2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multi-query TDSP for Path Planning in Time-varying Flow Fields

James Ju Heon Lee[1], Chanyeol Yoo[1], Stuart Anstee[2] and Robert Fitch[2]

Abstract— **Many applications of path planning in time-varying flow fields, particularly in areas such as marine robotics and ship routing, can be modelled as instances of the *time-varying shortest path (TDSP)* problem. Although there are no known polynomial-time solutions to TDSP in general, our recent work has identified a tractable case where the flow is modelled as piecewise constant. Extending this method to allow for computational reuse in larger multi-query problems, however, requires additional thought. This paper shows that the piecewise-linear form of the cost function employed in previously work can be used to build an analogy of a shortest path tree, thereby enabling optimal concatenation of sub-problem solutions in the absence of an optimal substructure, and without uniform time discretisation. We present a framework for multi-query TDSP that finds an optimal path that passes through a defined sequence of waypoints and is computationally efficient. Performance comparison is provided in simulation that shows large (up to 100x) speedup compared to a naive approach. This result is significant for applications such as ship routing, where route evaluation is a desirable capability.**

I. Introduction

Path planning problems where costs can change over time are surprisingly difficult compared to corresponding time-invariant versions with well-known solutions. Planning in ocean currents is an interesting example with important applications such as marine robotics [1–6] and ship routing [7, 8]. This work considers the *time-dependent shortest path (TDSP)* problem in flow fields in a multi-query setting that, for example, allows for a cost evaluation for a path that must visit a sequence of waypoints, such as would appear in route evaluation for ship routing among other applications.

Recent theoretical progress in solving TDSP problems has shown that polynomial-time solutions are possible for a particular variant relevant to planning in time-varying flow fields [9, 10]. This variant models the flow field as piecewise time-invariant, that is, changing instantaneously at fixed time intervals and remaining fixed during the intervening periods. This is analogous to accepting the output of a numerical forecasting system, which predicts the flow at a sequence of fixed times, as being literally true. Although the performance of this approach is encouraging, the question of how it can be extended to more complicated problem settings through computational reuse remains unanswered.

Multi-query vehicle routing problems arise in applications such as ship routing and scientific observations, where the vessel or vehicle is required to visit a sequence of ports or sampling stations in some defined order, but there is scope to vary factors such as departure times to minimise the total travel time or fuel expenditure. Efficient approaches to such problems often rely on computational reuse in the time-invariant case as a means to achieve computational efficiency.

A critical challenge in solving TDSP problems in flow fields is that typical problem formulations do not necessarily have the property of optimal substructure, which allows optimal solutions to subproblems to be readily combined [11]. In the present case, departure and arrival times would have to coincide to permit combinations of sub-paths, and so solutions at arbitrary times would have to be computed and stored. Further, it is known that small deviations in departure time can lead to major deviations in arrival time [12]. This property disallows techniques such as uniform discretisation of start time as a way to pre-compute paths for later reuse.

The key advance we make in this paper is to observe that a piecewise-linear representation of the cost function used in the single-query solution possesses properties that can be exploited to avoid the need for time discretisation and thereby combine solutions to subproblems in the absence of optimal substructure. We present a method that builds a structure analogous to a shortest path tree by storing pre-computed paths to any given node in a graph of locations. Using this structure, we then derive a multi-query solution for TDSP in flow fields that precisely resolves the alignment between arrival and departure times at each node. The algorithm takes as input a desired sequence of nodes and returns an optimal path that passes through them. The time complexity of this algorithm is polynomial in the length of the input sequence and we provide evaluations in simulation using ocean forecast data that demonstrate its substantial performance advantage relative to a naive approach.

The main contribution of this work is a multi-query solution to TDSP problems in flow fields, which was previously unavailable due to the challenge of computational reuse. Our results are significant for a wide range of applications in marine robotics and ship routing, which have been limited to poor approximations or impractically small problem sizes due to the complexity of the underlying TDSP problem.

II. Related Work

A common approach to solving for an optimal path over time-varying flow fields is to discretise time. There have been multiple variants of A* [13–15] that generate a time-optimal path in a time-varying ocean current. A recent version implements the A* algorithm along a graph that adaptively samples in spatio-temporal space reflecting the

*This work is supported by Australia's Defence Science and Technology Group and the University of Technology Sydney.

[1]Authors are with the University of Technology Sydney, Ultimo, NSW 2006, Australia {JuHeon.Lee,Chanyeol.Yoo,Robert.Fitch}@uts.edu.au

[2]Author is with the Defence Science and Technology Group, Department of Defence, Australia Stuart.Anstee@defence.gov.au

time-varying flow field [16, 17]. Beyond the A* approaches, recent work [18] proposes a planning framework for time-varying currents that finds a locally optimal solution at every time step. A well-known issue with the discrete-time approach is that the solution is resolution complete in time. This property makes reusing results for different initial states difficult without a finer time resolution, as deviations in starting time not captured by the original resolution can result in a suboptimal solution.

Approaches for solving an optimal path in time-varying flow fields in continuous time include level set methods [19, 20] and time-varying Markov decision processes [21]. Both assume a general edge cost function, which makes their worst-case computational complexity non-polynomial.

Our previous work on TDSP planning in time-varying flow fields showed that it is possible to generate a travel time-optimal policy in polynomial time [9, 10]. We also showed that the same policy could generate a travel time-optimal path for different starting times. In this paper, we extend this approach by reformulating it as a multi-query algorithm, where it can generate a travel time-optimal path using the same sets of travel time-optimal policies.

