schulte, who has provided me a research environment. A teacher who is cautious enough to reject my absurd ideas, but also encourages me to pursue other crazy ideas. A wise advisor who guides me through reality.

Many thanks to Prof. Eva Besada-Portas, who has been very committed to advise me. Your meticulousness is invigorating.

Many thanks to the passionate researchers from the AI community, especially Enrico Scala, who continues to answer many novice questions from me, Miquel Ramirez Javega, whom I have never met in person, but gave precious hints to my works, Prof. Ramon López de Mántaras, who has been an inspiring figure, and many more that I might have briefly crossed at several occasions, but your words guided me.

Many thanks to my colleagues and StraVARIA partners, who have helped me whenever I needed, and who patiently listen to my rather poor German. Thanks to Franziska Funk, who has been a dear colleague, tolerating officemate and friend throughout the thesis.

Many thanks to my students, who helped me in many investigations and discussed their findings with me.

Many thanks to my friends, who joked with me and who tried to fit to my schedule when I was too engrossed in work.

Many thanks to my unconventional parents, who have always supported me, but questioned me at the same time. You bring me back down to Earth after every roller coaster ride; you helped me make conscious choices.

Contents

1 Introduction

Civil Unmanned Aerial Systems (UAS) possess a total market estimated at $73,5 billion over the next decade, making the systems one of the most exciting market-pull developments. The market for low-cost High-Altitude Low-Endurance (HALE) will emerge to almost $2 billion by 2026, and are dominated by big players like Airbus, Facebook and AeroVironment, which develop long-endurance stratospheric UAS [Finnegan, 2017].

HALE systems are used as remotely operated UAS, which are commonly employed to pursue tasks in which the presence of humans on board would result in being uneconomical, uncomfortable or hazardous. Solar-powered HALE are also commonly known as High-Altitude Pseudo-Satellites (HAPS). An example HAPS platform can be seen in Figure 1. The main applications of the platform consist of providing internet to rural areas or to monitor ground activities continuously. Even if HAPS could be deployed by the military, HAPS fall in general into the category of "low-cost" systems [Finnegan, 2017], meaning cost efficiency is critical, which does not only concern the production cost, but also the operation cost; the latter is a motivating factor to increase autonomy by reducing piloting to cut the continuous fix operation costs and thereby *improve the economic viability* [Johnson et al., 2017].

HAPS are very often constructed using light-weight material and motors with modest thrust to reduce the bulkiness and to increase endurance. These properties penalize however the maneuverability, because the platform is more sensitive to its time-varying physical environment, i.e. wind, convection, icing, turbulences etc. The dynamics also depend on the wind vector field, which cannot be neglected since wind magnitude can be of the same order of magnitude as the airspeed of a HAPS. Similar challenges are also experienced by Autonomous Underwater Vehicles (AUVs). The case of HAPS is but more challenging; as a fixed-wing platform, it cannot halt in the air. Therefore, apart from the economic benefits, increasing the autonomy of HAPS operation is also *desirable for safety purposes*.

Figure 1. Zephyr 7 during launch ©Airbus Defence and Space GmbH

Many works attempt to solve the notorious path planning problem for a vehicle subject to constraints of its dynamics in a vector field [Lolla et al., 2012] (which could or could not be applied to HAPS). The path planning problem is even more challenging when the vector field varies over time. Although some solutions have been proposed, even in the case where static and dynamic obstacles are present [Lolla et al., 2015; Otte et al., 2016], little is documented on a more general mission planning problem, in which the vehicle is required to *carry out tasks while subject to inhomogeneous constraints expressed at different abstraction levels for different purposes* (e.g. mission constraints, environment constraints, platform-related dynamics constraints etc.). Some of these constraints are even *known only partially* at the moment the plan is computed, for example information concerning the time-varying environment (e.g. wind field, critical weather zones etc.). This can be even more challenging in a *large- scale operation area* for *missions that span over long durations*, since the complexity of most path planning methods increase with the state space and time domain.

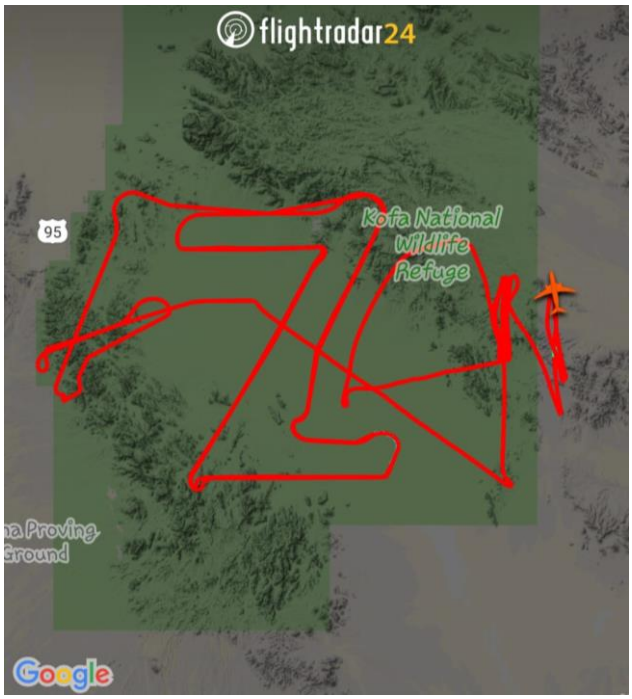


Figure 2. A screenshot on the live flight path of Kelleher in summer 2018 © flightradar24

This work presents a *framework that intends to solve the HAPS pre-execution mission planning problem*[1], i.e. mission planning performed prior to mission execution, which is essential since HAPS operate in a managed airspace [Everaerts and Lewyckyj, 2011], in which case the flight plans are often required to be communicated in advance.

1.1 Typical HAPS Platform and its Challenges

This work focuses on solar-powered HAPS that are designed to stay in the lower stratosphere over long periods to carry out missions but are confronted with challenges mentioned above.

The term HAPS was coined to refer to the class of light-weight long-endurance UAS that are a viable and more flexible alternative to fixed-orbit satellites in a number of applications [Klöckner, 2016] due to their extreme endurance. The HAPS depicted in Figure 1, Zephyr 7, was tested in 2010 and holds the record for a continuous unmanned flight of 14 days. Later in summer 2018, Zephyr S was tested again under the name "Kelleher" in Arizona and broke the record of its predecessor with a continuous unmanned flight of 25 days, 23 hours and 57 minutes[2].

Using the Automatic Dependent Surveillance – Broadcast (ADS-B) data available in JSON[3] from ADS-B Exchange[4], some insights of the flight paths and flight performance of Zephyr S during the test in Arizona above Yuma (see Figure 2) within an area of about 100 km × 50 km[5] can be viewed. Note that the flight performance analysed here is derived from the observed position data from the test flight of the Zephyr S (dubbed Kelleher). More general parameters of a HAPS will be discussed in the following section. Although the analysed data might not be directly relevant for this work, and the interpretation has to be done without insider information of the missions, some platform-specific properties can be recognized, which are important to justify the implemented framework.

Figure 3 shows the flight path of the ascending flight during the first test day and the flight path of the descending flight right before landing on the last day. It is interesting to observe, from Figure 3a and Figure 3c that the lateral flight paths (i.e. flight path projected on the latitude-longitude plane) are rather random. The climb from 6 km to the operation altitude of 18 km takes more than 4 hours, as seen in Figure 3b. The ascending phase is almost uninterrupted, with steady increase of altitude, which is expected, since the start can be planned to take place under perfect weather conditions or delayed if any difficult weather conditions appear. The descent of 8 km takes more than 8 hours, as seen in Figure

[1] The term "offline mission planning" is avoided here since the usual "offline" planning is performed even before the operation has begun. However, HAPS operates continuously; but mission planning is required at certain intervals before the execution of the mission-related tasks, much like mission planning for satellites.

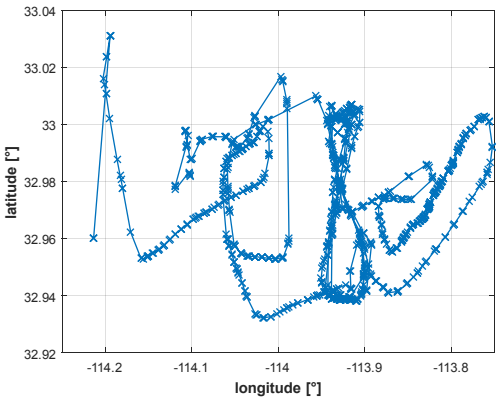
[2] https://www.airbus.com/newsroom/press-releases/en/2018/08/Airbus-Zephyr-Solar-High-Altitude-Pseudo-Satellite-flies-for-longer-than-any-other-aircraft.html (last visited on 4 March 2019)

[3] JSON (JavaScript Object Notation) is a text data format for easy reading and writing manually by a human and efficient to parse and generate by a machine. It is object-oriented and comparable to, but more compact than XML (Extensible Markup Language).

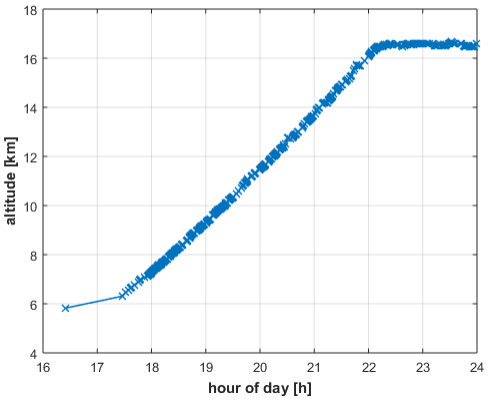
[4] https://www.adsbexchange.com/ (last visited on 4 March 2019)

[5] As a rule of thumb, 1° in the World Geodetic System 1984 (WGS84) is about 110 km.

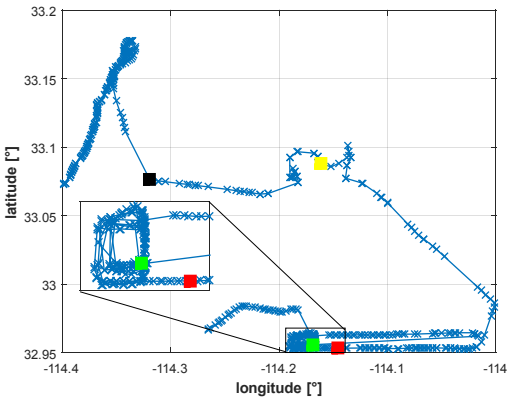
3d, which could be due to more challenging circumstances, in which a safe landing is required as soon as possible. The vertical profile of the flight path shows that the platform was required to fly laterally (see the phases of level flight) for a couple of times (start and end of lateral flights are labelled with the colored markers), while the lateral profile shows that the aircraft travelled, during the lateral flights, often also from one area to the other (also labelled with the corresponding colored markers), which suggests that Kelleher was trying to access other safe vertical corridors.



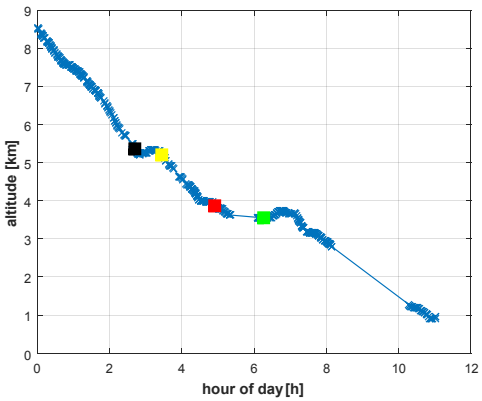
a. Flight path projected on the latitude-longitude plane during the ascending flight on the 11th July 2018



b. Flight path projected on the vertical plane during the ascending flight on the 11th July 2018

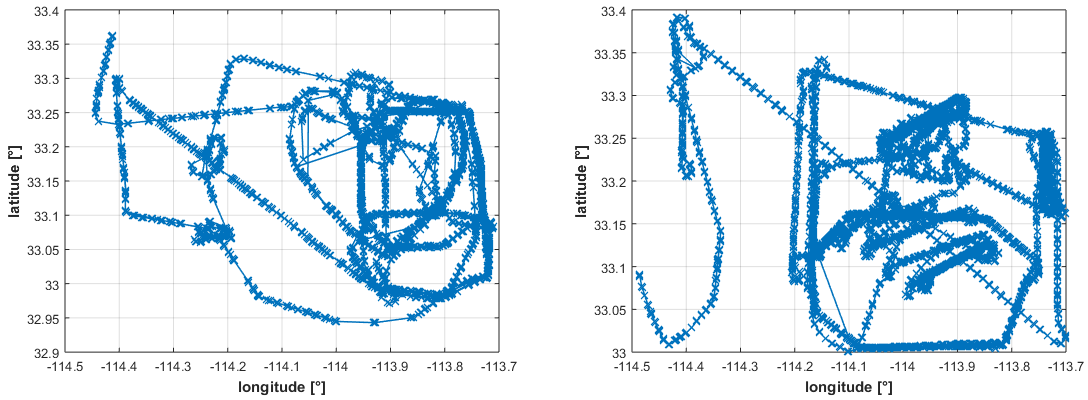


c. Flight path projected on the latitude-longitude plane during the descending flight on the 6th August 2018



d. Flight path projected on the vertical plane during the descending flight on the 6th August 2018

Figure 3. Ascending flight of Kelleher on the first day of test (11 July 2018) and descending flight before landing on the last day (06 August 2018)



a. Lateral flight path on 21st July 2018

b. Lateral flight path on the 30th July 2018

Figure 4. Flight paths projected on the latitude-longitude plane during mission flight on different days

The "random-walk" observed from the flight paths are not limited to lower-altitude flight during start and landing (see Figure 3a and Figure 3b), but also during the routine flight at operation altitude, as seen in Figure 4, in which flight paths projected on the latitude-longitude plane of different days are depicted. A random flight path profile similar to a glider plane is expected: the wind effect on the aircraft cannot be completely neglected, since the HAPS is equipped with weak electro-motors for energy-saving purposes.

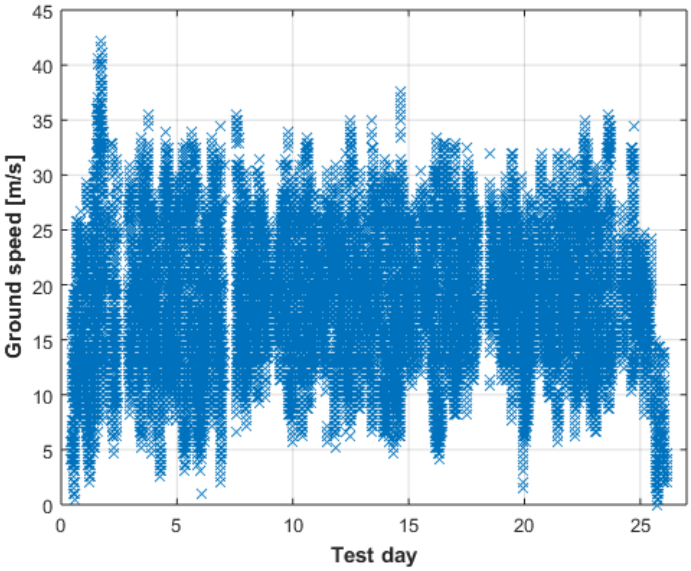
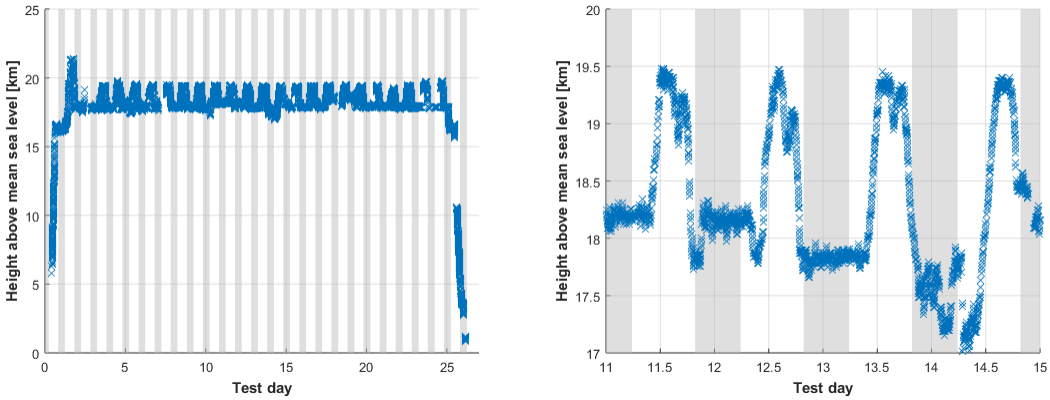


Figure 5. Ground speed of Kelleher obtained from ADS-B data (Test day is relative to local time)

Furthermore, with the weak electro-motors, the aircraft may fly at airspeeds comparable to that of a racing cyclist. This analogy can be deduced from Figure 5, in which the ground speed obtained from ADS-B ('Spd' in the JSON data field) during the complete test and converted from knots to m/s (to comply with the standard metric units

used in the rest of the work) is shown. The test day on the x-axis is computed relative to local time (UTC-7 in Arizona). The ground speed of the platform at the operation altitude varies between 5 m/s and 35 m/s, with a mean value of about 20 m/s, while the wind speed at the operating altitude of HAPS is normally of the range of 0-5 m/s and could be up to 10 m/s or more [Brasefield, 1949].



a. Height above mean sea level during the test with respect to the test day in local time

b. Partial zoom-in: Climb after sunrise and descent before sunrise (evaluated at local time)

Figure 6. Vertical flight profile of Kelleher; delimited in gray are the sunset and sunrise time

Another interesting feature is the vertical flight profile over time of the aircraft, as illustrated in Figure 6. The height of the aircraft oscillates periodically, a characteristic commonly known as the "yo-yo" flight [Klöckner, 2016]. In the partial zoom-in in Figure 6b, the climb takes place after sunrise, when the sun is way above horizon to illuminate the solar panels of the HAPS more efficiently, while the decent of about 1-1.5 km is observed slightly before sunset, or rather once the sun is no longer high above the HAPS.

As a fixed-wing aircraft equipped with weak motors, HAPS is also expected to have limited dynamics. The track angle is available from the ADS-B data ("Trak" in the JSON data field). Although the track turn rate is not exactly the turn rate of an aircraft [Beard R. W. and McLain, 2012], the first order derivation with respect to time of the track angle data provides valuable information on the dynamics of the aircraft, which can be obtained using the simple formulation: $\big(\chi(t+\Delta\mathrm{t})-\chi(\mathrm{t})\big)/\Delta\mathrm{t}$. However, due to the numerous anomalies in the ADS-B data obtained, e.g. abrupt jumps in the track angle values, missing epochs etc., the first order derivation only takes into account consecutive data with a time difference equal to $\Delta t = 1$ min, which is the time interval of the ADS-B data. Plotted in Figure 7 is the cumulative occurrence probability of the track turn rate:

$$P_{occ}(\dot{X}) = \int_0^{\dot{X}} p(\dot{\chi})\, d\dot{\chi}\,. \qquad 1\text{-}1$$

Due to the numerous abrupt jumps in the track angle values, some track turn rates are unrealistic. But the plot shows that almost 90% of the track turn rates are below 3°/s.

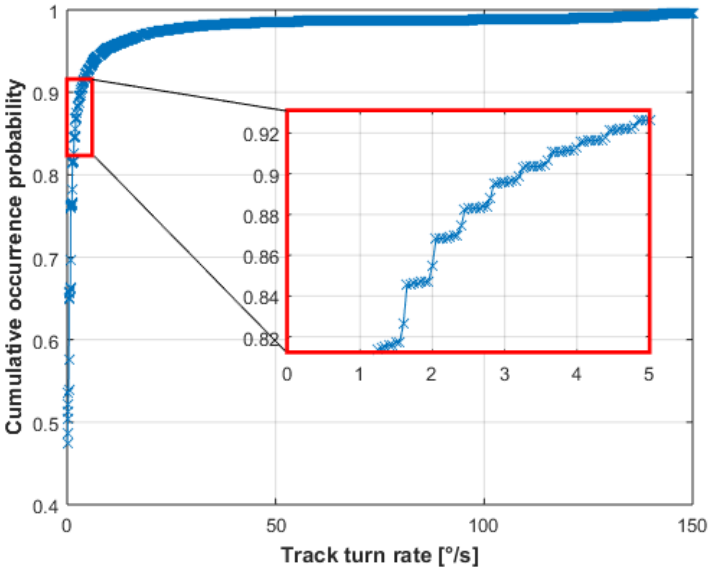


Figure 7. Cumulative occurrence probability of the track turn rate for Kelleher

The data analysis of Kelleher is although based on rather incomplete data, and represents only a specific platform, from which even its future-generation build is likely to deviate, it does provide insights on how the cutting-edge solar-powered HAPS performs airborne, from which the challenges it could face can be deduced.

Table 1. Properties of a HAPS and the challenges in planning it poses

Properties	Challenges during operation
High operating altitude	The planning or takeoff and landing can be time consuming.
Long endurance (continuous operation for weeks or months)	Operating cost is high, and the ground crew can be distracted while performing repetitive tasks.
Light-weight	Sensitive to adverse weather conditions (see Section 1.2).
Low airspeed	Wind effects cannot be neglected, and critical weather zones cannot be assumed static.
Fixed-wind and large wingspan	Limited maneuverability.
Battery capacity	Energy management and the use of motors with low-energy consumption are necessary.
Energy-efficient electro-motors	Difficulty to navigate in a wind field.
Limited payload	Passive sensors for onboard cloud detection and for monitoring missions, as well as limited onboard computation power.

As summarized in Table 1, the operation of HAPS faces challenges that arise either due to how HAPS are deployed or due to the physical build of the platform. The long-endurance operations suggest that manpower be reduced, or the operation costs could be too high to sustain. The light-weight build of the platform makes it susceptible to critical weather conditions, such as turbulences, or thunderstorm; however, for the sake of energy efficiency, the weak motors and the large wingspan (designed also to accommodate the

solar panel) limit the dynamics or the platform, either in terms of airspeed or maneuverability, e.g. limited turn rate, thereby making it difficult to fly around critical weather zones swiftly. Critical weather must also be avoided laterally in advance, to avoid frequent emergency landing in case of confinement within critical weather zones, since the takeoff and landing take hours. Besides, the avoidance must be performed principally laterally, since the vertical profile is regulated, as shown in Figure 6. Furthermore, mission-related tasks are time-dependent, i.e. either the tasks must be executed at within specific time slots as requested, or the airspace to execute the tasks is accessible only at specific time slots, due to the time-varying weather conditions. It is therefore essential to plan the operation of HAPS, not only with positional navigation, but also with a proper estimation of the time of arrival (ETA: Estimated Time of Arrival).

1.2 Hazardous Weather for HAPS

While HAPS operate at rather calm altitudes of ~ 18 km in the lower stratosphere[6], they are still subject to a multitude of hazardous weather conditions which are ought to be either predicted or detected and avoided during operations. Below is a non-exhaustive list of weather-related hazards a HAPS must avoid in order to prevent impairment.

1.2.1 Cumulonimbus clouds

Cumulonimbus (Cb) clouds are structures to be avoided by all means, as they account for turbulences, lighting, heavy precipitation and hail [Airbus, 2007]. Due to the aggressiveness of the deep convective structures, typically, they are to be cleared by 20 NM (~ 37 km) laterally and 5000 ft (~ 1.5 km) vertically. Although most part of cumulonimbus clouds are found in the troposphere, the top, or rather the anvil can overshoot and reach the lower stratosphere, as evidence from satellite images shows [Bedka et al., 2010].

1.2.2 Turbulences

Turbulences occur when volumes of air moving at different speeds meet. Although rare in the lower stratosphere, turbulences are not excluded, and are mostly caused by wind shear [Leena et al., 2012]. While turbulences are rarely dangerous for bigger and more robust airplanes, they could be threatening for HAPS due to the light-weight structure, and complicate the navigation of HAPS, i.e. the ability of the HAPS to follow a planned path might be impaired.

1.2.3 Precipitation/Hail

Precipitation, especially in form of hail is damaging for the solar panel mounted on the wings of the HAPS. According to [Airbus, 2007], hail is most likely (> 80%) to be encountered inside a cumulonimbus cloud above Flight Level (FL) 200 (~ 6km), and therefore could also be found in the lower stratosphere.

1.2.4 Wind gusts/Strong wind

Although wind in the lower stratosphere is calmer than in the troposphere and also than the higher stratosphere, wind of magnitude up to 5 m/s is usual, and can at some times of the year be more than 10 m/s [Brasefield, 1949]. Mild wind can be exploited as extra

[6] The lower stratosphere is separated from its lower layer, the troposphere, by tropopause, which marks a cease to the temperature gradient with respect to height. The stratosphere starts at about 20 km at the equator and can start as early as about 10 km near the North pole [Andrews et al., 1987].

thrust, when it is a tail wind; however, given the low airspeed of the HAPS, airspace of strong wind should be avoided, so that the aircraft remains under control.

1.3 Weather Data

In general, aviation weather data stems from two sources:

1) third-party meteorological nowcast and forecast data provided by services on the ground that can be used for route planning,
2) observed nowcast data during the flight provided either by on-board sensors or other pilots.

1.3.1.1 Third-Party Meteorological Data

Flight meteorology, in short, consists of wrapping meteorological data into weather forecast data made relevant and available for flight operations. The task is usually carried out by an aviation weather service and overseen by the federal airspace authority, e.g. the Federal Supervisory Authority for Air Navigation Services in Germany (BAF, Bundesaufsichtsamt für Flugsicherung, BAF). The German Meteorological Office (DWD, Deutscher Wetterdienst) is in Germany the only certified aviation weather service. The following of this subsection intends to provide an overview of the aviation data in Germany. Although not identical, other countries adopt a similar system, for it is necessary to conform with the international standards set by International Civil Aviation Organization (ICAO).

The aviation weather data provided by DWD are in general divided into three categories, namely weather warning, weather nowcast from current observation data, and weather forecast, some of which are briefly described in the list (non-exhaustive) in Table 2 [Deutscher Wetterdienst, 2015; Deutscher Wetterdienst, 2018].

Advisory weather information from Table 2 is wide-area information, meaning it cannot be used to compute detailed flight plan, but only for dictating if an operation can or cannot take place. Apart from aviation-specific advisory weather information, other numerical global weather data derived from sophisticated atmospheric fluid-dynamics and thermodynamics models are also available. Complex mathematical tools to solve partial differential equations, fuzzy logic etc. are used to extrapolate or improve the atmospheric model. The accuracy and resolution of numerical global weather data has benefited a lot from the computing power of modern CPUs. Table 3 is a short list of the most commonly used numerical global weather data. These data are four-dimensional, i.e. three-dimensional spatial and time discretization. Some of them are of much higher resolution, for example, the COSMO-DE with a ~ 2.8 km resolution, or its successor, CODMO-D2, with a ~ 2.2 km resolution, are provided by DWD as advisory weather data for air balloons, or ultra-lightweight aircrafts. Such numerical weather data is in general practical for the tedious planning of weather sensitive platforms, and therefore is suitable to be used for HAPS as well in its mission planning.

Table 2. Weather data products included in the aviation weather service provided by DWD

Warnings	**Description**
SIGMET	The SIGMET (SIGnificant METeorological phenomena) warns flights of hazardous weather conditions such as severe turbulences, heavy hail, or aggressive thunderstorm zones. The planning or takeoff and landing can be laborious. DWD provides the SIGMET data by broadcasting findings from regional stations. Data from SIGMET is valid for four hours.
Warnings for GAFOR areas	Warnings of dangerous weather zones in the corresponding GAFOR (see GAFOR in the "Forecast" list) areas.
GAMET	Updated four times a day with each data having a validity of six hours, GAMET (General Aviation METeorological information) complements the GAFOR warnings for low-altitude flights or near the mountains.
AIRMET	AIRMET (AIRman's METeorological information) is not updated regularly but only broadcasts weather alerts for low-altitude flights which are missing in GAMET. The validity lasts until the next update of GAMET data.
Forecast	**Description**
TAF	Coded similarly to METAR (see below in the list of "Nowcast"), TAF (Terminal Aerodrome Forecast) provides weather forecast at the airport areas. Updated every three hours, each set of forecast has a validity of 9 hours, or 18 hours for international flights.
GAFOR	GAFOR (General Aviation FORecast) is an area weather forecast used for Visual Flight Rules (VFR), and is updated every three hours and each dataset is valid for the next six hours. Germany is divided into 68 GAFOR-areas.
3-day forecast	Once a day, a 3-day forecast is provided for each regional in Germany (North, South and Middle), and is used mainly to predict conditions for VFR and aviation sports.
Nowcast/ Observation	**Description**
METAR	METAR (METeorological Aerodrome Routine weather report) is a coded standard weather report on visibility, weather and clouds, derived from observations obtained from weather stations placed at airports. Each weather dataset is valid for two hours and is updated every half an hour at each airport.
SAT IR (WAFS)/ Sat Europa HRV, RGB, IR/ SAT-Bild GOES-E, SAT-Bild Himawari, etc.	Satellite imagery using various satellites (WAFS, Europa, GOES-E, Himawari etc.) equipped with either InfraRed (IR) sensors, true-color sensors (Red-Green-Blue, RGB), or High-Resolution Visible (HRV) image sensors can provide observation images of the weather conditions up to every quarter of an hour with high-resolution up to 3 km.
Analysis map	Numerous ground-based weather stations across the country provide isobaric maps every three or six hours.

Table 3. Numerical global weather data

Model	Description
GFS	Based in the USA, GFS (Global Forecast System) is operated by the National Center for Environmental Prediction (NCEP) and managed by the National Oceanic and Atmospheric Administration (NOAA). The planning or takeoff and landing can be laborious. GFS is based on the atmospheric, oceanic and Earth model. Some data are made available to public[7]; they are of different update rates and of different spatial-temporal resolutions. For shorter-term forecast, a spatial resolution of 28 km is available and for longer term (14 days) forecast, a resolution of 70 km is available.
ECMWF	ECMWF (European Center for Medium-Range Weather Forecast) is a numerical weather forecast and remote sensing research facility and service provider created by a joint-force of 34 nations. ECMWF provides forecast for different time periods [ECMWF, 2018]: *medium-range*: updated twice daily with high-resolution forecast data for up to 10 days *extended-range*: updated twice weekly with forecast data that tracks above all the weekly weather changes for the next 46 days *long-range*: updated monthly with forecast data that summarizes predictable trends over long periods for the next 7 months, such as El Niño Southern Oscillation[8] [Stockdale et al., 2017]
COSMO-DE/ COSMO-D2 [Baldauf et al., 2011]	COSMO-DE (COnsortium for Small-scale MOdeling) a weather forecast model of the DWD covering only Germany, Austria, Switzerland and parts of the other neighboring countries. The geographical limitation is so that the forecast can support high-resolution model, which is beneficial for the forecast of cumulonimbus clouds for example, which are often only of a few kilometers in dimension, and therefore do not appear on a weather forecast with spatial insufficient spatial resolution. With the 421×461 horizontal grid cells over a total coverage of 1160×1280 km^2, COSMO-DE provides a weather forecast map with a horizontal resolution of ~2.8 km; it also provides 50 altitude levels. Since the computation complexity increases with increasing resolution, COSMO-DE provides only short-range weather forecast of 27 hours. COSMO-DE was replaced by COSMO-D2 in May 2018. COSMO-D2 has a larger coverage area of 1440×1590 km^2, with an even higher horizontal resolution of ~2.2 km and a total of 65 altitude levels [DWD, 2018].

The numerous sources of weather data provide valuable insights to the dangerous zones, which in the case of HAPS, must be considered during the flight, and also already

[7] GFS weather download. https://www.ncdc.noaa.gov/data-access/model-data/model-datasets/global-forcast-system-gfs (last visited 10 March 2019)

[8] El Niño Southern Oscillation (ENSO) is the periodic variation in winds and surface temperatures of the equatorial Pacific Ocean.

in the planning phase prior to execution (see Section 1.1). To abide by the IFR, HAPS operate only in the "safe-zone", i.e. the airspace complement of the no-fly zones predicted by the aviation advisory weather data. There, high-resolution weather data must be used, since HAPS is a sensitive platform. In this work, COSMO-DE data is primarily exploited; thanks to its high resolution to predict the behavior of HAPS in the time-varying environment, the COSMO-DE data is also able to capture cumulonimbus clouds, which are indeed a threat to HAPS. Furthermore, the high-resolution wind data from COSMO-DE can be used to predict the movement of the HAPS induced by the surrounding airflow, so that the absolute position of the HAPS (i.e. position relative to the ground) can be more precisely predicted (see Section 3).

1.3.1.2 In-Flight Weather Data

The global data is used for penetrating/navigating in the airspace [Airbus, 2007], like the landmarks and road conditions used by the car navigation system. However, weather conditions can be missed and therefore many aircrafts carry onboard weather sensors.

Table 4. On-board weather sensors

Sensor type	Description
Radar	Onboard weather radars can detect structures with heavy precipitation (e.g. rainfall, wet hail, wet turbulence, ice crystals, etc.), while they have difficulties in general to detect dry hail and dry snow since the due to small reflections [Airbus, 2007]. Dryer structures such as clouds, fog, wind, clear air turbulence, wind shear, sandstorms and lighting cannot be detected by radars. Weather radars are active sensors, i.e. energy consumption is higher (usually in the order of magnitude of a few hundred Watts [Honeywell, 2016], but is reduced to ~ 40 W with miniature weather radar [Garmin, 2018]. Furthermore, they are bulky and weigh in general more; even the most light-weight radars weigh at least a few kilograms. Given the relatively high energy consumption and bulkiness of this class of sensor, the use of a weather radar for ultra-lightweight UAS is rare.
RGB/ EO sensors	Electro-Optical (EO) or RGB sensors are passive sensors that provide visible images in true color. Using a segmentation method, outlines of clouds can be differentiated from the background [Funk and Stütz, 2017]. Two-dimensional cloud coverage map (i.e. percentage of area of each grid cell of the map covered by clouds) or a cloud map (i.e. map marked by cloud polygons) can be drawn to guide the UAV around the clouds in order to conform with VFR, or to identify "clear-sky" for more efficient optimal communication. Using Structure from Motion (SfM) techniques, a monocular camera can also be used to determine the distance of the cloud to the platform, thereby enabling three-dimensional cloud maps. This class of sensors is usually lightweight and has low energy consumption.
IR sensors	Much like the EO sensors, thermal InfraRed (IR) sensors are also passive sensors. Multiple works have proven that ground-based thermal IR sensors can effectively detect clouds [Nugent et al., 2009; Redman et al., 2018].

Be it active or passive, onboard weather sensors provide additional information on the weather in the proximity of the aircraft, and therefore are beneficial for weather avoidance purposes. Also known as "sense-and-avoid", this is comparable to the car driver's visual sensor to spot the danger missed by the navigation system and is useful for reactive avoidance. In Table 4, the features of the most commonly used on-board weather sensors are summarized.

1.4 Airspace Regulations

Air traffic management is mature and widely practiced in Class A airspace, that is delimited by an altitude of 18,000 feet above mean sea level and stops at FL600 (~ 60,000 fts) [Hunter, 2015]. But as technology pushes its limits, with promises of more commercial operations of HALEs above FL600, it is clear that technological and regulatory measures have to be considered to enable navigation, tracking and communications of this flight level, although it is still not sure yet "how" [Hunter, 2015]. Nevertheless, airspace above FL600 is currently relatively traffic-free, since airliners fly below, some airborne vehicles can still operate through or at this altitude, for example the North American X-15, F-15s, F-22s, Lockheed U-2s, Google Project Loon, space rockets, Virgin Galactic's Spaceship 1, XCOR's Lync, SpaceX, Armadillo [Hunter, 2015].

Currently, according to the Federal Aviation Administration (FAA), the airspace above FL600 is classified as Class E airspace, in which no Air Traffic Control (ATC) clearance or radio communication is required for VFR flight. However, as mentioned in the latest operational concept on Air Traffic Management (ATM) for UAS jointly conceived in November 2018 by the European Organisation for the Safety of the Air Navigation (EUROCONTROL) and the European Aviation Safety Agency (EASA) [EUROCONTROL and EASA, 2018], High-level Flight Rules (HFR) should apply for unmanned flights operating in the stratosphere, i.e. above FL600, although HFR is not yet developed.

According to the guidelines in [EUROCONTROL and EASA, 2018], High-level Flight Rules (HFR) must be compatible with Instrumental Flight Rules (IFR), with some additional requirements that could also apply. Although airspace above FL600 does not fall into the category of "controlled airspace", as according to the guidelines, the imposed compatibility with IFR implies that either the UAS be able to sense and avoid, or the airspace be systematically organized to prevent collisions. And in Europe, since the number of UAS operating at this altitude is relatively substantial, the airspace must be "managed", meaning *any UAS that intend to operate at this flight level must provide a flight plan that includes the platform type, contingency procedure, planned operation (navigation, route, level, etc.) and contact details*.

Most importantly, although rare since the missions are mostly long-endurance, any flight through the controlled airspace below FL600 must be communicated in advance, e.g. before landing, or a descent for collision avoidance.

1.5 Goal, Contributions and Outline of this Work

Recent successful tests on HAPS, particularly of Zephyr and Rainbow Solar[9], as well as upcoming tests from the competitor[10] show that the Technology Readiness Level (TRL) of similar platforms increases steadily [Müller et al., 2018]. Moreover, with the setup of a serial production facility for Zephyr[11], a vast deployment of the HAPS can be expected in near future. Continuous border surveillance, ground mapping applications using HAPS are expectedly the typical tasks, since the deployment is more flexible, as an alternative to Low Earth Orbit (LEO) satellites and aircraft that require frequent refueling. However, human operators are required on a 24/7 basis, since typical missions span over long periods and HAPS are intended to remain in the lower stratosphere continuously. From a safety as well as pragmatic point of view, increasing autonomy is essential in such continuous long-endurance operations to reduce manpower and human error.

Phases involved in a typical space flight operations start from planning, processing, departure operations, flight operations, return and landing, refurbishment and turnaround [Hunter, 2015]. Since HAPS is a long-endurance platform that is intended to remain airborne, planning must also be performed during flight. This work *focuses on increasing autonomy and efficiency in mission planning during flight operations, but before the execution of the mission-related tasks*. The main goal is to optimize mission success rate, while reducing the risk of replanning, by considering the predicted time-varying environment, as well as the platform constraints at the planning phase. Part of the work also *proposes a plan repair method* to fly around unforeseen danger without aborting the computed plan completely.

This work is organized as shown in Figure 8. We first lay out a mission scenario and state the problem descriptively as well as formally for a HAPS-like UAV (see Section "2. Problem Statement" in Figure 8). In this Section, also some widely-used planning methods will be presented and analysed according to its suitability for solving the HAPS pre-execution mission planning problem. Subsequently, in section "3. Fight Path Planning", the most obvious problem will first be dealt with, namely the flight path planning problem, which is essentially for planning numerically a flight path for the HAPS from a start to a goal position in a time-varying environment, while estimating the time of arrival as well. A flight path planning method is known for being inconvenient to include elements of task planning, which is usually performed by classical planners. In section "4. Hierarchical Task Planning for HAPS", a hierarchical task planning structure is proposed to consider the heterogeneous mission-related constraints and requirements as listed in Table 8 and Table 9). The plans determined by the hierarchical task planner are presented in form of a sequence of logical tasks to be executed at very roughly estimated times (due to the lack of consideration of numeric details), and will be "refined" by the flight path planner, forming what is often referred to a strategic-tactical modularized hybrid planner. The hierarchical task planner needs however a more efficient

[9] Chinese solar drone "Rainbow" passed its maiden flight for over 15 hours at 20 km during the first test flight. http://www.xinhuanet.com//english/2017-06/13/c_136363018.htm (last visited 22nd April 2019)

[10] Aurora 2018. Odysseus press release. https://www.aurora.aero/odysseus-high-altitude-pseudo-satellite-haps/ (last visited 8th March 2019)

[11] Airbus 2018. Serial production facility for Zephyr HAPS. https://www.airbus.com/newsroom/press-releases/en/2018/07/Airbus-opens-first-serial-production-facility-for-Zephyr-High-Altitude-Pseudo-Satellites.html (last visited 08 March 2019)

method to search for quality task plans, which will be described in section “5. Extension of the Mission Planner to Multiple HAPS”, since the combinations of tasks constituting the plans become innumerable. A strategy to reactively avoid sporadic unforeseen obstacles (in the planning) is determined by the mission planner as well, as a plan repair method to be executed on-board, in order to avoid frequent replanning. The computation of the strategy is described in section “6. Plan Repair via Reactive Avoidance”. In the last section “7. Implementation and Validation”, the implementation is described, followed by an analysis of the validation results and performance tests.

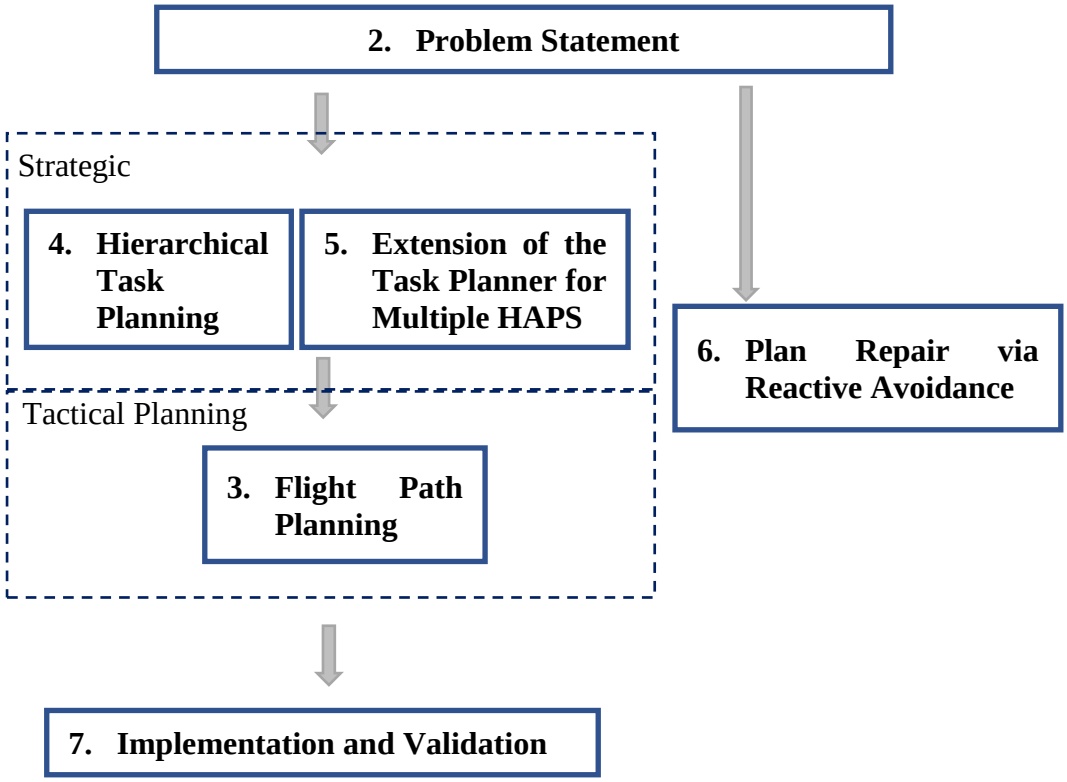


Figure 8. Roadmap of the work

"La technique est utile, mais elle est aussi bien plus qu'un outil.
Elle forme un système, où tout est interdépendant"
- Thierry Gaudin in *2100 Récit du Prochain Siècle*

2 *Mission Planning Problem for HAPS*

As demonstrated by the observation summarized in Section 1.1, HAPS is a lightweight, completely solar-powered, fixed-wing UAS that can stay airborne for weeks or even months. In order to be energy efficient, the platform is equipped with weak motors to cruise at relatively low speed in the lower stratosphere, where the airspace is relatively calm with mild wind and with relatively little congestion, since the airliners fly below. However, as listed in Section 1.2, although relatively rare, some critical weather may occur and must be avoided; the influence of the wind on the flight trajectory especially must also be taken into consideration, in order to predict the whereabouts of a HAPS at a given time in advance. Additionally, airspace regulations must also be considered to avoid collisions with other stratospheric aircrafts. To this end, a dynamic allocation of airspace is recommended [Cervo, 2014].

In this chapter, the in-operation pre-execution mission planning problem will be properly defined, first with the description of the mission scenario HAPS is confronted with, followed by a detailed analysis of the HAPS Mission Management System (MMS) (that wraps around the mission planner), in which the interactions of the mission planner with the external entities will be laid out. Subsequently, a formal description of the problem as well as the form of the solution are defined. Some related works that could be used for the mission planning for HAPS are described, including some more detailed description of the most prominent methods and analysis of why they should or should not be considered for solving the HAPS mission planning problem.

2.1 Mission Scenario of a HAPS

The studies in this work are based on realistic parameters of HAPS platforms in general, as described in [Müller et al., 2018]. On top of that, unless mentioned otherwise, a scenario is used throughout the work to study the mission planning problem for HAPS in a time-varying environment. The following paragraphs provide the details and thereby also draw up the scope of the work.

2.1.1 Platform Specifications

Table 5 summarizes the specifications of the build and flight performance of the HAPS on which this work is based [Müller et al., 2018]. The specifications of Airbus 320 are

also provided side-by-side in the table to serve as an intuitive comparison [Airbus S.A.S., 2005, Revised February 2019].

Table 5. Specifications of HAPS vs. Airbus 320

Build	**Parameter values**	
	HAPS	Airbus 320
Weight	100 kg	73×10^3kg
Wingspan	30 m	34 m
Payload/ Passenger capacity	5-10 kg	< 150 passengers
Battery capacity/ Range	15 kWh	
Electro-motor maximum propulsive power/ Engine thrust range	1700 W	111-120 kN
Flight Performance	HAPS	Airbus 320
Operating altitude	18 km	3-12 km
Cruise airspeed at the operating altitude	30 m/s	Mach 0.82 (~ 281 m/s)
Endurance	3 months	4800 - 5700 km

Each HAPS is tracked by an antenna on the Ground Control Station (GCS) to ensure line-of-sight communication.

2.1.2 Mission Payload

In this work, we assume that the HAPS is equipped with an EO mission camera. The advantage of an EO-camera is multifold:

1. It takes images in true color.
2. High-resolution camera models also exist in light-weight format.

Specifications of a mission camera are often given in form of the Field of View (FoV) of the sensor, and the pixel counts of the height h_{I} and width w_{I} of the image [Sun et al., 2016]. The specifications summarized in Table 6 are assumed for the EO mission camera mounted on the HAPS. These specifications are similar to MEDUSA designed by VITO under the ESA-PRODEX program for stratospheric solar powered UAVs [Delauré et al., 2013]. The camera is equipped with two custom designed CMOS image sensors, and is light-weight (~ 2.6 kg), consumes little energy (<50 W), and resistant to low pressure (down to 60 mbar) and to a wide range of temperature (-70°C to 60°C).

Table 6. Example parameters of a mission camera based on MEDUSA

$\boldsymbol{h_{\mathrm{I}}}$ **x** $\boldsymbol{w_{\mathrm{I}}}$ (px)	**Ground sampling distance at 18 km**	$\boldsymbol{h_{\mathrm{I}}}$ **x** $\boldsymbol{w_{\mathrm{I}}}$ at $\boldsymbol{h =}$ **18 km** at Nadir position	$\boldsymbol{f}$**, focal length**	$\boldsymbol{\mu}$**, pixel size**	$\boldsymbol{FoV_{\mathrm{H}}}$	$\boldsymbol{FoV_{\mathrm{v}}}$
(10000 px) × (1200 px)	30 cm	3000 ×360 m	330 mm	5.5µm	9.5°	1.1°

The conversion between the FoV and the focal length of a monocular camera can be found in Appendix 1.

The height and width of an image (h_I and w_I) on the ground taken by the mission camera vary according to the tilt angle of the pan-tilt unit, also known as the gimbal (see Figure 9). At an altitude of h, the format of the image can be determined geometrically with these equations, if the tilt angle of the pan-tilt unit, θ_{gimbal}, is smaller than $\frac{\pi}{2}$:

$$h_\mathrm{I} = h \cdot \left(\tan\left(\theta_{\mathrm{gimbal}} + \frac{FoV_\mathrm{V}}{2}\right) - \tan\left(\theta_{\mathrm{gimbal}} - \frac{FoV_\mathrm{V}}{2}\right)\right), \quad 2\text{-}3$$

$$w_\mathrm{I} = h \cdot \left(\tan\left(\theta_{\mathrm{gimbal}} + \frac{FoV_\mathrm{H}}{2}\right) - \tan\left(\theta_{\mathrm{gimbal}} - \frac{FoV_\mathrm{H}}{2}\right)\right). \quad 2\text{-}4$$

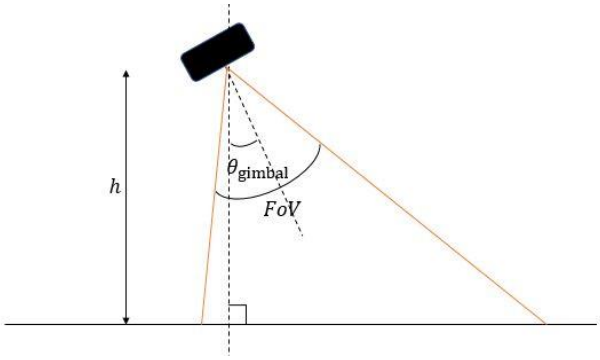


Figure 9. View of a camera mounted on a pan-tilt unit

Also assumed is the maximum horizontal tilt angle of the pan-tilt unit, which is 45°. At the maximum tilt angle, the horizontal dimension of the ground image can be up to 6 km. Although the flight pattern for optimal image coverage of the ground is not the core of this work, given the above mission payload specification, it is safe to assume that the image recording of an LoI (see Figure 10) can be performed in a lawnmower sweep pattern with a distance between stripes to be 2×6 km.

2.1.3 Deployment of HAPS in Monitoring Missions

HAPS are not yet in actual operation, although several tests were conducted. HAPS can be used either for communications relay, providing internet to rural areas or for ground activity monitoring and mapping [Airbus Defence and Space, 2017b]. However, communications relay, providing internet, or continuous monitoring/mapping of the same site require the platform to remain more of less static in the air. This can be better achieved with an "aerostatic" HAPS platform, or rather blimp, a semi-rigid large volume filled with helium [ESA, 2017]. The studies in this work are hence based on a ground activity monitoring mission of various locations within different time slots as required. Considered hereafter is only the mission flight, i.e. the effort does not study the launch and recovery of the platform.

Since flight operations in the stratosphere must also conform with IFR [EUROCONTROL and EASA, 2018], the airspace must be organized dynamically, so that the HAPS can carry out the missions without the risk of collisions, and also without occupying airspaces unnecessarily, which may hinder the sharing of the airspace with other stratospheric platforms, of which the number is expected to grow. It is recommended in [EUROCONTROL and EASA, 2018] that the Advanced Flexible Use

of Airspace (AFUA) concept [Cervo, 2014] be the guideline for the airspace organization of UAS, i.e. areas with defined lateral and vertical borders are allocated for the time the areas are needed for carrying out the related missions.

2.1.3.1 Mission Scenario for HAPS

Figure 10 shows a realistic continuous surveillance and mapping mission scenario, which will be used throughout this work for the mission planning studies, unless mentioned otherwise. The Locations of Interest (LoI) represented by green polygons mark the ground areas to be monitored in the lateral dimensions. According to the requirements imposed by the clients, some are to be monitored any time of the day, some only within certain timeslots of the day. The Mission Areas (MA) in blue encompass LOIs with the same requirements of the same client. LoI of the same client all bear the same set of requirements, and if these are fulfilled, a reward will be received by the HAPS team. The MA denote the allocated airspace for HAPS to carry out the monitoring/ mapping tasks at the operating altitude (~ 18 km). The LoI of a MA must be in the vicinity of one another and must be visited one after another as they define altogether a "mission" unit. For example, if the Deutsche Bahn request for the monitoring of the rail maintenance at two areas far apart, e.g. in Munich and in Bayreuth, the LoI cannot be grouped together into one MA. The Waiting Areas (WA) represented in yellow are airspace in which the HAPS can loiter freely while not in mission execution, e.g. at night (see Figure 6 for the typical vertical profile of the HAPS entering the loiter mode at sunset). A HAPS is allowed to move between MAs only through the designated corridors (C) or WA, implying also that MA are not to be used as "corridors". The reason being that in an MA, the HAPS should focus on fulfilling the task requirements without being bothered by collision avoidance with other HAPS, should multiple HAPS be involved in the mission.

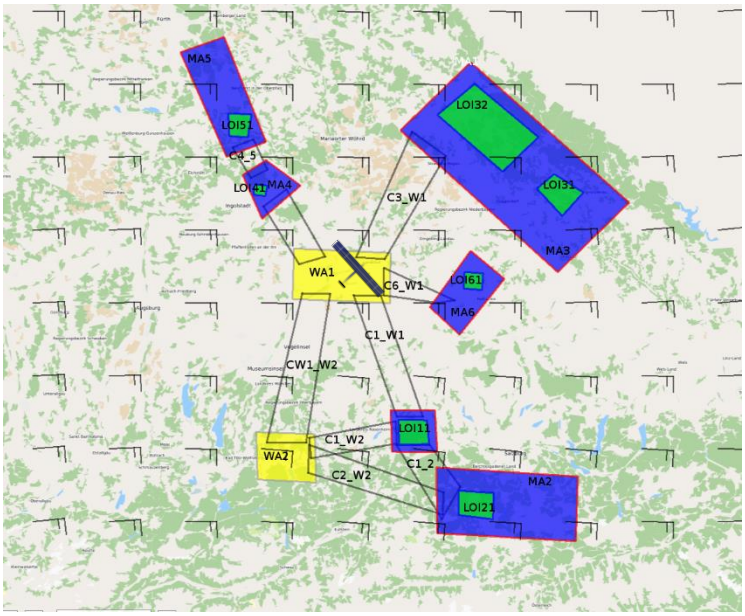


Figure 10. Mission Scenario for HAPS to be deployed for ground activity monitoring

The deployment of HAPS can be at a more massive scale, as also suggested by the serial production of Zephyr by Airbus[12], in order to increase coverage of the operation areas. In a scenario where multiple HAPS are involved, the "network" of MAs and LOIs can be expanded to the mission scenario shown in Figure 11.

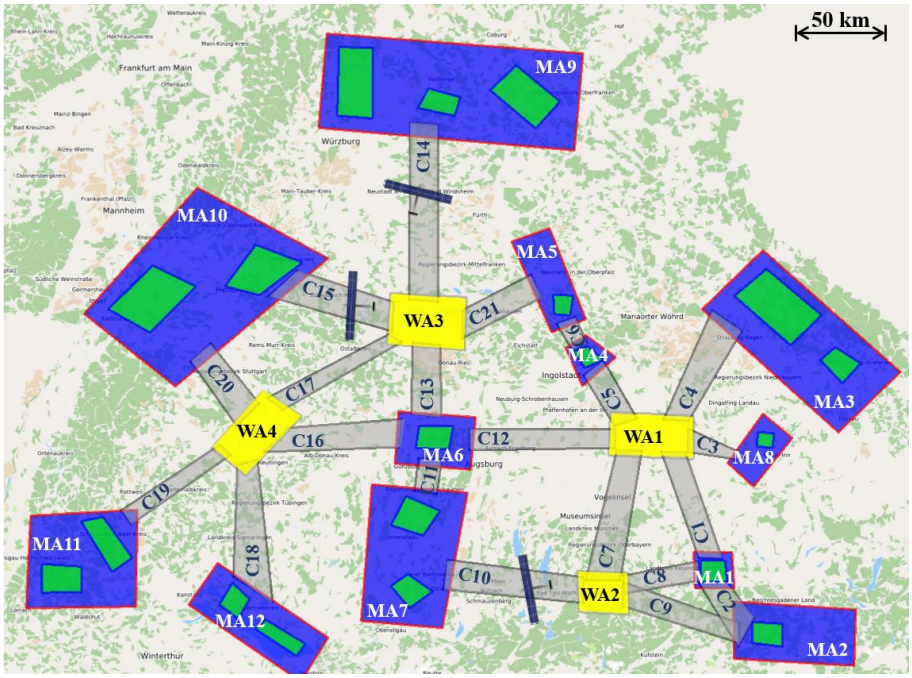


Figure 11. Typical scenario for monitoring missions with multiple HAPS

The dimensions of the mission elements depicted in Figure 10 and Figure 11 in form of their longest diagonal in kilometers are given in Table 7. Each mission element is available only within certain time windows, as required by the mission or as according to airspace availability. Furthermore, each LoI has a minimum revisit time. Therefore, time of arrival at the mission elements is closely relevant to the success of a mission plan. In the studies, it is assumed that the HAPS is equipped with an EO mission camera (see Section 2.1.2); therefore, the clouds between the operating altitude and ground can aggravate the mission success rate. Furthermore, for safety purposes, the HAPS is not allowed to operate in areas with high risk, i.e. substantial obstacle occlusion, neither is the platform allowed to operate in areas where the wind magnitude is substantial for fear of losing control of the aircraft.

In the rest of the work, a "mission" refers to the set of tasks to be carried out for all LoI of the same MA. "Mission Constraints (MC)" is used to denote conditions related to either operation safety or airspace regulations that must not be violated at all times, and "Mission Requirements (MR)" to denote the conditions to meet in order to succeed in a mission. The mission constraints and requirements described in the above paragraphs are summarized and listed in Table 8 and Table 9 respectively.

[12] Airbus 2018. Serial production facility for Zephyr HAPS. https://www.airbus.com/newsroom/press-releases/en/2018/07/Airbus-opens-first-serial-production-facility-for-Zephyr-High-Altitude-Pseudo-Satellites.html (last visited 08 March 2019)

Table 7. Dimensions of the mission elements: longest diagonals in kilometers

MA/WA	Longest diagonal [km]	Longest diagonal [km]			C	Longest diagonal [km]
		LoI 1	LoI 2	LoI 3		
MA1	27.58	17.10			**C1**	60.78
MA2	69.49	19.83			**C2**	37.83
MA3	103.11	20.07	44.27		**C3**	33.67
MA4	25.32	15.56			**C4**	58.88
MA5	52.90	13.25			**C5**	34.22
MA6	46.51	21.61			**C6**	17.64
MA7	84.03	21.42	25.88		**C7**	70.41
MA8	36.62	10.84			**C8**	40.27
MA9	133.30	34.36	21.24	38.13	**C9**	66.19
MA10	123.35	39.96	44.74		**C10**	72.10
MA11	74.98	31.62	24.53		**C11**	20.52
MA12	73.49	18.28	26.38		**C12**	73.35
WA1	47.90				**C13**	39.34
WA2	34.33				**C14**	88.46
WA3	46.41				**C15**	63.04
WA4	47.96				**C16**	70.70
					C17	71.98
					C18	71.38
					C19	61.77
					C20	42.75
					C21	46.67

Table 8. Mission Constraints (MC) for safety and airspace regulations

Identifier	
MC1	The HAPS is allowed to carry out its tasks in a MA or to loiter in a WA, if the obstacle occlusion in the area is less than 30%. The HAPS is allowed to cross a corridor only if no obstacle is present there.
MC2	The HAPS is allowed to fly in mission elements where wind magnitude $\lvert v_w \rvert$ is less than 5 m/s.
MC3	No more than one HAPS can coexist in a MA.
MC4	A MA cannot be used as a corridor (i.e. if a HAPS flies into the MA, it must try to carry out the mission).
MC5	There must be an existing corridor that connects to the previous mission element.
MC6	LOIs cannot be repeatedly monitored at each MA visit.
MC7	Weather critical zones and other airborne vehicle must be avoided (collision avoidance).

Table 9. Mission Requirements (MR) for successful monitoring

Identifier	
MR1	All the LoIs of the MA are visually recorded and the images are sent to the GCS.
MR2	Ground image coverage of each LoI of the MA is higher than the minimum coverage threshold.
MR3	Images captured of each LoI within each MA are taken within the corresponding monitoring time windows.
MR4	MA has not been (successfully) visited that day more times than allowed by its maximum revisit frequency.
MR5	Time-lapse between two consecutive visits to the same MA is higher than its allowed minimum inter-visit time.

It shall be noted that images of the LoI of the same MA are traded for rewards if all the MR in Table 9 are met (meeting all MR implies a successful mission). Should that be the case, the HAPS team will be rewarded according to Table 10. The rewards in this work are expressed in a financial form; without loss of generality, they can also be expressed in metrics that represent importance or urgency.

Table 10. Rewards to be given for each MA ($\times 10^3$)

Mission area	Coverage (%)	Reward (€)	Mission area	Coverage (%)	Reward (€)
MA1	80	4	**MA7**	70	15
MA2	80	50	**MA8**	80	3
MA3	60	100	**MA9**	60	13
MA4	80	20	**MA10**	60	18
MA5	80	3	**MA11**	70	20
MA6	70	5	**MA12**	70	10

2.2 System Analysis of the HAPS Mission Planning

A real-world system involves many actors and stakeholders, while the engineering of it (Systems Engineering) brings them together to achieve the underlying objectives [Beihoff et al., 2014]. This work focuses solely on the HAPS mission planning during mission flight and prior to execution (i.e. *pre-execution mission planning*), which is intended to plan for successful missions for the scenario described in Section 2.1.3. The mission planner is also a system unit incubated in the bigger HAPS Mission Management System (MMS). The following subsections provide an overview of the MMS that wraps around the mission planner for a better understanding of the role of the mission planner, its stakes and the expectation about the performance of the planner.

2.2.1 Mission Management System (MMS)

HAPS operate either in a controlled airspace [Everaerts and Lewyckyj, 2011] during the climb phase after takeoff and the descent before landing, or in a managed [EUROCONTROL and EASA, 2018; Cervo, 2014] during operation. The upcoming operation details (flight routes, navigation, level etc.) must be preplanned and communicated.

