Optimal Motion Paths in Ambient Fields

Thomas Kämpke
InMach Intelligente Maschinen GmbH
Kässbohrerstr. 19, 89077 Ulm, Germany
E-mail: kaempke@inmach.de, www.inmach.de

Keywords: autonomous mobile platforms, graph algorithms, energy minimization

Received: July 3, 2012

The motion of water or the atmosphere that surrounds or supports a mobile platform affects the motion of the platform in a favorable or unfavorable way. Exploiting and compensating the ambient field is investigated for different motion objectives like the classical point to point motion, observation motion, exploration motion and so-called survival motion. Time and energy optimal trajectories in the presence of fields are formalized and shown to be computable by means of discrete optimization.

Povzetek: Analizirano je gibanje mobilne platforme v okolju.

1 Introduction

Mobile platforms that float, dive or fly are affected by the motion of their ambient fields. These fields may be wind, oceanic streams like the Gulf and the Humboldt stream, tidal or diurnal motions, currents and arbitrary combinations thereof. The fields may be stationary or nonstationary.

Platforms considered here include autonomous sailing ships, commercial vessels that receive additional propulsion from airfoils such as skysails [14], (gliding) planes and aerobots designed for the Earth or for a foreign planet with an atmosphere such Venus and Mars as well as Titan (moon of Saturn) and Europe (moon of Jupiter). The motion of a platform is abstracted here from almost all kinematic and dynamic constraints, so that neither inertia nor restrictions on the control variables like bounds on turning angles apply. This allows, in particular, to ignore the motion history for computing any motion continuation. Uncontrolled motion (drift) is considered separately from motion with controls.

Key issues for trajectories without control are reachability of a destination point by pure drift and the computability of the distance of pure drift towards a destination point. Key issues for trajectories with controls are computations of minimum time and minimum energy trajectories towards destination points, optimal trajectories according to the two foregoing issues towards a destination region, minimum energy trajectories towards a destination point to meet a given due date, optimal orbits to continuously observe a stationary point of interest, optimal trajectories to explore a given region of interest and longest survival orbits allowing a flying platform to stay airborne with a fixed energy budget for a maximum time.

Horizontal motion will be considered first, followed by straightforward as well as by a particular extension to three dimensions. The field is assumed to be known throughout. Thus, no uncertainty of the field is admitted, neither in strength nor in direction. This amounts to deterministic path planning rather than handling the unexpected like motion adaptation for collision avoidance [15] or even managing completely unforeseeable environmental changes. Also, selflocalization, which is the determination of the position platform in some reference frame, is not part of this work.

Time and energy minimization of trajectories will be obtained by discrete optimization methods, in particular by graph algorithms. The majority of the motion objectives which are introduced below, go beyond point to point motion and use this well-known problem as starter. Optimal trajectories for two-dimensional motion and some extensions to the 3D case of homogenous, vertically stacked wind layers were investigated in [5]. Optimal control of hot-air balloons is considered for so-called linear wind fields in [3]. A 3D wind field is linear if it changes linearly with the position. For large scales, this model does not apply for two reasons. First, the wind speed does typically not grow linearly in the vertical component. Second, the Coriolis force, terrain effects and else cause a wind field to be curved and even be locally turbulent.

A sophisticated analysis of continuous 2D flows and potentially emerging cyclic structures is given in [13]. The cyclic structures allow orbits, but no platform motion is considered. Platforms may execute controls that exploit the fields but they may even oppose the field, at least along part of the trajectory. The case of exploiting the wind field for sailing ships has been covered by several references including [1].

The remainder of this paper is organized as follows. Sections 2, 3 and 4 cover problems in two dimensions at increasing complexity levels. The main distinction is that between drift and deliberate motion. Section 5 deals with point to point motion under different and joint objectives. Section 6 considers motion for continuous observation of a

point location. Section 7 deals with exploration of an area. Section 8 extends approaches to three dimensions and considers a so-called survival problem in systems of updraft and downdraft areas.

2 Basic Model

The platform position at any moment is a 2D point $P(t) = (x(t), y(t))^T$ with the superscript T denoting transposition. Start and destination points of a motion are abbreviated by $P_S = (x_S, y_S)^T$ and $P_D = (x_D, y_D)^T$. The scale of considerations is chosen so (large) that platform orientation does not matter. Positions at future moments resulting only from drift by the ambient fields are given by the forward equation $P(t + \Delta t) = P(t) + \Delta t \cdot w(P(t), t) + \Delta t \cdot c(P(t), t)$. The effects of all ambient fields are linearized over time.

Time increments Δt range from a fraction of a second to many minutes and even to six hours for vessel routing [12]. The wind field w, the current field c and possible further fields are dealt with in the same way. Fields are vector fields $Dom \to I\!R^2$ in the stationary case and vector fields $Dom \times T \to I\!R^2$ in the nonstationary case with $Dom \subseteq I\!R^2$ in both cases. T is the index set for time with typical setting $T \subseteq [0, \infty)$. The drift caused by a vector field may eventually lead to positions where the field is not defined. This view circumvents the specification of domain bounds. Fields are assumed to generously cover considered regions so that boundary problems are practically irrelevant.

Fields can be specified discretely in space and, if applicable, in time with hyperbolic spatial interpolation. This means that the field at some point P in the convex hull of, say, four support points $P_1, \ldots, P_4$ is

$$f(P) = \sum_{i=1}^{4} \frac{\frac{1}{\|P-P_i\|}}{\frac{1}{\|P-P_1\|} + \ldots + \frac{1}{\|P-P_4\|}} f(P_i)$$

in the stationary case and

$$f(P,t) = \sum_{i=1}^{4} \frac{\frac{1}{\|P-P_i\|}}{\frac{1}{\|P-P_1\|} + \ldots + \frac{1}{\|P-P_4\|}} f(P_i,t)$$

in the nonstationary case. The effect of one support point on a point of interest is inversely proportional to its relative distance from that support point. A feature of this interpolation is that it is also operational if the point of interest lies to the outside of the convex hull of the support points. Discontinuities may occur when the set of considered support points changes along a trajectory but this applies to other interpolation schemes as well. Interpolation in time is always linear between two adjacent support moments $t_1 < t_2$. For any moment t between the two support moments, time interpolation is

$$\begin{aligned} f(P,t) &= \frac{t_2 - t}{t_2 - t_1} f(P,t_1) + \frac{t - t_1}{t_2 - t_1} f(P,t_2) \\ &= \frac{\frac{1}{t-t_1}}{\frac{1}{t-t_1} + \frac{1}{t_2-t}} f(P,t_1) + \frac{\frac{1}{t_2-t}}{\frac{1}{t-t_1} + \frac{1}{t_2-t}} f(P,t_2). \end{aligned}$$

3 Motion Without Control

Natural fields may be curved so that a drifting platform may come closer to a destination point, then increase its distance and repeat this behavior. A field with two distance minima is sketched in figure 1. A refined analysis reveals that a second approach towards the destination point can occur even if the cumulative angular change along all drift trajectories is less than 90^o. The existence of more complicated behaviors cannot be excluded completely but is ignored here. Yet fields in which optimal trajectories may spiral have been investigated [9]. The closest approximation to a destination point is computable by tracing the platform drift until distances begin to increase for the second time or until a boundary of the field is reached; whatever is first. Passing through the destination point amounts to shortest distance of value zero. The shortest distance between the destination and a drift trajectory can be denoted as drift distance. This is not a regular distance, because it is not even symmetric. The drift distance may serve as guideline for trajectory computations since it is a shortest segment along which the platform requires to invoke propulsion.

