Decomposition Algorithm for Global Reachability Analysis on a Time-varying Graph with an Application to Planetary Exploration

Yoshiaki Kuwata, Lars Blackmore, Michael Wolf, Nanaz Fathpour, Claire Newman, and Alberto Elfes

***Abstract*— Hot air (Montgolfiere) balloons represent a promising vehicle system for possible future exploration of planets and moons with thick atmospheres such as Venus and Titan. To go to a desired location, this vehicle can primarily use the horizontal wind that varies with altitude, with a small help of its own actuation. A main challenge is how to plan such trajectory in a highly nonlinear and time-varying wind field. This paper poses this trajectory planning as a graph search on the space-time grid and addresses its computational aspects. When capturing various time scales involved in the wind field over the duration of long exploration mission, the size of the graph becomes excessively large. We show that the adjacency matrix of the graph is block-triangular, and by exploiting this structure, we decompose the large planning problem into several smaller subproblems, whose memory requirement stays almost constant as the problem size grows. The approach is demonstrated on a global reachability analysis of a possible Titan mission scenario.**

***Index Terms*— Dijkstra's algorithm, Decomposition, Block-triangular matrix, Reachability**

I. INTRODUCTION

Recent studies have proposed the use of a hot air (Montgolfiere) balloon for possible exploration of Titan and Venus because these bodies have thick haze or cloud layers that limit the science return from an orbiter, and the atmospheres would provide enough buoyancy for balloons [1], [2], [3]. One of NASA's Outer Planet Flagship mission concepts under study is the Titan Saturn System Mission, which would be a joint NASA-ESA partnership that plans to employ a Montgolfiere balloon along with a lake lander and an orbiter [4]. This balloon would circle Titan, investigating how Titan and Saturn operate as a system and determining how far prebiotic chemistry has developed. In the study of such mission, one of the important questions that need to be addressed is: "What surface locations can the balloon reach from an initial location, and if so how long would it take?" We called this the global reachability problem, where the paths from starting locations to all possible target locations must be computed.

The balloon could be driven with its own actuation, but due to its large inertia, slow dynamics, and the mass requirements, its actuation capability is fairly limited. It would be far more efficient to take advantage of the wind field and "ride" on the wind that is much stronger than what the actuator could produce. However, path planning in a flow field that is much stronger than the vehicle's actuation limits is very little studied [5]. One of the challenges in dealing with planet's wind field is that it is driven by a complex combination of many factors such as tidal effects, seasonal effects, and natural modes of the atmosphere, and therefore is highly non-linear and time-varying. It is possible to pose the path planning problem as a graph search problem on a directed graph [6] by discretizing the space-time world [7], [8] and the vehicle actuation.

However, the size of this graph becomes excessive when considering the global reachability problem for a planetary exploration. This is because for each dimension (e.g., latitude, longitude, altitude, and time) the size of the graph increases exponentially, and also because of the resolution required to capture the various time scales of the wind field while planning over the long mission duration.

To address this issue, this paper presents a decomposition algorithm for reachability analysis of a time-varying graph. Because the balloon only moves in the positive direction in time, the adjacency matrix of the graph can be represented with an upper block-triangular matrix, and this upper block-triangular structure can be exploited to decompose a graph search problem. Instead of solving a single large problem, our algorithm solves subproblems sequentially whose size is much smaller than the original problem. The new approach therefore consumes much smaller amount of memory, which also helps speed up the overall computation when the computing resource has a limited physical memory compared to the problem size.

The rest of the paper is organized as follows. Section II presents the problem statements and how the graph is defined. Section III presents the decomposition approach that can handle graphs of much larger sizes than the naive approach. Finally, Section IV shows results from the numerical simulation.

II. PROBLEM SETUP

A. Problem Statement

The problem of interest in this paper is to find the minimum-time trajectory from an initial location $\boldsymbol{r}_0$ to each of the n_g goal locations $\boldsymbol{r}_i$, $i = 1, \ldots, n_g$, given a time-varying wind model $\boldsymbol{w}(\boldsymbol{r}, t)$. In the Titan mission, radio-thermal generators could constantly provide a power, so that the energy consumption is not considered as part of the cost.