Due to the physical properties of a HAPS, as described in Table 5, their operation and planning is more challenging (see Table 1). To be taken into account in the mission planning are:

1. mission requirements, e.g. tasks, execution time and location (defined by a lateral area),
2. the dynamic allocation of airspace for operation,
3. weather condition and the avoidance of dynamic critical weather zones, and
4. fleet information and flight dynamics, e.g. number of aircrafts available, speed, turn rate etc. of the aircraft in the time-varying wind field.

Note that this work focuses on developing a mission planner for HAPS. Given the limited payload of the platform (5-10 kg), it is essential to limit the on-board equipment to only safety-critical real-time applications, of which the functions are not interrupted even during communication link loss. Modules of the MMS such as flight control and reactive guidance must be on-board, while long-term operational mission planning that works at fix intervals to plan or re-plan the missions prior to the execution in the next hours can be performed in the Ground Control Station (GCS). With this architecture, the ground-based mission planner will not be limited hardware-wise, as computation power is critical to process the weather data [Müller et al., 2018; Köhler et al., 2017b] and plan accordingly.

2.2.2 Top-Level Role Description of the HAPS MMS

Fully autonomous UAS is beyond reachable. ICAO stated in 2011 that "Remotely-Piloted Aircraft (RPA) […] will be integrated into the international civil aviation system in the foreseeable future", thanks to the presence of the remote pilots(s) that enables the interaction with the existing ATM system; however, the future of fully autonomous aircraft operations was not mentioned, which implies a longer wait period for the realization. The lack of clarity in the certification process of a fully autonomous UAS is also confirmed by Clothier et al. in [Clothier et al., 2013]. Early 2019, the first fully autonomous drone for surveillance application was approved by the French Directorate-General for Civil Aviation (DGAC, Direction Générale de l'Aviation Civile) [Rees, 2019]. Although not piloted by a human, the drone must still be supervised by a remote operator, for example a security guard without a pilot license.

Although the future operation of HAPS is not yet outlined in detail and could alter according to innovations, at least one human operator is almost always in the MMS (i.e. human-in-the-loop system), as suggested by the regulatory documents and report mentioned above. Furthermore, take Zephyr from Airbus Defence and Space (ADS) for example, the platform is currently remotely piloted and operated; nevertheless, the goal is to enable a single pilot to fly multiple Zephyr platforms simultaneously [Airbus Defence and Space, 2017a].

The MMS conceived for HAPS obeys hence the rule of having one (or more) human operator(s). Moreover, a modern autonomous, context-sensitive task management and decision support tool [Mosier et al., 2017] requires information management to increase the level of transparency and thereby also the level of trust without overloading the human operators, as well as and adaptive automation for adaptive aiding. The MMS designed for HAPS intends to comply with this standard, of which the design elements will be detailed in the following.

Used here is the diagram-based language developed by Schulte et al. [Schulte et al., 2016; Schulte and Donath, 2018] to describe a system with Human-Autonomy Teaming (HAT) in order to facilitate the comprehension at the top-level of a human-in-the-loop system, by capturing the work objectives (`WObjs`) of each work process (`WProc`) in an environment (`Env`). According to the axioms of systems engineering proposed by Felder and Collopy [Miller, 2018], a system must engage in a goal seeking behavior. Each `WProc`, according to [Schulte and Donath, 2018], produces work process outputs (`WPOuts`) that can be a subset of the `Env` of their own `WProc` or of another `WProc`. Note also that the `WObjs` of a `WProc` should be a subset of the `WPOuts`, as indicated also by one of the axioms proposed by Felder and Collopy in [Miller, 2018]. Figure 12 illustrates how the HAPS MMS, which is a system with HAT can be represented using notions defined in [Schulte et al., 2016].

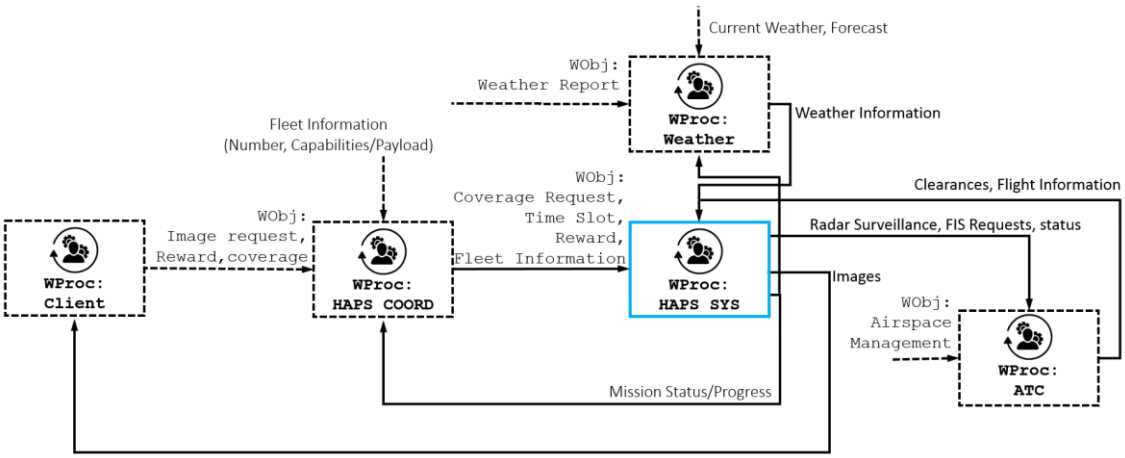


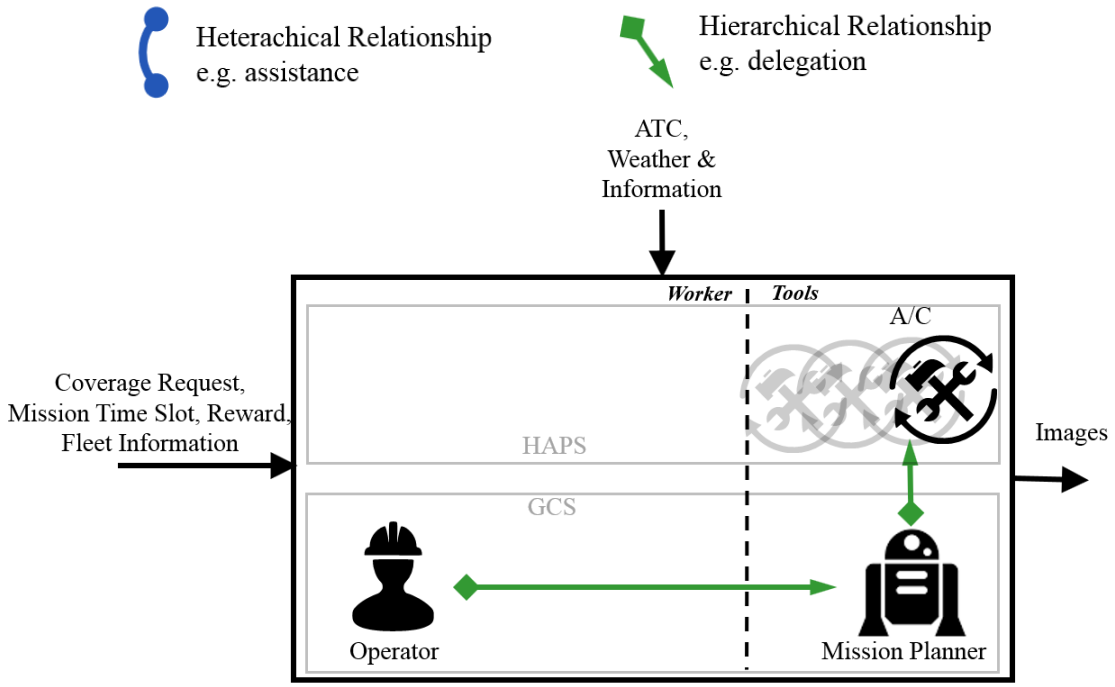
Figure 12. Work Processes (WProcs) of a HAPS MMS

The HAPS MMS consists of several work processes. The clients (`WProc:Client`) define the missions to be accomplished as well as the accompanying criteria (e.g. minimum coverage threshold, time windows, etc. that can be taken from Table 9) and reward. These will later be processed by a coordinator (`WProc: HAPS COORD`), e.g. the HAPS business analytical team, so that the payload, availability of fleets, etc. can be determined. The requirements are then properly translated into conditions that can be understood by the HAPS operating team (`WProc: HAPS SYS`). During planning and real-time operation, the HAPS team needs to be assisted by weather maps processed by `WProc: Weather` that summarize the current and forecasted weather situation in the operating airspace. As indicated in Table 9, the objective is to monitor ground activities and will be rewarded if the image coverage of the ground fulfills the minimum requested coverage. Therefore, part of the `WPOuts` of `WProc: HAPS Sys` must be images acquired by the mission payload, i.e. the on-board electro-optical camera. Other `WPOuts` also include flight information and mission plan, and status of the HAPS that must be communicated with the Air Traffic Control (ATC) (`WProc: ATC`).

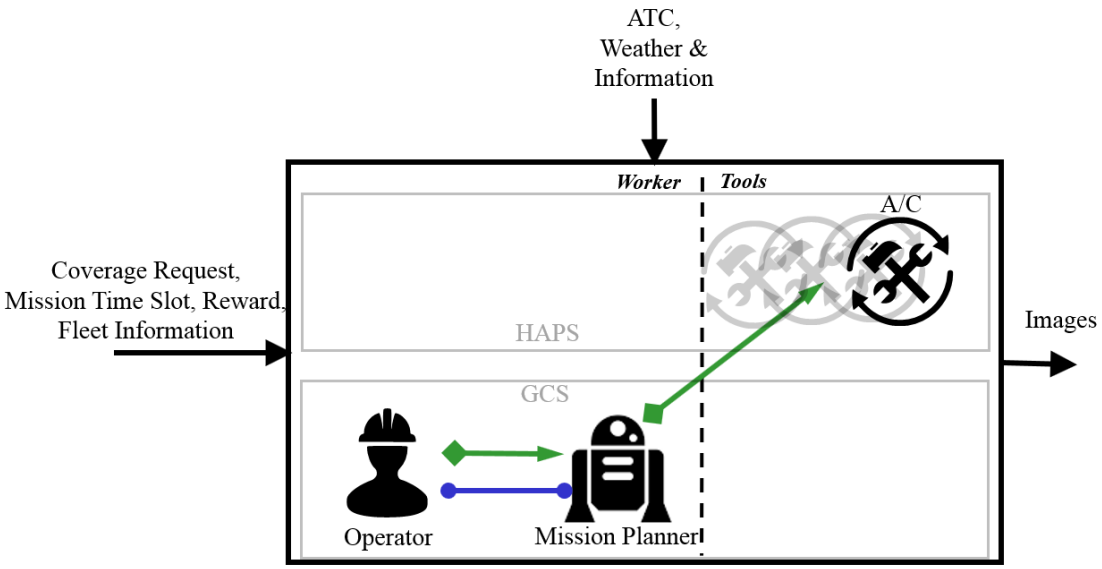
As indicated in [Schulte et al., 2016] in the guidelines for the initial design of a work system, after identifying the `WProcs`, each `WProc` can be opened up and subsequently, the worker and tools can be further specified, as well as the relations between workers, tools or between worker(s) and tool(s). Worker describes the role of a system component that can interpret the `WObjs` and translate them into tasks to be executed by the tools.

Subsequently, the relationships between worker-worker, worker-tool must be specified: hierarchical or heterarchical. A hierarchical relationship is directed from the delegate to the subordinate, and the heterarchical relationship inclines to a team work nature.

Figure 13a illustrates the `WSys` of the `WProc: HAPS SYS` in the HAPS MMS being developed in this work. Worker in the `WSys: HAPS SYS` is the Operator(s) stationed in the GCS, while the Mission Planner in the GCS as well as the onboard Aircraft/Control (A/C) that includes the flight control system, the reactive guidance, the onboard weather and mission sensors, and the communication modules. The gray duplicates of the A/C imply that multiple aircraft can exist in the system. The mission planner in the `WSys: HAPS SYS` refers indeed to the pre-execution mission planner, which is the core of this work. Since it is not the focus of this work to study the necessary number of operator(s) needed, the singular form "operator" is used to indicate the HAPS human operating team. Taken into account by the Mission Planner are weather information, HAPS dynamics constraints, HAPS payload and energy management, as well as mission-related constraints to ensure feasibility and to minimize replanning occurrence frequency. The main function of the Mission Planner is to determine feasible flight plans prior to execution. A secondary function of the Mission Planner is to compute a reactive avoidance strategy to be applied by the onboard reactive guidance during flight in order to reactively dodge critical situations that are not considered in the pre-execution planning and adhere to the reference plan as soon as possible. This class of reactive avoidance is considered a "plan repair", since no replanning is involved. However, it is not always possible to further pursue the reference plan. In such cases, the reactive guidance module identifies a safe zone and guides the HAPS to it, while the Mission Planner replan, i.e. compute a new plan. Although the Mission Planner advises the Operator on mission-related actions and decisions, the Operator exerts a supervisory function on the Mission Planner, by having the last word in any decision made.



a. `WSys` of `WProc: HAPS SYS`



b. An alternative of the `WSys` of `WProc: HAPS SYS`

Figure 13. `WSys` **of** `WProc: HAPS SYS`

A second, more advance, alternative shown in Figure 13b is also possible for the `WSys`. In this `WSys`, parallel to the supervisory role of the Operator on the Mission Planner, both also collaborate, allowing hence closer interaction, negotiation and interference at different levels. This implicates the integration of mixed-initiative planning functions into the Mission Planner, which is out of the scope of this work, but can be studied for future works.

The above study deals with the roles of various actors in a system or a system of systems. According to Felder and Collopy in [Miller, 2018], the process at the heart of every system exists as a function of time, which is also the case for HAPS, given its operation in a time-varying environment. `WProc: HAPS SYS` can be further analysed with a functional block diagram and timelines.

2.2.3 Functional Block Diagram of the MMS

After analysing the various stakeholders in the system, a functional block diagram of the MMS consisting of three major components is laid out in Figure 14 to indicate explicitly the in- and output of each module [Müller et al., 2018]. Several modules are particularly relevant to the core of this work, and therefore, are summarized below.

1. The **mission planner** in the GCS plans for long-term missions prior to execution (“pre-execution”) by taking into account the airspace structure to recognize its operation areas as well as no-go areas. Flight dynamics, and mission requirements are considered too in the computation of a mission plan. A plan-repair strategy to reactively avoid unforeseen danger is also determined by the mission planner. This module is the core of this work.
2. The **flight control** system is integrated onboard, given its time-critical function to guide the vehicle to follow a given plan and in return, provide position and attitude information of the HAPS to other modules.

3. The **on-board weather sensor(s)** detect clouds in the vicinity of the platform during the flight. If the clouds pose danger to the HAPS, the reactive guidance module will apply the plan-repair strategy or abort the current reference plan.
4. The **reactive guidance** unit monitors and reactively steer the vehicle to safety in urgency, either by reactively avoiding the danger as according to the plan-repair strategy, or by steering the HAPS to a calm area farther away so that the HAPS can wait for a new plan (i.e. "replanning").

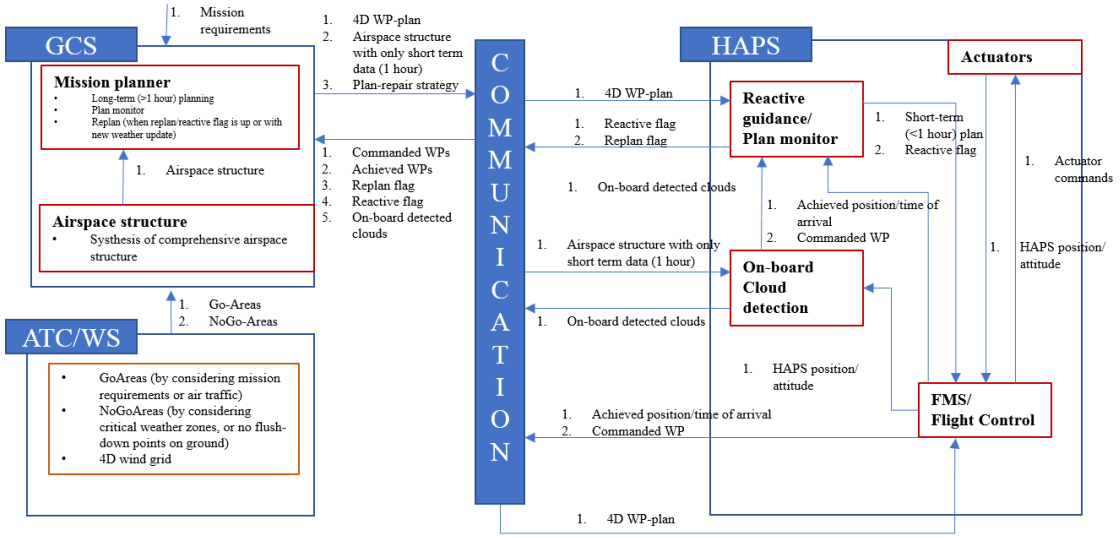


Figure 14. HAPS Mission Management System (MMS)

2.2.4 Temporal Analysis of Work Process WProc: HAPS SYS

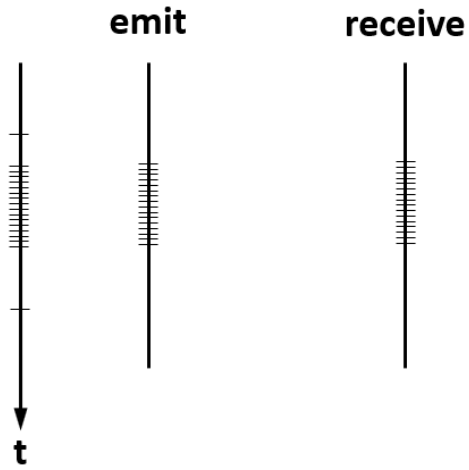
Conventional system modelling method uses an event-based methodology to describe the behavior of the system, e.g. the Unified Modeling Language (UML) sequence diagram. An event-based approach can describe the sequence of state changes very well, but not the duration of an imposed or automated action. In other words, the modelling language is incomplete: the temporal effect is missing. In automation, the consideration of temporal effects, or rather effects that are a time-dependent function is essential [Fox and Long, 2003; 2006]. Effects of a system can be caused by an event from the environment, or from an innate automated process that evolves with time.

A temporal sequence diagram is used to describe the possible happenings in `WProc: HAPS SYS`. Inspired from the UML sequence diagram [Rumbaugh et al., 1999], the temporal sequence diagram used here is also a two-dimensional diagram, with a vertical line dedicated to each actor involved and the discrete communication entities between actors are represented by the horizontal lines. It is worth noting that the temporal element in the UML sequence diagram is not clear. Signals are assumed to be discrete and instantaneous; it is not clear how a durative continuous or periodic signal can be represented on the diagram. We therefore add a few more objects to the diagram which are deemed essential to describe a temporal system.

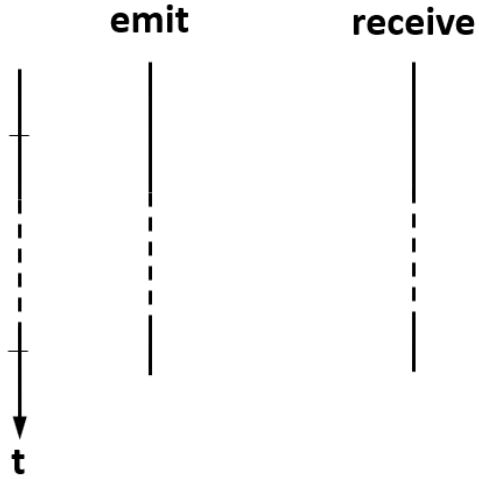
2.2.4.1 Extended Notations of the Temporal Sequence Diagram

Similar to the UML sequence diagram, the messages exchanged among different actors are referred to as "signal" in the temporal sequence diagram. A quantified timeline (see

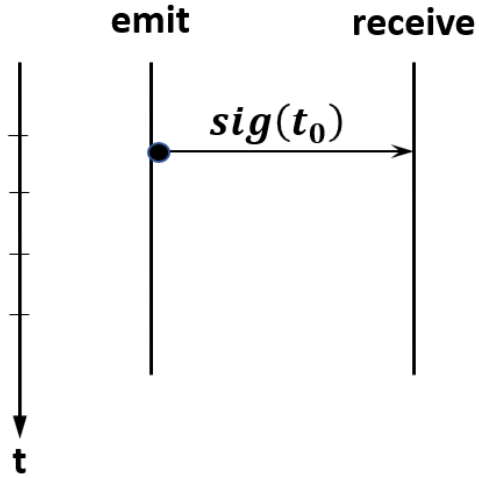
Figure 15) is included on the left of the sequence diagram to indicate the time instant an event happens or the start or end of a durative happening. As shown in Figure 15a and Figure 15b, the timeline can be stretched or compressed accordingly. And quite intuitively, the former is to accommodate more information, while the latter is to omit time windows that are lack of happenings.



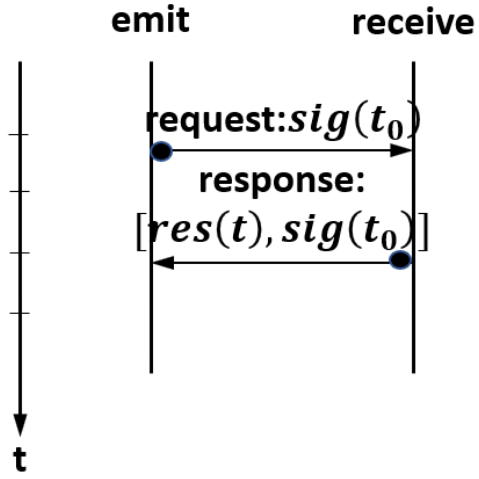
a. Stretched timeline



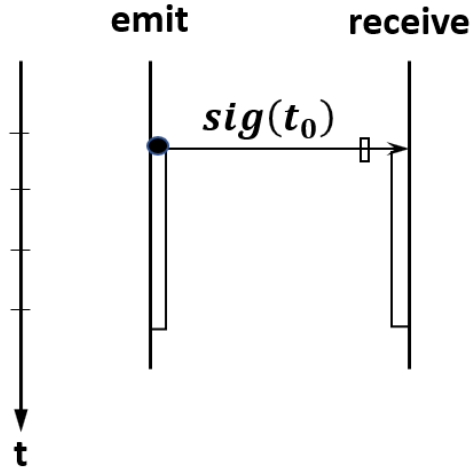
b. Compressed timeline



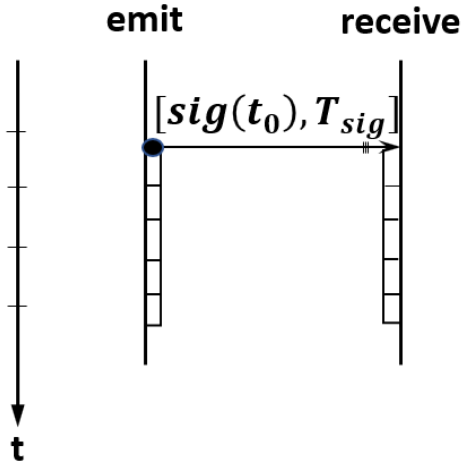
c. Instantaneous asynchronous signal



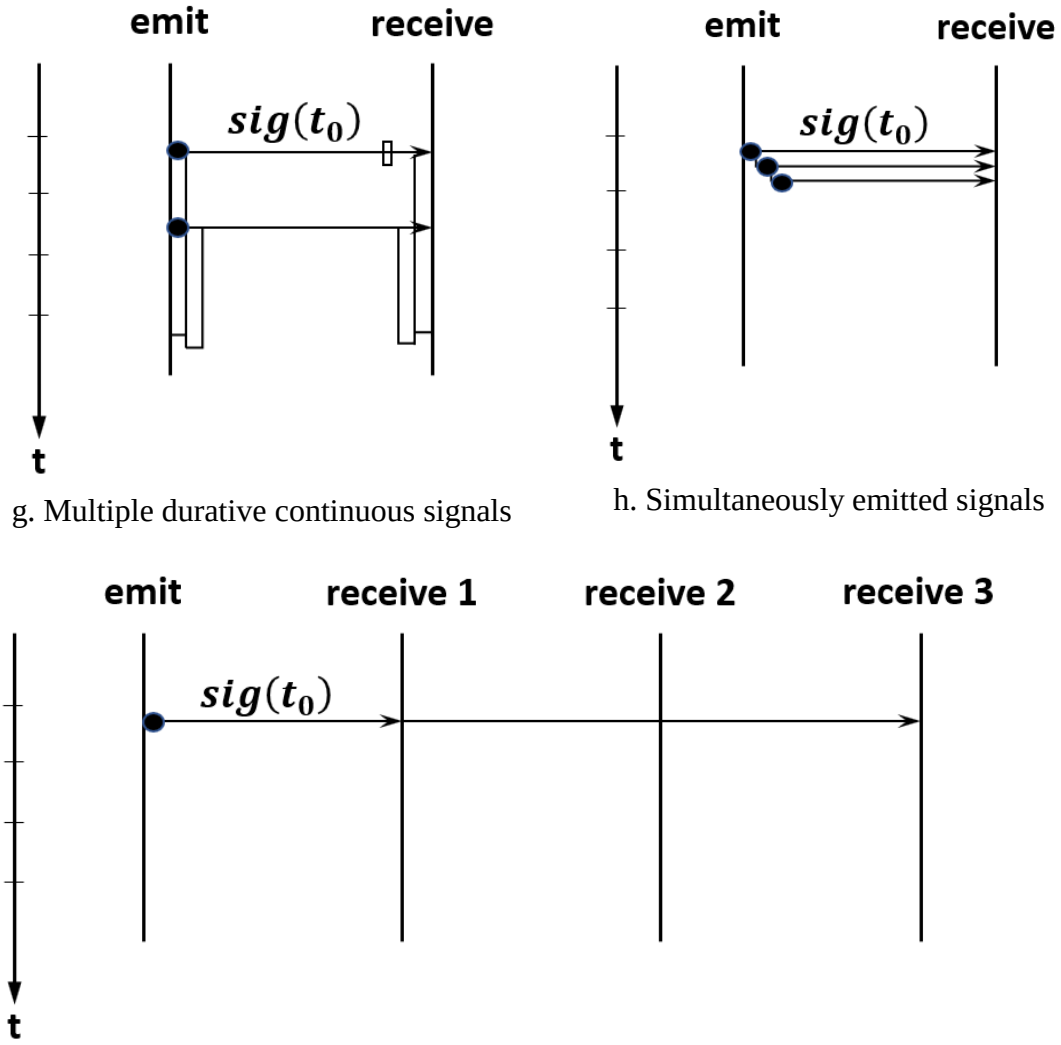
d. Instantaneous synchronous signal



e. Durative continuous signal



f. Durative periodic signal



g. Multiple durative continuous signals

h. Simultaneously emitted signals

i. Signals received only by certain actors

Figure 15. Temporal elements in the sequence diagram

In the simplest case, a signal is sent instantaneously at a time instant $sig(t_0)$. To represent that, line representing the signal is aligned with the quantified timeline and at the root of the line, a black dot represents the source of the signal. Like in the UML sequence diagram described in [Rumbaugh et al., 1999], some signals are only one-way, and are referred to as asynchronous signals, while others are synchronous, requiring a response from the receiving actor. A synchronous signal is represented with closed arrow heads, with the label initiated either by "request", or "response", that must follow sometime in the future, as shown in Figure 15d.

However, in a real system, not all signals are instantaneous. Some are durative, meaning the emission of the signal lasts for a certain duration, with the content of the signal being usually time-dependent, or not. Figure 15e shows the representation of a durative continuous signal. The bar attached to the timeline of the source indicates how long the signal emission lasts. A similar bar is also attached to the receiving actor to show how long the signal will be received by this actor. Additionally, a small rectangle is labeled before the receiving arrow as a hint for a "duratively" received signal. The advantage of having separate representation of the emission duration and the receiving duration is to be able to represent cases in which the receiving durations may differ from

actor to actor. Some durative signals are not emitted continuously but periodically. Lines are added to the bar attached to the timeline to indicate the interval between emissions. Before the receiving arrow, instead of a rectangle, two short parallel lines are drawn to indicate the durative reception of a periodic signal. In many modern digitalized systems, signals are discrete and even a continuous signal emission is repeated at a certain frequency. If a durative discrete signal is discrete periodic or continuous depends on the interval between two consecutive emissions and the time of response of the system. If the interval between emissions is negligible compared to the time of response of the system, the durative discrete signal is considered continuous.

Figure 15g shows the graphical representation of multiple durative signals emitted by the same source but at different time instants. The black dots at the root of sources indicate the sources of the signals. Similarly, if multiple signals are emitted at the same time instant by the same source, instead of drawing the roots of the signals right below the first signal represented, the roots are slightly shifted to the front as shown in Figure 15b and is linked by a vertical line to the first signal. Such representation may take up too much space on the diagram; therefore, if necessary, the timeline can be stretched. Of course, if all signals have the same properties, i.e. same duration, same receiving actors etc., a single line may represent all signals, provided all signals are labeled on the line.

When multiple actors are involved, and some receive a signal while others do not, the signal can cross the vertical lines of the non-receiving actors, but no arrow will be drawn, as illustrated in Figure 15i, in which $sig(t_0)$ is only received by "receive 1" and "receive 3". The arrow head at the edge of the horizontal line clearly indicates "reception".

2.2.4.2 Temporal Sequence Diagram for WProc: HAPS SYS

Using the standard defined by UML sequence diagram, together with the notations defined in the Section 2.2.4.1, the `WProc: HAPS SYS` can be represented as in Figure 16 and Figure 17. All actors external to `WProc: HAPS SYS` are grouped as "External". "External (GCS)" indicates external actors on the ground like weather station but leaving out the operator for clarity in the description of the human-machine interactions; "External (onboard)" refers to the onboard weather detection modules for example. $sig(t)$ is the signal containing the state parameters of the HAPS, i.e. position in WGS84, attitude, airspeed etc. The state parameters are communicated to all actors represented in the figures. The knowledge of the state of the HAPS is important for decision making, planning as well as for mission execution.

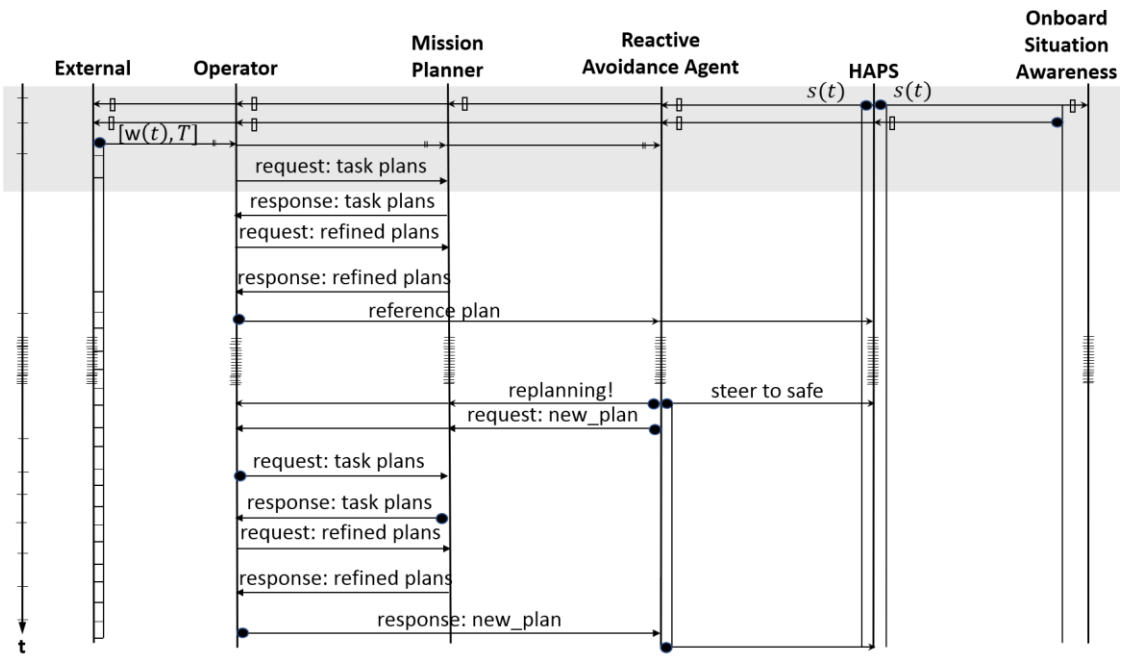


Figure 16. HAPS MMS temporal sequence diagram - part 1

The HAPS system receives periodically from the external entity, which in practice is a weather operator sitting in the GCS, the weather map $w(t)$. The weather map is updated at a fixed interval, e.g. hourly (as is usually the case with COSMO-EU [Baldauf et al., 2011], GFS etc.). The weather map shows the isolated critical weather zones, marked as "NoGo-Areas", which are derived from processed weather information [Köhler et al., 2017a]. Note that the weather map for HAPS must be specially customized, i.e. lower upper limits on the wind speed, thunderstorm likelihood, Reynolds number for turbulence detection etc., since HAPS is typically sensitive to weather conditions [Müller et al., 2018]. Wide area weather information cannot be the sole source of information for the operation of HAPS.

The functionalities of HAPS with its typical daily activities are first discussed, starting from nighttime (marked by the gray shadow). At night, the HAPS loiter within a waiting area to save energy, since the platform is completely solar-powered. It is worth noting that this could be different in the future with more advance battery technology. However, to the best of our knowledge, HAPS is rather inactive after sunset, as seen in Figure 6. Slightly before dawn, the latest weather information is communicated and the operator requests plan suggestions from the mission planner (`request: plans`), which is given a specific planning time within which, it must be able to compute relatively fast at least a plan for the day for HAPS. Depending on the time allocated for offline planning, the longer it is, the more likely the mission planner can improve the plans found, which are to be suggested to the operator. For our study, we set this duration for pre-execution mission planning time to 15 minutes.

Task plans (sequences consisting of time-stamped higher-level tasks) will be displayed and "explained" to the operator (`response: task plans`). These task plans are ranked with very roughly computed fitness and reward. The evaluation of plans will be explained later in Section 4. The operator can decide to go with the ranking or alter it. The task plans will be "refined" by the mission planner in the order they are ranked, i.e. more details will be considered by the mission planner while determining the travel time and reward of the

HAPS executing the ordered tasks. The “refined” plans are returned to the operator again and he selects subsequently the “best” plan and communicate to the onboard reactive avoidance agent and HAPS Flight Control System (FCS). This plan will further be referred to as the “reference plan”.

In the event of unforeseen dangerous circumstances, which can happen sporadically, the onboard reactive avoidance agent realizes that by following the reference plan, the HAPS may be steered into dangerous zones, or that the electric motors of HAPS could be burdened too much, due to its difficulty to cope with the reference plan. The reactive avoidance agent will send a warning to the mission planner as well as to the operator that a replanning is necessary. Meanwhile, if proceeding with the reference plan is no longer an option, the reactive avoidance agent will start to steer the HAPS to a safe zone.

After evaluating the situation, the reactive avoidance agent will inform the GCS from which zone or start point the next long-term plan should be determined by including that piece of information in the `request: plans`. As with every long-term mission planning process, it is triggered by a request for plan suggestions sent from the operator to the mission planner. Subsequently, from the plan suggestions, the operator selects the best plan, aided by the decision-making assistant. The new reference plan will be communicated to the reactive avoidance agent. However, the agent will continue steering the HAPS to the new start point that was communicated to the GCS and that was also considered in the determination of the new reference plan. When the HAPS has reached the new start point, the reactive avoidance agent forwards the new reference plan to the HAPS FCS to be executed.

In other cases where pursuing the reference plan is still an option because the danger that is not considered in the reference plan will only take place in far future, the HAPS will not interfere to steer the HAPS out of the reference plan, but rather just request a new plan by including the waypoint beyond which the reference plan is no longer safe to execute (see Figure 17). The new plan selected will then be merged with the original reference plan and communicated to the HAPS.

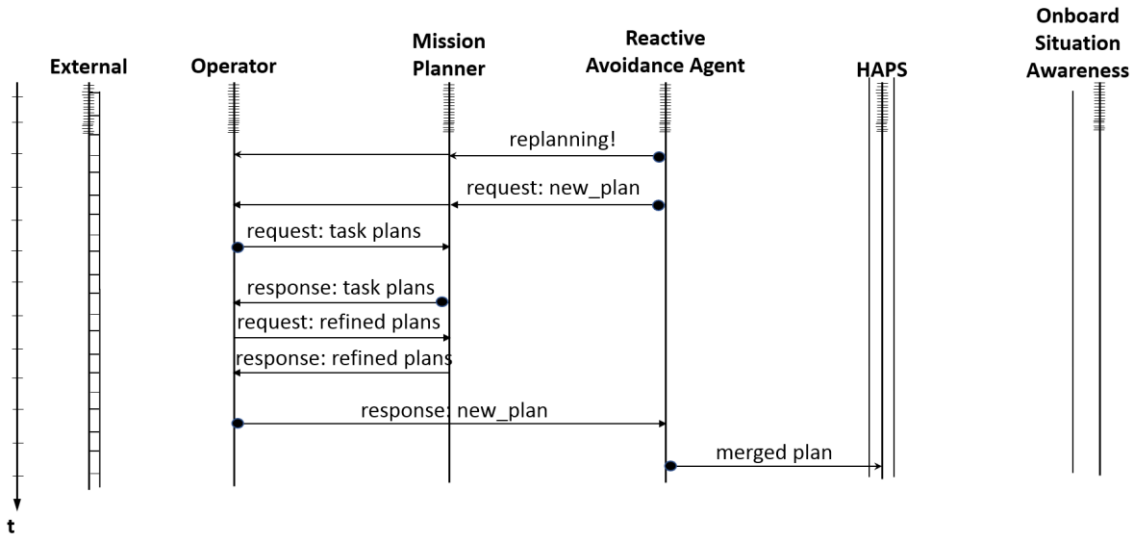


Figure 17. HAPS MMS temporal sequence diagram - part 2

In the events described above, a replanning is required, i.e. the reference plan will be aborted and replaced by a new plan. This however could be inefficient and unnecessary, especially when the unforeseen danger can be easily avoided, for example when only one or two newly detected NoGo-areas lie on the reference plan trajectory. In this case, a plan repair is advisable. As depicted in Figure 18, few newly detected NoGo-areas are detected by the onboard weather sensors, instead of aborting the reference plan, the reactive avoidance agent guides the HAPS to avoid the hazardous areas, while steering the HAPS to follow the reference plan trajectory whenever possible using a strategy pre-determined by the mission planner. The reference plan will be checked by the mission planner, since the ETA of the waypoints are different after the reactive avoidance is activated. If the remaining plan is executable, the ETA of the plan will be updated, as shown in Figure 18. Otherwise, a re-planning will be triggered.

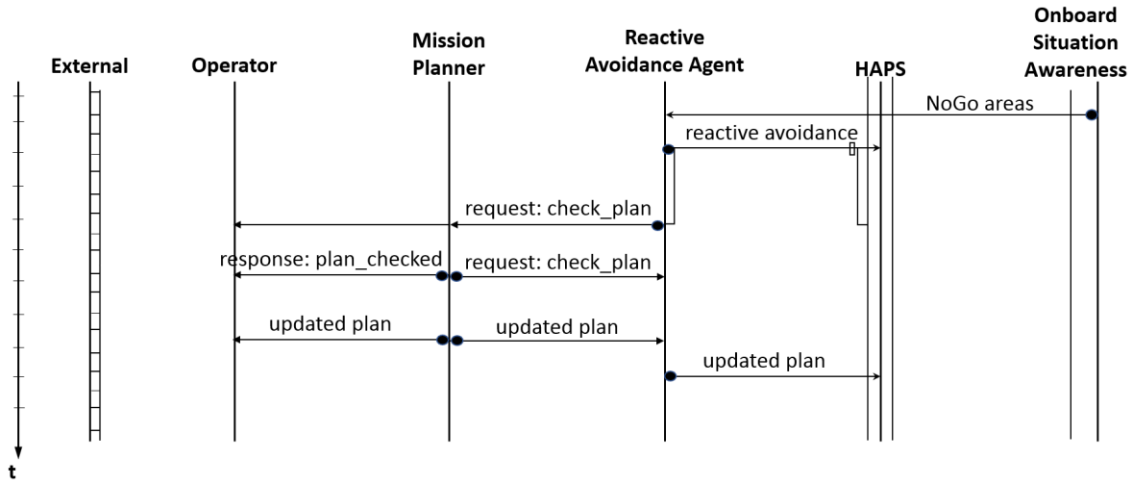


Figure 18. HAPS MMS temporal sequence diagram - part 3

2.3 Formal Mission Planning Problem Statement

The general HAPS mission planning problem $\mathcal{P}_{\mathrm{HAPS}}$ tackled in this work can be formally represented by the tuple of set- and non-set-elements:

$$< \mathrm{H}, \mathrm{X}, \mathrm{T}, X_0, T_0, \mathrm{A}, C_A, Z_1, \ldots, Z_I, C_M, \mathrm{r}, H_\pi >, \text{ where}$$

- H is the set of HAPS enumerated with integers.
- X is the continuous workspace, i.e. p^h position of HAPS $h \in H$, q^h attitude.
- X_0 is the set of initial states x_0^h for all $|H|$ HAPS.
- T is the continuous time domain $[t_{\min}; t_{\max}]$, with t_0^h being the initial time of the plan horizon for HAPS h.
- T_0 is the set of initial times t_0^h for all $|H|$ HAPS.
- C_X is the set of constraints on X within T, e.g. the time-stamped polyhedron delimiting the NoGo-Areas, the operation area.
- A is the set of primitive actions, i.e. actions that can be executed directly by the FCS.
- C_A is the set of preconditions of action $a \in A$.
- Z_i is the set of measurements of the physical environment from source i, either a third-party or an onboard sensor.

- C_M is the set of mission-related requirements (see Table 9), expressed by means of elements of X, T, and A.
- $r: A \times C_M \rightarrow \mathbb{R}$ is the set of reward mapping function if the mission requirements $c_M \in C_M$ are fulfilled.
- H_π is the set of remaining plans to be executed at $t_{\min}$ for all HAPS in H.

2.3.1 Solution

The solution to $\mathcal{P}_{\text{HAPS}}$ is the set of $|H|$ plans, where $|H|$ is the cardinality of the set of HAPS H. A plan π^h for HAPS h is a sequence of time-stamped actions $\left(< a_0^h, t_0^h >, < a_1^h, t_1^h >, \dots, < a_{n^h}^h, t_{n^h}^h >\right)$, while a partial plan $\pi^h(i,j) = \left(< a_i^h, t_i^h >, \dots, < a_j^h, t_j^h >\right)$ is a subsequence of π^h from time instant t_i to t_j and if the end time instant is omitted, i.e. $\pi^h(i)$, it simply represents the partial plan from time instant t_i, i.e. $\left(< a_i^h, t_i^h >, \dots, < a_{n^h}^h, t_{n^h}^h >\right)$.

2.3.2 Abstraction of the Planning Problem

Note that the formal definition of the planning problem can be inconvenient, especially when it comes to expressing MC and MR defined in Table 8 and Table 9. Furthermore, it is less intuitive for a human operator to recognize a position represented by a real number vector than a location like "MA1" or "LOI1" of "MA3". It is hence practical to represent the $\mathcal{P}_{\text{HAPS}}$ and solve it at a higher-level abstraction. $\widetilde{\mathcal{P}}_{\text{HAPS}}$ denotes the HAPS planning problem at a high-level abstraction, the set-elements of which are subsets of the set-elements of $\mathcal{P}_{\text{HAPS}}$. Moreover, each element of the set-elements and each non-set-element of $\widetilde{\mathcal{P}}_{\text{HAPS}}$ can be expressed (although the representation is more laborious) using the non-set-elements of $\mathcal{P}_{\text{HAPS}}$ or using the elements of the set-elements of $\mathcal{P}_{\text{HAPS}}$. For example, for a $\widetilde{\mathcal{P}}_{\text{HAPS}}$ abstracted at the MA-level, the state of HAPS is simply represented by MA*, i.e. MA/WA is the smallest physical unit.

The abstraction level for reasoning will be given in each of the following sections.

2.4 Related Works

Solving a mission planning problem involves multiple aspects, ranging from semantics representation, task planning and scheduling, motion planning, constraints handling, etc. Many works focus on a selection of aspects and neglect others, making the proposed solutions only partially applicable. In this work, it is strived to provide an overall solution framework to the realistic planning problem described in Section 2.1. Before detailing the underlying solution in the next sections, some prominent related works (but non-exhaustive) are summarized and analysed to provide an overview of the available state-of-the-art methods.

There is hardly a universal theorem for planning, although many try to formalize the methods and organize the various approaches with different branches, much like the branches in physics (mechanics, electromagnetism, optics, etc.). The most commonly used classification of planning methods is "classical planning" versus "motion planning". The former focuses on intuitive planning in a blocks-world that reasons at higher abstraction level, like "move Box-A to Warehouse-C", while the latter focuses on planning a path or motion numerically in a grid-world or in some more realistic application, also in a three-dimensional physical world. Such classification is of course inadequate. More complete classification of planning methods applied in Artificial

Intelligence (AI) can be found in prominent textbooks [Russell and Norvig, 2003] and [Ghallab et al., 2004]. However, as one notices immediately, both textbooks have a different approach of viewing "planning", although the topics covered are quite similar. The classification used in this section for planning approaches is based on experiences gathered from solving the HAPS mission planning problem and could deviate from the textbook or other classifications. It provides another crystallized perception and serves mainly to comprehend the choices made to select and adapt the planning approaches for HAPS.

Figure 19 draws an overview of the different paradigms of planning/scheduling approaches, as well as the coupled optimization methods. The former is merely a concept consisting of the organization of the problem knowledge, the representation of solution and the unit of the core elements (i.e. decision variables/state parameters) taken for reasoning, while the latter is a method to search for the optimal solution in the conceived planning world and works more as a back-end engine. Theoretically, the optimization methods can be substituted by one or another, provided the modelling of the planning world is clearly separated from the search engine. This ideology has led to the so-called "domain-independent planners", the particularities of which will be discussed more in-depth in Chapter 3.

Planning can be grouped into two main categories, namely model-based or data-based [Blanning, 1981]. Model-based planning approaches contain a structured model of the executive system, and based on the model, a plan is computed for or to aid decision making. A model-based planning approach often also uses either an appropriate front-end modelling language in which the planning problem is represented to be understood by the back-end decision engine, or an API in which the problem variables can be instantiated. A data-based planning approach has recently gained ground thanks to the emerging affordable fast processors and data storage. Data-based planning is used for model-free planning (for example using reinforcement learning in [Sutton and Barto, 2017], deep extensions in [Mnih et al., 2013], etc.), or when the model is incomplete, for example, planning with incomplete knowledge of a highly non-linear transition function and highly non-linear reward function can be solved using Monte Carlo search and deep learning from Tensorflow in [Keller and Helmert, 2013] and [Wu et al., 2017] respectively. The focus of this work is on model-based planning; thus, data-based planning will not be further elaborated in the following.

2.4.1 Model-Based Planning Methods

Some planning/scheduling paradigms as well as the search/optimization for model-based planning are presented and classified to show the strengths of the different methods, with no pretense that the classification is complete.

In the following, brief descriptions on the optimization and search methods shown in Figure 19 are first provided. It is worth noting that the classification of many modern search/optimization algorithms is vague, as a standalone classical algorithm hardly suffices, and therefore a combination of methods becomes inevitable.

Subsequently, the world of planning/scheduling paradigms will be presented to illustrate the numerous different ways of modeling a planning problem in order to gain some insights on why a paradigm is more suitable than the others. Note that planning and scheduling are also hardly distinguishable nowadays. Planning was conventionally seen a computation of "how to achieve a goal or a set of goals", while scheduling was

considered a resource allocation over time to check if all constraints are met [Ghallab et al., 2004]. With more advanced methods, often resource allocation and planning can be done within the same framework/tool; thus, drawing a line between both becomes almost meaningless, e.g. timeline-based planners also have scheduling capability [Fratini and Cesta, 2012]. The following subsections, as well as the rest of the work, consider that scheduling is also part of planning.

Lastly, some planning tools/frameworks are briefly described, with an emphasis on the search method, the planning paradigm, as well as the typical application(s).

2.4.1.1 Optimization Methods

Almost always, the back-end decision engine contains an optimization method that searches for the optimal solution, or at least attempts to approach optimality. The most commonly used search methods are based on a search in a graph made up of nodes representing the discrete states. The most straightforward is a forward search, in which the search begins at the initial node and starts to expand its explored set of nodes progressively toward the goal[13], while optimizing the objective of the problem, e.g. shortest distance, shortest time, maximum reward, minimum cost, etc. Listed in Figure 19 are several prominent search methods that are based either on heuristics to guide the exploration of nodes toward the goal node, or a random search principle to explore the graph as much as possible. The following describes the methods very briefly.

1. Dijkstra's algorithm: This is an uninformed search algorithm classically used to find the shortest path from the start to the goal node, by exploring the neighboring unvisited states progressively.
2. A*: This is the most basic heuristic-guided search extended from Dijkstra's algorithm with the introduction of a heuristic, which is an approximated estimation of the cost/reward of the remaining path to goal. The value function, i.e. the sum of the cost of the explored path and the heuristic of the remaining [14]path, guides the search faster toward the goal [Hart et al., 1968]. It guarantees reproducibility (since it is not random) and also optimality, if an admissible heuristic is in used.
3. WA* (Weighted A*): The WA* extends A* by introducing a weight $\epsilon > 1$ on the heuristic in the value function [Rüdiger and Drechsler, 2009; Pohl, 1970]. The weight induces more greediness via forcing a bias toward the goal in the search, accelerating hence the search at the expenses of optimality. WA* is said to be ϵ-suboptimal.
4. Dynamic A*: Another variant of A* for quick local re-planning to react to changes in the environment [Likhachev et al., 2005; Stentz, 1995; Ferguson and Stentz, 2005]. The algorithm is suboptimal and has so far been tested only for grid world path planning.

[13] A "goal" can be understood as a goal position in motion planning (as employed in Chapter 3 or more generally, a state that fulfills all the conditions defining an end state (also known as "goal conditions").

[14] A "path" does not always indicate a physical path from a point to another in the three-dimensional world. In AI planning, a path can also refer to a plan trace, which is a state outcome from start to goal after the execution of the plan.

5. hill climbing: Hill climbing and its variants for example Enforced Hill Climbing (EHC) are developed for greedy search guided by a heuristic to the local optima [Russell and Norvig, 2003].
6. RRT (Rapidly exploring Random Tree): RRT [LaValle, 1998] and its variants that are based on the principle of random expansion of the tree from the start node. While the reproducibility of the search is questionable, due to the randomness in the search space exploration, the expansion of the search is fast. Moreover, with the minimalist algorithm, RRT can be tailored very easily for many motion planning problems. More of RRT will be described in Section 3.
7. planning graph: Search methods 2-5 are heuristic-guided. Domain specific heuristics can sometimes be derived easily; for example, a metric distance to goal can be assumed as the heuristic for the planning of a shortest-path problem in a grid world without the presence of a vector field. However, in many more complex planning problems, heterogenous state parameters are present, i.e. state parameters that have different physical units, different abstraction levels (meta-state versus position parameter), or different importance (thereby one state is preferred to the other in the cost function). Planning graph is a neoclassical planning method (as claimed by [Ghallab et al., 2004]) debuted by GraphPlan in [Blum and Furst, 1997], and has ever since been the standard search technique used in domain-independent AI planners to determine the relaxed reachability heuristic of a state to the goal state [Bryce and Kambhampati, 2007]. Unlike a usual progression search graph, a planning graph structures prepositions (states) and actions into alternating layers within polynomial time. In many use cases, the heuristic is approximated by counting the number of steps, or rather the number of necessary layers to reach an approximated state from the initial state. The approximated state is in fact a union of sets of propositions (states) that includes the exact state desired to be reached.

Some search methods can be performed in a reverse mode, by searching using the same approach to explore nodes from the goal to the initial state. RRT and the planning graph are sometimes being implemented this way, if the graph is symmetric. This is being exploited to reduce the search complexity by introducing a forward-backward search. The nodes expanded from the initial state and from the goal state respectively will meet, and the partial plans will be juxtaposed to form a plan to the planning problem in the case of RRT, or to determine the heuristics in a planning graph.

Another interesting approach is by using an incremental search, which consists of first finding a plan or plans that will be improved incrementally. “Incremental” was used to describe the techniques adopted by LPG [Gerevini et al., 2003]: an initial (partial) plan is first found, and transformed into a valid plan or optimized to obtain a better plan by improving the found plan locally each time. The “incremental” technique has also been employed in other planners.

Constraint Satisfaction Problems (CSP) techniques can be very powerful in handling constraints of a planning problem. The planning problem can be compiled into a CSP (with a graph made up of nodes of constraints, instead of states), and by applying CSP tools like forward-checking, backtracking etc., the domain can be reduced, and redundant constraints can be excluded. Very often, mixed-integer programming and planning graphs use CSP techniques to handle constraints [Ghallab et al., 2004].

Optimization can also be performed globally and iteratively without forming a graph. The solution to a Markov Decision Problem (MDP) or its variants for instance, is often the optimal solution of the Bellman equation and can be found using dynamic programming such as value iteration, policy iteration, etc. [Bertsekas and Tsitsiklis, 1996]. Evolutionary Algorithms (EA) and the variants of EA can be perceived as an iterative method to solve a planning problem too [Jong et al., 2017]. The EA search randomly in the state space and at each iteration, the neighbors of the fittest solutions will be searched for better solutions. Such iterative methods are generally used when the number of decision variables are known (but modern works could differ), or rather the upper limit of the number of combinations to the combinatorial problem is known, whereas in the case of a graph-based planning, the knowledge on the size of the combinatorial problem is unknown, e.g. the number of actions needed to travel from warehouse A to warehouse C while having transported box K to warehouse B is unknown. If the upper limit of the plan duration (or rather plan horizon) is known in advance, the EA can also be adopted, like the Genetic Algorithm (GA) adopted for the high-abstraction task planning for multiple HAPS described in Chapter 5. The advantage of the EA is multifold:

1. Its randomness in the search space exploration can help to accelerate the optimization, especially in a planning problem where it is difficult to draw a heuristic to guide the exploration of the most promising nodes.
2. Its principle of generating new population using the fittest individuals of the previous generation can help to retain some promising traits of the previously found plans.
3. Constraint-handling techniques can be integrated into the algorithm to deal with more complex problems that require the optimization of objectives, while conforming with the constraints.
4. The implementation of the EA implies already an incremental search, i.e. plans are available at each iteration and are incrementally improved or repaired. This offers the possibility of using the EA in an anytime planner, a planner that can provide a plan at any time instant, with the quality of the plan being better if more planning time is allocated.

Another less common way but elegant is by solving analytically an optimization problem. Least-squares and level-set methods are commonly used. The former is easy but is limited, since linearity is required, or a linearization is necessary. The latter is commonly used for Autonomous Underwater Vehicles (AUV) [Lolla et al., 2012; Lolla et al., 2015], which consists of propagating from the start to the goal or vice versa. The level sets of constant arrival times derived from the Hamilton-Jacobi differential equation formulated to represent the optimization problem. The method is however challenging to be applied on a planning problem in which many heterogenous variables are involved.

2.4.1.2 Planning/Scheduling Paradigms

Model-based planning/scheduling revolves around a set of rules defined to govern the evolution and the reaction of the system. Using this set of rules, or rather "model", a plan can be drawn to bring the system from its initial state to its goal state or to guide the system to execute a task.

While the term "classical planning" is a classification of planning methods preconceived with respect to the time the methods were developed, the term is avoided here. The intent is to group the planning and scheduling methods according to their core

reasoning elements that define the main characteristics of the planning problem in question, which are:

1. sampling-based in the physical world,
2. task- or action-based, and
3. timeline-based.

Sampling-based planning of the physical world is the most intuitive class of planning, which consists of solving numerically a motion/path planning problem every mobile agent faces by discretizing the continuous space. This class of planning problem is also often referred to as "motion planning". The typical characteristic of it lies within the reasoning in the 3-dimensional physical world (but can sometimes be reduced to fewer dimensions), while meta-representation of the world is often ignored. Typically, the configuration space of the physical world can be modelled deterministically or probabilistically. For a deterministic modelling, either 1) the physical world is sampled via a numeric discretization into geometric cells or 2) into potential field that facilitates the use of partial differential equations to describe the motions, or 3) the control-space is sampled into a numerable set of control parameters so that the constraints imposed by the motion actuators can be taken into account too. The probabilistic modelling of the physical world is performed with Probabilistic RoadMaps (PRM), that consists of selecting collision-free nodes probabilistically to form a graph in the configuration space, of which the edges (i.e. the paths to connect nodes) are computed using local planners [Kavraki et al., 1996].

The sampling of the configuration space is however inadequate for many planning problem instances involving meta-representation, for instance the well-known Dock-Worker-Robot (DWR) logistic planning problem [Ghallab et al., 2004], in which many states bear a meta-representation, which can be formalized using prepositions. The planning revolves either around the tasks/actions, in order to determine what to do, or around the resources of the system, in order to determine who does what and when. An action-based planner reasons in the state space and determine the actions needed to accomplish a goal or many goals. Deterministically, the model can be formalized using Planning Domain Definition Language (PDDL) [McDermott, 2000], a planning problem modelling language that captures explicitly the causal effects of actions. Some other variants like Multi-agent PDDL (MPDDL) was developed to also support the formalization of an action-based planning problem involving multiple agents, while PPDDL [Younes and Littman, 2004] was developed to model a probabilistic planning problem. Likewise, Markov Decision Problem (MDP) is also for the modelling of a probabilistic action-based planning problem. More about MDP will be explained in Chapter 6.

Since some latest development in the modelling tools for action-based planning problems like PDDL+ [Fox and Long, 2006], temporal planning problems can also be modelled, i.e. the plan must decide what to do, but also when to carry out the actions. With the parameterization allowed since PDDL 2.1 [Fox and Long, 2003] and subsequently also in PDDL+, it is only possible to formalize multiple-agent temporal planning problems in PDDL+, making thus the line between planning and scheduling more ambiguous than ever. PDDL+ is used in Chapter 3 to plan for the feasible flight path for HAPS in a time-varying environment (i.e. time-dependent weather conditions and dynamic airspace allocation). The language is chosen since the semantics can fully represent the control-based motion planning problem of the HAPS; furthermore, with the

integrated heuristics in many PDDL+ planners, the planning efficiency is competitive with other alternative motion planners. The domain-independent nature of the PDDL+ planners is also promising for future easy adaptions of the flight path planning problem, i.e. no further alteration on the planner is required for each adaption of the problem model.

However, the modelling language of a scheduler, like the Action Notation Modeling Language (ANML), captures much better the roles of each actor in the system and is more natural in encoding the known duration of an action, making it still a preferred choice for many scheduling problems. Furthermore, timeline-based planning tools have quite matured in space applications [Fratini and Cesta, 2012]. It is worth nothing that ANML models a problem in what is referred to as the plan space, in which each node is a partial plan that encodes domain knowledge on action routine. This space representation, although rather implicit, captures first and foremost partial plans that can be determined using domain-specific knowledge, i.e. the actions/tasks to perform a meta-task, or to achieve a (sub)goal. A Hierarchical Task Network (HTN) in which a high-level task is decomposed into lower-level tasks and subsequently into primitive executable tasks, can in some cases also be nested in ANML, which is a paradigm used in the FAPE planning and acting framework [Dvorak et al., 2014], although HTN is usually employed for a task-based planning problem. A temporal HTN paradigm is used to plan for the high-level tasks for HAPS (see Chapter 4), in order to encode the routine of HAPS operation. However, since a generic framework or HTN planner does not exist to cope with the HAPS task planning problem, the modelling using ANML is first left out in this work, since the process of developing an ANML-compatible domain-independent planner for such a complex problem is extremely laborious. It is also worth noting that HTN paradigm is often used with domain-dependent planners, due to a lack of formal definition of the paradigm (unlike action-based paradigm like PDDL).

An elegant way of modelling the planning problem is by simply formulating the dynamics of the system with a set of equations (e.g. Hamilton-Jacobi equations, ordinary differential equations, etc.) with variables from the continuous real-number set. This is often used in optimal control problems. Solving this class of problems include first the study of the existence of an optimal solution given the conditions, followed by applying a solution method from dynamic programming [Gerdts, 2012].

A CSP is rarely used to model an entire planning problem for plan computation, but a planning problem can be compiled into a CSP, enabling hence the use the CSP-tools for constraint handling [Ghallab et al., 2004].

2.4.1.3 A Quick View of Planning Tools/Frameworks for Complex Real-World Systems

The above tools or methodologies stem from solid theoretical works and have prevailed in solving many planning problems, for example PDDL planners were used to plan for batch chemical plants [Della Penna et al., 2010] and urban traffic [Vallati et al., 2016], MDP-based variants for collision avoidance for unmanned vehicles [Temizer et al., 2010; Ragi and Chong, 2013] to name a few. These planning problems present however only a part of a larger-scope mission planning problem, of which

1. the goals are multiple and involve heterogeneous representations, e.g. numeric or meta-state;
2. the multiple objectives that reward/penalize actions of different abstraction levels, e.g. primitive actions or meta-actions;

3. the information on the environment is available at different precisions with respect to the spatial-temporal resolutions.

The following paragraphs recapitulate briefly a few frameworks that assemble multiple planning tools cited above in Section 2.4.1.1 and 2.4.1.2 to solve more thoroughly complex hybrid mission planning problems.