One benefit of formulating the TDSP algorithm for multi-query application is that it allows for fast action evaluation. This property helps address TDSP variant of multi-robot coordination by applying our work in well-known multi-robot algorithms [22–26]. It can also potentially address TDSP variants of the travelling salesman problem [27], vehicle routing problems [28] and their variants [3, 29].

III. Background

A. Vehicle dynamics in time-varying flow fields

Suppose $\mathbf{v}_v(\mathbf{x}(t), \mathbf{u}(t)) = [u_v(t), v_v(t)]^T$ is the still-water velocity of a given vehicle with n-dimensional state $\mathbf{x}(t)$ and m-dimensional control input $\mathbf{u}(t)$. Then, the vehicle's motion across a time-varying 2D flow field environment $\mathbf{v}_c(\mathbf{x}(t), t) = [u_c(t), v_c(t)]^T$ is defined as:

$$\dot{\mathbf{x}}(t) = \mathbf{v}_v(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{v}_c(\mathbf{x}(t), t). \tag{1}$$

Remark 1. *The vector flow field is modelled as* piecewise-constant *at each successive time interval. This representation matches the output of ocean forecasting systems.*

Remark 2. *The vehicle operates under a sparse control scheme, common for operating* autonomous underwater gliders*, where the vehicle control input is changed infrequently and remains constant between updates [30, 31].*

We define an $(n+1)$-state path $\psi = \mathbf{x}_0\mathbf{x}_1 \cdots \mathbf{x}_n$ as the sequence of vehicle states where control changes occur. We denote by $\psi_k = \mathbf{x}_0\mathbf{x}_1 \cdots \mathbf{x}_k$ the path ψ up to the k-th state, and by $\psi[k] = \mathbf{x}_k$ the k-th state for $k \in [0, \cdots, n]$.

B. Arrival and travel time

The *arrival time* $a_{\mathbf{x}\mathbf{x}'}(t)$ is the time when a vehicle departing from state $\mathbf{x}$ at time t would arrive at a state $\mathbf{x}'$ in reference to a datum time (i.e., $t = 0$). For example, if a vehicle takes 2 seconds to travel from $\mathbf{x}$ to $\mathbf{x}'$ after departing at $t = 3$, then $a_{\mathbf{x}\mathbf{x}'}(3) = 5$. Assuming waiting is not allowed, i.e., the vehicle must depart as soon as it arrives, the recursive arrival time representation of path ψ is:

$$a_\psi(t) = a_{\mathbf{x}_{n-1}\mathbf{x}_n}(a_{\psi_{n-1}}(t)). \tag{2}$$

The *travel time* $T_{\mathbf{x}\mathbf{x}'}(t)$ is the time duration for the vehicle to move from state $\mathbf{x}$ to $\mathbf{x}'$ departing at time t. The travel time from the previous example is $T_{\mathbf{x}\mathbf{x}'}(3) = 2$. Assuming no waiting, the travel time of path ψ is:

$$T_\psi(t) = a_\psi(t) - t. \tag{3}$$

C. Problem formulation

The objective is for a vehicle in a time-varying flow field to visit every user-defined goal state at least once while minimising the travel time, which is a variant of the Travelling Salesman Problem. Formally:

Problem 1 (Travelling Salesman Problem on time-varying flow fields)**.** *Given a vehicle velocity model $\mathbf{v}_v$, time-varying flow field $\mathbf{v}_c$ in piecewise-constant form, initial vehicle state $\mathbf{x}_{init}$, a deployment time t_0, and a set of goal vehicle states $\mathbf{X}_{goal}$ that the vehicle must visit at least once, find the path ψ^* that minimises the travel time:*

$$\begin{aligned} &\psi^* = \arg\min_{\psi} a_\psi(t_0) - t_0 \\ &\textit{s.t. } \psi[0] = \mathbf{x}_{init} \textit{ and } \mathbf{X}_{goal} \subset \psi. \end{aligned} \tag{4}$$

Minimising travel time is more applicable for robotics than arrival time, as vehicle energy costs are often proportionate to mission duration. The time-varying flow field may exhibit *non-first-in-first-out (non-FIFO)* properties; that is, temporal changes in the flow velocity may allow the vehicle to arrive earlier at a downstream position by departing later [32]. We assume waiting at a state is not allowed, but allow revisiting a state. The resulting cyclic path is the equivalent of waiting at a state for the duration of the cycle. Therefore, we allow revisiting goal states in the TSP solution for a time-varying flow field. Unlike static TSP, such a path can still be optimal due to the non-FIFO properties.

IV. Multi-query TDSP in Flow Field

We previously presented a single-query approach to the TDSP problem in time-varying flow fields with non-FIFO properties that solves it in polynomial time [9, 10]. In this section, we propose a multi-query framework variant.

The framework consists of *offline* and *online* components. The offline component computes the travel policies for all goal positions $\mathbf{X}_{goal}$ *a priori* across a graph representation of the time-varying flow field. The online component extracts the policy for a goal position and synthesises a time-optimal path from any initial vehicle position that may not be on any vertices. In Sec. V, we show that the online component can generate the optimal path in linear time.