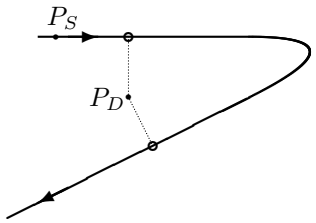

Figure 1: Drift trajectory with two local distance minima (white points) to the destination point.

4 Control Model

4.1 Structure

The position update for motion with control is $P(t+\Delta t) = P(t)+\Delta t \cdot f(P(t))+\Delta t \cdot u(t)$ for a stationary field $f(\cdot)$ and $P(t + \Delta t) = P(t) + \Delta t \cdot f(P(t), t) + \Delta t \cdot u(t)$ for a nonstationary field $f(\cdot, \cdot)$. $u(t)$ is the control vector executed at time t and held fixed over the time increment. Control vectors specify heading and speed relative to the field and, depending on the type of platform, may head into the opposite direction of the field. The effect of the field along the transition during the time increment is, so far, attributed only to the emanation point of the transition. Since the field typically varies along the transition, its effect should be inferred from the field along the transition or, at least, from the first and the last point of the transition. Thus, for $P = P(t)$ and $Q = P(t + \Delta t)$ the effect of the field is

$$\frac{f(P) + f(Q)}{2} \text{ or } \frac{f(P,t) + f(Q, t + \Delta t)}{2}.$$

In the stationary case, the control vector for reaching Q after the time increment is

$$u(t) = \frac{P(t+\Delta t) - P(t)}{\Delta t} - \frac{f(P) + f(Q)}{2};$$

the formula for nonstationary fields is analogous. Controls and fields need not superimpose linearly. The execution of control vectors entails cost that are integrated over time until the destination point is reached. Following standard control approaches with some performance function $cost : \mathbb{R}^2 \to \mathbb{R}_{\geq}$, the total cost is

$$C = \int_{t_S}^{t_D} cost(u(t))\, dt.$$

The starting time is assumed to be known while the time of arrival in the destination point depends on the motion which depends on the sequence of controls. Prominent examples of cost functions are $cost(u(t)) = 1$ with total cost accounting for the time and $cost(u(t)) = power(u(t))$ with total cost accounting for the energy spent until the destination point is reached. The function $power(\cdot)$ encodes the physical energy spent per time to execute the control. Executing no control incurs no cost. The aim of finding an optimal motion is a minimization problem over the set of feasible controls $\mathcal{U}$

$$\min_{u(t)\in\mathcal{U}} \int_{t_S}^{t_D} cost(u(t))\, dt.$$

The minimization may be endowed with a further constraint so that trajectories meet a due date

$$\begin{array}{ll} \min_{u(t)\in\mathcal{U}} & \int_{t_S}^{t_D} cost(u(t))\, dt \\ \text{such that} & t_D \leq t_{due}. \end{array}$$

Trajectory optimization in its most general form, thus, is a variational problem so that the ultimate method is solving the Euler-Lagrange equation (computing the "zero" of the derivative). But the difficulties of finding an exact solution are enormous as illustrated, for example, by fields that force a mobile platform to make abrupt turns [10]. Even the most simple problem version is not trivial when the cost function is not trivial. This is illustrated for the field being zero everywhere. The cost minimal trajectory then connects the starting point to the destination point by a straight line leaving the optimal transition time to be computed as a univariate minimization problem. The controls are set to

$$u(t_S) = \frac{P_D - P_S}{t_D - t_S}$$

with the arrival time in the destination point being in variation. The energy spent along the trajectory is

$$\begin{aligned} C &= \int_{t_S}^{t_D} cost(\frac{P_D - P_S}{t_D - t_S})\, dt \\ &= (t_D - t_S) \cdot cost(\frac{P_D - P_S}{t_D - t_S}). \end{aligned}$$

The univariate minimization problem with variable t_D thus tentatively becomes

$$\min_{t_D : t_D > t_S} (t_D - t_S) \cdot cost(\frac{P_D - P_S}{t_D - t_S}).$$

Transition times between distinct positions cannot be arbitrarily small. If the minimum transition time from P_S to P_D admitted by the controls is $t_{min} > 0$, then the final optimization problem becomes

$$\min_{t_D : t_D \geq t_S + t_{min}} (t_D - t_S) \cdot cost(\frac{P_D - P_S}{t_D - t_S}).$$

The solution strongly depends on the cost function. Minimum energy solutions may have the "single crossing property" which describes a certain uniformity of the field in relation to a given trajectory and which is stated here without formal proof. This property claims that an energy minimal trajectory crosses every drift trajectory at most once, if all vectors of the ambient field on every normal of the energy minimal trajectory point into the halfspace which contains the destination point. This property is illustrated by figure 2.

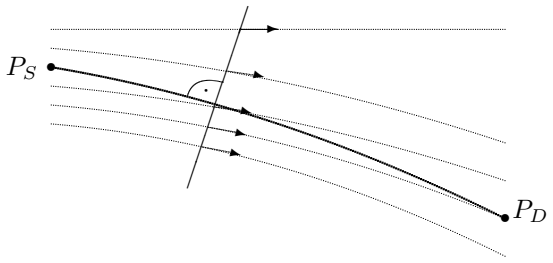

Figure 2: Energy optimal trajectory with one normal and five drift trajectories shown. The field at the intersections of the normal with all five trajectories points into the same ("lower right") halfspace of the normal as is supposed for all non-visualized drift trajectories. The destination point lies in that halfspace.

The single crossing property is not a necessary optimality criterion as it may or may not hold in heavily turning fields such as in figure 1. Also, the property does not hold if the destination point lies too far upstream. The energy minimal trajectory may then have to slalom around regions of strongly opposing drift thus crossing at least one drift trajectory more than once, see figure 3.

4.2 Computational preliminaries

Computations of optimal trajectories follow two principal approaches. The first is put a grid into the region of interest and compute motion for grid points only. The spatial resolution of the grid can be selected independently from the spatial resolution of the fields. An interesting, non-uniform grid has been derived for long distance vessel routing [7].

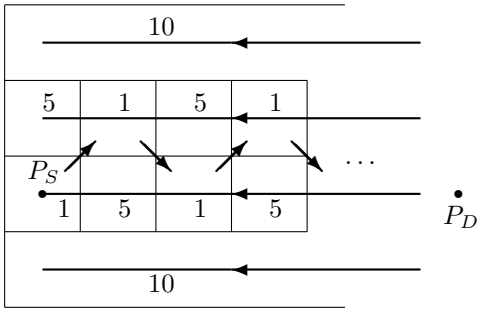

Figure 3: Numbers and arrows indicate field velocity and direction in the rectangles. An energy minimal trajectory of a platform with suitable characteristics will multiply cross drift trajectories (slanted arrows).