FAPE

The Flexible Action and Planning Environment (FAPE) is one of the most all-rounded generic planning framework for robotics [Ingrand and Ghallab, 2013], which incorporates multiple modules to plan and to act (also known generally as "Planning and Acting"). ANML is employed to formulate the temporal problem, while a plan-space based planning approach is adopted via the use of a HTN-based planner in order to enable plan repair during acting. While the planner instantiates the actions only partially, the acting module refines the actions into closed-loop functions by fully instantiating them, with more detailed time managements, local reaction actions, consideration of non-determinism, etc. In case of failure, the plan repair removes the failed action/task from the task network, as well as all effects of it; subsequently, another action/task will be planned to compensate for the deleted effects.

Interfacing of Task+Motion Planning

[Srivastava et al., 2014] identified the need of a combined task+motion planning even for very simple robotics planning problems. To this end, a generic interface was developed to aid communication between a symbolic task planner and a numeric motion planner. Task and motion planning are performed iteratively, i.e. if the motion of an instantiation from the task planning cannot be planned, due to an obstacle lying in the way for example, a "fail" feedback is communicated via the interface to the task planner; subsequently another task instantiation will be provided.

This combined task+motion planning approach is very similar to the planning framework developed here for HAPS: However, in order to avoid the iterations in case of failures in motion planning, an appropriate motion planner is carefully chosen and adapted to suit the flight path planning for HAPS in a time-varying environment in order to minimize the failure rate. Meanwhile, the task planner does not only provide one task plan as a planning instantiation for the motion planner, but multiple task plans, enabling hence

1. an immediate switch to another task plan in case of a failure in motion planning, without re-invoking the task planner
2. a simultaneous motion planning of many task plans with a multi-threading implementation.

ROSPlan

Following the same principle, ROSPlan was created as a framework to integrate generic task planners (and more specifically PDDL 2.1 planners) into the Robot Operating System (ROS) [Cashmore et al., 2015]. ROS is a widely used framework containing tools and modules for robotics, which include communication between sensors and actuators, motion planners, visualization and many sensor data processing libraries. ROSPlan takes

in a model of the executor, interprets the sensor data collected from the ROS modules, and generates problem instances in PDDL 2.1 that can be understood by compatible task planners.

While this work intends to pursue the development of a more thorough planning framework, the "generic" feature of the framework is however left for future works, since the effort is different from the planning thoroughly for a specific use case. Furthermore, the plan repair is only dealt with partially: the effort of the mission planner on the GCS (see Figure 14) is described, but the on-board plan repair functions remain out of the scope.

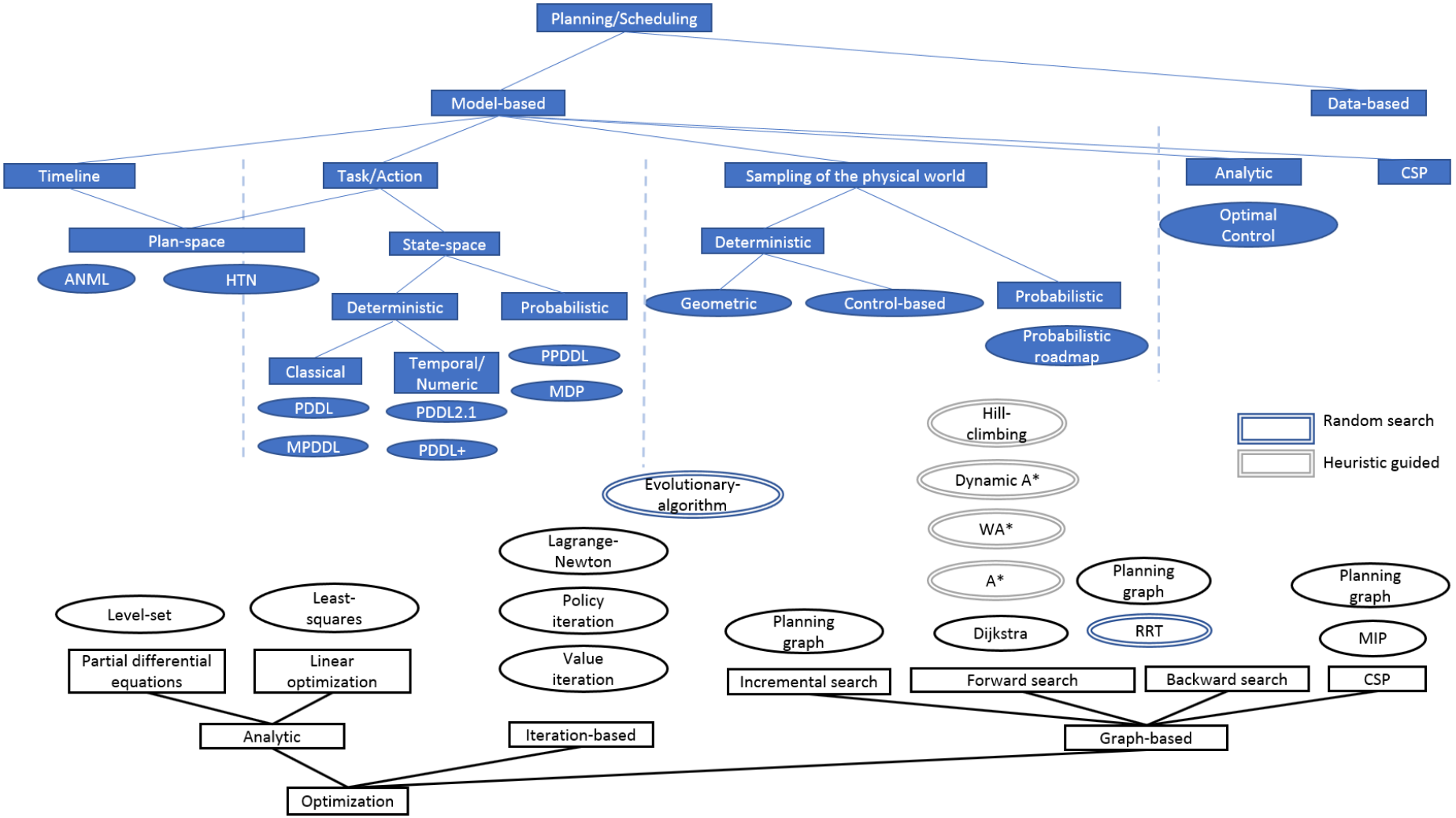


Figure 19. Overview of planning and scheduling methods

"Ce qui est et n'est pas par soi est par un autre."
- Claude Paulot in *Science et Création*

3 *Path Planning in Vector Field*

As described in Section 2.1, a HAPS is sensitive to its time-varying environment; additionally, the mission requirements and mission constraints listed in Table 8 and Table 9 are time-dependent. Hence, in order to know if the mission requirements can be fulfilled, or if the compliance with the mission constraints is achieved, the planner must be able to predict the time-dependent position $p(t)$ to a satisfactory level (subject to the knowledge of the time-varying environment at the time planning is performed), and take this prediction into account in computation of a plan, which can be performed by a path planner.

This chapter discusses the approach employed to plan feasible flight paths for fixed-wing aircrafts in a time-varying wind field using an action-based automated AI-planner. Also taken into account are the dynamic weather zones, as well as the airspace structure. The choice of exploiting an automated AI PDDL+ planner is explained in this section. The planner is model-based and domain-independent, i.e. the planning domain (system) and the problem instance (state parameters and goal conditions) are first formulated rigorously in a modelling language with clear axioms on the causal of effects, and subsequently being processed by the planner, also called a solver, that is capable of understanding the formulated problem and find a or multiple solution(s) with the search engine integrated within the solver. A domain-independent planner is not developed for a specific system or problem, but for any that can be formulated in the language it understands. Using this approach, the planning method does not need to be adapted accordingly even if the domain or the problem is altered. Nevertheless, it is worth noting that a domain-independent planner can only understand the modelling language designed for formulating problems of the same kind of planning world, while very few modelling languages can be compatible for different planning approaches depicted transversely in Figure 19.

In this section, the isolated path planning problem for HAPS is first discussed, followed by a brief overview of some commonly used path planning techniques, and a description of the Problem Domain Definition Language (PDDL) and the compatible planner solvers. The latter are habitually used for higher-level planning. Subsequently, it is shown in this section that the formulation of the HAPS path planning problem in PDDL (namely PDDL+, a variant of PDDL [Fox and Long,

2006]) is possible. The plannability using a domain-independent planner is also demonstrated. Lastly, the performance and the limits of the approach are tested and presented; results obtained using a domain-independent planner is not inferior to a commonly used motion planning that is capable of solving the HAPS path planning problem, and in some case even outperforms. Finally, a wraparound framework to overcome some of the limits of the domain-independent planner.

3.1 Path Planning Problem for HAPS

Flight path planning from a start to a goal state must consider three important factors [Filippis and Guglieri, 2012]:

1. mission-related requirements (e.g. goal conditions),
2. environment (e.g. other participants[15], obstacles, boundaries of the operation area, vector field),
3. dynamics constraints of the vehicle (e.g. the range of velocity, acceleration).

These, in the case of flight path planning for HAPS, are the weather conditions (i.e. weather critical zones, wind field etc.), aircraft kinematics, and mission requirements (i.e. goal position, optimal reward, mission tasks etc.) respectively.

Path planning is often classified as motion planning in which a state space is defined in the three-dimensional geometric world, along with state validity (i.e. collision checking, existence of path etc.). If differential constraints must be considered[16], control-based planning is necessary, in which a control space must also be defined [Sucan et al., 2012]. Often enough, the state space of many highly optimized motion planning methods is not time-dependent; therefore the methods are limited to rather static environments [Li et al., 2018; Mandalika et al., 2018], which are very convenient for laboratory testing, but could be restrictive in real-world applications. Meanwhile, many advanced domain-dependent planners built for applications with a more dynamic environment also only consider the dynamics of land-based robots or vehicles, that can easily decelerate, halt and wait for the dynamic obstacles to pass [Pecora et al., 2018; Zhou et al., 2018].

HAPS has very atypical properties, as summarized in Table 5, leading to numerous challenges faced in the flight path planning. Due to the temporal factors of the environment and mission requirements, i.e. dynamic obstacles, time-varying wind field, temporarily available operating areas, and the time-dependent goal conditions; thus, the time-dimension must be added to the state space. The following list provides the key difficulties while planning for feasible flight path for HAPS:

[15] An active participant can be another agent/vehicle used to help to fulfill the MR, but at the same time must be avoided, leading to the need of a multi-agent planning method. A passive participant is not involved in the missions and must be avoided; therefore it plays the role of a mere obstacle.

[16] A vehicle subject to differential constraint(s) is also referred to as a nonholonomic vehicle. HAPS is one of the nonholonomic aircrafts [Roussos et al., 2009]. Many works done to study the path planning of nonholonomic ground robots can be found in [Laumond, 1998] However, these works may not be applied directly on HAPS, since the airborne platform is also subject to a vector field (i.e. wind).

1. As a fixed-wing aircraft, HAPS cannot stop in mid-air.
2. HAPS has limited maneuverability (i.e. low turn rate, low climb rate).
3. HAPS operates in a time-varying vector field that can affect its movement, since the airspeed of the vehicle is at the same order of magnitude as the air flow velocity around.
4. Due to the rather slow motion of HAPS, hazardous weather zones cannot be considered static obstacles.
5. Operating in a dynamically managed airspace, the area within which HAPS is allowed to fly is spatially and temporally limited.
6. The spatial-temporal search space is large, since HAPS operates in a wide area to fulfill various mission-related tasks that span long durations.
7. The HAPS team intends to collect as much rewards as possible during the operation; therefore, the goal is to search for the feasible shortest path in terms of travel time, or rather fastest path, instead of the shortest path in terms of distance travelled.

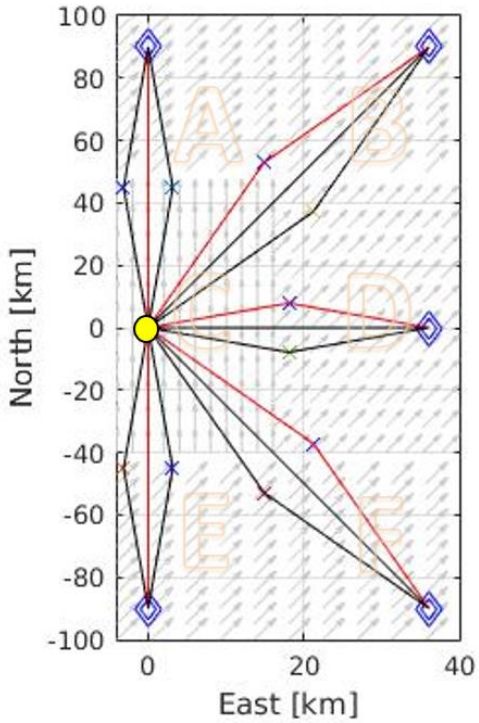


Figure 20. The fastest paths are marked in red. In a wind field, the fastest paths are not necessarily the shortest paths, which are the direct paths the goals.

The above difficulties are also similar to Autonomous Underwater Vehicles (AUV) [Lolla et al., 2012; Lolla et al., 2015; Wolek, 2015] or glider planes [Chakrabarty and Langelaan, 2010; 2013], of which the path planning problems have been studied to consider the water currents for AUV and to exploit the atmospheric energy glider planes. Sophisticated numeric planning capability is necessary to surmount the difficulties of such class of path planning problem and better aid the decision making. Figure 20 depicts a simple example to show the need of advanced numeric computation facilities, and that human abstract perception could be deceiving. In the figure, a HAPS flies at a true airspeed of 28 m/s starting at the origin marked by a yellow circle to different goal positions marked by the blue diamonds.

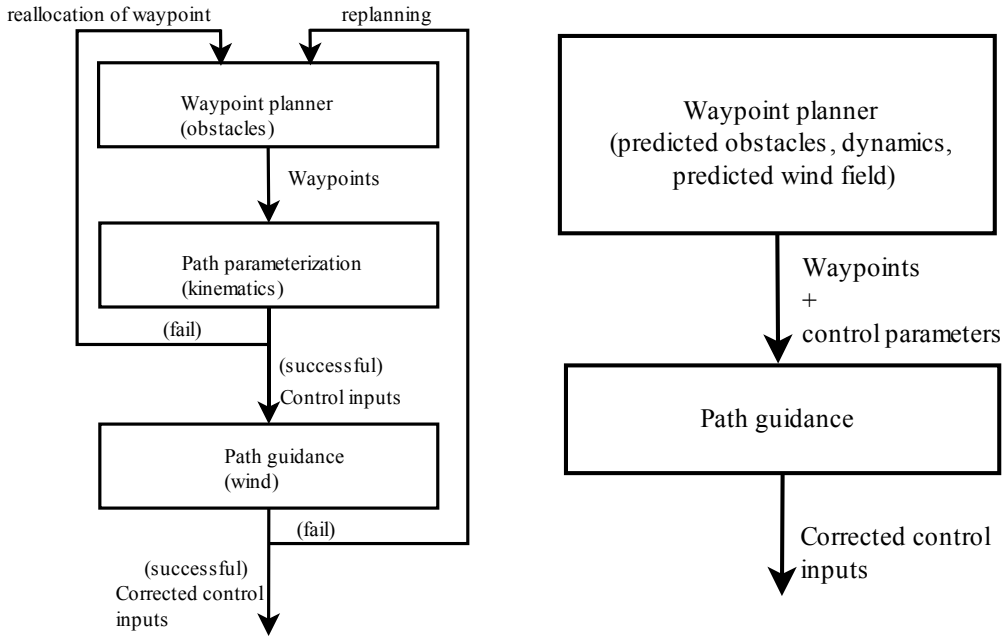
In this example, the wind field is divided into six different regions with regions A, B, D, E, F having a wind velocity of 3 m/s in the North and in the East direction, while region C with 3 m/s in the North direction and 0 m/s in the East direction. Three paths are proposed for each start-goal pair, with the second path (linear and direct path) being the shortest in terms of distance travelled. The paths marked in red, which are not necessary with the shortest distance travelled, are however the fastest paths. This is due to the non-negligible effects of wind on the HAPS dynamics.

Since the differential constraints must be considered in order to obtain flight paths that are dynamically feasible, a control-based path planning is necessary. The next subsection describes how the HAPS path planning problem can be formulated as a control-based path planning problem and cites a few example solving methods.

3.2 Control-Based Motion Planning for HAPS

Most flight path planning problems for fixed-wing aircraft rely on a geometric planning method, for example [Krozel and Andrisani II, 1990; Pehlivanoglu, 2012; Hammouri and Matalgah, 2008] use Voronoi diagram or a regular grid to partition the mission environment and plan. Instead of connecting waypoints linearly, some use Dubins paths to consider the dynamics of the fixed-wing airplane, but leave the effect of wind to the flight controller, as wind is deemed "disturbance" to correct [Owen et al., 2013]. Very often, as pointed out in [Beard R. W. and McLain, 2012], a classical flight path planning process using a geometric motion planner undergoes three steps (see Figure 21a):

1. point-to-point path planning to determine waypoints, which define the flight path that avoid obstacles in the airspace, while assuming that the obstacles are static;
2. path parameterization to determine, based on the dynamics of the aircraft, the course to follow while transiting from one waypoint to the next, and while considering the movement of the obstacles, in which case a reallocation of waypoint could take place if the obstacle avoidance fails;
3. path guidance to ensure that during execution, the planned path is followed despite disturbances (e.g. from wind). If the planned path cannot be followed, a replanning or plan repair is hence necessary.



a) Classical flight planning

b) Flight planning with minimal posterior control

Figure 21. Flow of different flight path planning schemes

This planning structure is however inappropriate in the case of HAPS, especially when the dynamics of the aircraft are considered only after the waypoint planning, and can lead to a frequent waypoint reallocation, since the speed of the dynamic obstacles is at the same order of magnitude as HAPS. Furthermore, although highly efficient in terms of short planning time, the risk of iterative replanning cannot be ignored, since the airspeed of the vehicle is at the same order of magnitude as the wind speed. The iterative replanning can be inconvenient when multiple HAPS are involved to cooperatively fulfill the mission-related tasks. More importantly, the optimality of the plan determined by the waypoint planner can be affected by the reallocation of waypoints and by the wind, since the waypoint planner determines the shortest path with respect to the distance travelled, which may not be the fastest path (see Figure 20).

A more straightforward path planning approach consists of taking into account the obstacles, vehicle dynamics as well as the vector field in the planning phase and requires during execution only minimal control effort as shown in Figure 21b. Such a path planning problem was already considered in Donald et al. [Donald et al., 1993] and LaValle and Kuffner [LaValle and Kuffner, 2001], and is referred to as *kinodynamic planning problem* in which the state space is also subject to kinematic constraints such as obstacles and boundaries to avoid, and is subject to dynamics constraints on the time-derivatives of the vehicle's physical configuration. Conforming with the definition of a state in a kinodynamic planning problem as according to [Donald et al., 1993], each state x of the state space X is defined by $x = (p, q, \dot{p}, \dot{q})$, where p, q are the position and attitude vectors describing the geometrical configurations of the HAPS. If U denotes the control space, the state transition will be governed by

$$\dot{x} = f(x,u), \qquad u \in U, \tag{3-1}$$

where f denotes the process of the movement model of the HAPS, while u denotes the vector of control parameters. In the case of HAPS, as a fixed-wing aircraft with rather weak electro-motors, the aircraft is subject to quite tight operational bounds [Müller et al., 2018], limiting hence the equivalent airspeed that can be commanded, especially in wind field. It is recommended to fly at an equivalent airspeed at 9 m/s, at which the angle of attack is optimal, i.e. the drag is minimal. The equivalent airspeed should also be constant, as any correction command to the speed can result either in a saturation in the engine power or an alteration of flight altitude, as simulation results in [Müller et al., 2018] show. At constant airspeed (optimal airspeed), the roll angle depends on the turn rate, and therefore does not constitute a control parameter. The effect of the vector field, or rather the wind field in HAPS path planning problem, can be included in f. The consideration of wind especially is essential, since wind can affect the ground speed of the aerial vehicle rather substantially or can be harvested as "free" energy to shorten travel time.

3.2.1 Formal Point-to-Point Flight Path Planning Problem Statement

The point-to-point flight path planning problem $\widetilde{\mathcal{P}}^{\mathrm{FP}}_{\mathrm{HAPS}}$ for HAPS is a subset of the problem defined in Section 2.3, and can be represented by a similar tuple

$$< H, X, T, X_0, T_0, C_X, \tilde{A}^{\mathrm{FP}}, \tilde{C}_A^{\mathrm{FP}}, Z_1, \dots, Z_I, C_M, r >,$$

with all elements being identical to the definition in Section 2.3, except for $\tilde{A}^{\mathrm{FP}}$, and $\tilde{C}_A^{\mathrm{FP}}$ being the subsets of A and C_A, i.e. $\tilde{A}^{\mathrm{FP}}$ is the set of discretized yaw rate and climb rate control parameters, and $\tilde{C}_A^{\mathrm{FP}}$ are only the constraints applicable to $\tilde{A}^{\mathrm{FP}}$.

Principally tackled in the section is the point-to-point flight path planning for a single HAPS. The extension to multiple HAPS will be dealt with in Section 5.

3.2.2 Suitable Path Search/ Planning Methods

Such class of planning problem is NP-hard[17] and relies usually on approximation algorithms, sampling-based motion planner being one, to balance optimality and performance [Allen and Pavone, 2015; Webb and van den Berg, 2013]. An approach to solve the kinodynamic planning problem is to use the *Rapidly exploring Random Trees (RRT)* that was first reported in [LaValle, 1998] for holonomic and later extended for nonholonomic planning problems [Kuffner and LaValle, 2000] and in [LaValle and Kuffner, 2001] for a kinodynamic problem. The RRT-based method for kinodynamic planning problems is quite straightforward: a new vertex that represents a state in proximity of the searched tree is chosen randomly and the nearest vertex of the tree to the chosen vertex must be determined. The control parameter u that connects the two vertices, given the movement model f in Equation 3-1, will then be determined. If the new vertex is reachable and does not violate any constraint (i.e. obstacles that are considered global constraints), the new vertex will be added to the

[17] NP-hard (NP: Non-deterministic Polynomial time) is a classification used commonly in theoretical computational complexity to describe a problem of which a given solution can be verified within polynomial time, but the solution cannot be determined within polynomial time, and may even be undecidable.

tree. The configuration space can be explored rapidly by doing so. Although the RRT-based approach in [LaValle and Kuffner, 2001] does not consider a vector field, it can be easily adapted to also consider a time-varying wind in the movement model f, and the dynamic obstacles. However, the faster bidirectional variant consisting of exploring from the initial state and from the goal state, and eventually connecting both trees, can unfortunately not be exploited for our problem, due to the time-varying environment and constraints (i.e. time-varying wind field and moving obstacles).

A flight path planner for a motor-less glider in [Chakrabarty and Langelaan, 2013] uses a *kinematic tree* approach to take wind as well as the flight dynamics into account. A similar approach, but known by the name *Kinematic A** in [Filippis and Guglieri, 2012], is used to plan flight paths for a fixed-wing aircraft in a time-varying wind field while respecting the kinematic constraints of the platform. In order to be more efficient with computational memory, the search for a path is performed by expanding a search tree while considering only the reachable nodes after Δt for each set of the action primitives from the control space, as shown in Figure 22, instead of laying all out like in a classical A*. Given an initial state $x(t_0)$, the search algorithm expands the kinematic tree by appending reachable nodes over each discrete time step found using the following:

$$x(t+\Delta t) = x(t) + \underbrace{f\big(x(t), u(t)\big)}_{\dot{x}(t)}. \tag{3-2}$$

In the case of HAPS flight path planning, x is the state vector of $(\lambda, \varphi, z_h, \chi, \gamma)^{\mathrm{T}}$, where the elements denote the WGS84 coordinates longitude, latitude, altitude, as well as yaw, and pitch angles respectively. u is be a control vector of feasible control parameters $\big(\dot{\chi}(t), \gamma(t)\big)$ from the discretized control space for yaw rate $A_{\dot{\chi}} = \{-|\dot{\chi}_{\max}|, -|\dot{\chi}_{\max}| + \Delta\dot{\chi}, \dots, |\dot{\chi}_{\max}| - \Delta\dot{\chi}, |\dot{\chi}_{\max}|\}$ and for climb angles $A_{\gamma} = \{-|\gamma_{\max}|, -|\gamma_{\max}| + \Delta\gamma, \dots, |\gamma_{\max}| - \Delta\gamma, |\gamma_{\max}|\}$, the union of which constitute $\tilde{A}^{\mathrm{FP}}$, and

$$x(t+\Delta t) = \begin{pmatrix} \dot{\lambda}(t)\Delta t + \lambda(t) \\ \dot{\varphi}(t)\Delta t + \varphi(t) \\ \dot{z_h}(t)\Delta t + z_h(t) \\ \dot{\chi}(t)\Delta t + \chi(t) \\ \dot{\gamma}(t)\Delta t + \gamma(t) \end{pmatrix} \tag{3-3}$$

with $\dot{\lambda}, \dot{\varphi}$ and $\dot{z_h}$ being found by the following equations of movement with a spherical Earth assumption [Müller et al., 2018]:

$$\dot{\lambda}(t) = \frac{v_{\mathrm{w,E}}(t) + v^*_{\mathrm{TAS}} \cos\gamma(t) \sin(\chi(t-1) + \dot{\chi}(t)\Delta t)}{\big(R_E + z_h(t)\big)\cos\varphi(t)}, \tag{3-4}$$

$$\dot{\varphi}(\mathrm{t}) = \frac{v_{\mathrm{w,N}}(\mathrm{t}) + v^*_{\mathrm{TAS}} \cos\gamma(t) \cos(\chi(t-1) + \dot{\chi}(t)\Delta t)}{R_E + \mathrm{z}_h(t)}, \tag{3-5}$$

$$\dot{z_h}(\mathrm{t}) = v_{\mathrm{w,U}}(\mathrm{t}) + v^*_{\mathrm{TAS}} \sin\gamma(t), \tag{3-6}$$

where R_E is the Earth radius, $\left(v_{\mathrm{w,E}}, v_{\mathrm{w,N}}, v_{\mathrm{w,U}}\right)^{\mathrm{T}}$ is the wind velocity vector in East, North and Up directions respectively, and v_{TAS}^{*} is the optimal True Air Speed (TAS) at the altitude the HAPS flies.

[De and Guglieri, 2012] also demonstrated that ideally, dynamic obstacles can be taken into account using the approach, in which variable search steps are also introduced, so that the search timestep can be denser around critical regions, e.g. around an obstacle, and looser in calm regions. However, the validation time-step (marked by the small dots in Figure 22) must be set sufficiently small so that no obstacle is missed, but at the small not too small, so that the plan validation does not become unnecessarily computationally expensive. In practice, 1-2 seconds is sufficient for HAPS for its low airspeed.

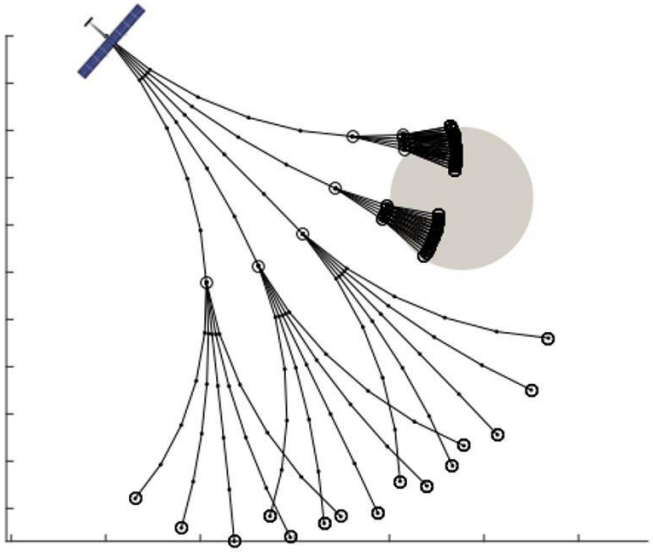


Figure 22. Reachable nodes marked by the hollow circles in the kinematic tree; the gray-filled circle represents an obstacle

3.2.3 Geometric Constraints of the State Space Configuration

The airspace structure as shown in Figure 11 and the various weather data described in [Köhler et al., 2017a] impose conditions to be fulfilled for safety purposes, e.g. “remain in the operation area” or “avoid weather obstacles”. Numerically, these conditions can be represented by time-varying geometric constraints to fulfill. The operation areas (MAs and LOIs) are given as convex polygons for the operation flight level; the weather critical NoGo-areas, as according to [Köhler et al., 2017a], are given as polytopes with identical upper and lower bases. A weather critical NoGo-area is

1. either represented by a single polytope for all flight levels, in the case of aggressive weather hazards, such as cumulonimbus clouds,
2. or represented by a different polytope as each flight level, e.g. strong-wind area, turbulence area etc.

Therefore, for the flight level at which the HAPS operates, the polytopes can be projected onto the plane of the flight altitude, reducing thereby the weather avoidance

problem to two-dimensional. These polygons can be convex or non-convex. In the latter case, a convex polygon encompassing the vertices of the non-convex polygon can be computed using convex-hull algorithms. [Edelsbrunner et al., 1983] introduced the concept of α-hull, which is the intersection of all closed discs of radius $1/\alpha$ containing the given set of points. If α is set to 0, a convex hull is obtained, which is also the least tight encompassment of the set of points. However, convex hulls are convenient for computation, and thus have been used widely in pattern recognition [Yang and Cohen, 1999; Corney et al., 2002] and also in path planning [Meeran and Share, 1997; Gaschler et al., 2013]. Furthermore, the "lack of tightness" in the encompassment of weather critical zones is a reasonable safety margin to allow the error in the forecast and the error in the data extrapolation, since the weather data is timely discretized (e.g. quarter-hourly, hourly, three-hourly etc.). Therefore, in the flight path planning for HAPS, only convex-obstacles are considered.

A convex-hull encompassing a non-convex polygon can be found for example by identifying the nearest neighboring point or by determining the extreme points of the latter. The methods can be found in some classical literature on geometry computation [Graham, 1972; Anderson, 1978; Jarvis, 1973], with a complexity of smaller than $O(n)$, where n is the number of vertices of the non-convex polygon [Anderson, 1978]. These algorithms can be found in standard functions/libraries of Matlab®, for example convhull(), and of C++, for example Computational Geometry Algorithms Library (CGAL) [CGAL, 2012].

3.2.3.1 Checking for Interior Point

If V represents a weather critical area and takes the form of a convex polygon found by one of the above-mentioned standard libraries, it is required to check that the HAPS stays outside the polygon at all times. Let $p = (\lambda_p, \varphi_p)$ be the two-dimensional position vector of the HAPS, and V a set of ordered vertices of a convex polygon, p must remain an exterior point of V. On the contrary, if V represents an operation area, e.g. a MA, it is required that the HAPS stays within the polygon, or rather p is an interior point of V.

A point-in-polygon test [Skala, 2015] using linear inequalities can check if p lies in a convex polygon. For each edge $v_i v_{\overline{i+1}}$[18] of the polygon, where $v_i, v_{\overline{i+1}} \in V$, if p lies on the same side of the edge $v_i v_{\overline{i+1}}$ as an arbitrary interior point of the polygon, then p in included in the polygon. Algorithm 3-1 recapitulates concisely the point-in-polygon test for a convex-polygonal area suitable for the focused application in this work. The first step (see Line 2 of Algorithm 3-1) determines the coefficients of each non-vertical edge $v_i v_{\overline{i+1}}$ of V using the following equations:

$$\begin{aligned} a_i &= -(\varphi_{\overline{i+1}} - \varphi_i), \\ b_i &= \lambda_{\overline{i+1}} - \lambda_i, \\ c_i &= b_i \varphi_i + a_i \lambda_i. \end{aligned} \qquad \text{3-7}$$

[18] In programming, circular indexing $\bar{\imath}$ uses modulus formulas while incrementing or decrementing the index so that it remains within the size of the array (set in this case).

The derivation of the coefficients in Equation 3-7 is recapitulated in Appendix 2. Since a line segment of two non-identical vertices is interior of the convex polygon, we can determine a strict interior point $p_{\text{int}} = (\lambda_{\text{int}}, \varphi_{\text{int}})$ by taking the mid-point of a line segment defined by two non-neighboring vertices (see Line 3 of Algorithm 3-1). To ease the representation of sides, it is fixed that p_{int} fulfills the inequality for all edges i:

$$a_i \lambda_{\text{int}} + b_i \varphi_{\text{int}} < c_i. \qquad 3\text{-}8$$

If this is not the case, the signs of the coefficients must be inverted (see Line 4-6 of Algorithm 3-1). Finally, to check if the HAPS at $p = (\lambda, \varphi)$ is interior of the polygon or not, a conjunction is used (see Line 8-12 of Algorithm 3-1), to ensure that Equation 3-8 is fulfilled for all edges.

Algorithm 3-1	Determine the inclusion of a point in a convex polygon	
Require	a point $p = (\lambda, \varphi)$ and an ordered set of vertices of a convex polygon, V	
1:	**for** each edge $v_i v_{\overline{i+1}}$, where v_i, $v_{\overline{i+1}} \in V$ do	
2:	determine the coefficients a_i, b_i, c_i such that	
3:	determine a strict interior point $p_{\text{int}} = (\lambda_{\text{int}}, \varphi_{\text{int}})$	
4:	**if** $a_i \lambda_{\text{int}} + b_i \phi_{\text{int}} > c_i$	
5:	$a_i \coloneqq -a_i$, $b_i \coloneqq -b_i$, $c_i \coloneqq -c_i$	
6:	**end if**	
7:	**end for**	
8:	**if** $\bigwedge_i (a_i \lambda + b_i \phi \leq c_i)$ **then**	
9:	p is in the convex polygon described by V	
10:	**else**	
11:	p is **NOT** in the convex polygon V	
12:	**end if**	

Note that Equation 3-7 is only valid for non-vertical edges, i.e. $\lambda_i \neq \lambda_{\overline{i+1}}$. If it is a vertical edge, Line 4 and Line 8 of Algorithm 3-1 can be simply replaced by an inequality of $\lambda < a \cdot \lambda_{\text{int}}$, where a is either +1 or -1, depending on which side of the vertical edge the interior point lies.

Weather forecast data is timely discretized, with hourly intervals typically. If the obstacle is dynamic, and the data has to be extrapolated for continuous planning. For example the Cumulonimbus clouds [Köhler et al., 2017a] move along the wind. The movement over time can be extrapolated linearly as in [Kiam et al., 2016; Kiam et al., 2018]. An interpolation over time is not conceivable, since feature points of the weather critical zones are not always recognizable since the form and size could vary tremendously over long time intervals (i.e. at least an hour). If $(v_{w,\text{E}}, v_{w,\text{N}}, v_{w,\text{U}})^{\text{T}}$ denotes the wind speed vector, the coefficients of the edges can be updated as processes with [Kiam et al., 2018]:

$$a(t + \Delta t) = a(t), \qquad 3\text{-}9$$

$$b(t+\Delta t) = b(t),$$
$$c(t+\Delta t) = c(t) + b(t)v_{w,\mathrm{N}}(t)\cdot\Delta t + a(t)v_{w,\mathrm{E}}(t)\cdot\Delta t.$$

Proof:

At time instant t, an edge (line) defined by vertices v_i and v_{i+1} of V can be described by the following linear equation:

$$a(t)\lambda + b(t)\varphi = c(t), \tag{3-10}$$

where, with the parameters expanded using Equation 3-8,

$$\begin{aligned} a(t) &= -\left(\varphi_{v_{i+1}}(t) - \varphi_{v_i}(t)\right), \\ b(t) &= \left(\lambda_{v_{i+1}}(t) - \lambda_{v_i}(t)\right), \\ c(t) &= \left(\lambda_{v_{i+1}}(t) - \lambda_{v_i}(t)\right)\varphi_{v_i}(t) - \left(\varphi_{v_{i+1}}(t) - \varphi_{v_i}(t)\right)\lambda_{v_{i+1}}(t). \end{aligned} \tag{3-11}$$

At time instant $t+\Delta t$,

$$\begin{aligned} a(t+\Delta t) &= -\left((\varphi_{v_{i+1}}(t) + v_{w,N}(t)\cdot\Delta t) - (\varphi_{v_i}(t) + v_{w,\mathrm{N}}(t)\cdot\Delta t)\right) \\ &= a(t)), \\ b(t+\Delta t) &= \left((\lambda_{v_{i+1}}(t) + v_{w,\mathrm{E}}(t)\cdot\Delta t) - (\lambda_{v_i}(t) + v_{w,\mathrm{E}}(t)\cdot\Delta t)\right) \\ &= b(t), \\ c(t+\Delta t) &= \left(\lambda_{v_{i+1}}(t) - \lambda_{v_i}(t)\right)(\varphi_{v_i}(t) + v_{w,\mathrm{N}}(t)\cdot\Delta t) \\ &\quad - \left(\varphi_{v_{i+1}}(t) - \varphi_{v_i}(t)\right)(\lambda_i(t) + v_{w,\mathrm{E}}(t)\cdot\Delta t) \\ &= c(t) + b(t)v_{w,\mathrm{N}}(t)\cdot\Delta t + a(t)v_{w,\mathrm{E}}(t)\cdot\Delta t. \end{aligned} \tag{3-12}$$

Note however that the above movement model does not consider the varying shapes of the clouds, which are too complicated to model. Therefore, a safety margin has to be added to the obstacle polygons to allow for errors in the movement prediction.

3.2.4 **Existing Planner: OMPL**

Motion planning is coined to refer to the computation of discrete motions in the physical space, ranging from moving an arm to moving a vehicle. The physical space or the motions of the agent can be constrained. The Open Motion Planning Library (OMPL) is an open source library implemented in C++, collecting generic implementations of sampling-based motion planning methods [Sucan et al., 2012][19]. OMPL also provides an Application Programming Interface (API) for the definition of the problem, which will then be processed by the planner solver, separating hence the problem modelling and the planning as two independent entities.

[19] OMPL is also integrated as part of the Robot Operating System (ROS).

The generic motion planners available in OMPL can be grouped into two categories: namely *geometric* and *control-based planners*. The former relies on discretizing the physical space into a grid world and assumes that any planned path can be interpolated and turned into a dynamically feasible trajectory, while the latter propagates the state with the vehicle dynamics and thereby enables the generation of feasible motions. Since the expansion of the search tree in a kinodynamic planning problem is subject to differential constraints that take into account the vehicle dynamics, a control-based planner is necessary [Sucan et al., 2012], for which, in addition to the state space, a control space must also be defined, which in the case of HAPS flight path planning is $\tilde{A}^{\mathrm{FP}}$

As summarized in Figure 24, the usual sampling-based planner such as Open Motion Planning Library (OMPL) [Sucan et al., 2012], deals with a kinodynamic motion planning problem, and also that of HAPS, by using an API to setup the state space via the definition of

- a propagation function to define how the state space propagates over time using the first and second derivatives (e.g. integration step, search step, movement model, etc.),
- a collision checking function,
- the start and goal states,
- the control space and its sampling, if a control-based planner is in use,
- the boundaries of the configuration space (i.e. operation areas).

The dynamic obstacles are avoided using a collision checker. However, the operation areas can by default only be defined by imposing a minimum and maximum bound on the state space, thereby restricting the geometry to only regular shapes. A state validation function using Algorithm 3-1 is necessary if the operation area has an arbitrary convex-polygonal shape.

3.3 Domain-Independent Planners and the Standardized Problem Domain Definition Language (PDDL)

One of the main progresses made in AI planning in the past two decades is focused on *domain-independent planners*, i.e. planners that are able to solve a planning problem without knowing what it is about. In other words, the planners are not custom-made for a problem. In general, such planners understand the problem via a modelling language, and translate the problem into a search space the planners are apt to reason [Haslum et al., 2019]. A modelling language for a domain-independent AI planner can be seen as the front-end interface, separating hence the planning problem from the back-end search engine. Such an architecture is advantageous because:

1. a planning problem, once modelled, can be solved by different planners that understand the same modelling language;
2. an alteration in the problem model does not incite the need to adapt the solver;
3. the planner can be used off-the-shelf by experts of the system to be solved, who are not necessarily apt in planning algorithms;
4. the planners are less prone to errors, since the planners are tested extensively;

5. the planning approach has enormous potential to be explainable, given the rigorousness of the modelling language to express the logical causality dependencies [Fox et al., 2017].

Problem Domain Definition Language (PDDL), originated from the International Planning Competition (IPC) in 1998, is a, if not the most widely used, problem modelling language, with its main strength focusing on capturing state transformation via the application of an action. Therefore, PDDL compatible planners are also known as action-based automated AI planners; a plan determined using PDDL is a sequence of actions [Ghallab et al., 2004]. Quoting from the organizing committee of the competition, PDDL was an input language for all participating planners, designed to specify the *physics* of the planning problems as *neutral* as possible without favoring any planning system by providing *advice* [McDermott, 2000]. While competitors encountered insurmountable difficulties to express a hierarchical planning problem with PDDL, the language worked well with classical planning problems.

Soon, the use of PDDL went beyond the IPCs, and has been reported to scale up to huge discrete planning tasks [Silvia Richter, 2010], given the advantages listed above. Although not yet a mature technique, the last advantage on the list above on "explainability" is promising; much work in the AI community is gearing towards increasing explainability, even for complex planning problem. A PDDL validator for example can validate a plan and displays the effects of the plan. If the plan fails, the cause can be identified, and advice is provided to fix the plan. Appendix 4 illustrates an example use of VAL [Howey et al., 2004], a PDDL validator, to highlight the benefits of explainability in Mixed-Initiative Planning (MIP). VAL, however, does not support transcendental functions; therefore, it is not applicable for HAPS. The example in Appendix 4 is based on simpler linear motions of an unmanned ground vehicle.

3.3.1 **Background of PDDL Planners**

Although mainly known as the modelling language for classical planning problems (i.e. deterministic, discrete and non-temporal [Haslum et al., 2019]), since its birth, many extensions have to be made to enable the capturing of more real-world problems involving infinite discrete space (numeric planning), time-stamped actions (temporal planning), continuous processes over time, exogenous events, just to name a few. Figure 23 gives a general overview of the extensions to PDDL since the first version was published at the IPC in 1998. Each extension can be summarized as follows:

1. Classical planning
   All states are Boolean, i.e. the number of reachable states of the system is finite. The effect of an action is either "adding" a true Boolean state to the list or "delete" from the list if it is a negation.
2. Numeric planning
   As the name suggests, state variables can take value of any real number. Such variables are referred to as "fluents". Although still discretized, the possible states of each fluent can be infinite, spanning the real number space. Instead of "add" or "delete" as the effect(s) of an action, a fluent is "updated" and

always takes a numeric value. Together with this extension, arithmetic as well as comparison operators were introduced to express the effects on the fluents. More advanced mathematical expressions are however not excluded.

3. Temporal planning
   Inspired from "scheduling", another class of planning widely addressed by the AI community consisting of a decision for "when to do what". The determined plan is hence a sequence of time-stamped actions, indicating when the action should be executed.
4. Hybrid planning
   Previously, a new effect is always caused by an intended action to be executed. The forward search engine in the motion planner expands the tree by propagating the initial states with the control parameters, i.e. integration over time down to the configuration space, which can be formulated as autonomous processes in PDDL+ with the process first-derivative to update to the first derivation using the selected control parameter. In many real-world automated systems, an effect can best be described as a flow that is repeated without incitement. "Processes" were introduced in the extension of PDDL+ [Fox and Long, 2006]. "Events" were also introduced with the release of PDDL+.

By abuse of language, we simply refer to an automated AI-planner with a PDDL interface as a "PDDL planner", although PDDL itself is only a modelling language.

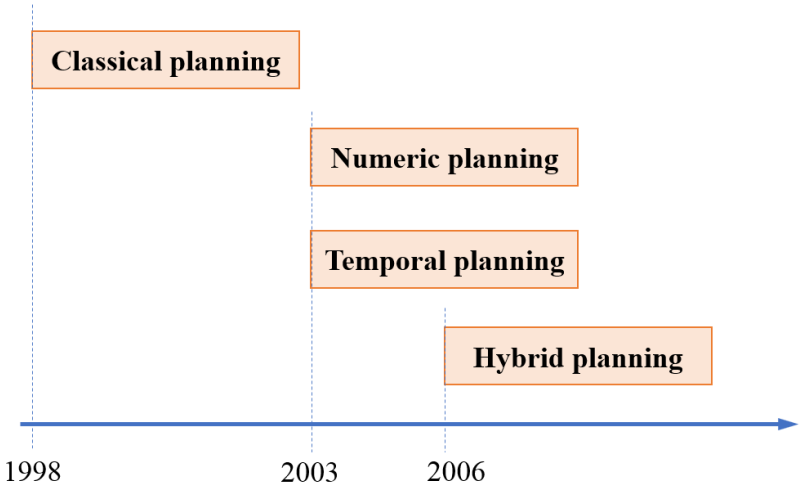


Figure 23. PDDL has come a long way from classical planning to hybrid planning combining numeric and temporal state variables

AI and robotics have, to a certain extent, evolved in a parallel manner, with each community having their own benchmarking methods or competitions, like the IPC for AI-planning and RoboCup for robotics [Lima et al., 2018]. Several works on planning for robotics using PDDL have been reported in [Bernardini et al., 2014] for autonomous drones and [Crosby et al., 2017] for industrial robots. Having witnessed the promising work of PDDL planners in robotics, ROSPlan was created in 2015 [Cashmore et al., 2015] as a tool to bridge a PDDL-planner with the Robot Operating System (ROS) framework. In [Lima et al., 2018], ROSPlan was successfully tested

using the PDDL planner Mercury[20] [Katz and Hoffmann, 2014] to solve problems from the RoboCup competitions, and has also been repeatedly reported to be useful to integrate a PDDL2.1 planner as a task planner within ROS [Antony et al., 2018; Lima et al., 2018].

However, all the mentioned works have only proven the potential of PDDL-compatible planners for solving high-level prepositional planning, while neglecting the handling of more complex numeric problems, such as motion planning in a dynamic and continuous space. Even the attempt to find a path for a robot in a dynamic environment in [Estivill-Castro and Ferrer-Mestres, 2013] considers a grid world represented by logical prepositions (i.e. "at pos0", "at pos1", etc.), instead of a continuous space, and is thereby impossible to consider the physics of the environment and of the robot itself. Motion planners from the robotics community are still favored as solvers, even though with the release of PDDL 2.1 [Fox and Long, 2003] and subsequently of PDDL+ [Fox and Long, 2006], numeric effects an autonomous process can be represented compactly. In the following subsections, PDDL planners are explored to solve the HAPS path planning problem, which requires usually a control-based motion planner [Sucan et al., 2012]. It is interesting to also realize that some fusion of task-based planning, scheduling and classical motion planning can be done within the same framework using a PDDL-planner, gearing planning hence towards a tightly-coupled fashion. The scalability, however, suffers, as discussed at the end of this section.

3.3.1.1 Basic Planning Techniques

Instead of state-space graph, many PDDL planners are based on planning graphs, introduced first formally in [Blum and Furst, 1997], in which the preconditions and effects are linked by actions. Using planning graphs, it is also possible to represent approximate reachability by relaxing some features of an action in the heuristic that guides the search [Bryce and Kambhampati, 2007]. Furthermore, most PDDL planners have level-based heuristics which assume that the cost is the number of levels the search has to traverse to achieve its goal state. For a PDDL planner that uses level-based heuristics, if unit cost actions are considered (i.e. each action bears the same cost, which is the case for the majority of PDDL planners), the search for a plan is often based on the shortest plan length, namely the fewest number of actions or the shortest time in temporal planning.

Theoretically, many major PDDL planners support parameterized domain problems, and hence also parameterized actions (and predicates). Parameterization in a PDDL model is when an action or predicate can be applicable to many objects. However, the first step after parsing the problem domain, most PDDL planners perform what is called "grounding", in which the symbolic parameters of the actions are removed, i.e. each combination of action (or predicate) and its object is seen as an action (or a predicate). If the employed PDDL planner uses grounded actions to establish its search graph, the fewer the grounded actions are, the more efficient the search is. Therefore, instead of having a very generalized action that is applied to many objects with the exceptions expressed using constraints nested in the

[20] Mercury is a temporal planner that supports PDDL2.1.

preconditions, it would be wiser to have multiple specialized actions that are applicable to different classes of objects.

Considering the above commonly used planning techniques, a HAPS flight path planning problem model is established.

3.3.2 **Modelling the HAPS Flight Path Planning Problem in PDDL+**

PDDL+ is used to model mixed discrete-continuous problem domains, also known as hybrid planning problems. More specifically, PDDL+ is a more evolved version of the line of PDDL language variants, providing a convenient formalism for planning problems involving infinite discrete space (numeric planning), time-stamped actions (temporal planning), continuous processes over time, exogenous events etc. [Fox and Long, 2006]. Appendix 3 illustrates a typical planning problem that can be formulated using PDDL+. For a more formal understanding of a hybrid planning problem, the general definition is recapitulated in the following [Fox and Long, 2006]; note that the symbols used are specific for this definition only and do not conform with the list of symbols used for the rest of this work.

Definition 3-1 (Hybrid Planning problem) A planning problem H is given by the tuple $< X_p, X_n, A, P, X_0, G, C >$, where:

- X_p and X_n are the propositional and numeric state variables respectively,
- A is the set of instantaneous actions,
- P is the set of autonomous processes,
- X_0 is the initial state,
- G is the set of goal conditions, and
- C is the set of global constraints.

Actions $a \in A$ are pairs $< pre(a), eff(a) >$, where $pre(a)$ is a set (conjunction) of propositional and numeric preconditions, and $eff(a)$ is a set of effects boolean or numeric expressions indicating instantaneous changes of values in X_p and X_n. An autonomous process $p \in P$ has a continuous effect on variables X_n over time. Like actions, they are a pair $< pre(p), eff(p) >$, where preconditions $pre(p)$ are like those of actions, but effects $eff(p)$ are Ordinary Differential Equations (ODE) $x \coloneqq \exp(e)$, where $x \in X_n$ and $\exp(e)$ can be a well-formed arithmetic expression featuring standard mathematical operators, variables $x \in X_n$, constants or transcendental functions. While being syntactically equivalent to action precondition, a process precondition expresses an invariant condition along the execution of the process itself. Their violation causes the process to stop, so switching in what the hybrid automaton literature calls, another mode of execution. Global constraints $c \in C$ are arbitrary quantified-free formula over variables in X_n and X_p. They must be satisfied by any state throughout the plan timeline.

More details on the semantics aspects of PDDL+ can be found in [Fox and Long, 2006]. Compatible domain-independent PDDL+ planners such as UPMurphi [Della Penna et al., 2009], DiNo [Piotrowski et al., 2016], ENHSP [Scala et al., 2016] etc., are theoretically able to solve the modelled problems. Solutions to H are plans, sequences of time-stamped actions $a \in A$. More complex real-world problem have been successfully modelled using PDDL+ and solved by the compatible planners to

optimize the processes in a chemical plant or to control urban traffic [Vallati et al., 2016; Della Penna et al., 2010].

Starting from the observation that in PDDL+, it is possible to separate the decisions of the actions to take from the dynamics of the system (by using actions and processes), whilst making sure that a set of global constraints remain satisfied along the resulting plan, in this subsection, it will be demonstrated how PDDL+ can be used to model the HAPS flight path planning problem described in Section 3.1 and 3.2, which is commonly classified as a kinodynamic motion planning problem.

Figure 24 first illustrates the analogy observed between the formulation of a general kinodynamic motion planning problem in PDDL+ and the problem modelling using the API of OMPL, a typical motion planning library. The sampling of the control space can be mapped to actions in PDDL+, i.e. if a `second-derivative` is a control parameter, the planner can choose to increase or decrease the parameter by the sampling step `delta-second-derivative`, as the actions `increase_second_derivative_p1` and `decrease_second_derivative_p1` shown in Figure 24 for the state variable `configuration_p1`. This, in a motion planner, corresponds to selecting an action in the control space made up of parameters of the second derivative $\ddot{p}$, but subject to $\|\ddot{p}\| < a$. This condition is also formulated as preconditions in the actions `increase_second_derivative_p1` and `decrease_second_derivative_p1` in PDDL+. The kinematic constraints (e.g. obstacles or bounds of the configuration space) can be encoded as conditions to fulfill. If these constraints are geometrically convex polygons, Algorithm 3-1 is used to check if the HAPS is an interior point or not. The check using inequalities can be compactly formulated as conditions of an action, a process or an event using the `exists` operator, as shown in Figure 24. If the constraints are global, i.e. they must be fulfilled at all time instants, like obstacles to avoid for safety purposes, an encoding using a global constraint with the prefix `:constraint` is possible. Although the `exists` operator is not a compulsory requirement of the PDDL+ language, it is sometimes supported by PDDL planners and is, but uses a disjunction instead. By using the `exists` operator, the convex polygon can have an arbitrary number of edges. Likewise, the global constraint is not part of the standard PDDL+ language, but is supported by many planners, as it is more compact and natural [Haslum et al., 2019], compared to encoding the constraints as preconditions to meet for every action, process and event.

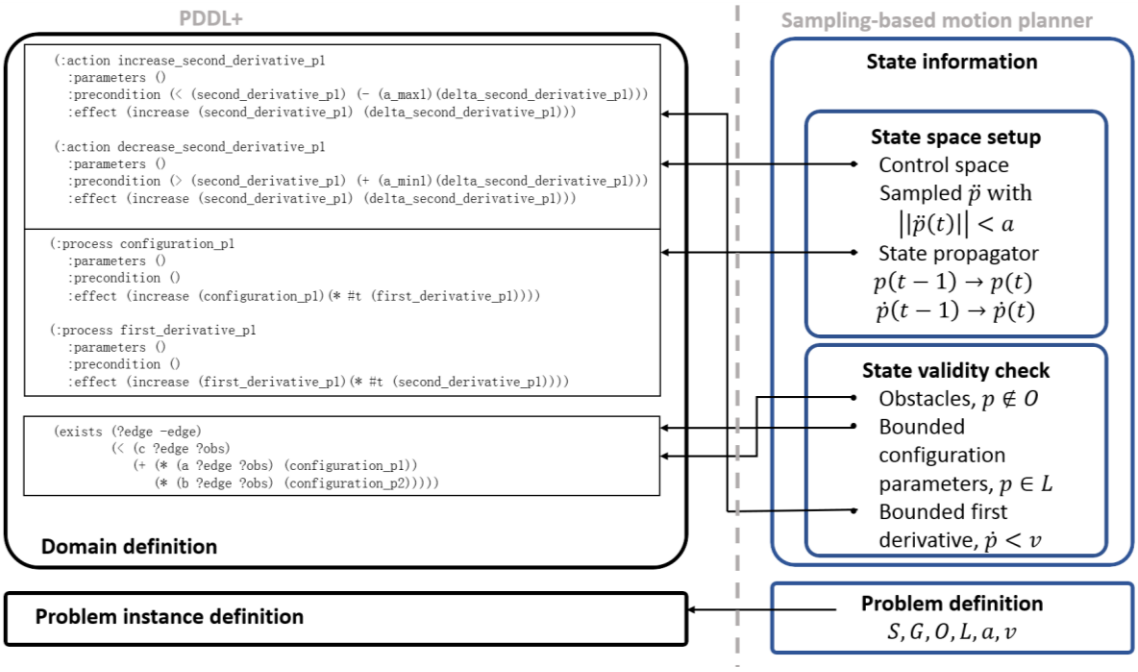


Figure 24. Analogy between the problem definition for a classical sampling-based motion planner and the formulation in PDDL+

Both structures admit in general that a planning problem definition can be divided into two parts: a symbolic/ functional definition of the system and an instantiation of system parameters. While in OMPL, the division is not clear, for example, the default state space boundaries are within the system instantiation, the division in PDDL is strict. The system is defined in the *domain* file, while the parameter instantiation is in the *problem* file.

As can be observed from the formulations, PDDL+, like all PDDL languages, uses a declarative formalism inherited from the LISP programming language. The syntax rules are not as rigorous and strict as in other programming languages like C, C++, Java, etc.; different planners accept different naming, which unfortunately limits in practice the use of multiple planners to solve for the same planning problem without much adaptation. A few invariants are however worth noticing to better understand the modelling language [Haslum et al., 2019]. Keywords that are not a logic operator and are not `domain` or `define` are always preceded by a colon (`:`). A parameter is always preceded by a question mark (`?`) and is separated from its type name proceeded by a hyphen (`-`). These symbols are therefore reserved and are not to be used as the beginning of names. Similar to LISP, the operator or action in every expression precedes the variables (i.e. "fluents" for infinite variables and "predicates" for binary variables).

Using the analogy observed between the definition of a motion planning problem and its modelling using PDDL+, Table 11 shows the modelling for the HAPS flight path planning in PDDL+. Each row in white shows the motion planning model, while the following gray row shows the representation in PDDL+.

Table 11. Modelling the HAPS kinodynamic planning problem with PDDL+

Motion planning

Initialization of the zero-order state space

$\boldsymbol{x} = \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{\varphi} \\ \boldsymbol{z} \\ \boldsymbol{v} \\ \boldsymbol{\chi} \\ \boldsymbol{\gamma} \end{pmatrix}$, with $\boldsymbol{\lambda}, \boldsymbol{\varphi}, \boldsymbol{z}$ being the longitude, latitude and altitude, $\boldsymbol{v}$ being the optimal speed of the HAPS, $\boldsymbol{\chi}$ and $\boldsymbol{\gamma}$ being the yaw and the climb angle.

PDDL+

Represented by fluents in the domain definition and are initialized in the problem definition

```
(:functions
  (lambda ?haps -haps)
  (phi ?haps -haps)
  (z ?haps -haps)
  (speed ?haps -haps)
  (chi ?haps -haps)
  (gamma ?haps -haps)
)
```

Motion planning

Control space

$u = (\dot{\chi}, \gamma)^{\mathrm{T}}$, the control vector, with $\dot{\chi}$ being the yaw rate and χ being the climb angle, selected from

$$A_{\dot{\chi}} = \{-|\dot{\chi}_{\max}|, -|\dot{\chi}_{\max}| + \Delta\dot{\chi}, \dots, |\dot{\chi}_{\max}| - \Delta\dot{\chi}, |\dot{\chi}_{\max}|\} \text{ and}$$
$$A_{\gamma} = \{-|\gamma_{\max}|, -|\gamma_{\max}| + \Delta\gamma, \dots, |\gamma_{\max}| - \Delta\gamma, |\gamma_{\max}|\},$$

where $\Delta\dot{\chi}$ and $\Delta\gamma$ denote the discretization step of the control space and $|*_{\max}|$ the maximum magnitude.

PDDL+

Control parameters are chosen by the following actions

The parameter `?haps -haps` is to identify which HAPS the action is applied to.

```
(:action increase_turn_rate
  :parameters (?haps -haps)
  :precondition
    ((< (chi_rate ?haps)
        (- (max_chi_rate ?haps) (delta_chi_rate ?haps)))
  :effect
    (and (increase (chi_rate ?haps)
                   (delta_chi_rate ?haps)))
```

```
(:action decrease_turn_rate
  :parameters (?haps -haps)
  :precondition
    ((> (chi_rate ?haps)
        (+ (min_chi_rate ?haps) (delta_chi_rate ?haps)))
  :effect
    (and (decrease (chi_rate ?haps)
                   (delta_chi_rate ?haps)))

(:action increase_climb_angle
  :parameters (?haps -haps)
  :precondition
    ((< (gamma ?haps)
        (- (max_gamma ?haps) (delta_gamma ?haps)))
  :effect (and (increase (gamma ?haps)
                         (delta_gamma ?haps)))

(:action decrease_climb_angle
  :parameters (?haps -haps)
  :precondition
    ((> (gamma ?haps)
        (+ (min_gamma ?haps) (delta_gamma ?haps)))
  :effect (and (decrease (gamma ?haps)
                         (delta_gamma ?haps)))
```

Motion Planning

Convex polygonal obstacles

As explained in Section 3.2.3.1, an interior point $p = (\lambda, \varphi)$ of a convex polygon lies on the same side of each edge i of the polygon as an arbitrary interior point $p_{\text{int}} = (\lambda_{\text{int}}, \varphi_{\text{int}})$, i.e. $a_i\lambda + b_i\varphi \leq c_i$, if $a_i\lambda_{\text{int}} + b_i\varphi_{\text{int}} < c_i$, where a_i, b_i, c_i are parameters of the edge i. Logically, a conjunction can be used to ensure that the inequality holds true for all edges:

$$\wedge_i \, (a_i\lambda + b_i\varphi \leq c_i).$$

The HAPS must not be an interior point of an obstacle, i.e. there must be at least an edge i, where

$$a_i\lambda + b_i\varphi > c_i.$$

The latter can be tested using the following disjunction:

$$\vee_i \, (a_i\lambda + b_i\varphi > c_i).$$

PDDL+

The condition that at least one edge must exist, of which the HAPS does not lie on the same side as an interior point, can be encoded using the `exists` quantificator to test the disjunction. The disjunction is nested in a global constraint.

```
(:constraint convex_Cb_like_obstacle
  :parameters (?obs -obstacle ?haps -haps)
  :condition
    (exists (?edge -edge )
            (> (c ?edge ?obs)
               (+ (* (a ?edge ?obs) (lambda ?haps))
                  (* (b ?edge ?obs) (phi ?haps))))))
```

Motion Planning

Linear movement of the convex polygonal obstacle

As described in Section 3.2.3.1, the parameters of the edges can be updated with

$$a(t + \Delta t) = a(t),$$
$$b(t + \Delta t) = b(t),$$
$$c(t + \Delta t) = c(t) + b(t)v_{w,\mathrm{N}}(t) \cdot \Delta t + a(t)v_{w,\mathrm{E}}(t) \cdot \Delta t.$$

PDDL+

The update of the *c* edge parameter can be performed using an integration over time nested in an automated process.

```
(:process update_dynamic_obstacle_edge_parameter
  :parameters (?edge -edge ?obs -obstacle)
  :precondition ()
  :effect
    ((increase (c ?edge ?obs)
               (+ (* (* (b ?edge ?obs) (wind_v ?obs)) #t)
                  (* (* (a ?edge ?obs) (wind_u ?obs))
                     #t))))
```

Note that `(a ?edge)` and `(b ?edge)` are invariant.

