A General Technique for Fast Comprehensive Multi-Root Planning on Graphs by Coloring Vertices and Deferring Edges

Christopher M. Dellin Siddhartha S. Srinivasa
{cdellin,siddh}@cs.cmu.edu
The Robotics Institute
Carnegie Mellon University

***Abstract*—We formulate and study the *comprehensive multi-root* (CMR) planning problem, in which feasible paths are desired between multiple regions. We propose two primary contributions which allow us to extend state-of-the-art sampling-based planners. First, we propose the notion of *vertex coloring* as a compact representation of the CMR objective on graphs. Second, we propose a method for *deferring edge evaluations* which do not advance our objective, by way of a simple criterion over these vertex colorings. The resulting approach can be applied to any CMR-agnostic graph-based planner which evaluates a sequence of edges. We prove that the theoretical performance of the colored algorithm is always strictly better than (or equal to) that of the corresponding uncolored version. We then apply the approach to the Probabalistic RoadMap (PRM) algorithm; the resulting *Colored Probabalistic RoadMap* (cPRM) is illustrated on 2D and 7D CMR problems.**

I. Introduction

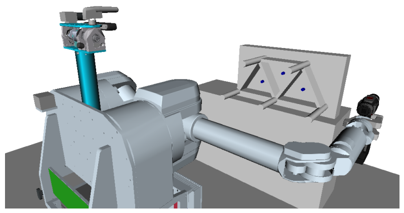
Many real-world tasks require a robot to quickly accomplish multiple subtasks without a prescribed order. Consider a personal assistant robot clearing several objects from a tabletop, or a manufacturing robot performing several welds on a novel workpiece. Furthermore, each subtask often permits multiple suitable robot configurations such as grasps of an object or orientations of a tool. Even if only one end effector pose is valid, manipulator redundancy enables many feasible configurations. We are interested in efficiently finding feasible paths for such problems.

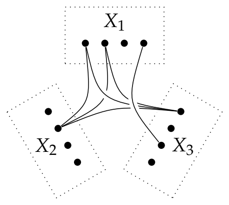
For example, consider the robot in Fig.1a. It is tasked with using a handheld drill to tighten three bolts (blue) during assembly of a truss structure. Each bolt permits an entire manifold of permissible robot configurations which would allow completion of the sub-task, many of which may be either directly infeasible or difficult to reach given the environment and other subtasks. We call each of these configurations a *root*, and collect all roots which satisfy a particular subtask into a *root set*.

The planning problem, then, can be formulated as finding a diverse set of paths between these root sets. We formalize this problem as the *comprehensive multi-root* (CMR) planning problem (Sec. III) and ask the question:

> How can we efficiently maximize connections between pairs of roots in *different* root sets?

Fig.1b illustrates a partial solution to a problem with three root sets where 6 out of a total possible 27 connections have been discovered.


(a) This drilling task can be formulated as a multi-root planning problem.



(b) Three root sets each containing four roots.

Fig. 1: The comprehensive multi-root (CMR) problem.

We focus our attention on graph-based planners (e.g. the PRM [1]) which build an *explicit* graph approximating the free configuration space of the robot by incrementally evaluating (for collisions) an *implicit* graph comprising edges between sampled vertices (Sec. IV). This explicit graph is then queried to find feasible paths between different roots.

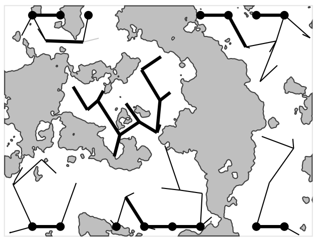
We can abstract these algorithms as edge sources which are emitting potential edges for evaluation. Our key insight is to introduce a *deferred edge queue* which postpones the evaluation of certain edges in order to maximize the CMR objective during planning.

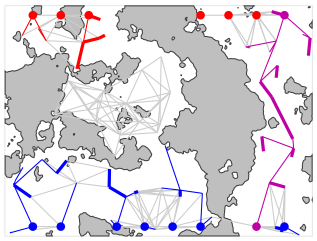
We do so by formally representing the CMR objective in terms of *vertex coloring*. This naturally suggests an evaluation criterion which we can use to prioritize some edge evaluations over others (Sec. V). When applied to any edge-evaluating algorithm, the resulting colored algorithm is provably superior to the original (Sec. VI).

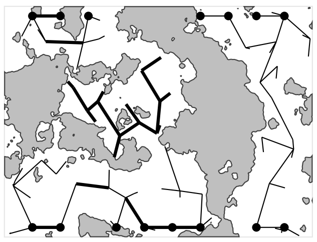
As an example of this approach, we implemented a colored version of the PRM algorithm (Sec. VII), and applied it to CMR problems in 2D and 7D.

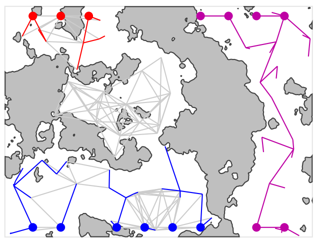
Fig.2 shows the results of running both the uncolored and colored versions of the forest-of-trees PRM on a 2D problem with two root sets (one at top, one at bottom). Each algorithm works on the same set of samples; since they share the same connection rule, this induces the same set of potential edges. The remainder of this section calls out the primary features of colored algorithms by way of this example.