A. Time-dependent directed graph

A time-dependent directed graph $G = (S, E)$ is defined by a finite set of states sampled in an n-dimensional free space $S \subset \mathbb{R}^n$ and edges $(s, s') \in E$ where $s, s' \in S$. The graph state s here can represent the vehicle state $\mathbf{x}$ or its position. We expand on the definition of the $(N+1)$-length path as a sequence of graph states $\psi = s_0 s_1 \cdots s_N$ where $(s_k, s_{k+1}) \in E \ \forall k \in [0, N)$. The *neighbourhood* of state s is defined as a set of states $S_s \subseteq S$ that are immediately reachable from s such that $(s, s') \in E$ for $s' \in S_s$. A set of goal states is defined as $S_g \subset S$ where $|S_g| \geq 1$.

Remark 3. *A self-transition edge is added at each goal state. I.e.,* $(s_g, s_g) \in E \ \forall s_g \in S_g$ *with* $C_{s_g s_g}(t) = 0 \ \forall t \geq 0$.

Each edge $(s, s') \in E$ is associated with a piecewise-constant *edge cost* $C_{ss'}(t)$ that represents the time to traverse from state s to s' when *starting* at time t. The partitioning of the time subdomains is taken from the ocean forecast datasets. The edge cost to move through a time-varying flow is derived by forward integrating (1) from s at time t over a finite test set of vehicle controls $\mathbf{u} \in \mathbf{U}$, then choosing the travel time of one of the test paths that pass sufficiently close to s'. Using the control scheme outlined in Remark 2, the control parameter for each forward integration is constant.

Remark 4. *As the vehicle traverses faster moving downstream,* $C_{ss'}(t) \neq C_{s's}(t)$ *for some* $t \in \mathbb{R}$. *That is, the graph has directional properties.*

Remark 5. *Because the edge costs are evaluated by forward integration (1), the corresponding paths between pairs of states are continuous and curved.*

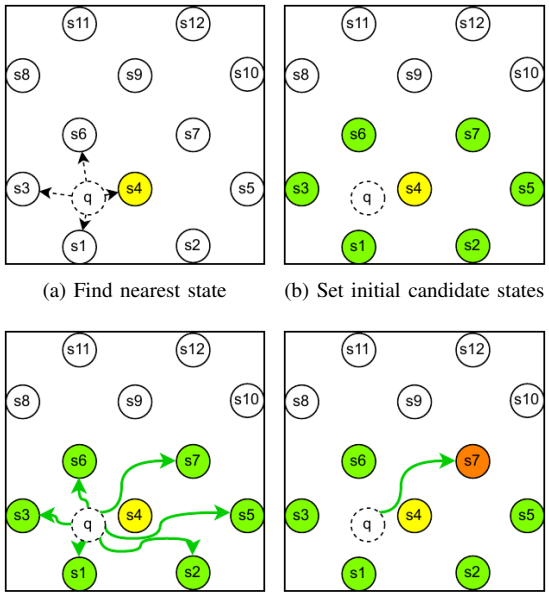
B. Constructing the TDSP multi-query framework

We present how the offline component constructs the multi-query framework using the single-query TDSP approach. We present how the offline component constructs the multi-query framework using the single-query TDSP approach. We showed in [10] that we can formally re-express the travel time for a path up to the k-th state $\psi = s_0 s_1 \cdots s_k$ over graph G from (3) to a piecewise-constant Bellman equation. Abusing the notation so that T_s^{k+1} represents the travel time after $k+1$ state transition from state s:

$$T_s^{k+1}(t) = \begin{cases} \vdots & \vdots \\ \min\limits_{s' \in S_s} C_{ss'}(t) + T_{s'}^{k}(t + C_{ss'}(t)) & \text{if } t > t_i \\ \vdots & \vdots \end{cases}, \tag{5}$$

where t_i is the i-th time subdomain. We define the *converged* travel time $T_s^*(t)$ as when $T_s^{k+1}(t) = T_s^k(t) \ \forall t \in \mathbb{R}$. That is, further state transitions do not change the travel time.

Example 1 (Path with converged travel time)**.** *Consider a k-length path ψ where each state transition satisfied (5) and $s_{k-1} \in S_g$. Then by (5) and Remark 3, the only valid transition beyond s_{k-1} is the self-transition. Therefore, such path has a converged travel time $T_s^*(t) = T_s^k(t) = T_s^{k+1}(t)$.*



(a) Find nearest state (b) Set initial candidate states

(c) Deriving travel times (d) Optimal initial state

Fig. 1. Step-by-step visualisation of an example choosing the optimal initial state from the initial vehicle position q. Graph edges are omitted.

The resulting $T_s^*(t)$ from Example 1 is the optimal travel time function to a goal state. Formally, the optimal travel time for a path from s to the goal state s_g is expressed as:

$$T_{s,s_g}^*(t) = \begin{cases} \vdots & \vdots \\ \min\limits_{s' \in S_s} C_{ss'}(t) + T_{s',s_g}^*(t + C_{ss'}(t)) & \text{if } t > t_i \\ \vdots & \vdots \end{cases}. \tag{6}$$

The TDSP path is defined by a piecewise-constant *travel policy* $\pi_s(t)$. The policy represents the state transition from s within some time subdomain. Formally, the optimal travel time policy that minimises the path's travel time to the goal state s_g is expressed as:

$$\pi_{s,s_g}^*(t) = \begin{cases} \vdots & \vdots \\ \arg\min\limits_{s' \in S_s} C_{ss'}(t) + T_{s',s_g}^*(t + C_{ss'}(t)) & \text{if } t > t_i \\ \vdots & \vdots \end{cases}. \tag{7}$$

