An AI-Based Planning Framework for HAPS in a Time-Varying Environment

Jane Jean Kiam,[1] Enrico Scala,[2] Miquel Ramirez Javega,[3] Axel Schulte[1]
[1]Bundeswehr University Munich, [2]University of Brescia, [3]The University of Melbourne
{jane.kiam, axel.schulte}@unibw.de, enrico.scala@unibs.it, miquel.ramirez@unimelb.edu.au

Abstract

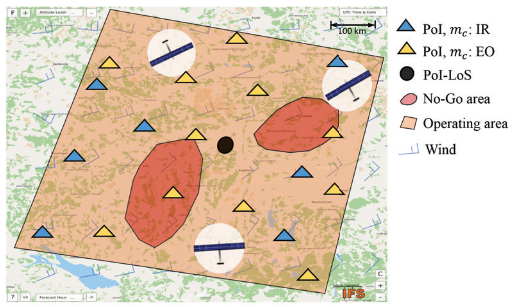
A High-Altitude Pseudo-Satellite (HAPS) is a fixed-wing, solar-powered Unmanned Aerial Vehicle (UAV) developed to become a flexible alternative to satellites with fixed-orbits for monitoring ground activities over long periods of time. However, given its lightweight build and weak electro-motors, the platform is rather sensitive to weather and cannot fly around hazardous weather zones swiftly. In this work, we formulate the problem of planning missions for multiple HAPS as a hybrid planning problem expressed in PDDL+. The formulation also considers the problem of modeling the platform dynamics, the time-varying environment, and the heterogeneous tasks that need to be carried out. Additionally, we propose a framework that combines a PDDL+ automated planner with an Adaptive Large Neighborhood Search (ALNS) approach, developed to couple the automated planner with a meta-heuristic that is specific for the problem. The task and motion planning are done in an intertwined way within the framework, preserving hence a common decision/search space. We validate our approach with a third-party realistic simulator for HAPS, as well as with a set of benchmark tests, showing that our integrated approach produces executable mission plans.

Introduction

High-Altitude Pseudo-Satellites (HAPS) (see Figure 1a) are solar-powered, High-Altitude Long-Endurance (HALE) Unmanned Aerial Vehicles (UAVs) operating in the lower stratosphere (~18 km), and are intended as an alternative to satellites with fixed-orbits or aircraft that need refueling to monitor ground activities. To maximize endurance, HAPS are built of lightweight material (~100 kg), therefore susceptible to impairment in difficult weather conditions, such as strong winds, thunderstorms etc. The weak electro-motors (~1.7 kW) hardly allow the platform to maneuver around hazards. With its low airspeed (~9 m/s), the effects of wind cannot be neglected (Müller, Kiam, and Mothes 2018).

HAPS operate in a managed airspace (Everaerts and Lewyckyj 2011) above FL600. Therefore, a flight plan with details on the operation must be made available prior to execution (EUROCONTROL and EASA 2018). Successful tests have motivated plans for large scale deployments of HAPS

(a) Zephyr 7 at launch ©ADS GmbH



(b) A typical airspace structure defined for repetitive monitoring tasks

Figure 1: HAPS and its deployment as a satellite substitute

in the near future (Airbus Defence and Space GmbH 2018)[1]. However, planning can be laborious, due to the idiosyncratic operating characteristics discussed above. We will show that planning can be automatized to a great degree.

Mission Scenario and Problem Statement

Figure 1b illustrates a typical mission scenario involving multiple HAPS h_i deployed to monitor Points of Interest (PoIs), which are marked with triangles, within an operating area. Each HAPS is equipped with a passive sensor, either an electro-optical (EO) or an infrared (IR) camera. The PoIs must be monitored (repeatedly) using an appropriate sensor. Each PoI is associated to a task $o_{\mathrm{PoI}}(p, W_{\mathrm{s,e}}, m_c, m)$ where p is the position of the PoI, $W_{\mathrm{s,e}}$ is a set of time windows within

[1]The test on Zephyr from Airbus in 2018 has set the longest continuous unmanned flight record of almost 26 days. https://www.airbus.com/defence/uav/zephyr.html

which an image taken of the PoI is rewarded, m_c is the type of sensor required, and m is a Boolean variable, set to true when the task is active, or false when the task is inactive, e.g. when the limits of the revisit frequency is exceeded.

Since each HAPS has limited onboard memory, after monitoring a certain number of PoIs, it is required to send the recorded images to the ground control station (GCS) at the black circle in Figure 1b, where a line-of-sight (LoS) communication link to the GCS is available. This task, $o_{\mathrm{LoS}}(p, a)$, is characterized by p the position enabling communications with the GCS, and a the Boolean representing the availability of a communication channel. In the scenarios we consider, the GCS can only communicate with one HAPS at a time.

For mission safety and plan executability, besides the dynamics of the platform, a realistic time-varying model of the environment must be considered too. (Köhler et al. 2017) suggests the use of high-resolution numerical global weather data for operating weather-sensitive platforms. For example, COSMO-DE (COnsortium for Small-scale MOdeling) has a resolution of ~2.8 km (Baldauf et al. 2011). Prediction of dynamic no-fly zones $D = \{v_i(t)\}$, polygons given by an (ordered) set of vertices, can subsequently be derived from the models to exclude areas with dangerous weather patterns (Köhler, Gerz, and Tafferner 2016). Additionally, four-dimensional wind fields (Baldauf et al. 2011)[2] are useful to predict the movement of the HAPS induced by the surrounding airflow $v_{\mathrm{wind}}(p, t)$. Furthermore, the image recorded by an EO-camera is strongly affected by the cloud layer(s) between the ground and the operating altitude, implicating hence the need to consider the time-varying cloud coverage $cc(t)$.