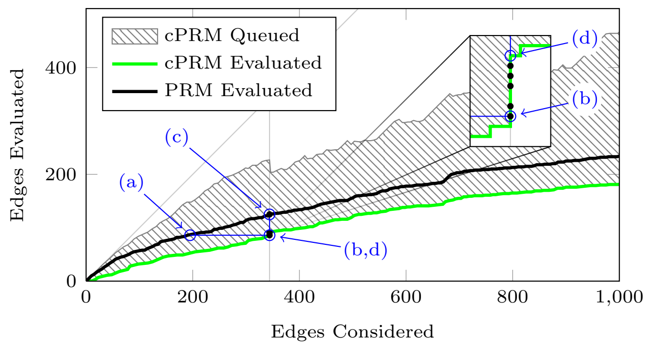
Fig. 2a and 2b compare the tree states of the algorithms after having performed a collision check on the same number of edges (86). Bolded edges show the difference between the free-edge graphs built by each algorithm. By this point, the colored algorithm


(a) PRM, 86 Edges Evaluated (195 Edges Considered, 0 Pairs)

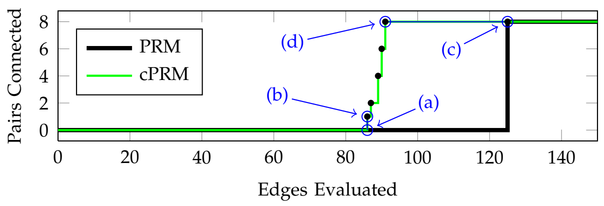

(b) cPRM, 86 Edges Evaluated (344 Edges Considered, 1 Pair)


(c) PRM, 344 Edges Considered) (125 Edges Evaluated, 8 Pairs)


(d) cPRM, 344 Edges Considered) (91 Edges Evaluated, 8 Pairs)



(e) Edges considered, evaluated, deferred, and skipped.



(f) CMR objective vs edges evaluated.

Fig. 2: A comparison between an uncolored (2a,2c) and colored (2b,2d) forest-of-trees PRM on the same sequence of edges. The latter algorithm colors the graph according to root set connectivity as described in Section IV. Plot (2e) shows the evolution of edges in both algorithms as they progress as outlined in the introduction, and plot (2f) shows the number of between-rootset pairs found by each.

has found its first connection between the two root sets (at right, shown in violet), whereas the uncolored algorithm has yet to find a connection. This is possible because the colored algorithm has chosen to defer evaluation of many edges (shown in light grey).

Fig. 2c and 2d show a different comparison. In this case, the tree states are compared after each algorithm has finished considering the same number of edges (344). Here, both algorithms have achieved the same objective by connecting the same number of pairs (8). However, the tree built by the colored algorithm is a subset of that built by the uncolored one; it has therefore performed fewer edge evaluations (91 to 125).

Fig.2e shows the evolution of edges for each algorithm. When the uncolored algorithm considers each edge, it is either evaluated for collision (below the black line) or skipped in order to maintain a forest of trees (above the black line). The colored algorithm additionally maintains a queue of deferred edges (pattern). This allows it to perform fewer edge evaluations (below the green line) for a given input edge. In Section VI, we prove that these deferments do not affect the CMR objective. The tree state comparisons discussed above are called out in blue. Each edge evaluation which results in additional connected pairs is additionally denoted with a black circle.

Fig.2f shows the number of connected inter-root pairs as a function of the number of edges evaluated. The colored algorithm finds its first pair earlier than the uncolored algorithm (86 vs 125). Once the colored algorithm finds one pair, it quickly finds others. Again, the tree state comparisons are called out in blue.

II. Related Work

While our formulation of the CMR problem itself is newly introduced, so that no existing planners are explicitly tailored for it, it is intimately related to several classes of well-studied motion planning problems.

The classical FindPath or mover's problem [2], [3] concerns finding a path between two single states. However, many approaches are designed to handle connections between multiple starts, goals, or both. A problem with multiple starts and goals can be represented as a CMR problem with $N = 2$ root sets.

For example, the original A* algorithm [4] searched for a path between a single start state and multiple goal states. While single-query sampling-based algorithms such as the RRT [5], [6] generally consider a single start and goal state, extensions [7] have allowed such planners to consider both multiple starts and multiple goals. Trajectory optimization can also be applied to the motion planning problem; while typically considered for single start and goal states, recent generalizations [8], [9] have extended them to sets of starts or goals.

However, even those approaches that handle start and/or goal sets typically terminate once a single path is found, and do not attempt to find multiple diverse connections between different root sets.

To accomodate more than two root sets, approaches for multi-query planning (e.g. [1]) may appear better suited for finding multiple diverse connections. For example, they have been used for efficiently solving the single-object manipulation problem [10].

Additionally, some approaches perform similar deferment on edge evaluations to our approach. For ex-

Algorithm 1 Edge Evaluation (graph or forest)

```
1: procedure EVALUATE($g, v_a, v_b$)
2:     if $v_a$ and $v_b$ in same connected component then
3:         return                          ▷ Optional, for forest
4:     if EDGEFREE($v_a, v_b$) then
5:         $g.E \leftarrow g.E \cup \{(v_a, v_b)\}$
```

ample, so-called "lazy" algorithms, such as [11], [12] defer collision checking until edges are to be used.