We define $\Pi_{s_g} = \{\pi_{s,s_g}^* \mid \forall s \in S\}$ as a set of optimal travel policies to goal state s_g from all states in S. We also define $\mathcal{T}_{s_g} = \{T_{s,s_g}^* \mid \forall s \in S\}$ as a set of optimal travel time functions to goal state s_g from all states in S.. Both Π_{s_g} and $\mathcal{T}_{s_g}$ are evaluated directly by solving the Bellman equation (5). We construct the TDSP multi-query framework by solving Π_{s_g} and $\mathcal{T}_{s_g}$ for all $s_g \in S_g$, then cache them for use in the online component. We denote the cached policies and travel time functions as $\Pi = \{\Pi_{s_g} \mid \forall s_g \in S_g\}$ and $\mathcal{T} = \{\mathcal{T}_{s_g} \mid \forall s_g \in S_g\}$ respectively.

C. Querying from the TDSP multi-query framework

We present how the TDSP is queried online from the multi-query framework defined in Sec. IV-B. We assume

that all desired goal vehicle states $\mathbf{X}_{goal}$ are represented in a time-dependent directed graph G, but the initial vehicle position q may not be represented.

Given the time-dependent directed graph G, cached policies Π, and travel time functions $\mathcal{T}$ from the offline component, we construct a travel time-optimal path from the deployment time t_0, desired goal state $s_g \in S$, and the initial vehicle position q. First, we query the set of optimal travel time functions $\mathcal{T}_{s_g} \in \mathcal{T}$ and the corresponding set of optimal travel policies $\Pi_{s_g} \in \Pi$ for the desired goal state s_g. We then connect an edge from the initial vehicle position q to graph G such that $(q, s_q) \in E \ \forall s_q \in S_{\bar{s}}$, where:

$$\bar{s} = \arg\min_{s \in S} \|s - q\|, \tag{8}$$

and derive the *query travel time* τ_{s_i} from q departing at t_0 to each neighbouring state $s_i \in S_{s_q}$. We do this by forward integrating (1) over set of controls $\mathbf{u} \in \mathbf{U}$, using the same control scheme as shown in Sec. IV-B. Finally, we choose $s_{init} \in S_{\bar{s}}$ that minimises the path travel time to s_g at the corresponding query time. That is, given $T^*_{s,s_g} \in \mathcal{T}_{s_g}$,

$$s_{init} = \arg\min_{s \in S_{\bar{s}}} T^*_{s,s_g}(t_0 + \tau_s), \tag{9}$$

If we want to evaluate the travel time from q to s_g, we call the optimal travel time function $T^*_{s_{init},s_g}(t_0 + \tau_s)$. If we want to evaluate the TDSP in graph space, we follow the optimal travel policies, starting from $\pi^*_{s_{init},s_g}(t_0 + \tau_s)$ and evaluating the time at each state with T^*_{s,s_g}. Figure 1 visualises this approach to finding the optimal initial state.

V. Analysis

A. Multi-query policy computation

The worst case time complexity for building all edge cost functions in the graph G over time-varying flow fields is $\mathcal{O}(|S|^2|\mathbf{U}||\mathbf{T}|)$, as we forward integrate the trajectories over $|\mathbf{U}|$ number of controls for $|\mathbf{T}|$ number of time partitions. We can reduce this time complexity to $\mathcal{O}(|S|)$ by generating the trajectories in parallel and reusing them when evaluating the edge cost to other neighbouring states.

The time complexity to generate optimal policies to a single goal state in graph G is $\mathcal{O}((|S||C_m|)^{k+2})$, where $|S|$ is the number of states in the graph G, $|C_m|$ is the worst case number of subdomains over the edge time function, and k is the number of state transitions [10]. Solving a problem with multiple goal states requires generating optimal policies for each goal multiple times. For example, a brute-force solution of a Vehicle Routing Problem (VRP) requires the generation of $|S| \cdot |S|!$ number of optimal policies. However, recall that we can synthesise an optimal path solution from any initial state $s_i \in S$ to the goal state using a set of policies Π_{s_g}. By caching the set of policies for all $s_g \in S$, we avoid resolving for policies that share the same goal state. Hence, the policy only needs to be generated at most $|S|$ times, and the overall time complexity remains polynomial, at $\mathcal{O}(|S|^{k+3}|C_m|^{k+2})$.

Consider that policies for all goal states $s_g \in S$ can be constructed independently from each other. As such, using parallel processing can significantly speed up policy generation and reduce the overall time complexity of generating a single set of policies to $\mathcal{O}((|S||C_m|)^{k+2})$.

B. Multi-query policy evaluation

As discussed in Sec. IV-C, extracting a time-optimal path from the sets of policies comes in four parts. We first extract the set of travel policies for a given goal state s_g. Then, we find the nearest state $\bar{s}$ to the initial vehicle position q. Next, we query the time from q to each of the states $s_q \in S_{\bar{s}}$. Finally, we find the starting state $s_{init} \in S_{\bar{s}}$ that minimises travel time from q to s_g.

The worst-case time complexity for extracting the sets of travel policies and finding the nearest state to q is $\mathcal{O}(|S|)$. Similar to when we evaluated the edge cost between states, we can reduce the time complexity for evaluating the query time to $\mathcal{O}(1)$ by caching the forward integrated trajectories. Otherwise, the worst case time complexity is $\mathcal{O}(|S|)$. Finally, the worst-case time complexity for finding the optimal starting state is $\mathcal{O}(|S|)$. Therefore, the overall time complexity of extracting the solution from the policy lookup table is $\mathcal{O}(|S|)$ with parallel processing. Otherwise, it is $\mathcal{O}(|S|^2)$.