Y. Kuwata, L. Blackmore, M. Wolf, N. Fathpour, and A. Elfes are with Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive, Pasadena, CA, USA. Yoshiaki.Kuwata@jpl.nasa.gov

C. Newman is with Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA.

Because the time constant of the vehicle dynamics (order of seconds) is significantly smaller than the time constant of the path planning problem (order of days), the vehicle is assumed to move with the wind when no actuation is applied [9]. The vehicle is also assumed to have vertical and horizontal actuators that can generate additional velocity $\boldsymbol{u}(t)$ with respect to the air, so that the model used in the trajectory planning is written as

$$\frac{d\boldsymbol{r}(t)}{dt} = \boldsymbol{w}(\boldsymbol{r},\, t) + \boldsymbol{u}(t) \tag{1}$$

where $\boldsymbol{r}(t)$ is a 3D position of the vehicle. The vertical actuation is subject to the maximum sink and rise limits, denoted by v_{rise} and v_{sink}, and the 2-norm of the horizontal actuation is also subject to the maximum limit $u_{h\max}$

$$\boldsymbol{u}(t) = \begin{bmatrix} u_x(t) & u_y(t) & u_z(t) \end{bmatrix}^T$$
$$-v_{\text{sink}} \le u_z(t) \le v_{\text{rise}} \tag{2}$$
$$\sqrt{u_x(t)^2 + u_y(t)^2} \le u_{h\max} \tag{3}$$

where the subscripts x, y, z represent easterly, northerly, and vertically upwards direction respectively. Then, the problem is written as

$$\forall i = 1, \ldots, n_g: \qquad \min_{\boldsymbol{u}(\cdot)} t_{g_i} \qquad \text{s.t.} \qquad (1),\ (2),\ (3)$$
$$\boldsymbol{r}(t_0) = \boldsymbol{r}_0$$
$$\boldsymbol{r}(t_{g_i}) = \boldsymbol{r}_i$$

where t_{g_i} is the time of arrival at the i^{th} goal $\boldsymbol{r}_i$. For simplicity, the initial time t_0 is assumed to be 0.

In the Titan wind field, the vertical wind is about 100 times weaker than the horizontal wind [10] and is also much weaker than the vertical actuation capability of the balloon (typically by heating or venting the air). Therefore, we assume that the vertical motion of the Montgolfiere is fully controllable subject to the maximum rise and sink rates.

B. Graph Construction

The wind field $\boldsymbol{w}(\boldsymbol{r},\, t)$ is obtained as an output of the large-scale numerical simulation [10] and is highly nonlinear. In order to handle this nonlinearity, we convert the trajectory planning problem into a graph search problem on a discretized environment. Because the wind at a given location differs depending on when the balloon reaches there, a temporal as well as spatial discretization is performed.

1) Node: Let s_i denote an i^{th} node in the graph. Each node s_i is a function of the position $(x_i,\, y_i,\, z_i)$ and the time t_i. A uniform grid is used to represent the world, and let n_x, n_y, n_z, and n_t respectively denote the number of cells in x, y, z, and the time axes. Furthermore, let Δx, Δy, Δz, and Δt respectively denote the discretized step size along these axes. For the global reachability problem, $\Delta x = 2\pi/n_x$ and $\Delta y = \pi/n_y$ in radians. Note that the Euclidean distance of Δx is different depending on the latitude in this case. Define $n_r \triangleq n_x n_y n_z$ as the number of cell positions in the environment, and $N \triangleq n_r n_t$ as the total number of cells. The subscript i for the graph node s_i ranges from $i = 1, \ldots, N$.

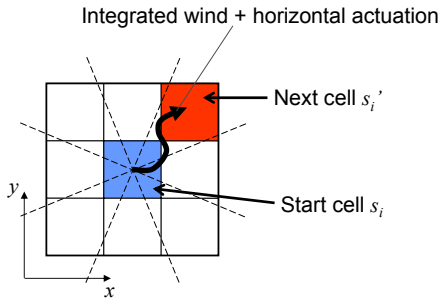


Fig. 1. Integration in the horizontal plane to find the edge connection, given a horizontal actuation.

2) Edge: If the wind were stationary, one could find which adjacent cell the balloon will hit from a given cell by looking only at the wind direction at the cell center. The edge cost in this paper is a time of travel from one node to the next and could be similarly computed by simply calculating the distance to the next cell and dividing it by the wind velocity. However, this does not apply to the time-varying wind case because the wind direction/magnitude could change as the balloon travels.