Motion Planning

Remain in a convex polygonal operation area

HAPS must be an interior point of the operation area. Therefore, the conjunction
$\wedge_i\ (a_i\lambda + b_i\varphi \leq c_i)$ must hold true, which, by De Morgan's laws, the negation of the disjunction $\vee_i\ (a_i\lambda + b_i\varphi > c_i)$.

PDDL+

The encoding is similar to a convex polygonal obstacle and can be expressed using the negation of a disjunction expressed with the `exists` operator.

```
(:constraint in_operation_area
  :parameters (?ops -operation_area ?haps -haps)
  :condition
    (not (exists (?edge -edge )
         (> (c ?edge ?ops)
```

```
            (+ (* (a ?edge ?ops) (lambda ?haps))
               (* (b ?edge ?ops) (phi ?haps)))))))
```

Motion Planning

Determine the optimal airspeed

HAPS, as a fixed-wing aircraft, flies at the given optimal Equivalent AirSpeed (EAS) of ~9 m/s [Müller et al., 2018], which can then be scaled using the following equation to obtain the TAS at different altitude levels:

$$v^*_{\mathrm{TAS}} = v^*_{\mathrm{EAS}} \cdot \sqrt{\rho_0/\rho(z)},$$

where $\rho(z)$ and ρ_0 are respectively the ambient and seal level air densities given by the International Standard Atmosphere.

PDDL+

If the optimal TAS must be determined at each time instant, due to varying pressure (because of the varying altitude), an event can be used.

```
(:event determine_optimal_airspeed
  :parameters (?haps -haps ?z_level -z_level)
  :precondition (and (<= (z ?haps)
                         (z_max ?altitude_level))
                     (> (z ?haps) (z_min ?z_level)))
  :effect
    (and (assign (speed ?haps)
                 (* (speed_eas ?haps)
                    (^ (/ (rho_0) (rho ?z_level)) 0.5))))
```

The precondition is to check at which altitude level the HAPS is currently.

Motion Planning

Keep track of time

In temporal planning, it is also essential to keep track of the system time, which in a discrete world is simply performed by incrementing the time variable $time\ (n+1) = time(n) + \Delta t$.

PDDL+

The automatic update of the system current time can be encoded as a process.

```
(:process update_current_time
  :parameters ()
  :effect(increase time #t))
```

Motion Planning

Determination of the wind vector

The four-dimensional wind field provided can be viewed in the physical space as a polytope shown in Figure 25. Some polytopes have identical rectangular upper and lower base, while others are convex polygonal.

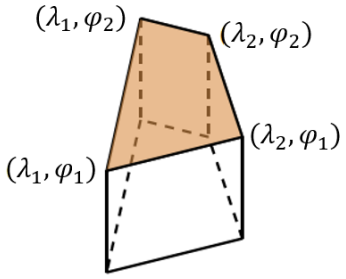


Figure 25. Visualization in 3D of the polytope of a windfield cell

PDDL+

Like the optimal TAS, the wind vector acting on the HAPS can be assigned at each time instant with values taken from the weather forecast data. An event can be used to determine in which four-dimensional wind grid cell the HAPS is situated, and thereby also the wind vector.

```
(:event determine_u_wind
  :parameters (?haps –haps ?wind_grid -wind_grid
               ?z_level -z_level
               ?time_interval -time_interval)
  :precondition (and
    (< (z ?haps) (z_level_max_bound ?z_level))
    (>= (z ?haps) (z_level_min_bound ?z_level))
    (< time (end_time_interval ?time_interval))
    (>= time (start_time_interval ?time_interval))
    (not (exists (?edge -edge )
         (> (c ?edge ?wind_grid)
            (+ (* (a ?edge ?wind_grid) (lambda ?haps))
            (* (b ?edge ?wind_grid) (phi ?haps))))))
  :effect (and
    (assign (north_wind ?haps)
       (north_wind ?wind_grid ?z_level ?time_interval))
    (assign (east_wind ?haps)
       (east_wind ?wind_grid ?z_level ?time_interval))))
```

The exists operator is used for the case where the wind grid in the three-dimensional space has cells of convex polygonal bases. If the bases are rectangular, simple inequalities are sufficient to determine the wind grid in which the HAPS is situated:

```
(>= (lambda ?haps) (lambda_min ?wind_grid))
(< (lambda ?haps) (lambda_max ?wind_grid))
(>= (phi ? haps) (phi_min ?wind_grid))
(< (phi ? haps) (phi_max ?wind_grid)))
```

Motio

Equations of motion

Update of heading
$\chi(t+1) = \chi(t) + \dot{\chi}(t+1)\Delta t$, where

$\dot{\chi}(t+1) = \dot{\chi}(t) + \Delta\dot{\chi}$

Update of longitude
$\lambda(t+1) = \lambda(t) + \dot{\lambda}(t+1)\Delta t$, where
$\dot{\lambda}(t) = \frac{v_{\mathrm{w,E}}(t) + v^*_{\mathrm{TAS}} \cos\gamma(t) \sin(\chi(t))}{(R + z_h(t)) \cos\varphi(t)}$

Update of latitude
$\varphi(t+1) = \varphi(t) + \dot{\varphi}(\mathrm{t}+1)\Delta t$, where
$\dot{\varphi}(\mathrm{t}) = \frac{v_{\mathrm{w,N}}(\mathrm{t}) + v^*_{\mathrm{TAS}} \cos\gamma(t) \cos(\chi(t))}{R + \mathrm{z}_h(t)}$

Update of altitude
$z(t+1) = z(t) + \dot{z}(\mathrm{t}+1)$, where
$\dot{z_h}(\mathrm{t}) = v_{\mathrm{w,U}}(\mathrm{t}) + v^*_{\mathrm{TAS}} \sin\gamma(t)$

PDDL+

The position and attitude of the HAPS can be updated using automated processes. `#t` is the integration time step.

```
(:process update_heading
  :parameters (?haps -haps)
  :precondition ()
  :effect (and (increase (chi ?haps) (chi_rate ?haps))))

(:process update_longitude
  :parameters (?haps -haps)
  :precondition ()
  :effect
    ((increase (lambda ?haps)
               (* #t (/ (+ (* (speed ?haps)
                              (* (cos (gamma ?haps))
                                 (sin (chi ?haps))))
                           (east_wind ?haps))
                        (* (+ R (altitude ?haps)
                           (cos (phi ?haps)))))))))

(:process update_latitude
  :parameters (?haps -haps)
  :precondition ()
  :effect
    ((increase (phi ?haps)
               (* #t (/ (+ (* (speed ?haps)
                              (* (cos (gamma ?haps))
                                 (cos (chi ?haps))))
                           (north_wind ?haps))
                        (+ R (z ?haps)))))))

(:process increase_altitude
  :parameters (?haps -haps)
  :precondition ()
```

```
  :effect
    ((increase (z ?haps)
               (* #t (+ (* (speed ?haps)
                           (sin (pitch ?haps)))
                        (up_wind ?haps))))
```

3.3.3 **HAPS Flight Path Planning Problem Using an Automated AI Planner**

While quite a number of domain-independent planners have been developed for some fragment of the PDDL+ semantics [Gerevini et al., 2003; Della Penna et al., 2010; Coles et al., 2012; Cashmore et al., 2016], only until recently domains with non-linear dynamics have been supported more effectively [Piotrowski et al., 2016; Scala et al., 2016], i.e. f in Equation 3-2 is non-linear, due to Equations 3-3 to 3-6. In particular, Expressive Numeric Heuristic Search Planner (ENHSP) [Scala et al., 2016] offers support to trigonometric functions and global constraints, which are of critical importance to our application, as can be observed from the automated processes used to update the HAPS configuration in Table 11.

ENHSP[21] is a heuristic search forward state planner expanding the tree rooted at the initial state [Ghallab et al., 2004; Geffner and Bonet, 2013]. Like many automated planners, ENHSP uses heuristic (an approximated cost) to accelerate the search for a plan. However, completeness and optimality are not guaranteed. The heuristic component of ENHSP is a general algorithm that computes automatically and efficiently a relaxation of a given planning problem H (Definition 3-1) for each state in the search tree. The relaxed problem is often represented by $H+$, in which the reachable values of the numeric variables are approximated and bounded. Such relaxation technique is also known as interval-based relaxation and is used by many numeric planners [Hoffmann, 2003; Scala et al., 2016], since it reduces the complexity of the automated planning [Aldinger et al., 2015]. $H+$ is then readily solved by whatever methods deemed suitable to produce a heuristic estimate of the sequence of transitions required to reach goal states.

ENHSP heuristic component, the Additive Interval-Based Relaxation (AIBR) heuristic, has been shown experimentally to provide effective guidance, thus limiting the size of the search tree considered over a very diverse set of domains [Scala et al., 2016]. ENHSP includes many search methods, namely the weighted A*, greedy weighted A* (i.e. with a quadruple times weighting on the heuristic), greedy best-first search, depth-first search, uniform cost search, enhanced hill climbing.

Also configurable in ENHSP is the integration time step `#t` and can be set different for the search (search time step $\texttt{\#t}_s$) and for the plan validation (validation time step $\texttt{\#t}_v$), in which the state is properly determined and the constraints (e.g. collision with an obstacle) are checked. By setting a bigger $\texttt{\#t}_s$, the complexity of the search is reduced. By setting a reasonably small $\texttt{\#t}_v$, the computed plan deviates less from the execution, since the flight controller usually works at a smaller integration time step; furthermore, small obstacles will not be missed.

[21] ENHSP is available on https://gitlab.com/enricos83/ENHSP-Public.

As shown in Figure 24, the HAPS system dynamic is encoded formally in the domain file, while the variables of the system are instantiated in the problem file. Although PDDL is a modelling language for domain-independent planners, the currently available automated planners unfortunately cannot cope with large problem. The modelling of the problem must be carried out in a way so compact that only necessary actions and variables are involved. For example, during the operation, the HAPS maintains its flight altitude; it is therefore more efficient to leave out the climb angle and set the altitude to a constant. The handbook by Haslum et al. [Haslum et al., 2019] provides more details about the encoding of PDDL and how it affects the planning efficiency, together with reports on the usability of PDDL, as well as the difficulties one might face.

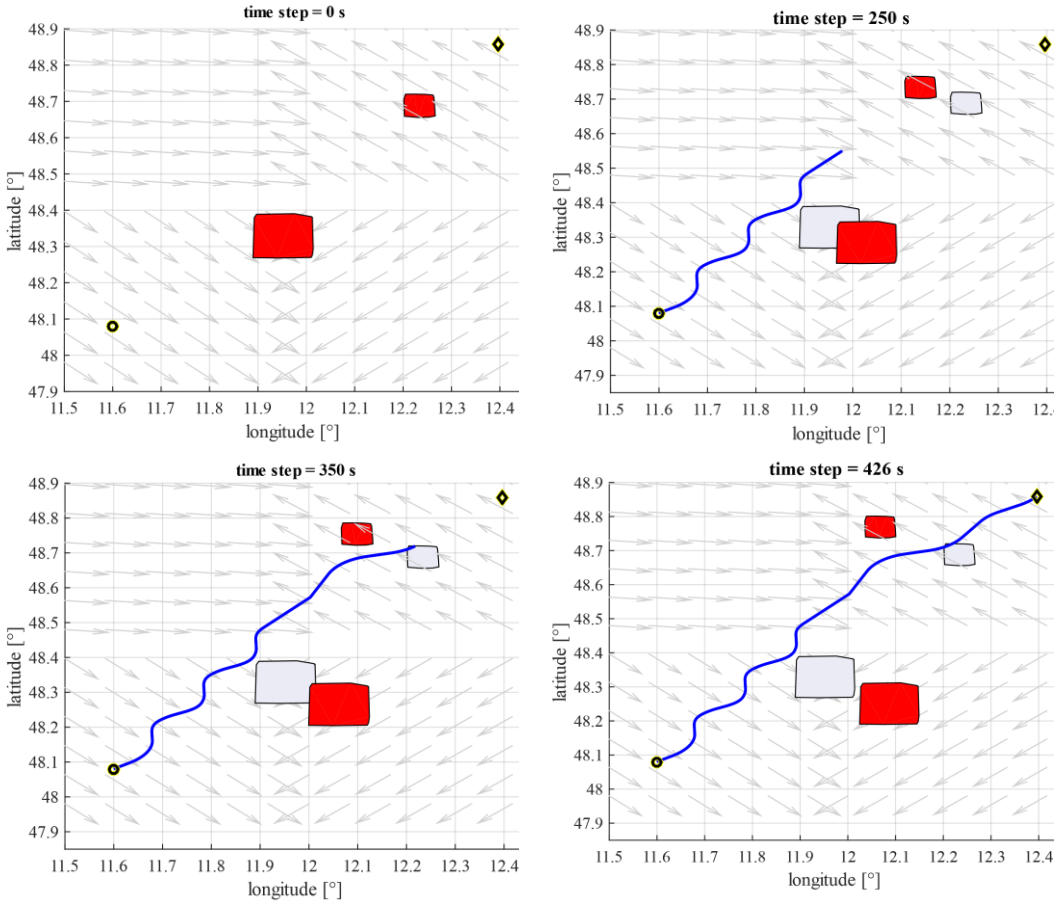


Figure 26. Path planning in the presence of wind and moving obstacles

Figure 26 shows a planned path from a start to a goal position in the presence of wind, while avoiding two moving obstacles. The wind vectors were randomly generated for the four grid cells, with $(v_{w,\mathrm{E}}, v_{w,\mathrm{N}})$ being $(4.9388, -0.44205)$ at the upper-left cell, $(-3.119, 2.3049)$ at the upper-right cell, $(2.5775, -2.2143)$ at the lower-left cell, and $(-3.3729, -2.6587)$ at the lower-right cell. The convex-polygonal obstacles are dynamic and move linearly along the wind. Their position can be updated using Equation 3-9. In the figure, the obstacles shown in red are the current positions, while in light gray are the initial positions. The $\#t_s$ and $\#t_v$ were set to 100 s and 1 s respectively. The plan to fly from the start (marked with a yellow circle) to the goal (marked with a yellow diamond) was obtained within 3 s.

Since the performance of the planner also depends on the planer configuration, systematic tests on the performance of ENHSP were carried out. The next subsection reports the systematic performance tests and analyses the obtained results, specifically on how a "fine tuning" in the implementation can improve the planning efficiency. Additionally, a benchmarking against the RRT-based motion planner from OMPL will also be reported. Section 7.2.1 will be dedicated to proving the correctness of the problem model by demonstrating the executability of the computed flight plans.

3.3.3.1 Systematic Performance Tests and Benchmarking

Hooker suggested in [Hooker, 1995] that comparing performance (i.e. planning time) between planners is not the sole means to measure the efficiency of the heuristics used in the planners, as such comparison could be biased, since the planner that is compared against may not be tuned or engineered properly for the problem in question. On the other hand, in many cases, systematic tests are more constructive to help to understand how the heuristic performs, and why it fails with one problem but excels with another. With the knowledge on the behavior of the heuristic, it can be exploited and engineered properly to help improve planning performance.

Although heuristic is not the main focus here, the idea from Hooker briefly mentioned above is adopted in this work. Given that ENHSP is used off-the-shelf, it is necessary helpful to understand the behavior of the planner, hence allowing better engineering in the problem modelling. To evaluate the robustness of ENHSP in handling the problem, systematic tests were performed by generating a variety of instances, differing in each set only one test parameter setting. The parameters used to test the performance of the planner are the dimension of the operation area, wind magnitude, number of obstacles, obstacle occlusion ratio, distance from the goal, and initial bearing with respect to the goal. The latter is the angle difference b between the initial course heading of the HAPS and the heading of the initial start-goal vector, as shown in Figure 27.

The tests were performed assuming a constant altitude at 18 km, an optimal EAS of v^*_{EAS} = 9.68 m/s, and as Earth radius $R_E = 6371.28$ km.

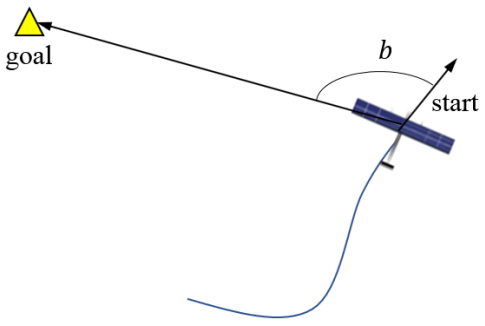


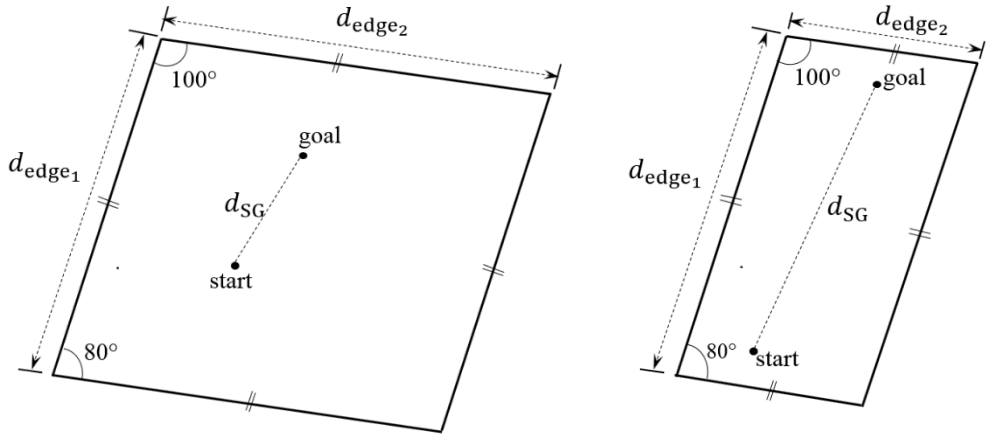
Figure 27. Initial bearing

Performance Tests in Wide Operation Areas

In this set of test configurations, no obstacle is present. The aim is to test the planning efficiency of ENHSP with different search time step $\#t_s$ for different distances to goal and different initial bearings to goal b. The start-goal distances selected are reasonable for the use case shown in Figure 10.

Table 12. Parameter configuration for performance tests in wide operation areas

$\#t_s$	$\#t_v$	b	d_{SG}	TO
{30, 100, 150} s	1 s	{20, 100, 180} °	{30,50,80,110,140} km	5 s



a. Wide area

b. Narrow area

Figure 28. Wide and narrow operation areas used for the tests

A test configuration is a combination of the parameter values taken from the corresponding parameter value sets in Table 12. Each bar of Figure 29 indicates a test configuration; for each configuration, 20 tests were run, each with randomly chosen configuration parameters that are not fixed yet, namely the dimension of the operation area and the wind vector. A wide operation area is a quadrilateral area like the mission areas shown in Figure 10. For the tests, a wide area takes the form of the quadrilateral area shown in Figure 28a, where the edge dimensions d_{edge_*} are selected randomly within the range $[2 \cdot d_{SG}, 4 \cdot d_{SG}]$, with d_{SG} being the start-goal distance. The wind magnitude is selected randomly at each test between 0-5 m/s, which is usual at altitudes of ~ 18 km).

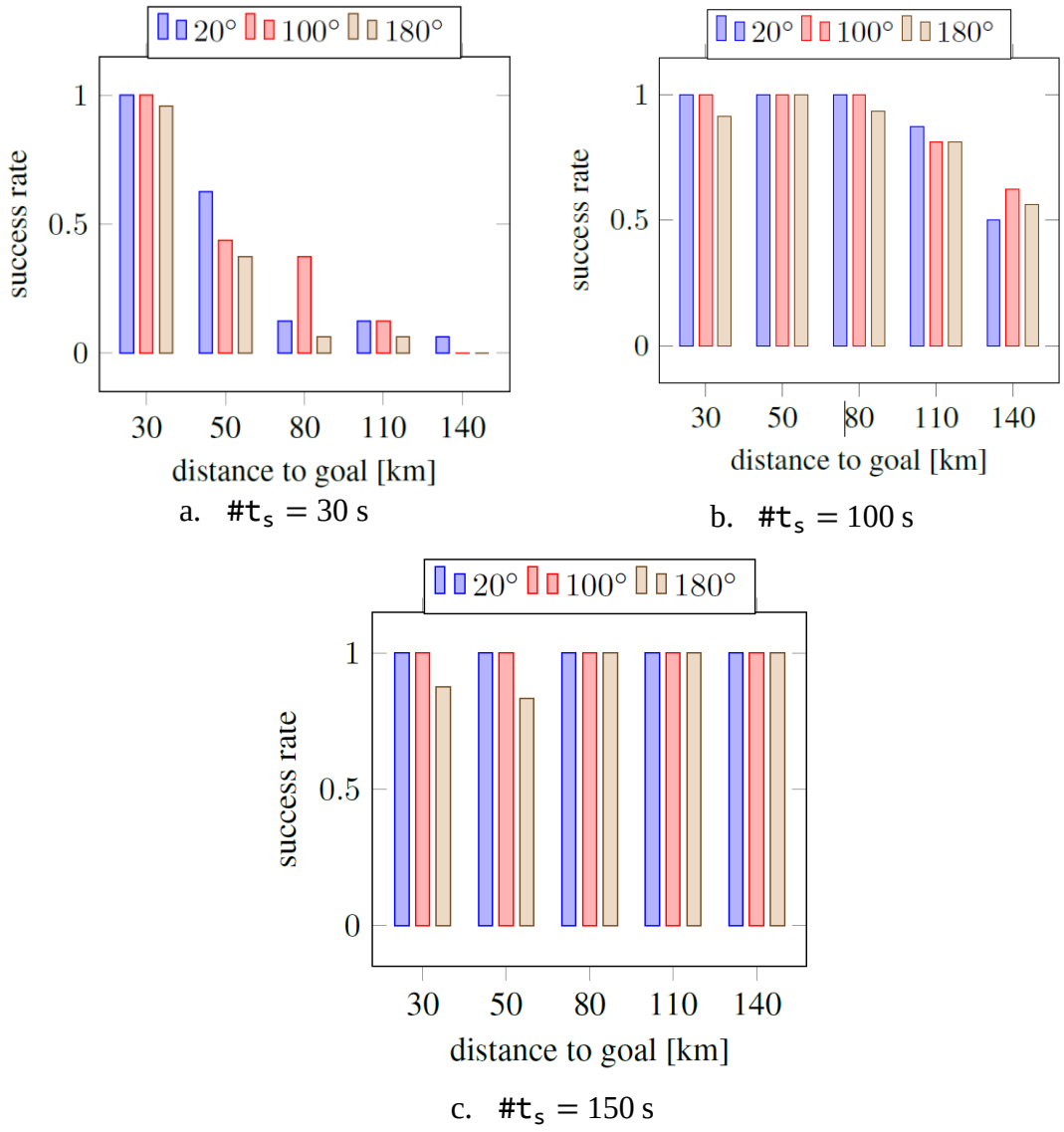


Figure 29. Success rate to plan within 5s in a wide operation area for different distances from start to goal and different initial bearings $\{\mathbf{20°}, \mathbf{100°}, \mathbf{180°}\}$

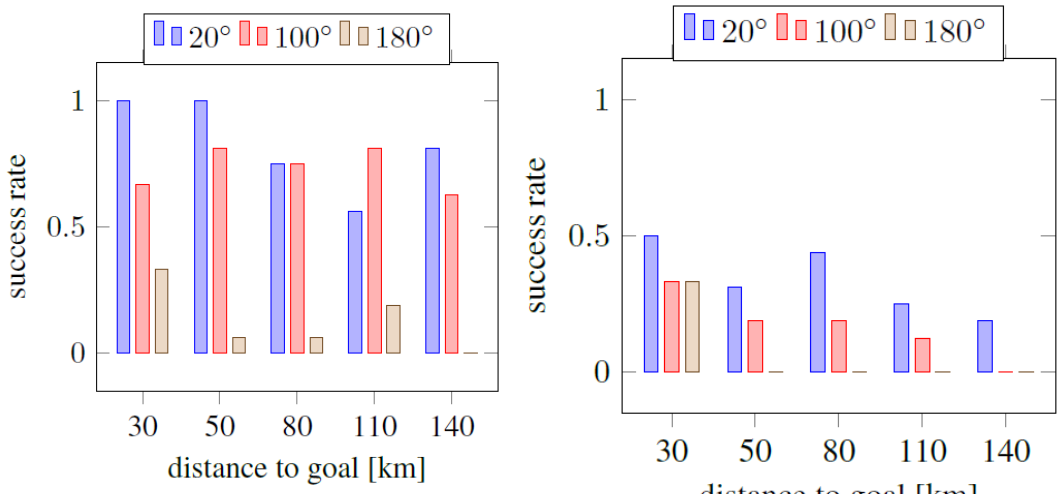
Empirical observations show that in the absence of obstacles, a plan is obtained within a few seconds, or the search could take an unreasonable (i.e. several minutes or up to several hours) or eventually an infinite amount of time. Furthermore, for practical reasons, obtaining a point-to-point flight path plan from a start to a goal position within seconds is necessary, since within the complete mission planning time (i.e. maximum 15 minutes) allocated for calculating a plan that spans the day, multiple point-to-point flight path planning is required (see Section 7.1). In the experiments, and also in the implementation (described in Section 7.1), the planning time out for finding a flight path plan from a start to a goal position is set to 5 seconds. The performance tests were conducted to evaluate the success rate in computing a plan within the imposed planning time out.

Figure 29 shows the test results of the performance of the planner for a flight path planning from a start to a goal position in a wide operation area at a constant altitude, reducing thereby the problem to two dimensions. The search time step $\#t_s$ selected

are 30, 100 and 150 seconds. As visible in Figure 29, a bigger search step improves tremendously the planning efficiency, especially when the distance to the goal is larger. The results confirm the expectation that the bigger the search time step is, the smaller the search space is, thereby the less complex the search is. However, if the distance to the goal is 50 km, a $\#t_s$ of 100 s is preferred.

Performance Tests in Narrow Operation Areas

The planner however behaves differently in a narrow operation area, i.e. the smallest diagonal of the quadrilateral operation area is smaller than the start-goal distance. Figure 30 shows the success rate of finding a plan within 5 seconds with the same parameter variation as the previous set of tests. The operation areas are reduced in this set of tests to narrow corridor-like areas, as shown in Figure 28b, with d_{edge_2} being selected randomly from $[1.1 \cdot d_{\text{SG}},\ 1.3 \cdot d_{\text{SG}}]$, while d_{edge_2} being selected randomly from $[0.3 \cdot d_{\text{SG}},\ 0.8 \cdot d_{\text{SG}}]$. The operation area must be rotated or repositioned so that the start and goal points are included.



a. $\#t_s = 100$ s

b. $\#t_s = 150$ s

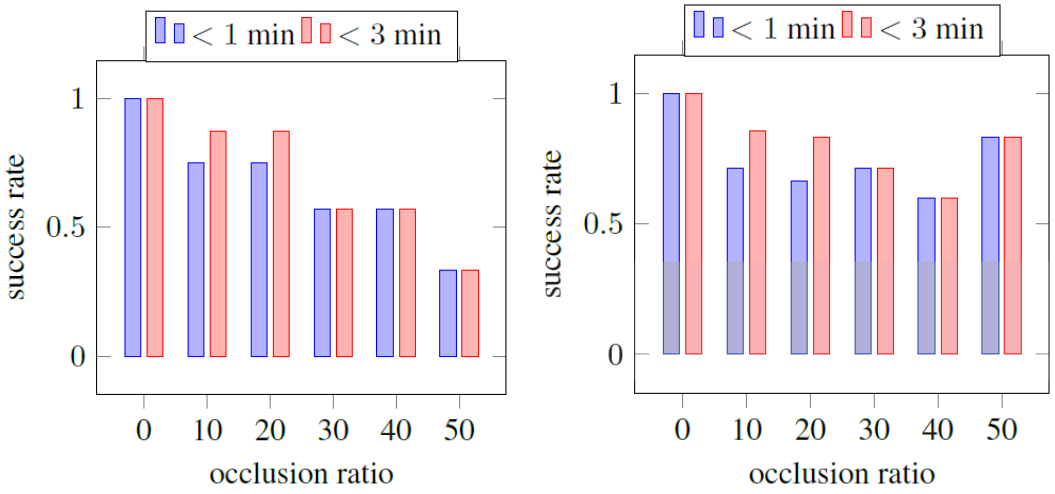
Figure 30. Success rate to plan in a corridor-like narrows space from start to goal within 5 s

The planning performance deteriorates significantly with larger search step, and also with larger initial bearing to goal *b*, as seen in Figure 30. Acceptable planning performance is achieved only with a $\#t_s$ set to 100 s for problems where the initial bearing to goal is less than 20°, and where the distance to goal is less than 50 km. A viable explanation is that a bigger search time step can lead to a frequent overstepping of the boundaries of the narrow operation area without a chance to turn around before. Having identified and proven the inadequacy of the planner for other problem settings, it is therefore essential to amend the planner for better planning performance, by developing a wrap-around framework to call the planner iteratively, as described in Section 3.3.3.2.

Performance Tests in the Presence of Obstacles

Another interesting test result with respect to the performance in the presence of obstacles is shown in Figure 31. The tests were performed in a wide operation area by fixing the number of obstacles (i.e. two or five obstacles) in each set of tests, whilst varying the obstacle occlusion ratio in the search space $occ = \Sigma_i A_{obs_i}/A_{\text{operation}}$, where A_{obs_i} is the surface area of obstacle i, and $A_{\text{operation}}$ is the surface area of the wide operation area. Therefore, for the same occlusion ratio, the smaller the number of obstacles is, the larger the obstacles are. Similar to the tests conducted in obstacle-free operation areas, empirical observations show that flight path plans in a wide operation area in the presence of obstacles are either obtained within a few tens of seconds up to a minute, or the planning time takes an unreasonable or eventually an infinite amount of time. The timeout for planning in the presence of obstacles in the tests is set to one minute; the success rate of finding a plan within the timeout is summarized in Figure 31.

The planning success rate reduces with increasing obstacle occlusion. However, in the case of only two obstacles, the success rate decreases more than in the case of five obstacles, mainly due to the size of the obstacles. The AIBR heuristics of the planner guides the search toward the goal. However, if a large obstacle happens to be in the way, it is harder for the planner to go round it, since the heuristic determination using AIBR does not consider the global constraints [Scala et al., 2016].



a. 2 obstacles

b. 5 obstacles

Figure 31. Performance of the planner with respect to obstacles occlusion ratio in the case of two and five obstacles respectively

Benchmarking with RRT from OMPL

While it is not the aim of this subsection to prove that ENHSP is better than any existing motion planners to solve a task and kinodynamic planning problem, the benchmarking performed is intended to put forth the usability, by showing comparable or reasonable performance with respect to another standard motion planner suitable for solving this class of problem: the control-based RRT from OMPL [Sucan et al., 2012]. This planner is chosen for its standardized API that allows the

modelling of a system be independent from the planner, as explained in Section 3.2.4, and can therefore be used off-the-shelf like ENHSP. Other motion planners, with some or much tuning, may also be suitable candidates to solve the HAPS flight path planning problem. However, as pointed out by Hooker in [Hooker, 1995] in the context of heuristics testing, yet applicable for other forms of benchmarking tests, the implementation of a method/solver also affects the fairness of the benchmarking, since the tuning of the benchmarking planner will not be properly refined, resulting hence in probably much worse performance.

Note that RRT in OMPL is implemented as an anytime planner, i.e. planning is performed until timeout and the plan found closest to the goal is provided. For the benchmarking tests, a planning attempt is considered successful only if the end position is within three search time steps to the goal, i.e.

$$\left\| p - p_{\text{goal}} \right\| = 3 \cdot t_s \cdot v_{\text{TAS}}, \tag{3-13}$$

where $\| \cdot \|_2$ denotes the Euclidean norm. Similar goal condition is also set in the PDDL+ problem instance file. The timeout is set to 5 s for both planners for tests without obstacles and 3 minutes in the presence of obstacles. As a reminder, the timeout for ENHSP is just a timely cut-off point to build the statistics of the planning success rate, while the actual planning time could be shorter or longer than the timeout duration.

The flight path planner is first tested without the presence of obstacle in three different settings, namely wide, fitting and narrow operation area. Wide and narrow operation areas bear the same configuration as described before (see Figure 28), while a fitting operation area consists of a quadrilateral area similar to the wide area in Figure 28a, but with $d_{\text{edge}_*} = [1.1 \cdot d_{\text{SG}}, 1.3 \cdot d_{\text{SG}}]$. For each type of operation area, the tests are grouped into three sets, as according to the start-goal distance, i.e. 0-50 km (‘]0,50]’), 50-100 km (‘]50,100]’) and 100-150 km (‘]100,150]’). The wind field in the operation area bears a magnitude of less than 5 m/s.

Again, for each test configuration, 20 settings were randomly generated, with the wind vector set randomly, and the exact start-goal distance chosen randomly within the given range. Four planner configurations are in used to test each setting:

1. control-based RRT from OMPL with a search time step of 1 s,
2. control-based RRT from OMPL with a search time step of 1 s,
3. ENHSP with a search time step $\#t_s$ of 30 s and a validation time step $\#t_v$ of 1 s;
4. ENHSP with a search time step $\#t_s$ of 150 s and a validation time step $\#t_v$ of 1 s.

The planning performance varies with different search steps. The performance of the planners is evaluated using the success rate of obtaining a plan within timeout (i.e. 5 s without obstacle and 3 min in the presence of obstacles), as summarized in Table 13.

Table 13. Success rate of obtaining a plan within 5 s timeout for point-to-point kinodynamic motion planning

a. without obstacles

Distance to goal [km]	**Wide area**			**Fitting area**		
	[0,50]	]50,100]	]100,150]	[0,50]	]50,100]	]100,150]
RRT $\#t_s$ = 30 s	0.02	0.01	0.01	0.88	0.65	0.23
RRT $\#t_s$ = 150 s	0.04	0.02	0.01	0.94	0.98	1
ENHSP $\#t_s$ = 30 s, $\#t_v$ = 10 s	0.73	0.15	0.02	0.78	0.25	0.11
ENHSP $\#t_s$ = 150 s, $\#t_v$ = 10 s	0.95	1	1	0.96	1	1

Distance to goal [km]	**Narrow area**		
	[0,50]	]50,100]	]100,150]
RRT $\#t_s$ = 30 s	0.92	0.89	0.85
RRT $\#t_s$ = 150 s	0.98	0.89	0.88
ENHSP $\#t_s$ = 30 s, $\#t_v$ = 10 s	0.65	0.52	0.48
ENHSP $\#t_s$ = 150 s, $\#t_v$ = 10 s	0.28	0.17	0.06

b. with obstacles

Distance to goal [km]	**Wide area, Distance to goal]100, 150]**	
	occ = [0,30]	*occ* =]30,50]
RRT $\#t_s$ = 150 s	0.04	0.05
ENHSP $\#t_s$ = 150 s, $\#t_v$ = 10 s	0.87	0.58

RRT performs comparably as ENHSP in the case of a fitting operation area, and even outperforms ENHSP in narrow operation areas. However, in a wide operation area, RRT suffers due to the lack of heuristic to guide the search towards the goal, resulting hence in the random search algorithm "getting lost" in the vast search space. ENHSP, on the other hand, does not require an additional definition of heuristic. Thanks to the integrated AIBR, a heuristic is generated for each set of problem, thus the search is guided towards the goal and a larger dimension of the operation area does not affect the search efficient much. Although there are reportedly many heuristics that could be defined for a more efficient biased search using [Urmson and

Simmons, 2003], they were tested mainly for grid-world path planning problems without the presence of a vector field.

Subsequent tests were performed in the presence of obstacles in a wide operation area, with an obstacle occlusion of 0-30% ($occ = [0; 30]$) or 30-50% ($occ =]30;\ 50]$). Only wide operation areas are tested, since due to safety purposes, a HAPS is allowed to fly in a narrow area only at the absence of obstacles, as there might not be enough space to fly around the obstacles (see MC1 of Table 8). The success rate for ENHSP drops, but remains at a satisfactory level, as shown in Table 13, especially for $occ = [0; 30]$, which is also the usual case in an operation, or the operation will be aborted due to high risk of collision.

It is also worth noting that, for the same search time step, the length of the plans in terms of travel time obtained using RRT or ENHSP during the tests is comparable, i.e. the plan quality is almost as good with both planners, without having a clear winner.

The benchmarking tests show that a kinodynamic flight path planning using ENHSP is not unreasonable, and even beneficial, especially in wide operation areas. In narrow corridor-like areas, however, regardless of the fact of RRT outperforms ENHSP, the usability of ENHSP in narrow areas must be improved. The next subsection shows an increase in planning efficiency using an implementation framework to call ENHSP in a "receding horizon" fashion.

3.3.3.2 Fine Tuning for More Performance

From the observed performance of the planner shown in Figure 29, the search step for planning within a MA or WA using ENHSP is set to 150 s if the start-goal distance is larger than 80 km and 100 s otherwise. An advantage of using ENHSP is that the search step and the validation step can be set separately. Therefore, even if the explored nodes are spaced quite far apart, the smaller obstacles between nodes will not be missed since the plan validation is performed with a smaller step.

Owing to the test results depicted in Figure 30, in the case where the search is to be performed within a narrow search space (e.g. a corridor-like operation area), the search step is set to 100 s. However, if the initial bearing to goal b is larger than 30°, the efficiency of ENHSP drops drastically, as shown in Figure 30. An iterative call to the planner can improve vastly the planning efficiency [Kiam et al., 2018]. The planner will be called iteratively to first reduce the bearing by imposing subgoals placed between the start and goal positions, so that the course heading of the HAPS approaches the heading of the HAPS-goal vector. Algorithm 3-2 describes how ENHSP is called iteratively.

The main purpose of Algorithm 3-2 is to relax the goal condition, by imposing at each call to ENHSP, that the bearing b be reduced, while approaching the goal. ENHSP at each call intends to plan a path from the HAPS initial position p_{init}. Before each call to ENHSP, the bearing b_{init} between the current course tangent and the vector between the initial and the goal position (see Line 5 and Line 20) is determined. If the magnitude of the bearing is larger than 20°, instead of setting the desired goal position as a goal condition for the planner, subgoals are imposed, which

consist of two conditions (Line 7-15 of Algorithm 3-2). The first one intends to approach the goal position. If the distance between the initial position and the goal position d_{IG} is larger than 50 km, the subgoal must be place 50 km from the initial position p_{init} (see Line 12); if d_{IG} is smaller than 50 km, it is sufficient that d_{IG} be reduced (see Line 10), resulting in the HAPS approaching the goal. The second subgoal condition is to reduce the bearing to goal by 20° at each call to the planner (see Line 15).

The problem file is parsed with the conditions to achieve the subgoals and ENHSP is called recursively and stops when the magnitude of the bearing is smaller than 20° (see Line 6). If the goal is still not reached (see Line 22), ENHSP will be called to compute a plan to the goal. The goal position is considered reached if the condition described by Equation 3-13 is fulfilled.

	Algorithm 3-2	Iterative planning with relaxed subgoals
	Require	HAPS start position vector p_{start}, goal position vector p_{goal}
1:		% assign initial position vector
2:		$p_{\mathrm{init}} = p_{\mathrm{start}}$
3:		% determine distance to goal
4:		$d = \left\|p_{\mathrm{goal}} - p_{\mathrm{init}}\right\|_2$
5:		determine bearing b_{init}, the angle difference between initial course heading and the heading of the vector connecting the initial and the goal position
6:		**while** $k = \left\lfloor \frac{b_{\mathrm{init}}}{20°} \right\rfloor > 1$ **do**
7:		set subgoal conditions to:
8:		**if** $\left\|p_{\mathrm{init}} - p_{\mathrm{goal}}\right\|_2 < 50$
9:		
10:		1)$\left\|p_{\mathrm{HAPS}} - p_{\mathrm{goal}}\right\|_2 < \left\|p_{\mathrm{init}} - p_{\mathrm{goal}}\right\|_2$
11:		**else**
12:		1)$\left\|p_{\mathrm{HAPS}} - p_{\mathrm{goal}}\right\|_2 < \left\|p_{\mathrm{init}} - p_{\mathrm{goal}}\right\|_2 - 50$ km
14:		**end if**
15:		2) $\|b\| < \|b_{\mathrm{init}}\| - 20°$
16:		parse problem instance and call ENHSP
17:		% assign the last HAPS position as the initial position vector
18:		$p_{\mathrm{init}} = p_{\mathrm{HAPS}}$
19:		$d = \left\|p_{\mathrm{goal}} - p_{\mathrm{init}}\right\|_2$
20:		determine initial bearing $b_{\mathrm{init}} = b$
21:		**end while**
22:		**if** $\left\|p_{\mathrm{HAPS}} - p_{\mathrm{goal}}\right\|_2 < 3 \cdot t_s \cdot v_{\mathrm{TAS}}$
23:		set goal condition to
24:		$\left\|p_{\mathrm{HAPS}} - p_{\mathrm{goal}}\right\|_2 < 3 \cdot t_s \cdot v_{\mathrm{TAS}}$
25:		**end if**

The framework is hence implemented in a receding horizon fashion, not exactly the receding horizon approach in control theory, but similar to other receding horizon

path planning methods, such as the Lazy Receding Horizon A* [Mandalika et al., 2018] implemented for a static world, in which the exact planning is performed only for the next upcoming steps, while "looking ahead" beyond these steps so that the agent approaches the goal. The lookahead in the recursive call to ENHSP is assured by the subgoal conditions listed in Line 7-1, forcing the HAPS to approach the goal at each iteration, while the plans from the start position are being piecewise computed by the planner.

3.3.4 Task Planning Problem

Since the mission requirements are time-dependent and the environment is time-varying, a numeric flight path planning is necessary to estimate the travel time and to avoid dynamic obstacles in the mission planning for HAPS. However, the mission planning problem described in Section 2.1.3 and depicted in Figure 10 and Figure 11, cannot be solved using solely a Single-Source Shortest-Path (SSSP) planner, as many mission-related requirements (MR) and constraints (MC) are expressed at a higher abstraction level by considering a MA or a LoI as a unit (see Table 8 and Table 9), i.e. the abstraction level for the physical space is higher, or rather the resolution of the space discretization is lower. To represent these MR and MC at a numeric level is either semantically challenging or algorithmically complex for the search. Many similar works rely on a task planner (to schedule at a higher level), loosely-coupled with a motion planner (to compute a more an exact, or executable plan) [Lima et al., 2018; Cashmore et al., 2015; Srivastava et al., 2014], in which PDDL planners are mostly used for task planning. One of the drawbacks of a loosely-coupled approach is the incoherence of the two planners, leading to probably frequent replannings in an environment with many constraints and dynamics.

Nevertheless, the previous subsections demonstrated the capability of a PDDL-based automated planner to solve a motion planning problem. It is hence reasonable to question if the task and motion planning problem can be solved in a tightly-coupled fashion by using a PDDL+ planner?

3.3.4.1 Can PDDL+ be Used for Task+Motion Planning?

PDDL is known for classical planning [McDermott, 2000; Fox and Long, 2003; 2006; Ghallab et al., 2004]. In PDDL+, a typical action to perform a task can be formulated for example as in Figure 31, in which a robot `?r` is commanded to perform a task `?t`. The effect of the action is to validate the predicate that the task `?t` is cleared `(cleared ?t)`. With the precondition `(not (cleared ?t))`. With this explicit precondition, a task cannot be repeatedly cleared. It is worth noting that by using the parameterization of the PDDL+ semantics, the formulation allows any robot `?r` to clear any task `?t`, thereby enabling the concise encoding of planning problems with multiple tasks and multiple robots.

In order to test the viability of a tightly-coupled task+motion planning using PDDL+ planners, a less complex mission scenario involving less mission constraints and requirements is used, as illustrated in Figure 33, yet conform with some aspects of the setting of the targeted mission scenario depicted in Figure 10 and Figure 11. In this scenario, multiples HAPS operate in a quadrilateral wide area encompassing Points of Interest (PoI, marked as yellow triangles) to monitor. It is assumed that no

wind and no obstacle is present. The aim is to determine the shortest plan in terms of mission duration to cover all the PoI. Furthermore, as a simplification of mission constraints, the mission duration is assumed unlimited, i.e. the HAPS can take as long as they need to cover all the PoI. The formulation of the flight dynamics and search space are similar to Table 11. Additional action definition and inter-HAPS collision avoidance must however be included. The formulation of these are provided in Table 14.

```
(:action clear-task
   :parameter (?r -robot ?t -task)
   :precondition (not (cleared ?t))
   :effect (and (cleared ?t)))
```

Figure 32. Formulation in PDDL+ of an action to carry out a task non-repeatedly

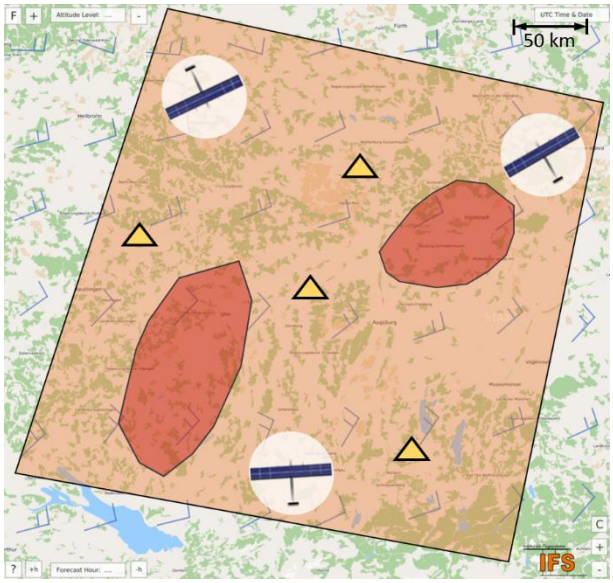


Figure 33. A typical airspace structure defined for repetitive monitoring tasks

Table 14. Formulation in PDDL+ for multiple HAPS and multiple tasks

Action to clear the monitoring task of a POI without repetition can be formulated similarly to Figure 31.

```
(:action clear-poi
   :parameters (?h -haps ?p -poi)
   :precondition (and (can-monitor ?h)
                      (not (cleared ?p)))
   :effect (and (cleared ?p))
```

Constraint to avoid collision between two different HAPS is drawn by imposing a minimum distance $\boldsymbol{\epsilon}$ between two non-identical HAPS:

$$\|\boldsymbol{p}_{\mathbf{HAPS1}} - \boldsymbol{p}_{\mathbf{HAPS2}}\|_2 < \boldsymbol{\epsilon}$$

```
  (:constraint collisions-haps
    :parameter (?h1 -haps ?h2 -haps)
    :precondition
      (and (not (different ?h1 ?h2))
           (> (^ (+ (^ (- (longitude ?h1) (longitude ?r2))
2)
                    (^ (- (latitude ?h1) (latitude ?h2)) 2)
0.5)
               epsilon))
```

The precondition tests if `?h1` and `?h2` are the same HAPS. The predicate of two different HAPS is encoded in the problem instance file for every pair of initiated non-identical HAPS.

Goal condition imposes that all PoIs be covered. The following shows the goal condition to be formulated in the case where three PoI are involved.

The goal condition(s) is encoded in the problem instance file. The following is an example goal condition to clear three PoI.

```
  (:goal
     (and
         (cleared p1)
         (cleared p2)
         (cleared p3)))
```

Table 14 shows that the formulation of scheduling for multiple tasks and multiple HAPS in PDDL+ is possible, therefore a tightly coupled task+motion planning is also potentially feasible. However, when more than one HAPS or one PoI to clear is involved, ENHSP has relatively low success rate. The tests on the planning efficient of ENHSP for multi-HAPS, multi-PoI were conducted based on 20 randomly generated problem instances, i.e. randomly placed PoI and initial positions of the HAPS, as well as randomly generated (wide or narrow) operation areas. ENHSP was configured to use the AIBR heuristic [Scala et al., 2016], as in the previous tests. Empirically observed, if planning is successful, a plan is either found within a minute, except for a few outliers with a few minutes of planning time. Therefore, a timeout of 15 minutes was set for the tests. Figure 34 depicts the success rate to plan using ENHSP.

It is interesting to note that when two PoI are involved, the success rate with only one HAPS is much lower than with two. A viable explanation is that the heuristic guides the HAPS to different PoI, resulting in a never-ending plan search if the PoI are placed far apart. With three PoI, the success rate suffers even more. The test results proved that, although the encoding of the model is possible, and that AI-planners can potentially be used to solve task and kinodynamic motion planning problem in a tightly-coupled manner, it is (almost) impossible to scale up, not even with the state-of-the-art PDDL+ planners, on more tasks (i.e. PoI to clear) or more agents (i.e. more HAPS), not even for this simple scenario.

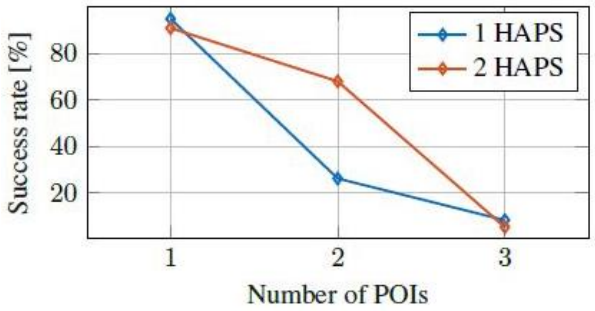


Figure 34. Success rate with respect to the number of POIs (number of tasks)

[Gaschler, 2016] developed a set of predicates for symbolic-geometric mapping, the goal of which is to call an external function or procedure to perform some geometric planning nested within a symbolic planner like PDDL. Although it helps to curb some shortages in symbolic planners, e.g. geometric formulations, computation complexity, etc., its structure still remains loosely coupled and is quite rigid to be adapted for tightly-coupled task+motion planning. Furthermore, the geometric predicates are quite limited and not tested for temporal symbolic planners. It is hence advisable to proceed on coupling the numeric flight path planner with an additional task planner like in [Siddharth Srivastava et al., 2014] to solve the mission planning problems depicted in Figure 10 and Figure 11, which is the core the works described in the following chapters.

"Different problems require different tools and techniques."
- John Horgan cited Rolf Landauer in *The End of Science*

4 Hierarchical Task Planning for HAPS

Often, planning is not performed at one go, but in several modules in order to achieve better performance in terms of planning time or for better results, like [Schmitt and Schulte, 2016], in which a classical planner is used for performing combinatorial search, while a constraint-optimization solver is used for fast linear optimization. As pointed out at the end of the previous chapter, a tightly-coupled task+motion planning using PDDL+ compatible AI planner does unfortunately not scale for more complex problems involving multiple tasks or multiple HAPS, even though the PDDL+ supports the formulation of a task+motion planning problem. It is hence reasonable to proceed using a loosely-coupled hybrid[22] approach to perform the task and the motion planning in separate modules. Planning for autonomous driving in urban traffic in [Srivastava et al., 2014] for example, also uses a classical two-tier approach, involving first the strategic planning level for optimal routing, followed by a tactical planning level that decides for how the car drives along the routes.

In this section, the two-tier planning approach for HAPS is described: a task planning method developed for the strategic planning level, together with its coupling with the numeric flight path planner described in the previous chapter at the tactical level. The *task planning for a single HAPS is considered here. The extension to multiple HAPS will be tackled in the next chapter.*

At the strategic level, the task planner reasons with meta-actions over the plan horizon within which the knowledge of the environment is (partially) known. Meta-actions are expressed in form of predicates to describe an action of higher abstraction levels, and can be decomposed into lower-level numeric actions during the transition from strategic to tactical planning. Such a decomposition can be done even multiple times, as inspired from human cognition. Often, human beings tend to solve a complicated problem with a top-down manner. For example, to clean a skyscraper, one thinks of which floor to start with, subsequently the order of the rooms, followed by the cleaning tasks in each room. Doing so helps to reduce the abstraction space down the hierarchy, thereby reducing the complexity of the problem. Additionally, it

[22] Note that "hybrid" here does no long bear the definition of a hybrid planner given in Section 3.3 for the context of domain-independent AI planning.

also eases the representation as well as the comprehension of constraints of the planning problem down the abstraction levels, i.e. a room is occupied at certain hours, there is no available power socket in a room, some tools are not allowed/ available in certain rooms or at certain storeys, the ability and availability of the cleaning staff etc. Finally, an exact timetable can be drawn to assign tasks to each worker.

Also widely used for AI planning are the Hierarchical Task Network (HTN) planners, e.g. SHOP2 [Nau et al., 2003], which is among the firsts to formalize a HTN, Markov-HTN [Chen et al., 2009] for web services such as booking a flight or a hotel room, and many more [Benton et al., 2018; Georgievski and Aiello, 2014; Fdez-Olivares et al., 2006; Sirin et al., 2004]. These planners rely on the decomposition of higher-level tasks into lower-level tasks and eventually primitive tasks, forming hence a hierarchical task network. They were above all developed for solving planning problems of domains in which many routine and protocols apply, since these can be conveniently encoded in the HTN. In aeronautics, multiple strict regulations are involved; therefore, mission execution often must comply with standard protocols, which can be understood as domain-specific knowledge in planning. HTN planners are found to be convenient for such applications, since the encoded protocols in the HTN impose how a high-level abstract task can be carried out. [Benton et al., 2018] for example uses a HTN-based planner to assist a pilot in plan execution. The assistant plays the role of a checklist in the system to ensure that the pilot follows all underwritten protocols while being non-invasive. Apart from being able to encode into the network how a high-level abstract task can be carried out, the decomposition of a higher-level meta-action to a lower-level one also reduces the search space down the hierarchy, which in return renders the computation of a plans/ plans to perform an action less complex. The encoding regulations and the reduction of the search space constitute the biggest motivations for the use of a HTN planner in this work.

4.1 Strategic and Tactical Planning for HAPS

For the pre-execution planning for HAPS, in order to compensate the inadequacy of a numeric planner to plan for multiple-task (see Section 3.3.4.1), a framework consisting of a hierarchical task planner at the strategic planning level and a numeric flight path planner at the tactical planning level is proposed. The general solution framework is shown in Figure 35. At the strategic level, a HTN-based task planner is employed, in order to exploit the advantage of encoding the routine execution of a task and also to consider the mission constraints and requirements expressed at different abstraction levels.

The HTN-based task planner is “temporal” (see [Castillo et al., 2006] for a more formal description of a temporal HTN planner); it decomposes a monitoring task into lower-level tasks, while estimating the duration of each task. The encoding of the HTN, as well as the decomposition method will be described in Section 4.3. Although the task plans found are time-stamped, i.e. the duration of each task is stated, but these durations are estimated very imprecisely, since the platform dynamics is considered linear, and the wind is only considered probabilistically. Detailed description will be provided in Section 4.3.

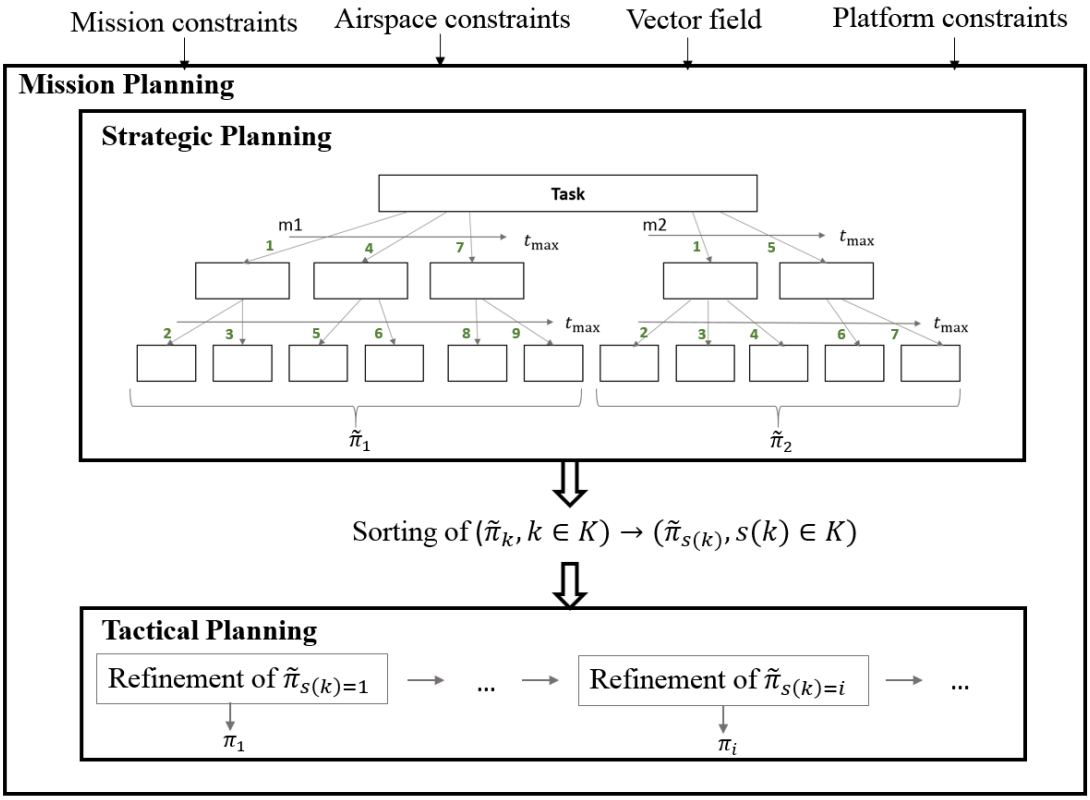


Figure 35. General architecture of the HAPS planning framework

Note that the HTN does not allow "dawdling", i.e. the agent, or more specifically the HAPS, does not have a time gap between tasks of the task plan. Even "loitering at the WA" is considered a task and must be assigned to the HAPS.

As shown in Figure 36, the difference in travel time as predicted by the strategic task planner and the tactical numeric planner tends to increase with the average wind magnitude of the operation area. The task planner at the strategic level assumes a linear movement model (i.e. linear motion with constant velocity) for the HAPS, which assumes that the derivative of the position of the HAPS with respect to time is constant, while the tactical numeric planner uses the kinematic model described in Equations 3-4 to 3-6, as described in detail in Chapter 3. Shown in Figure 36 is the average difference in travel time for a start-goal distance of 80 km predicted using with the movement model assumed at the strategic planning level (with a v^*_{TAS} of 28 m/s) and the kinematic model at the tactical level for 20 randomly generated wind vectors of the given wind magnitudes in the x-axis. The difference tends to increase with the wind magnitude and is due to the linear movement model of the HAPS assumed by the task planner, which also ignores the presence of wind. The overly simplified platform dynamics could result in:

1. low quality plans, since the tasks might not be rewarded accordingly due to the time dependency of the mission requirements (see Table 9);
2. non-executable plans, because some mission constraints from Table 8 can be violated, since the mission environment is time-variable.

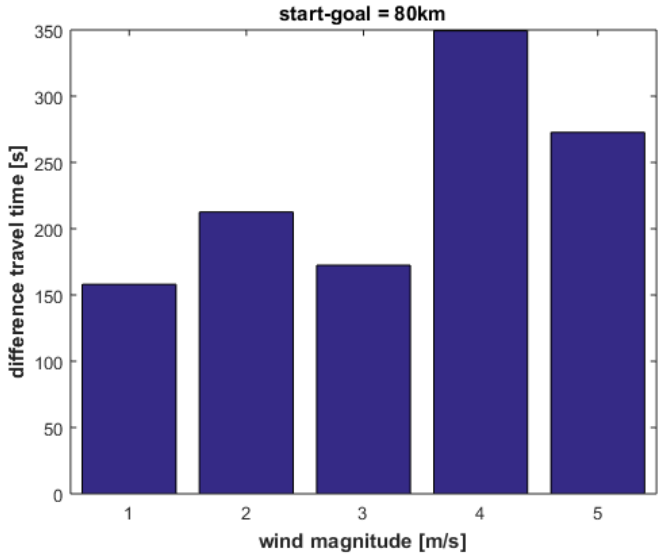


Figure 36. The average difference between the linearly predicted travel time using a constant derivative of the position with respect to time, and the feasible path found using ENHSP that considers the forecasted wind grid, platform dynamics and obstacles in the airspace.

The inadequacy of the task planner specifically in terms of guaranteeing feasibility is compensated by the integration of a numeric planner at the tactical level. As illustrated in Figure 35, the plans $\tilde{\pi}_k, k \in K$ found by the task planner consist of primitive tasks of the HTN. Each plan however bears its objective value, with the help of which $\tilde{\pi}_k$ will be sorted. The sorted index of the plans is represented by $\tilde{\pi}_{s(k)}$. Subsequently, the plans $\tilde{\pi}_{s(k)}$ from the strategic level will be "refined" in the sorted order $s(k)$ by the numeric flight path planner described in Section 3. The plans $\tilde{\pi}_{s(k)}$ are "refined" with a better estimation of the task duration, since the flight dynamics and the time-varying environment (i.e. wind field, static and dynamic obstacle avoidance etc.) are taken into account in the numeric flight path planner. The refined objective value/rewards will be used to re-sort the plans, thus obtaining π_i. Either the best plan will be executed or the ordering the of the plans serves as a decision support in a human-in-the-loop mission planning system, as discussed in Section 2.2.4 and later again in Section 7.1.1.