The plans must be optimized so that, 1) as many PoIs as possible are visited at their requested time windows, 2) the image quality is acceptable and 3) the geographical distribution of the visited PoIs is homogeneous. The weighted objective function is given by Equation 1, where $n_{o_{\mathrm{p}}}(s)$ is the total number of PoIs monitored in the plan s, $R(o_{\mathrm{p}})$ the preference/reward over tasks o_{p}, $prob_{\mathrm{succ}}(o_{\mathrm{p}})$ is the probability of an acceptable image taken of the PoI of o_{p}, which is the inverse of the $cc(t)$ above the PoI. $dn_{o_{\mathrm{p}}}(\mathcal{A})$ is the density of PoIs, i.e. the number of monitored PoIs divided by the number of PoI requests of a number of cells $\mathcal{A}$ the operation area is divided into.

$$f(s) = \omega_1 n_{o_{\mathrm{p}}}(s) + \omega_2 \sum_{o_{\mathrm{p}} \in s} R(o_{\mathrm{p}}) \cdot prob_{\mathrm{succ}}(o_{\mathrm{p}}) + \omega_3(\max_{\mathcal{A}}(dn_{o_{\mathrm{p}}}(\mathcal{A})) - \min_{\mathcal{A}}(dn_{o_{\mathrm{p}}}(\mathcal{A}))) \quad (1)$$

Contributions

In this work, we focus on performing task and motion planning in an integrated manner for HAPS, by intertwining task and motion planning decision.

To achieve that, we explore extensively the expressivity of PDDL+ (Problem Domain Definition Language), i.e. besides encoding for task planning, we also use processes and global constraints of PDDL+ to model a complex kinodynamic motion planning problem. The capacity of a PDDL+ compatible automated planner to solve the latter is compared, in an experimental setting, against a state-of-the art motion planner, distributed with the Open Motion Planning Library (OMPL) (Sucan, Moll, and Kavraki 2012). Results show that the PDDL+ motion planning is competitive. The simplified motion model in PDDL+ is also validated using a six degrees-of-freedom (6-DoF) HAPS simulator and real weather data. For scalability, we use an Adaptive Large Neighborhood Search (ALNS) framework to wrap around the automated planner, in order to use heuristics devised for our planning problem, while ensuring that the searches for task and motion intertwine, thereby avoiding a decoupled task+motion planning approach.

The aim is to overcome the difficulties arising from decoupling task and motion planning, which relies on first generating a task plan (by ignoring details on the motion), and refining it with a motion planner. This results often in non-usable task plans. We show how the tasks (associated to the PoIs) and the motion planning (considering the platform dynamics and the mission constraints) can be formulated all together, and how to tackle them, to yield a good compromise between fidelity of the model and overall optimality.

Task + Motion Planning in PDDL+ for HAPS

In this section, we summarize how PDDL+ (Fox and Long 2006) can be used to capture the continuous and discrete dynamics of the planning problem, as well as the tasks to be assigned to the HAPS by the planner executive in the development of a plan.

Task Planning

As a derivative of PDDL, PDDL+ captures classical planning (McDermott 2000; Fox and Long 2003; 2006). Figure 2 depicts typical classical planning actions necessary for our planning problem. `assign-poi-to-haps` assigns a PoI `?p` to a HAPS `?h` if 1) `?h` is not assigned a task, 2) the memory of `?h` is not full, 3) `?p` is not yet assigned to a HAPS, and 4) `?h` has the required sensor. `monitor-poi` commands the monitoring of a PoI `?p` when the HAPS `?h` is in the vicinity. `assign-send-image-to-gcs` engages the HAPS to communicate images to the GCS, if the LoS communication link is available. The action negates the predicate `GCS-is-free`, i.e. the GCS communication terminal is engaged.

Kinodynamic Motion Planning Problem

To ensure feasibility, the time varying environment and platform dynamics must be considered during planning. According to (Donald et al. 1993), a kinodynamic planning problem includes consideration of *kinematic constraints*, e.g. obstacles and boundaries to avoid, as well as *dynamics constraints*, e.g. time-derivatives of the agent's physical configuration. Most motion planning tools are sampling-based; if dynamics constraints are involved, control-based motion planners are used, among which the most commonly exploited are approaches based on Rapidly exploring Random Tree (RRT) (Allen and Pavone 2015; Webb and van den Berg 2013) and on A* (De Filippis and Guglieri 2012). Particularly with RRT, the control-based

[2]Daily wind data in GRIB-format available on https://www.dwd.de/EN.

```
(:action assign-poi-to-haps
:parameters (?h -haps ?p -poi)
:precondition (and (not (assigned-haps ?h))
               (< (collected-images ?h) max-images)
               (active ?p)
               (matching-payload ?p ?h)
:effect (and (haps-goal ?h ?p) (not (active ?p))
             (assigned-haps ?h))

(:action monitor-poi
:parameters (?h -haps ?p -poi)
:precondition (and (isnear ?h ?p) (haps-goal ?h ?p))
:effect (and (= (last-visit ?p) time-elapsed)
             (not (assigned-haps ?h))
             (increase (collected-images ?h) image-size))

(:action assign-send-image-to-gcs
:parameters (?h -haps)
:precondition (and (>= (collected-images ?h) 0)
                   (GCS-is-free) (not (assigned-haps ?h)))
:effect (and (not (GCS-is-free))
             (assigned-haps ?h)
             (haps-goal ?h los))
```

Figure 2: Tasks encoded as actions in PDDL+

variant is also available on OMPL (Sucan, Moll, and Kavraki 2012), an open source library with a well-defined API to be used *off-the-shelf*, i.e. the problem modelling (state space and constraints) and the solver are two independent entities[3].

We demonstrate in the following subsections more detailed and richer expressions of PDDL+ than in (Kiam, Schulte, and Scala 2019) to model the kinodynamic motion planning problem of a HAPS navigating in a time-varying envirionment with dynamic obstacles. Figure 3 provides a mapping of the problem formulation in PDDL+ and OMPL side-by-side.