In our approach, the first step is to consider the horizontal plane and numerically integrate the velocity vector (1) over time with a time step Δt. To simplify the planning problem, the actuation $\boldsymbol{u}(t)$ is assumed to be constant over this duration. From each cell, we consider n_h horizontal actuation vectors that are different in magnitude and/or direction. For each actuation vector, the integration starts from the center of each cell s_i and goes until the integrated position reaches one of the neighboring cells s_i', as shown in Figure 1. The integration gives both the neighboring cell s_i' that the vehicle will go next and the time of travel from s_i to s_i'. Let $c(s_i)$ denote this time of travel, which is also used as a cost.

To account for a continually weak wind field in which the balloon cannot reach a neighboring cell, the integration is terminated at time $n_{t_{\max}}\,\Delta t$, so that $c(s_i) < n_{t_{\max}}\,\Delta t$, $\forall i$. In such a case, the vehicle is assumed to stay at the same location for the first time step, reaching a node with the same position but a different time $t_i + \Delta t$.

The next step is to consider the vertical motion. By applying vertical actuation, cells above or below s_i' may also be reached. As stated before, the vertical range of the Montgolfiere depends only on the maximum rise and sink rates and the time of travel. Hence, the maximum altitude increase and decrease possible in traveling from s_i to an adjacent cell are

$$z_{\text{rise}}(s_i) = v_{\text{rise}} c(s_i), \qquad z_{\text{sink}}(s_i) = v_{\text{sink}} c(s_i)$$

Let $\mathcal{R}(s_i)$ denote a set of reachable cells from s_i. Then, using the information of node s_i', $\mathcal{R}(s_i)$ is characterized as

$$s_j \in \mathcal{R}(s_i) \quad \Leftrightarrow \quad \begin{cases} x_j = x_i' \\ y_j = y_i' \\ z_i - z_{\text{sink}}(s_i) \le z_j \le z_i + z_{\text{rise}}(s_i) \\ t_j = t_i + c(s_i) \end{cases}$$

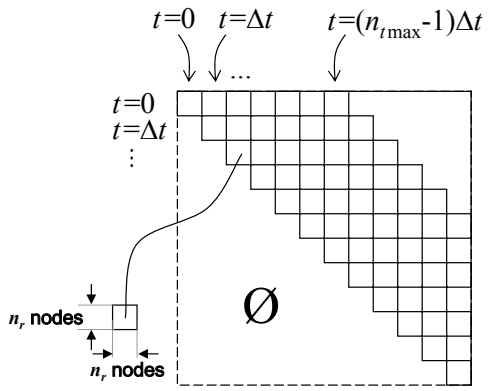


Fig. 2. Adjacency matrix.

From each node s_i, the connections are made to all nodes in $\mathcal{R}(s_i)$ with the associated cost $c(s_i)$. Note that the discussion above is for one horizontal actuation vector, so there will be multiple (up to n_h) $\mathcal{R}(s_i)$ for each s_i when forming the weighted adjacency matrix used in the graph search. Let A denote this weighted adjacency matrix of the graph.

C. Graph Search

Let n_s denote the number of starting nodes. The problem statement for the graph search is to find the shortest paths from n_s starting nodes s_i to all n_r 3D locations. Once the graph is constructed, it is straightforward to apply Dijkstra's algorithm to find the shortest paths. A matrix of size $n_s \times n_r$ can represent the minimum time of arrival at each 3D location from each starting node. Let this matrix be denoted by C^*. Its (p, q) element C_{pq} stores the time of travel from the p^{th} starting location to the q^{th} 3D location, and is set to be ∞ if there exists no such trajectory.

III. DECOMPOSITION APPROACH

The primary components of the time-varying wind field of Titan include (a) seasonal changes in the global atmospheric circulation driven by the changing solar forcing as Saturn orbits the Sun (period of 1 Titan year, ∼30 Earth years), (b) tidal effects driven by Titan's eccentric orbit around Saturn (period of 1 Titan day, ∼16 Earth days), and (c) small- and large-scale waves occurring naturally in Titan's atmosphere on a range of time-scales. A potential future mission to explore Titan would have a mission duration of 6–12 Earth months, and capturing the time-varying wind field of various time scales would require a large n_t to be used in the graph.

Figure 2 shows the adjacency matrix of the graph, which is of size $N \times N$. Because the balloon only moves in the positive direction in time, it can be represented with an upper block-triangular matrix, by ordering the nodes in terms of their time t_i. Each block contains a snapshot of the 3D world, whose size is $n_r \times n_r$. With a large n_t, the memory requirement for graph construction and Dijkstra's algorithm becomes significant. However, this upper block-triangular structure of the adjacency matrix can be exploited to decompose the problem into several smaller subproblems that uses much smaller memory. This paper considers minimum-time trajectories, but the edge cost can include other terms such as control penalty, without changes in the algorithm.

