2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Nonmyopic Planning for Long-Term Information Gathering with an Aerial Glider

Joseph L. Nguyen, Nicholas R.J. Lawrance and Salah Sukkarieh

***Abstract*—Thermal soaring can vastly increase the effectiveness of Unmanned Aerial Vehicles (UAVs) in information gathering tasks. However, knowing when to best collect information or regain energy is non-trivial. In this work, the problem is posed as a graph search problem where nodes are thermal positions, and edges are inter-thermal trajectories. Previous work has shown that this search problem is NP-hard, such that computing a long-duration plan is difficult without significant computational effort. This paper introduces two mechanisms to make this tractable. Firstly, Monte Carlo Tree Search (MCTS) is used to provide an anytime search strategy capable of generating long plans without exhaustive search. Secondly, a novel clustering approach isolates areas of interest on the information map to solve local cluster subproblems, followed by dynamic programming to optimally allocate search time to each cluster. Results demonstrate the improved performance of these approaches on longer missions.**

I. Introduction

This paper addresses the problem of informative path planning for a gliding Unmanned Aerial Vehicle (UAV). The deployment of small UAVs for information gathering has become ubiquitous; such applications include mapping, search and rescue, target-tracking and surveillance. However, conventional UAVs have limited on-board energy, which restricts their flight endurance and limits their potential mission effectiveness. This can be overcome by incorporating autonomous soaring research [1], [2], which has matured in recent years, into mission planning for long-term information gathering.

In this problem, the time duration of the information-gathering mission exceeds the UAV's flight endurance, and so the mission can only be completed with some form of online energy replenishment. Previous works [1]–[5] suggest that atmospheric *thermal soaring* can efficiently fulfil this requirement, whereby the glider periodically visits thermal energy sources to regain energy expended on informative path segments. The glider is equipped with on-board sensors used to search for a lost ground target, where knowledge of the target location is modelled probabilistically. The objective is to minimise the model's uncertainty, or equivalently maximise information gain, within a budgeted mission time.

This area of persistent information gathering and energy replenishment has recently been investigated in a number of contexts. In [6], charging robots are tasked to recharge worker robots executing predefined informative paths. A graph of possible rendezvous points is built and searched, which is akin to recharging at thermals. In [7], gliding UAVs are coordinated to visit points-of-interest in a region containing thermals. They modify depth-first search to generate rapid plans every second. In contrast, we are interested in a nonmyopic plan that extends up to the mission time.

We introduced the *informative soaring* problem in [5] and observed the key idea that any nonmyopic plan must alternate between two modes: 1) information gathering in non-thermal areas, and 2) visiting thermals to increase energy. This property reduced the set of feasible plans to connections of path segments that exist between thermal nodes. To find a near-optimal plan, we employed depth-limited (or finite-horizon) tree search. We also optimised time spent at every thermal in the plan, transforming energy cost into time cost.

This places informative soaring in a well-known class of problems that seeks maximally informative paths subject to a budget on traversal cost [8], [9]. However, the solutions from this literature cannot be used in this work because an a-priori graph of spatial node connections is required. Constructing such a graph is prohibited by the non-Markovian nature of the target-search scenario: states cannot be revisited, and all end-states can only be known by exhaustively forward propagating the sensor model (i.e. tree search). Even with depth-limited tree search [5], the subtree size grows exponentially with increasing budget, becoming infeasible to build for very large budgets. A computationally tractable solution should identify only relevant parts of the tree to build.

As [9] employs branch and bound to prune the search tree and effectively generate longer horizon plans, we adopt Monte Carlo Tree Search (MCTS) [10] to achieve computational reductions for the same high-quality plans from our previous solution [5]. MCTS is a best-first search strategy that utilises random rollouts at each node-expansion to bias tree growth towards high-yield end-states. This recent algorithm from game-AI was shown to work outstandingly well on computer *GO*, an NP-hard problem [10]. Our first contribution is extending the framework of MCTS to efficiently solve general, large-scale problems in informative soaring.

Secondly, we observe that near-optimal plans tend to concentrate search effort around clusters of information and transition between them with minimum-cost paths. This allows us to break up the problem and perform smaller, more computationally efficient tree searches on each cluster. We then determine the optimal plan using dynamic programming on every node in every cluster tree. We show that depth-limited planning under our clustering framework provides competitive solutions to MCTS; our numerical results illustrate that near-greedy depth-limited cluster tree search can even outperform MCTS for scenarios characterised by distinct information clusters.