Dynamics Constraints for HAPS The kinematic model of HAPS in a wind field with a spherical Earth assumption can be represented by the following (Müller, Kiam, and Mothes 2018):

$$\begin{aligned}
\dot{\lambda}(t) &= (v_{\text{wind,E}}(t) + v_{\text{TAS}} \cos\gamma(t) \sin\chi(t))/(R+h)\cos\phi(t),\\
\dot{\phi}(t) &= (v_{\text{wind,N}}(t) + v_{\text{TAS}} \cos\gamma(t) \cos\chi(t))/(R+h),\\
\dot{h}(t) &= v_{\text{wind,U}}(t) + v_{\text{TAS}} \sin\gamma(t),
\end{aligned} \tag{2}$$

where λ, ϕ and h denote respectively the longitude, latitude and altitude of HAPS. χ and γ are the yaw and pitch angle, R the radius of the Earth, $v_{\text{wind}} = (v_{\text{wind,E}}, v_{\text{wind,N}}, v_{\text{wind,U}})^{\text{T}}$ the wind velocity in the East-North-Up coordinates, and v_{TAS} the True Air Speed (TAS). As a fixed-wing aircraft, it is desired to fly at the optimal equivalent airspeed (EAS), which can be scaled to obtain the TAS at different altitude levels $v_{\text{TAS}} = v_{\text{EAS}}\sqrt{\rho_0/\rho(h)}$, where $\rho(h)$ and ρ_0 are respectively the ambient and sea-level air densities.

The HAPS can be controlled via its attitude configuration, by considering $A_{\dot{\chi}} = \{-|\dot{\chi}_{\max}|, -|\dot{\chi}_{\max}| + \Delta\dot{\chi}, \ldots, |\dot{\chi}_{\max}| - \Delta\dot{\chi}, |\dot{\chi}_{\max}|\}$ the set of feasible turn rate and $A_\gamma = \{-|\gamma_{\max}|, -|\gamma_{\max}| + \Delta\gamma, \ldots, |\gamma_{\max}| - \Delta\gamma, |\gamma_{\max}|\}$ the set of climb angles. The decision for the control parameters can also be formulated as actions in PDDL+, for example `increase-turn-rate` in Figure 3 to increase turn rate $\dot{\chi}(t+\Delta t) := \dot{\chi}(t) + \Delta\dot{\chi}$, applicable only if $\dot{\chi}(t) < |\dot{\chi}_{\max}| - \Delta\dot{\chi}$. Similar formulation also applies for decreasing turn rate, increasing and decreasing climb angle.

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

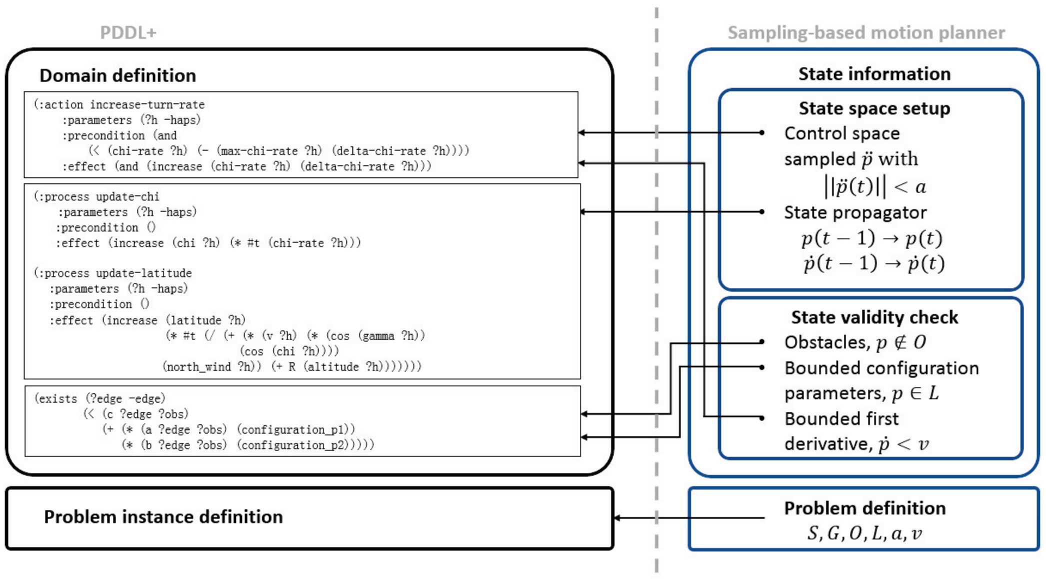
Subsequently, the position of the HAPS in WGS84 coordinates can be updated using Equations 2 with processes, for example `update-chi` and `update-latitude` in Figure 3 to update the heading and latitude respectively. Similar formulations are also used for updating longitude and altitude.

Kinematic Constraints for HAPS Weather critical no-go areas (e.g. strong wind, turbulences, thunderstorm clouds etc.) predicted by weather forecast sources (Köhler et al. 2017), as well as the operation area (since HAPS operates at a constant flight level) can be represented as convex-polygonal obstacles applicable for each altitude level. Algorithm 1 checks if a point p lies within a polygon represented by a set of ordered vertices $V = (v_i)$. The parameters of the edges (Line 1 to 6) can be pre-processed and parsed to the PDDL+ problem instance definition file. Line 7 to 11 are intended to examine if the HAPS lies on the same side of each edge $v_i v_{i+1}$ of the polygon as an arbitrary interior point, and can be formulated as given in Figure 3 using an existential quantified formula, which we include in our set of global constraints. This either ensures that HAPS fly within the allocated operation area or that the HAPS are not an interior point of the no-go areas.

Algorithm 1 Determine the inclusion of a point $p = (\lambda, \phi)$ in a convex polygon

```
Require: $V$, an ordered set of vertices of a convex polygon
 1: for each edge $v_i v_{\overline{i+1}}$ do
 2:     determine $a_i$, $b_i$, $c_i$ such that
 3:     $a_i\lambda_i + b_i\phi_i == c_i$ and
 4:     $a_i\lambda_{\overline{i+1}} + b_i\phi_{\overline{i+1}} == c_i$ and    % $\overline{*}$: circular indexing
 5:     $a_i\lambda_{\overline{i+2}} + b_i\phi_{\overline{i+2}} \leq c_i$
 6: end for
 7: if $\wedge_i(a_i\lambda + b_i\phi \leq c_i)$ then
 8:     $p$ is in the convex polygon described by $V$
 9: else
10:     $p$ is NOT in the convex polygon $V$
11: end if
```