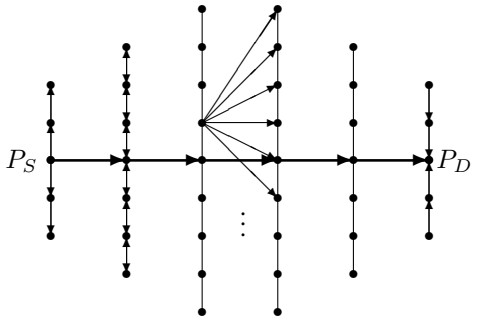

Figure 4: Static waypoint graph with herringbone pattern. Arcs reach from all points of a bone to all points of the next bone until the bone with destination point is reached. Only five of the nine arcs reaching from one node to the subsequent bone are indicated. Also, arcs within a bone are only indicated for three bones.

The nodes are placed along the great circle between start and the destination point and perpendicular to the great circle up to a specified width. Alternatively, the motion is free meaning that waypoints are not set before computations. This is the dynamic graph approach and dynamically placed nodes have been proposed for motion planning by splines [4].

Static and dynamic graphs are directed and denoted as waypoint graphs. Static waypoint graphs allow computations by standard graph algorithms like the Dijkstra algorithm and dynamic programming. The disadvantage is that an optimal path must, typically, be smoothed to become a real-world path. Dynamic waypoint graphs allow to place nodes in control-compliant positions. But then, computations may suffer from the number of nodes growing at an unmanageable rate which puts heavy burden on elimination techniques.

The static graph is the concept of choice here. A static waypoint graph may be defined universally for a region or it may be defined specifically for the starting and destination points as in figures 4 and 11. Several points may lie at different distances in the same direction from some waypoint P_0 which is not intuitive for neighbors. This phenomenon is excluded by requiring that only the closest point in a given direction belongs to any neighbor set [17]. Moreover, far points are excluded from that set if the direction of the field deviates beyond a threshold angle from the direction in P_0. The set of all waypoints is denoted by V.

Arc labels for waypoint graphs depends on characteristics of the platform. When the objective is time, velocity predictions and interpolations from so-called polar plots may apply [1]. Computations of energy values are illustrated by concrete data and figure 5. The distance from A to B is 10 nautical miles with current setting in orthogonal direction at 3 knots (nautical miles per hour). When traveling at an apparent speed of 5 (8) knots, fuel consumption is 10 (21) liters per hour. Only the two speeds are allowed. In the first case, ground speed towards B is 4 knots, so that the transition time from A to B is $10nm/(4nm/h) = 2.5h$ and fuel consumption is $10l/h \cdot 2.5h = 25l$. In the second case, ground speed towards B is $\sqrt{55} = 7.416$ knots, so that the transition time from A to B is $10nm/(7.416nm/h) = 1.348h$ and fuel consumption is $21l/h \cdot 1.348h = 28.31l$. The better of the two options results in the energy label 25 for the arc from A to B.

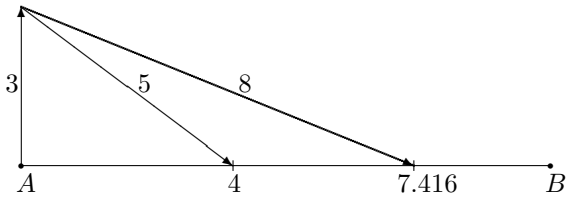

Figure 5: Vector additions for sample energy computations.

Transition costs along an arc (A, B) are specified by a label $c(A, B)$ for stationary fields and by parameterized labels $c(A, B; t)$ for nonstationary fields with t denoting the moment when the platform leaves A and heads for B. Two waypoints may or may not be connected by opposing arcs. If so, their labels may be different.

5 Point Motion with Controls

5.1 Single objective for point to point motion

Paths will preferably be computed by the Dijkstra algorithm and modifications thereof instead by dynamic programming. Though optimal paths satisfy the dynamic principle [2], the Dijkstra algorithm proceeds along forward computations while standard dynamic programming applies backward computations. Backward computations are not intuitive for nonstationary fields; the field is more likely

to be known at starting conditions than at terminating conditions since the latter, among others, depend on the computed trajectory.

The Dijkstra algorithm in standard form [6] suffices for shortest path computations in stationary fields. For non-stationary fields the algorithm is formulated for transitions along arcs without idleness. This means that the platform will never wait for more opportunistic conditions of the field; the arrival time in an intermediate waypoint is identical to the departure time in that waypoint. The departure time from the starting node is $t_{P_S} \geq 0$ and arrival times at nodes are specified by the labels $m(\cdot)$.

Dt (Dijkstra algorithm for nonstationary fields)

1. (Initialization). Set $L = V$, $m(l) = \infty \; \forall \, l \in V - \{P_S\}$, and $m(P_S) = t_{P_S}$.
2. (Iteration). While $P_D \in L$ do:
   (a) If $min_{l \in L} \, m(l) < \infty$ then selection of $i = argmin_{l \in L} \, m(l)$, else output "Destination not reachable" and stop.
   (b) $L = L - \{i\}$.
   (c) $\forall \, j \in L$ with $(i, j) \in A$ do:
   if $m(i) + c(i, j; m(i)) < m(j)$, then $m(j) = m(i) + c(i, j; m(i))$ and $pred(j) = i$.
3. (Termination). Output $m(P_D)$ and $P_D, pred(P_D), pred(pred(P_D)), \ldots, pred(\ldots (P_D) \ldots) = P_S$.

An optimal path from start to destination results from tracing the waypoints which attain minima of the node labels. These labels denote the cost of the best path found so far from the starting point. The list L consists of the waypoints to which an optimal path has not yet been found or has not yet been confirmed to be found. These waypoints are tentatively labeled, while all others are permanently labeled. Preceding vertices are stored in the $pred(\cdot)$ function so that the waypoints of an optimal motion are specified reversely.

Replacing the time dependent labels $c(i, j; t)$ by time independent labels $c(i, j)$ results in the ordinary Dijkstra algorithm. The Dijkstra algorithm for both label kinds is a one-to-many algorithm. This means that the algorithm may find optimal paths from the initial node to several nodes as discussed next.

5.2 Single objective for point to region motion

Computations of optimal paths from the starting point to all other waypoints can be facilitated by a single run of algorithm **Dt** by modifying the stopping criterion as to the list L being empty. Optimal paths to each node of a selected region of waypoints are computed when the list is cleared of all target nodes. A variation of the last problem is to reach only one waypoint in a given region, namely the one which is reachable at minimal cost. This point need not be known prior to the computations.

An interesting case is that of the region consisting of all waypoints from which the destination point is observable. The computed motion is then one from which the destination point comes earliest "in sight", see figure 6. If observability solely depends on the Euclidean distance to the destination point, the optimization problem is formalized as

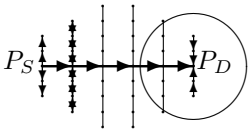

Figure 6: Sketch of waypoint graph from figure 4 and a circle around the destination point from which the destination point is observable.

$$\min_{i \in V: \, \|i - P_D\| \leq d} \; m(i);$$

$m(i)$ is the permanent label assigned to waypoint i by algorithm **Dt** and the maximum observation distance $d > 0$ must be specified as input. The region from which to reach a waypoint is the observability region $O_D(d) = \{i \in V : \|i - P_D\| \leq d\}$. The cost function for reaching the region of observability may differ from that for motion within that region. For example, it may be appropriate to reach the region of observability in minimal time but then to finally approach the destination with minimum energy.