This work was supported by the Australian Centre for Field Robotics and funded by the New South Wales State Government.

The authors are with the Australian Centre for Field Robotics (ACFR), The University of Sydney, NSW 2006, Australia {j.nguyen,n.lawrance,salah}@acfr.usyd.edu.au

II. Problem Formulation

In the informative soaring target-search scenario, an autonomous aerial glider equipped with sensors is tasked to search for a lost ground target. A probabilistic belief function of the target location is provided on which actions are planned. The glider is energy-constrained and must periodically visit thermals to replenish energy. Thermals are assumed to be known, stationary, and cover the search area densely enough for it to be explored. The goal is to find an optimal plan that maximises information gain over the entire mission time.

A. Path Planning Notation

We formulate informative soaring as a discrete problem by constructing a tree T of nodes $v \in V$. Starting with an empty root v_0, a child node v_d at depth d is constructed by appending one inter-thermal path segment to its parent v_{d-1}, so that any v represents a sequence of path segments. Each node has an associated utility $J(v)$ and cost $C(v)$. Here, $J(v)$ is the cumulative probability of target detection defined in [11], and $C(v)$ is the traversal time, since energy gained by climbing in a thermal can be transformed into time lost [5]. A feasible plan is a leaf or terminal node with cost $C(v_t)$ within some budget $B > 0$. An optimal plan is one with maximum $J(v_t) \leq 1$.

B. Glider Motion Model and Action Vector

An inter-thermal path segment consists of N glider sensing states $[\mathbf{X}_{k+1}, \mathbf{X}_{k+2}, \ldots, \mathbf{X}_{k+N}]$ that result from applying an action vector $\mathbf{v}_k = [\mathbf{u}_k, \mathbf{u}_{k+1}, \ldots, \mathbf{u}_{k+N-1}]$ starting at time instance k. The reduced-form glider state vector $\mathbf{X} = [p,\, \phi,\, \psi,\, x,\, y,\, z]^T$ comprises roll-rate p, bank angle ϕ, heading angle ψ and position $[x,\, y,\, z]^T$. A control action is a waypoint $\mathbf{u} = [x^u,\, y^u]^T$. It is converted into a heading error ψ_{err} in (1), and then ψ_{err} is managed by a PID controller to command a roll rate p_{cmd} in (2). The glider motion model and controller equations are shown in (1) to (6); further details are in [5], [12].

$$\psi_{err} = f(\psi, \mathbf{u}) \tag{1}$$

$$p_{cmd} = g(\psi_{err}) \tag{2}$$

$$\frac{dp}{dt} = -p + p_{cmd} \tag{3}$$

$$\frac{d\psi}{dt} = \frac{g\, tan\, \phi}{V} \tag{4}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \begin{bmatrix} V\, cos\, \gamma\, cos\, \psi \\ V\, cos\, \gamma\, sin\, \psi \\ -V\, sin\, \gamma \end{bmatrix} \tag{5}$$

$$sin\, \gamma = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{6a}$$

$$a = m^2 g^2 \tag{6b}$$

$$b = -mg\, cos^2 \phi\, qSk \tag{6c}$$

$$c = -(m^2 g^2 + cos^2 \phi\, q^2 S^2 k C_{D,0}) \tag{6d}$$

C. Sensor Model

The probability of detecting the target at xy-position $\xi \in \Xi$ with sensor observation z_j is captured by the seeability sensor model [13], a smooth and differentiable function:

$$p(D_j \,|\, \xi, \mathbf{X}_j) = \frac{cos\, \theta_j}{1 + (d_j / d_{norm})^2}. \tag{7}$$

D_j abbreviates $z_j = D_j$, which means the target is detected within the sensor observation, θ_j is the viewing angle from the glider down-vector to ξ, $d_j = ||\xi - \mathbf{x}_j||$ is the Euclidean distance from the glider position $\mathbf{x}_j = [x,\, y,\, z]_j^T$, and d_{norm} is a fixed quantity that defines the sensor footprint.