A. Algorithm

Figure 3 schematically shows the main idea of the decomposition algorithm. We assume the balloon starts at $t = 0$, so that all the starting nodes are in the (1, 1) block. This approach splits the weighted adjacency matrix A into several submatrices M_k, $k = 1, \ldots, k_{\max}$, and repeatedly applies Dijkstra's algorithm to each submatrix. The result of each subproblem can be represented by a matrix (shown with a red rectangle) whose size is much smaller than the submatrix used in the subproblem (marked with a blue rectangle). The next subproblem is formed by appending to its submatrix the small matrix obtained in the previous subproblem. This process is repeated until all the submatrices are processed or all the shortest paths from the starting nodes to the 3D locations are found.

More formally, let b_r and b_c be the number of row blocks and column blocks in each submatrix M_k respectively. Because each block has a size $n_r \times n_r$, the submatrix M_k consists of rows from $(k-1)b_r n_r + 1$ to $kb_r n_r$ and columns from $(k-1)b_r n_r + 1$ to $(k-1)b_r n_r + b_c n_r$ of A, i.e.,

$$A = \begin{bmatrix} \longleftarrow M_1 \longrightarrow & & \\ & \longleftarrow M_2 \longrightarrow & \\ & & \ddots \\ O & & \longleftarrow M_{k\max} \longrightarrow \end{bmatrix}.$$

Note that b_c is written as

$$b_c = n_{t_{\max}} + b_r - 1.$$

Algorithm 1 shows the pseudo-code of the decomposition algorithm. The first step (line 1) is to run Dijkstra's algorithm with M_1 as a weighted adjacency matrix, which we call Subproblem 1. The resultant shortest path costs (from n_s starting locations to all the cells considered in Subproblem 1) can be represented by a matrix D_1 whose size is of n_s-by-$b_c n_r$. This D_1 can be partitioned into two matrices

$$D_1 = \begin{bmatrix} E_1 \mid F_1 \end{bmatrix} \tag{4}$$

where E_1 is a matrix of size n_s-by-$b_r n_r$, and F_1 is a matrix of size n_s-by-$(b_c - b_r)n_r$. The matrix E_1 corresponds to cells with time steps between 0 and $(b_r - 1)\Delta t$, and the optimal paths to these cells have been obtained. The matrix F_1 corresponds to cells with time stamps after $b_r \Delta t$, and not all the options to reach them have been explored yet.

Using E_1, the cost matrix C^* is initialized. We first reshape E_1 into a three dimensional array of size n_s-by-n_r-by-b_r, and then take the minimum along the third dimension, obtaining a n_s-by-n_r matrix denote by C_1. For the nodes that cannot be reached from the starting cells, Dijkstra's algorithm is assumed to output ∞ as the cost. The cost matrix C^* is then set (line 3) to be