Other recent work has married symbolic and geometric planning for multiple subtasks with multi-modal planning [13], temporal logic [14], or hierarchical or bridged representations and interfaces [15], [16], [17].

III. The Comprehensive Multi-Root Problem

We work with the robot's configuration space X and its collision-free subset X_{free}. We consider the general problem with N root sets in this space $\{X_1, \ldots, X_N\}$. We seek a diverse set of feasible paths between root sets – that is, we want to maximize the number of connected roots between sets. We call this the *comprehensive multi-root* (CMR) planning problem.

We track progress via the r-score:

$$r = \left|\{\textsc{Path}(x_a, x_b) \mid x_a, x_b \text{ in different root sets}\}\right|. \quad (1)$$

For example, the solution illustrated in Fig.1b has an r-score of 6 out of a maximum of 27. Our objective is to *maximize* the r-score as *quickly* as possible. We define this more formally next.

IV. The CMR Problem on Graphs

We focus on motion planning approaches where X_{free} is approximated by a graph g. For example, in a search-based planning approach such as A* [4], g is a state-lattice, whereas in a sampling-based planning approach such as the PRM [1], g is a random geometric graph.

A key feature of these approaches is that the graph g is *implicit*, i.e. incrementally discovered and/or evaluated as the search progresses. For example, in a PRM, a sample is drawn at random from X_{free}, all of the local paths to its neighbors on g are evaluated for collisions (using a function like Algorithm 1) and collision-free paths are incrementally added to g as edges.

However, for applications we're interested in, edge evaluations are expensive [3] because they require several collision checking queries between complex geometries. We therefore want to maximize our objective (1) while minimizing the number of edges to be evaluated. To accomplish this, we return to our CMR objective.

V. A General Technique for Fast CMR Planning on Graphs

We use the structure of the CMR problem to motivate a new class of algorithms. We do so by redefining our objective in terms of the graph's structure, and then describing the two insights which characterize our approach: *coloring vertices* and *deferring edges*. We then show how an algorithm can be extended to include these features.

A. Viewing our Objective with Vertex and Graph Colorings

Consider a graph g in X_{free} applied to a CMR problem. Here, we show how we can succinctly represent our objective (1) with respect to this graph.

We define the *coloring* $\mathbf{c}(v)$ of vertex v as

$$\mathbf{c}(v) = \begin{bmatrix} c_1 \\ \vdots \\ c_N \end{bmatrix} \text{ with } c_i = \left|\{\textsc{Path}(v, x) \mid x \in X_i\}\right|. \quad (2)$$

The coloring[1] of a vertex is an N-vector, with each element c_i the number of reachable roots in the corresponding root set X_i w.r.t. the graph.

Further, we define the *colored norm* $||\cdot||$ as

$$||\mathbf{c}(v)|| = \sum c_i c_j \text{ for } i < j. \quad (3)$$

The colored norm counts the number of pairs of roots in different root sets that are connected through the vertex.

Proposition 1: Every vertex in a given connected component k has the same coloring $\mathbf{c}_k$.

Proof By definition, every vertex in a connected component has a PATH to every other vertex; therefore the color of all constituent vertices is equal. □

We now define the colored norm of a connected component as the colored norm of *any* of its constituent vertices.

Finally, we define the colored norm of a graph g as the sum of the colored norms of all of its connected components:

$$||g|| = \sum_k ||\mathbf{c}_k|| \quad (4)$$

For a graph solving a CMR problem, the colored norm of the graph (4) is equal to the CMR objective (1).

B. Prioritizing Edges via the Graph's Colored Norm

Due to the expense of evaluating edges as mentioned in Sec. IV, we endeavor to maximize the CMR objective (4) while minimizing the number of edges evaluated. The reformulation of the objective in terms of graph and vertex coloring suggests a method to prioritize certain edge evaluations over others.

We first consider the myopic criterion shown in Algorithm 2. When a potential edge between two vertices v_a, v_b is considered, we determine whether the inclusion of this edge in our graph g would immediately improve our objective. If so, we proceed to evaluate the edge.

However, when initialized with a graph consisting of only roots, this criterion will only allow edges which connect directly between roots of different root sets. In complex problems, the probability that such edges

[1]Our use of the term "coloring" with respect to graph vertices is not to be confused with the proper graph color assignment problem.

Algorithm 2 Myopic and Balanced Edge Criterions

```
1:  function MyopicCriterion($g, v_a, v_b$)
2:      $g'.V \leftarrow g.V$
3:      $g'.E \leftarrow g.E \cup \{(v_a, v_b)\}$
4:      if $||g'|| > ||g||$ then                ▷ Check objective
5:          return True                          ▷ Evaluate edge
6:      else
7:          return False                  ▷ Do not evaluate edge
8:  function Criterion($g, v_a, v_b$)              ▷ Balanaced
9:      $g'.V \leftarrow g.V$
10:     $g'.E \leftarrow g.E \cup \{(v_a, v_b)\}$
11:     if $||g'|| > ||g||$ then                ▷ Check objective
12:         return True                          ▷ Evaluate edge
13:     else if exatly one of $\mathbf{c}(v_a), \mathbf{c}(v_b)$ is $\mathbf{0}$ then
14:         return True                          ▷ Evaluate edge
15:     else
16:         return False                  ▷ Do not evaluate edge
```

are feasible is quite low – the purpose of intermediate vertices is to find a path within complex X_{free} spaces.