5.3 Multiple objectives for point to point motion

Energy minimization subject to a due date can be handled as a two-criteria minimization problem. Such problems are solvable by a variation of the Dijkstra algorithm which operates on sets of labels for arcs and nodes instead of single labels. For the sake of simplicity, we consider only finite sets of arc labels. The "classical" multicriteria shortest path problem with one vector-valued label per arc already is NP-hard. But, occasionally, it is considered as one of the least intractable problems [16].

The arc (i, j) receives paired labels of the form $(d(i, j; t), e(i, j; t))$. The value $d(i, j; t)$ is a feasible duration of the transition along the arc while the transition begins at time t. The complete energy required therefore is $e(i, j; t)$. Longer durations correspond to smaller energy values. Practically, very long durations may again lead to larger energy values as engines consume fuel even if the platform does not move or move at extremely low speed. Such combinations of time and energy are not considered for the purpose of optimization. Thus, the labels of each arc form a Pareto optimal set which means that

an improvement in one coordinate can only be achieved by a deterioration of the other. All feasible labels are arranged in the list $\Lambda(i,j;t)$ with time dependency disappearing in the stationary case. An example is $\Lambda(2,7) = \{(10,95),(11,91),(13,80)\}$. The transition from waypoint 2 to waypoint 7 may take 10 time units requiring 95 energy units or 11 time units requiring 91 energy units etc. For convenience all these lists are sorted by increasing time.

A simple approximation of an energy minimal path with due date in a stationary field is to compute a shortest path and then to maintain the sequence of waypoints while successively allowing more time for arc transitions. These relaxations, which conserve energy, terminate if no further prolongation is admitted.

Relaxations are selected along maximal energy savings per additional transition time. Therefore, $\Delta t(t(i,j)) = next(t(i,j)) - t(i,j)$ denotes the time increment for the duration $t(i,j)$ being replaced by the next value in $\Lambda(i,j)$. Similarly, the increment of saved energy is denoted as $\Delta e(e(i,j)) = e(i,j) - next(e(i,j))$. Both increments are positive due to sorting. Each relaxation then picks an arc attaining the maximum of the savings ratio provided that the due date is still met. Formally, a relaxation amounts to

$$\max_{(i,j)\in\mathcal{P}} \{\frac{\Delta e(e(i,j))}{\Delta t(t(i,j))} \mid t_D + \Delta t(t(i,j)) \leq t_{due}\},$$

where $\mathcal{P}$ is an arc-wise specified path from P_S to P_D. The arrival time at the destination point is updated after each relaxation by $t_D^{new} = t_D + \Delta t(t(i,j))$ and $t_D \leftarrow t_D^{new}$. The foregoing procedure is of a greedy type and, therefore, need not find the energy minimum even along the given path.

An exact algorithm for energy minimization under due dates requires a quite extensive modification of the Dijkstra algorithm. First, nodes will receive sets of labels in the same way as arcs have received sets of labels prior to the start of the algorithm. Second, sets of labels will be propagated. Yet the strongest modification is concerned with permanence declarations of node labels. It is not correct – in a straightforward adaptation of the original Dijkstra method – to declare that tentatively labeled node as the next one permanently labeled which carries the minimum energy value. This wrongful propagation is illustrated in figure 7. It calls for decoupling of label propagation and permanence declarations. For stationary fields, label propagation along one arc (i,j) proceeds along the following three steps which may have to be executed multiply.

PL (Propagation of Labels)

1. $L'(j) = L(i) \oplus \Lambda(i,j) = \{((d_1,e_1) + (d_2,e_2),i) \mid (d_1,e_1) \in L(i) \text{ and } (d_2,e_2) \in \Lambda(i,j)\}$.
2. $L''(j) = L(j) \cup L'(j)$.
3. $L(j) = \text{Pareto boundary}(L''(j))$.

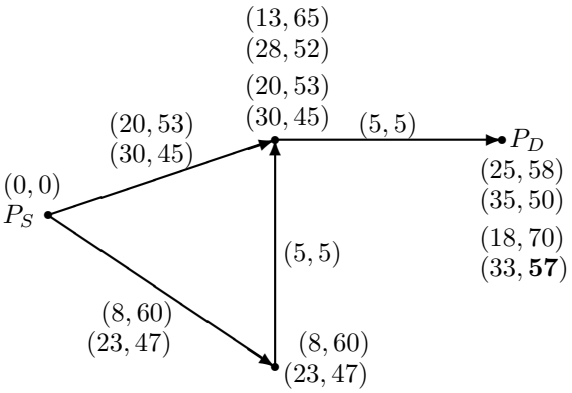

Figure 7: Initially, any reasonable computing scheme will label both successors of the starting point with the label sets of the two emanating arcs. The minimum energy value over the four pairs is 45. So, selecting the upper intermediate waypoint as the next one permanently labeled – by an oversimplified propagation scheme – results in only two label pairs for the destination point; one with arrival time 25 and another with 35. For due date $t_{due} = 34$ only label $(25,58)$ remains. But propagation from the 'lowest' node gives the energy minimal solution taking time 33 and requiring only 57 energy units.

Each label has an additional coordinate which will eventually allow to identify a sequence of predecessors. This information cannot be attached – in contrast to the single criterion case – to the nodes themselves because different pairs of objective values may require different paths to attain them. The additional coordinate is occasionally omitted for easy of notation and the starting node does not have an additional coordinate. The first step of **PL** is the propagation step in which all arc labels are added to all node labels. The second steps unites old and new node labels and third step cleans them up. It deletes all labels for which there is another label which has a smaller duration as well as a smaller energy value.

The overall algorithm works with a list T of tentative node labels instead of a list of tentative nodes. Labels from this list do not loose their node assignments. This allows multiple propagations from a node during one run of the subsequent labeling algorithm. When the list of tentative labels becomes void, the best energy value is selected from all labels of the destination node whose time coordinate meets the due date.

E-due (Energy with due dates)

1. (Initialization). Set $T = (0,0)$ with $(0,0)$ belonging to $P_S = 1$.
2. (Iteration). While $T \neq \emptyset$ do:
   (a) Selection of lexicographically smallest label (d_0,e_0) from T belonging to node $i \in V$.

(b) $T = T - \{(d_0, e_0)\}$.

(c) $\forall\, j \in V$ with $(i, j) \in A$ do:

i. update node label set $L(j)$ by **PL**.

ii. $T = T \cup L(j)$.

iii. Deletion of all labels of T belonging to j that are dominated by other labels of T that also belong to j.

3. (Termination). Output $e(P_D) = \min\{e|\,(d, e) \in L(P_D)$ and $d \leq t_{due}\}$ and optimal path traced by labels.

Though the multicriteria shortest path problem is symmetric as it considers both objectives as equally important, even in the present case of constrained energy minimization, it introduces a lexicographic order of the objectives. This break of symmetry affects the operations but not on the result. Considering time as the more important criterion renders a depth-first search behavior to the algorithm. Considering energy as the more important criterion renders a breadth-first search behavior to the algorithm.