D. Utility Function Formulation

The cumulative probability of detection over N sensing locations is obtained by assuming independent consecutive observations, so that no-detection likelihoods can be combined. A no-detection likelihood is the complement of (7):

$$p(\overline{D}_j \,|\, \xi, \mathbf{X}_j) = 1 - p(D_j \,|\, \xi, \mathbf{X}_j). \tag{8}$$

The (scalar) joint probability of no-detection is obtained by conditioning on the prior target belief function $b_k^\xi : \Xi \mapsto \mathbb{R}$ and marginalising over all possible target positions ξ:

$$p(\overline{D}_{k+1:k+N} \,|\, \mathbf{X}_k, \mathbf{v}_k, b_k^\xi) = \int_\xi b_k^\xi \prod_{j=1}^{N} p(\overline{D}_{k+j} \,|\, \xi, \mathbf{X}_{k+j}) d\xi. \tag{9}$$

The belief b_k^ξ is uniformly grided and determines the target probability of detection at $\xi \in \Xi$ with value in $[0, 1]$. The cumulative probability of detection after this planning horizon is the complement of (9), so the utility function is:

$$J(v) \triangleq J(\mathbf{X}_k, \mathbf{v}_k, b_k^\xi) = 1 - p(\overline{D}_{k+1:k+N} \,|\, \mathbf{X}_k, \mathbf{v}_k, b_k^\xi). \tag{10}$$

E. Gradient-Based Optimisation for Inter-thermal Paths

This idea was a contribution of our previous work [5]; we summarise it here. Between a pair of thermal nodes, there exists a continuous spectrum of path segments varying in $J(.)$ and $C(.)$. However, we only consider a subset Q of up to three options: 1) the maximum utility, 2) the minimum cost, and 3) the median utility/cost path segments. $Q \subseteq R$ is selected from the path segments set R generated by deforming an initial path r_0 (discretised into a finite set of points $\mathbf{x}_{k+1:k+N-1}$) using gradient descent on the underlying belief function b_k^ξ:

$$\mathbf{v}_k^{i+1} = \mathbf{v}_k^i - \alpha \int_\xi \frac{\partial b_k^\xi}{\partial \mathbf{x}_{k+1:k+N-1}^i} d\xi, \tag{11a}$$

$$\mathbf{v}_k^i = \mathbf{x}_{k+1:k+N-1}^i. \tag{11b}$$

$\mathbf{x} = [x,\, y]^T$ is both a glider position and waypoint, i is the update iteration, and $\alpha > 0$ is the update step size. The process terminates when all gradients are zero or the final-state altitude is below a limit. In this paper, we allow thermal self-transitions (absent in [5]) with four initial paths, as illustrated in Fig. 1.

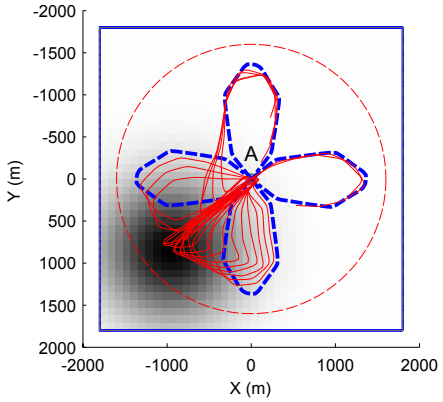


Fig. 1. Inter-thermal path segments for thermal self-transition. Darker regions represent more information. The circle marks the return-trip gliding range from thermal A. There are four possible initial paths r_0 (blue-dashed lines) from which gradient-based optimisation returns the red-solid line paths.

III. Efficient Nonmyopic Planning

As we have previously highlighted [5], the optimal plan can only be discovered by an exhaustive tree search or forward-propagation of the sensor model because of the non-Markovian nature of the target-search scenario. Consequently, computational resources grow exponentially with increasing budget B. We solve this problem by first extending an algorithm called MCTS to efficiently obtain near-optimal solutions. We then present a clustering tree search algorithm, and augment it with dynamic programming to achieve similarly high-quality plans, but with greatly reduced computational effort.

A. Monte Carlo Tree Search (MCTS)

MCTS is a best-first search algorithm that finds near-optimal solutions in large state spaces by taking random samples to bias tree search down promising nodes. In Alg. 1, `Selection` always starts at the root node v_0 and descends the tree T using `BestChild` to recursively select a child node with the best MCTS reward. It returns a node v_d at depth d that has not yet been fully expanded. `Expansion` adds a new child v_{d+1} to v_d. From this state s_{d+1} (glider state $\mathbf{X}$ and belief b^{ξ}), a rollout `Simulation` is performed to the mission time, with random actions taken. The end-state utility of this rollout Δ is back-propagated (`Backup`) to all nodes along the path from v_0 to v_{d+1}. Once a computational limit is reached, the best child v_1 of the root node is returned with state s_1.