Wind Vector Field for HAPS High-resolution wind data (e.g. from COSMO-DE) provides independently zonal wind v_λ, meridional wind v_ϕ, and vertical wind v_{up} for a discrete four-dimensional grid (longitude-latitude-altitude-time), with regular spacing for the altitude and time dimensions, but irregular spacing for the longitude-latitude grid, forming hence at a given altitude, cells of arbitrary quadrilaterals (see Figure 4) (Baldauf et al. 2011). The mean value of the wind components of each 4D-grid can be pre-determined and parsed into the problem file in PDDL+ as `(***-wind ?lat-lon-grid ?altitude-level ?time-interval)`, where *** can be


PDDL+
Sampling-based motion planner
Domain definition
State information
State space setup
Control space sampled $\ddot{p}$ with $\|\ddot{p}(t)\| < a$
State propagator
$p(t-1) \rightarrow p(t)$
$\dot{p}(t-1) \rightarrow \dot{p}(t)$
State validity check
Obstacles, $p \notin O$
Bounded configuration parameters, $p \in L$
Bounded first derivative, $\dot{p} < v$
Problem instance definition
Problem definition
S, G, O, L, a, v

```
(:action increase-turn-rate
    :parameters (?h -haps)
    :precondition (and
        (< (chi-rate ?h) (- (max-chi-rate ?h) (delta-chi-rate ?h))))
    :effect (and (increase (chi-rate ?h) (delta-chi-rate ?h)))
```

```
(:process update-chi
    :parameters (?h -haps)
   :precondition ()
   :effect (increase (chi ?h) (* #t (chi-rate ?h)))

(:process update-latitude
  :parameters (?h -haps)
  :precondition ()
  :effect (increase (latitude ?h)
                    (* #t (/ (+ (* (v ?h) (* (cos (gamma ?h))
                                       (cos (chi ?h))))
                   (north_wind ?h)) (+ R (altitude ?h)))))))
```

```
(exists (?edge -edge)
       (< (c ?edge ?obs)
          (+ (* (a ?edge ?obs) (configuration_p1))
             (* (b ?edge ?obs) (configuration_p2)))))
```

Figure 3: Mapping between the problem definition for a classical sampling-based motion planner and the formulation in PDDL+

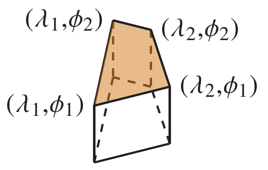


Figure 4: 3D (longigude-latitude-altitude) cell of a wind grid

east-, north- or up-wind. An event in PDDL+ can be used to continuously determine (using the `assign` operator) the wind in the vicinity of the HAPS, by using Algorithm 1 (and `exists` quantificator of Figure 3) to check which cell the HAPS is in, as well as inequalities to determine its altitude level and the current time interval.

Some no-go areas (e.g. clouds), move along with the wind. In our work, this movement is assumed linear, with each vertex moving at the speed of the wind evaluated at the barycenter of the polygon. The variation over time of the inequality parameters in Algorithm 1 are given by

$$a(t+\delta t) = a(t), \quad b(t+\delta t) = b(t) \tag{3}$$
$$c(t+\delta t) = c(t) + v_{\phi,\text{wind}}(t) \cdot b(t) \cdot \Delta t + v_{\lambda,\text{wind}}(t) \cdot a(t) \cdot \Delta t,$$

where $v_{\lambda,\text{wind}}$ and $v_{\phi,\text{wind}}$ are the zonal and meridional wind components in rad/s at the barycenter of the polygon. Although this assumption is simplified, it is practical, and necessary given the low airspeed of the HAPS and the wide mission areas. A safety margin can be added to the polygons to allow for deformation of the clouds and the neglected non-linear movements. Equations 3 can be represented as processes in PDDL+ using basic algebraic functions.

Top-Level Implementation Framework

The encoding in PDDL+ allows the use of automated planners off-the-shelf to solve a task+motion planning problem in a tightly-coupled manner. However, this does not always scale with the complexity of the problem. Moreover, even if PDDL+ allows the expression of objective functions like Equation 1 using a plan metric (with the keyword `:metric`), few planners support the optimization of it.

Therefore, we propose a top-level intelligent framework (see Algorithm 2) to wrap around the automated planner in order to 1) scale the planning for larger problems, i.e. more HAPS, more goals, and 2) support the optimization of complex objective functions. The framework intends to take over the burden to assign PoI/LoS-zone to HAPS (high-level tasks), rather than leaving it to the PDDL+ automated planner, in which an action that assigns a PoI to one of the HAPS is considered at the same level as an action that decides for a control parameter for the guidance of a HAPS. While the implementation is no longer a true tightly-coupled approach to solve a task+motion planning problem, it enables task+motion planning to be done in an integrated manner, i.e. the search switches from the assignment of high-level tasks to the low-level control-based numeric planning in a intertwining fashion. Therefore, the decision of each high-level task to be inserted to the plan solution is followed immediately by a lower-level numeric planning. By doing so, the feasibility and quality of the explored solutions at each search step can be more precisely estimated. Beside encouraging scalability, our integrated task+motion approach has another obvious advantage: *The common search space for task and control actions is still preserved, although the search heuristics for task-actions and for control-actions can be different.*

ALNS

As described in Algorithm 2, the task assignment for multiple HAPS in the intelligent framework uses the Adaptive Large Neighborhood Search (ALNS) adapted from (Ropke and Pisinger 2004). ALNS is a commonly used method to solve job-shop scheduling problems with metaheuristics (Dang, Rudová, and Nguyen 2019). The advantage of ALNS is multifold:

1. Given the complex objective function, a single heuristic often works poorly. ALNS allows the use of multiple heuristics to explore the search space and exploit the good solutions.
2. Only one initial solution plan is needed to initialize the algorithm, which greatly reduces the effort of numeric planning at initialization.
3. The algorithm is anytime, i.e. the (initial) solution plan is improved as much as possible within the allocated planning time $pt_{\max}$.