For the sake of clarity, a plan determined by the task planner is referred to as a *task* plan and is represented by $\tilde{\pi}$, while a plan computed by the numeric flight path planner is simply a plan and is represented symbolically by π without the tilde. The re-sorting of plans by the tactical planning is a better judgement for plan quality, since the plan parameters considered are more refined, or rather of higher spatial resolution than in the strategic planning level. Note however that the first sorting of plans by the strategic planner is still necessary, especially when planning time is limited, so that seemingly good plans stand a better chance to be "refined" by the tactical planner.

4.2 HTN for the HAPS Task Planner

Most HTN planners only decide the sequence of tasks to execute. The formalism of a HTN provided in [Nau et al., 2003] leaves out the temporal component. Note however that the formalism of a HTN is different from the PDDL formalism seen in

Section 3. As mentioned in [Haslum et al., 2019], PDDL is not suitable for encoding a HTN, since the language lacks expressions for the hierarchical relations.

The HTN planner in SIADEX [Fdez-Olivares et al., 2006] is the first to include the temporal element formally, and is also being referred to as the *temporal HTN*. SIADEX is a planning framework developed to plan dynamically for a forest fire fighting, in which case time and duration of actions are important, since fire spreads over time. The planner also allows concurrency (i.e. many tasks can be executed at the same time) and time-dependent goals. The handling of the temporal knowledge in the HTN planner used in SIADEX was documented in [Castillo et al., 2006].

Some definitions of the temporal HTN used for HAPS task planning are provided next. A significant difference between the handling of the temporal HTN used in this work and that in SIADEX lies with the evaluation of the plans, which is important for HAPS, while being neglected in SIADEX. In the HAPS hierarchical task planner, many possible sequences of subtasks can be obtained by applying a function to decompose a non-primitive task[23]. It is hence important to evaluate which method is more promising.

In the following, some formal definitions for the HAPS temporal HTN adapted from the classical HTN definition in [Nau et al., 2003; Höller et al., 2018] and from the temporal HTN in [Fdez-Olivares et al., 2006] are provided. The symbols used might not conform with the list of symbols, and are specific only to the definitions given below in order to understand the "formalism" of a temporal HTN.

Definition 4-1 (Temporal Hierarchical Planning Problem) A temporal hierarchical planning problem with finite temporal horizon T can be represented by a tuple $P = (x_0, s_0, t_0, O, O_p, M, g, S_{\delta_t})$, where x_0, s_0 and t_0 are the initial states (continuous and discrete respectively) and initial time instant, O is the set of task names. If $o \in O$, o can be expressed as a sequence of primitive actions $\left(o_{p_1}, \dots, o_{p_n}\right) \in O_p^n$. Note that $O_p \subset O$. $M$is the set of decomposition methods, g is a set of goal conditions and S_{δ_t} is the set of durations$\delta_t\left(o_{p_n}, o_{p_{n-1}}\right)$ of the primitive tasks o_{p_n}, which may depend on the corresponding previous task $o_{p_{n-1}}$.

Definition 4-2 (Method) A method $m \in M$ is represented by a 3-tuple $< o(t, \delta_t)$, subtask($o(t, \delta_t)$) , $d >$, where $o \in O \backslash O_p$ must be a non-primitive task and subtask($o(t, \delta_t)$) is the set of sequences of tasks obtained by applying the decomposition function d to o that starts at time t and takes a duration of δ_t. The decomposition function d can be a combination of tasks, a permutation of tasks or a fix order of task.

Definition 4-3 (Solution) A solution to the temporal hierarchical planning problem P over a plan horizon of duration T is a task plan $\tilde{\pi} = \langle o_{p_1}(t_1, \delta_{t_1}), \dots, o_{p_n}(t_n, \delta_{t_n})\rangle$, a sequence of primitive tasks o_{p_k} that take a duration of δ_{t_k}, ordered with respect to the task execution time $t_k < T$. The constraints are satisfied and for all $k \in \{1, \dots, n\}$,

[23] Only total-ordered tasks are considered in this work.

and the tasks must all terminate before the plan horizon ends, i.e. if $t_0 = 0$, then $\max_{k\in\{1,\ldots,n\}} t_k + \delta_{t_k} < T$.

The hierarchy of the task planner in the strategic planner as illustrated in Figure 35 can be formed concretely for HAPS by considering the spatial resolution, thereby allowing:

1. finer details to be successively considered down the hierarchy in a smaller and more isolated abstraction space (e.g. first the sequence of MAs which will be decomposed into sequences of LOIs and subsequently into sequences of Points-of-Interests (PoIs) and the actions to be executed at the PoIs. The tasks of the last level are the primitive tasks; the positions of consecutive PoIs will be taken as start and goal position by the numeric flight path planner at the tactical planning level.
2. different decomposition functions at each level to determine the task order, either via a combination, or a permutation of a known protocol derived from the domain knowledge.

The HTN for HAPS task planning will be described in the following sections.

4.3 Hierarchical Task Network for HAPS

Figure 37 shows graphically a decomposition of a monitoring task over $[T_{\text{start}}, T_{\text{end}}]$ to obtain a task plan. The decomposition works successively towards a more reduced abstraction space, from MA (MA#) to LOI ($\text{monitor}_{\text{LOI\#}}$) and subsequently to waypoints at which concurrent tasks of managing the payload and of reporting to/communicating with the GCS are also detailed, such as report flying to corridor C#, fly to corridor C# ($\text{to}_{\text{C\#}}$), cross corridor ($\text{cross}_{\text{C\#}}$), fly to the nearest vertex of the LoI (NPL), scan LoI (scan), send images to ground, turn on/off mission camera etc. Note that the start time t_0 of the first task is not necessary T_{start}. t_0 is the time instant at which the HAPS has completed the last task of its previous plan.

A task plan for a HAPS is per definition a sequence of n time-stamped primitive tasks of the HTN $\tilde{\pi} = \langle o_{p_1}(t_0, \delta_{t_1}), \ldots, o_{p_n}(t_{n-1}, \delta_{t_n})\rangle$, along with the durations of the tasks δ_{t_i}. However, for the ease of representation, a task plan can also be expressed with higher-level tasks; for example, a task plan at the MA-level $\tilde{\pi}^{\text{MA}}$ is the sequence of time-stamped tasks of this level, e.g. <MA5 $(t_0^{\text{MA}}, \delta_{t_1}^{\text{MA}})$, MA4$(t_1^{\text{MA}}, \delta_{t_2}^{\text{MA}})$, WA13$(t_2^{\text{MA}}, \delta_{t_3}^{\text{MA}})$> and the corresponding plan at the LOI-level can be represented by a sequence of time-stamped tasks of this level, for example for the hierarchical plan shown in Figure 37, $\tilde{\pi}^{\text{LOI}} = <$ $\text{to}_{\text{C2}}(t_0^{\text{LOI}}, \delta_{t_1}^{\text{LOI}}), \text{cross}_{\text{C2}}(t_1^{\text{LOI}}, \delta_{t_2}^{\text{LOI}}), \text{to}_{\text{LOI1}}(t_2^{\text{LOI}}, \delta_{t_3}^{\text{LOI}}), \ldots, \text{cross}_{\text{C5}}(t_9^{\text{LOI}}, \delta_{t_{10}}^{\text{LOI}}) >$.

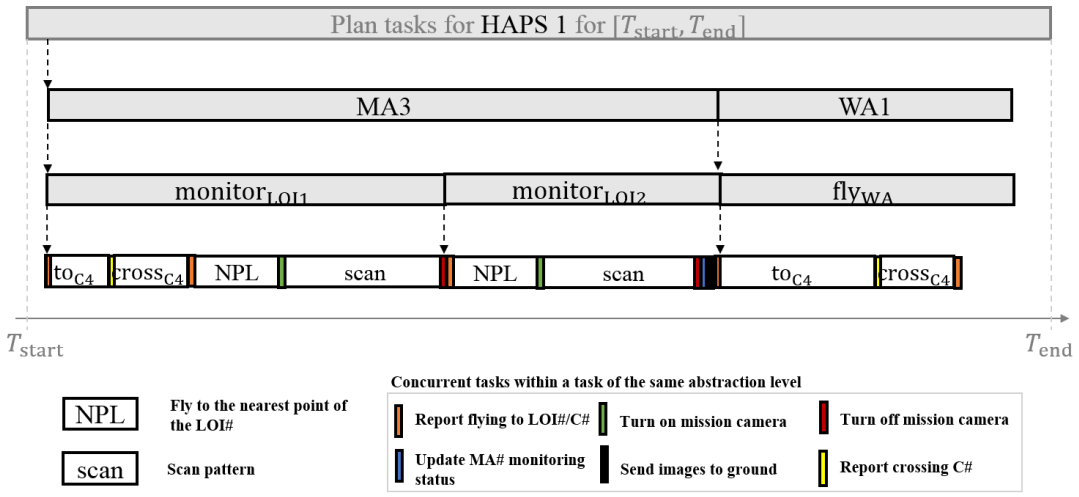


Figure 37. Temporal hierarchical plan example for one HAPS

For the rest of this work, the temporal notation $o(t_{i-1}, \delta_i)$ of task o that commences at time t_{i-1} and takes a duration of δ_i can be used interchangeably with $o(t_i, \delta_i)$, where $t_i = t_{i-1} + \delta_i$ is the completion time of the task. As a rule of thumb, if the subscript of t is identical with the subscript of the duration, the time represents the completion time, while a decremented subscript of t signifies the commencement time of the task.

Table 15 summarizes continuous tasks of various levels in the HTN planner at the strategic level, along with the enumerated mission constraints listed out in Table 8. The different MC each task is subject to highlight the advantage of using a HTN so that these constraints can be expressed at different abstraction levels. Since "no-dawdling" is assumed, at each time instant, one of the continuous tasks is performed. Furthermore, continuous tasks cannot be performed concurrently. The mini colored bars in Figure 37 are instantaneous tasks which can take place concurrently as the concurrent tasks.

Table 15. Tasks at higher abstraction levels to be carried out for HAPS mission

Task description	Task names	Mission constraints
Fly in WA#	WA#	MC1, MC2, MC5
Monitor MA#	MA#	MC1, MC2, MC3, MC4, MC5
Monitor LOI#	$monitor_{LOI\#}$	MC6
Fly to corridor C#	$to_{C\#}$	MC1, MC2
Cross corridor C#	$cross_{C\#}$	MC1, MC2
Fly to the nearest vertex of LOI#	NPL	MC6
Scan LOI#	scan	MC6

4.3.1 Task Decomposition

Like most HTN planners, a top-down, forward decomposing strategy is used. Graphically, the decomposition order of a high-level task to lower-level tasks is as

shown in the numbered order in Figure 35. Concretely, in the task planning for HAPS, the monitoring task "Plan tasks for HAPS 1 over plan horizon $[T_{\text{start}}, T_{\text{end}}]$" can be decomposed via combination of subtasks. Not knowing how many subtasks should constitute the combination of subtasks, a first subtask is assigned to the combination, and is further decomposed until all tasks stemming from it becomes primitive tasks, before the subsequent subtask is assigned and decomposed to primitive tasks. The process is sequentially repeated until the plan horizon T_{end} is reached. By numbering the order of decisions made during the search of a task plan in the HTN, Figure 38 illustrates how the decomposition is performed using part of the mission scenario depicted in Figure 11 for one HAPS. Encircled in orange is the decision made at the MA-level, in red at the LoI-level, while decisions at the PoI level are marked with an "x". Since at the MA-level, the decomposition consists of a combination of subtasks, the HTN task planner does not perform a complete decomposition, but decide only for the first MA the HAPS will visit (see number 1 marked at MA1). Subsequently, the order of the LoI are decided (see number 2 and 3 in the LoI of MA1), followed by the orders of PoI, which consist either of the mid-point of the intersection between a MA and a corridor (i.e. number 4 and 5), or vertices of the LoI for the monitoring pattern (i.e. number 6, 7, 8, and 9). The process is repeated for the next mission element WA15, followed by MA9. Note that the decomposition into subtasks of the LoI- and PoI-levels are performed entirely, since the decompositions consist either of permutations of LoIs or of a fix routine of setting of PoIs.

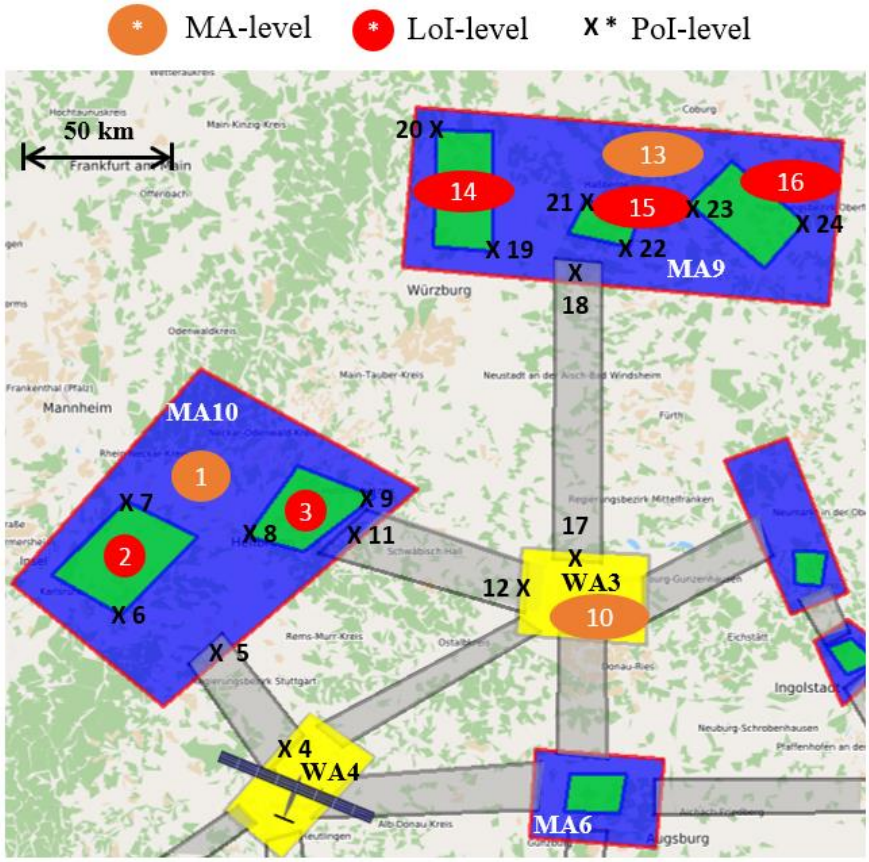


Figure 38. Top-down, forward task decomposition order

As briefly demonstrated in the illustrative example, the decomposition of a non-primitive task can be performed either via a fix-order decomposition function, a permutation decomposition function or a combinatorial decomposition function. The decomposition to subtasks at MA-level is combinatorial, since the planner chooses one of the numerous available mission elements to visit each time it has to decide for the next task at the MA-level. The decomposition to subtasks at the LoI-level consists of permutations, since all LoI of the MA must be monitored in order for the mission to be rewarded in case of success (see MR1 of Table 9). Lastly, the decomposition at the PoI-level follows a set of fix rules: the HAPS must communicate with the GCS before crossing a corridor (see number 4 of Figure 38), check again while leaving the corridor (see number 5), fly to the nearest vertex of an LoI and turn on the mission camera (see number 6), perform a lawnmower sweep pattern to record images of the LoI. To simplify the graphical illustration, the sweep pattern is not shown in detailed; but rather, a PoI is places at the diagonal vertex to mark the end of the sweep (see number 7).

Algorithm 4-1 to 4-3 describe in a more general manner how a non-primitive task is decomposed in a temporal HTN to subtasks per fixed order, per permutation and per combination respectively, which are the three kinds of decomposition used in the HTN planner at the strategic level of the mission planner for HAPS.

Algorithm 4-1 describes the decomposition of a non-primitive task o using a *decomposition-per-fixed-order function* d_f. It first lists out all the possible sequences of subtasks, as according to task execution protocols (see Line 3). Note that the durations of the subtasks at this level are not computed yet, unless if the subtasks are already the primitive tasks. Sequences that violate the mission constraints and can already be identified at this stage (without knowing the durations of the subtasks) will be eliminated. The check for constraint violation for the first task in the decomposed subtask sequence requires the subtask network of the previous task. For each sequence in subtask($o(t)$), the subtasks are further decomposed (in the order of the task in the sequence) with the imposed decomposition function (see Line 7-10). When no further decomposition is possible, meaning all tasks are primitive, the durations of the task $o(t)$ are then estimated, by summing of the durations of the subtasks stemming from it. Note that $o(t)$ can have many possible durations since there can be more than one subtask networks stemming from it.

Algorithm 4-1	Decomposing a non-primitive task $o_n(t_n)$ with the *decomposition-per-fixed-order* function d_f
Require	the task network of the previous task, i.e. subtask($o_{n-1}(t_{n-1}), \delta_{t_{n-1}}$), and the current task to decompose $o(t)$
1:	**if** $o_n(t_n)$ is not a primitive task
2:	find all sequences of subtasks emerging from $o_n(t_n)$ and group into the set subtask($o_n(t_n)$)
3:	eliminate sequences from the set subtask($o_n(t_n)$) that are identified to violate the constraints at the subtask level
4:	**end if**
5:	**for** each sequence of subtask($o_n(t_n)$)
6:	decompose each task to its primitive tasks with the assigned decomposition function
7:	determine the duration of each δ_{t_n} sequence
8:	eliminate sequences that violate constraints
9:	**end for**

Decomposing a non-primitive task *o* with the *decomposition-per-permutation* function d_p is performed in a similar manner, as described in Algorithm 4-2. A permutation decomposition is performed when the subtasks of *o* are known and are presented in the form of a subtasks array with the necessary repeated occurrence. However, the order of the tasks is unknown and shall be decided by d_p.

Algorithm 4-2	Decomposing a non-primitive task $o_n(t_n)$ with the *decomposition-per-permutation* function d_p
Require	the task network of the previous task, i.e. subtask($o_{n-1}(t_{n-1}), \delta_{t_{n-1}}$), and the current task to decompose $o_n(t_n)$
1:	**if** $o_n(t_n)$ is not a primitive task
2:	find the subtasks emerging from $o_n(t_n)$ and order them into sequences by permutation to be grouped into the set subtask($o_n(t_n)$)
3:	eliminate sequences from subtask($o_n(t_n)$) that are identified to violate the constraints at the subtask level
4:	**end if**
5:	**for** each sequence of subtask($o_n(t_n)$)
6:	decompose each task to its primitive tasks with the assigned decomposition function
7:	determine the duration of each δ_{t_n} sequence
8:	eliminate sequences that violate constraints
9:	**end for**

A more different decomposition approach is the *decomposition-per-combination* function d_c. The non-primitive can be decomposed to a number of subtasks chosen from a list of subtasks. No repetition is allowed in the list. The size of the subtask-network is decided either by the number of subtasks or by the horizon T of the task o. It is worth noting that the latter is considered in this work at the decomposition of

the highest-level task to sequences of MA to visit. A d_c is performed in a top-down forward manner. The first subtask o_i^{l-1} that fulfills all constraints, where $i = 1$ and $l - 1$ denotes an abstraction level lower, is selected from the list of subtasks (see Line 4 in Algorithm 4-3). If the size of the subtask-network of o is decided by the length of the subtasks, then all possible combination of subtasks from the list will be determined, and the rest of the decomposition will then follow the same approach as described in Line 5-9 of Algorithm 4-1 and Algorithm 4-2. If the size of the subtask-network of o is limited by the horizon T, then Algorithm 4-3 applies, which is the method used for the decomposition into sequences of MA. The selected o_i^{l-1} is then further decomposed until all tasks stemming from it are decomposed into primitive tasks (see Line 6-12 of Algorithm 4-3). The duration of o_i^{l-1} will be computed based on the sum of the durations of the primitive tasks. A o_{i+1}^{l-1} will be juxtaposed to o_i^{l-1}, if the sum of durations of all $l - 1$ level subtasks stemming from o does not exceed T and the process to decompose o_{i+1}^{l-1} to primitive tasks will be successively performed (see Line 7-12). If the execution of the last subtask exceeds the given plan horizon T, the subtask will be truncated.

Algorithm 4-3	Decomposing a non-primitive task $\boldsymbol{o_n(t_n)}$ with the *decomposition-per-combination* function $\boldsymbol{d_c}$ over a task horizon $\boldsymbol{T}$
Require	the task network of the previous task, i. e. subtask$(o_{n-1}(t_{n-1}), \delta_{t_{n-1}})$, the current task to decompose $o_n(t_n)$, the duration of the task horizon T
1:	**if** $o(t_n)$ is not a primitive task
2:	find all the first tasks of the level below o_1^{l-1} that can be executed at/ after t_n
3:	**for** each o_1^{l-1}
4:	decompose o_1^{l-1} into primitive tasks with the decomposition associated to o_1^{l-1} to obtain subtask$(o_1^{l-1}(t_1^{l-1}), \delta_{t_1^{l-1}})$
5:	$i \coloneqq 1$
6:	**while** $t_i^{l-1} + \delta_{t_i^{l-1}} < t_n + T$
7:	find all the $(i+1)$-th task of the level below o_{i+1}^{l-1} that can be executed at/ after $t_i^{l-1} + \delta_{t_i^{l-1}}$
8:	**for** each o_{i+1}^{l-1}
9:	decompose o_i^{l-1} into primitive tasks with the given decomposition function to obtain subtask$(o_i^{l-1}(t_i^{l-1}), \delta_{t_i^{l-1}})$
10:	$i \coloneqq i + 1$
11:	**end for**
12:	**end while**
13:	**end for**
14:	**end if**

It is worth noting that if a *decomposition-per-combination* is nested in another *decomposition-per-combination*, for example if the decomposition of o_1^{l-1} in Line 5 is again dictated by a *decomposition-per-combination*, then the nested d_c stops when

1. the imposed length of the sequence of subtasks at level $l - 2$ is reached, or
2. the maximum duration of o_1^{l-1} is exceeded.

The decomposition of a task can be made up of a sequence of decomposition functions, e.g. the first few subtasks of a non-primitive task can be decomposed via a d_f, with the subsequent subtasks being decomposed via d_p. However, the joint decomposition function will not be dealt with in this work, since it is not required for the HTN in HAPS task planner.

Instead of performing a greedy search which could result in infeasible solutions, the hierarchical task planner naturally limits the plans to those feasible, given that a subtask network that violates the constraints will be eliminated (Line 5 and 9 of Algorithm 4-1 and 4-2), or that the compliance with the constraints are already considered during the decomposition (see Line 2 and 7 of Algorithm 4-3).

4.3.2 Estimation of the Duration of a Task

Considered in the task planning are only the mission constraints related to the airspace structure as listed in Table 8, i.e. MC4 and MC5 that checks if consecutive mission elements are connected by a corridor, which is a constraint at the MA# level, as well as MC6 . The duration of a task in the case for HAPS is conditioned by the dynamics of the platform, or at a less precise level, the optimal cruise speed of the platform. The other tasks are concurrent to the flying actions, and therefore do not influence the task durations.

The duration of a non-primitive task is estimated by summing the durations of all the primitive tasks stemming from it, since it is assumed in the HTN that no dawdling is involved (see Section 4.1). A primitive task o_p that flies the HAPS from a PoI with the position vector $p_{o_p(i)}$, and to another with the position vector $p_{o_p(i+1)}$ has a duration that equals the travel time δ_{o_p} between both which is estimated linearly, i.e. with the assumption of a linear trajectory travelled at constant optimal TAS, v^*_{TAS}, using the following:

$$\delta_{o_p} = \frac{d_{o_p}}{v^*_{\mathrm{TAS}}}, \tag{4-1}$$

where

$$d_{o_p} = \left\| p_{o_p(i+1)} - p_{o_p(i)} \right\|_2 \tag{4-2}$$

is the distance travelled while executing the primitive task o_p.

The fact that wind is not considered in the above estimation of travel times between PoI results in a non-negligible accumulative error in the estimation of the time of arrival, as shown in Figure 36, especially when the plan spans long period of time, which in the case of HAPS is usually hours, if not days.

The duration of a task in HTN can be relaxed[24], since some complex numeric constraints are not considered, for example the wind vector, the HAPS non-linear

[24] "Relaxed" is the term often used in AI planning literature to represent the numeric parameter of a state using a numeric range.

dynamics, obstacle avoidance etc. In the relaxed estimation, the estimation of a task is represented by a time interval $[\delta_{t,\min}, \delta_{t,\max}]$.

Considering the time-varying wind field already at the strategic planning level defies the purpose of having a two-tier planning architecture (see Figure 35) that is meant to reduce the search space of the complete mission planning problem down the hierarchy in the strategic level to a state space that the numeric planner at the tactical level can cope with. However, the wind can be very roughly considered at the strategic planning level, and subsequently be treated numerically at the tactical planning level. By considering that the wind has a maximum magnitude of $|v_{\mathrm{w}}|_{\mathrm{MAX}}$, which is the case since the HAPS is only allowed to fly during operation in areas with $|v_{\mathrm{w}}| < 5$ m/s (see Table 8), it is hence assumed that a HAPS has a ground speed between $[v^{\min}, v^{\max}] = [v_{\mathrm{TAS}} - |v_w|_{\mathrm{MAX}}, v_{\mathrm{TAS}} + |v_w|_{\mathrm{MAX}}]$. The duration to execute the primitive task o_p is identical to the travel time, which can be modelled probabilistically as a uniform distribution over $[d_{o_p}/v^{\max}, d_{o_p}/v^{\min}] = [\delta_{o_p}^{\min}, \delta_{o_p}^{\max}]$.

4.4 Combinatorial Problem in Selecting the Best Decomposition(s)

The decomposition at the MA-level in Figure 37 involves a decomposition-per-combination, while at the LOI-level, it involves a decomposition-per-permutation. Especially at the MA-level, the combinatorial problem can be complex, with a complexity that grows exponentially with respect to the number of mission elements to visit within the plan duration T, i.e. $n_{\mathrm{ME}}^{n_{o_{\mathrm{ME}}}}$, where n_{ME} is the total number of mission elements (MA or WA shown in Figure 10 and Figure 11) and n_o is the number of mission elements that the HAPS manage to visit within T. Since only a single HAPS is considered in this chapter, tests and methods developed here consider the mission scenario depicted in Figure 10. The extension to multiple HAPS and a larger coverage with more mission elements as depicted by Figure 11 will be tackled in the upcoming chapter.

Although pre-mission planning is not as time critical as reactive planning, it must be completed before planning timeout. Therefore, not all task plans will be refined in due course by the tactical planner (see Figure 35 for the two-tier planning architecture). Since there can be many possible task plans $\tilde{\pi}_k$, $k \in K$, with |K| being bounded by $n_{\mathrm{ME}}^{n_{o_{\mathrm{ME}}}}$, the task plans can be ranked by the prematurely estimated objective/reward to obtain $(\tilde{\pi}_{s(k)})$, where $s(k)$ is the index of the sorted task plans with descending objective. The ranking is said "prematurely", since the objective of each task plan is estimated approximately, because of the assumption on the linear motion of the HAPS, as well as the probabilistically estimated travel time described in Section 4.3.2. Subsequently, the task plans are being refined (in the order of their sorting $s(k)$) by the numeric flight path planner at the tactical level described in Chapter 3 within the planning timeout [Kiam and Schulte, 2017b; 2017a].

Yet, little has been done on optimizing the combinatorial problem in the task decomposition of HTN planning. In this regard, [Nau et al., 2003] proposes three kinds of searches for a task planning problem with multiple decomposition methods that require the derivation of a reasonable heuristic, which is not always trivial (especially in problems with heterogeneous inter-dependent objectives and

constraints). Alternatively, [Castillo et al., 2006; Fdez-Olivares et al., 2006] takes randomly one of the decomposition methods to break down to non-primitive task, while the temporal HTN planner used in [Kiam and Schulte, 2017b] uses the k-best policies of pre-determined ranked solutions of the Markov Decision Problem. The latter is fast, since the policies are predetermined, and the search for ranked task plans is basically performed by checking the lookup-table generated by the policies. However, if the mission scenario is altered, by adding more HAPS or adding/deleting mission elements, the k-best policies must be re-computed, which is time-consuming. A more flexible approach for solving the combinatorial problem in the decomposition into subtasks at the MA-level for single HAPS monitoring can be done by performing a brute-force search over all decomposition methods, which is a viable strategy for scenarios involving just a few MA, like in Figure 10 [Kiam and Schulte, 2017a].

The brute-force approach to search for the best combination of MA will be used here to solve the task planning for single HAPS. The objective value estimation of each task plan will be formalized in the next subsections. However, as the complexity of the combinatorial problem grows exponentially with the number of HAPS and mission elements, brute-force search can no longer be completed within reasonable time and its implementation as an anytime approach (that ends after a given time with the best plan so far) does not ensure the quality of the proposed solution. Hence, a Genetic Algorithm (GA) is developed in the next chapter to deal with the more complex combinatorial problem in the task decomposition.

4.4.1 **Evaluation of the Task Quality**

Commercially speaking, the MA are the unit considered, since the MR in Table 9 concern either only the MA or the LoI, and the HAPS team is only rewarded with the reward list in Table 10 if the tasks within a MA are all fulfilled. The brute-force search explores every combination possible of MA and evaluates the task plans resulting from all combinations, in order to sort them in descending order of the objective values. The evaluation of the quality of a task plan refers hence to the representation of the plan at the MA level, e.g. $<\mathrm{MA5}\left(t_1^{\mathrm{MA}}, \delta_{t_1}^{\mathrm{MA}}\right), \mathrm{MA4}\left(t_2^{\mathrm{MA}}, \delta_{t_2}^{\mathrm{MA}}\right), \mathrm{WA13}\left(t_3^{\mathrm{MA}}, \delta_{t_3}^{\mathrm{MA}}\right), \ldots>$, and the plan represented at the LoI level, e.g. $\tilde{\pi}^{\mathrm{LOI}} = < \mathrm{to}_{\mathrm{C2}}\left(t_1^{\mathrm{LOI}}, \delta_{t_1}^{\mathrm{LOI}}\right), \mathrm{cross}_{\mathrm{C2}}\left(t_2^{\mathrm{LOI}}, \delta_{t_2}^{\mathrm{LOI}}\right), \mathrm{to}_{\mathrm{LOI1}}\left(t_3^{\mathrm{LOI}}, \delta_{t_3}^{\mathrm{LOI}}\right) \ldots, \mathrm{cross}_{\mathrm{C5}}\left(t_{10}^{\mathrm{LOI}}, \delta_{t_{10}}^{\mathrm{LOI}}\right) >$. The quality of a task plan is assessed based on three objective criteria: the expected cumulative reward, the effort to meet mission requirements, the diversity of the pool of clients the plan satisfies.

4.4.1.1 Criterion 1: Expected Cumulative Rewards per Hour

This criterion focuses on the contribution of the rewards accumulated in the probable events of successful tasks (at the MA-level) along the plan execution. The success of a task is considered "probable", since the durations of each task are only estimated using an approximated movement model of the HAPS and therefore, the estimated start and end time of a task can only probabilistically represent the truth. As already described in Equation 4-2, the duration of a task is modelled uniformly, by taking into account the minimum and maximum airspeed in the presence of wind, as well as the distance travelled to achieve the task.

It shall be noted that the start time of a task plan t_0 is known deterministically. A i-th task at the MA-level o_i^{MA} terminates at $t_i = t_0 + \sum_{j=1}^{i} \delta_j$, where t_0 is deterministic and $\sum_{j=1}^{i} \delta_j$ follows the distribution of the sum of i non-identically distributed uniform random variables δ_j. Therefore, the probability of completing o_i^{MA} at t_i by taking into account that $P(t_i) = P(t_i - t_0) = P\left(\sum_{j=1}^{i} \delta_j\right)$ can be calculated with the following equation, as derived in [Bradley and Gupta, 2002]:

$$P\left(\sum_{j=1}^{i} \delta_j\right) = \frac{\left[\sum_{\vec{\epsilon}^k \in \mathrm{V}^i} \left(\mathrm{f}(\vec{\epsilon}^k, \delta_{1:i})\right)^{i-1} \times \mathrm{sign}\left(\mathrm{f}(\vec{\epsilon}^k, \delta_{1:i})\right) \prod_{j=1}^{i} \epsilon_j\right]}{\left[(i-1)!\, 2^{i+1} \prod_{j=1}^{i} u_{\delta_j}\right]}, \qquad 4\text{-}3$$

where V^i denotes the set with all 2^i vectors of signs $\vec{\epsilon}^k = \left(\epsilon_1^k, \ldots \epsilon_i^k\right) \in \{-1,1\}^i$, $u_{\delta_i} = \left(\delta_i^{\max} - \delta_i^{\min}\right)/2$, $i!$ is the factorial of i, and

$$\mathrm{f}\left(\vec{\epsilon}^k, \delta_{1:i}^h\right) = \sum_{j=1}^{i} \delta_j + \sum_{j=1}^{i} \left(\epsilon_j\, u_{\delta_j} - m_{\delta_j}\right), \qquad 4\text{-}4$$

with m_{δ_i} being the median value of $\left[\delta_i^{\min}, \delta_i^{\max}\right]$.

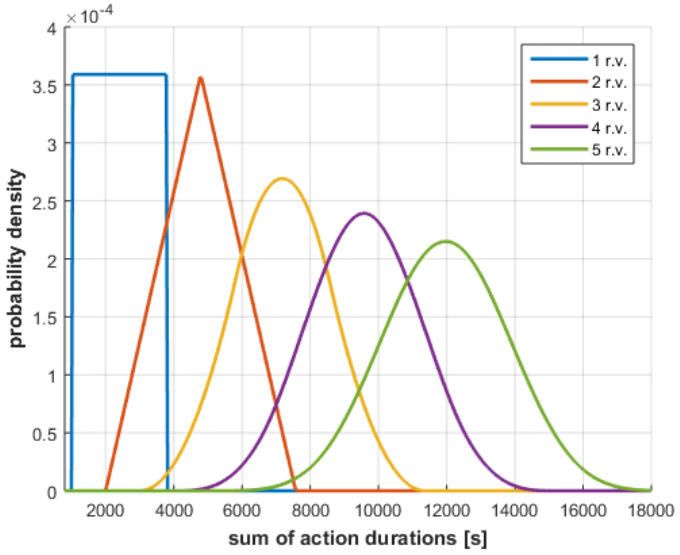


Figure 39. Probability density function of the sum of one to five identical uniform distributed random variables (r.v.) representing the travel time δ_t

The following paragraphs intend to shed some lights at a more intuitive level on the significance of the above equations. Graphically, the probability density function of the sum of more than two uniform distributed random variables depicted in Equation 4-3 approaches a Gaussian distribution and $P\left(\sum_{j=1}^{i} \delta_j\right)$ becomes more wide-spread over a larger range. The more random variables are involved in the sum, the smaller the cumulative probability is over a fixed range, given the progressively wide-spread probability density function as seen in Figure 39, in which the probability density functions of the sum of one to five identical uniform distributed

random variables representing the travel time are plotted. The blue curve depicts the uniform distribution, while the red curve is the probability distribution of the sum of two identical uniform distributed random variables, which resembles the convolution of two square pulses. Beyond two random variables, the distribution of the probability density function approaches normal distributions.

Nevertheless, the durations δ_j of each task are not all identical, as mentioned in the beginning of this subsection. Figure 40 depicts the probability density functions of the sum of one to five non-identical uniform distributed random variables representing the travel time[25]. The probability density functions of the sum of non-identical uniform distributed random variables are similar to those shown in Figure 39, apart from the sum of two random variables, which takes the form of a symmetric trapezium, as seen in Figure 40.

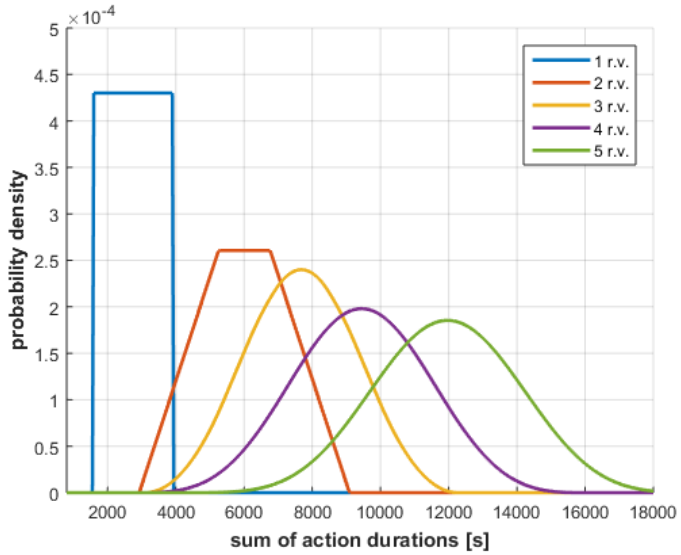


Figure 40. Probability density function of the sum of one to five non-identical uniform distributed random variables (r.v.) representing the travel time δ_{t_i}

As an abuse of notation, $\tilde{\pi}_{i:n}^{\mathrm{MA}} = (< o_i^{\mathrm{MA}}, t_i >, \cdot\ \cdot\ \cdot, < o_n^{\mathrm{MA}}, t_n >)$ denotes the partial plan from the i-th to n-th task at the MA-level for HAPS. To calculate the expected cumulative reward, obtained when applying at state s_i (corresponding to the last monitored MA or WA) and time t_i the remaining plan $\tilde{\pi}_{i:n}^{\mathrm{MA}}$ under the weather w_{t_i} forecasted for t_i, the Time-dependent Markov Decision Process (TiMDP) [Boyan and Littman, 2000] is exploited as a framework to model our problem. To ease comprehension, readers may refer to Appendix 6 for the definition of TiMDP in [Boyan and Littman, 2000]. Without the intention to solve the TiMDP problem, but instead, only the expected cumulative reward function is used to compute the

[25] The five uniformly distributed random variables used to plot Figure 40 are of $\mu_1 = 1163$, $\mu_2 = 1919$, $\mu_3 = 1498$, $\mu_4 = 1960$, $\mu_5 = 1340$ and $m_1 = 2756$, $m_2 = 3254$, $m_3 = 1671$, $m_4 = 1765$, m_5 2518, while Figure 39 was plotted with five identical uniformly distributed random variables of $\mu = 1163$ and $m = 2756$.

expected cumulative reward of a given policy spanning horizon T obtained from the hierarchical planner. The cumulative probabilistic reward of a task plan obtained at MA-level is given by:

$$E\left(\Sigma\middle|s_i, t_i, \tilde{\pi}_{i:n}^{\mathrm{MA}}, w_{t_i}\right) = \sum_{\mu\in\{\mathrm{succ,fail}\}} L\left(\mu\middle|s_i, t_i, o_i^{\mathrm{MA}}, w_{t_i}\right) \cdot \int_{\mathbb{R}} P(t_{i+1}) \cdot \left[R\left(\mu, o_i^{\mathrm{MA}}, t_i, t_{i+1}\right) + E(\Sigma|s_{i+1}, t_{i+1}, \tilde{\pi}_{i+1:n}^{\mathrm{MA}}, w_{t_{i+1}})\right] dt_{i+1}, \quad 4\text{-}5$$

where $L\left(\mu\middle|s_i, t_i, o_i^{\mathrm{MA}}, w_{t_i}\right)$ is the likelihood that action o_i^{MA}, performed at time t_i at state s_i, is successful ($\mu = \mathrm{succ}$) under weather conditions at t_i, i.e. w_{t_i}, or not ($\mu =$ fail); $P(t_i)$ is the probability density function of ending o_i^{MA} at t_i , and $R\left(\mu, o_i^{\mathrm{MA}}, t_{i-1}, t_i\right)$ is the immediate reward obtained when performing o_i^{MA} between times t_{i-1} and t_i successfully ($\mu = \mathrm{succ}$) or unsuccessfully ($\mu =$ fail). Note that Equation 4-5 has the same structure as the habitual formulation used to obtain the expected reward in time-dependent Markov Decision Processes [Boyan and Littman, 2000], which was originally designed to optimize the success rate of arriving in time at a destination via the use of different combinations of means of transport, with each bearing a probable failure or delay.

On the one hand, Equation 4-5 combines the immediate reward $R\left(\mu, o_i^{\mathrm{MA}}, t_i, t_{i+1}\right)$ obtained after monitoring the selected mission area successfully between start time t_i and end time t_{i+1} (using the corresponding values at the bottom of Table 10) or unsuccessfully (using a zero reward), with the expected reward of the remaining action plan $\tilde{\pi}_{i:n}^{\mathrm{MA}}$; on the other hand, it weights them with the probability of finishing the action $P(t_{i+1})$ at the given times and the likelihood of performing the actions successfully and unsuccessfully. Given the progressively wider spread probability distribution of the durations of tasks over time, as illustrated in Figure 40, the integral of $P(t_{i+1})$ over a time interval for the completion of a task in the far future along the task plan is likely to be inferior to that of a task in the near future of the task plan.

Moreover, this expected cumulative reward can be computed using a backward iteration approach as described in Algorithm 4-4, that benefits from the fact that the immediate reward $R(\mu, o_i^{\mathrm{MA}}, t_i, t_{i+1})$ is piecewise constant with respect to t_{i+1}, given that the cloud coverage is piecewise constant[26], and hence $E\left(\Sigma\middle|s_i^h, t_i^h, \tilde{\pi}_i^h, w_{t_i}\right)$ is piecewise constant too, enabling hence the integration in Equation 4-5 to be done piecewise with the analogy that $P(t_{i+1}) = P(t_{i+1} - t_0) = P\left(\Sigma_{j=0}^{i}\delta_j\right)$. The piecewise time intervals considered in the iterative integration are generated using the minimum and maximum of the start time of a task (which is also the end time of the previous task), as well as the minimum and maximum bounding times of the piecewise constant coverage.

[26] Cloud coverage data is usually available at an hour or three-hour resolution.

Algorithm 4-4: backward iteration of the cumulative probabilistic reward
Require: task plan at MA-level $\tilde{\pi}^{\mathrm{MA}}$

1: determine n, length of $\tilde{\pi}^{\mathrm{MA}}$
2: **for** $i = n$ to 1
3: **for** all piecewise intervals $\left[c_{t_i,a}, c_{t_i,b}\right]_j$ of the possible start time of o_i^{MA}
4: **for** all piecewise intervals $\left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k$ of the possible end time of o_i^{MA}
5: **e**valuate immediate reward
$$R\left(\mu, o_i^{\mathrm{MA}}, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, t_{i+1} \in \left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k\right)$$
6: **if** $i == n$
7:
$$E\left(\Sigma \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, \tilde{\pi}_{i:n}^{\mathrm{MA}}, w_{t_i}\right)$$
$$= \sum_{\mu \in \{\mathrm{succ,fail}\}} L\left(\mu \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, o_i^{\mathrm{MA}}, w_{t_i}\right) \cdot$$
$$\sum_k \left(c_{t_{i+1},b} - c_{t_{i+1},a}\right) \cdot \left(\int_{\left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k} P(t_{i+1}) \cdot d_{t_{i+1}}\right)$$
$$R\left(\mu, o_i^{\mathrm{MA}}, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, t_{i+1} \in \left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k\right)$$
8: **else**
9:
$$E\left(\Sigma \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, \tilde{\pi}_{i:n}^{\mathrm{MA}}, w_{t_i}\right)$$
$$= \sum_{\mu \in \{\mathrm{succ,fail}\}} L\left(\mu \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, o_i^{\mathrm{MA}}, w_{t_i}\right) \cdot$$
$$\sum_k \left(c_{t_{i+1},b} - c_{t_{i+1},a}\right) \cdot \left(\int_{\left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k} P(t_{i+1}) \cdot d_{t_{i+1}}\right) \cdot$$
$$[R\left(\mu, o_i^{\mathrm{MA}}, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, t_{i+1} \in \left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k\right) +$$
$$E\left(\Sigma \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, \tilde{\pi}_{i:n}^{\mathrm{MA}}, w_{t_i}\right)]$$
10: **end if**
11: **end for**
12: **end for**
13: **end for**

The determination of $R(\mu, o_i^{\mathrm{MA}}, t_i, t_{i+1})$ in Line 5 is based on mission requirements of Table 9. However, due to the numerous mission requirements, considering all at once could be challenging and might not even be possible given the absence of data during the pre-execution mission planning. Therefore, the mission requirements are only partially considered in the estimation of the probabilistic reward, while others are taken into account at some other steps during the planning. It is assumed that MR1[27] is always true, since a line-of-sight communication with large enough bandwidth is ensured 24/7. MR2[28] is not considered here as it is taken into account in the determination of $L\left(\mu \middle| s_i, t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j, o_i^{\mathrm{MA}}, w_{t_i}\right)$. Equation 7-1 shows the success likelihood values evaluated based on the cloud coverage $w_{t_i}(cc)$

[27] MR1: All the LoIs of the MA are visually recorded and the images are sent to the GCS.

[28] MR2: Ground image coverage of each LoI of the MA is higher than the minimum coverage threshold.

and image coverage threshold th_{image}. MR3[29] can be evaluated using the piecewise constant intervals of start and end time, $t_i \in \left[c_{t_i,a}, c_{t_i,b}\right]_j$ and $t_{i+1} \in \left[c_{t_{i+1},a}, c_{t_{i+1},b}\right]_k$ and determines if the immediate reward takes the value in Table 10 or is null. MR4[30] on the maximum daily revisits cannot be predicted in advance; nevertheless, with Criterium 3 described in Section 4.4.1.3 which intends to promotion diversity in the mission elements visited within the plan, revisit frequency of a mission element is already being curbed. Lastly, MR5[31] is fulfilled if $c_{t_i,a} - t_{end\_max,last}(\text{MA})$ is larger than the allowed minimum revisit time, where $t_{end\_max,last}(\text{MA})$ is the maximum end time of the last visit at the mission area. If the mission area has never been visited, $t_{end\_max,last}(\text{MA})$ is $-\infty$.

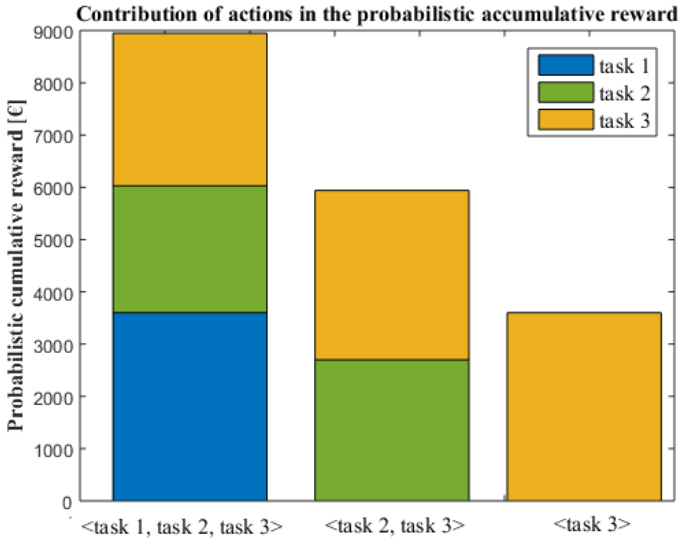


Figure 41. Contribution of actions in the cumulative probabilistic reward

Note also that this way of proceeding puts more emphasis on the rewards of the initial actions than on the final ones, since the probability density function of the sum of uniform distributed random variables $P(t_{i+1})$ become progressively more widespread as shown in Figure 39 and Figure 40. In other words, as seen in the example illustrated in Figure 41, the third task of the task plan, namely "task 3", has less contribution in the expected cumulative reward as the third task (yellow contribution) in the first plan (the first stack) containing three tasks <task 1, task 2, task 3> than as the second task in the second plan, and is most rewarding when it is a standalone task in the last plan, in which the reward gain with task 3 only weighted by its own probability distribution.

The emphasis on the first tasks is especially practical when planning for long time lapses, where there is time to re-plan for the future during the plan execution, which is also reasonable, since the updated weather forecast can be considered.

[29] MR3: Images captured of each LoI within each MA are taken within the corresponding monitoring time windows.

[30] MR4: MA has not been (successfully) visited that day more times than allowed by its maximum revisit frequency.

[31] MR5: Time-lapse between two consecutive visits to the same MA is higher than its allowed minimum inter-visit time.

Finally, since the plan duration varies, it is reasonable to normalize the expected cumulative reward by the plan duration (in seconds) to obtain the objective criterion OC_{rew}:

$$OC_{\text{rew}} = \frac{E\left(\Sigma \middle| s_0, t_0, \tilde{\pi}^{\text{MA}}, w_{t_0}\right)}{T_{\max} - t_0}, \quad \text{4-6}$$

where $T_{\max}$ is the upper limit of the planning horizon $[T_{\min}, T_{\max}]$, and t_0 the start time of the first task. Note however that the obtained plan can end earlier than $T_{\max}$ at t_{n+1}, but the normalization uses $T_{\max}$, instead of t_{n+1}, so that the planner strives to search for a plan that spans as long as possible over the planning horizon. The use of t_0 instead of $T_{\min}$ removes the restriction that the first task must start at the beginning of the planning horizon, which is practical when multiple HAPS are involved, since the start time of each HAPS can differ, as it depends on the end time of the last task.

4.4.1.2 Criterion 2: Effort

To keep the clientele satisfied, the HAPS team is required to perform monitoring missions for as much of their time in the air as possible. If $\delta_{\text{LOI}_\#}$ is the estimated median duration spent at each LOI monitored during the plan, the objective function representing the HAPS effort to satisfy the clients can be calculated with (while normalized with the plan duration):

$$OC_{\text{eff}} = \frac{\sum_l \ \delta_{\text{LOI}_l}}{T_{\max} - t_0}. \quad \text{4-7}$$

4.4.1.3 Criterion 3: Diversity

The plan must also satisfy a big and diverse clientele pool. That is, even when one client pays a much higher reward, the HAPS should strive to monitor the MAs of all the clients. To evaluate this diversity criterion OC_{div}, the Simpson index [Simpson, 1949] is used, which measures the probability of obtaining two different MA when two random draws of tasks are sampled without replacement from the high-level mission plan. In other words, the criterion intends to obtain the probability *of not* drawing the same MA when two of them are drawn without replacement from a given plan. This criterion is calculated with the following equation, where n_{MA} is the number of MAs (or clients), n_c is the number of occurrences of MA_c in the task plan, and N is the total number of rewarded MAs within the task plan, i.e. waiting areas do not count:

$$OC_{\text{div}} = 1 - \frac{\sum_{c=1}^{n_{\text{MA}}} n_c(n_c - 1)}{N(N-1)}. \quad \text{4-8}$$

4.5 Performance Analysis (Complexity, Memory, etc.)

The two-tier planning structure shown in Figure 35 is motivated by the fact that the task planner can be much more efficient but less precise in the objective estimation and the state space can be reduced by reasoning at a higher abstraction space (i.e.

lower temporal-spatial resolution). It aims to determine numerous task plans and sort them with the objective determined as according to the objective criteria described in Section 4.4.1 in a descending order. The task plans are refined with a more precise flight path planner at the tactical level, that considers the wind field as well as the realistic flight dynamics formulated in Equation 3-4 to 3-6.

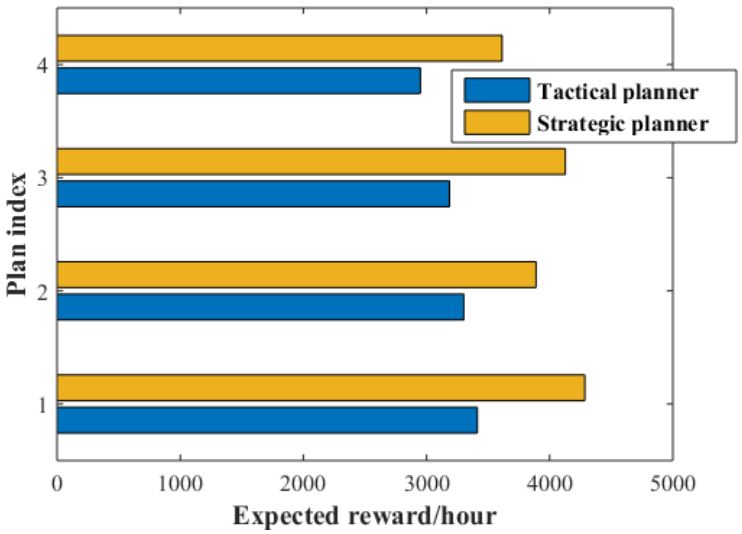


Figure 42. Comparison of reward per hour as predicted by the hierarchical task planner at the strategic planner and the numeric flight path planner at the tactical level

Although reward is not the only criterion to rank the quality of the plans, Figure 42 shows the difference in the cumulative reward (before normalization) estimated by the task planner at the strategic level using the median travel times so that the comparison is fair with both leaving out the weighting via the likelihood of success or probabilistic estimation of the task durations, and the cumulative reward estimated by the flight path planner at the tactical level. The ranking of the plans determined by the task and the path planners differ.

"Artificial Intelligence: it's not about the destination, it's about the journey"
- Frank von Harmelan at STAIRS 2016 in The Hague

5 *Extension of the Mission Planner to Multiple HAPS*

This chapter focuses on the extension of the automated pre-execution task planning described in the Section 4 for multiple HAPS, which is essential, as more HAPS will be deployed in practice to increase mission coverage area (see Figure 11). While still considering the heterogeneous mission constraints listed in Table 8 in a time-varying environment, the optimization for the fulfillment of time-dependent mission requirements (see Table 9) in the presence of multiple HAPS and especially in the presence of more MA complicates the constrained optimization problem [32] encountered for the task decomposition at the MA- and LOI-level even more. Due to the larger set of decision variables and a larger state space, a brute-force search for all task decompositions is no longer feasible within reasonable planning time. To recapitulate, the number of possible task decompositions at the MA-level, as already described in Section 4.4 becomes now $(n_{\mathrm{ME}})^{n_{\mathrm{HAPS}} \times n_{o_{\mathrm{ME}}}}$, where n_{HAPS} is the total number of HAPS, $n_{o_{\mathrm{ME}}}$ is the number the mission elements in the plan, and n_{ME} is the total number of mission elements in the considered scenario.

The Genetic Algorithm (GA) is known to be a flexible solver, given its straightforward random search algorithm that allows a flexible formulation of constraints [Jong et al., 2017]. The random search of the GA can be very efficient if configured appropriately and if the constraints are handled properly. Therefore, the GA is adopted in this work to cope with the optimization of the combinatorial problem that occurs during the decomposition into subtasks of the MA- and LoI-level [Hehtke, 2018; Kiam et al., 2019b; Kiam et al., 2019a]. The advantages of using the GA to guide the search of an optimal decomposition are twofold. First of all, a brute-force search exploring all possible combinations can be avoided. Secondly, if configured appropriately, the search tends toward convergence to optimality. Furthermore, the GA provides plans along each search iteration, and therefore can be

[32] Appendix 6 provides a more formal description of a constrained optimization problem.

implemented as an “anytime”-planner, having in mind that the longer the search is, the better the obtained plans are.

The chapter starts by briefly introducing the principles of a GA. A recapitulation of the objective criteria listed in the previous section will be provided with the generalization for an arbitrary number of HAPS. Subsequently, the implementation of the GA for optimal decomposition in the HTN for multiple HAPS will be described in detail. Results from random tests carried out will be analysed to explain how the GA-related planner parameters are configured.

5.1 Fundamentals of GA

Evolution strategy is a biologically motivated search method for optimization developed in the 1960s by Rechenberg followed by many variants of it, with Genetic Algorithm (GA) being one of the most prominent one [Mitchell, 1999]. The algorithm is reputable for solving combinatorial problem involving an enormous number of possible solutions and has proven to be effective in many UAV planning problems in which optimization is required [Shima et al., 2005; Eun and Bang, 2007; Besada-Portas et al., 2010; Perez-Carabaza et al., 2016; Ramirez Atencia et al., 2019]. The strength of GA lies in its genetic operators to migrate from one generation of individuals to the next in a random way while letting domain-dependent objectives to guide the evolution easily. The most commonly used genetic operators for generating new individuals from the parent generation are crossover, mutation and inversion [Mitchell, 1999].

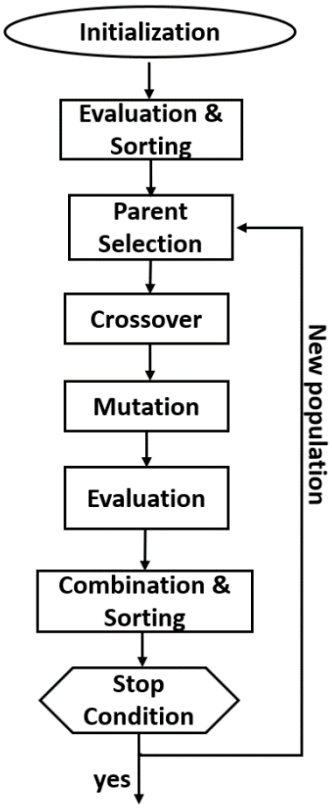


Figure 43. General flow of a GA

In principle, the decision variables to optimize are coded in the *genes* while a combination of parameters is referred to as a *chromosome*. The chromosomes are also often referred to as *individuals* to better suit the rest of the terms used such as a "population of individuals", "parents", "children", etc.

As shown in Figure 43, a basic GA [Mitchell, 1999] is initialized by randomly generating a population of individuals. Subsequently, the fitness of each individual is assessed. Very often, the problem requires the objective to be optimized and the penalty to be minimized. The sum of the objective and the factored penalty (i.e. multiplied by a coefficient) is referred to as the "fitness". The class of problem that consists of optimizing the fitness value is a constrained optimization problem (see Appendix 7). The individuals are then ranked according to their fitness. Some individuals will be selected to be the parents to create offspring that replace the older generation, in hope of bearing a fitter new generation. Therefore, often in practice, the fitter individuals are selected as parents. This rule may vary to maintain the randomness of the search method.

The creation of a new generation is carried out by using genetic operators such as *crossover* [Jong et al., 2017]. Many crossover methods are possible, with the most widely used one being a single-point crossover, in which the parent chromosomes are divided into two parts, namely *head* and *tail*, and by swapping the tails of both parents, new chromosomes are produced. The crossover probability P_{xover} is however not always 1; it can be set to a lower probability value so that some good parents do not undergo any crossover and will be duplicated into the next generation. Another commonly used operator is *mutation*, the purpose of which is to introduce a small probability P_{mutation} to modify each gene of the child chromosomes randomly. While crossover helps to randomly and rapidly explore the search space, it is still substantially guided by the chromosomes of the previous generation, hence the risk of converging too soon to a local optimum. By mutating the genes randomly, the search stands a chance to diversify in the search space explored.

The algorithm repeats with the ranking and creation of a new generation and stops when the criteria are met. The stop criteria can be as simple as the number of iterations, or when convergence is reached, for example, a convergence is reached when every gene of the population has converged, i.e. the value of the gene is identical with for example 95% of the population [Jong, 1975].

The search for optimum based on GA can be optimized via an improvement of any of the steps in Figure 43, resulting in different variants of the GA implementation. They must also be adapted accordingly to each planning problem. If given enough time and if properly modelled, GA is theoretically capable of finding the optimum. However, in practice, finding the global optimum with GA is not guaranteed. GA is especially good at finding an acceptable or solution within reasonable time [Beasley et al., 1993].

5.2 Extension of Objective Criteria for Multiple HAPS

The decision variables involved for the scenario with multiple HAPS $HAPS_1, HAPS_2, HAPS_3, \dots, HAPS_H$ are the same as those defined in Section 4.3. For each HAPS, a HTN is formed, of which the variables bear the superscript $h \in 1,2, \dots, H$ indicating variables of $HAPS_h$. For example, the i-th element in a sequence

of tasks planned for $HAPS_h$ is represented by o_i^h. The HTN of different HAPS must however not violate the inter-HAPS constraints, such as MC3 of Table 8, that prohibits the presence of more than one HAPS in a MA, and must fulfill the inter-HAPS mission requirements, such as MR4 and MR5 of Table 9, which dictate the revisit time and frequency. The task plan for $HAPS_h$ of a given level l of the HTN can be represented by $\tilde{\pi}^{l,h} = \langle o_1^{l,h}(t_1, \delta_{t_1}), \dots, o_n^{l,h}(t_n, \delta_{t_n})\rangle$, where o_i^h are tasks of the given level.

The three criteria $OC_{\text{rew}}, OC_{\text{eff}}, OC_{\text{div}}$ developed in Section 4.4.1 to evaluate the objective of a task plan are slightly adapted to sum up the objectives of all HAPS:

$$OC_{\text{rew}} = \frac{\sum_{h=1}^{n_{\text{HAPS}}} E\left(\Sigma \middle| s_0^h, t_0^h, \tilde{\pi}^{\text{MA},h}, w_{t_0}^h\right)}{\sum_{h=1}^{n_{\text{HAPS}}}\left(T_{\max} - t_0^h\right)}, \tag{5-1}$$

$$OC_{\text{eff}} = \frac{\sum_{h=1}^{n_{\text{HAPS}}} \sum_l \ \delta_{\text{LOI}_l}^h}{\sum_{h=1}^{n_{\text{HAPS}}}\left(T_{\max} - t_0^h\right)}, \tag{5-2}$$

while OC_{div} remains identical as defined before in Equation 4-8. Note that the denominators of the above objective criteria allow the start time of the task plan for each HAPS to differ.

5.3 Implementation of GA for the Search of Optimal Decomposition

The HTN depicted in Figure 37 for a single HAPS can be extended by stacking at each abstraction level the temporal task decomposition of all HAPS together, as shown in the hierarchical task plan for two HAPS in Figure 44. Of all the levels, the MA-level composes the biggest combinatorial problem. The possible orders of MAs to monitor are numerous; moreover, the choice of the decomposition at this level affects the rewards received by the HAPS team. The complexity of the hierarchical task planning problem is multiplied if the mission planning problem is extended to multiple vehicles with even more tasks involved in the mission scenario.