Therefore, we propose a balanced criterion which allows for exploration of X_{free}. This approach additionally evaluates edges which would provide an uncolored vertex (i.e. with $\mathbf{c}(v) = \mathbf{0}$) an initial coloring.

C. A Compact Depiction of Vertex Coloring

Note that in the graph colored norm test $||g'|| > ||g||$, the new graph g' differs from g by a single new edge (v_a, v_b). In the case that v_a and v_b are already in the same connected component, the additional edge trivially has no effect on the colored norm, and the test fails. Otherwise, the test can be restated as

$$||g'|| > ||g|| \text{ iff } ||\mathbf{c}_a + \mathbf{c}_b|| > ||\mathbf{c}_a|| + ||\mathbf{c}_b||. \quad (5)$$

By definition of the colored norm, taking for brevity $\mathbf{c}_x = [x_1, \ldots, x_N]^\top$, we can express the inequality in terms of the coloring's vector components as:

$$\sum_{i<j}(a_i + b_i)(a_j + b_j) > \sum_{i<j} a_i a_j + \sum_{i<j} b_i b_j \quad (6)$$

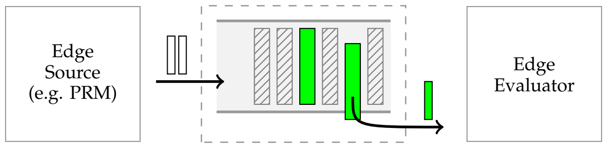
$$\sum_{i<j} a_i b_j + \sum_{i<j} b_i a_j > 0 \quad (7)$$

or simply

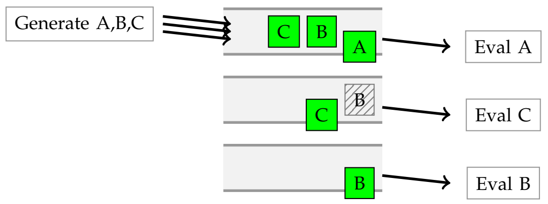
$$||g'|| > ||g|| \text{ iff } a_i, b_j \text{ nonzero for some } i \neq j. \quad (8)$$

Thus, the result of our criterion for an edge (v_a, v_b) is dependent only on the distribution of nonzero entries of each coloring $\mathbf{c}_a$ and $\mathbf{c}_b$. Therefore, in our examples, we depict an N-coloring by simply assigning a primary color to each vector component, e.g.

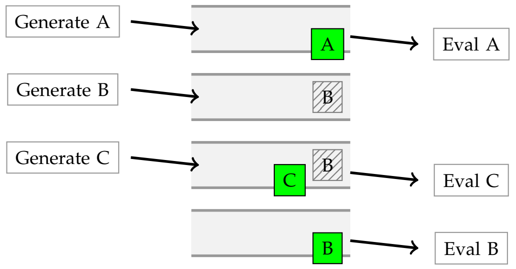
$$\mathbf{c} : \begin{bmatrix} 2 \\ 0 \end{bmatrix} \mapsto \text{Red}, \quad \mathbf{c} : \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mapsto \text{Blue, or} \quad \mathbf{c} : \begin{bmatrix} 4 \\ 5 \end{bmatrix} \mapsto \text{Violet.}$$



(a) The edge queue acts as a prioritized buffer which only dequeues edges which meet the given criterion (🟩) by indefinitely deferring failing edges (▨).



(b) The edge source can generate a batch of edges; the edge queue then processes until no more can be dequeued.



(c) The edge source and edge queue can be interleaved so that edges are generated in just-in-time fashion.

Fig. 3: Illustration of the edge queue. Edges which pass the criterion (shown in solid green) are dequeued in first-in-first-out order; others (shown in striped grey) remain in the queue. The algorithm produces identical results in either batch or interleaved operation. The examples from (3b,3c) are from the problem described later in Fig.4.

D. Deferring Edges

While a considered edge may fail our criterion early in the planning process, because the criterion depends on the state of the graph, the failed edge may eventually pass once the graph has grown. Therefore, instead of overlooking a failed edge entirely, we instead defer it until subsequent iterations. To support this, we propose the introduction of a *deferred edge queue*. The mechanics of this queue are illustrated in Fig.3.

The edge queue can be implemented as a priority queue with binary priority $\{low, high\}$ such that (a) only elements with *high* priority are dequeued, (b) elements with the same priority are dequeued in FIFO order, and (c) element priorities can be changed.

Such an edge queue can be introduced into any incremental graph construction algorithm. In the case that the edge adjacency rule for each considered edge is independent of past edge evaluations, the proofs in Section VI provide theoretical guarantees on performance as a result of the introduction of the queue.