C. Practical application of multi-query approach

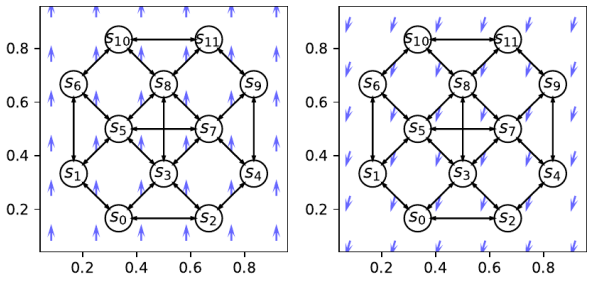
The construction of the multi-query framework is useful in a number of practical applications. One important use is the option to offload the burden of computing a TDSP solution from robotic platforms with limited computing power. Maritime applications are a prime example. Directly computing a TDSP solution over a time-varying environment in a large search space is computationally demanding and well beyond the capabilities of a marine robot. By pre-generating the policies, the robot can simply query the optimal solution for a given start and goal state.

VI. Simulation

We present our multi-query TDSP approach by using it to solve the TSP in time-varying flow fields. We first illustrate its performance by comparing the computation time between multi-query and single-query approaches when exhaustively solving for a TSP. We also compare the computation time between our framework and a naive sequential method to evaluate a path's travel time. Finally, we solve the TSP with real-world time-varying ocean forecast data. We ran all experiments on a standard desktop with Intel i5-9500 3.00GHz CPU and 32GB RAM.

A. Computation of TSP with time-varying flow fields

We compare the computation time to exhaustively solve for the travel time-optimal TSP path between various methods to highlight the computational efficiency of our approach. The *single-query* approach computes the travel time presented in [10] for every possible pair of vehicle positions. The *policy caching* approach is similar to the single-query approach, except that it caches the policies for the goal state and reuses them when evaluating paths with the same goal state. The *multi-query* approach uses



(a) t = 0 second (b) t = 1 second

Fig. 2. Example time-dependent directed graph with states (circle nodes) and edges (black arrows). The flow field (blue quiver) changes with time.

the precomputed policies to compute the travel time as presented in Sec. IV-C. We evaluate the computation time for different numbers of goal states, chosen from the following set: $\{s_1, s_2, s_3, s_4, s_6, s_8, s_9\}$.

Figure 2 illustrates the TSP problem setup. Starting at the initial vehicle position $q = [0.8, 0.167]^T$, a vehicle with a constant forward speed of 0.5 m/s must visit all goal states at least once. The flow field has a constant magnitude of 0.45 m/s, and its flow direction rotates counter-clockwise with a period of 2.2 seconds.

Table. I shows the computation time comparison. Any time longer than 1800 seconds is excluded. As expected, the computation time for the single-query approach grows exponentially with the number of goal states. The computation time for the policy cache approach shows a linear relation to the number of goal states, as the policies for each goal only need to be computed once. Finally, the multi-query approach finds the travel time-optimal path for all tested numbers of goal states in less than a second. The computation time grows exponentially, while the increase in the number of goal state sequences is factorial.

B. Comparison with a sequential approach

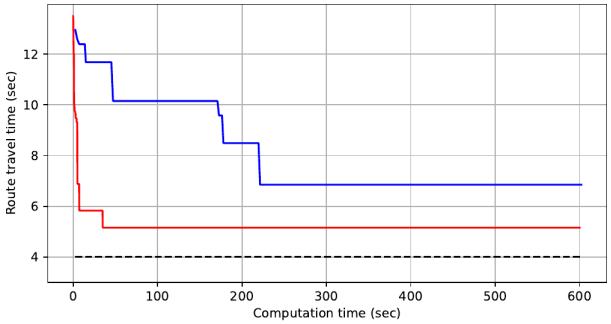
We compare our approach against a naive *sequential path evaluation* method, which is a control-driven approach. Given a path over graph G, this approach directly solves equation (3) by forward integrating the vehicle model (1) over the node sequence.

We compare the approaches over the graph shown in Fig. 2, with all nodes set as goal states ($S_g = S$), giving 12! permutations of goal state sequences. For a fair comparison, we evaluate our multi-query and control-driven approach over the same path randomly generated with the following constraints: 1) Each path always starts from a user-defined initial state. 2) Each path is built from a random sequence

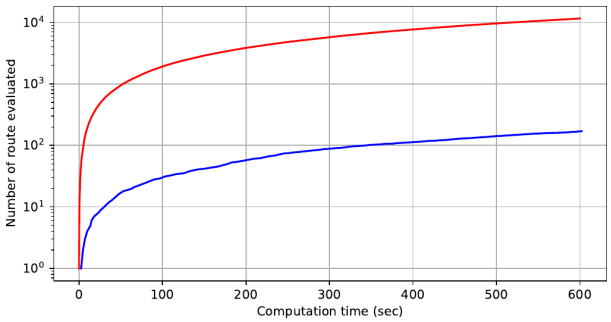
TABLE I

COMPARING COMPUTATION TIME BETWEEN SINGLE AND MULTI-QUERY TDSP FOR SOLVING TSP IN TIME-VARYING FLOW FIELD.