Algorithm 1 *MctsSearch*(s_0)

1: create root node v_0 with state s_0
2: **while** within computational limit **do**
3: $v_d \leftarrow$ `Selection`(v_0)
4: $v_{d+1} \leftarrow$ `Expansion`(v_d)
5: $\Delta \leftarrow$ `Simulation`(v_{d+1})
6: `Backup`(v_{d+1}, Δ)
7: **end while**
8: **return** $[s_1, v_1] \leftarrow$ `BestChild`(v_0)

The MCTS reward function in `BestChild` is the Upper Confidence Bound for Trees (UCT) used in Bandit problems and proposed by [14]:

$$UCT = \overline{\Delta}_d + C_p\sqrt{\frac{2\ln n}{n_d}}. \tag{12}$$

$\overline{\Delta}_d$ is the average rollout utility of node v_d, n_d and n are the number of times v_d and its parent v_{d-1} have been visited, and $C_p > 0$ is a constant weight for the exploration term. The UCT policy resolves the exploration-exploitation dilemma as its growth of regret is within a constant factor of $O(\log n)$, the slowest possible rate [14], [15]. For $\overline{\Delta}_d \in [0, 1]$, which is true of the cumulative probability of detection, [14] proposes an exploration weight $C_p = 1/\sqrt{2}$, but adjusting it will vary the degree of exploration, and hence the solution quality. We explore a range of C_p values for a scenario from [5] and select a value that retains good solutions but reduces computational time. In practice, MCTS is an anytime algorithm by virtue of this adjustable C_p term and the maximum number of expansions allowed (line 2) each time Alg. 1 is called.

We modify (12) by replacing the average $\overline{\Delta}_d$ with the maximum value encountered so far on all rollouts from node v_d, its children, grand-children, etc…While the average is used in the MCTS literature for 2-player games, the maximum is appropriate for informative soaring, which results in maximising the end-state utility. Subsequently, instead of building a new tree on every call of Alg. 1, we keep all nodes of the successor v_1 to retain these strong lines of play.

B. Cluster Tree Search Algorithm

Interesting scenarios arise when the a-priori probabilistic belief function b^{ξ}_{init} is partitioned. This could be due to a series of prior uninformed local-area searches, and now the search region has been broadened. For long-term planning, this is problematic for state-of-the-art search schemes, which can be too myopic. The results from our previous work [5] demonstrate that high-quality solutions concentrate search effort at clusters of high belief uncertainty (or information). We exploit this property and develop Alg. 2 to reduce computational effort compared to single-shot planning methods.

Algorithm 2 *ClusterSearch*

1: $[b^{\xi^1}, b^{\xi^2}, \ldots, b^{\xi^n}] \leftarrow$ `GmmDistribution`(b^{ξ}_{init})
2: $P \leftarrow \emptyset$
3: $s^1_{init} \leftarrow s_{init}$
4: **for** $i = 1 : n$ **do**
5: $s_0 \leftarrow s^i_{init}$
6: $s^i_t \leftarrow$ `Cluster`$(b^{\xi^i}, b^{\xi}_{init}, B)$
7: **while** $s_0 \neq s^i_t$ **do**
8: $[s_0, v^i_t] \leftarrow$ `MctsSearch`(s_0)
9: **end while**
10: $P \leftarrow$ `TspRewire`(v^i_t, P)
11: $s^{i+1}_{init} \leftarrow s_0$
12: **end for**
13: **return** P

In Alg. 2, information clusters are firstly identified using a Gaussian mixture model [16] (line 1). They are ordered in sequence of distance from the glider start position. The plan P is initially empty, and the initial state s_{init} (glider state $\mathbf{X}_{init}$ and belief b^{ξ}_{init}) is assigned to the first cluster. `Cluster` assigns time budget to the terminal state s^i_t proportionally to the fraction of b^{ξ^i} in b^{ξ}_{init}. It also associates thermals with the cluster based on their inverse distance to the cluster centre, which may result in some thermals being assigned to multiple clusters. The terminal state s^i_t always finishes in a thermal node of cluster $i+1$, using minimum-cost link segments if necessary. Lines 7-9 repeatedly invoke `MctsSearch` until s_0 is terminal. Note that lines 7-9 may also be replaced by any other search method. Finally, the cluster plan v^i_t is added to P in line 10 with the possibility of *rewiring*.

`TspRewire` is called to remove any low-yield segments that may result from the multiple thermals-to-clusters situation. A simple example is depicted in Fig. 2. Each cluster evenly splits b^{ξ}_{init} so that the identical path segments shown are locally optimal for the given mission time. `TspRewire` recognises, via thresholding, that the straight-line repeat path of cluster two is low-yield and removes it. The surplus search time can be allocated to additional subsequent clusters. The issue of visiting the minimum number of low-yield path segments required in a feasible intermediate plan is framed as a Travelling Salesman Problem (TSP), and a randomised solver is used to get the optimal plan in this context.