An example of introducing the colored edge queue into a generic algorithm is shown in Algorithm 3. The GetNextEdge implements any desired graph construction rule, such as an r-disk or k-nearest PRM. Note

Algorithm 3 Converting a Generic Algorithm to Use the Colored Edge Queue

```
1: procedure GenericAlgorithm
2:     $g.V \leftarrow \emptyset$; $g.E \leftarrow \emptyset$
3:
4:     loop
5:         $(v_a, v_b) \leftarrow$ GetNextEdge()
6:         Evaluate$(g, v_a, v_b)$    ▷ graph or forest
```

```
1: procedure GenericColoredAlgorithm
2:     $g.V \leftarrow \emptyset$; $g.E \leftarrow \emptyset$
3:     $Q_{edge} \leftarrow \emptyset$    ▷ Initialize an empty edge queue
4:     loop
5:         $(v_a, v_b) \leftarrow$ GetNextEdge()
6:         Consider$(g, Q_{edge}, v_a, v_b)$    ▷ Instead of Evaluate
```

```
1: procedure Consider$(g, Q_{edge}, v_a, v_b)$
2:     $Q_{edge}$.Push$((v_a, v_b))$
3:     while $(v_a, v_b) = Q_{edge}$.PopFiltered() using Criterion$(g, \cdot)$ do    ▷ Dequeue first edge meeting criterion
4:         Evaluate$(g, v_a, v_b)$    ▷ graph or forest
```

that while the edge source and queue processing can be batched, we show the interleaved case for convenience. During queue processing, each edge (in the order it was enqueued) is considered via our criterion (line 3). If the edge passes this check, it is dequeued for evaluation; otherwise, it is deferred until later, remaining in the queue. The dequeued edge is then evaluated according to the traditional EVALUATE rule (e.g. forest-of-trees).

The introduction of the deferred edge queue based on our criterion produces the behavior seen in Fig.2. A more illuminating example is shown in Fig.4.

VI. Analysis

Here, we analyze and compare a generic forest-of-trees (i.e. forest) algorithm with its colored variant (as shown in Algorithm 3). The proofs in this section depend on two lemmas which we investigate in turn.

A. Criterion Failure Transitivity

Lemma 1: Given a graph g and three of its constituent vertices v_a, v_b, and v_c, if CRITERION(g, v_a, v_b) and CRITERION(g, v_b, v_c) both evaluate to False, then CRITERION(g, v_a, v_c) also evaluates to False.

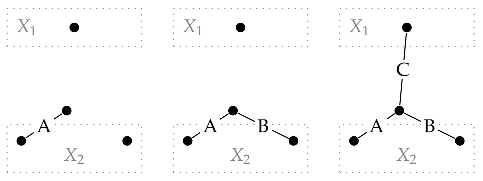
Proof If the criterion evaluates to False for both (v_a, v_b) and (v_b, v_c), then it implies two propertices of their colorings $\mathbf{c}_a$, $\mathbf{c}_b$, and $\mathbf{c}_c$. First, either they are all zero or they are all nonzero. Second, from (8) it holds that that $a_i b_j = 0 \; \forall \; i \neq j$ and $b_j c_k = 0 \; \forall \; j \neq k$.

In the case that they are all the zero coloring, then CRITERION(g, v_a, v_c) is trivially False.

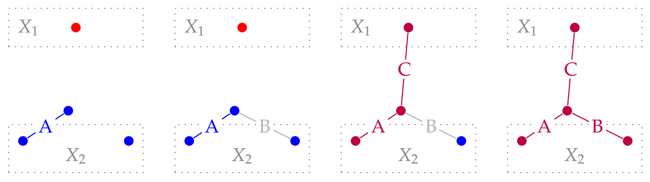
If they are all nonzero, there must be some nonzero element b_j in $\mathbf{c}_b$. It therefore holds that $a_i = 0 \; \forall \; i \neq j$ and $c_k = 0 \; \forall \; k \neq j$. Thus, $a_i c_k$ can only be nonzero if $i = j = k$, implying that $a_i c_k = 0 \; \forall \; i \neq k$. By (8), this shows that CRITERION(g, v_a, v_c) evaluates to False. □

B. Comparative Analysis via a Composite Algorithm

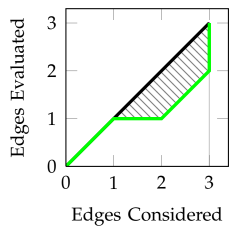
For the purpose of comparing the uncolored and colored forest algorithms, we consider the behavior of both algorithms considering the same sequence of edges $\{e_1, e_2, \dots\}$. When a new edge in the sequence is considered, the composite algorithm first allows the uncolored version to update, and then allows the colored version to update. While the algorithms evolve independently, we can directly compare their behavior because they share the same set of potential edges.



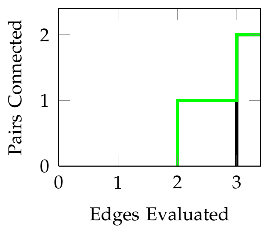
(a) The uncolored algorithm evaluates each edge immediately.



(b) The colored algorithm defers edge B until a pair is found.



(c) Queue state



(d) Pairs vs Work

Fig. 4: A very small problem as an example. Edges are considered by both algorithms in order A,B,C. The colored algorithm achieves a higher r-score earlier by deferring edge B. Fig.4c shows that while the uncolored algorithm (black) evaluates each edge as it is considered, the colored version (green) defers an edge evaluation. Fig.4d shows that the colored algorithm connects its first pair after two evaluations.