$$C^* = C_1.$$

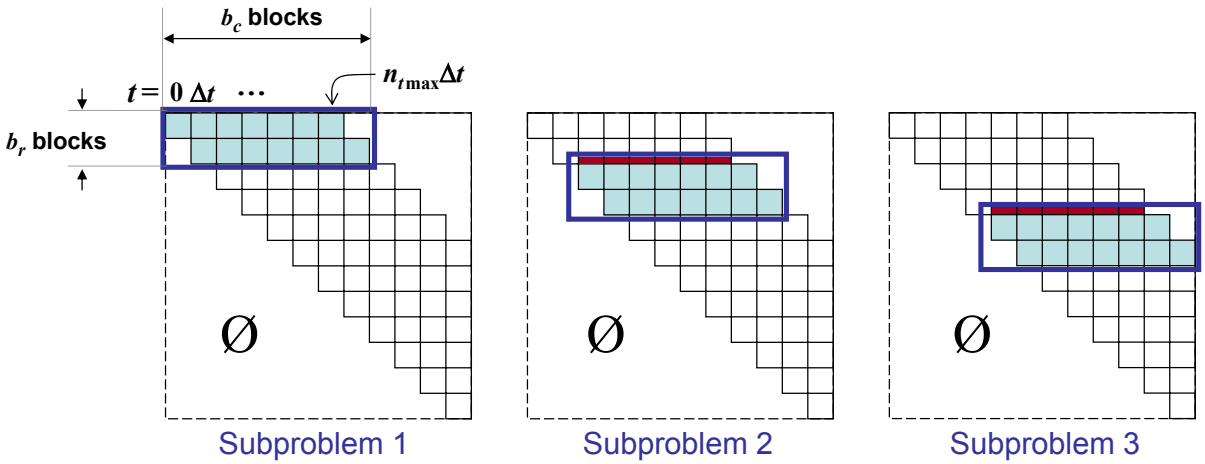


Fig. 3. Adjacency matrix used in the sequence of subproblems. The blue rectangle shows the part of the adjacency matrix that are used in each subproblem. The red flat rectangle represents the results obtained from the subproblems solved so far.

The next subproblem uses M_2 and considers times from $b_c\Delta t$ to $(2b_c - 1)\Delta t$, but it also includes n_s starting nodes. Therefore, in order to form the weighted adjacency matrix of this subproblem, M_2 must be augmented using the result of the previous subproblem (line 8). Let $O_{p,q}$ denote a zero matrix of size p-by-q. Then, the weighted adjacency matrix used in Subproblem 2 is written as

$$\left[\begin{array}{c|c|c} O_{n_s,n_s} & F_1 & O_{n_s,b_r n_r} \\ \hline O_{b_r n_r,n_s} & \multicolumn{2}{c}{M_2} \end{array}\right].$$

Note that an infinity matrix of width n_s is appended from the left. This is because the starting nodes at time $t = 0$ have only outgoing edges. The matrix F_1, obtained in the previous subproblem, stores the cost of moving from the starting cells to all cells at time between $(b_r + 1)\Delta t$ and $b_c\Delta t$.

After solving subproblem 2, the resultant shortest path costs are represented by a matrix D_2. The first n_s columns of D_2 correspond to the starting cells, so that D_2 can be partitioned into $D_2 = \left[G_2 \mid E_2 \mid F_2\right]$, where G_2 is a square matrix of size n_s, and E_2 and F_2 respectively have the same sizes as E_1 and F_1 defined in (4).

From E_2, the minimum path costs from starting locations to each 3D location are computed, and C^* is updated (line 9–10). As shown in the Algorithm 1, the process is repeated until all the rows of adjacency matrix A are used, or all paths to the 3D locations are found and C^* has no non-∞ elements.

Algorithm 1 Decomposed Dijkstra for global reachability

1: Solve for shortest paths with weighted adjacency matrix M_1. Obtain a matrix $[E_1 \mid F_1]$ with the minimum cost.
2: From E_1, compute the minimum path cost C_1 from n_s starting locations to each 3D location.
3: Initialize the cost matrix $C^* := C_1$.
4: **for** k = 2 to $k_{\max}$ **do**
5: **if** all elements of C^* are not ∞ **then**
6: break
7: **end if**
8: Solve for shortest paths with weighted adjacency matrix $\left[\begin{array}{c|c|c} O_{n_s,n_s} & F_{k-1} & O_{n_s,b_r n_r} \\ \hline O_{b_r n_r,n_s} & \multicolumn{2}{c}{M_k} \end{array}\right]$. Obtain a matrix $\left[\begin{array}{c|c|c} G_k & E_k & F_k \end{array}\right]$ with the minimum cost.
9: From E_k, compute the minimum path cost from starting locations to each 3D location C_k.
10: Update the cost matrix $C^* := \min(C^*, C_k)$. Note that this min operation is element-wise.
11: **end for**

IV. SIMULATION RESULTS

A. Setup

The results in this section were generated using the wind field model of Titan developed by [10]. The grid resolution of the wind model was 5.625 degrees in longitude, 5 degrees in latitude, 500 meters in altitude, and 0.125 Titan day in time. The grid resolution for planning was 10 degrees in longitude and latitude, 500 meters in altitude, and 0.02 Titan day in time. This makes $n_x = 36$, $n_y = 18$, $n_z = 20$, and hence $n_r = 12960$ in Figure 2. Other parameters used are:

$$\begin{aligned} n_t &= 520, & n_{t_{\max}} &= 100 \\ n_s &= 24, & n_g &= 12960 \\ b_r &= 10, & b_c &= 109 \\ v_{\text{sink}} &= 0.6 \text{ [m/s]}, & v_{\text{rise}} &= 0.3 \text{ [m/s]} \\ u_{h\max} &= 0.25 \text{ [m/s]} \end{aligned}$$

Because we consider a global reachability problem, the number of goals n_g in the graph search is the same as n_r. If this were solved as a single graph search problem, the graph would contain about 7 million nodes with billions of edge connections. The starting altitude is set to be 5000 m. It is also assumed that the horizontal actuation is either zero or of full magnitude in one of the 8 directions, so that $n_h = 9$.