C. Cluster Time Allocation via Dynamic Programming

The problem in Fig. 2 can alternatively be solved by assigning less time to cluster one and more to cluster two from the onset. Since trees are built for every cluster, where each node has a utility $J(v)$ and cost $C(v)$, we can directly utilise them to find the optimal combination of cluster plans. We assume cluster trees are independent and apply dynamic programming (DP) instead of brute force, which works well even with the multiple thermals-to-clusters situation.

The downside of this optimised cluster time allocation algorithm is that all clusters must be computed before a plan is available. The cluster tree search method in Sec. III-B (Alg. 2) is sequential, so it provides plan snippets that would be more readily available for real-time planning. Therefore, for the

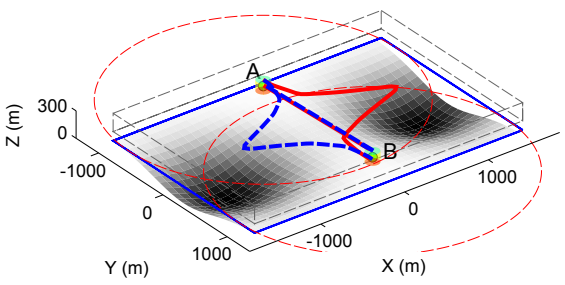


Fig. 2. Two clusters with even splits on belief b^{ξ}_{init} result in identical paths between thermals A and B for each cluster. Thermal self-transitions were not allowed here. Blue-dashed lines for cluster one, red-solid lines for cluster two.

Algorithm 3 *ClusterSearchDp*

1: $[b^{\xi^1}, b^{\xi^2}, \ldots, b^{\xi^n}] \leftarrow$ `GmmDistribution`(b^{ξ}_{init})
2: $\mathbb{T} \leftarrow \emptyset$
3: **for** $i = 1 : n$ **do**
4: $\quad s^i_t \leftarrow$ `Cluster`$(b^{\xi^i}, b^{\xi}_{init}, B, \epsilon)$
5: $\quad T^i \leftarrow$ `DepthLimitedSearch`(s^i_t)
6: $\quad \mathbb{T} \leftarrow \mathbb{T} \cup T^i$
7: **end for**
8: **return** $P \leftarrow$ `Dp`$(\mathbb{T})$

DP solution to be computationally competitive, a faster tree search method should replace lines 7-9 of Alg. 2. It suffices to use near-greedy depth-limited tree search because most of the information gain is captured by locally greedy paths within clusters. This replacement `DepthLimitedSearch` is shown in line 5 of Alg. 3. In addition, $\mathbb{T}$ is the set of cluster trees with which `Dp` finds the optimal time-allocation solution. The cluster time budget in line 4 is a fraction ϵ greater than the belief proportion b^{ξ^i} in b^{ξ}_{init} for DP to be able to trade time among clusters.

IV. Numerical Simulations

We analyse two experiments using Monte-Carlo simulation. The first selects an appropriate exploration weight for the MCTS algorithm, while the second compares variants of MCTS and clustering with depth-limited tree search on maps of different thermal locations and numbers of clusters.

A. MCTS Exploration Weight Selection

We aim to choose a weight C_p in (12) that is appropriate for the complex map scenario from [5], illustrated in Fig. 3. If this C_p value can provide a good plan for the complex map, we reason that it will be suitable for other interesting scenarios. Parameters of the glider motion and thermal models are identical to those in [5]. The budget mission time is likewise $B = 60$ minutes, a practically realistic value.

We evaluate discrete values of C_p over $[0, 1]$, each one with 50 trials, record their final utilities and computational times, and compare them against (full) depth-limited tree