Algorithm 2 PDDL+ automated planner wrapped around by ALNS to perform integrated task+motion planning

```
Require: $\mathcal{R}_{o_p}$ the request-list of tasks $o_{\text{PoI}}$ and $o_{\text{LoS}}$, $H$ a set of HAPS, $T = [T_{\min}, T_{\max}]$ the plan horizon, $pt_{\max}$ allocated planning time
 1: generate initial solution plan $s_0 = \{o_1, ..., o_i, ..., o_{|s_0|}\}$
 2: $s \leftarrow s_0$ % assign $s_0$ as the current solution
 3: $s_{\text{best}} \leftarrow s_0$ % assign $s_0$ as the best solution
 4: while $pt < pt_{\max}$ do
 5:     update request bank
 6:     flag_neighbor_plan_not_found = true
 7:     while flag_neighbor_plan_not_found do
 8:         $s' \leftarrow s$
 9:         remove with a heuristic $h_{Ri}$ a task $o_i$ from $s'$
10:         insert with a heuristic $h_{I_i}$ a task $o_i^{'}$ into $s'$
11:         call automated planner to stitch $o_{i-1}$ and $o_i'$
12:         call automated planner to stitch $o_i'$ and $o_{i+1}$
13:         if automated planning is successful then
14:             flag_neighbor_plan_not_found = false
15:         end if
16:     end while
17:     update all time stamps of $o_j$, $j > i$
18:     while $T_{\max}$ is not reached do
19:         insert with a heuristic $h_{Ii}$ $o_k$ into the end of $s'$
20:     end while
21:     if $valid(s')$ ==true and $f(s') > f(s_{\text{best}})$ then
22:         accept $s_{\text{best}} \leftarrow s'$
23:     else if $f(s') > f(s)$ or $r_p < \exp^{\frac{f(s')-f(s_{\text{best}})}{Tp}}$ then
24:         $s \leftarrow s'$
25:         update $Tp$
26:     end if
27:     update scores for the remove/insert heuristics
28:     update $\mathcal{R}_{o_p}$
29: end while
```

The initial solution plan s_0 (see Line 1) is generated by assigning the PoI of $\mathcal{R}_{o_p}$ with the shortest Euclidean distance to the HAPS at each step. If the planner cannot compute a plan (at the control-action level) before timeout, the PoI is removed and the next closest is assigned.

While there is still planning time left (Line 4), local search in the neighborhood is performed by removing a task (and the associated actions), and by inserting a task to be assigned to the HAPS that is supposed to perform the removed task (see Line 9 to 10). The task removal and insertion are carried out using the heuristics in Table 1, followed by stitching the inserted task $o_i^{'}$ with the previous task o_{i-1} and the following task o_{i+1}. This is achieved by calling the automated planner to plan from the initial problem instance defined by the last state (including time) after o_{i-1} is completed toward the goal defined by the state to be reached after $o_i^{'}$ is performed. This is done similarly to stitch $o_i^{'}$ and o_{i+1}. Subsequently, the state the HAPS are supposed to reach after 10% of the original partial plan trace from o_{i+1} to o_{i+2} will be set as the new goal to plan from the state and time reached after performing o_{i+1}. The rest of the plan (actions list) remains the same except for the shifted timestamps resulting from the stitching. If the plan horizon is not reached, tasks are added to the end of the plan (Line 18 to 20). s' is accepted as the new best solution s_{best}, if it is valid and if its objective value is better than the best solution s_{best}. Or else, s' is accepted as the current solution s with a probability $\exp^{\frac{f(s')-f(s_{\text{best}})}{Tp}}$, even if s' is not valid (see Line 23). The underlying reason is to benefit from the simulated annealing to avoid convergence to a local minimum. Tp is the "temperature" that decreases at the rate c with each iteration of a segment ($Tp \leftarrow Tp \cdot c$), to reduce the "exploring" effect of ALNS.

For the adaptive selection of heuristics, the entire search is divided into segments, i.e. the number of iterations of the local search from Line 1 to 27. Heuristic h_{R_i} and h_{I_i} are chosen (in Line 9 to 10) in segment $j+1$ with a probability $w_{i,j+1} = w_{i,j}(1-r) + rS_{i,j}/a_{i,j}$ where S_i is the score of heuristic i in segment j, and a_i is the total number of times heuristic i has been used in segment j. r is an exploitation factor, i.e. when r is 1, we favor heuristic w_i based solely on the score it has obtained so far. Lastly, the score of a heuristic increases by 1) σ_1 if the new solution becomes the global best solution, 2) σ_2 if $f(s') > f(s)$, 3) σ_3 if $f(s') < f(s)$, but the solution is accepted, 4) σ_4 if the numeric planner finds a plan before timeout. The weighting of heuristics encourages convergence to an optimum, while the random selection broadens the exploration of search space.

The remove and insert heuristics devised specifically for our planning problem are listed in Table 1. The remove heuristics intend to include any factor that could disfavor the mission, and insert heuristics try to exploit favorable factors. Some randomness is encouraged in the remove and insert mechanism for more exploration in the search space.

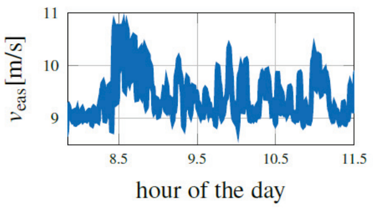
Experimental Evaluation

This section describes the validation and performance tests (based on the described scenario) conducted on the AI planning framework for an integrated task+motion planning.

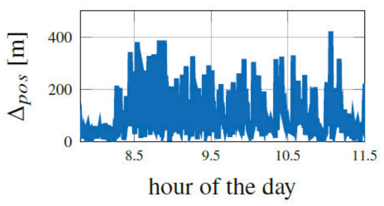
Experimental Setup

Although many automated planners come with a PDDL+ front-end interface, to the best of our knowledge, ENHSP[4] is

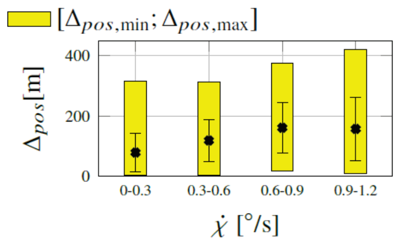
[4]https://gitlab.com/enricos83/ENHSP-Public



(a) Equivalent airspeed of HAPS during plan execution



(b) Lateral deviation between planned and simulated path



(c) Lateral position error (mean, standard deviation and distribution) vs. turn rate