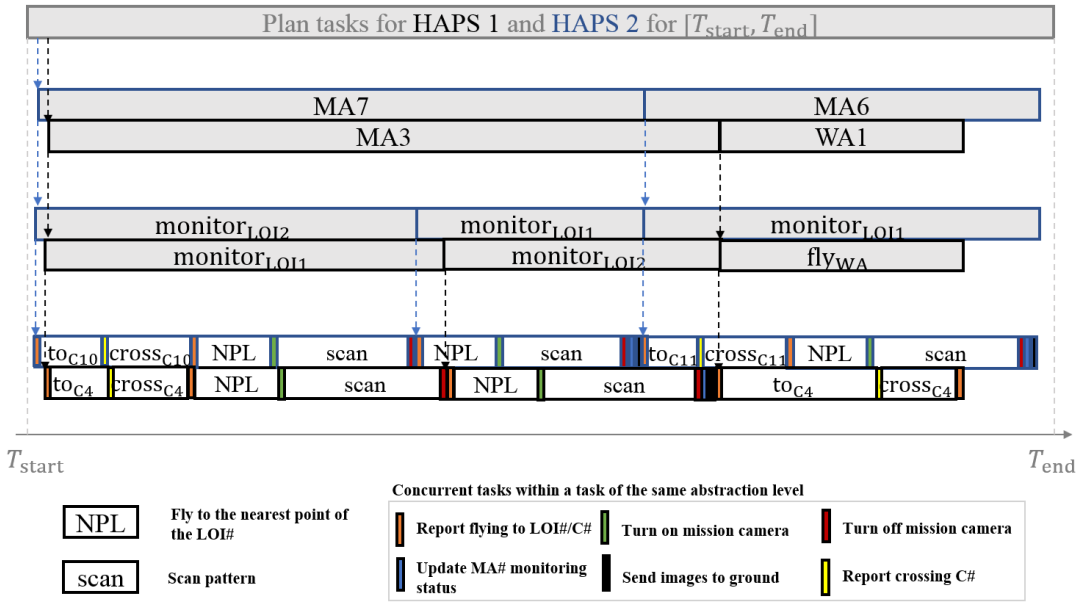


Figure 44. An example hierarchical task planning for two HAPS

The GA is implemented at the MA- and the LOI-level, at which the decomposition poses a combinatorial (or selection out of permutation) problem, to help the search for good plans within the allocated planning time. The tasks at the PoI-level (below the LoI-level) are obtained by undergoing a decomposition-per-fixed-order as described in Algorithm 4-1. Therefore, a search problem is non-existing for the decomposition at this level.

Some problem specific adaptions of the GA are included and will be elaborated in the following subsections.

5.3.1 Encoding of the Decision Variables

The encoding of the genes (decision variables) of the GA is limited to the tasks at the MA-level, i.e. MA# and WA#, and those at the level below, namely the LoI-level, i.e. monitor$_{LOI\#}$ (see Figure 45). Although the task durations are still estimated to check for time-dependent mission constraints and requirements by further decomposing the tasks into primitive tasks at the PoI-level, the temporal factor is left out in the gene encoding, i.e. the new generation does not inherit the task durations of the preceding generation.

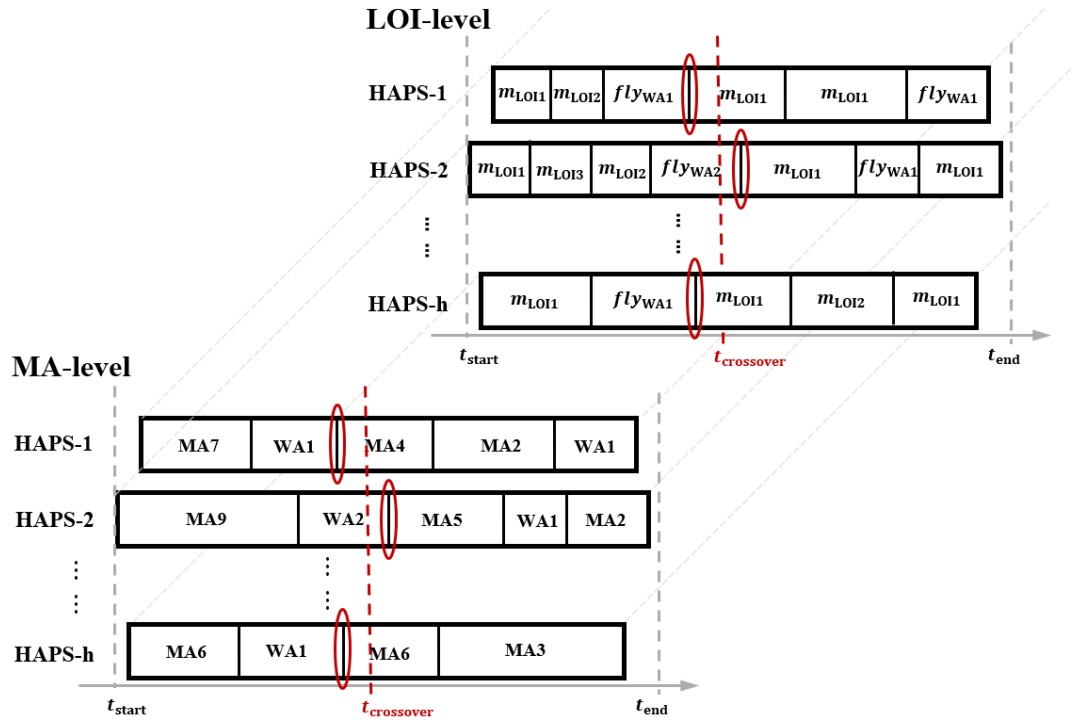


Figure 45. Encoding of a chromosome for *h* HAPS and the temporal crossover

The high-level decision variables for each plan are stored in the GA in an array (with an element for each HAPS) of lists of discrete mission element identifier, with each identifier associated to a task MA# or WA#, as illustrated in Figure 45. Each MA# action in the aforementioned lists is subsequently decomposed per permutation (see Algorithm 4-2) to encode the visiting ordering of its corresponding LoI. Finally, it is worth noting that the number of mission elements n of each high-level plan is variable and is determined while decomposing per combination in order to let the high-level plan span between the mission start and end times $[T_{\text{start}}, T_{\text{end}}]$.

5.3.2 Initialization

The first population is randomly initialized while taking into account the connections of MAs and WAs supported by the scenario, in order to increase the proportion of feasible solutions/chromosomes in the population. The high-level tasks are decomposed using Algorithm 4-1 to 4-3 to primitive tasks of the HTN (see Figure 44) to estimate the duration of each task, as well as the complete mission space, in order to ensure that it remains within the plan horizon $[T_{\text{start}}, T_{\text{end}}]$. Unfeasible solutions according to the MC in Table 8 are discarded and new plans are randomly regenerated, until the initial population is only formed by feasible solutions.

5.3.3 Fitness Evaluation and Constraints Handling

The constraints of the problem are classified into mission constraints (MC in Table 8) affecting the safety of the operation, and mission requirements (MR in Table 9) affecting the rewards received by the HAPS team. The distinction has further implications: MC are hard-constraints that feasible plans are required to fulfill, while

MR are soft-constraints, the violation of which reflects on the reward received for the monitoring of the corresponding MA. Hence, they are treated differently in the GA: the number of times the MC are violated by a multi-HAPS mission plan is accumulated in the constraint violation criterion (CC), while the violation of the latter is considered in the computation of the expected cumulative rewards using Algorithm 4-4. This class of problem is also known as a constrained optimization problem, of which the general definition is provided in Appendix 7.

It is worth noting that the genetic operators (crossover and mutation explained in the next subsection – Section 5.3.4) can generate unfeasible solutions (which do not fulfill the task constraints). It is not unknown that constrained optimization problems can be tricky, since a solution that approaches the optimum may lie in the infeasible region for its constraint violation(s). Discarding an infeasible solution however is not always the best approach, since pursuing the search in its direction may lead an iterative algorithm to get closer to the optimum. Many literature has tried to transform the constrained optimization problem to an unconstrained one, by introducing a fitness function that sums up the objective and a penalty function that transforms the constraint violation into an analytical expression that can for example represent the metric distance of a solution to the closes infeasible region [Jong et al., 2017].

The challenge of such approaches lies with the conception of the penalty function, especially for a real-world problem, in which constraints can exist in all forms for continuous real number parameters or discrete parameters (including predicates represented by integers). [Runarsson and Yao, 2000] introduced stochastic bubble sorting for handling constraints in an evolutionary algorithm. Infeasible solutions, thanks to the stochastic ranking, have the opportunity of surviving some. By doing so, the search for optimum can move from one feasible region to another through an infeasible one, as shown in Figure 46. The stochastic ranking procedure is recapitulated in Algorithm 5-1, so that readers do not have to refer to the original literature. The GA selects the solutions of the survival population among the ones in the new and old population using the stochastic ranking procedure with a low swap probability P_{swap} between consecutive chromosomes, or rather plans. The constraint violation criterion $\text{CC}(i)$ evaluates the number of constraints violated in each task plan at the MA-level $\tilde{\pi}^{\text{MA}}$ (i.e. the number of times a MC is violated). With a low probability P_{swap}, even plans that violate the constraints could be better ranked than plans that do not (see Line 6-8 of Algorithm 5-1), if their objective are better (i.e. $f(i) < f(i+1)$ for a maximization problem, in which the ranking is in the order of descending objective), thus treating both adjacent feasible or infeasible plan equally. If both consecutive plans have violated the mission constraints, i.e. $\text{CC}(i) > 0$, the one that has the higher CC will be moved to the bottom, as described in Line 11-12, disregarding hence completely the objective values of the plans. The comparison of two adjacent plans in the sorting occurs in many sweeps through the population (see Line 2), with a maximum number of sweeps that is equal to the size of the population (see Line 1). This is an example setting of the maximum number of sweeps given by [Runarsson and Yao, 2000]. However, the sorting can cease earlier if during a sweep, no swapping is performed (see Line 16-18).

Algorithm 5-1: constraints handling with stochastic ranking by [Runarsson and Yao, 2000]

Require: task plans at MA $\{\tilde{\pi}^{\mathrm{MA}}\} = (\tilde{\pi}_1^{\mathrm{MA},\boldsymbol{h}}, \tilde{\pi}_2^{\mathrm{MA},\boldsymbol{h}}, \ldots, \tilde{\pi}_n^{\mathrm{MA},\boldsymbol{h}})$

```
n_sweep = size(population) % number of sweep
for i_sweep=1:n_sweep
   flag_swap = false
   for i=1: size(population)-1
      u = rand(1)  % generate a random number between 0 and 1
      if u < P_swap  or (CC(i) == 0  and CC(i + 1) == 0)
         if f(i) is worse than f(i + 1) %f is the weighted objective function
            swap ({π̃_i^MA}, {π̃_(i+1)^MA})
            flag_swap = true
         end if
      else if CC(i) worse than CC(i + 1)
         swap ({π̃_i^MA}, {π̃_(i+1)^MA})
         flag_swap = true
      end if
   end for
   if flag_swap == false
      break
   end if
end for
```

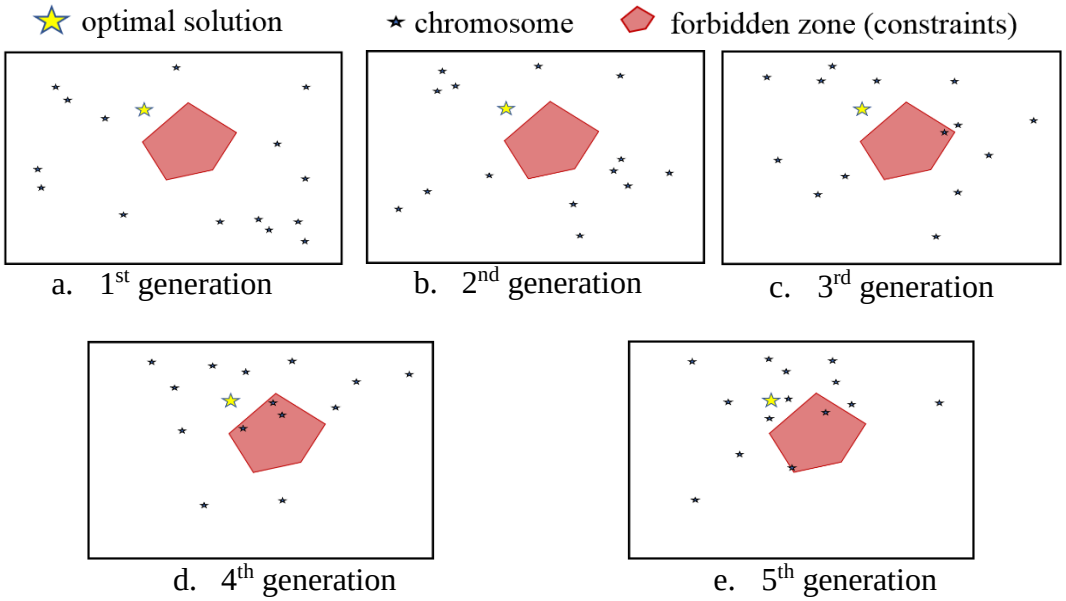


Figure 46. Search for optimum using GA while handling constraints using stochastic ranking

The benefit of using the stochastic ranking described in Algorithm 5-1 can be graphically emphasized using a two-dimensional state space, as shown in Figure 46. Individuals that violate the constraints could survive (see solutions in the forbidden zones in Figure 46c-e), thanks to the stochastic ranking, and thereby accelerates the

convergence to the optimum. Furthermore, being able to traverse through forbidden zones (search space with individuals that violate the hard constraints MC, or rather infeasible regions) is extremely useful, especially if the optimum is surrounded by forbidden areas, while the initial population does not contain any individual that is not separated from the optimum by the forbidden zones.

5.3.4 **Generation of New Population**

The GA uses binary tournament selection shown in Figure 47 for choosing the pairs of parents to perform problem-specific crossover with a high mating probability P_{xover}. In the tournament selection, a few chromosomes (size of tournament $k_{\text{tournament}} = 2$) are randomly selected from the population to pre-select a pool of chromosomes. The chromosome with the best evaluated fitness (as according to the stochastic bubble-sort ranking) is chosen as a parent. The purpose of using a tournament selection is to avoid choosing the same fittest parents repeatedly. Therefore, the smaller $k_{\text{tournament}}$ is, the less likely the same parents will be selected.

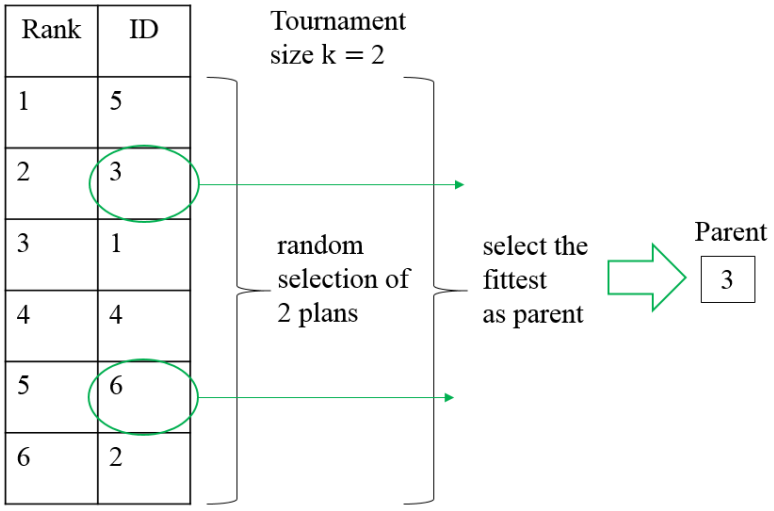


Figure 47. Tournament selection

The implemented crossover is slightly unusual, given the fact that each solution stores the highest-level sequence of actions (i.e. MA# and WA#) of several HAPS and the middle-level LOIs permutation expansions of each MA of the highest-level action sequences. Moreover, the duration of the highest-level actions (as well as the middle ones) are different and random. To deal with that variability in the action durations the crossover performed is from the approach used in [Andrés-Toro et al., 2004]. The crossover is recapitulated in Algorithm 5-2. In particular, a value within $[T_{\text{start}}, T_{\text{end}}]$ is randomly selected as crossing-time t_{xover} (see Line 1). Subsequently, if the mission elements of both parent chromosomes are mission areas and are different, i.e. MA($t_{\text{xover,parent1}}$) != MA($t_{\text{xover,parent2}}$), the nearest estimated median starting time to t_{xover} of MA-level actions is identified for each HAPS (see Line 4). Any action that takes place after this starting time belongs to the tail, while others belong to the head, as illustrated in Figure 45 (for one of the possible high-level solutions when monitoring the scenario Figure 11 with h HAPS). Next, a single-point crossover is performed for MA-level actions for each HAPS, maintaining in the children the existing LoIs ordering associated to each MA in the parents.

If the mission areas that t_{xover} crosses are identical for both parents, i.e. $\text{MA}(t_{\text{xover,parent1}}) == \text{MA}(t_{\text{xover,parent2}})$, a permutation order crossover at the LoI-level will be performed for both parents, and for each HAPS (see Line 6).

	Algorithm 5-2: Selection of a crossover section **Require:** Parent chromosomes of multiple HAPS
1:	randomly select a time instant $t_{\text{crossover}}$ for crossover
2:	**for** each HAPS do
3:	**if** $\text{MA}(t_{\text{xover}})$ of both parents are different **then**
4:	identify the nearest median starting time at MA level and perform crossover
5:	**else**
6:	permute the LoIs at the lower LoI-level
7:	**end if**
8:	**end for**

Each gene at the MA-level of the new generation can be mutated with a relatively low mutation probability P_{mut}. The advantage of increasing the degree of randomness with probable mutations and also with the probable swapping of "illegal" individuals to the higher rank using Algorithm 5-1 are multifold:

1. to avoid converging to a local optimum;
2. to accelerate the search by creating individuals at very different areas that might not be explored without the randomness;
3. to accelerate the convergence to the optimum, or to an optimum isolated by forbidden areas.

The next generation consists of the best from the preceding and new generations, i.e. individuals of parent and child generation are all set together to be ranked using the stochastic sorting method described in Algorithm 5-1.

5.4 Configuration of the GA

GA is known to be flexible for its encoding of decision variables, its ability to handle constraints, as well as its straightforward implementation. However, the search for optimum is efficient (i.e. fast convergence to the optimum) only if the GA is properly configured with optimally tuned parameters like in [Besada-Portas et al., 2010; Perez-Carabaza et al., 2016; 2018]. Some parameters are rather straightforward to set, or rather its variation does not affect the efficiency of the planner much. The choices of these parameters are defined in the following manner:

1. Population size = 50
   The limiting factor to the choice of the population size is the computation time. Test results have shown that convergence is obtained within 50 iterations. Furthermore, with a population size of 50, the computation time of 50 iterations remains less than 3 minutes.
2. $k_{\text{tournament}} = 3$
   The parameter is usually set to a small value in order to increase the randomness induced by the tournament. Experimental results show that other

neighboring values do not have much impact on the efficiency of the planner [Hehtke, 2018].

3. $P_{\text{xover}} = 0{,}9$
   The crossover probability is usually set to almost 1 in order to promote the occurrence of a crossover when generating new chromosomes, instead of retaining the parents. Furthermore, the new generation is selected among the precedent and newly generated chromosomes. Retaining parents decreasing the diversity of the population, thereby affecting the randomness of the search.
4. The weightings of the objective criteria w_{rew}, w_{div}, w_{eff} are set to 0.5, 0.3, and 0.2, so that the weighted objective criteria are brought down to almost the same contributions in the weighted sum of objectives with more weight allocated to the reward term $f = w_{\text{rew}}OC_{\text{rew}} + w_{\text{eff}}OC_{\text{eff}} + w_{\text{div}}OC_{\text{div}}$.

Table 16. Configurations to test for tuning the GA-guided task planner

	$\boldsymbol{P_{\text{swap}}} = \{0.1, 0.2\}$	$\boldsymbol{P_{\text{mut}}} = \{0.05\ 0.1\}$	**Duplicate handling** 1. $P_{\text{mut}} = 0.2$ 2. No handling
C1	0.2	0.05	1
C2	0.2	0.05	2
C3	0.2	0.1	1
C4	0.2	0.1	2
C5	0.1	0.05	1
C6	0.1	0.05	2
C7	0.1	0.1	1
C8	0.1	0.1	2

Nevertheless, not all parameters can be defined in advance, and require therefore experimental tests to decide for the optimal settings, for example the swap probability P_{swap} of Algorithm 5-1, the mutation probability of genes P_{mut}, and the different duplicate handling approaches. P_{swap} must remain low, or the majority of the population are infeasible plans; however, swapping in the case of constraint violation promotes faster convergence, as seen in Figure 46. Additionally, to promote randomness in view of a faster convergence or of avoiding a convergence to a local optimum, mutation is also introduced. However, the mutation probability P_{mut} shall remain low; otherwise, unfeasible plans can be generated, or convergence will be unnecessarily delayed. Furthermore, while generating new chromosomes for the next generation, duplicates are unavoidable, and can dominate the population, leading hence to an early convergence to the duplicate plan. Two duplicate handling methods are tested: either 1) the mutation probability is increased when a duplicate is detected, or 2) the duplicates are not discarded.

The tuning can be done using Wilcoxon tests (a pairwise test of different algorithms or different parameter settings) to compare the statistical results. Different configuration sets listed in Table 16, combining different parameters.

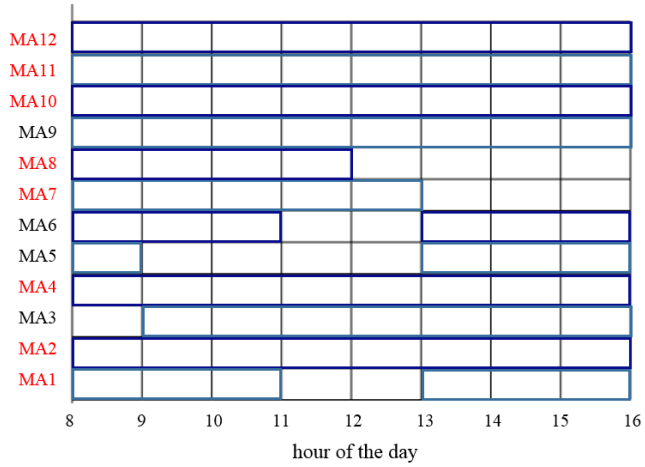


Figure 48. Time windows for ground activity monitoring

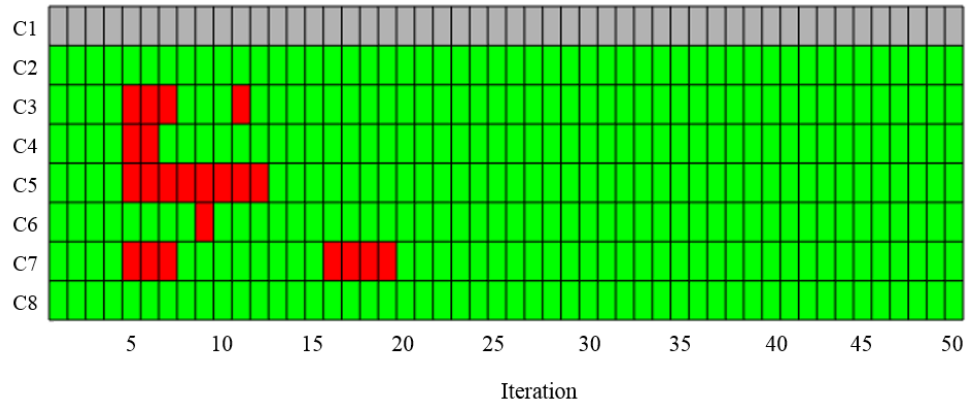
Three randomly generated mission scenarios are used to test the different configuration sets. More than one mission scenario was used, in order to ensure that a configuration does not dominate by chance. The mission scenarios are all based on the airspace structure depicted in Figure 11, while the rewarding time windows for the monitoring tasks are marked with the blue rectangles shown in Figure 48. Besides, the MA with names in black encompass the LoI that can be visited only once during the day; more visits will not be rewarded. MA with names in red encompass LoIs that can be visited as frequently as possible. The minimum time lapse between two visits to the same MA is set to two hours. The wind magnitude and cloud coverage in each mission area are randomly generated using a normal distribution with a mean and standard deviation of wind magnitude of $(\mu_w, \sigma_w) = (3,\ 1)$ m/s for the wind magnitude and $(\mu_{cc}, \sigma_{cc}) = (30\%,\ 15\%)$ for cloud coverage. Since the GA is based on random search principle, 20 runs are performed for each configuration in each test scenario, in order to obtain statistical results of the tests.

Figure 49 summarizes the results of the tests performed using dominance graphs over the number of iterations. To read the dominance graphs, the uppermost configuration is the "base" configuration, while the rest are compared against it. If the base configuration dominates (with higher mean objective value of the 20 runs) against another configuration, the cell is filled with green; if the base configuration is dominated by the compared configuration, the cell is filled with red. The first row is gray, since the base configuration cannot dominate itself, nor can it be dominated by itself.

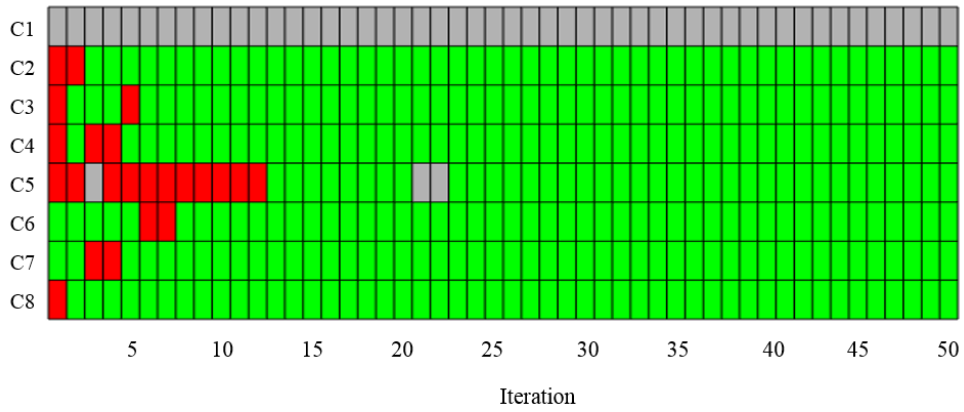
Although the test results could vary slightly from one randomly generated scenario to another, Configuration 1 (C1) thrives in all the tests. The following figure shows that configuration sets like C2 and C8 without duplicate handling have relatively small standard deviation, resulting hence in the tendency to "converge" too early to a suboptimal objective value, f, or rather have difficulties to divert from the population to search in a broader area in order to approach optimum. It is also observed that C3 has a larger standard deviation than C1, and a lower mean. In fact, during the tests, the best plan found by C3 among the 20 runs dominates the best found by C1. However, the larger mutation probability results in a much more

substantial probability in losing more quality chromosomes, and in return, obtaining a more diverse chromosome sample, which prevents the convergence to the optimum in most test runs; therefore standard deviation in objective value of the 20 test runs is larger.

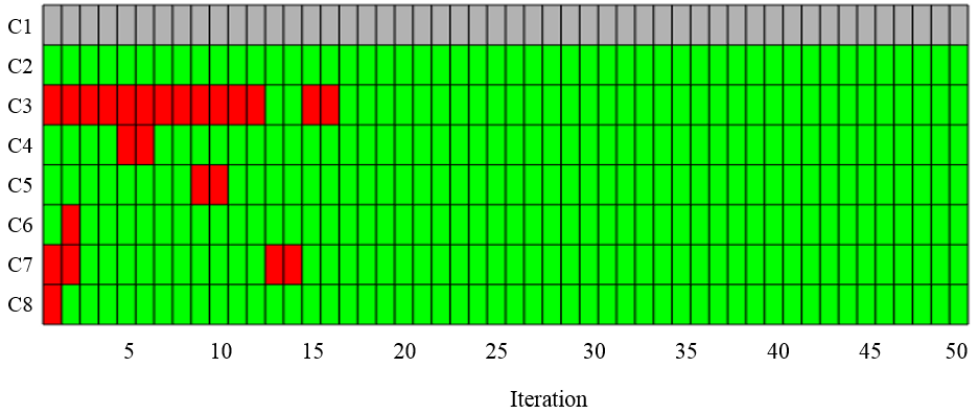
The parameters of C1 are therefore selected. For the GA integrated in the hierarchical task planner at the strategic planning level.



a. First random test scenario



b. Second random test scenario



c. Third random test scenario

Figure 49. Tests for optimal configuration set for the GA

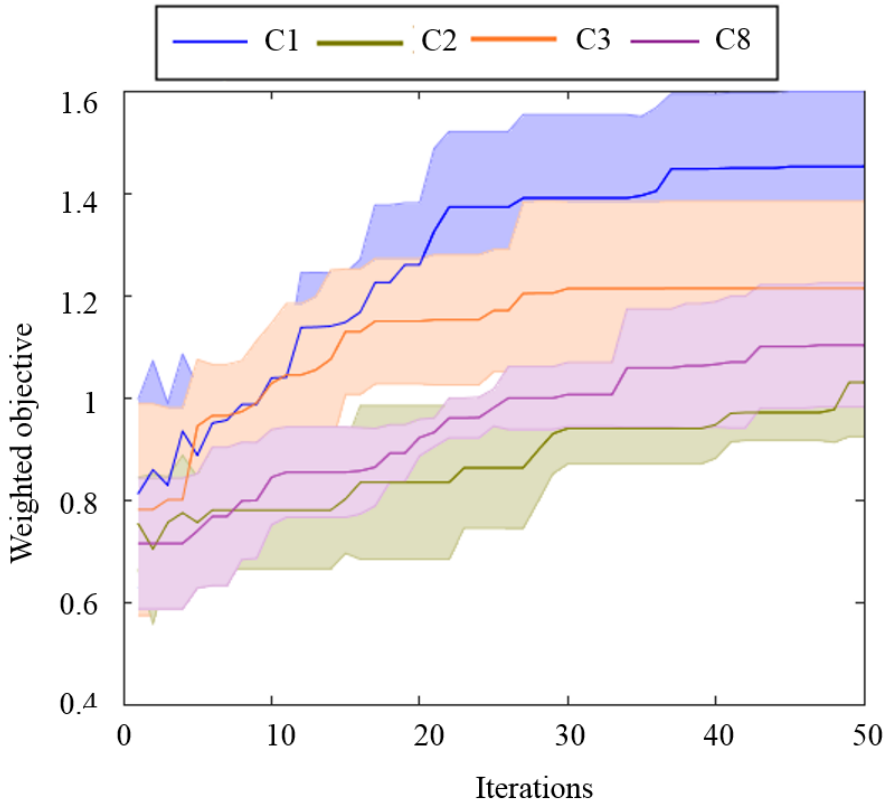


Figure 50. Objective value of the first randomly generated scenario

"What we have to find is a middle way, to find a probabilistic description which says something, not everything, and also not nothing."
- John Horgan cited Ilya Prigogine in *The End of Science*

6 Plan Repair via Reactive Avoidance

As explained and illustrated in the temporal sequence diagram in Figure 16, Figure 17 and Figure 18, the possibility of a danger unforeseen by the pre-execution mission planner cannot be ruled out. If the danger happens far in the future, there is time left for the mission planner to compute a new plan. However, if the danger is nearby and a reactive avoidance is needed, the HAPS can either abandon its current plan and be steered to a safe area to await a new plan from the mission planner, or the reactive strategy can be triggered to guide the HAPS to stay out of the short-term danger and merge back subsequently to executing its reference plan.

The previous is a more straight-forward strategy that will not be treated here in this work. Principally, it can be achieved by first identifying a nearby safe flight zone (i.e. a flight zone that has little to no disturbance for the near future) and a route there will be calculated using the techniques mentioned in Chapter 3, while omitting all mission constraints and requirements and considering only the safety features (i.e. flight dynamics, obstacles and flight zones) [Müller et al., 2018]. The latter is less straight-forward given that the HAPS must consider some hard constraints involving the static or dynamic dangerous zones, while also consider its fidelity to the original mission plan [Attmanspacher, 2019].

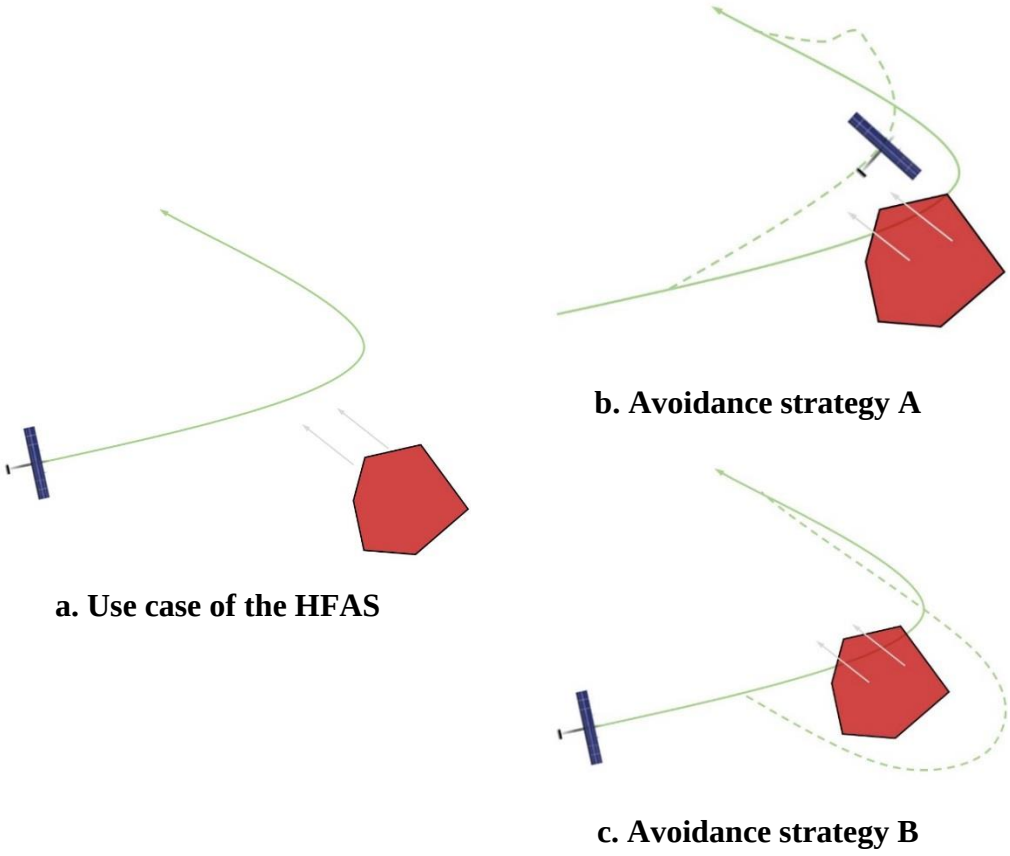
However, the latter, also referred to as the High-Fidelity Avoidance Strategy (HFAS) within this work, has multiple advantages and can be deployed in the following situations:

1. If a small unexpected obstacle is detected by the onboard situation awareness sensors, e.g. cloud, the reference plan can be "repaired" by having the HAPS deviate shortly from the plan without requesting for a new plan. Therefore, with a lower risk of re-planning, the mission management is less complex.
2. If the communications link to the GCS is lost, the deviation from the original path is temporary and can be performed without the GCS. The subsequent merging back to the original plan helps the antenna of the GCS to track the HAPS and reestablish a line-of-sight communication.

In this chapter, the scenario in which the avoidance strategy is applied to is first described graphically. Its Markov Decision Process (MDP) model is then formally detailed. Subsequently, the concise state space setting and the tractability of the model are analysed. The results of the avoidance strategy are shown, and some potential future improvements are also listed.

6.1 Model of Reactive Avoidance Strategy

Graphically, the HFAS can be deployed in a typical situation illustrated in Figure 51, where an unforeseen dynamic obstacle marked with a red polygon is approaching the reference planned trajectory and might collide with the HAPS, should the HAPS proceed with the plan (see Figure 51a). Multiple strategies could be applied to avoid the dynamic obstacle, while trying to remain adhered to the original plan. The HAPS can either try the faster route as shown in Figure 51b to fly pass the colliding point or a longer route to fly behind the obstacle as in Figure 51c. By taking avoidance strategy A, the HAPS must do some extra miles in order to adhere to the original flight trajectory at the right time instant; meanwhile, in avoidance strategy B, the HAPS starts to deviate later to minimize the partial route against the wind vector as much as possible and prolongs the ride with the wind.



a. Use case of the HFAS

b. Avoidance strategy A

c. Avoidance strategy B

Figure 51. Use case of the HFAS

In order to be deployed also in the case of communication link loss, the HFAS should be implemented onboard. Many reactive avoidance techniques for unmanned flights are based on Markov Decision Process (MDP) [Cheng et al., 2019; Temizer

et al., 2010] for practical reasons. The solution to a MDP is computed offline; also known as the strategy, it is a function that maps a state to an action that probabilistically has the highest chance of bringing the agent to its goal. The strategy is used to make decisions on-the-fly, according to the state the HAPS is currently in; no extra computation is necessary, enabling hence the decision-making to be instantaneous, even when the call to the strategy is implemented onboard on a microcontroller with minimal computation capacity.

6.2 Markov Decision Process

In the real world for robotics, the truth is often unknown and can only be estimated. The uncertainty arises from [Thrun et al., 2005]:

1. the erroneous perception of the world using sensors,
2. the inaccuracy of the problem model due to higher-level abstraction to reduce the complexity,
3. the unknown future that can only be predicted.

Markov decision process (MDP) is a widely used technique to determine a control policy for an autonomous agent to decide accordingly and promptly at a given state in order to maximize the rewards and minimize the risks, while the consequences of the action are uncertain [Thrun et al., 2005]. Due to the uncertainties involved in the problem, the decision can only be made probabilistically, i.e. only expected rewards (or expected risks) can be maximized (or minimized). Mathematically, a MDP can be defined as follows [Bertsekas and Tsitsiklis, 1996].

Definition 6-1 (Markov Decision Process, MDP) A MDP is a 4-tuple (S, A, P, R, γ) where S is a finite set of states, A is a finite set of actions, $P: S \times S \times A \to [0,1]$ is the state transition probability, and $R: S \times A \to R^+$ is the reward function. γ is the discount factor selected within the range $]0,1]$, in order to prioritize the rewards in the near future.

The solution to an MDP model is a policy which seeks to optimize the rewards of a mission accomplishment by maximizing the discounted expected cumulative reward:

$$R_T = E\left[\sum_{t=0}^{T} \gamma^t r_t\right], \qquad 6\text{-}1$$

where r_t is the immediate reward, T is the planning horizon.

Note that MDP does not deal with the uncertainty due to perception, which is dealt with in Partially Observable Markov Decision Processes (POMDP).

An interesting advantage of the MDP lies with its two-step deployment, which consists of the offline determination of a decision policy (also known as the "strategy" in the reactive avoidance for HAPS) and subsequently the online invoking of the policy during execution [Thrun et al., 2005]. The online step is efficient since the complex computation of the policy is completed offline. By doing so, the consequence of an action can be determined after the execution of each control decision, and the next action can be decided with the knowledge of the current state

after the execution of the previous action[33]. This is especially beneficial for reactive avoidance since first-of-all, there is no time allowance for tedious computation of a plan and secondly, the control parameter can be decided online while taking the total rewards/risks over the execution horizon into account, by assuming an infinite horizon while determining the optimal policy.

6.2.1 Modelling the HFAS

Given a plan determined with the hybrid HAPS mission planner depicted in Figure 35, a reference four-dimensional trajectory $\{p\}$ can be obtained by integrating over time the numeric control parameters decided by the flight path planner at the tactical planning level. If the trajectory is closely followed by the flight controller, the position where the HAPS should be at time t is $p(t)$, as shown in Figure 52. In the HFAS, since the HAPS maintains its altitude during operation, the altitude element is neglected, and only lateral avoidance is considered.

If obstacles unforeseen during the offline planning are detected at t and appear to be in the way (like obs_1) or approaching the reference trajectory (like obs_2), the HAPS must deviate from the reference plan, but still keep the next waypoint $p(t + \Delta t)$ in mind, so that it adheres to $\{p\}$ whenever possible. Δt is the decision step, or time interval at which a decision will be made based on the information gathered from the situation awareness module described in Section 2.2.4.

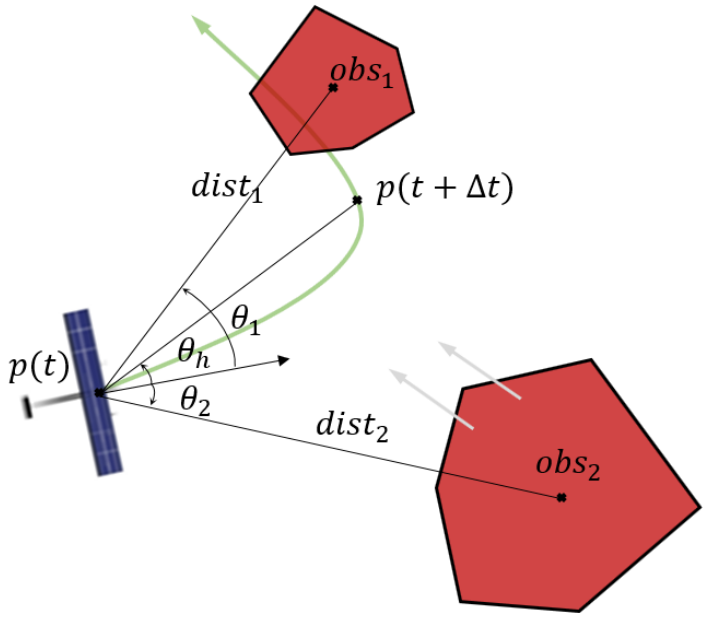


Figure 52. Problem parameters for avoiding an obstacle reactively

Note that it is essential to describe the situation as concisely as possible with parameters so that a MDP remains tractable. The workspace *X* of the HAPS defined in Section 2.3 is excessive. In the reactive avoidance use case, it is worth noting that the absolute positions of the HAPS or the obstacles are unnecessary information. Rather, the relative configurations between the HAPS and the obstacles are important

[33] The MDP is a "memoryless" model.

to help the decision-making process, as depicted in Figure 52. The relative configurations are described by the relative distance $dist_i$ and the relative bearing θ_i between the HAPS velocity vector and the vector connecting the HAPS to the barycenter of the obstacle obs_i. However, experimental results show that $dist_i$ better denotes the minimum distance to the obstacle (edges), as it is more consequent for large obstacles, as shown in Figure 56.

In the case of HAPS, since only a lateral deviation is considered here, the position vector $p(t)$ can be reduced to a two-dimensional vector and the pitch angle between the HAPS and the barycenter of the obstacles can be neglected in the problem statement. $dist_i$ is hence the distance projected on the two-dimensional plane. Similarly, the relative configuration to the next waypoint $p(t + \Delta t)$ on the trajectory can be described by the relative distance $dist_h$ and the bearing θ_h between the velocity vector and the vector connecting the HAPS to $p(t + \Delta t)$.

For the sake of "fidelity" to the reference path plan, it is also necessary to keep track of the reference path, or rather the position the HAPS is supposed to achieve at the next time instant, if the unforeseen obstacles did not exist. $dist_h$ denotes the distance to the next position of the reference path $p(t + \Delta t)$, while θ_h denotes the bearing between the HAPS velocity vector and the vector connecting the HAPS to the $p(t + \Delta t)$.

The actions considered, a, are increase/decrease its own heading by $\Delta\theta$, thus changing also θ_{obs} and θ_h, or "do nothing", while maintaining the true airspeed. Qualitatively, if the obstacle(s) is far away, or if the HAPS course heading is already leading the platform away from the obstacle, "do nothing" is the reasonable action. If the obstacle(s) is close by, the HAPS must be steered away from the obstacle by either increasing or decreasing its heading, therefore changes also θ_{obs} and θ_h accordingly.

The distance and the bearing to an obstacle at the next state (i.e. $dist'_{obs}$ and θ'_{obs}) are independent of the other obstacles, and $dist'_{obs}$ depends on the $dist_{obs}$, θ_{obs} of the current state and the action taken a, while θ'_{obs} depends only on θ_{obs} and the action taken. Likewise, the dependency for $dist'_h$ and θ'_h of their previous states is deduced. Therefore, the state transition probability for $\{obs_1, \dots, obs_o\}$ can be simplified to Equation 6-2.

$$
\begin{aligned}
&P(s'|s,a) \\
&= P(dist'_1, \theta'_1, \dots, dist'_O, \theta'_O, dist'_h, \theta'_h | dist_1, \theta_1, \dots, dist_O, \theta_O,\ dist_h, \theta_h, a) \\
&= \left(\prod_{obs=1}^{O} P(dist'_{obs} | dist_{obs}, \theta_{obs}, a) \cdot P(\theta'_{obs} | \theta_{obs}, a) \right) \\
&\quad \cdot P(dist'_h | dist_h, \theta_h, a) \cdot P(\theta'_h | \theta_h, a)
\end{aligned}
\qquad 6\text{-}2
$$

Note that in practice, the problem is formulated only for up to two obstacles $\{obs_1, obs_2\}$. The reason being that a reactive avoidance in the presence of many obstacles is too risky; in that case, the reference flight plan should be aborted, and the HAPS shall either be steered to a safe zone awaiting a new plan [Müller et al., 2018]. If this is also not an option, an emergency landing is required.

Thus, a state $s \in S$ of the MDP is hence $s = (dist_1, \theta_1, dist_h, \theta_h)$ in the case where only one unforeseen obstacle is present in the vicinity of the HAPS, and $s = (dist_1, dist_2, \theta_1, \theta_2, dist_h, \theta_h)$ if two unforeseen obstacles are present in the vicinity. Strategies for a single obstacle and for two obstacles are determined separately and invoked onboard by the reactive avoidance module according to the number of detected unforeseen obstacles.

6.2.2 **Solution to MDP**

An MDP can be solved for infinite planning horizon using the value iteration[34] [Bertsekas and Tsitsiklis, 1996]. The solution is a policy π_{HFAS}, which is a mapping function $\pi_{\mathrm{HFAS}}: S \rightarrow A$ that maps a state $s \in S$ to the action $a \in A$ that maximizes the expected value:

$$V(s) = \max_{a \in A} \left[r(s,a) + \sum_{j=1}^{N} V(s') P(s'|a,s) \right], \tag{6-3}$$

and the policy obtained $\pi_{\mathrm{HFAS}}: S \rightarrow A$ is as follows:

$$\pi_{\mathrm{HFAS}}(s) = \underset{a \in A}{\operatorname{argmax}} \left(r(s,a) + \sum_{j=1}^{N} V(s') P(s'|a,s) \right). \tag{6-4}$$

6.3 Implementation and Results

Although the model in Section 6.2.1 could be expressed in a continuous space, the MDP is however intractable without discretization. The discretization of the state space must not be too fine, or the complexity of the problem increases exponentially. However, it must be reasonable so that the problem is not overly abstracted, resulting in a loss of information or maneuverability. Table 17 summarizes the discretization of the state space.

Table 17. State space discretization of the MDP

State parameter set	Domain discretization
$\boldsymbol{A = \{a\}}$	{-1, 0, +1} °/s
$\boldsymbol{\Theta_h = \{\theta_h\}}$	{-15, 0, 15, 180} °
$\boldsymbol{Dist_h = \{dist_h\}}$	{1, 100} km
$\boldsymbol{\Theta_{obs} = \{\theta_o\}}$	{-90, -45, -15, 0, 15, 45, 90, 180} °
$\boldsymbol{Dist_{obs} = \{dist_o\}}$	{5, 15, 100} km

Note that the values in the discretized space denote the upper limit of a range with the lower limit taking either the value before or the natural lower limit the metric measures. For example, "-15" in the state parameter set for the bearing with the next position on the reference path, Θ_h, denotes the range [-180,-15[, and "100" of $Dist_{obs}$

[34] Solving the MDP formulated for reactive avoidance for HAPS with value iteration or policy iteration results in the same policy. Furthermore, no significant difference in performance was observed [Attmanspacher, 2019].

denote the distance range [15, 100[. The distances are discretized based on thresholds that signify the levels of "criticality".

The distance to the obstacle(s) or to the next waypoint on the reference path are set in a way that the avoidance strategy is able to decide which obstacle is more critical and if it is far enough from danger to proceed to adhering the reference path. Meanwhile, the bearing to the obstacle(s) or to the next waypoint on the reference path is to inform the reactive avoidance module if the HAPS is flying into or away from the obstacle(s) or the next waypoint.

The HFAS computed on the GCS by the mission planner is invoked onboard by the reactive avoidance module when at least one unforeseen obstacle is in the vicinity of the HAPS, i.e. $dist_h < 100$ km. The HFAS stops to apply when all obstacles are beyond 100 km from the HAPS, and if the HAPS $\|p(t+\Delta t) - p(t)\|_2 < 1$ km, or if $\|p(t+\Delta t) - p(t)\|_2$ converges to a constant value. The latter implies that the HAPS has already adhere to the reference path but is unable to catch up with the next waypoint, especially in the case where the reference path is linear.

6.3.1 **Single Static Obstacle**

Figure 53 shows on the North-East plane the alternative trajectory taken to avoid an unforeseen static obstacle lying on the reference path (marked in red). After the avoidance, the HAPS adheres back to the reference path, avoiding possibly hence a re-planning in this case. The markers on the blue alternative path denote time instants or positions at which an action that is not "do-nothing" is undertaken, i.e. $a > 0$ °/s.

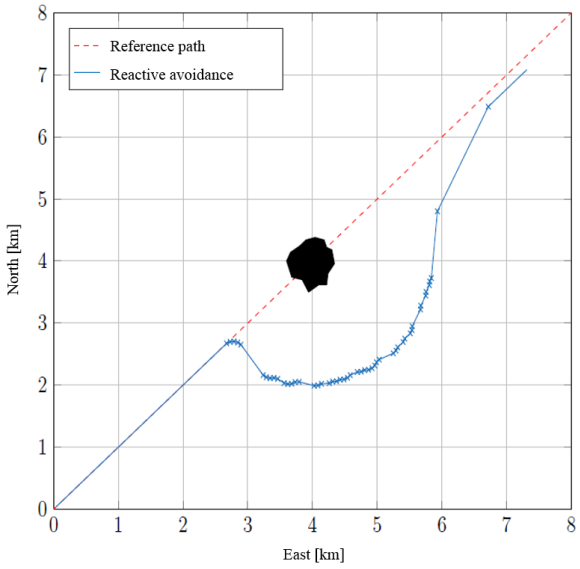


Figure 53. HFAS for a single static unforeseen obstacle

The deviation from the reference path over time (shortest distance to the reference path) is shown in Figure 54, while Figure 55 shows the difference in position $\|p(t+\Delta t) - p(t)\|_2$ at each time instant. Although, the difference does not necessary decrease to zero, since the HAPS is "behind schedule", but it does not increase unlimitedly, resulting in a loss of track of the reference path, and requires thus a re-planning. In this particular case, the HAPS is 610 s behind schedule.

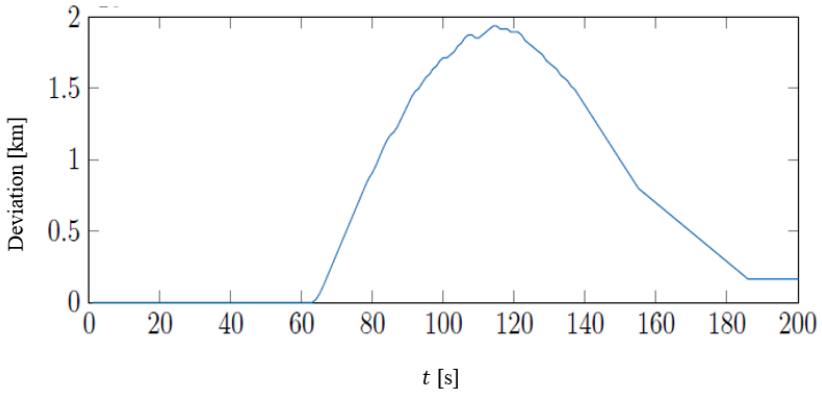


Figure 54. Deviation from the reference path over time

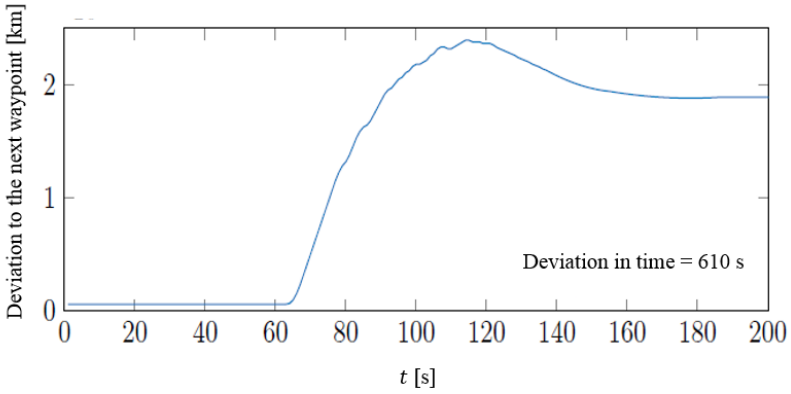


Figure 55. $\|\boldsymbol{p}(\boldsymbol{t}+\boldsymbol{\Delta t})-\boldsymbol{p}(\boldsymbol{t})\|_2$ at each instant

6.3.2 Two Static Obstacles

Similarly, Figure 56 shows the undertaken flight path to avoid the two static obstacles. The difference in this example is that the avoidance is done much closer to the obstacles. The reason being that instead of taking the distance to the barycenter of the obstacles for $dist_o$, in this example, the state parameter denotes the minimal distance to the obs. By doing so, the deviation from the reference path is reduced, and there is also more space for avoidance, especially when threat map becomes denser with the presence of a second obstacle.

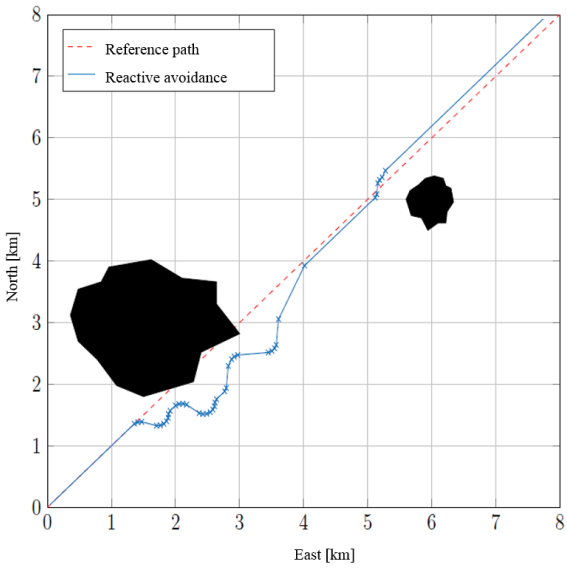


Figure 56. HFAS for two static unforeseen obstacles

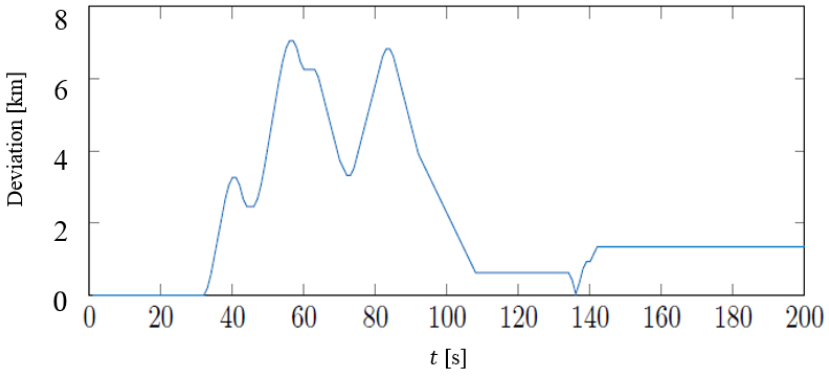


Figure 57. Deviation from the reference path over time (2 static obstacles)

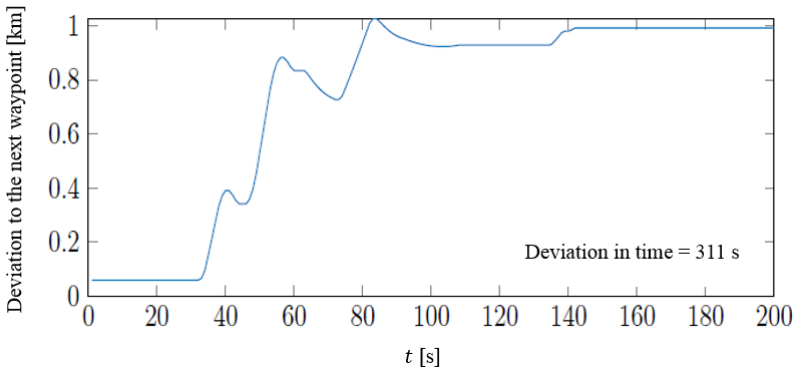


Figure 58. ‖p(t+Δt)-p(t)‖ at each instant (2 static obstacles)

6.3.3 **Moving Obstacle(s)**

The computed HFAS was also tested for moving obstacles, although the movement of the obstacles is not modelled. However, this abstraction of state parameter is reasonable, since the action taken to avoid a collision is taken at every instant, freezing thus the movement of the obstacles, of which the HAPS has no control. Figure 59 shows the alternative flight path taken over time to avoid the moving

obstacle, while the deviations from the reference path and the quantitative description of how far behind schedule the HAPS is are shown in Figure 60a and Figure 60b respectively.

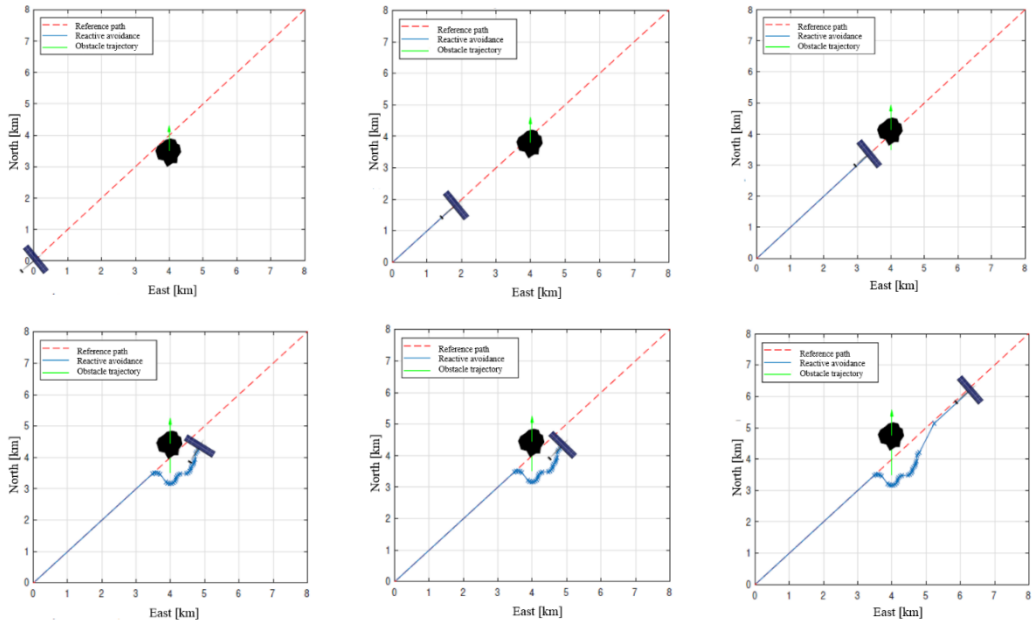
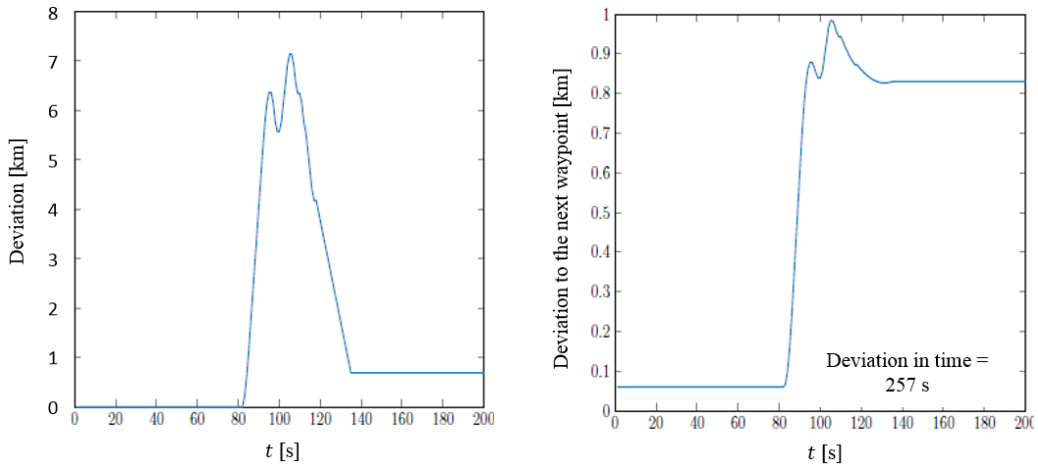


Figure 59. HFAS a single unforeseen moving obstacle



a. b.

Figure 60. a) Deviations from the reference path; b) ‖p(t+Δt)-p(t)‖ for a single unforeseen moving obstacle

More challenging tests were conducted on two moving obstacles. HFAS succeeded to reactively avoid collisions with the obstacles, as shown in an example test in Figure 61 and Figure 62.