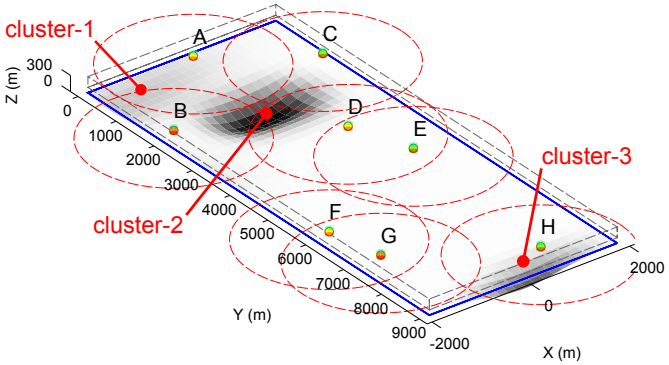
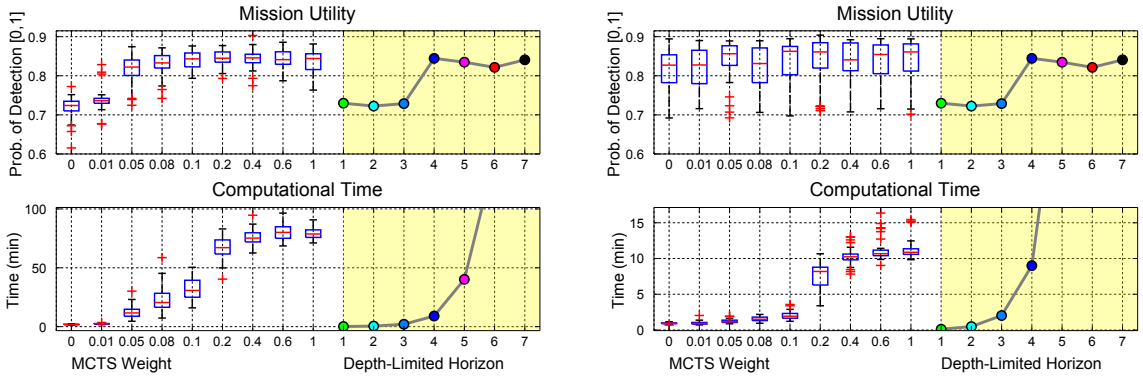


Fig. 3. Complex map from [5] with three clusters of belief uncertainty (or information) represented by the dark patches. Thermals are labelled A to H.



(a) Full-MCTS search (b) Sequential-Cluster-MCTS search

Fig. 4. Full and Sequential-Cluster MCTS searches with 50 trials for each weight C_p vs. (full) depth-limited tree search on complex map (Fig. 3).

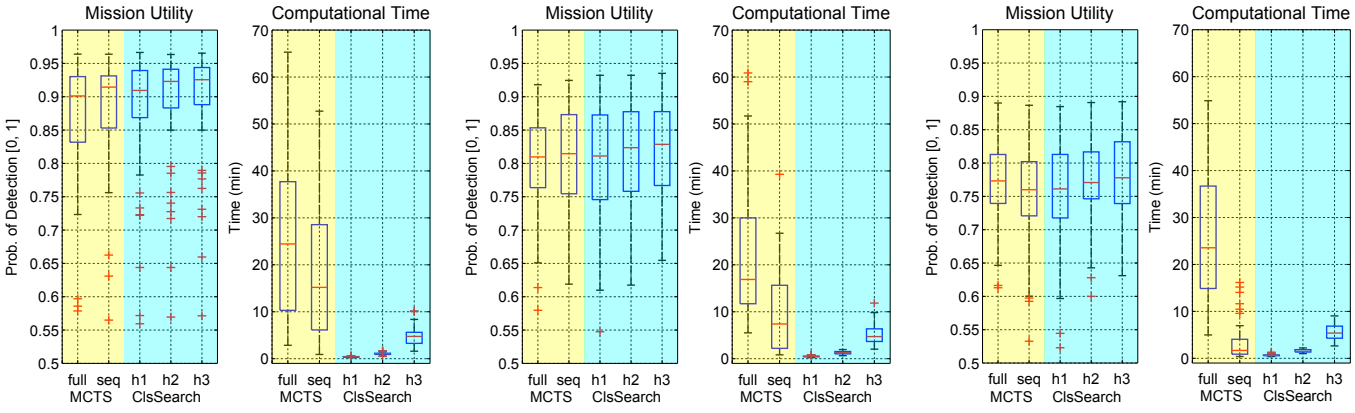
search from [5]. The results of Full-MCTS search and the seqMCTS (Sequential-Cluster-MCTS) search version (Alg. 2) are laid out respectively in Figs. 4(a) and 4(b). Full-MCTS, Fig. 4(a), achieves comparable utility performance to the best horizon of depth-limited search ($= 4$) with $C_p \geq 0.05$ and reasonable computational time for $C_p \in [0.05, 0.1]$. In contrast, Fig. 4(b) portrays the dexterity of seqMCTS; solving cluster subproblems consistently yields higher utilities across all weights and requires very much lower computational times. This result confirms our initial observation that near-optimal plans concentrate search effort at clusters and transition between them with minimum-cost paths.