$\lvert S_g \rvert$	Single-query	Policy cache	Multi-query
3	111.765	53.755	0.029
4	374.951	63.059	0.045
5	1758.341	72.304	0.070
6	$>$ 1800	95.865	0.140
7	$>$ 1800	112.162	0.482



(a) Plot of best travel time out of randomly sampled routes versus run time. The dashed black line is the precomputed optimal travel time.



(b) Plot of number of routes evaluated versus run time (log scale).

Fig. 3. Analysis for route evaluation in a time-varying flow with non-FIFO properties between the multi-query policy (red) and the control-driven (blue) approach. Each approach evaluates a series of randomly generated routes for 600 seconds but only counts valid routes to the run time.

of state transitions along graph edges. 3) All generated paths visit all goal states at least once, and total travel time must not exceed 30 seconds. The comparison excludes any time spent generating and evaluating invalid paths.

For both methods, we evaluate the travel time along as many randomly generated paths as possible for 600 seconds. After each path evaluation, we save the path count and the historically best travel time. We also compare both methods against the exhaustively evaluated true optimal travel time.

Figure 3 shows the empirical analysis. The multi-query method evaluates randomly-chosen routes roughly 100 times faster than the sequential path evaluation method. As a result, the multi-query method converges closer and earlier to the optimal solution.

C. TSP over time-varying forecast data

Given a real-world ocean forecast, we exhaustively solve for the travel time-optimal path that visits 9 goal states. The forecast was generated using the *Relocatable Ocean Atmosphere Model (ROAM)*, providing an hourly partitioned 5-day forecast starting from 25 November 2022. The strongest oceanic flow strength was forecasted as 0.22 m/s and occurred during the second day. The vehicle's initial position is set to $q = [-31.825, 151.690]^T$ in geographical coordinates and travels with a constant speed of 0.3 m/s.

Figure 4 shows a time-lapse representation of the vehicle visiting each of the 9 goal states, which was exhaustively searched in 2 minutes using the multi-query approach. Note that using either a single-query or policy cache method would result in the same path, only it would take considerably

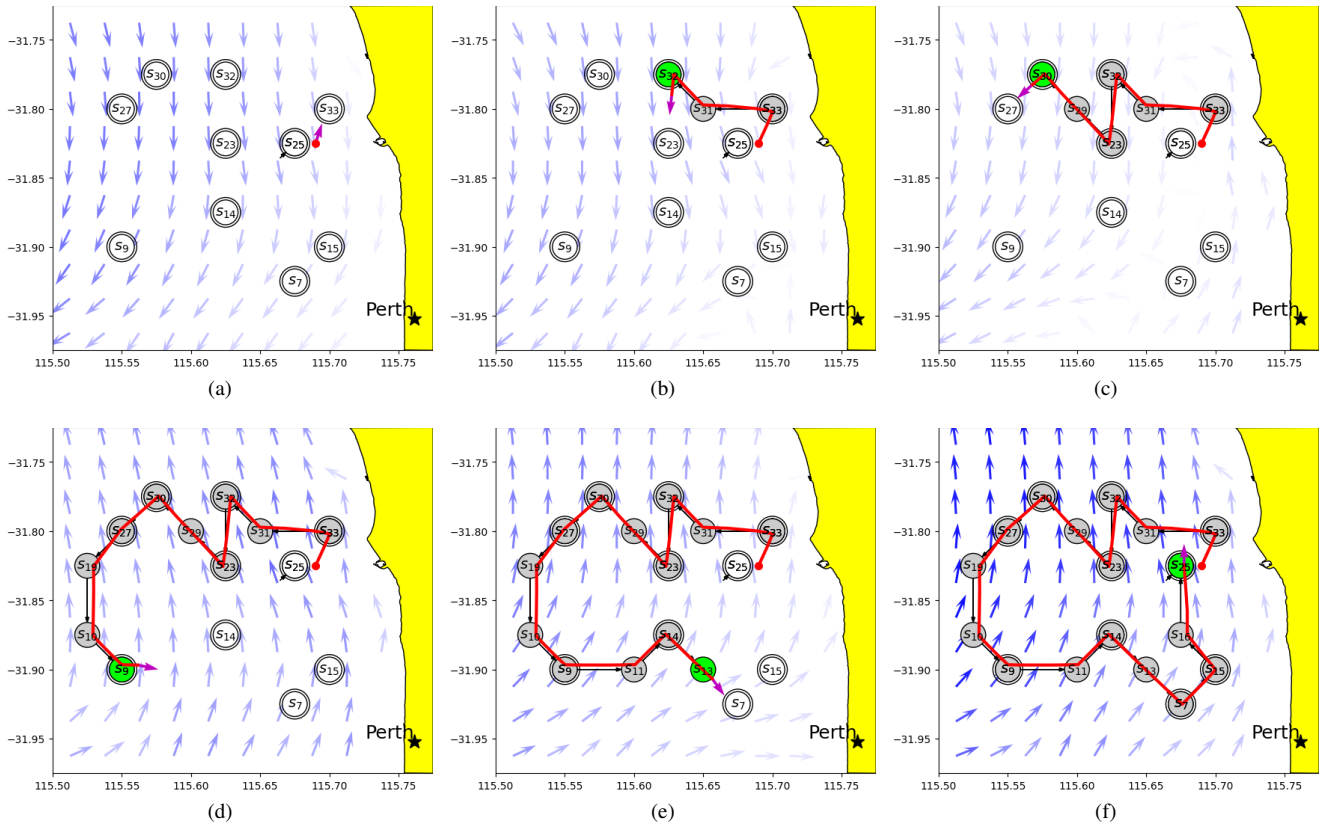


Fig. 4. TSP solution over the Perth coast ocean current forecast from 25 November 2022, showing goal states (double circle) the traversed graph states (grey), latest visited state (green), The optimal continuous path (red), and the corresponding headings (magenta arrow). The path initially moves against the current to reach its goals so that it can later ride the strong current near the end. Overall travel time is 222,900 seconds (approx. 2.58 days).

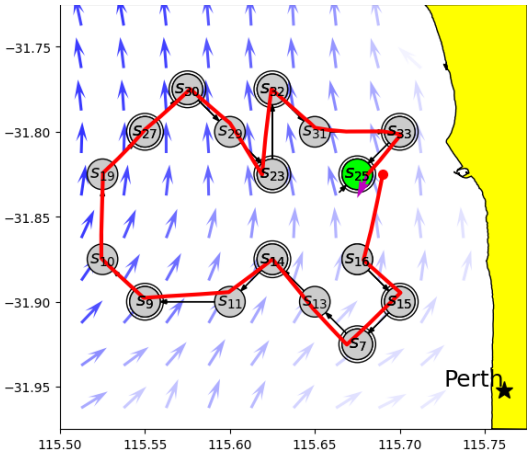


Fig. 5. Clockwise path of Fig. 4 where the vehicle ride the current at the start. The overall travel time is 234,000 seconds (approx. 2.71 days), which is more than three hours longer to complete.

longer to search exhaustively. The continuous vehicle path across the flow field is shown in red. The flow field is initially weak and flows south before becoming a stronger current and flowing north. Counter-intuitively, the vehicle initially heads against the current to reach its first goal before riding the strong northwards current to complete the path.

We compare this optimal solution with a manually defined path that follows the optimal path in reverse order, shown in Fig. 5. In this path, the vehicle takes advantage of the ocean current early and moves south towards its first goal states. However, as the current changes, it must traverse against a stronger oceanic current to reach the remaining goals. The resulting travel time is roughly 3 hours slower than the optimal path. This result highlights the non-FIFO property, where initially operating in slower areas results in the vehicle arriving at a stronger downstream position for a faster overall travel time. Such properties are difficult to define heuristically.

VII. Conclusion

We have presented a multi-query TDSP framework that lets us quickly construct travel time-optimal vehicle paths in a time-varying flow field. We have demonstrated its computational capability by showing it can solve the TSP problem 100 times faster and converge closer to the optimal solution earlier than a naive control-driven method. We have also shown an example of solving the TSP problem with real-world ocean forecast data.

With the capabilities of this multi-query approach established, future work includes applying this method to evaluate actions for many well-established multi-robot coordination algorithms, such as MCTS [24, 33] and Dec-MCTS [26, 34]. We also intend to investigate reducing the computation time for policy building by reusing policies generated from previous goal states. Finally, we intend to validate our new multi-query TDSP path planner by conducting multi-robot field trials.

REFERENCES

[1] L. M. Russell-Cargill, B. S. Craddock, R. B. Dinsdale, J. G. Doran, B. N. Hunt, and B. Hollings, "Using Autonomous Underwater Gliders for Geochemical Exploration Surveys," *APPEA J.*, vol. 58, pp. 367–380, 2018.

[2] D. L. Rudnick, R. E. Davis, C. C. Eriksen, D. M. Fratantoni, and M. J. Perry, "Underwater Gliders for Ocean Research," *Mar. Technol. Soc. J.*, vol. 38, no. 2, pp. 73–84, 2004.

[3] G. D'urso, J. J. H. Lee, O. Pizarro, C. Yoo, and R. Fitch, "Hierarchical MCTS for Scalable Multi-Vessel Multi-Float Systems," in *Proc. of IEEE ICRA*, 2021.

[4] K. Y. C. To, F. H. Kong, K. M. B. Lee, C. Yoo, S. Anstee, and R. Fitch, "Estimation of Spatially-correlated Ocean Currents from Ensemble Forecasts and Online Measurements," *Proc. of IEEE ICRA*, 2021.

[5] K. C. To, C. Yoo, S. Anstee, and R. Fitch, "Distance and steering heuristics for streamline-based flow field planning," in *Proc. of IEEE ICRA*. IEEE, 2020, pp. 1867–1873.

[6] J. J. H. Lee, C. Yoo, S. Anstee, and R. Fitch, "Hierarchical planning in time-dependent flow fields for marine robots," in *Proc. of IEEE ICRA*. IEEE, 2020, pp. 885–891.

[7] T. P. Zis, H. N. Psaraftis, and L. Ding, "Ship Weather Routing: A Taxonomy and Survey," *Ocean Engineering*, vol. 213, p. 107697, 2020.

[8] C. Yoo, J. J. H. Lee, S. Anstee, and R. Fitch, "Path Planning in Uncertain Ocean Currents Using Ensemble Forecasts," in *Proc. of IEEE ICRA*. IEEE, 2021, pp. 8323–8329.

[9] J. J. H. Lee, C. Yoo, S. Anstee, and R. Fitch, "Hierarchical Planning in Time-Dependent Flow Fields for Marine Robots," in *Proc. of IEEE ICRA*, 2020.

[10] ——, "Efficient Optiminal Planning in non-FIFO Time-Dependent Flow Fields," in *Proc. of IEEE ICRA*, 2023.