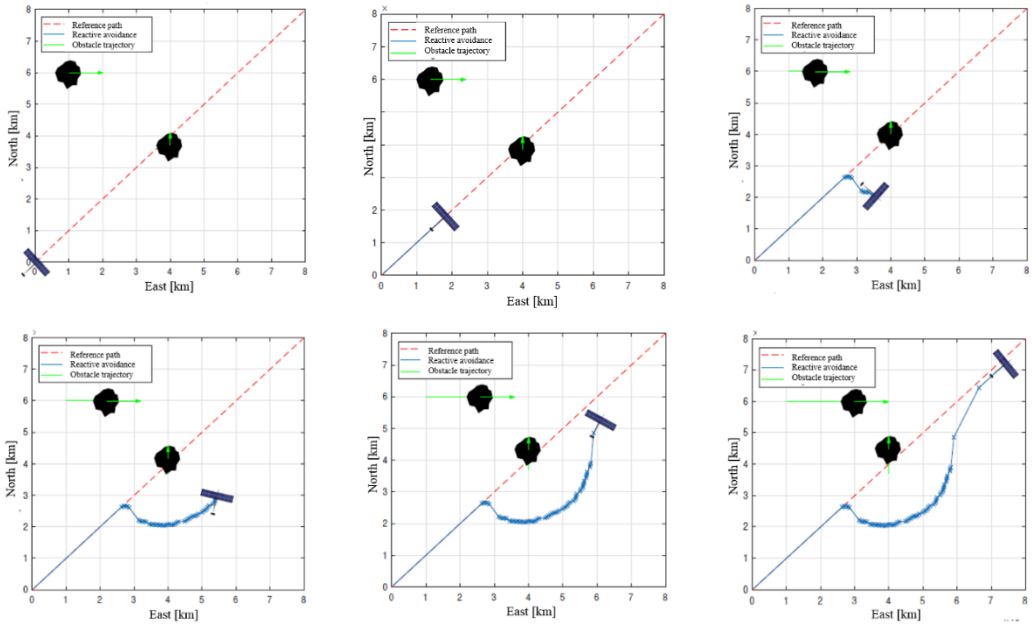
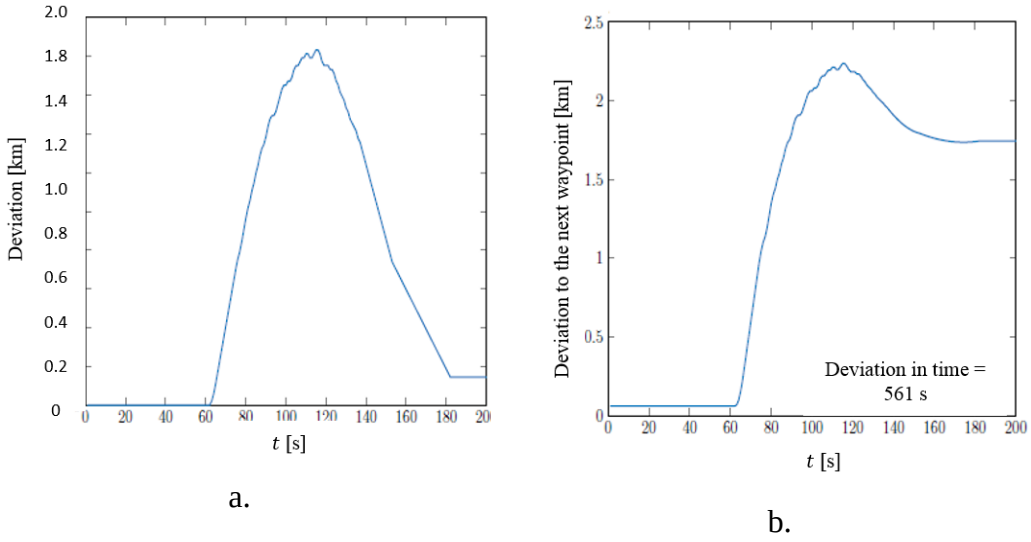


Figure 61. HFAS two unforeseen moving obstacles



a. b.

Figure 62. a) Deviations from the reference path; b) ‖p(t+Δt)-p(t)‖ for two unforeseen moving obstacles

"The problems today are not caused by super smart AI, but stupid AI"
- Toby Walsh in *WIRED* on 20 September 2017

7 Implementation and Validation

In this section, the implementation of the pre-execution mission planner described from Section 3 to Section 5 is illustrated to show how the various modules interact as a whole, as well as with the operator, in order to fulfill the system specifications described in Section 2.2. Test results obtained using real weather data and a HAPS simulator are also analysed to validate the planning concept. Although the computation of HFAS described in Section 6 is also performed prior to mission execution by the mission planner on the GCS, it is independent from the rest of the modules and therefore, will not be further elaborated in this section. Furthermore, since the aircraft dynamics model used in the HFAS is based on the same model adopted for the point-to-point flight path planning (see Section 3), the validation of the latter in this section implies the correctness of the HFAS. How the HFAS is triggered during a mission flight was briefly described in Section 6, but the exact implementation falls out of the scope of this work.

To recall, as fixed-wing aircrafts, HAPS can fly typically at an optimal equivalent airspeed of 9 m/s [Müller et al., 2018] and cruise at the operating altitude at a speed of $|v_{\mathrm{TAS}}|\sim$ 29 m/s (see Table 5 in Section 1 for the specifications of the platform). Its ground velocity can be obtained via $v_{\mathrm{GS}} = v_{\mathrm{TAS}} + v_{\mathrm{w}}$. Besides, they are equipped with an electro-optical mission camera and therefore must perform the monitoring tasks during daylight. For example, on a day in spring, the ground activity monitoring mission can start at 08:00 in the morning and finish at 16:00 in the afternoon. Therefore, for the tests used in this section, the HAPS are set to linger at the WA (HAPS-1 in WA13 and HAPS-2 in WA14) to await commands at 07:00 local time; meanwhile, a plan for the next 8 hours must be determined and approved by the operator. Note that the start time of the daily mission depends on the local sunrise and sunset hours.

7.1 Implementation of the Mission Planner

While Figure 35 only shows the functional architecture among the modules, Figure 63 shows here in a much more concise manner the functions of the modules as well as their interaction with the human operator. The task planner implemented at the strategic planning level first plans for the tasks the HAPS must execute over a given

plan horizon. At this level, mission constraints and requirements are taken into account, but the four-dimensional wind field, as well as the HAPS non-linear dynamics constraints are only very approximately considered, therefore the constraints are marked in light gray. Only a range of the possible wind magnitude is considered, i.e. $|v_w| \in [0, 5]$ m/s, while the platform dynamics are assumed linear at constant speed.

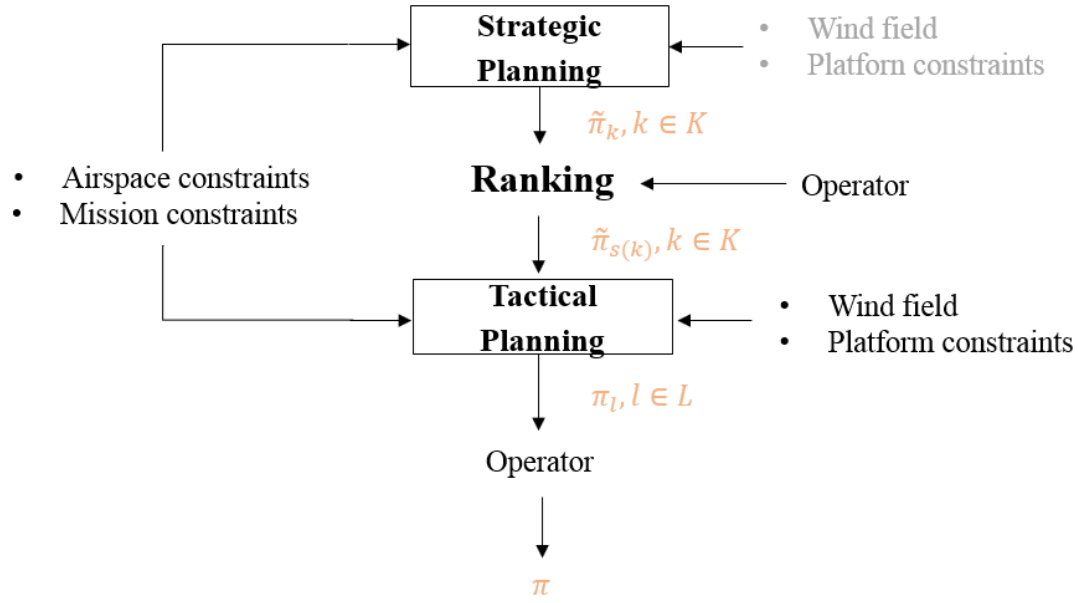


Figure 63. Procedure of a mission planning

The feasible task plans found within the allocated time for strategic planning (3 minutes) are then ranked accordingly according to their objective values and presented to the operator. At this level, the operator is also entitled to intervene and alter the ranking of the task plans.

With the rest of the planning time, the task plans are "refined" by the numeric flight path planner implemented at the tactical planning level, in order to better estimate the time and position of the HAPS over the plan horizon, thereby estimate better the quality of the plans. At this stage, wind field and the platform dynamics are numerically treated. The refined plans will be sorted accordingly and presented to the operator, so that the plan to execute π can be selected.

In order to "refine" as many task plans as possible, a multithreading is implemented, so that two task plans are being concurrently refined by the path planner. The multithreading can be multiplied if necessary. The primitive task plan can be represented by a sequence of time-stamped PoIs, which are sequentially used (in total-order) as start and goal conditions for the tactical numeric flight path planner.

7.1.1 User Interface of the Mission Planner

In this subsection, the User Interface (UI) will be briefly described, for a better understanding of the potential interaction with the human operator the mission planner offers[35].

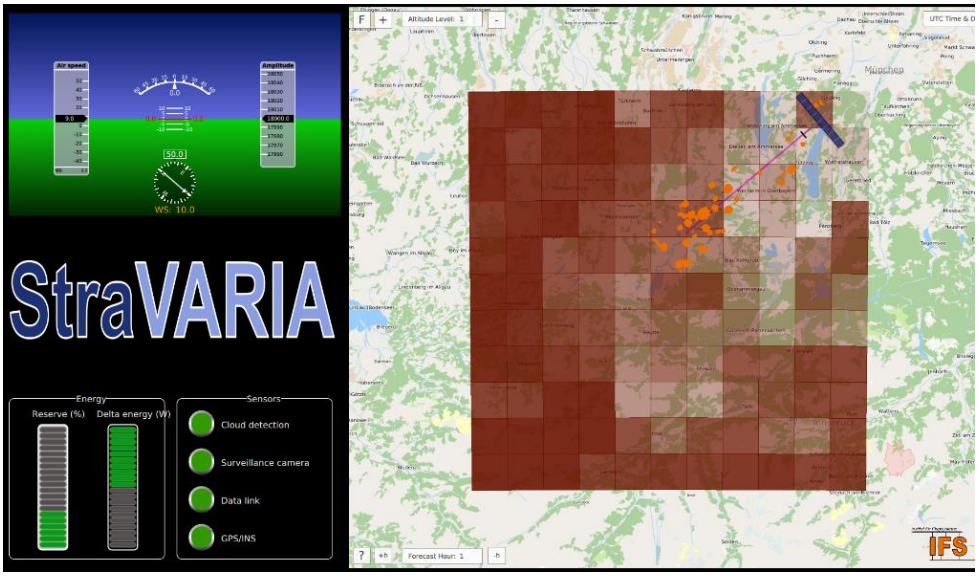


Figure 64. Cloud map on coverage map from onboard EO-sensor shown on the user interface

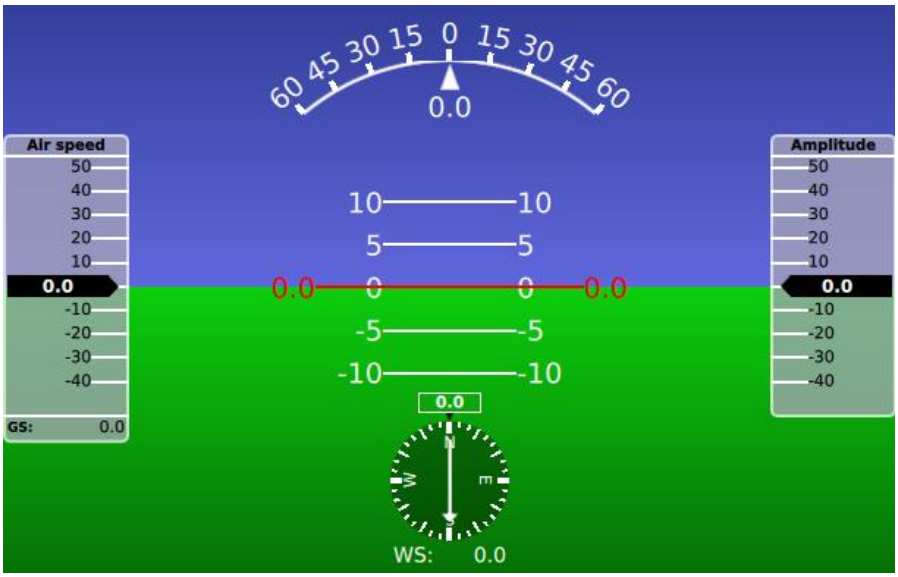


Figure 65. Flight instrument

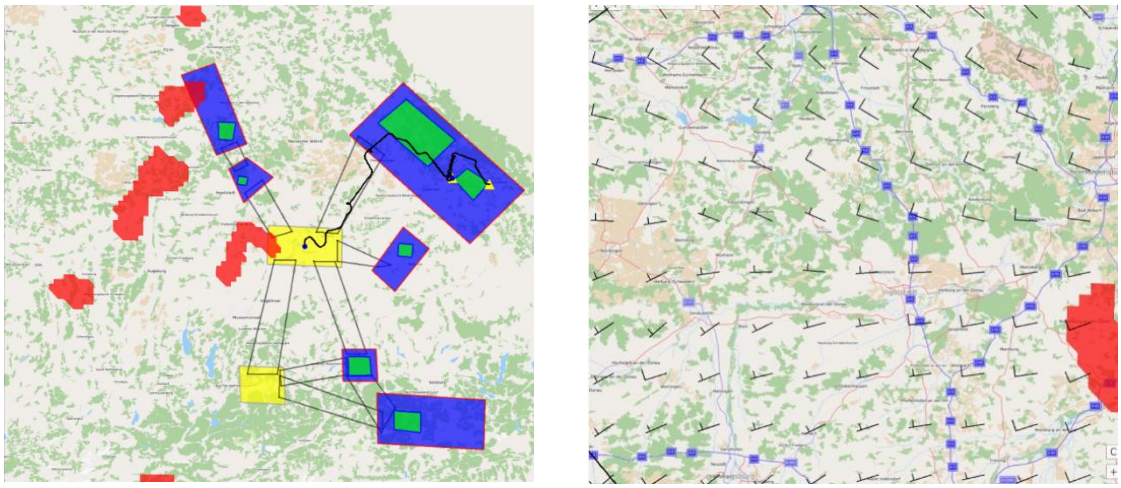
Shown in Figure 64 is the main display of the UI. On the left are the flight instrument display, and the display for sensor health status as well as the energy

[35] Note that no systematic study was conducted in this work on how the planner should interact with the human operator(s) and when a human intervention is deemed optimal. Although essential, the study falls out of the scope of this work.

management. An enlarged view of the flight instrument display can be seen in Figure 65, in which the true airspeed (“Air speed”), the ground speed (GS) are shown on the left bar, the amplitude (“Amplitude”) on the right, the pitch and roll angles in the middle. The compass at the bottom shows the yaw angle of the HAPS. An important display here is also the wind direction, which is indicated by the arrow in the compass.

On the right of the display is a map on which the current location of the HAPS is marked. Also found on the map is the weather map derived from the onboard cloud detection unit; in this particular display, the cloud coverage map in grid and the cloud map with units of cloud situated between the HAPS and the ground are displayed.

Optionally, the map display can show other information as well. In Figure 66a is the mission scenario for a single HAPS depicted on the map, together with the planned flight path marked with a black line from the WA to multiple POIs in a MA. The red polygons are the weather critical zones summarized from the weather forecast data. The display of additional information, for example the motorway flags on the ground or wind field etc. can also be activated if necessary (see Figure 66b).

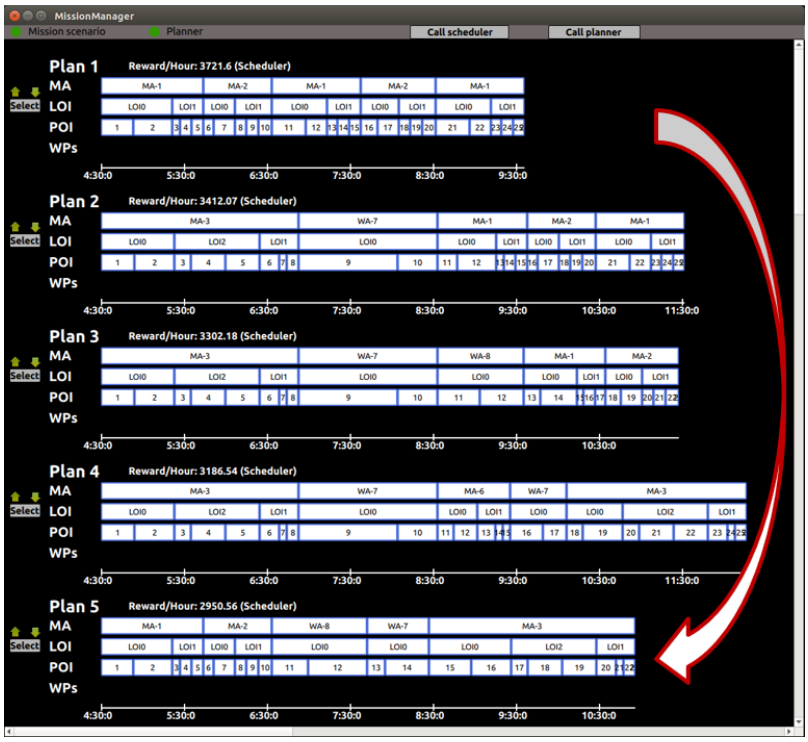


a. Mission scenario with the planned path

b. Wind vector field and the details of the ground

Figure 66. Optional information displayed on the map

A second display is dedicated for the mission planner, as seen in Figure 67, in which the plans for a single HAPS over a plan horizon of 8 hours are displayed. At this stage, only the task planner at the strategic level has completed its planning. The suggested task plans are displayed in a sorted order, according to the objective of each. The hierarchical structure of the plans is also shown, i.e. the different physical abstraction levels from the MA, to the LOI and to the POI are displayed, in which case, the operator can decide which level of detail to look at while trying to understand the plans.



a. Display of the task plans in the ranking suggested by the strategic planner

b. Ranking of the task plans after alteration by the operator

Figure 67. Plan display for a single HAPS

On the mission planner display, a function is implemented for the operator to alter the ranking of the task plans (see arrows “up” and “down” on the left in Figure 67a). To recall, the ranking is essential for the order in which the task plans will be refined by the numeric flight path planner in the tactical planning level (see Figure 63). In this particular example, the operator decides to move “Plan 1” to the last on the list, resulting hence in the ranking displayed in Figure 67b.

7.2 Validation of the Planning Functions

The planning techniques developed for HAPS are model-based, i.e. by using the expert knowledge, the formal model of the system was developed, which is used to search for the optimal plan[36]. The results of the tests for efficiency of the planners were presented in each section, and some analysis of the results have helped to improve the performance of the planner, for instance, by introducing an external framework to call the numeric flight path planner iteratively (see Section 3.3.3.2), or by configuring the GA-guided hierarchical task planner properly for better and faster convergence (see Section 5.4).

Left to validate in this section is the executability, i.e. if the plans conform with reality and how well the planners can cope with the versatile real world.

The following subsection intends to:

1. validate the model of the HAPS platform dynamics and the model of the environment used in the flight path planner in Section 3 by checking the executability of the flight path plans;
2. analyse the ability of the GA in Section 5 to cope with the versatile real world despite the abstraction of information for faster and better[37] task planning.

7.2.1 Validation: Executability of the Flight Path Planner

The scenario used for the validation tests is as depicted in Figure 11, with mission elements placed in the South of Germany. The region was selected due the availability of historical weather data (useful to realistically simulate the environment and flight dynamics [Köhler et al., 2017a]).

7.2.1.1 External 6-DoF HAPS Simulator

To validate the generated paths, we use a six degrees of freedom (6-DoF) aircraft simulator provided by the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt, DLR) constructed based on a realistic HAPS model coupled with a four-dimensional flight controller [Müller et al., 2018]. The latter ensures that the aircraft is controlled in the vertical and lateral directions with respect to the reference flight path, as well as the airspeed are followed to keep track of the time of arrival at each point of the path.

[36] Although the planning techniques described in Section 3-5 do not guarantee optimality, they are guided by optimality by trying to decrease flight time, increase rewards etc.

[37] The task planner is also judged by how well the fitness value, or more specifically, the cumulative probabilistic rewards reflect the quality of the plan.

The weather forecast data used for planning as well as the nowcast data for the flight simulation are historical data from 27th June 2015. On that day, sunrise starts around 05:15 local time and by 07:00, the sun is shines above the horizon. Therefore, the mission flight starts at 07:00, but it would also be wise for the HAPS to fly to its first mission area slightly before.

The exploited weather data considered in the pre-mission flight path planning are the COSMO-DE wind data [Baldauf et al., 2011], the Cumulonimbus forecast predicted with fuzzy logic [Köhler et al., 2017b], additional data to highlight strong wind and turbulence zones [Köhler et al., 2017a].

7.2.1.2 Validation of the Flight Dynamics Model

The plan issuing from the flight path planner, as described in Section 3.3 is a sequence of time-stamped actions, which will then be used to predict with extrapolation the four-dimensional (space and time) flight path. To recapitulate, the HAPS dynamics, as well as the time varying environment, are modelled with PDDL+, a declarative language that allows to specify the dynamics and constraints characterizing complex hybrid control systems with ease. Flight plans, derived from PDDL+ formulation of non-linear, non-homogeneous dynamics constraints, as well as collision avoidance with mobile obstacles, can be calculated efficiently using ENHSP [Scala et al., 2016], a domain-independent hybrid planner, as an off-the-shelf planner. Albeit plans are generated on a more abstract model of the world, we show that these plans result executable when tested on a high-fidelity simulator.

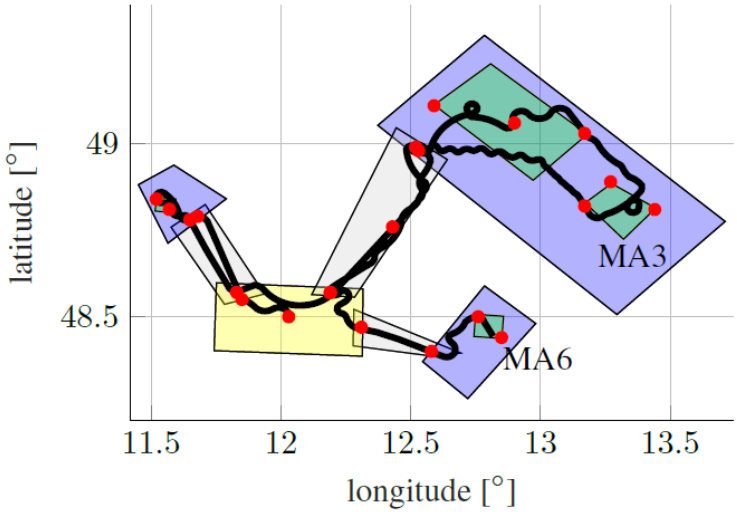


Figure 68. Typical flight path

A six-hour plan was computed by the mission planner within five minutes planning time with an Intel i7-6700K, 4GHz processor. The plan computed by the numeric flight path planner at the tactical planning level is a sequence of time-stamped actions. These are then used to predict by integrating over time the reference path. Figure 68 shows partially the reference path from 06:30am local Bavarian time until noon on the 27th June 2015, generated by the planner and the corresponding path flown by the HAPS simulator. In fact, if the forecasted weather is admissible, flight

paths that were successfully computed are feasible. The feasibility is measured by the deviation from the reference.

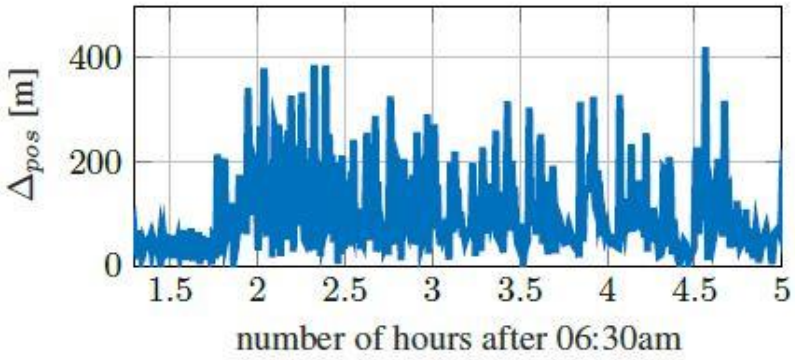


Figure 69. Lateral deviation in position between the planned path and the simulated flight path

Figure 69 shows the lateral deviation of a few hours of simulation data. The maximum deviation between the planned reference flight path and the simulated flight path is less than 420 m; simulation results also show that the mean lateral deviation between the planned reference flight path and the simulated flight path is about 143 m, which is acceptable for a HAPS [Müller et al., 2018], as the safety margin to any physical constraint (NoGo-areas of mission area) is set to at least 1 km, on account of the unforeseen movement or development of the weather situation, which is not considered in the coarsely discretized weather forecast (time interval between forecast sets is usually 1 hour, and can be also 3 hours) and cannot be properly extrapolation.

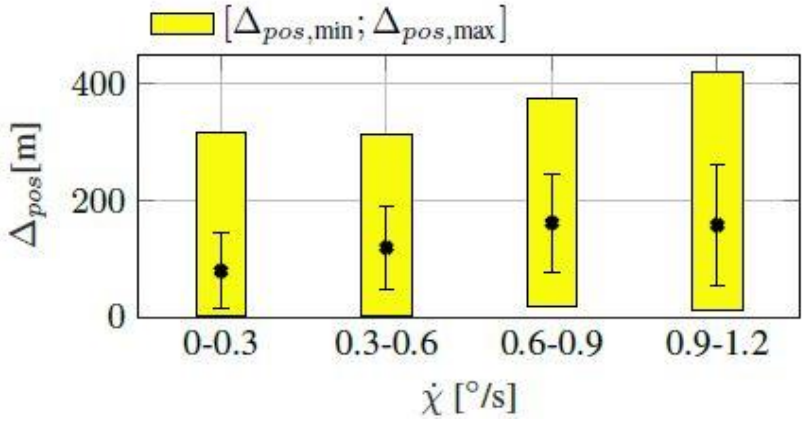


Figure 70. The relation between the lateral deviation in position and the turn rate of HAPS. The yellow bars indicate the range of the deviation, while the black error bars indicate the standard deviations with the cross marking the mean error.

Several factors could have caused the deviation from the reference path. The most obvious being

1. the different weather data used in the planner and in the simulator;
2. the integration step in the planner and the flight controller.

The first cause is unavoidable, which is also realistic, since the planning phase is carried out prior to mission execution. Therefore, only forecast data can be considered during planning, while the simulator uses the nowcast weather data. This conforms with the reality, in which the real weather situation can be different from the forecast.

The flight control operates at much higher frequency, e.g. 1000 Hz, while the integration step to predict the flight path out of the path plan is set to 1 s (same as the validation step set for the flight path planner, see Section 3.3.3.1). However, as observed in Figure 70, it is established that the greater the turn rate is, the harder it is to follow the planned path, due to the much larger integration time step used to generate the reference path, compared to the frequency of the flight controller. This also motivates an adaption of the planning model so that frequent turns will be penalized and avoided. This, however, is not as straightforward, since the planner's aim is to minimize travel time. Therefore, if turns are considered an additional cost, either it must be weighted appropriately, or a pareto-front optimization can be used. The inclusion of a penalty on turns in the flight path planning is left for future work.

Not only that the plans produced by the underlying flight path planner is executable, the model also comes with another advantage, namely less burden is exerted on the electro-motors. Figure 71 shows that the flight controller can follow the paths by maintaining an equivalent airspeed of around $9-10$ m/s, which is the optimal equivalent airspeed (see Section 2.1.1). It is hence more energy efficient and operationally safer since it is unlikely that the electro-motors are pushed to their power limit, while trying to follow the planned reference path.

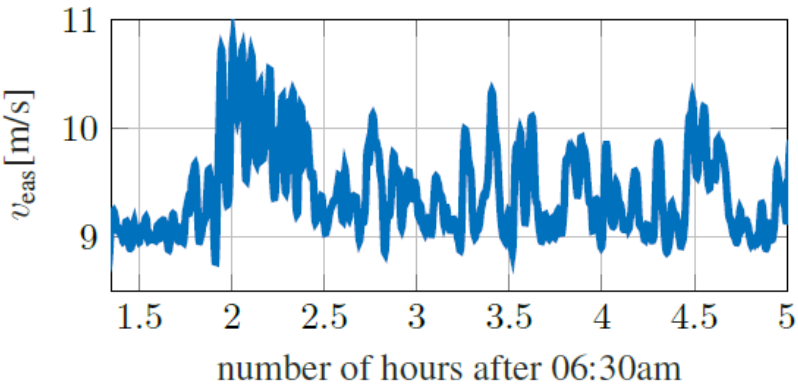


Figure 71. EAS during the test

7.2.2 Validation: Ability of the Task Planner to Cope with the Versatile Environment

In this section, the temporal hierarchical task planner described in Chapter 4 and the extension for multiple HAPS in Chapter 5 are tested, and the results are analysed with respect to the planner's ability to cope with the time-varying environment using the underlying probabilistic model. To put forth the benefit of our new approach, the

results are compared with tests conducted using several different scenarios (S1, S2, and S3) to be solved with different planners: planners using the GA guided hierarchical task planner or without guidance using a brute-force search. The tests are run on a 4.00 GHz x 8 Intel Core i7-6700K CPU with 32GB RAM, using a Matlab implementation of the GA, except for the integral operations required in Equation 4-5, that are implemented in C.

All scenarios under test share the layout of MAs, WAs, LoIs and Cs depicted in Figure 11, which comprises an area of 500 km × 500 km. LoIs and MAs dimensions, respectively; the mission elements have orders of magnitude of tens or of hundreds of kilometers as summarized in Table 7.

Scenario 1 and 2 (S1 & S2) use the same weather forecast, taken from the National Meteorological Service[38] at 05:00 local time on the 24th April of 2018[39]. Scenario 3 (S3) uses synthetic weather data to test the performance and convergence speed of the GA-guided planner on cloudy days (where the cloud coverage is higher than 50% for almost 50% time of the day). Both forecasts are summarized in timeline diagrams for each MA the plan horizon (08:00-16:00), as shown in Figure 72. The mission flight starts at 08:00 local time instead, since the sunrise hour on the 24th April 2018 was around 06:10[40]. The blue-gray shade each hour indicates the mean cloud coverage: the darker the shade is, the more substantial the cloud coverage is. The red dots indicate the hours within which the mean wind magnitude is greater than 3 m/s.

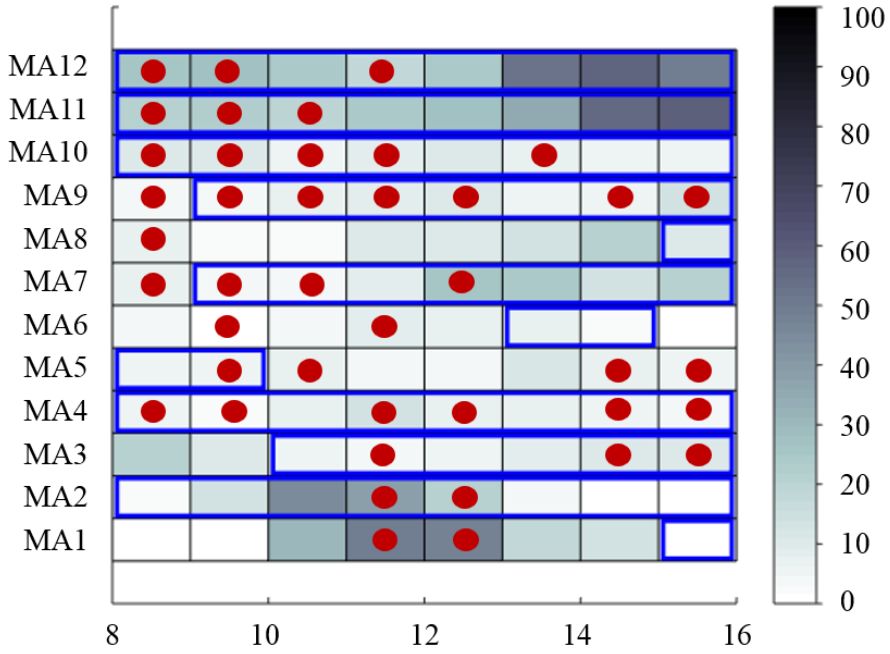
The blue (dark or light) rectangular edges denote the time windows at which the LoIs of the respective MAs are requested by the clients to be monitored. Besides, the MA with names in black encompass the LoIs that can be visited only once during the day; more visits will not be rewarded. MA with names in red encompass LoIs that can be visited as frequently as possible. The minimum time lapse between two visits to the same MA is set to two hours.

Finally, in S1 only one HAPS is contracted to carry out the mission, while in S2 and S3 a second HAPS is incorporated, making the combinatorial problem at the highest decomposition level exponentially more complex and activating hence the constraint that prohibits the coexistence of multiple HAPS in the same MA (see Section 5.3).

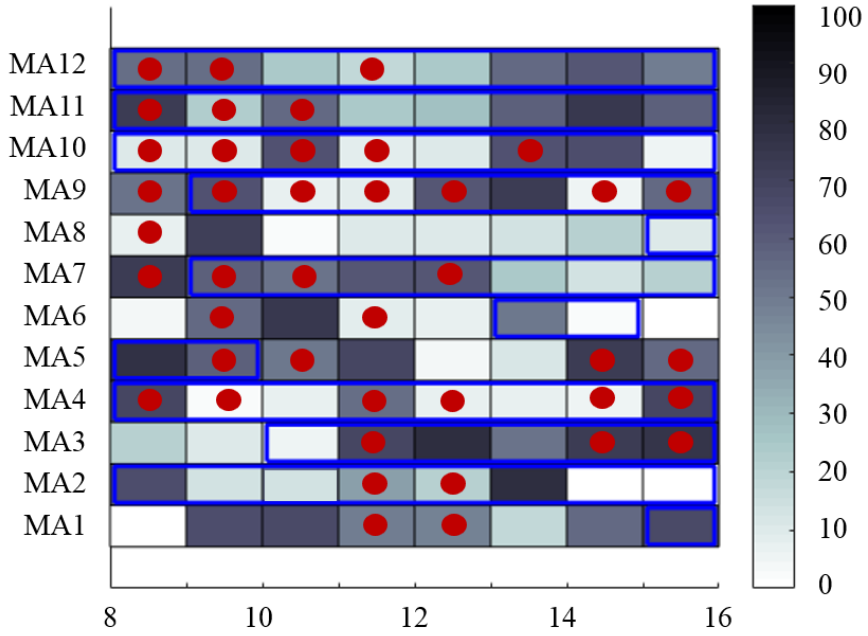
[38] Weather Data. 2018. German National Weather Service. https://www.dwd.de/EN. (2018).

[39] A different day is selected for the test since the 27th June 2015 was in general very cloudy, with a cloud coverage of above 90% most time of the day, and therefore is not beneficial for the test on the ability of the planner to cope with versatile weather.

[40] Mission operation commences when the sun shines on the solar panels of the stratospheric platform.



a. Historical cloud coverage data used for S1 and S2



b. Synthetical generated cloudy scenario used for S3

Figure 72. Cloud coverage used for the tests

If Equation 4-5 is used to evaluate the expected cumulative reward of a task plan, the likelihood of a successful and a failed monitoring task at a mission area are given as follows:

$$L\left(\text{succ}\middle|s_i, t_i, o_i^{\text{MA}}, w_{t_i}\right) = \begin{cases} 0.9, & w_{t_i}(cc) < th_{image} \\ 0.3, & \text{otherwise} \end{cases}, \qquad 7\text{-}1$$

$$L\left(\text{fail}\middle|s_i, t_i, o_i^{\text{MA}}, w_{t_i}\right) = \begin{cases} 0.1, & w_{t_i}(cc) < th_{image} \\ 0.7, & \text{otherwise} \end{cases},$$

where $w_{t_i}(cc)$ is the cloud coverage in percentage at t_i and th_{image} is the threshold in percentage for the recorded image coverage of all LoI of the MA (see Table 10), in order to be rewarded.

7.2.2.1 Planner Configurations

Four planner configurations are used to perform the tests. The first configuration (P1) uses the GA described in Section 5.3 to guide the search for the optimal decomposition at the mission level of the temporal HTN planner. As the performance of the GA depends on its parameter settings, the parameters summarized in Table 18 were chosen after determining statistically the best configuration for several scenarios, as described in Section 5.4.

Table 18. Parameterization of the GA planner

Iterations	50
Population size	50
Tournament size	3
Crossover, P_{xover}	0.9
Mutation, P_{mut}	0.05
Stochastic ranking swapping, P_{swap}	0.2
Duplicate handling	$P_{\text{mut}} = 0.2$ for duplicates
Objective function addition weights:	
w_{rew}	0.5
w_{div}	0.3
w_{eff}	0.2

In the second planner configuration (P2), the decomposition at the highest level of the temporal HTN planner is performed by brute-force search. Not only is the purpose of this configuration to show the computational benefits of using a GA to optimize the task decomposition, but also to determine the optimal solution and check if P1 converges towards it.

The third and fourth planner configurations (P3 and P4) substitute the probabilistic cumulative reward in Equation 4-5 used in the reward objective criterion 5-1 by the cumulative reward presented in Equation 7-1:

$$E\left(\Sigma\middle|s_0^h, t_0^h, \pi_{n,0}^h\right) = \sum_{a_i \in \pi_{n,0}^h} L\left(\mu = \text{succ}\middle|s_i^h, t_i^{h,\text{det}}, a_i^h, w_{t_i}\right) \cdot R\left(\mu = \text{succ}, a_i^h, t_i^{h,\text{det}}, t_{i+1}^{h,\text{det}}\right), \qquad 7\text{-}2$$

where the ending time $t_i^{h,\mathrm{det}}$ for each action is considered deterministic, i.e. the uncertainty of the execution time or duration of each task arising from the abstraction of the wind data and the platform dynamics in the task planner is neglected.

In short, and also summarized in Table 19, P1 and P2 consider the wind effects in the randomness of the ending times of the high-level actions (see Figure 40 for the distribution of the ending times), while P3 and P4 use a deterministic ending time obtained ignoring the variability caused by the wind. Finally, P3 is similar to P1 (i.e. both use the GA described in Section 3.2 to guide the decomposition at the highest mission levels), while P4 is similar to P2 (i.e. both use brute-force search). Hence, the solution by P4 also serves as the reference solution for P3, as the solution of P2 does it for P1.

Table 19. Configuration of planners

	Search for optimal decomposition	Estimation of task durations	Optimal?
P1	GA	probabilistic	Not guaranteed
P2	Brute-force	probabilistic	Yes
P3	GA	deterministic	Not guaranteed
P4	Brute-force	deterministic	Yes

7.2.2.2 Results and Analysis

For planners using the GA (i.e. P1 and P3), the tests were run for a total of 50 iterations, as indicated in Table 18. P1 and P3 clearly benefit from the fact that each GA iteration takes about 1-2 seconds and obtain their solution in less than 3 minutes after 50 iterations. The time used to obtain, by brute-force search, the optimal solution in P2 and P4 is scenario-dependent. For S1, the tests were conducted with only one HAPS, P2 took 2.43 hours while P4 took 2.55 hours. When two HAPS are involved, for instance in S2 and S3, P2 and P4 required 20.97 hours and 16.1 hours respectively. It is also worth noting that the brute-force search ran out of memory when two HAPS were involved, and, hence, read/write operations on the SSD hard-drive were required, contributing hence also to the longer computation time. From the computational point of view, the GA clearly accelerated the solution identification, or rather the search or optimal higher-level task decomposition within the HTN planner.

The planning for a single HAPS always find the optimal solution within the 30 first iterations, either with P1 (with the optimal solution of P2 found), or with P3 (with the optimal solution of P4 found).

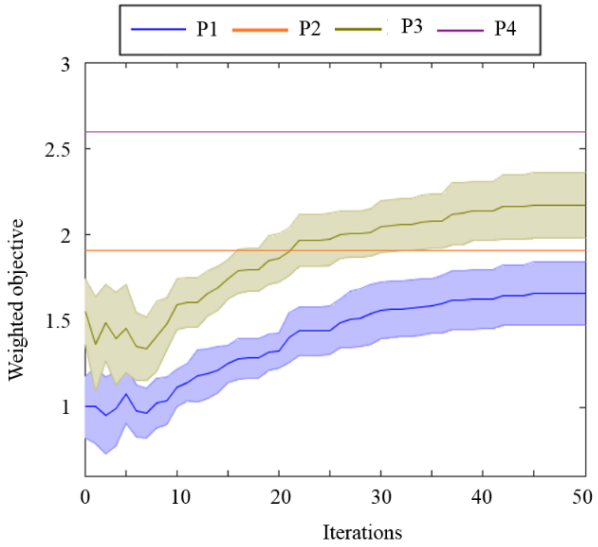
The performance and the plans by the four planner configurations for S2 and S3 are shown in Figure 73 and Figure 74 in the three tested scenarios S1, S2, and S3 respectively. Due to the randomness of the GA algorithm, a total of 20 tests were run for P1 and P3, in order to obtain statistical results. Figure 73a and Figure 74a show the evolution of the statistical results of the 20 test runs obtained by each GA planner over the number of iterations for P1 (in blue) and P3 (in green), and the maximum objective value found after the brute-force search by P2 (in orange) and P4 (in red). The mean is represented by the solid lines, while the standard deviation is represented

by the shaded areas of the weighted objective f of the best solution at each GA iteration. The objective values of P1 and P2 are lower than those of P3 and P4, which is normal, mainly due to the fact that the reward of each task in the plans of P1 and P2 only contributes probabilistically to the objective value (as described in Equation 4-5). Finally, although the GA planners are not always able to identify the best solution obtained by brute-force search, they are capable of obtaining a feasible overall-good solution in less than 3 minutes.

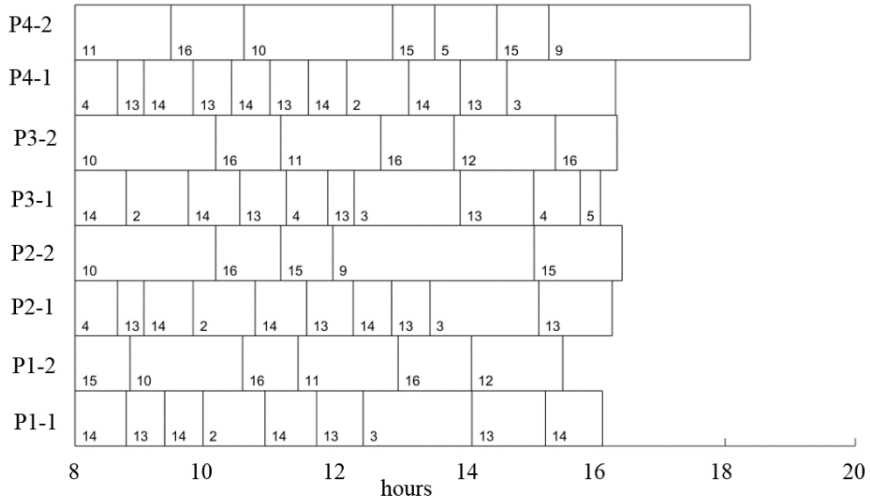
Figure 73b and Figure 74b show the plans at the MA-level found by each planner, with integer 1-12 being the MA and 13-16 being the WA 1-4. The labelling of the mission elements can be found in Figure 11. The task ordering is displayed while taking into account the median time of the actions predicted by the temporal HTN planner for P1 and P2, and the deterministic timing (without wind) for P3 and P4. Some plans exceed the end time of the planning horizon, e.g. 16:00, while others end clearly too early. This is due to the time-dependent crossover that is rounded up to the closest start time of a neighboring task of t_{xover}, resulting hence in varying plan lengths. The labels on the y-axis 'P#-$' identify the plans found using planner configuration P# for HAPS-$. The plans shown for planner P2 and P4 are simply the optimal ones obtained by brute-force according to their corresponding objective functions. For P1 and P3, the displayed plan is the best feasible plan found most often out of the 20 test runs. If several ones are found with the same frequency, in that case, the one with the highest objective value is shown. Note that the degree of similarity between plans found by P1 (or P3) and the optimal brute-force planner configuration P2 (or P4) indicates the degree of optimality of the GA.

The best plans shown in Figure 73b and Figure 74b are refined by the numeric flight path planner at the tactical level described in Section 3. The final refined plans decide if a task is successfully executed or not, in order to compare the plan quality of P1, P2, P3 and P4. Figure 73c and Figure 74c show the timing of the refined plans; MA that are successfully monitored are highlighted with a green dot, while those fail to be rewarded are highlighted with a red dot. In white are mission elements that correspond to WA, and therefore, are not rewarded.

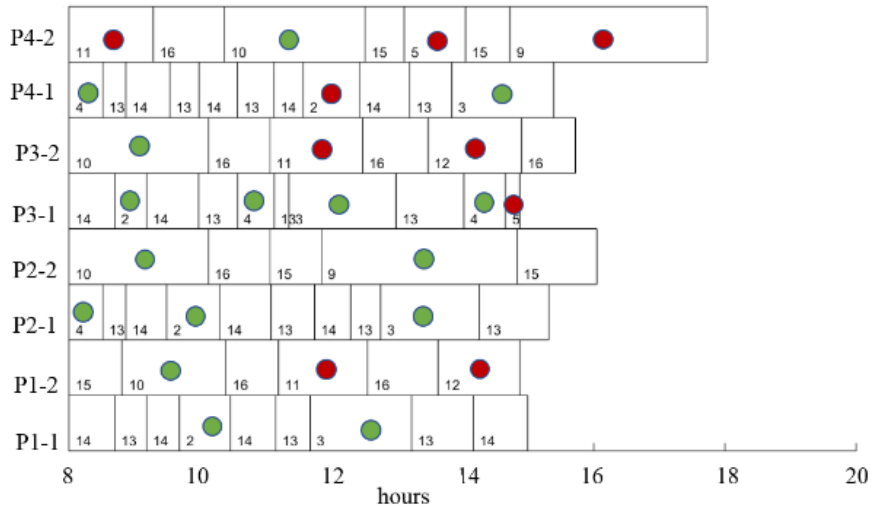
The results for S3 show especially the benefits of using a probabilistic arrival time (within Equation 4-5) in P1 and P2, versus using a deterministic arrival time (within Equation 7-1) in P3 and P4. The plans of HAPS-1 by P3 and P4 (P3-1 and P4-1) neglect the existence of the wind that could induce an error in the arrival times. This led to the unsuccessful monitoring of their initial mission areas (within its allowed time window) during the simulations. It is also worth noting that although not all MAs of the plans by P1 and P2 are successfully monitored, but failures occur later in the plans. This effect is expected in the plans by P1 an P2, which use Equation 4-5 to consider the distribution of each MA arrival time (spread wider in the far future MAs of the plan, as shown in Figure 40). In other words, the probabilistic execution time of each task used in Equation 4-5 ensures that, the further a MA is in the mission time, the less important its contribution to the probabilistic reward is, because the reliability of a near future task plan is more important than a far future one. This is a good planning strategy for HAPS operations, since re-planning is often required when new weather forecast update is available (e.g. typically hourly).



a. Convergence of objective value

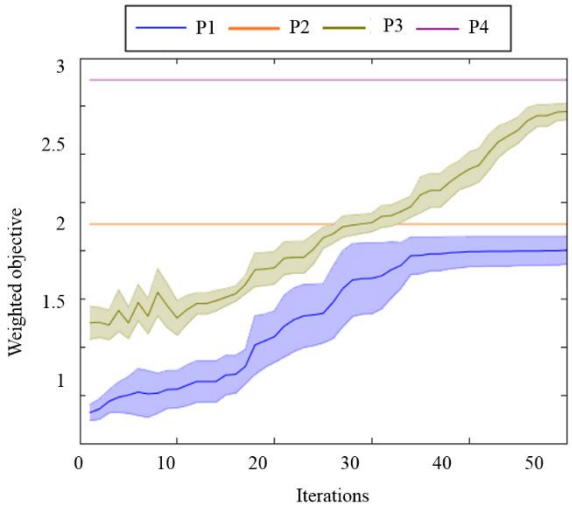


b. Obtained plan

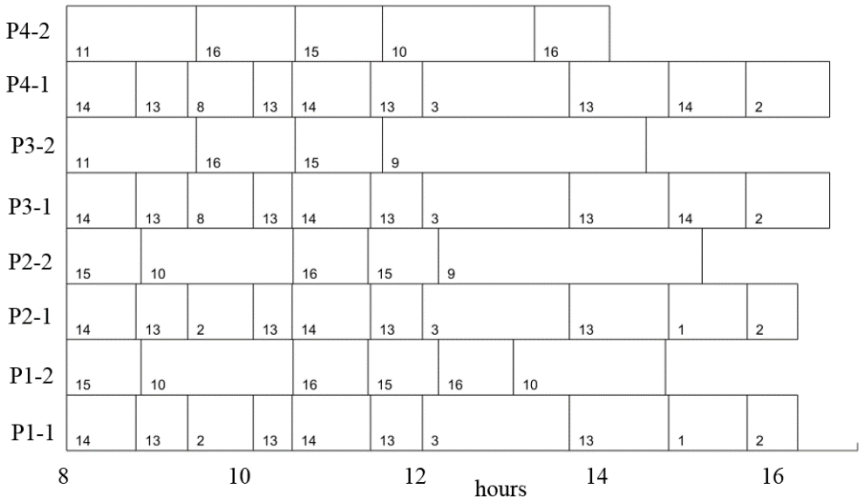


c. Plan execution timeline

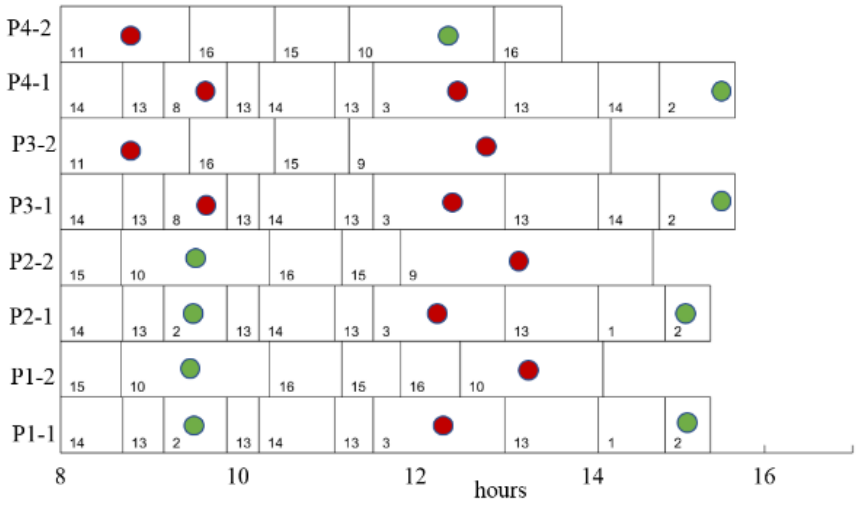
Figure 73. Benchmarking: Hierarchical Task Planning in Scenario 2



a. Convergence of objective value



b. Obtained plan



c. Plan execution timeline

Figure 74. Benchmarking: Hierarchical Task Planning in Scenario 3

Last, but not least, the increasing mean objective value over the iterations also indicate that P1 is more suitable to be implemented as an anytime planner (i.e. an implementation that provides a plan at any time instant), than by simply using a brute force search. A brute force search does not reflect on the quality of the found solutions with respect to search time, while with a GA-guided search like in P1 or P3, the task planner can be implemented as an anytime planner, whose solution improves with the increasing search time (i.e. more iterations).

8 Conclusion

In the previous chapter, the planning functionalities and the plan feasibility computed by the pre-execution mission planner were successfully tested and validated on the mission scenario described in Section 2.1 using a realistic 6-DoF HAPS simulator and historical weather data.

The pre-execution mission planner is developed as a two-tier planner: with a task planner working at the strategic level, and a numeric flight path planner working at the tactical level.

The task planner uses a HTN to structure the various tasks of HAPS missions, for practical reasons and for transparency. The HTN is practical because mission constraints and requirements expressed at different abstraction levels can be directly included into the HTN (see Section 4.3). It is transparent because the plans issuing from the task planner can be understood at different abstraction levels too by the human operator (see Figure 67). A very important aspect of the task planner is that the HTN is temporal, allowing hence:

1. to keep track of task execution times, which is essential for a dynamic environment,
2. task concurrency, which is important when multiple agents are involved.

Although abstracting information during planning, the task planner tries to take into account the uncertainties arising from the abstraction in the search of good plans using the probabilistic cumulative reward in the objective evaluation adopted by the GA (see Section 5.3). Furthermore, by using the GA to guide the search for optimal plan, the unsolved combinatorial problem subject to heterogeneous constraints in a HTN planner is surmounted.

The sorted task plans (with descending objective values) from the strategic planning level will then be presented to the HAPS operator, who can, as desired, alter the sorting order or not. Although not thoroughly studied, the potential of a human intervention at this level is highlighted, so that the pre-execution mission planner can, if necessary, also behave as a worker in collaboration with the operator in the `WSys` illustrated in Figure 13b.

The task plans will be “refined” by the tactical planner in the order they are sorted, which consists of a numeric flight path planner that can optimize the travel time (although optimality is not guaranteed) and better estimated the mission success rate, since the wind field, platform dynamics and dynamic obstacles are numerically considered too. At this level, a deterministic problem model is assumed. The flight path planning problem of HAPS is classified as a kinodynamic path planning problem; a PDDL AI planner is used for the first time to solve this class of problem. The planning performance is competitive with the RRT planner from OMPL.

A reactive avoidance strategy HFAS found by solving an MDP is also proposed. HFAS is high fidelity, as it provides a guidance to the HAPS to avoid sporadic obstacles on the way, while trying to find its way back to the reference plan. HFAS shall be triggered in the presence of scarce unforeseen obstacles during plan execution, i.e. maximum in the presence of two unforeseen obstacles. By doing so, the plan can be repaired, and a re-planning can be avoided. The strategy is computed by the mission planner on the GCS, due to its computational complexity. The deployment of it is done onboard, together with plan monitoring and flight control and is therefore out of the scope of this work.

Although works carried out are based on HAPS in a time-varying environment, without the loss of generality, the developed methods can also be adapted and applied on other moving agents in a dynamic environment (see example applications briefly discussed in Section 8.3).

8.1 Future Improvements on the Mission Planning for HAPS

The planner proposed in this work, from the modelling of the problem to the solving of it, is only one possible solution. The individual modules can be improved for more efficiency and better-quality plans. Additionally, the structure of the planner, i.e. how the modules are coupled with each other, can also be altered. The following lists a number of possible improvements and alterations to test for the HAPS pre-execution mission planning problem.

1. The reactive avoidance method developed in Section 6.2.1 does not consider the velocity of the obstacles to avoid in the model. This is acceptable for the case of HAPS, although the model is not rigorous. However, if the obstacle is a fast-moving object, e.g. another aircraft, the consideration of the velocity of the obstacles might be necessary, which can increase the computational complexity and leads to intractability. Clustering techniques can be applied to reduce the state space of the MDP.
2. In this work, the number of HAPS available to fulfill tasks of a mission is fixed. A method to determine the minimal number of HAPS (fixed in the current version of the planner) required to fulfill the mission can be developed. This is especially useful for resource management during operation.
3. The GA in Section 5.3 can be accelerated using parallelization techniques, since many steps in the GA are independent among individuals.
4. The weighted objective functions used in the GA as summarized in Table 18 can be optimized in a pareto-front fashion; the benefits against the current straightforward weight function can be analysed.

5. As already mentioned in Section 7.2.1.2, frequent turns can be considered as additional costs in the flight path planning. A balance between this cost and the cost on travel time must be found either by introducing an appropriate weighting function or by using a pareto-front optimization method.
6. The human-machine teaming can be studied more systematically in order to know at which stage the human operator should intervene, but at the same time is not overloaded, especially when he/she has to control multiple HAPS. The level of automation of the mission planner must then be adapted according to its role as a worker or a tool in the `WSys` (see Figure 13).
7. Although out of the scope of this work, the plan execution and plan monitoring for HAPS must be studied carefully in the future; by doing so, the needs to adapt the underlying methods in the mission planner for more efficient plan repair can be identified.

8.2 Lessons Learnt in AI Planning for Real-World Applications

Mission planning methods used for HAPS in this work are based on AI tools and optimization approaches. Some tools like the PDDL planner can be used off-the-shelf for flight path planning with minimal adaption on the implementation, while others are deemed unusable and customized planners have to be developed for this specific application, like the temporal HTN planner in Chapter 4 and 5. A few observations were made during this work and could be further developed and implemented for domain-independent AI planners.

1. A more general framework for the HTN can be developed with a modelling language that allows the formulation of realistic planning problem in a time-varying environment of such class. The domain-independent decomposition functions applicable to any task planning problem defined in this framework in Section 4.3.1 can be studied more in-depth, in order to be generalized to any HTN. Furthermore, a cascaded inter-level decomposition theory is missing and must be conceived for the generalization of HTN decomposition methods. As already pointed out in Section 4.3.1, the decomposition of a non-primitive task using successive methods is not studied here. This must be developed in order to generalize the decomposition methods of HTN.
2. An attempt to integrate the GA into a domain-independent temporal HTN to solve the combinatorial problem in the task decomposition can be pursued.
3. Although a complete task+motion planning in a tightly-coupled fashion using a PDDL+ planner is not feasible for the lack of scalability, as shown in Section 3.3.4.1, the degree of “tightly-coupled”-ness can be improved, by including also some task planning in the motion planning problem modelling with PDDL+. A tightly-coupled task+motion planning can guarantee coherence, since the goal is unified. Furthermore, the use of a PDDL+ planner is advantageous for plan explainability.
4. The temporal sequence diagram in Section 2.2.4.1 is developed for a deterministic temporal (semi)-automated system. It is worth noting that there is a lack of representation for probabilistic or fuzzy information signals or timeline. For example, how much time the operator needs to respond to the system, or if he/she responds, must be represented.

8.3 Reusability of the Mission Planning Methods on Other Applications

The methods described in this work can be adapted to suit other applications in dynamic environments, specifically applications which require a centralized mission planner at the command and control center. The following is a non-exhaustive list of descriptions of possible applications.

1. First responders for emergencies must often cooperate between different organizations to cope with the highly dynamic situations, especially in case of a terrorist attack, earthquake, etc. Very often, the success rate depends highly on the time of response and efficient deployment of resources. A centralized mission planning can improve the cooperation between various first responder teams and optimize the positive outcomes while being conform with protocols and regulations by using a temporal HTN planner. It can also minimize the travel time of first responders to the scene (e.g. which route an ambulance should take to avoid heavy traffic) by using an appropriate numeric path planner. Increasing the level of autonomy can help operators at the command and control center to respond and react appropriately, even under stress. However, open-loop actions must be considered in the planning, since many the situations can often only be partially known or even unknown to the planner. Open-loop actions allow the on-site first responders to react to unforeseen critical situations, while only minimal alteration is needed for the remaining plan.
2. In order to increase safety and reduce damage, modern manned fighter planes are often accompanied by fighter Unmanned Combat Aerial Vehicles (UCAV), so that the latter can disable the Surface-to-Air Missile (SAM) sites, before the fighter plan approaches its target. The success rate of disabling SAM sites can only be modelled probabilistically, while the actions of the fighter planes to decide by the command center must react accordingly. The environment can be considered dynamic, by not with a continuous process of updated state parameters, but with the presence of probabilistic events during the operation. The computation of a strategy similar to the reactive planning described in Chapter 6 can be extended. However, in the highly reactive planning problem, each UCAV may have to decide for its action. In order to ensure tractability while still keeping an eye on the coordination of multiple vehicles, a decentralized multi-agent MDP may be used instead.

Appendix 1: Monocular Camera

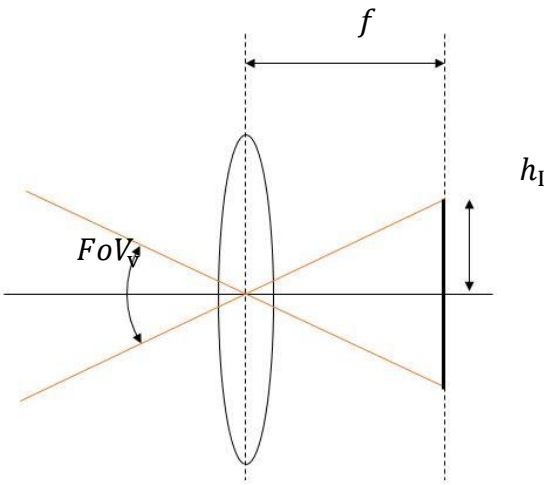


Figure 75. The relation between the field of view and the focal length of a pinhole camera

The horizontal field-of-view, FoV_{H}, is given by

$$FoV_{\mathrm{H}} = 2 \cdot \mathrm{atan}\left(\frac{0.5\ h_{\mathrm{I}}}{f}\right). \qquad \text{A1-1}$$

Similarly, the vertical field-of-view, FoV_{V} can be determined too.