For MCTS search, $C_p = 0$ corresponds to depth-first search, while increasing its value approaches breadth-first search. In Fig. 4(b), the discernable jump in computational time between $C_p = 0.1$ and 0.2 is the product of a large increase in branching such that tree search almost resembles breadth-first. This is due to a tight difference between the exploration and exploitation terms in (12) around these C_p values. Additionally, the greater variance in utility across 50 trials in Fig. 4(b) compared to Fig. 4(a) arises from `TspRewire` in Alg. 2 intermediately removing low-yield path segments to find better plans in future clusters. This sometimes works well as seen with utilities ≈ 0.9. At other times, the minor utilities of removed low-yield paths add up to marginally reduce the final utility. We also have to accept that on rarer occasions, `TspRewire` removes too many path segments from intermediate clusters; the algorithm cannot recover, resulting in poor outlier solutions.

From this analysis, we choose $C_p = 0.1$ for the experiments in Sec. IV-B because it can provide high utility plans with relatively low computational time (avg. 2mins from Fig. 4(b)).

B. Comparison of Search Algorithms

To prove that the MCTS and clustering algorithms generalise to any informative soaring problem, we apply them on three maps with $\{2, 3, 4\}$ clusters. Each map is tested with 50 setups of 10 randomised thermal positions. The algorithms implemented are: 1) Full-MCTS, 2) seqMCTS (Sequential-Cluster-MCTS) (Alg. 2), and 3) ClsSearch (depth-limited cluster tree search) (Alg. 3) with three horizons $\{1, 2, 3\}$ and $\epsilon = 20\%$. The simulation results are displayed in Fig. 5. Spatial plan instances for the 2-cluster map with: 1) Full-MCTS, and 2) horizon-3 ClsSearch are visualised in Fig. 6 and Fig. 7.



(a) 2-cluster map (b) 3-cluster map (c) 4-cluster map

Fig. 5. Simulation results comparing MCTS and depth-limited cluster tree search in yellow and cyan fill respectively. The two variants of MCTS search are: 1) Full-MCTS, and 2) Sequential-Cluster-MCTS (Alg. 2). *ClsSearch* is depth-limited cluster tree search (Alg. 3) with finite horizons $\{1, 2, 3\}$ using dynamic programming (DP) for optimal cluster time allocation. Every algorithm is trialled on the same 50 setups of 10 randomised thermal locations for each map.

In Fig. 5, the utilities from all methods have high variances because some setups of randomised thermals offer poor map coverage; thus, their maximum theoretic utility is low. With this in mind, Fig. 5 provides a few algorithm performance insights. Firstly, the utility of SeqMCTS is generally better than Full-MCTS because it isolates and searches clusters of information. Performance between the two are comparable in Fig. 5(c) as clusters are spatially closer together. Secondly, SeqMCTS utility is better than greedy (horizon-1) ClsSearch. However, the horizon-$2, 3$ variants tend to outperform MCTS since they build more cluster tree nodes with intermediate $J(v)$ and $C(v)$ than greedy. This offers DP more intermediate options to optimally allocate search time among clusters. Thirdly, while there is no trend in computational time with the number of clusters for Full-MCTS, there is a monotonic decreasing trend for SeqMCTS. This desirable trait is the payoff of dividing the problem into efficiently manageable parts. Likewise, computational efficiency is mirrored in ClsSearch; at this scale, computational effort is dictated by the horizon size. In conclusion, horizon-$2, 3$ ClsSearches offer the best performances with low computational cost.

V. Conclusion and Future Work

We have presented two algorithms that solve the long-term informative soaring problem, where a gliding UAV conducts a search mission for a lost ground target while periodically replenishing energy at thermal energy nodes. We verified the effectiveness of these algorithms using Monte-Carlo simulation. The significance of our work is in generating high-quality nonmyopic plans for large-scale informative soaring problems with greatly reduced computational effort, such that it can be implemented on physical hardware systems.

For general, large-scale problems characterised by tight information clusters, our first algorithm, an extension of MCTS, preserves the information gathering performance of depth-limited search [5], but also offers more predictable solutions as a function of the MCTS exploration tuning variable. This variable offers anytime performance in practice. For scenarios involving many distinct clusters of information, our results have validated the superior performance of our second search algorithm. It is capable of generating high-quality plans with very little computational cost by virtue of dividing the problem into efficiently manageable subproblems.