Figure 5: Validation of the planned paths

the only one capable of solving a problem involving kinodynamic motion planning, given its ability to cope with more advanced mathematical operations (Scala et al. 2016), such as exponential and trigonometric functions, necessary to describe the dynamics of the HAPS (see Figure 3). ENHSP is a heuristic search forward state planner (Ghallab, Nau, and Traverso 2004; Geffner and Bonet 2013), with built-in heuristic components to guide the search to the goal state, by extending the search tree rooted at the initial state with edges corresponding to spontaneous (processes) or instantaneous (actions) state transitions. It was experimentally determined that the greedy best-first search (GBFS), combined with the AIBR heuristic (Scala et al. 2016), produces the best results for our problem. Therefore they are used to perform the following tests. ENHSP also offers the possibility to set the search step $\#\mathtt{t}_s$ and the plan validation step $\#\mathtt{t}_v$ to reduce the computational complexity without compromising feasibility. They are set to 150 s and 10 s respectively, unless specified otherwise. All planning processes were carried out with an Intel i7-6700K, 4GHz x 8 processor and 32GB RAM.

Table 1: Remove and insert heuristics used in ALNS

Remove heuristics	**Insert heuristics**
h_{R_1}: randomly remove a task	h_{I_1}: insert a random new task
h_{R_2}: select randomly among the tasks that take the most time to reach	h_{I_2}: insert a task that is closest to the HAPS of which the task is removed
h_{R_3}: select randomly among the tasks that are most likely to be unsatisfactory (due to the bad image quality or onboard memory full)	h_{I_3}: insert a random task that has a better chance of being successful (either a PoI with less dense cloud under or encourage communication of images)
h_{R_4}: select randomly among the tasks positioned in a region densely visited	h_{I_4}: insert a random task that is positioned in a region that is least visited
h_{R_5}: select randomly among the tasks in which the HAPS is required to move against head-winds	h_{I_5}: insert a random task that moves along tail-winds

Validation Tests: Executability of Plans

In this set of tests, we validate the model of the problem, i.e. the correctness of the higher abstraction of the HAPS dynamics in the wind field. We use a third-party 6-DoF HAPS simulator (Müller, Kiam, and Mothes 2018) coupled with a four-dimensional flight controller (Müller and Looye 2013) to keep up with the reference path in terms of vertical and lateral positions, as well as the times of arrival. Historical weather forecast data from COSMO-DE of a summer day in 2015 was used, containing high-resolution (i.e. 2.8 km on the horizontal plane) four-dimensional wind field vectors (Baldauf et al. 2011) and weather-critical convex-polygonal no-go areas marked by the likely presence of Cumulonimbus clouds, strong winds and turbulences (Köhler et al. 2017). Nowcast data was used in the simulator. We consider only weather forecast data that is not too erroneous (i.e. substantial deviation from the nowcast data), in order to isolate the source of error of the HAPS movement model and that of the weather forecast.

The HAPS manages to follow the planner path, while flying at a stable equivalent airspeed around the optimal value of 9 m/s (see Figure 5a). The lateral deviation bounded by 420 m (as observed in Figure 5b) is acceptable, since the no-go areas marking weather critical zones include a margin of at least several kilometers to allow for uncertainties (Köhler et al. 2017). It is however interesting to identify the causes of the lateral deviation, which can be due to 1) the time discretization of 10 s to compute the planned path from the plan trace in ENHSP, as opposed to the integration time step of 1 ms in the flight simulator, 2) difference in the now- and forecast wind, and 3) the substantial turn rate, which can be observed in Figure 5c (i.e. the greater the turn rate is, the harder it is to follow the planned path).

Tuning of the ALNS in the Framework

The efficiency of ALNS in performing the search is affected by the parameter choices of Tp, c, r, and σ_*. As advised by (Ropke and Pisinger 2004), a reasonable setting for Tp is to ensure that the probability of accepting an inferior solution at the start of each segment equals 0.5. The rest are tuned by running multiple tests (20 test runs) for each parameter set, the mean of the objective values is examined to decide for the best parameter setting. Empirical results show that changing a single PoI in a solution seldom result in an improvement of

Success rate of obtaining a plan within 5 s timeout for point-to-point kinodynamic motion planning											
	without obstacle									**with obstacles**	
	Wide area			Fitting area			Narrow area			occ = $[0, 30]$	occ = $]30, 50]$
distance to goal [km]	[0,50]	]50,100]	]100,150]	[0,50]	]50,100]	]100,150]	[0,50]	]50,100]	]100,150]	]100,150]	]100,150]
RRT #t = 30 s	0.02	0.01	0.01	0.88	0.65	0.23	0.92	0.89	0.85	not tested	not tested
RRT #t = 150 s	0.035	0.02	0.01	0.94	0.98	1	0.98	0.89	0.88	0.04	0.05
ENHSP $\#t_s$ = 30 s, $\#t_v$ = 10 s	0.73	0.15	0.02	0.78	0.25	0.11	0.65	0.52	0.48	not tested	not tested
ENHSP $\#t_s$ = 150 s, $\#t_v$ = 10 s	0.95	1	1	0.96	1	1	0.28	0.17	0.06	0.87	0.58

Task + kinodynamic motion planning									
		2 PoIs		5 PoIs			8 PoIs		
		1 HAPS	2 HAPS	1 HAPS	2 HAPS	5 HAPS	1 HAPS	2 HAPS	5 HAPS
ENHSP	success rate	0.2	0.62	0	0	0.4	0	0	0
Standalone	monitored PoIs	2	2	0	0	5	0	0	0
Task planner +	success rate	1	0.9	1	0.9	0.6	1	0.9	0.7
ENHSP (decoupled)	monitored PoIs	1	2	2.4	4.7	5	1.9	6.3	8
ALNS + ENHSP	success rate	1	1	1	1	1	1	1	1
(integrated)	monitored PoIs	2	2	2.4	3.8	5	3.1	6.1	7.1

Table 2: Performance tests 1) to evaluate the planning efficiency of ENHSP as a motion planner for HAPS and 2) to evaluate the integrated task and motion planning architecture in terms of efficiency and executability

the objective value. This effect is rather intuitive, since the overall objective depends a lot on the flight routes connecting PoI. Setting $c = 0.9$ yields empirically the best results, prolonging thus the exploitation of an accepted inferior solution. r is set to 0.7, which is empirically the best setting for the same reason: It allows longer exploitation of an accepted solution. The score increment for the heuristics as described in the “ALNS" section, σ_1, σ_2, σ_3 and σ_4, are set to 0.9, 0.7, 0.1 and 0.3 respectively.