Several technical improvements are possible. When a label generated in step 2(c)i. has an arrival time that exceeds the due date, this label can trivially be omitted. Also, a label of some node that is dominated by a label of the destination node can be omitted. The method is illustrated in figure 8.

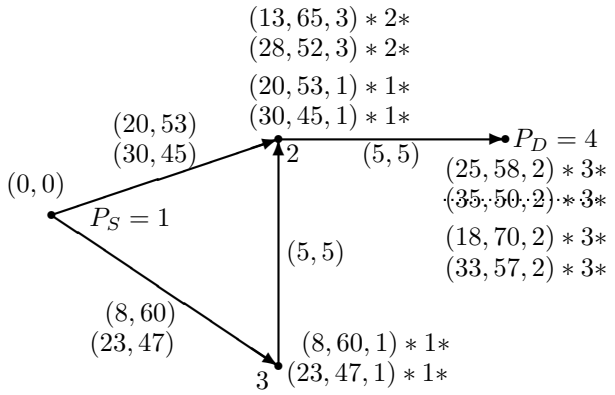

Figure 8: Labels set by the multicriteria Dijkstra algorithm for the problem from figure 7. In addition, nodes are numbered and predecessors are specified by these numbers as third coordinates of the node labels. Starred numbers indicate the iteration of step 2 in which the node labels are generated. The second label of the destination node is deleted because the due date $t_{due} = 34$ is violated. At most nine iterations of step 2 empty the tentative label set T. The node label $(8, 60)$ is the first to be deleted from T, $(13, 65)$ is the second etc. The minimum energy value belonging to feasible arrival times is $57 = \min\{58, 70, 57\}$.

6 Observability Motion

Instead of reaching a destination point, coming (and staying) close may suffice for observation. Staying close is specified by a region of observability RoO around the destination point so that the objective becomes to uninterruptedly stay within the RoO at minimum energy use. If a time bound for the observation is lacking, the formal objective becomes to indefinitely stay in the RoO at minimum energy use per time. Once the RoO has been reached, the compensation of drift caused by the ambient field is considered to be the only reason for energy consumption. The RoO may be circular as in figure 6, but other shapes are feasible. The essential trade-off for observability motions is that between staying in one position while continuously consuming little energy vs. drifting for some time at no energy consumption, moving upstream at some energy consumption and drifting again etc. Depending on the characteristics of the platform and the ambient field, the second strategy, called orbiting or oscillating, may be superior.

As an example, a ship may remain still somewhere in a RoO see figure 9. This requires to continuously compensate the drift which is assumed to be constant at two knots throughout the field. Fuel consumption for drift compensation is assumed to equal 5l/h. When the distance between A and B is 10 nautical miles and the platform initially is located at A, it may drift five hours at zero consumption and then travel back to A at an assumed speed of six knots through the water at an assumed fuel consumption of 10l/h. So, it takes 10nm/(6 knots - 2 knots) = 2.5h to travel from B to A thereby consuming 25l fuel. Thus, the average consumption over one orbit is $25l/7.5h = 3.333l/h$ which is less than the consumption for immobility.

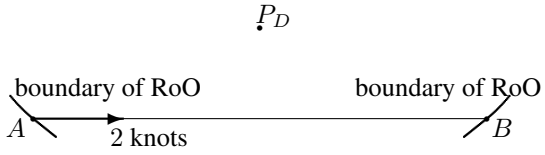

Figure 9: Region of observability for P_D and drift in a constant field from boundary point A to boundary point B. Upon arrival in B, the platform travels upstream to A where it resumes drifting.

Boundary points of a RoO at which the field directs into the interior of the RoO are denoted as entry points (of the RoO) and boundary points at which the field directs to the outside are denoted as exit points (of the RoO). An orbit is a closed trajectory from an entry point to an exit point and back. A drift orbit is an orbit of which the partial trajectory from the entry to the exit point is a mere drift trajectory. The trajectory from the entry point to the exit point may be different from the inverse trajectory back to the entry point. The situation from figure 9 is special in this respect.

When the observation time is unbounded, an optimal orbit or, more precisely, a minimum average power (MAP)

orbit adheres to the fractional program

$$\min_{O \in \mathcal{O}} \frac{energy(O)}{time(O)}.$$

The orbits O range through some set $\mathcal{O}$ of orbits that may be finite or infinite. In the finite case, a MAP drift orbit can be approximated by the following two steps:

1. (Drift trajectory) For each of the entry points compute the drift trajectory according to the forward equation $P(t + \Delta t) = P(t) + \Delta t \cdot f(P(t))$ until an exit point is reached. A drift trajectory with maximum duration is selected.

2. (Partial trajectory back to entry point) An energy minimal path is computed by algorithm **Dt** from the exit point of the selected drift trajectory to its entry point.

Minimum average power orbits need not be of a drift type. "Jockeying" between drift trajectories, which consumes energy, can be favorable if the field changes differently along different drift trajectories, figure 10.

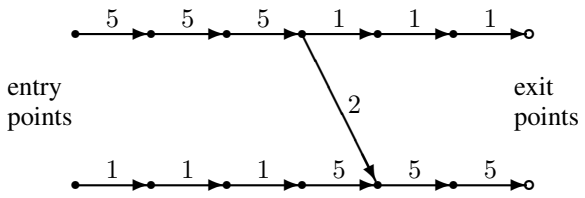

Figure 10: Two drift trajectories with labels indicating time. Maximizing drift time requires to switch from the upper to the lower trajectory. The return trajectory to the upper entry point is omitted.

A MAP orbit which is not necessarily of a drift type can be approximated by concatenations of minimum energy paths from entry points to exit points and back. As orbits are cyclic, this order can be reversed so that an exit point becomes the starting point of an orbit. When a platform arrives at a RoO it may do so either in an entry point or in an exit point and the beginning of an orbit is chosen accordingly. If the best computed orbit does not pass through the arrival point, the arrival point will connected to the best orbit. One way to do so is to compute a minimum energy path from the arrival point to the orbit by the Dijkstra algorithm with all orbit points forming a destination region. The orbit computation itself is as follows.

MAP-orbit (Minimum Average Power-orbit approximation)

1. Input RoO and finite set En of entry points.
   (Initialization). For each $a \in En$ compute its exit point $ex(a)$ reached by mere drift. All these exit points are summarized as exit set $Ex = \{ex(a) | \, a \in En\}$.

2. (Computations).
   (a) For all $a \in En$ and all $b \in Ex$ computation of a minimum energy path $P(a, b)$ from a to b.
   (b) For all $b \in Ex$ and all $a \in En$ computation of a minimum energy path $Q(b, a)$ from b to a.
   (c) For all $a \in En$ and all $b \in Ex$ concatenate minimum energy paths to form orbits through a and b as $O(a, b) = P(a, b) \circ Q(b, a)$.
   (d) $O_{MAP} = argmin_{O(a,b),\, a \in En,\, b \in Ex} \frac{energy(O(a,b))}{time(O(a,b))}$.

3. (Termination). Output O_{MAP}.