All the algorithms are implemented in MATLAB. The simulation was run on a desktop computer with Core 2 Duo

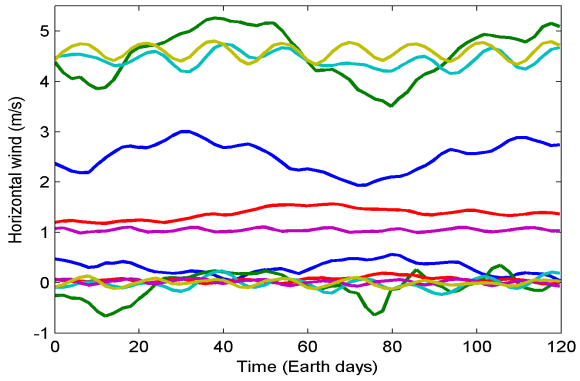


Fig. 4. Time-varying wind field

2.40 GHz and 4 GB of RAM running 64-bit Linux and 64-bit MATLAB. Dijkstra's algorithm was implemented as a MEX function using Fibonacci heaps [11].

Figure 4 shows the time plot of the horizontal wind $w_x(t)$ and $w_y(t)$ at various longitudes, latitudes, and altitudes over 120 days. Depending on the 3D location, both the magnitude and its variation are very different. Note also that the wind could be much stronger than the horizontal actuation limit.

B. Results

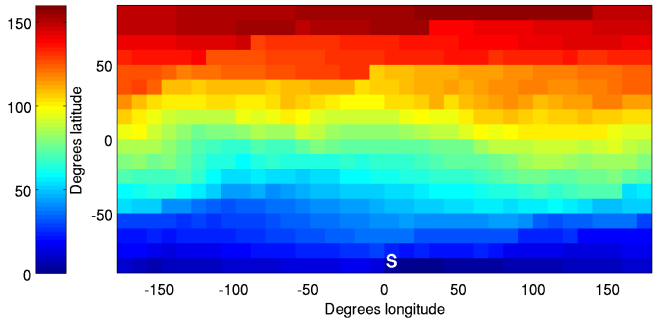
Figure 5 shows reachability plots from different starting locations to all the cells at altitude 250 m. The starting cell is marked with "S". The color of each cell shows the minimum time it takes to move from the start to each cell. Note that depending on where the balloon starts, the reachability map varies significantly.

Figure 6 shows the percentage of the areas of Titan's surface that would be reachable in a given time. For example, 50% on the y axis means that 50% of the points on Titan's surface could be reached in a given time if set as a destination; it does not mean that the balloon would sweep over 50% of the Titan surface in this time. The plot has 24 different starting locations with the following combination of longitude and latitude.

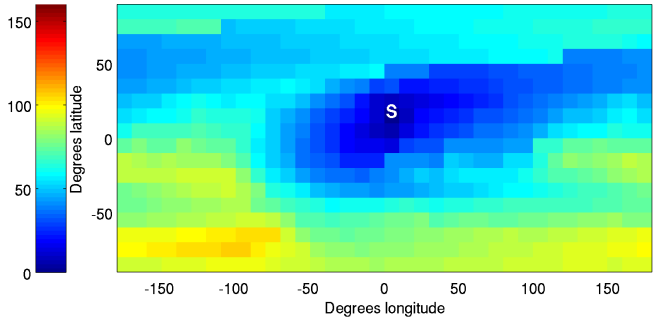
- 4 longitudes (175°W, 85°W, 5°E, 95°E)
- 6 latitudes (85°S, 45°S, 15°S, 15°N, 45°N, 85°N)

Because the wind field changes over the latitude much more than the longitude, the lines corresponding to the same latitude are plotted with the same color. In fact, the lines with the same color have a similar trend: if starting at 15°S (shown in red), it could initially reach only limited areas, but the reachable area grows rapidly after a few months; if starting near the southern pole (85°S), the reachable area does not grow as fast as others.