We track the state of each algorithm by assigning one of four labels to each considered edge:

Q	in the queue
F	evaluated and collision-free
C	evaluated and in collision
S	skipped due to forest constraint

To compare the progress of the uncolored and colored algorithms, each considered edge is tagged using a pair of these labels. For example, an edge labeled FQ has been evaluated to be collision-free by the uncolored algorithm, but is still in the queue of the colored

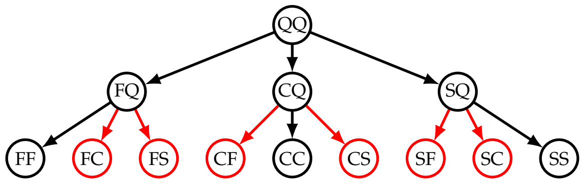


Fig. 5: Possible composite label transitions. Section VI-B proves that the red states can never be reached.

algorithm. There are sixteen possible composite labels.

The proofs in this section rely on a proposed invariant over these composite labels. Once we have shown that the invariant holds, we can use it to prove relations between the trees built by each algorithm.

Invariant 1: Every composite label for a considered edge is in $\{QQ, FQ, CQ, SQ, FF, CC, SS\}$.

Lemma 2: Invariant 1 holds throughout the composite algorithm.

Proof To prove Lemma 2, we show that it is true at the start of execution, and then show inductively that it is maintained throughout execution of the composite algorithm. See Fig.5 for an illustration of allowed transitions that we demonstrate here.

Base case: When the composite algorithm begins, no edges have yet been considered, so the invariant is trivially true.

When a new edge is considered, it is placed in both queues with label QQ (and the invariant holds).

Next, the uncolored algorithm updates. Since edges in its queue are always immediately considered, the edge is dequeued and evaluated. Depending on the connectivity of the space and the existing graph, each transitions to one of $\{FQ, CQ, SQ\}$ (and the invariant holds). When finished, the uncolored queue is empty, and so no edges remain as QQ.

Next, the colored algorithm updates. Zero or more edges are iteratively dequeued because they satisfy the CRITERION. Each edge dequeued is initially labeled as one of $\{FQ, CQ, SQ\}$. We consider each case in turn.

If the dequeued edge e_i (connecting v_a and v_b) is labeled FQ or CQ, we first show that it cannot transition to FS and CS, respectively. Imagine that it did get skipped by the colored algorithm. That would imply an existing path P through the colored graph between v_a and v_b consisting of edges of type XF. Due to our invariant, such existing edges must be FF. But, then, the uncolored algorithm would have also skipped it. Therefore, dequeued edges having FQ or CQ cannot transition to FS and CS, respectively.

Furthermore, FQ cannot transition to FC, since that would imply that the two algorithms evaluated the same edge yielding a different result. Similarly, CQ cannot transition to CF. Therefore, FQ must transition to FF and CQ to CC, and the invariant holds.

Last, we consider the case that the dequeued edge e_i is labeled SQ. Since it was skipped by the uncolored algorithm, it must have been in the same connected component w.r.t. the uncolored graph at the time it was considered. Therefore, there must be a path P of edges $(e_{p_1}, e_{p_2}, \dots)$ consisting of *earlier* edges (i.e. $p_j < i$) through X_{free}, such that each edge e_{p_1} is labeled FX for some colored label X. In fact, due to our invariant, each edge must be either FQ or FF.

We now show that it is impossible for any edge in P to be FQ. For a moment, consider that one or more edges in P are FQ. Since all edges in P are earlier than e_i, all that are FQ must not satisfy the CRITERION, or they would have been dequeued before e_i. By Lemma 1, the endpoints of P must therefore also fail CRITERION. However, since edge e_i was dequeued, its vertices v_a, v_b must simultaneously satisfy CRITERION. Due to this contradiction, we know that no edges on P are FQ.

Therefore, all edges in P must be FF, and e_i will be skipped by the colored algorithm because both v_a and v_b are in the same connected component. Therefore, e_i will receive SS, and the invariant holds. □

C. Theoretical Proofs

Theorem 1: For the same sequence of edges, the free-edge graph built by the uncolored forest algorithm is a subset of that built by the colored forest algorithm.

Proof To prove Theorem 1, it is sufficient to show that no edge is non-free in the uncolored algorithm and free in the colored algorithm. Under our labeling, such an edge would have a label in $\{QF, CF, SF\}$. Lemma 2 asserts that such a labeling is impossible. □

Theorem 2: For the same sequence of edges, every edge evaluated by the colored forest algorithm is also evaluated by the uncolored forest algorithm.

Proof To prove Theorem 2, it is sufficient to show that no edge is both unevaluated by the uncolored algorithm and evaluated by the colored algorithm. Such an edge would have a label in $\{QC, QF, SC, SF\}$. Lemma 2 asserts that such a labeling is impossible. □

Theorem 3: For the same sequence of edges, after both algorithms have finished processing, the graph built by the colored forest algorithm has the same colored norm $||g||$ as that built by the uncolored forest algorithm.

Proof To prove Theorem 3, we show that the value of the colored norm of the graph built by the colored algorithm $||g_c||$ can be neither greater than nor less than that built by the uncolored algorithm $||g_u||$.

First, since we have shown in Theorem 1 that the colored free-edge graph is a subset of the uncolored free-edge graph, it follows directly that every pair of roots connected by g_c must be also connected by g_u. Therefore, $||g_c||$ must not be greater than $||g_u||$.