Algorithm **Dt** can be used for the path computations in steps 2 (a) and (b) and durations are recorded for all paths so that the cycle times of all orbits are known in step 2 (d). The computations of algorithm **MAP-orbit** for all paths emanating from the same entry and same exit point can be interleaved. The formation of orbits from paths is obvious by aligning the waypoints of one path behind the other while avoiding immediate repetitions of the exit and entry points. Path alignment is denoted by the concatenation operation $\circ$.

An approximation of a MAP orbit without concatenation can be facilitated by integrating the two computing stages of **MAP-orbit**. Essentially, this requires to provide a double waypoint graph as sketched in figure 11. The first part of the waypoint graph allows approximations of energy minimal paths from entry to exit points and the second graph allows for the inverse. The structure of the second graph can be obtained by reflecting the first graph, but arc labels may be completely different. In order to admit a minimum path computation, a hypothetical source node s is connected at zero cost to all entry nodes and the copy of the entry nodes is connected at zero cost to a hypothetical sink node t.

An energy minimal path approximates a MAP orbit. If such a path uses the same entry point in both copies of the entry set, then the path actually amounts to an orbit. If not, a sequence of orbit-like paths is better than a single orbit traced repeatedly. Instances of the waypoint graph can then be added to result in a fourfold waypoint graph, a sixfold waypoint graph etc. A shortest path from source to sink then yields an optimal motion that is more complex than an orbit. This, particularly, applies to nonstationary fields.

When a best orbit is searched for, an energy minimal orbit is computed for each entry point by setting to infinity the arc labels between source and entry set as well as between the copy of the entry set and the sink except for one entry point. Then, the minimum energy orbit through this entry point is computed and the procedure is repeated for all other entry points. The best of all these orbits is eventually chosen. Noteworthy, the arc labels in the grid sections of the double waypoint graph are not affected. Computations are summarized in the next algorithm and a computation sample with Scilab [11] is illustrated in figure 12.

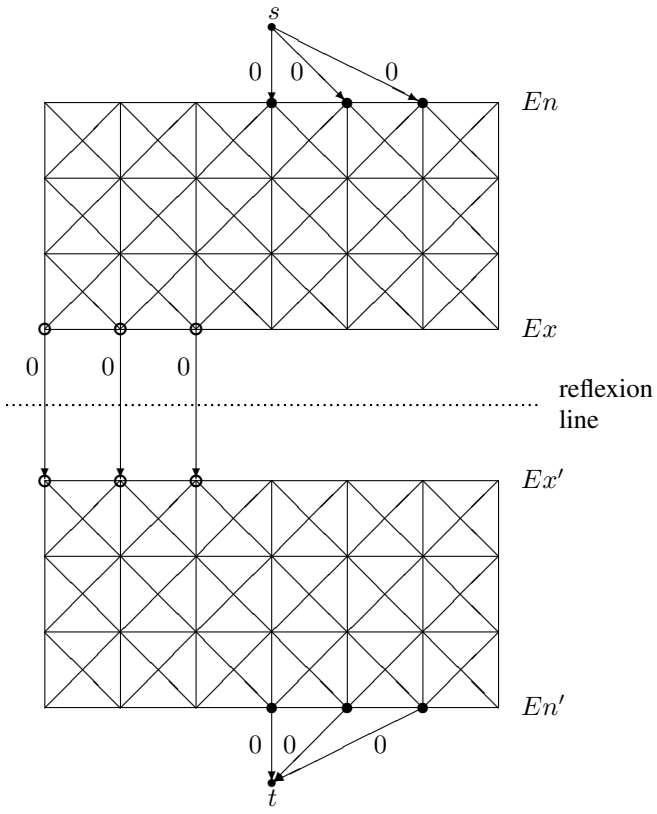

Figure 11: Double waypoint graph with arc directions and labels omitted in the RoO. The RoO is here assumed to be rectangular with a discretization of 28 nodes. Also, every node of the boundary either is an entry point or an exit point (unless the field is zero in a boundary point or heads along a straight segment of the boundary), but the entry set and the exit set are only shown partially.

MAP-orbit-double (Minimum Average Power-orbit approximation in double waypoint graph)

1. Input RoO and finite set En of entry points. (Initialization). For each $a \in En$ compute the exit point $ex(a)$, summarize all these as exit set $Ex = \{ex(a) | a \in En\}$ and create the double waypoint graph.
2. (Computations).
   (a) For all $a \in En$ do:
      i. $c(s,a) = 0$.
      ii. $c(s,b) = \infty$ for all $b \in En - \{a\}$.
      iii. $c(a',t) = 0$.
      iv. $c(b',t) = \infty$ for all $b' \in En' - \{a'\}$.
      v. Computation of a minimum energy path $P(s,t)$ from s to t by **Dt**.
      vi. Deletion of s and t from $P(s,t)$ to result in orbit $O(a)$ through a.

   (b) $O_{MAP} = argmin_{O(a),\, a \in En} \frac{energy(O(a))}{time(O(a))}$.
3. (Termination). Output O_{MAP}.

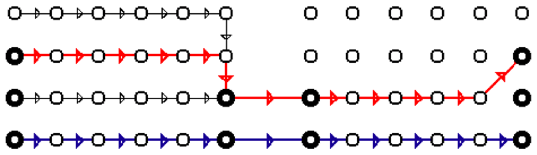

Figure 12: Double waypoint graph with original waypoint graph (left) with three entry nodes and two exit nodes. Only arcs with zero energy cost are shown in that part. The reflected graph section (right) has different energy labels: horizontal and vertical transitions incur 10 units and diagonal transitions incur 15 units. Transition times are identically one for all horizontal and vertical transitions and 1.5 for all diagonal transitions. The two prolonged arcs between the exit points and their reflections have no physical meaning and incur zero time and energy cost. The upper path requires 0 + 55 = 55 energy units while using 6+5.5 = 11.5 time units requiring average power 55/11.5 = 4.783 which is optimal. Another path is the lower path which requires 0 + 50 = 50 energy units and 5 + 5 = 10 time units thus requiring average power 50/10 = 5.

The average energy spent by orbiting is compared to the power spent by remaining still at the most favorable point inside RoO and the better of the two options is selected. A most favorable point is one with minimum field strength: $P_{min} = argmin_{P \in RoO} \|f(P)\|$. To remain still there requires to continuously execute the control vector $u(t) = -f(P_{min})$ which incurs the power $cost(u(t)) = power(-f(P_{min}))$, see section 4.1.

The objective of minimizing energy for uninterrupted observation can be overlayed with a variety of other objectives. These include the requirement for observation from sufficiently many positions and angles, intended unobservability or positional irregularity of the observing platform itself and interleaving observations of different objects that cannot be observed from any one position.

7 Exploration

Exploration of a region is similar to observation with the difference being that a set of locations must be traversed so that each point of the exploration region is observable at least once. If the whole exploration region were observable from a single location, say by a circular scan, then exploration were trivial. It is thus assumed that no single observation region – be it circular, rectangular or else – covers the exploration region so that the platform must move, see figure 13. It is further assumed that the cost of making observations is negligible compared to the cost of moving the platform with the exception of drift which incurs zero cost.

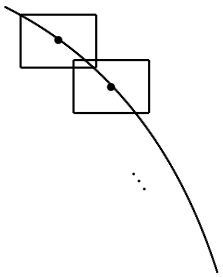

Figure 13: Trajectory with observation regions from two locations from which the observations are made.