Future work will address unknown thermals, annexing energy-uncertainty into our planning framework. The goal will be to detect and maintain estimation of their states, while simultaneously conducting the search mission. As a randomised sampling planner, MCTS is a strong candidate for addressing uncertainty. We additionally look towards demonstrating this work on a gliding UAV platform.

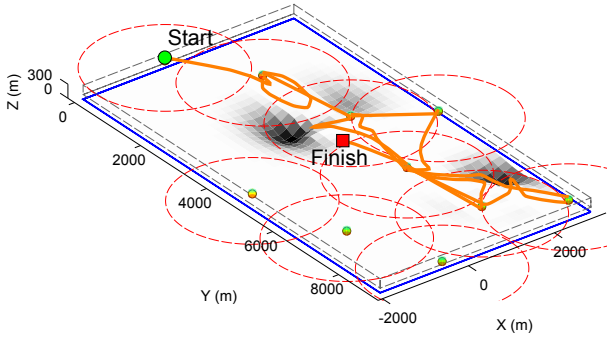


Fig. 6. Full-MCTS on an instance of 10 randomised thermals on the 2-cluster map. The final map profile is shown, and the final utility is 0.8488.

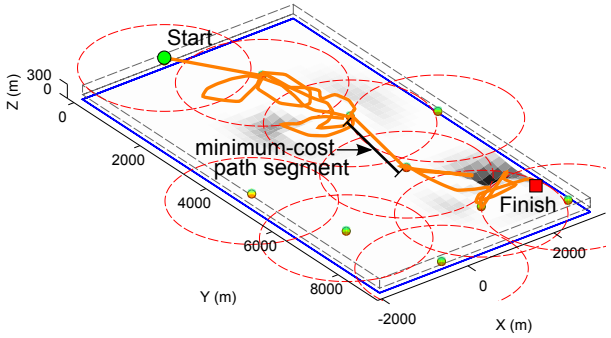


Fig. 7. Horizon-3 ClsSearch on the same instance in Fig. 6. The final utility is 0.9133. Two trees are generated for each cluster and DP finds the optimal plan. The cluster plans are linked by one minimum-cost path segment.

References

[1] M. J. Allen, "Autonomous soaring for improved endurance of a small uninhabited air vehicle," in *Proc. of AIAA ASM*, 2005.

[2] D. J. Edwards and L. M. Silverberg, "Autonomous soaring: The Montague cross-country challenge," *J. Aircraft*, vol. 47, no. 5, pp. 1763 – 1769, 2010.

[3] J. W. Langelaan, "Gust energy extraction for mini and micro uninhabited aerial vehicles," *J. Guid. Control Dynam.*, vol. 32, no. 2, pp. 464–473, 2009.

[4] N. R. Lawrance and S. Sukkarieh, "Autonomous exploration of a wind field with a gliding aircraft," *J. Guid. Control Dynam.*, vol. 34, no. 3, pp. 719–733, 2011.

[5] J. L. Nguyen, N. R. Lawrance, R. Fitch, and S. Sukkarieh, "Energy-constrained motion planning for information gathering with autonomous aerial soaring," in *Proc. of IEEE ICRA*, 2013.

[6] N. Mathew, S. L. Smith, and S. L. Waslander, "A graph-based approach to multi-robot rendezvous for recharging in persistent tasks," in *Proc. of IEEE ICRA*, 2013.

[7] J. A. Cobano, D. Alejo, S. Vera, G. Heredia, and A. Ollero, "Multiple gliding uav coordination for static soaring in real time applications," in *Proc. of IEEE ICRA*, 2013.

[8] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs," in *46th Annual IEEE Symposium on FOCS*, 2005.

[9] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning," in *Proc. of IEEE ICRA*, 2012.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Trans. Comput. Intell. AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[11] E. Wong, F. Bourgault, and T. Furukawa, "Multi-vehicle bayesian search for multiple lost targets," in *Proc. of IEEE ICRA*, 2005.

[12] N. R. J. Lawrance, "Autonomous soaring flight for unmanned aerial vehicles," Ph.D. dissertation, The University of Sydney, 2011.

[13] M. J. Cutler, T. W. McLain, R. W. Beard, and B. Capozzi, "Energy harvesting and mission effectiveness for small unmanned aircraft," in *Proc. of AIAA GNC*, 2010.

[14] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006*. Springer, 2006, pp. 282–293.

[15] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. in Appl. Math.*, vol. 6, no. 1, pp. 4–22, 1985.

[16] C. M. Bishop, *Pattern recognition and machine learning (information science and statistics)*. Springer, 2007.