Performance Evaluation Tests

Being aware that using PDDL+ automated planners to solve a problem involving kinodynamic motion planning is novel but also unusual, a first set of performance tests is designed to evaluate the aptitude of this approach, by isolating the motion planning problem and by comparing the performance with RRT from OMPL that can also be used off-the-shelf. Although other motion planning methods could be used, they do not provide an interface that separates the problem definition from the search algorithm. Out of fairness, they were not considered (Hooker 1995), so that the search efficiency is not affected by any workarounds necessary to model the HAPS planning problem. 20 problem files for each test setting are randomly generated (randomly set wind field with wind vectors of speed between 0-3 m/s, and obstacles). In the top part of Table 2 are the success rates to obtain a plan within 5 s to guide the HAPS from a start to a goal position for each planner configuration. Compared to ENSHP, RRT performs comparably in a fitting operation area, in which the distance start-goal is similar to the longest diagonal of the area, or even better in a narrow operation area, in which the transverse dimension of the area is narrow. However, due to its lack of heuristic-guided search, RRT is outperformed by ENHSP in a wide operation area, which is also the kind of setting we aim for (see Figure 1b). In the presence of obstacles, with an obstacle occlusion of 0-30% ($occ = [0, 30]$) or 30-50% ($occ =]30, 50]$), the success rate for ENHSP drops, but remains satisfactory, especially for $occ = [0, 30]$, which is also the usual case in an operation (or the operation will be aborted due to high risk).

The second set of tests is designed to analyse the performance of the implemented framework to cover as many PoIs as possible (without repetition) within $T = 2$ hours; we compare it with the following planning framework:
Standalone PDDL+ planner (ENHSP in GBFS + AIBR setting) that is not wrapped in the framework described by Algorithm 2. Since ENHSP does not support the optimization of a metric, it cannot be tested to cover as many PoIs as possible within a given duration. ENHSP is called once; the goal is reached only when all PoIs are monitored.
Decoupled task planner and ENHSP designed for HAPS in (Kiam et al. 2019). A plan for the symbolic tasks is first determined with a task planner without consideration of the realistic dynamics and the time-varying environment. The task plan is subsequently refined (in terms of time of arrival) using a motion planner. The planners were given 5 minutes to compute plans for a plan duration of 2 hours.

For each planner setting, again 20 test configurations were generated, of each the initial HAPS positions, wind field ($|v_{\text{wind}}| < 3$ m/s) and obstacles ($occ = [0, 30]$) are set randomly. The success rate of finding a plan and the mean number of PoIs covered are shown in the lower part of Table 2. Standalone ENHSP, as a tightly-coupled task+motion planner, cannot scale up (for more HAPS and PoIs). The decoupled, as well as the integrated task+motion planning approaches, scale up well and determine plans of comparable qualities in terms of the number of POIs monitored within 2 hours. Although not shown on the table, the other terms in the objective function, i.e. the cumulative reward, and the homogeneity in PoI distribution found by the decoupled and the integrated task+motion planning are comparable, since both work with the same objective function (see Equation 1).

However, in the "decoupled planning", some task plans can be invalid. Therefore no viable plan is found by the numeric automated planner at the motion planning step.

Discussion and Conclusion

Planning for real-world applications requires often "refinement" of coarsely computed plans, to remove assumptions made in the planning with higher-level actions (Ghallab, Nau, and Traverso 2016). Many applications in robotics (Pecora et al. 2018) and autonomous driving (Srivastava et al. 2014) adopt a "loosely-coupled" architecture: First task planning, followed by refinements (Dantam, Kingston, and Kavraki 2016).

In this paper we investigate the use of planning for controlling a set of HAPS. Similar to other works (Pecora et al. 2018; Srivastava et al. 2014), this problem has previously been dealt with a hierarchical architecture (Kiam and Schulte 2018; Kiam et al. 2019) that decouples the task and motion planning. Task plans are suggested without considering the dynamic or kinematic constraints, and are subsequently refined by a motion planner.

The reason for a segregation lies in 1) the absence of a symbolically and numerically expressive problem modelling language, and 2) the lack of appropriate methods to overcome the challenges in a combined symbolic-numeric search space. A way to address this is by using PDDL+ (Fox and Long 2006) and relative planners (e.g. UPMurphi (DellaPenna et al. 2009), DiNo (Piotrowski et al. 2016), ENHSP (Scala et al. 2016)). Indeed, PDDL+ is a convenient formalism for planning problems involving infinite discrete space (numeric planning), time-stamped actions (temporal planning), continuous processes, exogenous events etc. However, despite the increasing use of domain-independent planners in robotics, they are mostly limited to task planning (Cashmore et al. 2015; Srivastava et al. 2014; Lima, Ventura, and Awaad 2018; Estivill-Castro and Ferrer-Mestres 2013). As far as we know, there is no previous work attempting to use a PDDL+ domain independent planner for the class of problem we deal with.

With the objective of leveraging the expressiveness of PDDL+, we provide a PDDL+ encoding that comprehensively captures the task+motion planning problem for multiple agents, allowing hence compatible automated planners like ENHSP to be used *off-the-shelf* to solve the problem in a tightly-coupled manner. However, due to the difficulty to complement the planner(s) with problem-specific heuristics, the approach suffers from scalability. To overcome this issue, we implemented an ALNS-based high-level planning framework to wrap around the automated planner, in order to use heuristics specially devised for the problem to guide the task assignments. The implementation keeps the search levels intertwined, so that at each search step, even the fine dynamics of the platform in the time-varying environment is considered. Tests show that our integrated task+motion planning approach results in less invalid plans.