The aim is to find minimum time and minimum energy trajectories which to cover an exploration region by finite many observation regions. A minimum energy exploration trajectory need not trace "adjacent" drift trajectories as indicated by figure 14. For computing exploration paths, the exploration region is endowed with an exploration graph. This is a directed graph whose nodes indicate all locations that must be visited for making observations. The positions in figure 13 may serve as nodes of an exploration graph. Its arcs describe possible motions between nodes and each arc receives a label with the same meaning as in waypoint graphs. Physical resolutions of exploration graphs and waypoint graphs may differ.

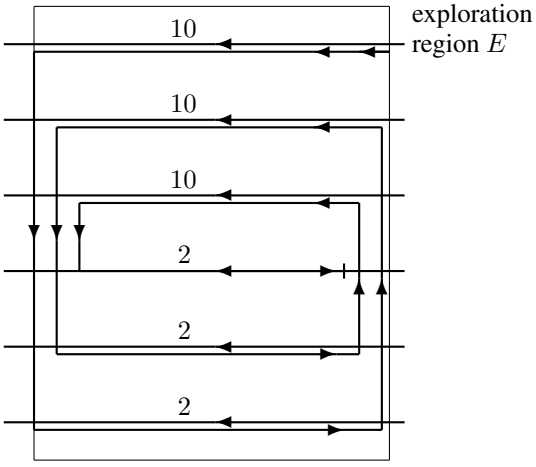

Figure 14: Drift trajectory with high velocity field regions being traveled downstream while low velocity field regions are traveled upstream for exploring region E. Observation regions are not shown.

An exploration path may repeatedly visit nodes of the exploration graph before all other nodes have been visited. Thus, exploration paths relax Hamilton paths which visit each node of a graph exactly once; a Hamilton path is a Hamilton cycle in which "the last" arc is missing. A standard transformation [8, p. 23] allows to reduce the computation of an exploration path to a Hamilton path. The exploration graph is therefore endowed with all missing arcs. The resulting complete directed graph receives arc labels that denote the shortest path from head to tail for each node in terms of the original arc labels. For stationary fields the new labels are formally denoted as

$$C(i,j) \quad = \quad \text{length of shortest path from } i \text{ to } j \text{ according to the arc labels } c(\cdot,\cdot).$$

The new labels can be computed by several applications of algorithm **Dt** or by a single application of the Floyd-Warshall (triple) algorithm [6]. A Hamilton path is then computed for the "completed" exploration graph. When such a path uses an arc which does not exist in the original exploration graph, the platform travels along the corresponding shortest path as indicated in figure 15. The starting node of an exploration path, typically, is that node of the exploration graph which is reachable at minimum cost from the present platform location – either inside or outside the exploration region. While the starting point P_S is predetermined for computations of exploration paths, the end point P_E is determined by the optimization.

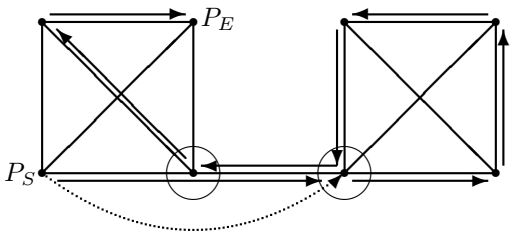

Figure 15: Original exploration graph (shown as undirected graph for notational ease) with one additional arc (dashed arrow) shown for the completed exploration graph and used by an optimal Hamilton path from a starting node P_S to an end node P_E. The resulting exploration trajectory in the original exploration graph is indicated by solid arrows. Two of the nodes (circled) are visited twice.

The shortest Hamilton path problem is known to allow a great variety of initialization and improvement heuristics. An initial heuristic which works under all circumstances is the nearest neighbor heuristic which is now adopted to yield exploration paths. The approach, simply, is to choose the nearest unvisited node for the next visit until all nodes have been visited. This greedy procedure works on the completed exploration graph is that node repetitions may occur.

NN (Nearest Neighbor heuristic)

1. Input exploration graph with node set V_E, original arc labels $c(\cdot,\cdot)$ and starting point P_S.
   (Initialization). Set $L = V_E - \{P_S\}$, $path = (P_S)$ and

$P_{current} = P_S$, computation of shortest path labels $C(\cdot, \cdot)$.

2. (Iteration). While $L \neq \emptyset$ do:
   (a) Computation of $j_0 = argmin_{j \in V_E - L} C(P_{current}, j)$.
   (b) $L = L - \{j_0\}$.
   (c) $path = path \circ j_0$.
   (d) $P_{current} = j_0$.
3. (Termination). Output exploration trajectory $path = (P_S, \ldots, P_E)$.

The exploration path is incrementally built by node concatenation in step 2(c). An alternative to greedy procedures is to solve an assignment problem which maps each node to a successor node such that each successor appears only once overall and the sum of transition costs from all nodes to their successors becomes minimal. Assignment problems can be solved efficiently. Subcycles which potentially occur in an optimal assignment need to be connected to form a Hamilton cycle. Eventually, this Hamilton cycle is broken up "right before" the starting point is revisited.

8 Motion with Controls in 3D

The foregoing approaches generalize to three dimensions in an evident manner with controls being 3D vectors. Static waypoint graphs typically are regular 3D grids with each interior node having at least six neighbors, depending on the platform characteristics; see figure 16. Regularity of the 3D grid may be weakened by the horizontal grid spacing being different from the vertical spacing. Also, depending on the characteristics of the platform, pure vertical motion may be feasible (hot air balloons) or impossible (planes). The last case is expressed by the time and energy labels for those arcs being artificially high or infinity.

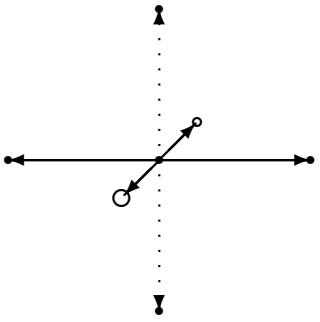

Figure 16: Grid point with four neighbors at the same altitude and two at different altitudes. If vertical transitions are impossible, these arcs are assigned high transition costs and slanted arcs are added.

The 3D case allows to pose original motion problems like finding a maximum survival trajectory. For an aerial platform with fixed energy budget this means that the platform should stay airborne with for a maximum time or travel a maximum distance. Both versions of the problem differ when energy for lift can be traded for propulsion. Survival problems make sense only for platforms that are not lighter than the atmosphere.

A survival trajectory will seek updraft areas and avoid downdraft areas, see figure 17. These may be part of circular atmospheric patterns like Hadley cells. Vertical and horizontal motion of the atmosphere are coupled inside one Hadley cell. At high altitude, the atmosphere horizontally moves from an updraft area towards a downdraft area while at low altitude motion heads in the opposite direction. It can hence be reasonable to even come close to downdraft areas. Boundaries of updraft and downdraft areas are not crisp.