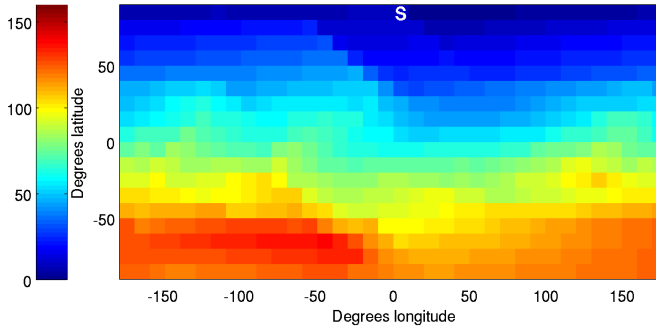
Figure 7 shows several trajectories from a start location of (5°S, 5°E) to 3 different goals G_1(5°S, 155°E), G_2(75°S, 85°W), and G_3(85°N, 85°E). The size of the circles represents the elapsed time from start, and the color of the circles represents the altitude of the trajectory. The background color represents the same reachability map shown in Figure 5(b). Note that because of the nonlinear and time-varying wind



(a) From 85 degrees north, 5 degrees east



(b) From 5 degrees south, 5 degrees east



(c) From 85 degrees south, 5 degrees east

Fig. 5. Reachability plots of different starting locations. The color bar on the left shows the time to reach each cell at an altitude of 250 m (in days).

field, the minimum-time trajectories involve several vertical actuation steps and are far from straight lines.

C. Comparison with the Approach without Decomposition

This subsection shows the computational aspects of the proposed approach. The same reachability analysis was performed without the decomposition.

Figure 8 compares the memory required to store the weighted adjacency matrix. The x axis is the number of discretized time steps involved in the problem. The line with $\times$ marks shows the memory usage for the weighted adjacency matrix when solving each subproblem of the decomposition algorithm. The line with $\circ$ marks shows the cases without the decomposition. Without the decomposition, the memory usage grows unboundedly, but with the decomposition, it stays almost constant after the second subproblem.

Figure 9 compares the computation times for the two approaches. Figure 9(a) shows the time spent by Dijkstra's algorithm, and Figure 9(b) is for the total computation including graph construction and book keeping. Without

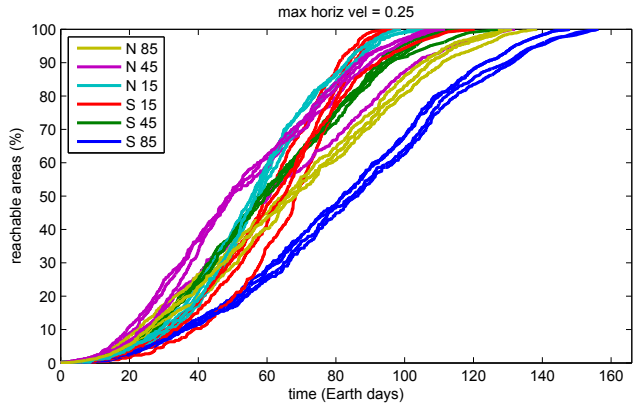


Fig. 6. The area that can be reached as a function of time.

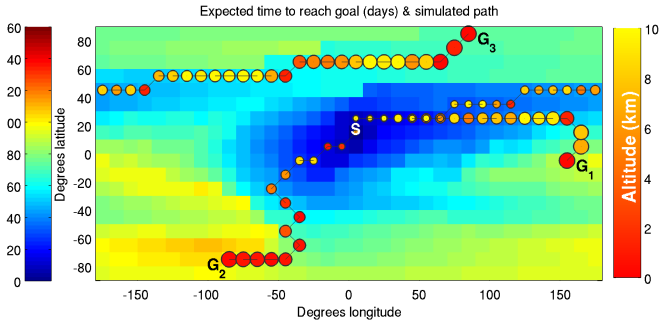


Fig. 7. The trajectories from a start to 3 different goals. The left color bar shows the time to reach each cell an altitude of 250 m. The right color bar shows the altitude of the trajectory.