Next, we suppose that $||g_c||$ is less than $||g_u||$ after both algorithms have processed their queues. In such a case, there must exist a pair of roots v_a, v_b in different root sets that are connected through g_u, but not through

Algorithm 4 Colored PRM

```
1:  procedure ColoredPRM($\{X_1, \ldots, X_N\}$)        ▷ Algorithm initialized with $N$ root sets
2:      $g.V \leftarrow \emptyset$; $g.E \leftarrow \emptyset$        ▷ Initialize with empty graph
3:      $Q_{edge} \leftarrow \emptyset$        ▷ Initialize edge queue
4:      while Continue($g$) do        ▷ Run until termination
5:          if all roots not added then
6:              $v_{new} \leftarrow$ NextRoot($\{X_1, \ldots, X_N\}$)        ▷ Initially add each root to the graph
7:          else
8:              $v_{new} \leftarrow$ SampleFree()        ▷ Sample a vertex in free space
9:          $g.V \leftarrow g.V \cup \{v_{new}\}$        ▷ Add vertex to graph
10:         for $v_{near} \in$ Nearby($g, v_{new}$) do        ▷ Consider all existing nearby vertices
11:             Consider($g, Q_{edge}, v_a, v_b$)
```

g_c. Consider such a path P connecting v_a and v_b; since it connects through g_u, it must be composed of composite edges labeled FQ or FF due to Lemma 2. Since the colored queue has finished processing, every edge labeled FQ must fail CRITERION. By Lemma 1, the endpoints of P must therefore also fail CRITERION. However, because v_a and v_b are in different root sets, they must satisfy CRITERION. This contradiction proves that $||g_c||$ must not be less than $||g_u||$. □

D. Conclusion

Due to Theorems 2 and 3, we have shown that after each iteration, the colored forest algorithm achieves the same CMR objective score as the uncolored algorithm, while performing fewer (or equal) edge evaluations.

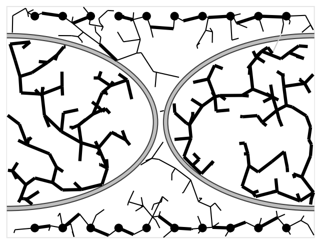
VII. The Colored PRM Algorithm

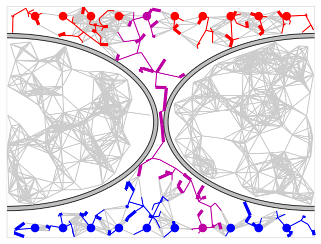
To illustrate the performance of the coloring approach, we applied it to the PRM. The resulting algorithm, the ColoredPRM, is shown in Algorithm 4. Note that we commit to neither a particular termination criterion Continue, a sampling procedure SampleFree, nor an edge evaluation function EdgeFree. Note that we also don't specify Nearby; indeed, the colored queue can be introduced into any algorithm which searches over edges of an incrementally-constructed implicit graph (e.g. classical PRM, r-disk PRM, etc) in any order (e.g. nearest first, etc).

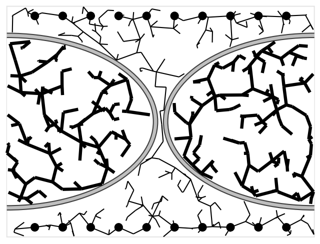
The algorithm begins with an empty edge queue (line 3). Each iteration proceeds as a classical PRM; a new vertex is sampled from X_{free} and added to the graph (lines 5-9), and we iterate over a subset of existing vertices (e.g. within a radius) in some order (e.g. by increasing distance) (line 10). Whereas a classical PRM would immediately evaluate each edge, we instead call Consider as defined in Algorithm 3 (line 11). This enqueues the edge and then processes the queue.

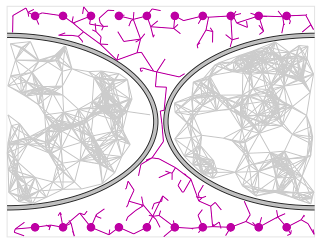
A. Qualitative Behavior

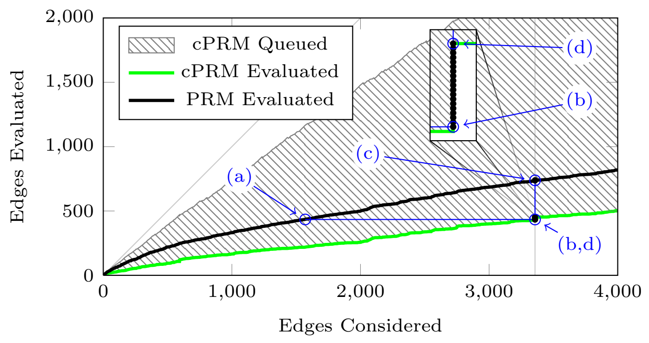
The forest-of-trees ColoredPRM evolves similarly to its uncolored variant, with two exceptions. First, edges between the same root set are initially deferred. Second, edges that are in unreached portions of the


(a) PRM, 434 Evaluated (1570 Considered, $r = 0$)


(b) cPRM, 434 Evaluated (3357 Considered, $r = 1$)


(c) PRM, 3357 Considered (738 Evaluated, $r = 100$)


(d) cPRM, 3357 Considered (452 Evaluated, $r = 100$)



(e) Edges considered, evaluated, deferred, and skipped.

Fig. 6: The cPRM indefinitely defers edges in unreachable regions. Fig. 6b shows the colored algorithm just as it finds its first connected pair; no other roots are yet connected yet, as potential connecting edges have been deferred. Once the first connection is made, the colored algorithm quickly connects all roots, from (b) to (d).