Appendix 2: Coefficients of a Line Segment

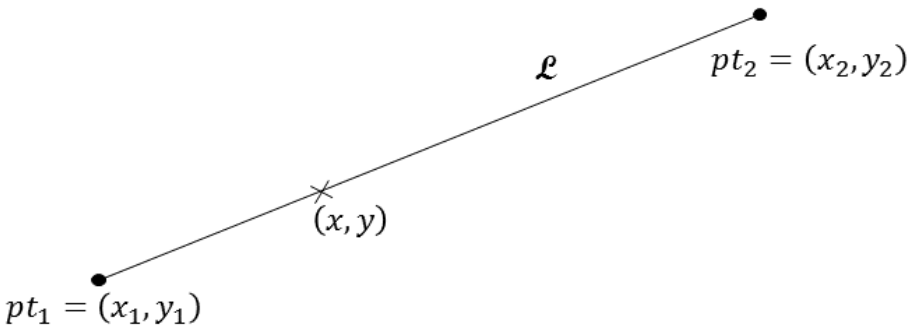


Figure 76. Line segment defined by pt_1 and pt_2

Let (x, y) be an arbitrary point lying on the non-vertical line segment $\mathcal{L}$ connecting pt_1 and pt_2. The gradient of $\mathcal{L}$ can be expressed by

$$m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{y_2 - y}{x_2 - x}. \qquad \text{A2-1}$$

We obtain thus

$$(x_2 - x)(y_2 - y_1) = (y_2 - y)(x_2 - x_1), \qquad \text{A2-2}$$

and by expanding, we obtain

$$\underbrace{(y_1 - y_2)}_{a} x + \underbrace{(x_2 - x_1)}_{b} y = x_2 y_1 - x_1 y_2 \qquad \text{A2-3}$$
$$= x_2 y_1 - x_1 y_1 + x_1 y_1 - x_1 y_2$$
$$= b y_1 + a x_1.$$

Appendix 3: Formulation in PDDL+

In the following, a fairly simple example of deterministic planning for an automated tap control of an irrigation system is illustrated to show a few key formulations offered by PDDL+ [Fox and Long, 2006]. In this example, the control of the tap of the water reservoir for the irrigation system installed at each farm on remote lands (without utility infrastructure) distributed over Germany is automated (see Figure 77). The operator of the remote farming is supposed to deliver water to refill the water reservoirs to ensure that they are never empty.

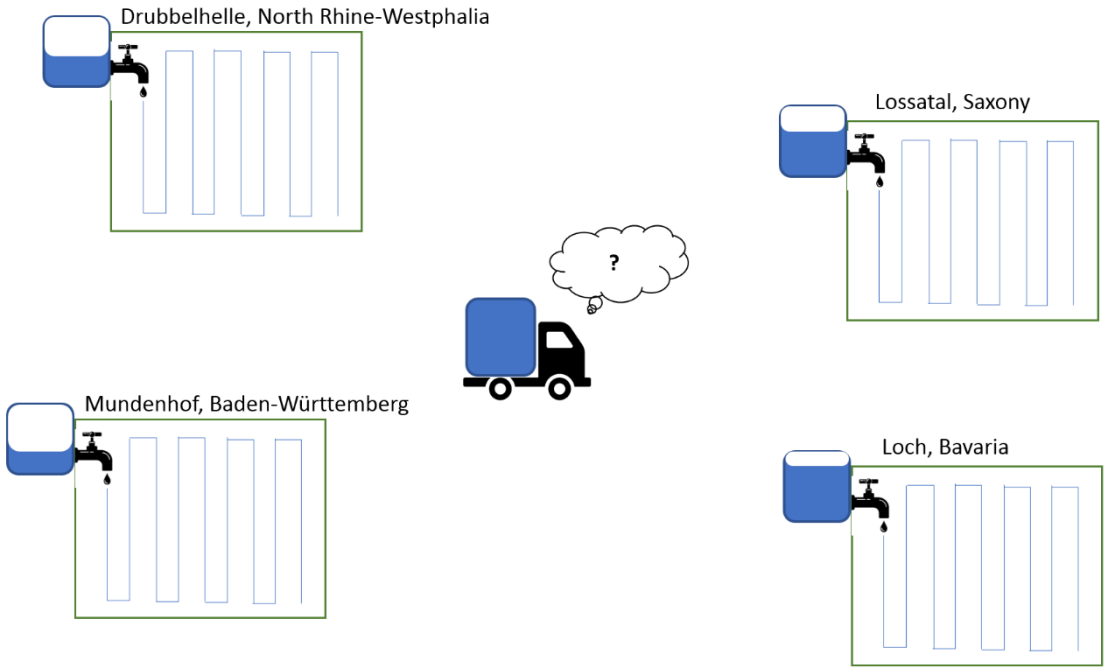


Figure 77. Automated tap control for farm irrigation systems distributed over Germany

However, in order to know the delivery schedule in advance, some planning must be done with respect to the rain forecast. If there is no rain for a long period, the soil is dry, and the water tap must be opened to irrigate. This is a hybrid planning problem with temporal and numeric factors (with only linear equations involved), as well as mixed discrete-continuous elements. The following excerpts show how PDDL+ can be used to model the planning problem using actions, events, processes and durative

actions. These formulations will all be included in the “domain” file. Note that the formulations in PDDL are declarative, declaring first the action or operator, followed by the attribute and a numeric parameter, if the operation is numeric.

1. Action

The action `open-tap` is to be decided by the planner and executed independently by each automated tap mounted on the water reservoirs of the remote garden boxes. The independent control is ensured by the object in the parameter formulation that will be instantiated in the problem instance definition, which is comparable to object-oriented programing. The action is to be performed when the soil is dry. After the action is performed, the effects will be “added”. The counters for the rain or drought duration are reset.

```
(:action open-tap
 :parameters(?garden-box -location)
 :preconditions(dry ?garden-box)
 :effect (and
  (tap-opened ?garden-box)
  (assign (duration-no-rain ?garden-box) 0)
  (assign (duration-rain ?garden-box) 0)
  (assign (duration-irrigation ?garden-box) 0))
 )
```

The action `close-tap` is performed once the irrigation duration has achieved `max-irrigation`. The state of `tap-opened`, as well as the `dry` state are then deleted.

```
(:action close-tap
 :parameters(?garden-box -location)
 :preconditions(and
  (tap-opened ?garden-box)
  (>= (duration-irrigation ?garden-box)
      max-irrigation)
 :effect (and
  (not (tap-opened ?garden-box)
  (not (dry ?garden-box)))
)
```

2. Event

Events are dynamic happenings that are instantaneous. When it has not rained for a long period (`max-no-rain`), the state `dry` is added for the garden box.

```
(:event set-dry
 :parameters(?garden-box -location)
 :preconditions(> (duration-no-rain ?garden-box)
                  max-no-rain)
```

```
   :effect((dry ?garden-box))
  )
```

Similarly, if it has rained for some time, i.e. `min-rain`, the `dry` state is deleted and the counters for rain or drought durations are reset, just like after the irrigation.

```
  (:event set-wet
   :parameters(?garden-box ?location)
   :preconditions(> (duration-rain ?garden-box)
                    min-rain)
   :effect (and
    (not (dry ?garden-box))
    (assign (duration-no-rain ?garden-box) 0)
    (assign (duration-rain ?garden-box) 0))
  )
```

3. Process

Processes are also dynamic happenings, but unlike events, they are continuous, i.e. they induce continuously a state parameter change in the system, as long as the preconditions are met. The counter `duration-no-rain for each garden box` is increased at each time step by `#t` when the rain forecast predicts dry weather. It is worth noting that PDDL+ has automated temporal planning functions; therefore, a discrete time step `#t` is incorporated.

```
  (:process increase-duration-no-rain
   :parameters (?garden-box -location)
   :precondition (not (raining ?garden-box ?time))
   :effect(increase (duration-no-rain ?garden-box) #t)
  )
```

Likewise, when rain is forecasted, the counter `duration-rain` is also increased stepwise continuously by #t over the forecasted rain period.

```
  (:process increase-duration-rain
   :parameters (?garden-box -location)
   :precondition (raining ?garden-box ?time)
   :effect (increase (duration-rain ?garden-box) #t)
  )
```

The planner keeps track of the time counter that starts from the beginning of the plan. However, the system `time` can be difference; it must be updated so that the planner knows which line of weather forecast to read.

```
  (:process increase-plantation-time
   :parameters()
   :precondition ()
```

```
    :effect (increase time #t)
   )
```

4. Durative action

A durative action is similarly to a process with limited and known duration. It can also be formulated using a process by included the limited duration as a precondition. The formulation in form of a durative action of reducing the water volume by `delta-water-volume` at each time step in the reservoir over the irrigation period `max-irrigation` is shown here, while the irrigation duration is increased discretely and continuously by `#t.`

```
(:durative-action reduce-water-volume
 :parameters (?garden-box -location)
 :duration (= ?duration max-irrigation)
 :condition (and
  (> (water-volume ?garden-box) 0)
  (tap-opened ?garden-box))
 :effect (and
  (decrease water-volume (* #t delta-water-volume))
  (increase (duration-irrigation ?garden-box) #t))
)
```

"Les grandes personnes […] ont toujours besoin d'explications."
- Antoine Saint-Exupéry in *Le Petit Prince*

Appendix 4: A Preliminary Case Study for XAIP

Fox et al. [Fox et al., 2017] advocated for the benefit of using AI planners, for example a PDDL planner, as a model-based planning tools, given their ability to capture the causal and temporal relations at each step of the plan. A planning domain properly modelled does not only help the planner to plan efficiently and correctly, the model also serves as an explanatory tool for humans to understand the output of the effect of an action, thereby understanding why a plan is decided and why some plans are not executable. Planning for problems with more advanced numeric operations, such as path planning, is often beyond human cognitive capacity, since human brain works at a higher abstraction level, thereby leaving little space for human intervention. However, the more prevalent and intelligent automated systems become, the more inevitable it is to consider a Human-Autonomy Teaming (HAT) approach that allows the human to play more than just a supervisory role [Schulte and Donath, 2018]. Much attention has been attributed in the past years to Explainable AI Planning (XAIP) [Fox et al., 2017] in order for the human to understand, interact and trust the automated planner. Such a relation between human and machine is desired to make possible Mixed-Initiative Planning (MIP) [Kiam et al., 2019c].

It was argued in [Fox et al., 2017] that model-based AI-planners, such as domain-independent planners that understand PDDL, have a unique potential to contribute to XAIP. By modelling a problem as axioms using proper formalism, the model can be also used to validate a plan and communicate with the human operator. Described in the next subsection is a simple path planning example to put forth the advantage of being explainable using a model-based AI planer. Thanks to a plan validator, MIP in numeric path planning is made possible, which is conventionally non-trivial as most numeric path planning tools are exploited more like a "intelligent calculator" and the decision of plans is often deemed as beyond comprehension to the user.

Due to the fact that many available AI planning tools do not support formulations with trigonometric functions, a simple grid world example is chosen instead of taking the HAPS path planning problem as an example.

Robot & Frank

A scenario consisting of a housekeeping robot assigned by his master, Frank, to collect the garbage in the backyard is used to demonstrate the possibility of a MIP for numeric path planning using a model-based planning approach.

The initial setup is illustrated in Figure 78, where objects r represent the robot, positioned initially at (x, y) = (100, 150) and g the garbage bin at (250, 300). The robot can move incrementally in x- and y-directions (i.e. linear movements) and are subject to dynamics constraints, which can be simply expressed as $|v_x|$ and $|v_y| < 5$, with $|v_x|$ and $|v_y|$ being the speed in x- and y- directions. The red lines represent tree rows that the robot cannot traverse, and the blue circle the mobile water jet, w, that will move across the garden along the dashed lines at a velocity of (2 m/s, 2 m/s). Any collision of the robots with the water jets or the tree must be avoided.

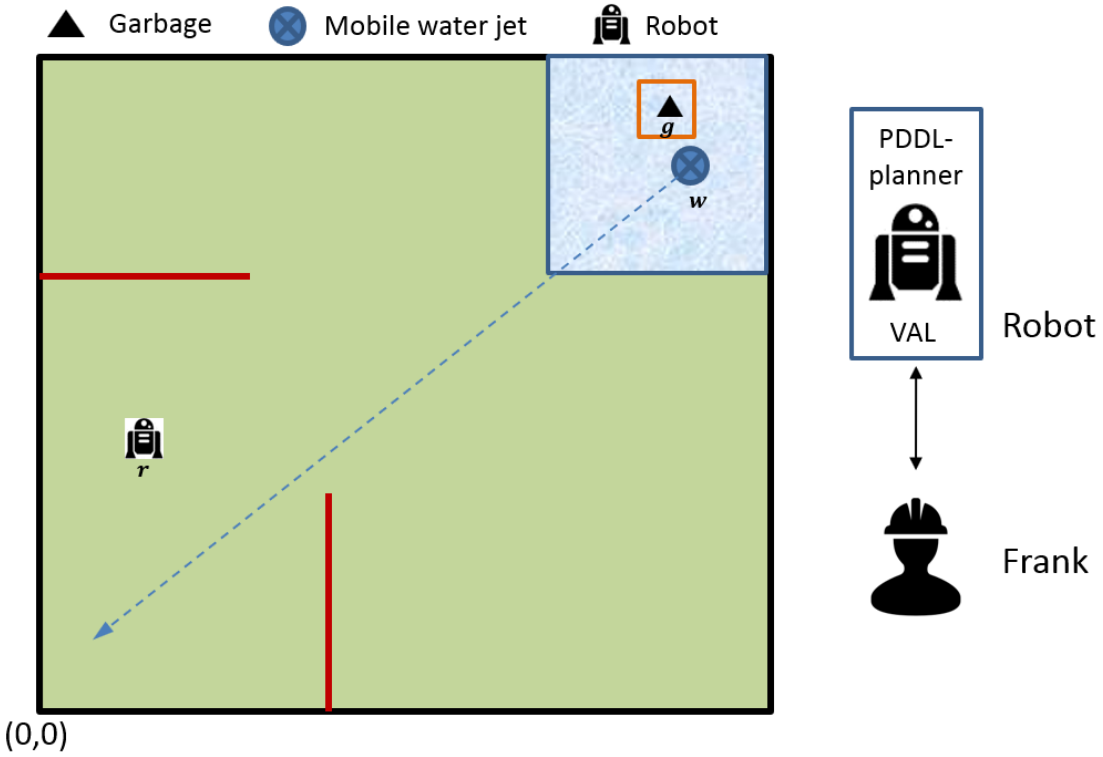


Figure 78. Simple MIP example: Robot & Frank

Additionally, any object within the light blue shaded areas (20 m × 20 m) around the water jet will get wet. It is hence forbidden for the robot to open the lid of the garbage bin g to collect the garbage if it is placed within the shaded blue area. Lastly, the robot must be in the vicinity of the garbage bin, as enclosed by the orange square (2 m × 2 m), to be able to reach for the lid and collect the garbage.

Planning

Processes to increase the position and velocity of the robot in x- and y- directions are similar to those described in Table 11 to update the heading rate and position of the HAPS. A few events are necessary in this example to model the garbage getting wet if the lid is open while the water jet is in its vicinity, or the contrary in the case where the water jet is far away from the garbage bin. Snippets of the events are presented in Figure 79.

```
  (:event garbageStaysDry
   :parameters ( )
   :precondition (and (not (permissionToCollect))
                     (or (> (/ (+ (edgeMinXDo) (edgeMaxXDo))
2)
                             (compoundWetGarbageMaxX))
                     (< (/ (+ (edgeMinXDo) (edgeMaxXDo)) 2)
                              (compoundWetGarbageMinX))
                     (> (/ (+ (edgeMinYDo) (edgeMaxYDo)) 2)
                             (compoundWetGarbageMaxY))
                     (< (/ (+ (edgeMinYDo) (edgeMaxYDo)) 2)
                             (compoundWetGarbageMinY))))
  :effect (permissionToCollect))

  (:event garbageGetWet
   :parameters ( )
   :precondition (and (permissionToCollect)
              (and (< (/ (+ (edgeMinXDo) (edgeMaxXDo)) 2)
                             (compoundWetGarbageMaxX))
                     (> (/ (+ (edgeMinXDo) (edgeMaxXDo)) 2)
                             (compoundWetGarbageMinX))
                     (< (/ (+ (edgeMinYDo) (edgeMaxYDo)) 2)
                             (compoundWetGarbageMaxY))
                     (> (/ (+ (edgeMinYDo) (edgeMaxYDo)) 2)
                             (compoundWetGarbageMinY))))
  :effect (not (permissionToCollect)))
```

Figure 79. PDDL+ snippets to formulate the events of the garbage staying dry or wet

The robot uses its integrated PDDL-based AI planner (ENHSP in this example) to compute a valid temporal plan. The plan trace is shown in Table 20. A plan trace consists of a list of time-stamped actions to be executed.

MIP

However, Frank is not happy with the plan as he thinks that it is disturbing to have the robot moving too fast in the household. Subsequently, he tries to negotiate that the robot does not move faster than 4.2 m/s in each direction, and thereby requests that the robot stops accelerating after time instant $t = 38$ s (see **MIP attempt 1** of

Table 20), by truncating all actions to increase velocity after $t = 38$ s, leaving however the goal action `collectGarbage` to still be executed at $t = 48$ s.

Equipped with the plan validator VAL, the robot checks the plan proposed by Frank. VAL [Howey et al., 2004] is a plan validator for problems modelled using PDDL[41]. The purpose of the plan validator is not only to check the numeric correctness of a plan[42], but can also explain why a plan fails by exhibiting the cause of failure ("Plan Repair Advice") and by advising a fix ("Follow each of").

In **MIP attempt 1**, VAL produces an error message explaining that the robot has not reached the garbage can yet at $t = 48$ s, if the robot does not continue increasing its speed after $t = 38$ s. Frank subsequently negotiates to delay the action to collect the garbage in **MIP attempt 2**. However, the plan still fails and, as indicated by VAL, the robot does not have the permission to collect the garbage, since the garbage bin is found too near (i.e. within the shaded area) to the mobile water jet -- the garbage could be wet if the robot attempts to open the lid. In **MIP attempt 3**, Frank delays the action to collect the garbage to allow enough time for the water jet to move away from the garbage bin. The plan can be successfully executed, although there is room for improvement, since the event `garbagestaysdry` is triggered at t =173.5 s, and that the robot has the `permissiontocollect,` while the robot collects the garbage almost 27 s later at $t = 200$ s. In his last **MIP attempt**, Frank sets the `collectGarbage` action to time instant 174 s.

Table 20. VAL used for mixed initiative planning

Plan trace	**Validator (VAL)**
0.00000: (moveRobot) 0.00000: (increaseVelX) 0.00000: (increaseVelY) ... 38.00000: (increaseVelX) 38.00000: (increaseVelY) ... 47.00000: (increaseVelX) 48.00000: (collectGarbage)	**Original plan:** … Checking next happening (time 38) Updating (velocityx) (4.2) by 0.1 increase Updating (velocityy) (4.2) by 0.1 increase ... Successful plans
0.00000: (moveRobot) 0.00000: (increaseVelX) 0.00000: (increaseVelY) ...	**MIP attempt 1:** Plan failed to execute Plan Repair Advice: (collectgarbage) has an unsatisfied precondition at time 48

[41] VAL, like most PDDL planners, does not support trigonometric functions. Hence it is not possible to test the plan validator against a more complex and realistic use case like the HAPS.

[42] If an external validator is used, the plan validation is even more objective, given that erros in the planner may not be recurring in the plan validator. Therefore, the integrity of the plan is increased.

38.00000: (increaseVelX) 38.00000: (increaseVelY) 48.00000: (collectGarbage)	(Follow each of: (Satisfy (positionx)[=235.8] > ((garbagepositionx)[=250] - (garbagemargin)[=10])) and (Satisfy (positiony)[=285.7] > ((garbagepositiony)[=300] - (garbagemargin)[=10])))
0.00000: (moveRobot) 0.00000: (increaseVelX) 0.00000: (increaseVelY) ... 38.00000: (increaseVelX) 38.00000: (increaseVelY) 50.00000: (collectGarbage)	**MIP attempt 2:** Plan failed to execute Plan Repair Advice: (collectgarbage) has an unsatisfied precondition at time 50 (Set (permissiontocollect) to true)
0.00000: (moveRobot) 0.00000: (increaseVelX) 0.00000: (increaseVelY) ... 50.00000: (waitForPermission) 200.00000: (collectGarbage)	**MIP attempt 3:** EVENT triggered at (time 173.5) Triggered event (garbagestaysdry) Adding (permissiontocollect) Checking next happening (time 200) Adding (robotstop) Adding (garbagecollected) Successful plans:
0.00000: (moveRobot) 0.00000: (increaseVelX) 0.00000: (increaseVelY) ... 38.00000: (increaseVelX) 38.00000: (increaseVelY) 50.00000: (waitForPermission) 174.00000: (collectGarbage)	**MIP attempt 4:** EVENT triggered at (time 173.5) Triggered event (garbagestaysdry) Adding (permissiontocollect) Checking next happening (time 174) Adding (robotstop) Adding (garbagecollected)

Appendix 5: Time-Dependent Markov Decision Process (TiMDP)

According to [Boyan and Littman, 2000], a TiMDP with the tuple $< X, S, T_n, M_\mu, L, R, K >$, where

- X, S are the continuous and discrete state space,
- T_n is the task name space,
- M_μ is the discrete set of outcomes $\mu =< succ, s'_\mu, T_\mu, P_\mu >$,with $succ$ representing the success ($succ \coloneqq true$) or failure ($succ \coloneqq false$) of the action that leads to the outcome μ, T_μ specifying the relative or absolute arrival time and P_μ the pdf over the relative or absolute arrival times,
- $L(\mu|x, s, t, a)$ is the likelihood of outcome $\mu \in M$ given $x \in X$, $s \in S$, $t \in T$, and $a \in A$,
- $R(\mu, t, \delta)$ is the reward for outcome μ at time t with duration δ.

Appendix 6: Constrained Optimization Problem

A system with n variables $\vec{x} = (x_1, \dots, x_n) \in \mathcal{S}$, where $\mathcal{S}$ denotes the search space, is subject to an objective function $f(\vec{x})$ to optimize and constraints $\mathcal{C}$ that are not to be violated, and thereby restraining the feasible space to $\mathcal{F} = \mathcal{S}$. Looking for the optimum in such a system can be treated as a constrained optimization problem.

It is not uncommon to solve the problem by introducing a penalty term $p(d(\vec{x}, \mathcal{C}))$ to form a fitness function combining the objective and the penalty functions:

$$\psi(\vec{x}) = f(\vec{x}) + p\big(d(\vec{x}, \mathcal{C})\big),$$

where $d(\vec{x}, \mathcal{C})$ can be understood as the metric distance to the infeasible regions and p the penalty function to avoid the search of optimum close to the infeasible regions. By doing so, the problem becomes an unconstrained one.

However, it is not always easy to define a suitable penalty function for the hard constraints [Runarsson and Yao, 2000], especially if the system is made up of a mixture of discrete and continuous variables of inhomogeneous nature. Many static, dynamic or even adaptive penalty functions have been developed to allow for different penalties throughout generations in the GA. All these methods require however extensive knowledge of the problem and if badly designed, can slow down the algorithm or even get trapped in local optima.

Abbreviations

ADS	Airbus Defence and Space
AFUA	Advanced Flexible Use of Airspace
ANML	Action Notation Modeling Language
AI	Artificial Intelligence
AIRMET	AIRman's METeorological information
API	Application Programming Interface
ATC	Air Traffic Control
ATM	Air Traffic Management
AUV	Autonomous Underwater Vehicle
BAF	Bundesaufsichtsamt für Flugsicherung
Cb	Cumulonimbus
CGAL	Computational Geometry Algorithms Library
COSMO-DE/COSMO-D2	COnsortium for Small-scale MOdeling
DGAC	French Directorate-General for Civil Aviation/ Direction Générale de l'Aviation Civile
DLR	Deutsches zentrum für Luft- und Raumfahrt
DoF	Degree of Freedom
EAS	Equivalent Air Speed
EASA	European Aviation Safety Agency
ECMWF	European Center for Medium-Range Weather Forecast
EHC	Enforced Hill Climbing
EO	Electro-Optical
ETA	Estimated Time of Arrival
EUROCONTROL	European Organisation for the Safety of the Air Navigation
FCS	Flight Control System
FL	Flight Level
FoV	Field of View
GA	Genetic Algorithm
GA	Genetic Algorithm
GAFOR	General Aviation FORecast
GAMET	General Aviation METeorological information
GCS	Ground Control Station
GFS	Global Forecast System

HALE	High-Altitude Long-Endurance
HAPPIEST	High-Altitude Pseudo-satellites: Proposal of Initiatives to Enhance SaTellite communication)
HAPS	High-Altitude Pseudo-Satellite
HAT	Human-Autonomy Teaming
HFAS	High-Fidelidy Avoidance Strategy
HFR	High-level Flight Rules
IPC	International Planning Competition
ICAO	International Civil Aviation Organization
IFR	Instrumental Flight Rules
IR	Infrared
LEO	Low Earth Orbit
MDP	Markov Decision Process/ Problem
METAR	METeorological Aerodrome Routine weather report
MIP	Mixed Initiative Planning
MIP	Mixed-Initiative Planning
MMS	Mission Management System
MOOP	Multi-Objective Optimization Problem
MP	Mission Planner
NCEP	National Center for Environmental Prediction
NOAA	National Oceanic and Atmospheric Administration
ODE	Ordinary Differential Equation
OMPL	Open Motion Planning Library
PDDL	Problem Domain Definition Language
PoI	Point of Interest
RGB	Red-Green-Blue (often used to represent the true-color palette)
ROS	Robot Operating System
RPA	Remotely-Piloted Aircraft
RRT	Rapidly exploring Random Tree
SIGMET	SIGnificant METeorological phenomena
SSSP	Single-Source Shortest Path
TAF	Terminal Aerodrome Forecast
TAS	True Air Speed
TC	Task Constraints
TiMDP	Time-dependent Markov Decision Process
TRL	Technology Readiness Level
UAS	Unmanned Aerial Systems
UI	User Interface
UML	Unified Modeling Language
VFR	Visual Flight Rules
WA*	Weighted A*
WP	WayPoint
XAIP	eXplainable AI Planning
WGS84	World Geodetic System 1984
ADS-B	Automatic Dependent Surveillance – Broadcast
HFR	High-level Flight Rules

Symbols

$\|S\|$	the cardinality or the size of the set S
A	set of actions
b	bearing between the course tangent and the start-goal vector
C	infeasible region
C_A	set of preconditions of the actions a ∈ A
C_M	set of mission requirements
C_X	constraints on the workspace
d	metric distance
d	decomposition function
δ	duration of a task
$dist_i$	distance between the HAPS and the barycenter of obstacle i
$dist_h$	distance between the HAPS and the next waypoint
F	feasible region
f	objective function
H	set of HAPS $h_1, \ldots, h_H$
h	a HAPS
$k_{tournament}$	size of the tournament selection
λ	longitude
n_*	number of *
ν	penalty
o	a task in the HTN
o_p	a primitive task of the HT
obs_i	i-th obstacle
p	position vector
$\{p\}$	reference trajectory obtained from the plan
P_{xover}	crossover probability in the genetic algorithm
$P_{mutation}$	mutation probability in the genetic algorithm
P_{swap}	swap probability in the stochastic ranking of the GA
φ	latitude
π	plan
$\tilde{\pi}$	task plan found at the strategic planning level

$\boldsymbol{\pi^h}$	plan for HAPS h
$\boldsymbol{\tilde{\pi}^{l,h}}$	task plan expressed at level l of the HTN for HAPS h
$\boldsymbol{\tilde{\pi}^h(i)}$	partial plan for HAPS h from time instant t_i
$\boldsymbol{\tilde{\pi}^h_{i:j}}$	partial plan for HAPS h from i-th task to j-th task
$\boldsymbol{\Psi}$	fitness function
$\boldsymbol{q}$	attitude vector
$\boldsymbol{r}$	reward mapping function
$\boldsymbol{S}$	search space
$\boldsymbol{s}$	signal
$\boldsymbol{t}$	time instant
$\boldsymbol{T}$	plan horizon $[t_{min}, t_{max}]$
$\boldsymbol{T}$	duration of a plan
$\boldsymbol{t_0}$	initial plan time
$\boldsymbol{\theta_i}$	relative bearing between the HAPS velocity vector and the vector connecting the HAPS to the barycenter of the obstacle i
$\boldsymbol{U}$	control space
$\boldsymbol{u}$	control parameter
$\boldsymbol{v}$	ground speed
$\boldsymbol{v_{EAS}}$	equivalent airspeed
$\boldsymbol{v_{TAS}}, \boldsymbol{v^*_{TAS}}$	true airspeed and optimal (*) true airspeed
$\boldsymbol{v_w}$	wind velocity
$\boldsymbol{\vec{x} = (x_1, \dots, x_n)}$	variables of a system written in vector
$\boldsymbol{X}$	workspace
$\boldsymbol{Z_i}$	set of measurements of the physical environment from sensor i
$\boldsymbol{z_h}$	altitude
$(\cdot)^T$	transpose of a vector/matrix

List of Figures

List of Tables

9 *References*

Airbus (2007) *'Adverse Weather Operations:* Optimum Use of the Weather Radar', Flight Operations Briefing Notes.

Airbus Defence and Space (2017a) *'Airbus Zephyr - Unique Contribution to Decision Superiority'*.

Airbus Defence and Space (2017b) *'Zephyr:* Focus of an aircraft. Endurance of a satellite.' [online] www.airbusdefenceandspace.com (Accessed 3rd March 2019).

Airbus S.A.S. (2005, Revised February 2019) *'A320: Aircraft Characteristics:* Airport and Maintenance Planning'.

Aldinger, J., Mattmüller, R. and Göbelbecker, M. (2015) *'Complexity of Interval Relaxed Numeric Planning'*, KI 2015: Advances in Artificial Intelligence, Dresden, Germany.

Allen, R. and Pavone, M. (2015) *'Toward a Real-Time Framework for Solving the Kinodynamic Motion Planning Problem'*, IEEE International Conference on Robotics and Automation, Seattle, Washington, USA.

Anderson, K.R. (1978) *'A Reevaluation of an Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set'*, Information Processing Letters, 7-1, pp.53–55.

Andrés-Toro, B., Girón-Sierra, J.M., Fernández-Blanco, P., López-Orozco, J.A. and Besada-Portas, E. (2004) *'Multiobjective Optimization and Multivariable Control of the Beer Fermentation Process with the Use of Evolutionary Algorithms'*, Journal of Zhejiang University SCIENCE, Vol. 5, No. 4, pp.378–389.

Andrews, D.G., Holton, J.R. and Leovy C. B. (1987) *'Middle Atmosphere Dynamics'*, International Geophysics Series, No. 40.

Antony, T., Amatya, S., Mastrogiovanni, F. and Baglietto, M. (2018) *'Task-Motion Planning in Belief Space'*, RSS Workshop on Exhibition and Benchmarking of Task and Motion Planners, Pittsburgh, Pennsylvania, USA.

Attmanspacher, J. (2019) *'Implementierung einer reaktiven Vermeidungsstrategie für HAPS'*, Bachelor's thesis, University of the Bundeswehr, Munich.

Baldauf, M., Seifert, A., Förstner, J., Majewski, D., Raschendorfer, M. and Reinhardt, T. (2011) *'Operational Convective-Scale Numerical Weather Prediction with the COSMO Model: Description and Sensitivities'*, Monthly Weather Review, Vol. 139, No. 12, pp.3887–3905.

Beard R. W. and McLain, T.W. (2012) *'Small Unmanned Aircraft:* Theory and Practice', Princeton University Press.

Beasley, D., Bull, D.R. and Martin, R.R. (1993) *'An Overview of Genetic Algorithms : Part 1, Fundamentals'*, University Computing, 15-2, pp.58–69.

Bedka, K., Brunner, J., Dworak, R., Feltz, W., Otkin, J. and Greenwald, T. (2010) *'Objective Satellite-Based Detection of Overshooting Tops Using Infrared Window Channel Brightness Temperature Gradients'*, Journal of Applied Meteorology and Climatology, Vol. 49, No. 2, pp.181–202.

Beihoff, B., Oster, C., Friedenthal, S., Paredis, C., Kemp, D., Stoewer, H., Nichols, D. and Wade, J. (2014) *'A world in motion:* Systems engineering vision 2025', International Council on Systems Engineering - INCOSE.

Benton, J., Smith, D., Kaneshige, J., Keely, L. and Stucky, T. (2018) *'CHAP-E: A Plan Execution Assistant for Pilots'*, Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS), Delft, The Netherlands.

Bernardini, S., Fox, M. and Long, D. (2014) *'Planning the Behaviour of Low-Cost Quadcopters for Surveillance Missions'*, Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, Portsmouth, New Hampshire, USA.

Bertsekas, D.P. and Tsitsiklis, J.N. (1996) *'Neuro-Dynamic Programming',* Athena Scientific, Belmont, Massachusetts.

Besada-Portas, E., La Torre, L. de, La Cruz, J.M. de and Andrés-Toro, B. de (2010) *'Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios'*, IEEE Transactions on Robotics, Vol. 26, No. 4, pp.619–634.

Blanning, R.W. (1981) *'Model-Based and Data-Based Planning Systems'*, OMEGA - The International Journal of Management Science, 9-2, pp.163–168.

Blum, A.L. and Furst, M.L. (1997) *'Fast Planning Through Planning Graph Analysis'*, Artificial Intelligence.

Boyan, J.A. and Littman, M.L. (2000) *'Exact Solutions to Time-Dependent MDPs'*, Proceedings of the 13th International Conference on Neural Information Processing Systems (NIPS), Denver, Colorado, USA.

Bradley, D.M. and Gupta, R.C. (2002) *'On the Distribution of the Sum of n Non-Identically Distributed Uniform Random Variables'*, Annals of the Institute of Statistical Mathematics, 54-3, pp.689–700.

Brasefield, C.J. (1949) *'Winds and Temperatures in the Lower Stratosphere'*, Journal of Meteorology, No. 7.

Bryce, D. and Kambhampati, S. (2007) *'A Tutorial on Planning Graph Based Reachability Heuristics'*, AI Magazine, 28(1), pp.47–83.

Cashmore, M., Fox, M., Long, D. and Magazzeni, D. (2016) *'A Compilation of the Full PDDL+ Language into SMT'*, Twenty-Sixth International Conference on Automated Planning and Scheduling (ICAPS), London, United Kingdom.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N. and Carreras, M. (2015) *'ROSPlan: Planning in the Robot Operating System'*, Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, Jerusalem, Israel.

Castillo, L., Fdez-Olivares, J., García-Pérez, Ó. and Palao, F. (2006) *'Efficiently Handling Temporal Knowledge in an HTN Planner'*, Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS), Cumbria, United Kingdom.

Cervo, F. (2014) *'AFUA: Advance Flexible Use of Airspace:* Optimizing Civil-Military Airspace Integration'.

CGAL (2012) *'CGAL-User and Reference Manual'*.

Chakrabarty, A. and Langelaan, J.W. (2013) *'UAV Flight Path Planning in Time Varying Complex Wind-Fields'*, American Control Conference (ACC), Washington, DC, USA.

Chakrabarty, A. and Langelaan, J.W. (2010) *'Flight Path Planning for UAV Atmospheric Energy Harvesting Using Heuristic Search'*, AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario Canada.

Chen, K., Xu, J. and Reiff-Marganiec, S. (2009) *'Markov-HTN Planning Approach to Enhance Flexibility of Automatic Web Service Composition'*, IEEE International Conference on Web Services (ICWS), Los Angeles, California, USA, pp.9–16.

Cheng, Q., Wang, X., Yang, J. and Shen, L. (2019) *'Automated Enemy Avoidance of Unmanned Aerial Vehicles Based on Reinforcement Learning'*, Applied Sciences, Vol. 9, No. 4, p.669.

Clothier, R.A., Williams, B. and Perez, T. (2013) *'A Review of the Concept of Autonomy in the Context of the Safety Regulation of Civil Unmanned Aircraft Systems'*, Australian System Safety Conference (ASSC), Adelaide, Australia.

Coles, A., Coles, A., Fox, M. and Long, D. (2012) *'COLIN: Planning with Continuous Linear Numeric Change'*, Journal of Artificial Intelligence Research, No. 44, pp.1–96.

Corney, J., Rea, H., Clark, D., Pritchard, J., Breaks, M. and Macleod, R. (2002) *'Coarse filters for shape matching'*, IEEE Computer Graphics and Applications, Vol. 22, No. 3, pp.65–74.

Crosby, M., Petrick, R.P.A., Rovida, F. and Krüger, V. (2017) *'Integrating Mission and Task Planning in an Industrial Robotics Framework'*, Proceedings of the Twenty-Seventh International Conference on Automated Planning and Scheduling (ICAPS), Pittsburgh, Pennsylvania, USA.

De, L. and Guglieri, G. (2012) *'Advanced Graph Search Algorithms for Path Planning of Flight Vehicles', in* Agarwal, R. (Ed.), *Recent Advances in Aircraft Technology*, InTech.

Delauré, B., Michiels, D., Lewyckyj, N. and van Achteren, T. (2013) *'The Development of a Family of Lightweight and Wide Swath UAV Camera Systems Around an Innovative Dual-Sensor On-Single-Chip Detector'*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL.

Della Penna, G., Intrigila, B., Magazzeni, D. and Mercorio, F. (2010) *'A PDDL+ Benchmark Problem: The Batch Chemical Plant'*, Proceedings of the Twentieth International Conference on Automated Planning and Scheduling (ICAPS), Toronto, Canada.

Della Penna, G., Magazzeni, D., Mercorio, F. and Intrigila, B. (2009) *'UPMurphi: A Tool for Universal Planning on PDDL+ Problems'*, Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS), Thessaloniki, Greece.

Deutscher Wetterdienst (2018) *'Produktliste Flugwetterdienst'* [online] www.dwd.de (Accessed 12 April 2019).

Deutscher Wetterdienst (2015) *'pc_met - Internet Service:* Ihr schneller Start zum aktuellen Flugwetter' [online] www.dwd.de (Accessed 12 April 2019).

Donald, B., Xavier, P., Canny, J. and Reif, J. (1993) *'Kinodynamic Motion Planning'*, Journal of the Association for Computing Machinery, 40-5, pp.1048–1066.

Dvorak, F., Bit-Monnot, A., Ingrand, F. and Ghallab, M. (2014) *'A Flexible ANML Actor and Planner in Robotics'*, ICAPS Workshop on Planning and Robotics (PlanRob), Portsmouth, USA.

DWD (2018) *'Ersetzung von COSMO-DE durch COSMO-D2'*.

ECMWF (2018) *'IFS Documentation - Cy45r1:* Operational implementation 5 June 2018', European Centre for Medium-Range Weather Forecasts, Shin.

Edelsbrunner, H., Kirkpatrick, D. and Seidel, R. (1983) *'On the shape of a set of points in the plane'*, IEEE Transactions on Information Theory, Vol. 29, No. 4, pp.551–559.

ESA (2017) *'NAVISP Element 1 Workplan for 2018',* Paris.

Estivill-Castro, V. and Ferrer-Mestres, J. (2013) *'Path-Finding in Dynamic Environments with PDDL-Planners'*, 16th International Conference on Advanced Robotics (ICAR), IEEE [online] http://ieeexplore.ieee.org/servlet/opac?punumber=6755997.

Eun, Y. and Bang, H. (2007) *'Cooperative Task Assignment and Path Planning of Multiple UAVs Using Genetic Algorithm'*, American Institute of Aeronautics and Astronautics (AIAA) Infotech, Aerospace, Rohnert Park, California, USA.

EUROCONTROL and EASA (2018) *'UAS ATM Integration:* Integration Operational Concept', European Organisation for the Safety of Air Navigation (EUROCONTROL).

Everaerts, J. and Lewyckyj, N. (2011) *'Obtaining a Permit-to-Fly for a HALE-UAV in Begium'*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 38-1.

Fdez-Olivares, J., Castillo, L., García-Pérez, Ó. and Palao, F. (2006) *'Bringing Users and Planning Technology Together. Experiences in SIADEX'*, Proceedings of the Sixteenth International Conference on International Conference on Automated Planning and Scheduling, Cumbria, United Kingdom.

Ferguson, D. and Stentz, A. (2005) *'The Field D* Algorithm for Improved Path Planning and Replanning in Uniform and Non-Uniform Cost Environments',* Carnegie Mellon University, Technical Report CMU-TR-RI-05-19.

Filippis, L. de and Guglieri, G. (Eds.), (2012) *Advanced Graph Search Algorithms for Path Planning of Flight Vehicles*, InTech.

Finnegan, P. (2017) *'World Civil Unmanned Aerial Systems:* Teal Market Profile and Forecast', Teal Group Corporation.

Fox, M. and Long, D. (2006) *'Modelling Mixed Discrete-Continuous Domains for Planning'*, Journal of Artificial Intelligence Research (JAIR), No. 27, pp.235–297.

Fox, M. and Long, D. (2003) *'PDDL2.1 : An Extension to pddl for Expressing Temporal Planning Domains'*, Journal of Artificial Intelligence Research (JAIR), No. 20, pp.61–124.

Fox, M., Long, D. and Magazzeni, D. (2017) *'Explainable Planning'*, IJCAI workshop on Explainable AI [online] http://arxiv.org/pdf/1709.10256v1.

Fratini, S. and Cesta, A. (2012) *'The APSI Framework: A Platform for Timeline Synthesis'*, Proceedings of the Workshop on Planning and Scheduling with Timelines, Atibaia, Brazil, Vol. 2012.

Funk, F. and Stütz, P. (2017) *'A Passive Cloud Detection System for UAV: Weather Situation Mapping with Imaging Sensors'*, IEEE Aerospace Conference, Big Sky, Montana, USA, pp.1–12.

Garmin (2018) *'Garmin-GWX70'* [online] http://www.completeavionics.com/assets/gwx_70-spec-sheet.pdf (Accessed 11st March 2019).

Gaschler, A., Petrick, R.P.A., Kröger, T., Khatib, O. and KNoll, A. (2013) *'Robot task and motion planning with sets of convex polyhedra'*.

Gaschler, A.K. (2016) *'Efficient Geometric Predicates for Integrated Task and Motion Planning',* Technische Universität München.

Geffner, H. and Bonet, B. (2013) *'A Concise Introduction to Models and Methods for Automated Planning:* Synthesis Lectures on Artificial Intelligence and Machine Learning', Vol. 7, No. 2, pp.1–141.

Georgievski, I. and Aiello, M. (2014) *'An Overview of Hierarchical Task Network Planning',* arXiv. http://arxiv.org/pdf/1403.7426v1.

Gerdts, M. (2012) *'Optimal Control of ODEs and DAEs',* De Gruyter.

Gerevini, A., Saetti, A. and Serina, I. (2003) *'Planning through Stochastic Local Search and Temporal Action Graphs in LPG'*, Journal of Artificial Intelligence Research, No. 20, pp.239–290.

Ghallab, M., Nau, D. and Traverso, P. (2004) *'Automated Planning: Theory & Practice',* Elsevier.

Graham, R.L. (1972) *'An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set'*, Information Processing Letters, No. 1, pp.132–133.

Hammouri, O.M. and Matalgah, M.M. (2008) *'Voronoi Path Planning Technique for Recovering Communication in UAVs'*, IEEE/ACS International Conference on Computer Systems and Applications, Doha, Qatar, pp.403–406.

Hart, P., Nilsson, N. and Raphael, B. (1968) *'A Formal Basis for the Heuristic Determination of Minimum Cost Paths'*, IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp.100–107.

Haslum, P., Lipovetzky, N., Magazzeni, D. and Muise, C. (2019) *'An Introduction to the Planning Domain Definition Language:* Synthesis Lectures on Artificial Intelligence and Machine Learning', Morgan & Claypool Publishers.

Hehtke, V. (2018) *'Solving a Time-Dependent Multi-HALE Mission Planning Problem Using Genetic Algorithm'*, Master's thesis, University of the Bundeswehr, Munich.

Hoffmann, J. (2003) *'The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables'*, Journal of Artificial Intelligence Research, No. 20, pp.291–341.

Höller, D., Bercher, P., Behnke, G. and Biundo, S. (2018) *'A Generic Method to Guide HTN Progression Search with Classical Heuristics'*, Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS), Delft, The Netherlands.

Honeywell (2016) *'IntuVue® RDR-4000 3D Weather Radar Systems:* Technical White Paper', Honeywell (Ed.). aerospace.honeywell.com (Accessed 11st March 2019).

Hooker, J.N. (1995) *'Testing Heuristics: We Have It All Wrong'*, Journal of Heuristics, No. 1, pp.33–42.

Howey, R., Long, D. and Fox, M. (2004) *'VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL'*, 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Boca Raton, Florida, USA, pp.294–301.

Hunter, S.L.C. (2015) *'Safe Operations Above FL600',* Space Traffic Management Conference, Florida, USA.

Ingrand, F. and Ghallab, M. (2013) *'Robotics and Artificial Intelligence-a Perspective on Deliberation Functions'*, AI Communications.

Jarvis, R.A. (1973) *'On the Identification of the Convex Hull of a Finite Set of Points in the Plane'*, Information Process. Lett.

Johnson, M., Jung, J., Rios, J., Mercer, J., Homola, J., Prevot, T., Mulfinger, D. and Kopardekar, P. (2017) *'Flight Test Evaluation of an Unmanned Aircraft System Traffic Management (UTM) Concept for Multiple Beyond-Visual-Line-of-Sight Operations'*, Twelfth USA/Europe Air Traffic Management Research and Development Seminar (ATM), Seattle, Washington, USA.

Jong, K.A. de (1975) *'Analysis of the Behavior of a Class of Genetic Adaptive Systems',* The University of Michigan, Technical Report No: 185.

Jong, K. de, Fogel, L. and Schwefel, H.-P. (Eds.), (2017) *Handbook of Evolutionary Computation*, Oxford University Press.

Katz, M. and Hoffmann, J. (2014) *'Mercury Planner: Pushing the Limits of Partial Delete Relaxation'*, Eighth International Planning Competition (IPC 2014), Portsmouth, New Hampshire, USA.

Kavraki, L.E., Švestka, P., Latombe, J.-C. and Overmars, M.H. (1996) *'Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces'*, IEEE Transactions on Robotics and Automation,, 12(4), pp.566–580.

Keller, T. and Helmert, M. (Eds.), (2013) *Trial-based Heuristic Tree Search for Finite Horizon MDPs*.

Kiam, J.J., Besada-Portas, E., Hehtke, V. and Schulte, A. (2019a) *'GA-Guided Task Planning for Multiple-HAPS in Realistic Time-Varying Operation Environments'*, The Genetic and Evolutionary Computation Conference (GECCO), Prague, Czech Republic.

Kiam, J.J., Gerdts, M. and Schulte, A. (2016) *'Fast Subset Path Planning/ Replanning to Avoid Obstacles with Time-Varying Probabilistic Motion Patterns'*, Eighth European Starting AI Researcher Symposium (STAIRS), The Hague, The Netherlands, volume 284.

Kiam, J.J., Hehtke, V., Besada-Portas, E. and Schulte, A. (2019b) *'Hierarchical Planning Guided by Genetic Algorithms for Multiple HAPS in a Time-Varying Environment'*, International Conference on Intelligent Human Systems Integration (IHSI), San Diego, California, USA.

Kiam, J.J., Scala, E., Ramirez, M. and Schulte, A. (2018) *'Using a Hybrid AI-Planner to Plan Feasible Flight Paths for HAPS-Like UAVs'*, International Conference of Planning and Scheduing (ICAPS) PlanRob Workshop, Delft, The Netherlands.

Kiam, J.J. and Schulte, A. (2017a) *'Multilateral Mission Planning in a Time-Varying Vector Field with Dynamic Constraints'*, IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, pp.305–310.

Kiam, J.J. and Schulte, A. (2017b) *'Multilateral Quality Mission Planning for Solar-Powered Long-Endurance UAV'*, IEEE Aerospace Conference, Big Sky, Montana, USA.

Kiam, J.J., Schulte, A. and Scala, E. (2019c) *'Using AI-Planning to Solve a Kinodynamic Path Planning Problem and its Application for HAPS'*, International Conference on Intelligent Human Systems Integration (IHSI), San Diego, California, USA.

Klöckner, A. (2016) *'Behavior Trees for Mission Management of High-Altitude Pseudo-Satellites',* Verlag Dr. Hut.

Köhler, M., Funk, F., Gerz, T., Mothes, F. and Stenzel, E. (2017a) *'Comprehensive Weather Situation Map Based on XML-Format as Decision Support for UAVs'*, The Journal of Unmanned System Technology, 5-1.

Köhler, M., Tafferner, A. and Gerz, T. (2017b) *'Cb-LIKE – Cumulonimbus Likelihood: Thunderstorm forecasting with fuzzy logic'*, Meteorologische Zeitschrift, Vol. 26, No. 2, pp.127–145.

Krozel, J. and Andrisani II, D. (1990) *'Navigation path planning for autonomous aircraft: Voronoi diagram approach'*, Journal of Guidance, Control, and Dynamics, Vol. 13, No. 6, pp.1152–1154.

Kuffner, J.J. and LaValle, S.M. (2000) *'RRT-Connect: An Efficient Approach to Single-Query Path Planning'*, IEEE International Conference on Robotics and Automation (ICRA), San Francisco, California, USA.

Laumond, J.-P. (Ed.), (1998) *Robot Motion Planning and Control*, Springer, London.

LaValle, S.M. (1998) *'Rapidly-Exploring Random Trees: A New Tool for Path Planning'*, Technical Report 98-11.

LaValle, S.M. and Kuffner, J.J. (2001) *'Randomized Kinodynamic Planning'*, The International Journal of Robotics Research, Vol. 20, No. 5, pp.378–400.

Leena, P.P., Ratnam, M.V., Murthy, B.K.V. and Rao, S.V.B. (2012) *'Detection of High Frequency Gravity Waves Using High Resolution Radiosonde Observations'*, Journal of Atmospheric and Solar-Terrestrial Physics, Vol. 77, pp.254–259.

Li, F., Manerikar, A.V. and Kak, A.C. (2018) *'RMPD -- A Recursive Mid-Point Displacement Algorithm for Path Planning'*, Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS), Delft, The Netherlands.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A. and Thrun, S. (2005) *'Anytime Dynamic AStar - An Anytime, Replanning Algorithm'*, Proceedings of the Fifteenth International Conference on International Conference on Automated Planning and Scheduling, Montery, California, USA.

Lima, O., Ventura, R. and Awaad, I. (2018) *'Integrating Classical Planning and Real Robots in Industrial and Service Robotics Domains'*, Planning and Robotics (PlanRob) Workshop (ICAPS), Delft, The Netherlands.

Lolla, T., Haley Jr., P.J. and Lermusiaux, P.F.J. (2015) *'Path Planning in Multi-Scale Ocean Flows: Coordination and Dynamic Obstacles'*, Ocean Modelling, Vol. 94, pp.46–66.

Lolla, T., Ueckermann, M.P., Yigit, K., Haley, P. and Lermusiaux, P.F.J. (2012) *'Path Planning in Time Dependent Flow Fields using Level Set Methods'*, IEEE International Conference on Robotics and Automation (ICRA), IEEE [online] http://ieeexplore.ieee.org/servlet/opac?punumber=6215071.

Mandalika, A., Salzman, O. and Srinivasa, S. (2018) *'Lazy Receding Horizon A* for Efficient Path Planning in Graphs with Expensive-to-Evaluate Edges'*, Twenty-Eighth International Conference on Automated Planning and Scheduling (ICAPS), Delft, The Netherlands.

McDermott, D. (2000) *'The 1998 AI Planning Systems Competition'*, AI Magazine.

Meeran, S. and Share, A. (1997) *'Optimum Path Planning Using Convex Hull and Local Search Heuristic Algorithms'*, Mechatronics, Vol. 7(8), pp.737–756.

Miller, W.D. (2018) *'V&V of Cyber-Physical, Autonomous, Artificial Intelligence, and Deep Learning Systems'* [online] [PowerPoint presentation] https://www.faa.gov/about/office_org/headquarters_offices/ang/offices/tc/library

/v&vsummit/v&vsummit2018/v&vsummit2018.html (Accessed 20 January 2020).

Mitchell, M. (1999) *'An Introduction to Genetic Algorithms',* The MIT Press.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (Eds.), (2013) *Playing Atari with Deep Reinforcement Learning*.

Mosier, K.L., Fischer, U., Burian, B.K. and Kochan, J.A. (2017) *'Autonomous, Context-Sensitive, Task Managemenet Systems and Decision Support Tools I: Human Autonomy Teaming Fundamntals and State of the Art',* National Aeronautics and Space Administration - NASA, Ames Research Center, Moffett Field, California.

Müller, R., Kiam, J.J. and Mothes, F. (2018) *'Multiphysical Simulation of a Semi-Autonomous Solar Powered High Altitude Pseudo-Satellite'*, IEEE Aerospace Conference, Big Sky, Montana, USA.

Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J.W., Wu, D. and Yaman, F. (2003) *'SHOP2-An HTN Planning System'*, Journal of Artificial Intelligence Research, No. 20, pp.379–404.

Nugent, P.W., Shaw, J.A. and Piazzolla, S. (2009) *'Infrared Cloud Imaging in Support of Earth-Space Optical Communication'*, Optics Express, 17-10, pp.7862–7872.

Otte, M., Silva, W. and Frew, E. (2016) *'Any-Time Path-Planning: Time-Varying Wind Field + Moving Obstacles'*, IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden.

Owen, M., Beard, R.W. and McLain, T.W. (Eds.), (2013) *Implementing Dubins Airplane Paths on Fixed-wing UAVs*, Springer.

Pecora, F., Andreasson, H., Mansouri, M. and Petkov, V. (2018) *'A Loosely-Coupled Approach for Multi-Robot Coordination, Motion Planning and Control'*, International Conference on Automated Planning and Scheduling (ICAPS), Delft, The Netherlands.

Pehlivanoglu, Y.V. (2012) *'A New Vibrational Genetic Algorithm Enhanced with a Voronoi Diagram for Path Planning of Autonomous UAV'*, Aerospace Science and Technology, No. 16, pp.47–55.

Perez-Carabaza, S., Besada-Portas, E., Lopez-Orozco, J.A. and La Cruz, J.M. de (2018) *'Ant Colony Optimization for Multi-UAV Minimum Time Search in Uncertain Domains'*, Applied Soft Computing, Vol. 62, pp.789–806.

Perez-Carabaza, S., Besada-Portas, E., Lopez-Orozco, J.A. and La Cruz, J.M. de (2016) *'A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection' in* the 2016. Denver, Colorado, USA, ACM Press, New York, New York, USA, pp.981–988.

Piotrowski, W., Fox, M., Long, D., Magazzeni, D. and Mercorio, F. (2016) *'Heuristic Planning in PDDL+ Domains'*, Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, Phoenix, Arizona, USA.

Pohl, I. (1970) *'Heuristic Search Viewed as Path Finding in a Graph'*, Artificial Intelligence, No. 1, pp.193–204.

Ragi, S. and Chong, E. (2013) *'UAV Path Planning in a Dynamic Environment via Partially Observable Markov Decision Process'*, IEEE Transactions on Aerospace and Electronic Systems, Vol. 49(4), pp.2397–2412.

Ramirez Atencia, C., Del Ser, J. and Camacho, D. (2019) *'Weighted strategies to guide a multi-objective evolutionary algorithm for multi-UAV mission planning'*, Swarm and Evolutionary Computation, Vol. 44, pp.480–495.

Redman, B.J., Shaw, J.A., Nugent, P.W., Clark, R.T. and Piazzolla, S. (2018) *'Reflective All-Sky Thermal Infrared Cloud Imager'*, Optics Express, Vol. 26, No. 9, pp.11276–11283.

Rees, M. (2019) *'First Approval for Fully Autonomous Drone Flights in Europe Granted'* [online] www.unmannedsystemstechnology.com (Accessed 14th March 2019).

Roussos, G., Dimarogonas, D.V. and Kyriakopoulos, K.J. (2009) *'3D Navigation and Collision Avoidance for Nonholonomic Aircraft-like Vehicles'*, International Journal of Adaptive Control and Signal Processing, No. 0, pp.1–21.

Rüdiger, E. and Drechsler, R. (2009) *'Weighted A∗ Search – Unifying View and Application'*, Artificial Intelligence, Vol. 173, No. 14, pp.1310–1342.

Rumbaugh, J., Jacobson, I. and Booch, G. (1999) *'The Unified Modeling Language Reference Manual - UML',* Addision-Wesley.

Runarsson, T.P. and Yao, X. (2000) *'Stochastic ranking for constrained evolutionary optimization'*, IEEE Transactions on Evolutionary Computation, Vol. 4, No. 3, pp.284–294.

Russell, S.J. and Norvig, P. (2003) *'Artificial Intelligence: A Modern Approach'*.

Scala, E., Haslum, P., Thiebaux, S. and Ramirez, M. (2016) *'Interval-Based Relaxation for General Numeric Planning'*, 22nd European Conference on Artificial Intelligence (ECAI), The Hague, The Netherlands.

Schmitt, F. and Schulte, A. (2016) *'Mixed-Initiative Missionsplanung in militärischen Hubschraubermissionen'*, Workshop Kognitive Systeme, Bochum, Germany.

Schulte, A. and Donath, D. (2018) *'A Design and Description Method for Human-Autonomy Teaming Systems', in* Karwowski, W. and Ahram, T. (Eds.), *Intelligent Human Systems Integration*, Springer International Publishing, Cham, pp.3–9.

Schulte, A., Donath, D. and Lange, D.S. (2016) *'Design Patterns for Human-Cognitive Agent Teaming'*, International Conference on Engineering Psychology and Cognitive Ergonomics, Toronto, Canada.

Shima, T., Rasmussen, S. and Sparks, A.G. (2005) *'UAV Cooperative Multiple Task Assignments using Genetic Algorithms'*, Proceedings of the 2005 American Control Conference, American Automatic Control Council [online] http://ieeexplore.ieee.org/servlet/opac?punumber=9861.

Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell and Pieter Abbeel (2014) *'Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer'*.

Silvia Richter, M.W. (2010) *'The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks'*, Journal of Artificial Intelligence Research, No. 39, pp.127–177.

Simpson, E.H. (1949) *'Measurement of Diversity'*, Nature, No. 163, p.688.

Sirin, E., Parsia, B., Wu, D., Hendler, J. and Nau, D. (2004) *'HTN Planning for Web Service Composition Using SHOP2'*, Journal Web Semantics: Science, Services and Agents on the World Wide Web, Fort Belvoir, VA, 1-4, pp.377–396.

Skala, V. (2015) *'Point-in-Convex Polygon and Point-in-Convex Polyhedron Algorithms with O(1) Complexity Using Space Subdivision'*, International Conference of Numerical Analysis and Applied Mathematics (ICNAAM).

Srivastava, S., Fang, E., Riano, L., Chitnis, R., Russell, S. and Abbeel, P. (2014) *'Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer'*, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China.

Stentz, A. (1995) *'The Focussed D* Algorithm for Real-Time Replanning'*, In Proceedings of the International Joint Conference on Artificial Intelligence, Quebec, Canada.

Stockdale, T., Balmaseda, M. and Ferranti, L. (2017) *'The 2015/2016 El Niño and beyond',* ECMWF (Ed.).

Sucan, I.A., Moll, M. and Kavraki, L.E. (2012) *'The Open Motion Planning Library'*, IEEE Robotics & Automation Magazine, Vol. 19, No. 4, pp.72–82.

Sun, J., Li, B., Jiang, Y. and Wen, C.-Y. (2016) *'A Camera-Based Target Detection and Positioning UAV System for Search and Rescue (SAR) Purposes'*, Sensors, Basel, Switzerland, Vol. 16, No. 11.

Sutton, R.S. and Barto, A.G. (2017) *'Reinforcement Learning: An Introduction',* The MIT Press.

Temizer, S., Kochenderfer, M.J., Kaelbling, L.P., Lozano-Perez, T. and Kuchar, J.K. (2010) *'Collision Avoidance for Unmanned Aircraft using Markov Decision Processes'*, American Institute of Aeronautics and Astronautics (AIAA) Guidance, Navigation, and Control Conference (GNC), Toronto, Ontario, Canada.

Thrun, S., Burgard, W. and Fox, D. (2005) *'Probabilistic Robotics',* MIT Press.

Urmson, C. and Simmons, R. (2003) *'Approaches for Heuristically Biasing RRT Growth'*, Proceedings 2003 IEEE/RSJ International Conference, Las Vegas, Nevada, USA, pp.1178–1183.

Vallati, M., Magazzeni, D., Schutter, B. de, Chrpa, L. and McCluskey, T.L. (2016) *'Efficient Macroscopic Urban Traffic Models for Reducing Congestion: a PDDL+ Planning Approach'*, Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA.

Webb, D. and van den Berg, J. (2013) *'Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Dynamics'*, IEEE International Conference on Robotics and Automation (ICRA), Karlsruhe, Germany.

Wolek, A. (2015) *'Optimal Paths in Gliding Flight',* Virginia Polytechnic Institute and State University.

Wu, G., Say, B. and Sanner, S. (Eds.), (2017) *Scalable Planning with Tensorflow for Hybrid Nonlinear Domains*.

Yang, Z. and Cohen, F.S. (1999) *'Image registration and object recognition using affine invariants and convex hulls - Image Processing, IEEE Transactions on'*, IEEE Transactions on Image Processing, 8-7, pp.934–946.

Younes, H.L.S. and Littman, M.L. (2004) *'PPDDL1.0: An Extension to PDDL for Expressing Planning Domains with Probabilistic Effects',* School of Computer Science, Carnegie Mellon University.

Zhou, B., Schwarting, W., Rus, D. and Alonso-Mora, J. (2018) *'Joint Multi-Policy Behavior Estimation and Receding-Horizon Trajectory Planning for Automated Urban Driving'*, IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia.