2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

A Generalized A* Algorithm for Finding Globally Optimal Paths in Weighted Colored Graphs

Jaein Lim[1] and Panagiotis Tsiotras[2]

Abstract— **Both geometric and semantic information of the search space are imperative for a good plan. We encode those properties in a weighted colored graph (geometric information in terms of edge weight and semantic information in terms of edge and vertex color) and propose a generalized A* to find the shortest path among the set of paths with minimal inclusion of low-ranked color edges. We prove the completeness and optimality of this Class-Ordered A* (COA*) algorithm with respect to the hereto defined notion of optimality. The utility of COA* is numerically validated in a ternary graph with feasible, infeasible, and unknown vertices and edges for the cases of a 2D mobile robot, a 3D robotic arm, and a 5D robotic arm with limited sensing capabilities. We compare the results of COA* to that of the regular A* algorithm, the latter of which finds a shortest path regardless of the semantic information, and we show that the COA* dominates the A* solution in terms of finding less uncertain paths.**

I. Introduction

Path planning in partially known or unstructured environments is a relatively under-studied problem [1], compared to the vast effort dedicated during the past decades to efficiently solve for optimal paths in fully known search spaces [2]–[11]. The notion of optimality is clear and consistent when the search space is known. For example, the space can be approximated with a weighted graph, with each edge assigned a cost; then the optimal path is simply the path with minimum cost (e.g., the shortest path) in this graph [12]–[19]. In many applications however, the environment is rarely fully known to the planning agent because of either sensor limitations and/or memory constraints. What is a suitable notion of optimality in such a search space, so that the autonomous agent can make reasonable and consistent decisions?

A common framework to cope with partially observable or dynamic environments is re-planning [12]–[15]. In this framework, obstacles are a priori unknown and are only revealed during the course of the plan execution; a plan is first found on an optimistically perceived environment (i.e., assuming no obstacles) and then repaired whenever it is found to be infeasible. The benefit of these methods is their algorithmic efficiency to optimally repair the graph based on the newly perceived part of the environment. A fundamental question remains unanswered, however: "what should an optimal policy strive for in an unknown space?" [1].

[1]Jaein Lim is a graduate student at the School of Aerospace Engineering, Georgia Institute of Technology, Atlanta. GA, USA. Email: jaeinlim126@gatech.edu

[2] Panagiotis Tsiotras is a Professor and David and Andrew Lewis Chair at the School of Aerospace Engineering and Associate Director at the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta. GA, USA. Email: tsiotras@gatech.edu

In the works of [16]–[19], when the search space is only partially known, the "risk" of traversing unknown edges is combined to the original length of the edge to construct a new cost measure. The optimal path is then defined as the cost minimizing path with respect to this new measure. These methods treat planning in partially known search spaces as a shortest path planning problem with respect to a user-defined measure, where uncertain paths are deferred. Yet, traversing through some high-uncertainty edges cannot be explicitly avoided, if such a path is short enough in terms of the user-defined measure. Moreover, the user-defined measure may not be consistent, in general, for re-planning purposes, e.g., when new information changes the perceived environment, reusing the previous search results becomes difficult.

To illustrate the issues with potential inconsistencies, consider two agents with identical planning strategies acting in the same environment. Assume that one agent has more accurate information about the environment than the other agent. If we were to choose one of the two solution paths provided by either agent, then a reasonable choice would be to choose the path computed by the agent having more information, regardless of the path length. This is because a reasonable measure on the quality of the path should be that it is monotonic with the information of the underlying planner. However, monotonicity cannot be imposed, in general, in a soft-constrained framework. One can easily construct counter-examples where the less knowledgeable agent produces a "shorter" path than the more knowledgeable agent, by modifying how much the uncertainty is penalized.

To this end, we pay close attention to a more explicit approach presented in [21], [22], which defines a notion of optimality in the semantic search space. In that approach, the vertices and edges of the graph are classified into equivalent classes (or "colors") so as to encode semantic information of the perceived environment. Afterwards, a constrained shortest path planning problem is solved in this weighted colored graph. The optimal path is not necessarily the shortest path, but it is constrained by the constituent colors. In [21] the planning problem in an unknown environment is cast as a constrained shortest path planning problem in a weighted bi-colored graph, where the vertices are partitioned into white (known) or gray (unknown) vertices to approximate partially known environments by excluding known, but infeasible vertices. The authors then find the shortest path in the graph such that the number of gray vertices does not exceed a certain number, or they find the path that has the least number of gray vertices and whose length does not exceed a certain threshold.

978-1-7281-9077-8/21/$31.00 ©2021 IEEE

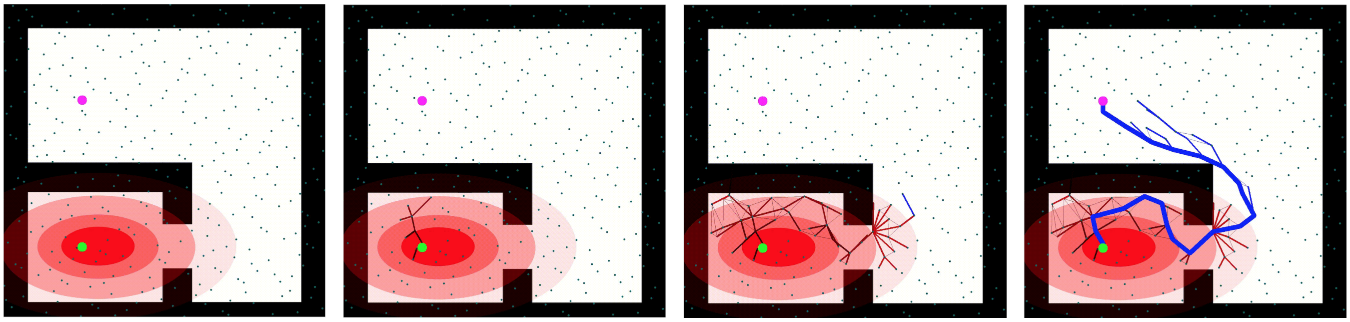


Fig. 1: COA* search propagation from left to right on the weighted colored graph with Halton sequence [20] vertices where the red region is known to be hazardous. COA* grows the search tree lazily and incrementally to find the shortest path among the set of paths which have the shortest edges in the redder region. Bold edges are the current search tree, and light edges are the evaluated edges.

A general weighted colored graph is used in [22] to relax the binary representation of the environment to encode various ranked semantic information of the search space. A weight function devised in [22] assigns a positive real number to each edge incorporating the original edge weight and color, such that a total ordering is placed over paths with mixed colors. This total ordering favors paths with a fewer number of edges of inferior classes, and favors shorter paths if the constituting colors are the same. With the modified weight, a standard graph search (e.g., Dijkstra [2]) ensures that the optimal path respects the total order. That is, the resulting path is guaranteed to be the shortest path among the set of paths which include the minimum number of inferior class edges. However, global properties of the entire graph, such as the cardinality of the edge set of each color, are required to compute the modified weight, which may be intractable to compute online.

In this work, we extend regular edge-relaxation algorithms (for instance, A* [3]) to find an optimal path in a weighted colored graph based on the notion of optimality defined in [22]. Informally, an optimal path is the shortest path in the set of paths with minimal inclusion of inferior-class edges. The original weight or the length of the path is used only as a tie-breaker among paths within the same class. For example, a shorter path with bad edges will be considered worse than a longer path with better color edges. This total ordering over the set of paths will be formally defined in the next section. The aim of our work is to find an optimal path without resorting to the global properties of the graph by propagating forward the cost-to-come and the path-class information to build an optimal search tree. The key insight is to use an abstract priority queue to order the expansion of optimal path candidates. We call this algorithm Class-Ordered A* (COA*).

One immediate application of planning on a colored graph is that we can plan for an information-conservative path in a partially known environment. Suppose we partition the search environment into three classes: feasible, infeasible, unknown. Then, we can find the shortest path that maximizes feasibility (i.e., minimize infeasible and unknown vertices). For example, if there exist two paths, an unknown short path and a feasible longer path, then the feasible path is ranked higher over the short, unknown path. Consequently, the agent traverses through the region that is already known. This may be useful for planning problems such as a surgical robot, a marine probe, or a rescue robot where the impact on the surroundings should be minimized.

Another application where planning on a colored graph is beneficial is the case of a cooperative multi-agent planning problem, where agents having only local information communicate with each other by exchanging their local plans in order to to achieve a consensus on a globally acceptable optimal solution [23]–[25]. We can encode information passed to other agents with a "color" depending on the information quality (e.g., agent credibility, noise-level, etc.,) such that we can suppress the communication of inferior information below a certain class. The colored graph planning provides another layer of ordering of paths (besides a single length+risk measure, for instance), thus being capable of rejecting shorter paths of inferior classes.

The contributions of this paper can be summarized as follows. First, we develop a complete and optimal algorithm to solve for the shortest path among the set of paths with minimal inclusion of low-class edges in a weighted colored graph. Second, we prove the completeness and optimality of the proposed algorithm. The proposed algorithm efficiently builds an optimal search tree by heuristically and lazily expanding only the optimal path candidates. Third, we validate the utility of planning on a weighted colored graph using numerical experiments.

II. Problem Formulation

Before discussing the problem to be solved in this paper, let us define some background notation that will be frequently used in the rest of the paper.

A. *Weighted Colored Graph*

Let $G = (V, E)$ be a graph with vertices V and edges E. Let $\phi_V : V \to \mathcal{L}$ be a perception vertex function that classifies each vertex $v \in V$ to a class $\ell \in \mathcal{L} = \{1, ..., L\}$, such that ϕ_V partitions the set V, that is, $V = \bigcup_{\ell \in \mathcal{L}} V_\ell$ where $V_\ell = \{v \in V : \phi_V(v) = \ell\}$ and $V_i \cap V_j = \varnothing$ for $i \neq j$. Similarly, define $\phi_E : E \to \mathcal{K}$ to be a perception edge function that classifies each edge $e \in E$ to a class in $\mathcal{K} = \{1, ..., K\}$, such that $E = \bigcup_{k \in \mathcal{K}} E_k$ where each $E_k = \{e \in E : \phi_E(e) = k\}$ and $E_i \cap E_j = \varnothing$ for $i \neq j$. We assume that the smaller the class number, the better the class. For example, an edge e with $\phi_E(e) = 1$ belongs to the best class and $\phi_E(e) = K$ belongs to the worst class. We will assume that the edge class set is not smaller than the vertex class set, i.e., $K \geq L$, and that, for each edge, $e = (u, v)$, it holds $\phi_E(e) \geq \max\{\phi_V(u), \phi_V(v)\}$. This assumption allows us to quickly underestimate the edge class by classifying the end vertices first. We only use the vertex class to compute the heuristic edge class before the actual, and perhaps computationally expensive, edge classification takes place. Also, for each edge $e \in E$, a weight function $w : E \to \mathbb{R}_+$ assigns a non-negative real number, e.g., the distance to traverse this edge. We will assume that the functions ϕ_E and w are independent and given, but are expensive to compute. Hence, we will approximate them with functions $\widehat{\phi}_E$, and $\widehat{w}$, respectively, that underestimate their true values, i.e., $\widehat{\phi}_E \leq \phi_E$ and $\widehat{w} \leq w$. We will use these admissible heuristic estimators to prioritize the expansion of optimal path candidates in an attempt to delay the actual computation until it becomes necessary. We say that an edge e is evaluated when both the values $\phi_E(e)$ and $w(e)$ are computed.

B. Optimal Path

Define a path $\pi = (v_1, v_2, \ldots, v_m)$ on the graph $G = (V, E)$ as an ordered set of distinct vertices $v_i \in V$, $i = 1, \ldots, m$ such that for any two consecutive vertices v_i, v_{i+1}, there exists an edge $e = (v_i, v_{i+1}) \in E$. Throughout this paper, we will interchangeably denote a path as the set of such edges. Let $v_\mathrm{s}, v_\mathrm{g} \in V$ be the start and goal vertices, respectively. Denote by $\Pi(v_\mathrm{s}, v)$, or $\Pi(v)$ for short if there is no danger for ambiguity, the set of all acyclic paths connecting v_s and some vertex v in G. Let $\Pi_k(v)$ be the set of paths in which the worst (greatest) edge class included in each path in $\Pi_k(v)$ is k. That is,

$$\Pi_k(v) = \{\pi \in \Pi(v) : w(\pi, k) > 0 \text{ and } w(\pi, \ell) = 0, \forall \ell > k\},$$

where $w(\pi, k) = \sum_{\{e \in \pi : \phi_E(e) = k\}} w(e)$ is the sum of edge weights that are of class k in path π with some abuse of notation. We will call the elements of $\Pi_k(v)$ class-k paths. Furthermore, define

$$\Pi_k^\alpha(v) = \{\pi \in \Pi_k(v) : w(\pi, k) = \alpha\},$$

to be the set of class-k paths which have exactly α-long edges of class k. We shall impose a total order on the set of paths with $\Pi_k^\alpha(v) \prec \Pi_\ell^\beta(v)$ for any α, β whenever $k < \ell$ and $\Pi_k^\alpha(v) \prec \Pi_k^\beta(v)$ for all $\alpha < \beta$. Hence, we define the optimal path set $\Pi^*(v)$ as the nonempty set of paths having best worst-class edge with the shortest worst-class edges, that is,

$$\Pi^*(v) = \min_{\alpha \in \mathbb{R}} \min_{k \in \mathcal{K}} \{\Pi_k^\alpha(v)\},$$

where the minimum is defined with respect to the total path order. We wish to solve for the optimal path π^* which is the shortest path in Π^*, that is,

$$\pi^*(v_\mathrm{g}) = \operatorname*{argmin}_{\pi \in \Pi^*(v_\mathrm{g})} \sum_{e \in \pi} w(e).$$

C. Search Tree

In typical search algorithms, each node stores a real number for cost-to-come value to represent the accumulated weights along the path. However, in our setting, we store the cost value for a path as a vector in $\mathbb{R}^K$, and we denote this vector value by $\theta(\pi) = (w(\pi, k))_{k \in \mathcal{K}} \in \mathbb{R}^K$, where the sum of edge weights that are of class k in the path π is stored in the k-th index. Note that the total order defined over two paths π_1 and π_2 from the same start vertex to the same end vertex induces a total order on the vector values, that is, $\theta(\pi_1) \prec \theta(\pi_2)$ if $\pi_1 \prec \pi_2$. The other direction is also true, when a pair of cost-to-come values for the paths from the same start to the same end vertices are compared. That is, $\pi_1 \prec \pi_2$ if $\theta(\pi_1) \prec \theta(\pi_2)$ for any π_1, π_2 with the same start and end vertices. We also define an admissible estimate of $\theta(\pi)$ with $\widehat{\theta}(\pi)$, such that $\widehat{\theta} \preceq \theta$. A trivial heuristic estimate of $\widehat{\theta}$ is the zero sequence.

Initially, all edges of G are implicitly defined and are not evaluated. As the search propagates, an explicit search tree T rooted at v_s is built by adding evaluated edges. Each node in the search tree corresponds to a path from v_s to a given vertex v. It stores the cost-to-come value accmulated along the current best path in the tree, denoted by $g(v)$, and a back-pointer to its parent vertex, denoted by $bp(v)$. Note that $g(v) = \theta(\pi(v))$, where $\pi(v)$ is the path from v_s to v found so far in the current search tree. All vertices not in the search tree have an infinite cost-to-come and an empty back-pointer.

Note that the standard A* algorithm stores for each vertex in a search tree T a back-pointer to its parent vertex and also stores the cost-to-come accumulated by traversing the back-pointed path. Despite the fact that the cost-to-come of a vertex in T can be retrieved by following the back-pointed path in the forward direction, the numerical value of the cost-to-come is explicitly stored at each vertex to expedite the ordering of expanding leaf vertices so that the completeness and optimality of the algorithm with respect to the cost-to-come are preserved. The same idea can be generalized to a total ordering of paths, without losing correctness of the algorithm. The completeness and optimality of such generalization with respect to any ordered set of paths can be also proven, as it will be shown formally in Section IV.

D. Priority Queue

We will use an edge queue Q to prioritize the expansion (i.e., evaluation) of edges based on an admissible estimate of the total cost constrained to an edge, breaking ties using an admissible cost-to-come constrained to this edge. The key $k(e)$ of an edge $e = (u, v)$ contains two components: $k(e) = [k_1(e); k_2(e)]$, where $k_1(e) = g(u) + \widehat{\theta}(u, v) + \widehat{\theta}(v, v_\mathrm{g})$ and $k_2(e) = g(u) + \widehat{\theta}(u, v)$, in which the addition is a vector addition in $\mathbb{R}^K$. Note that $\widehat{\theta} \preceq \theta$ is an admissible estimate of path cost, and we can compute $\widehat{\theta}(u, v)$ using an admissible heuristic weight $\widehat{w}$ and an admissible heuristic class $\widehat{\phi}_E$ of an edge $e = (u, v)$. Likewise, we can also compute $\widehat{\theta}(v, v_\mathrm{g})$, the cost of an admissible heuristic path from v to the goal, using an admissible heuristic weight (e.g., distance to the goal) and an admissible heuristic class. Edges are sorted by their key values in lexicographical order, that is $k(e) \prec k'(e)$ if and only if either $k_1(e) \prec k'_1(e)$ or ($k_1(e) = k'_1(e)$ and $k_2(e) \prec k'_2(e)$). These heuristic estimates of the path cost delay the actual evaluation of the edge, as the priority queue prioritizes edges based on their heuristic estimates.

III. Algorithm

We begin the search by initiating a search tree T with the start vertex as the root and by putting the promising outgoing edges in the priority queue Q, where a promising edge is an edge such that the path utilizing this edge may improve the current search tree T. That is, only edges that could potentially improve the current solution with heuristic estimates are first chosen, and then among the chosen edges, only those that can potentially improve the current path to the child vertex of the edge with the heuristic estimates are selected to be inserted in the queue. The edge with the highest index is removed from Q for evaluation, and then the edge is added to the tree T if such an evaluation reveals that the edge improves the child vertex's g-value. The outgoing edges of the newly added leaf vertex of T are then inserted in the queue accordingly. The iteration continues until the goal vertex is reached. The resulting search tree consists of connected classified edges rooted at v_{s}, and it contains an optimal path from v_{s} to v_{g}. The solution path is traced back from v_{g} to its back-pointer up to v_{s}.

Figure 1 shows the search instances of COA* on a weighted colored graph, where the vertices are generated using a Halton sequence [20] and the edges are defined for two vertices if they are within a certain radius. The environment has some known hazardous region colored with red ellipses, and the edge set is partitioned into 6 classes according to their hazardous levels. COA* lazily and incrementally grows the search tree until the shortest path to the goal with the shortest edges in the redder region is found.

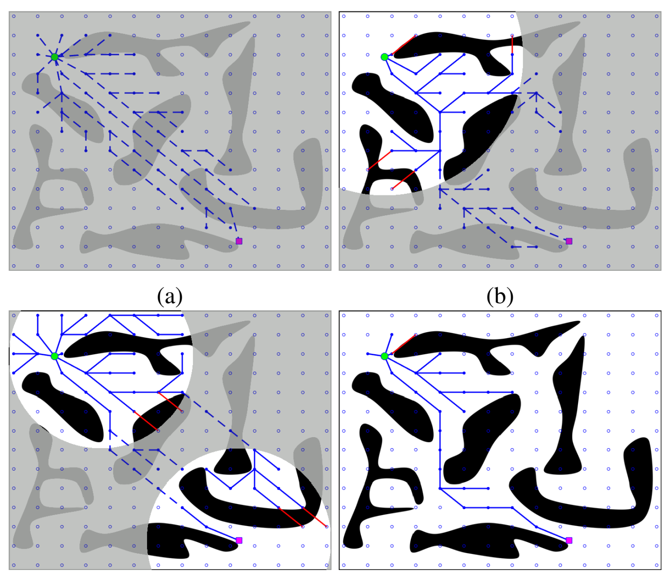
IV. Analysis

We prove the completeness and optimality of COA* by extending the results of [3]. The key observation here is that the original A* algorithm uses an ordinary ordering of real numbers for expanding paths to find a path minimizing the cost-to-come, but the vertex ordering of the algorithm can be generalized to a total ordering of paths. In our analysis, we will assume that a total path ordering exists such that for any pair of paths having the same start and goal vertices, the order gives a binary relation between the two. Then we will prove that the resulting path of such an algorithm is optimal with respect to the defined ordering.

We will present an edge queue version of the proof instead of the vertex queue proof of the original A* to utilize admissible estimators of edge evaluation functions. This choice relaxes the heuristic look-ahead over to an edge, such that the actual evaluation of the edge is delayed until it is necessary, under the assumption that edge evaluations are computationally expensive [8], [11], [26]. However, since the goal is to find a path from a start vertex to a goal vertex, we need to first define the trivial edges involving the start and

Algorithm 1 Class Ordered A*$(v_{\mathrm{s}}, v_{\mathrm{g}}, G)$

procedure EnqueueOutgoingEdges(v)
  for all $w \in succ(v)$ **do**
    if $g(v) + \widehat{\theta}(v,w) + \widehat{\theta}(w, v_{\mathrm{g}}) \preceq g(v_{\mathrm{g}})$ **then**
      if $g(v) + \widehat{\theta}(v,w) \prec g(w)$ **then**
        $Q \leftarrow Q \cup (v,w)$;
procedure RewireTree(u, v)
  $bp(v) = u$;
  $g(v) = g(u) + \theta(u,v)$;
  $T \leftarrow T \cup (u,v)$;
procedure Main()
  $T \leftarrow \varnothing$; $Q \leftarrow \varnothing$;
  $bp(v_{\mathrm{s}}) = \varnothing$;
  $g(v_{\mathrm{s}}) = 0$;
  EnqueueOutgoingEdges(v_{s})
  while true **do**
    $(u,v) \leftarrow$ popFrontEdge(Q)
    if $u = v_{\mathrm{g}}$ **then**
      break;
    if $g(u) + \theta(u,v) \prec g(v)$ **then**
      RewireTree$((u,v))$;
      EnqueueOutgoingEdges(v);



(a) (b) (c) (d)

Fig. 2: Search trees built from top left start toward bottom right goal on different perceived environments, where gray region is unknown: (a) completely unknown environment, (b) partially known around start, (c) partially known around start and goal, and (d) fully known environment. Solid blue line, dashed blue line, and solid red line indicate three different edge classes: feasible, unknown, infeasible, respectively, classified along the expansion of COA*. The algorithm always finds the shortest path with minimal inclusion of infeasible, then unknown, and then feasible edges

goal vertices. To this end, we define a start edge $s = (v_s, v_s)$ for a start vertex v_s, and a goal edge $t = (v_g, v_g)$ for a goal vertex v_g, where the evaluations of these trivial edges are naturally defined as: $w(s) = 0$, $w(t) = 0$, $\phi_E(s) = \phi_V(v_s)$ and $\phi_E(t) = \phi_V(v_g)$.

With some abuse of notation, we will denote the optimal path $(e, \ldots, e')$ from a start edge e to an end edge e' in the graph with $\pi(e, e')$. Also, we will denote the path from an edge e to e' found by the COA* algorithm so far in the current search tree with $\pi_T(e, e')$. We will assume that there exists an ordering for any pair of paths from the same start and end edges. For example, we have $\pi(e, e') \preceq \pi_T(e, e')$ since any search tree built by the algorithm cannot contain a strictly better path than the optimal path. Also, we denote an underestimating heuristic path from e to e' with $\widehat{\pi}(e, e')$, such that $\widehat{\pi}(e, e') \preceq \pi(e, e')$ for any edges e, e'. Concatenation of paths is denoted with $\pi(e, e') \cup \pi(e', e'') = \pi(e, e'')$ for two adjacent paths. If e' and e'' are adjacent edges on the optimal path, then, $\pi(e, e') \cup e'' = \pi(e, e'')$. We will use $\pi(e, e) = \varnothing$ for any $e \in E$, and $\varnothing \preceq \pi(e, e')$ for any edges e, e'.

We are now ready to present Lemma 1, which states that the path ordering is invariant under the concatenation with the same subpath.

Lemma 1: Let $\widehat{\pi}(e, t) \preceq \pi(e, t)$. Then $\pi(s, e) \cup \widehat{\pi}(e, t) \preceq \pi(s, e) \cup \pi(e, t)$, for any e.

Proof: Fix an edge $e = (u, v)$, and assume that $\widehat{\pi}(e, t) \preceq \pi(e, t)$. Then, $\widehat{\theta}(u, v) + \widehat{\theta}(v, t) \preceq \theta(u, v) + \theta(v, t)$. Hence, $\theta(s, u) + \widehat{\theta}(u, v) + \widehat{\theta}(v, t) \prec \theta(s, u) + \theta(u, v) + \theta(v, t)$, since the ordering is invariant under the addition. This implies that $\pi(s, e) \cup \widehat{\pi}(e, t) \preceq \pi(s, e) \cup \pi(e, t)$, thus completing the proof. ∎

Next, Lemma 2 states that if an edge on an optimal path is not evaluated, then the edge queue Q must contain an edge such that an optimal subpath from the start edge to this edge is in the search tree T.

Lemma 2: For any unevaluated edge e, and for any optimal path P from s to e, there exists an edge $e' \in Q$ on P such that $\pi(s, e') = \pi_T(s, e')$.

Proof: Let $P = (s = e_0, e_1, \ldots, e_n = e)$. Let $s \in Q$, that is, COA* has not completed the first iteration. Let $e' = s$, then $\pi(s, s) = \pi_T(s, s) = \varnothing$, and hence the lemma is trivially true. Now, let s be evaluated. Let Δ be the set of all evaluated edges e_i in P such that $\pi(s, e_i) = \pi_T(s, e_i)$. Then, Δ is not empty, since by assumption $s \in \Delta$. Let e^* be the element of Δ with the highest index. Clearly, $e^* \neq e$, since e has not yet been evaluated. Let e' be the successor of e^* on P. Then, $\pi_T(s, e') \preceq \pi_T(s, e^*) \cup e'$. However, $\pi_T(s, e^*) = \pi(s, e^*)$ since $e^* \in \Delta$. Moreover, $\pi(s, e') = \pi(s, e^*) \cup e'$, since e^* and e' are on π^*. Therefore $\pi_T(s, e') \preceq \pi(s, e')$. In general, $\pi_T(s, e') \succeq \pi(s, e')$, and hence $\pi_T(s, e') = \pi(s, e')$. Also, e' must be inserted in Q since e' is adjacent to e^*, the highest index edge in Δ. ∎

Corollary 1: Suppose $\widehat{\pi}(e, t) \preceq \pi(e, t)$ for all e, and suppose COA* has not terminated. Then, for any optimal path P from s to a goal t, there exists an edge $e' \in Q$ on P with $\pi_T(s, e') \cup \widehat{\pi}(e', t) \preceq \pi(s, t)$.

Proof: By the Lemma 2, there exists an edge $e' \in Q$ in P with $\pi(s, e') = \pi_T(s, e')$. Then, by applying the Lemma 1,

$$\begin{aligned} \pi_T(s, e') \cup \widehat{\pi}(e', t) &= \pi(s, e') \cup \widehat{\pi}(e', t) \\ &\preceq \pi(s, e') \cup \pi(e', t) \\ &= \pi(s, t), \end{aligned}$$

where the last equality holds since $e' \in P$. ∎

Now we are ready to prove the completeness and optimality properties of COA*.

Theorem 1: Suppose $\widehat{\pi}(e, t) \preceq \pi(e, t)$ for all e. Then COA* terminates in a finite number of iterations and finds the optimal path from s to a goal t, if one exists.

Proof: We prove this theorem by contradiction. Suppose the algorithm does not terminate by finding an optimal solution. There are three cases to consider:

1) *The algorithm terminates at a non-goal.* This contradicts the termination condition (Line 17, Algorithm 1).
2) *The algorithm fails to terminate.* Let $\pi(s, t)$ be an optimal path from s to the goal t. Clearly, no edges with $\pi_T(s, e) \cup \widehat{\pi}(e, t) \succ \pi(s, t)$ will ever be evaluated, since by Corollary 1 there is some e' with $\pi_T(s, e') \cup \widehat{\pi}(e', t) \preceq \pi(s, t)$, and hence, COA* will select e' instead of e. Since there is only a finite number of acyclic paths from the start edge to any edge, all edges with $\pi(s, e) \cup \widehat{\pi}(e, t) \preceq \pi(s, t)$ can be re-evaluated at most a finite number of times. Hence, the only possibility left for COA* that fails to terminate is when the queue Q becomes empty before a goal is reached. Suppose, ad absurdum, that Q becomes empty before the goal is reached. If there exists a path to the goal, and the goal is not already in the tree T, then there exists at least one edge that could improve the current solution path. Then, this edge would have been inserted in the queue Q by Algorithm 1, contradicting the assumption that Q becomes empty. Hence, COA* must terminate.
3) *The algorithm terminates at the goal without finding an optimal solution.* Suppose COA* terminates at some goal edge t with $\pi_T(s, t) \succ \pi(s, t)$. By Corollary 1, just before termination, there must exist an edge such that $e' \in Q$ with $\pi_T(s, e') \cup \widehat{\pi}(e', t) \preceq \pi(s, t)$. Hence, at this stage, e' would have been selected for evaluation rather than t, contradicting the assumption that COA* terminated.

Therefore, the algorithm must terminate by finding the optimal solution, if one exists. ∎

Note that we proved the completeness and optimality with respect to the total order defined in Section II. However, the same technique can be applied to a different total order using more general properties of paths if the properties are additive along the path and the start vertex is already on the optimal path.

V. Numerical Results

In this section, we present numerical results of COA* in scenarios where the planning agent has limited perception of the search space and incrementally gains knowledge of the space as it moves along the local plan. We then compare the results of COA* to a regular A* which finds the shortest path regardless of uncertainty on an optimistically perceived

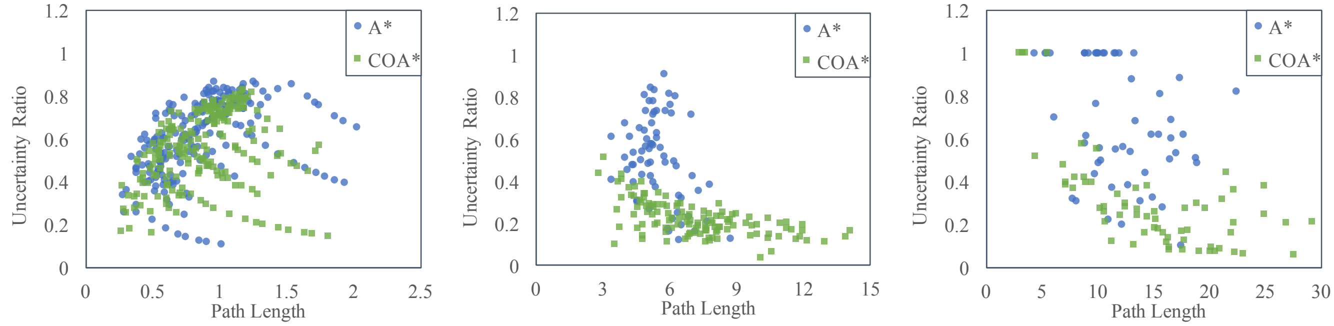


Fig. 3: Collection of path length and uncertainty ratio for each local path recorded during the perception-plan-action loop, excluding fully known paths: from left to right: 2D mobile robot, 3D robotic arm, 5D robotic arm.

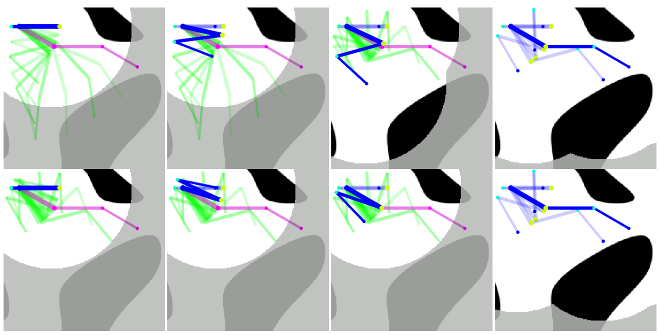


Fig. 4: Propagation of 3D arm with A* (top row) and COA* (bottom row) from left to right, where the solid blue arm is the current vertex, the green arms are the vertices on the locally optimal path, the magenta arm is the goal vertex, and the transparent blue arms show the trace of the global solution.

environment, namely, assuming the unknown region is free of obstacles. The perception-plan-action loop is iterated until the agent reaches the goal on a graph with uniformly distributed vertices. At each planning instance, the edges of the graph are classified into three classes: feasible, unknown, and infeasible. The experiments include a 2D mobile robot, a 3D robotic arm, and a 5D robotic arm, where a sensor with a radial field of view is attached to the mobile robot and at the tip of the robotic arms, respectively. All the configurations outside the sensing region are classified as unknown, while each configuration within the sensing region can be classified either as feasible or infeasible [27], [28]. Once the agent finds a locally optimal solution in the current map, the agent executes the first action of the plan, i.e., it moves along the first vertex of the locally optimal path, and then updates its map accordingly.

Figure 4 shows the search instances of A* (top) and COA* (bottom) algorithms for the 3D robotic arm during this perception-plan-action loop. COA* always finds paths that minimize the uncertain edge lengths and then the feasible edge lengths, whereas A* finds the shortest path regardless of the underlying semantic information. Since the shortest paths computed by A* often turn out to be infeasible, following the paths computed by COA* results in a shorter distance travelled to reach the goal configuration.

During the perception-plan-action loop, two measures of interest for each locally optimal plan were recorded at each planning instance, namely, the path length (distance) and the uncertainty ratio. The local uncertainty ratio is the proportion of the sum of the lengths of the uncertain edges to the sum of the lengths of all the edges in the path, which measures the percentage of the uncertain segment for the path. Using sensors with different ranges, about 100 paths were collected for each robot in several work spaces. Figure 3 shows the results of A* in comparison to COA* for a 2D mobile robot, a 3D robotic arm, and a 5D robotic arm. On average, COA* finds less uncertain paths compared to A* . This becomes more evident in the higher dimensional search spaces.

VI. Conclusion

We have introduced the notion of optimality on a weighted colored graph, which encodes both geometric and semantic information of the search space. We present a new search algorithm, the Class-Ordered A* (COA*) to find a globally optimal path in a weighted colored graph by incrementally and lazily building an optimal search tree using a heuristic, and we proved the completeness and optimality of the algorithm. The optimal path of COA* is the shortest path among the set of paths with minimal inferior class edges. In addition, COA* monotonically finds a better path when the underlying graph has a strictly better class of vertices and edges. Finally, COA* was numerically evaluated on a ternary graph with feasible, unknown, and infeasible classes against the standard A* algorithm, which finds a shortest path regardless of uncertainty. The results of these numerical experiments confirmed the superiority of COA* in terms of finding safer plans when the search space is partially known.

Acknowledgement

This work has been supported by ARL under DCIST CRA W911NF-17-2-0181 and NSF under award IIS-2008686.

References

[1] L. Janson, T. Hu, and M. Pavone, "Safe motion planning in unknown environments: Optimality benchmarks and tractable policies," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, PA, June 26–30 2018.

[2] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[4] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *IEEE International Conference on Robotics and Automation*, vol. 1, San Francisco, CA, April 24–28 2000, pp. 521–528.

[5] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[6] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *IEEE International Conference on Robotics and Automation*, Karlsrühe, Germany, May 6–10 2013, pp. 2421–2428.

[7] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, 2015.

[8] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 26–30 2015, pp. 3067–3074.

[9] C. M. Dellin and S. S. Srinivasa, "A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors," in *Proceedings of the International Conference on Automated Planning and Scheduling*, no. 9, London, UK, 2016, pp. 459–467.

[10] A. Mandalika, S. Choudhury, O. Salzman, and S. Srinivasa, "Generalized lazy search for robot motion planning: Interleaving search and edge evaluation via event-based toggles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 29, no. 1, Berkeley, CA, 2019, pp. 745–753.

[11] M. P. Strub and J. D. Gammell, "Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques," in *IEEE International Conference on Robotics and Automation*, Paris, France, May 31–Aug 31 2020, pp. 130–136.

[12] S. Koenig and M. Likhachev, "D* lite," in *Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, July 28–Aug 1 2002, p. 476–483.

[13] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, no. 1, pp. 93 – 146, 2004.

[14] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun, "Anytime search in dynamic graphs," *Artificial Intelligence*, vol. 172, no. 14, pp. 1613 – 1643, 2008.

[24] R. Nissim and R. Brafman, "Distributed heuristic forward search

[15] S. Aine and M. Likhachev, "Truncated incremental search," *Artificial Intelligence*, vol. 234, pp. 49 – 77, 2016.

[16] S. Kambhampati and L. S. Davis., "Multiresolution path planning for mobile robots," *IEEE Journal of Robotics and Automation*, pp. 135–145, 1986.

[17] D. Jung, "Hierarchical path planning and control of a small fixed-wing uav: theory and experimental validation." Ph.D. dissertation, Georgia Institute of Technology, Atlanta, GA, 2007.

[18] R. V. Cowlagi and P. Tsiotras, "Multiresolution motion planning for autonomous agents via wavelet-based cell decompositions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 42, no. 5, pp. 1455–1469, Oct 2012.

[19] F. Hauer, A. Kundu, J. M. Rehg, and P. Tsiotras, "Multi-scale perception and path planning on probabilistic obstacle maps," in *IEEE International Conference on Robotics and Automation*, Seattle, WA, May 26–30 2015, pp. 4210–4215.

[20] J. H. Halton, "Algorithm 247: Radical-inverse quasi-random point sequence," *Communications of the ACM*, vol. 7, no. 12, pp. 701–702, December 1964.

[21] A. AliAbdi, A. Mohades, and M. Davoodi, "Constrained shortest path problems in bi-colored graphs: a label-setting approach," *GeoInformatica*, pp. 1 – 19, 2019.

[22] D. Wooden and M. Egerstedt, "On finding globally optimal paths through weighted colored graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, CA, December 13–15 2006, pp. 1948–1953.

[23] A. Torreño, E. Onaindia, and O. Sapena, "FMAP: Distributed cooperative multi-agent planning," *Applied Intelligence*, vol. 41, no. 2, pp. 606–626, 2014.
for multi-agent planning," *Journal of Artificial Intelligence Research*, vol. 51, no. 1, pp. 293–332, 2014.

[25] J. Lim and P. Tsiotras, "MAMS-A*: Multi-agent multi-scale A*," in *IEEE International Conference on Robotics and Automation*, Paris, France, May 31–Aug 31 2020, pp. 5583–5589.

[26] B. Cohen, M. Phillips, and M. Likhachev, "Planning single-arm manipulations with n-arm robots," in *Proceedings of Robotics: Science and Systems*, Berkeley, CA, July 12–16 2014.

[27] A. Zelinsky, "A mobile robot exploration algorithm," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 6, pp. 707–717, Dec 1992.

[28] H. Choset and J. Burdick, "Sensor based planning and nonsmooth analysis," in *IEEE International Conference on Robotics and Automation*, San Diego, CA, May 8–13 1994, pp. 3034–3041.