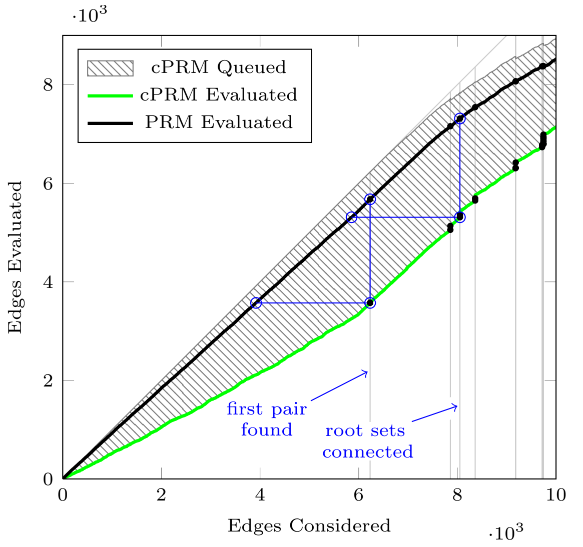


Fig. 7: Edge evolution for the HERB drilling problem. Black circles denote additional pairs connected.

state space are entirely deferred. This is most clearly exemplified in Fig.6. This tends to induce behavior reminiscent of a multi-directional RRT.

B. Complex 7-DOF Manipulator Drilling Problem

We applied the COLOREDPRM to a drilling problem on the HERB robot [18], shown in Fig.1a. Drill poses and manipulator inverse kinematic solutions for the 7-DOF arm were discretized and tested for collision; the resulting root sets contained 25, 112, and 142 roots. Together with the single starting configuration comprising its own root set, the problem consisted of four root sets with a total of 280 roots. Both the uncolored and colored variants of the PRM were run with the same set of samples with an r-disk radius of 3.0 rad and collision checking resolution of 0.02 rad.

The results are shown in Fig.7. The first pair was found after 3572 edge evaluations (5802 collision checks) The four root sets were fully connected after 5310 evaluations (23627 checks). For the uncolored PRM, this compares to 5675 evaluations (15493 checks) until the first pair was found, and 7314 evaluations (32861 checks) until the sets were fully connected.

VIII. Discussion

We formulated the comprehensive multi-root (CMR) planning problem, and by representing its objective over a graph using vertex coloring, we motivated the introduction of a deferred edge queue into incremental graph building algorithms. We proved that the resulting colored algorithms are superior to their uncolored counterparts for the CMR problem, and showed examples in both 2D and 7D spaces.

In this paper, we did not consider optimality of the solution, opting instead to focus on the forest-of-trees approach to graph building. We note, however, that our criterion applies equally well to full graphs; indeed it induces similar behavior to a forest-of-trees approach for all unconnected roots.

References

[1] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[2] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[3] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

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

[5] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 1, 1999, pp. 473–479.

[6] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 2, 2000, pp. 995–1001 vol.2.

[7] D. Berenson, S. Srinivasa, D. Ferguson, and J. Kuffner, "Manipulation planning on constraint manifolds," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2009.

[8] A. Dragan, N. Ratliff, and S. Srinivasa, "Manipulation planning with goal sets using constrained trajectory optimization," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, May 2011, pp. 4582–4588.

[9] J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: Science and Systems*, vol. 9, no. 1, 2013, pp. 1–10.

[10] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *The International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 729–746, 2004. [Online]. Available: http://ijr.sagepub.com/content/23/7-8/729.abstract

[11] R. Bohlin and E. Kavraki, "Path planning using Lazy PRM," in *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, vol. 1, 2000, pp. 521–528 vol.1.

[12] G. Sánchez-Ante and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *ISRR*, 2001, pp. 403–417.

[13] K. Hauser and J.-C. Latombe, "Multi-modal motion planning in non-expansive spaces," *The International Journal of Robotics Research*, vol. 29, no. 7, pp. 897–915, 2010. [Online]. Available: http://ijr.sagepub.com/content/29/7/897.abstract

[14] A. Bhatia, L. Kavraki, and M. Vardi, "Sampling-based motion planning with temporal goals," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, May 2010, pp. 2689–2696.

[15] S. Cambon, R. Alami, and F. Gravot, "A hybrid approach to intricate motion, manipulation and task planning," *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 104–126, 2009. [Online]. Available: http://ijr.sagepub.com/content/28/1/104.abstract

[16] F. Gravot, S. Cambon, and R. Alami, "aSyMov: A planner that deals with intricate symbolic and geometric problems," in *Robotics Research. The Eleventh International Symposium*, ser. Springer Tracts in Advanced Robotics, P. Dario and R. Chatila, Eds. Springer Berlin Heidelberg, 2005, vol. 15, pp. 100–110. [Online]. Available: http://dx.doi.org/10.1007/11008941_11

[17] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, and P. Abbeel, "Combined task and motion planning through an extensible planner-independent interface layer," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, May 2014, pp. 639–646.

[18] S. Srinivasa, D. Berenson, M. Cakmak, A. Collet, M. Dogar, A. Dragan, R. Knepper, T. Niemueller, K. Strabala, M. Vande Weghe, and J. Ziegler, "HERB 2.0: Lessons learned from developing a mobile manipulator for the home," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2410–2428, Aug. 2012.