The integrated task+motion planning architecture can be employed for other similar problems, requiring however another set of problem-specific heuristics. While this approach is no longer domain-independent, it gives the user the possibility to control the search using problem-specific heuristics, which is necessary for a complex problem.

As a material for future work, the integrated task+motion planning approach can be geared toward a domain-independent framework for off-the-shelf use. This can be achieved with an interface for defining the problem-specific heuristics, as well as by including a generic method to remove actions from the plan and repair, by extending the work by (Siddiqui and Haslum 2015) and (Scala and Torasso 2015) to PDDL+ problems. This is essential to stitch partial plans after each local neighboring search step (see Algorithm 2).

Another dimension worth to be explored is to use a net-benefit encoding of our problem (Keyder and Geffner 2009); this could lead to a crisper formalisation for integrating a more complex objective function into the planning problem. However, how to combine this with our PDDL+ formulation and relative techniques is unclear, but interesting to investigate as a future work.

Acknowledgements

The authors would like to express gratitude to the German Aerospace Center (DLR) for the access to their HAPS simulator and weather data, as well as to the partners of the StraVARIA for the valuable exchanged of ideas and know-how. StraVARIA is a project of Ludwig-Bölkow Campus.

References

Airbus Defence and Space GmbH. 2018. Airbus opens first serial production facility for zephyr high altitude pseudo-satellites.

Allen, R. E., and Pavone, M. 2015. Toward a real-time framework for solving the kinodynamic motion planning problem. In *IEEE International Conference on Robotics and Automation (ICRA)*.

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* 139:3887–3905.

Cashmore, M.; Fox, M.; Long, D.; Magazzeni, D.; Ridder, B.; Carrera, A.; Palomeras, N.; Hurtós, N.; and Carreras, M. 2015. Rosplan: Planning in the robot operating system. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling (ICAPS)*.

Dang, Q.-V.; Rudová, H.; and Nguyen, C. 2019. Adaptive large neighborhood search for scheduling of mobile robots. In *The Genetic and Evolutionary Computation Conference (GECCO)*.

Dantam, N. T.; Kingston, Z. K.; and Kavraki, L. E. 2016. Incremental task and motion planning: A constraint-based approach. In *Robotics: Science and System*.

De Filippis, L., and Guglieri, G. 2012. *Advanced Graph Search Algorithms for Path Planning of Flight Vehicles, Recent Advances in Aircraft Technology, Dr. Ramesh Agarwal (Ed.)*. InTech.

DellaPenna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: a tool for universal planning on PDDL+

problems. In *In Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*.

Donald, B.; Xavier, P.; Canny, J.; and Reif, J. 1993. Kinodynamic motion planning. *Journal of the ACM* 40:1048–1066.

Estivill-Castro, V., and Ferrer-Mestres, J. 2013. Path-finding in dynamic environments with pddl-planners. In *16th International Conference on Advanced Robotics, ICAR*.

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 belgium. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science, ISPRS Zurich 2011 Workshop,*.

Fox, M., and Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *JAIR* 20:61–124.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *JAIR* 27:235–297.

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press.

Hooker, J. N. 1995. Testing heuristics: We have it all wrong. *Journal of Heuristics* 1:33–42.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36:1547–556.

Kiam, J. J., and Schulte, A. 2018. Multilateral mission planning in a vector field with dynamic constraints. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*.

Kiam, J. J.; Besada-Portas, E.; Hehtke, V.; and Schulte, A. 2019. GA-guided task planning for multiple-haps in realistic time-varying operation environments. In *The Genetic and Evolutionary Computation Conference (GECCO)*.

Kiam, J. J.; Schulte, A.; and Scala, E. 2019. Using AI-planning to solve a kinodynamic path planning problem and its application for haps. In *Intelligent Human Systems Integration (IHSI)*.

Köhler, M.; Funk, F.; Gerz, T.; Mothes, F.; and Stenzel, E. 2017. Comprehensive weather situation map based on xml-format as decision support for uavs. *Journal of Unmanned System Technology* 5:13–23.

Köhler, M.; Gerz, T.; and Tafferner, A. 2016. Cb-like-cumulonimbus likelihood: Thunderstorm forecasting with fuzzy logic. *Meteorologische Zeitschrift* 25:1–19.

Lima, O.; Ventura, R.; and Awaad, I. 2018. Integrating classical planning and real robots in industrial and service robotics domains. In *Proceedings of the 6th Workshop on Planning and Robotics (PlanRob) at 28th ICAPS*.

McDermott, D. 2000. The 1998 AI planning systems competition. *AI Magazine* 21(2).

Müller, R., and Looye, G. 2013. A constrained inverse modeling approach for trajectory optimization. In *AIAA Guidance Navigation and Control Conference*.

Müller, R.; Kiam, J. J.; and Mothes, F. 2018. Multiphysical simulation of a semi-autonomous solar powered high altitude pseudo-satellite. In *IEEE Aerospace Conference*.

Pecora, F.; Andreasson, H.; Mansouri, M.; and Petkov, V. 2018. A loosely-coupled approach for multi-robot coordination, motion planning and control. In *International Conference on Automated Planning and Scheduling (ICAPS)*.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic planning for PDDL+ domains. In *Proc. of IJCAI*.

Ropke, S., and Pisinger, D. 2004. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Technical Report no. 2004/13* 13:1–27.

Scala, E., and Torasso, P. 2015. Deordering and numeric macro actions for plan repair. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Scala, E.; Haslum, P.; Thiebaux, S.; and Ramirez, M. 2016. Interval-based relaxation for general numeric planning. In *European Conference on Artificial Intelligence (ECAI)*.

Siddiqui, F. H., and Haslum, P. 2015. Continuing plan quality optimisation. *Journal of Artificial Intelligence Research* 54:369–435.

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. In *IEEE International Conference in Robotics and Automation (ICRA)*. Hong Kong, China: IEEE.

Sucan, I. A.; Moll, M.; and Kavraki, L. E. 2012. The open motion planning library. In *IEEE Robotics and Automation Magazine*.

Webb, D., and van den Berg, J. 2013. Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints. In *IEEE International Conference on Robotics and Automation (ICRA)*.