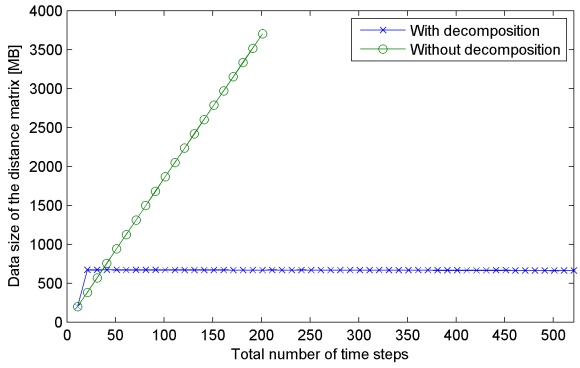
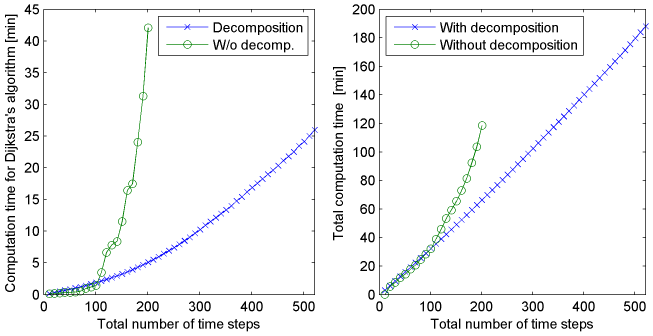


Fig. 8. Memory required to store the weighted adjacency matrix



(a) Dijkstra's algorithm (b) Total

Fig. 9. Computation time as a function of a problem size. The computation time of the decomposition approach is the summation over all the iterations.

decomposition, the computation time increases drastically beyond time step 100. If the global reachability analysis on a higher resolution grid is required, a computer with more CPUs and more memory could be used. In such a case, the algorithm presented in the paper will bring the same computational savings as shown here.

Note that the decomposition approach does not remove/add any nodes/edges in the graph and hence does not introduce suboptimality. Therefore, it produced the same solution that was obtained without decomposition.

V. CONCLUSION

This paper presented a decomposition algorithm for global reachability analysis on a space-time grid. By exploiting the upper block-triangular structure, the planning problem is decomposed into smaller subproblems. This enables us to analyze potential Titan mission scenarios of long duration, which would require excessive amount of memory when solved as a single graph search problem. Future work will investigate how to reduce the discretization errors, and evaluate how the uncertainty in the wind model affects the overall performance.

ACKNOWLEDGMENTS

The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

REFERENCES

[1] J. Jones, "Montgolfiere Balloon Missions from Mars and Titan," in *Proceedings of the 3rd International Planetary Probe Workshop*, 2005.

[2] J. O. Elliott, K. Reh, and T. Spilker, "Concept for Titan Exploration using a Radioisotopically Hated Montgolfiere," in *Proceedings of the IEEE Aerospace Conference*, 2007.

[3] D. Fairbrother, "Development of Planetary Balloons," in *Proceedings of the NASA Space Technology Conference*, 2007.

[4] NASA. Outer planet flagship mission: Titan saturn system mission (TSSM). [Online]. Available: http://opfm.jpl.nasa.gov/titansaturnsystemmissiontssm/

[5] J. Witt and M. Dunbabin, "Go with the Flow: Optimal AUV Path Planning in Coastal Environments," in *Proceedings of the Australasian Conference on Robotics and Automation*, 2008.

[6] B. Garau, A. Alvarez, and G. Oliver, "Path Planning of Autonomous Underwater Vehicles in Current Fields with Complex Spatial Variability: an A* Approach," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.

[7] T. Fraichard, "Trajectory planning in a dynamic workspace: a 'state-time space' approach," *Advanced Robotics*, vol. 13, no. 1, pp. 75–94, 1999.

[8] J. van den Berg and M. Overmars, "Roadmap-Based Motion Planning in Dynamic Environments," *IEEE Transactions on Robotics*, vol. 21, pp. 885– 897, 2005.

[9] T. Das, R. Mukherjee, and J. Cameron, "Optimal trajectory planning for hot-air balloons in linear wind fields," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 26, no. 3, pp. 416–424, 2003.

[10] M. I. Richardson, A. D. Toigo, and C. E. Newman, "PlanetWRF: A general purpose, local to global numerical model for planetary atmospheric and climate dynamics," *Journal of Geophysical Research*, vol. 112, 2007.

[11] J. Boyer, "Algorithm alley — heaps are usually implemented via binary trees, with the property that for every subtree, the root is the minimum item. here, John describes how to implement exceptionally fast 'Fibonacci' heaps," *Dr. Dobb's Journal of Software Tools*, vol. 22, no. 1, pp. 106–??, Jan. 1997.