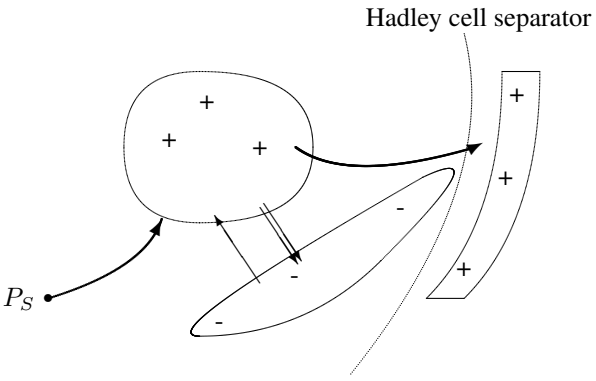

Figure 17: A survival trajectory connects the updraft areas while it avoids the downdraft area. Horizontal field motion is indicated for high altitudes (double arrow) and low altitudes (single arrow).

A trajectory that intends to visit all updraft areas with minimum energy – be it a survival trajectory or not – requires to determine the order of the visits and the entry and exit points. It is reasonable to allow revisits of updraft areas before all other updraft areas have been visited. This is in analogy to exploration paths, so that survival trajectories need not form Hamilton paths. A survival trajectory can be approximated by the nearest insertion method.

NIns (Nearest Insertion heuristic)

1. Input collection of updraft areas $L = (U_1, \ldots, U_m)$, starting position P_S.
   (Initialization). Set $P_{in} = P_S$, $\mathcal{P} = \emptyset$.
2. (Iteration). While $L \neq \emptyset$ do:
   (a) Computation of energy minimum path $\mathcal{Q}$ by algorithm **Dt** from P_{in} to an intermediate destination point = first point P_l reached in an updraft area $U_i \in L$.
   (b) $L = L - \{U_i\}$.
   (c) $\mathcal{P} = \mathcal{P} \circ \mathcal{Q}$.
   (d) $P_{in} = P_l$.
3. (Termination). Output survival trajectory $\mathcal{P}$.

Though all intermediate destination points computed in step 2(a) lie in different updraft areas, the incremental paths $\mathcal{Q}$ are allowed to revisit updraft areas as well as waypoints inside and outside updraft areas.

When an updraft area is not exited at maximum altitude, an additional ascent adds a safety margin. When vertical motion is expensive, it is not obvious how far to climb so that the next updraft area is entered higher than originally planned. Modifying a path to enter the next updraft area at a higher altitude can be facilitated by the following incremental lift procedure. To simplify the notation, it is assumed that sufficient (feasible) altitude can be gained in the current updraft area.

IncLift (Incremental Lift heuristic)

1. Input current entry point P_1 and exit point $P_2 = (x_2, y_2, z_2)^T$ of updraft area U_i and entry point $P_3 = (x_3, y_3, z_3)^T$ of the next updraft area U_j.
   (Initialization). Set $z'_3 = z_3$, $k = 1$.
2. (Iteration). While $z'_3 \leq z_3$ do:
   (a) Computation of energy minimal path $\mathcal{Q}_1$ by algorithm **Dt** from P_1 to the first point $P'_2 \in U'_i = \{P = (x, y, z)^T \in U_i |$ with $z = z_2 + k \cdot d\}$.
   (b) Computation of energy minimal path $\mathcal{Q}_2$ by algorithm **Dt** from P'_2 to the entry point $P'_3 = (x'_3, y'_3, z'_3)^T$ of U_j.
   (c) $k = k + 1$.
3. (Termination). Output new partial survival trajectory $\mathcal{Q} = \mathcal{Q}_1 \circ \mathcal{Q}_2$.

The remainder of the survival trajectory must be adapted to the next updraft area U_j and incremental liftings can be applied to other transitions as well. A greedy behavior suggests to continue climbing in each updraft area until the average cumulative climbing energy in the present updraft area increases.

9 Conclusion

It has been shown how to model and perform trajectory computations in ambient fields beyond mere point to point motion. Trajectories are specified in space and time resolutions that typically range above the control level, though the computations can be embedded into standard control frameworks. Advantageous field effects are exploited where possible and disadvantageous effects are mitigated where needed. All approaches lend to identical or similar spatial discretizations as well as to related graph algorithms.

Future work is to include uncertainty of the ambient field such as given for simple point to point motion in the framework of Markov decision processes [18]. Of particular interest are deviations from the assumed field which can be sensed by the mobile platform itself. The case of no a-priori information about the actual field is an even more challenging task.

References

[1] Allsopp, T., Mason, A., Philpott, A., "Optimizing yacht routes under uncertainty", Proceedings of the National Conference of the Operations Research Society of Japan, Tokyo, 2000, p. 176-183.

[2] Bertsekas, D., "Dynamic programming", Prentice Hall, Englewod Cliffs, 1987.

[3] Das, T., Mukherjee, R., Cameron, J., "Optimal trajectory planning for hot-air balloons in linear wind fields", Journal of Guidance, Control and Dynamics 26, 2003, p. 416-424.

[4] Harries, S., Hinnenthal, J., "A systematic study on posing and solving the problem of pareto optimal ship routing", 3rd International Conference on Computer Applications and InformationTechnology in the Maritime Industries COMPIT 2004, Sigu'nza, Spain, May 2004.

[5] Kämpke, T., Elfes, A., "Optimal aerobot trajectory planning for wind-based opportunistic flight control", Proceedings of the International Conference of Intelligent Robots and Systems IROS, Las Vegas, 2003, # 862.

[6] Lawler, E.S., "Combinatorial optimization: networks and matroids", 2nd ed., Dover, Mineola, 2001.

[7] Lee, H., et al., "Optimum ship routing and it's implementation on the web", Springer Lecture Notes in Computer Science 2402, Berlin, 2002, p. 125-136.

[8] Lawler, E.L., Lenstra, J.K., Rinnoy Kan, A.H.G., Shmoys, D.B., "The traveling salesman problem", Wiley, Chichester, 1985.

[9] Reif, J., Sun, Z., ”Movement planning in the presence of flows”, Springer Lecture Notes in Computer Science 2125, Berlin, 2000, p. 450ff.

[10] Rowe, N.C., ”Obtaining optimal mobile-robot paths with non-smoth anisotropic cost functions using qualitative state reasoning”, International Journal of Robotics Research 16, 1997, p. 375-399.

[11] Scilab, The free platform for numerical computation, www.scilab.org.

[12] Searoutes, ”The searoute system for vessel optimal routing planning”, www.searoutes.sg.

[13] Shadden, S.C., Lekien, F., Marsden, J.E., ”Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows”, Physica D 212, 2005, p. 271-304.

[14] Skysails, ”Skysails technology information”, www.skysails.info.

[15] Statheros, T., Howells, G., McDonald-Maier, K., ”Autonomous ship collision avoidance navigation concepts, technologies and techniques”, The Journal of Navigation 61, 2008, p. 129-142.

[16] Tarapate, Z., ”Selected multicriteria shortest path problems: an analysis of complexity, models and adaptation of standard algorithms”, Int. Journal of Mathematical Computing Science 17, 2007, p. 269-287.

[17] Veldhuizen, T., ”Sailing around the world in minimal time”, preprint, University of Waterloo, Canada, 2000.

[18] Wolf, M. et al., ”Probabilistic motion planning of balloons in strong, uncertain wind fields”, Proc. International Conference on Robotics and Automation ICRA2010, Anchorage, 2010, p. 1123-1129.