[11] S. E. Dreyfus, "An Appraisal of Some Shortest-Path Algorithms," *Oper. Res.*, vol. 17, no. 3, pp. 395–412, 1969.

[12] C. Yoo, R. Fitch, and S. Sukkarieh, "Probabilistic Temporal Logic for Motion Planning with Resource Threshold Constraints," in *Proc. of RSS*, 2012.

[13] E. Fernández-Perdomo, J. Cabrera-Gámez, D. Hernández-Sosa, J. Isern-González, A. C. Domínguez-Brito, A. Redondo, J. Coca, A. G. Ramos, E. A. Fanjul, and M. García, "Path Planning for Gliders Using Regional Ocean Models: Application of Pinzón Path Planner with the ESEOAT Model and the RU27 Trans-Atlantic Flight Data," in *Proc. of IEEE OCEANS*, 2010, pp. 1–10.

[14] E. Fernández-Perdomo, D. Hernández-Sosa, J. Isern-González, J. Cabrera-Gámez, A. C. Domínguez-Brito, and V. Prieto-Marañón, "Single and Multiple Glider Path Planning Using an Optimization-based Approach," in *Proc. of IEEE OCEANS*, 2011, pp. 1–10.

[15] J. Isern-González, D. Hernández-Sosa, E. Fernández-Perdomo, J. Cabrera-Gámez, A. C. Domínguez-Brito, and V. Prieto-Marañón, "Path Planning for Underwater Gliders Using Iterative Optimization," in *Proc. of IEEE ICRA*, 2011, pp. 1538–1543.

[16] D. Kularatne, S. Bhattacharya, and M. A. Hsieh, "Optimal Path Planning in Time-Varying Flows Using Adaptive Discretization," *IEEE RA-L*, vol. 3, no. 1, pp. 458–465, 2018.

[17] ——, "Optimal Path Planning in Time-Varying Flows with Forecasting Uncertainties," in *Proc. of IEEE ICRA*, 2018, pp. 1–8.

[18] M. Shu, X. Zheng, F. Li, K. Wang, and Q. Li, "Numerical Simulation of Time-Optimal Path Planning for Autonomous Underwater Vehicles Using a Markov Decision Process Method," *Appl. Sci.*, 2022.

[19] T. Lolla, M. P. Ueckermann, K. Yigit, P. J. Haley, and P. F. J. Lermusiaux, "Path planning in Time Dependent Flow Fields Using Level Set Methods," in *Proc. of IEEE ICRA*, 2012, pp. 166–173.

[20] T. Lolla, P. F. J. Lermusiaux, M. P. Ueckermann, and P. J. Haley, "Time-optimal Path Planning in Dynamic Flows Using Level Set Equations: Theory and Schemes," *Ocean Dynam.*, vol. 64, no. 10, pp. 1373–1397, 2014.

[21] L. Liu and G. S. Sukhatme, "A Solution to Time-Varying Markov Decision Processes," in *Proc. of IEEE ICRA*, 2018.

[22] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time Analysis of the Multiarmed Bandit Problem," *Mach. Learn.*, vol. 47, no. 2-3, pp. 235–256, 2002.

[23] G. Sánchez and J.-C. Latombe, "Using a PRM Planner to Compare Centralized and Decoupled planning for Multi-robot Systems," in *Proc. of IEEE ICRA*, 2002, pp. 2112–2119.

[24] A. Sadeghi and S. L. Smith, "Heterogeneous Task Allocation and Sequencing via Decentralized Large Neighborhood Search," *Unmanned Syst.*, vol. 5, no. 2, pp. 79–95, 2017.

[25] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The Complexity of Decentralized Control of Markov Decision Processes," *Math. Oper. Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[26] G. Best, O. M. Cliff, T. Patten, R. R. Mettu, and R. Fitch, "Dec-MCTS: Decentralized Planning for Multi-robot Active Perception," *Int. J. Robot. Res.*, vol. 38, no. 2-3, pp. 316–337, 2019.

[27] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[28] P. Toth and D. Vigo, "An Overview of Vehicle Routing Problems," in *The Vehicle Routing Problem*. SIAM, 2002, pp. 1–26.

[29] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon, and F. Soumis, *VRP with Pickup and Delivery*. SIAM, pp. 225–242.

[30] N. Leonard and J. Graver, "Model-Based Feedback Control of Autonomous Underwater Gliders," *IEEE J. Oceanic Eng.*, vol. 24, no. 4, pp. 633–645, 2001.

[31] J. J. H. Lee, C. Yoo, R. Hall, S. Anstee, and R. Fitch, "Energy-optimal Kinodynamic Planning for Underwater Gliders in Flow Fields," in *Proc. of ARAA ACRA*, 2017.

[32] B. C. Dean, "Shortest Paths in FIFO Time-dependent Networks: Theory and Algorithms," MIT, Tech. Rep., 2004.

[33] C. Yoo, S. Lensgraf, R. Fitch, L. M. Clemon, and R. Mettu, "Toward Optimal FDM Toolpath Planning with Monte Carlo Tree Search," in *Proc. of IEEE ICRA*. IEEE, 2020, pp. 4037–4043.

[34] K. M. B. Lee, F. Kong, R. Cannizzaro, J. L. Palmer, D. Johnson, C. Yoo, and R. Fitch, "An Upper Confidence Bound for Simultaneous Exploration and Exploitation in Heterogeneous Multi-robot Systems," in *Proc. of IEEE ICRA*. IEEE, 2021, pp. 8685–8691.