# The Minimum Constraint Removal Problem with Three Robotics Applications

Kris Hauser

**Abstract** This paper formulates a new Minimum Constraint Removal (MCR) motion planning problem in which the objective is to remove the fewest geometric constraints necessary to connect a start and goal state with a free path. I present a probabilistic roadmap motion planner for MCR in continuous configuration spaces. The planner operates by constructing increasingly refined roadmaps, and efficiently solves discrete MCR problems on these networks. A number of new theoretical results are given for discrete MCR, including a proof that it is NP-hard by reduction from SET-COVER, and I describe two search algorithms that perform well in practice. The motion planner is proven to produce the optimal MCR with probability approaching 1 as more time is spent, and its convergence rate is improved with various efficient sampling strategies. The planner is demonstrated on three example applications: generating human-interpretable excuses for failure, motion planning under uncertainty, and rearranging movable obstacles.

## 1 Introduction

Planners typically operate in binary fashion: they output a valid path if successful, but if they fail, then they provide no explanation for the failure. For several applications, it would be useful for planners to provide more informative explanations about their failures.

- In human-robot interaction, semantically meaningful explanations would help people diagnose and rectify problems.
- For a CAD/CAM designer of a compact assembly, a planner that explained why part cannot be accessed during assembly or repair would help the designer make appropriate modifications.
- Explanations for planner failures would help a technician debug robot crashes.

School of Informatics and Computing, Indiana University, e-mail: hauserk@indiana.edu

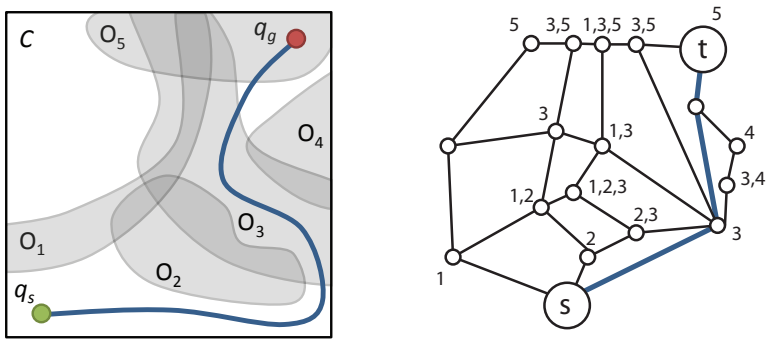


**Fig. 1** MCR illustrated in a 2D configuration space. At least two obstacles ($O_3$ and $O_5$) need to be removed for a path to connect the start and goal. At right, a partition of the continuous space yields a discrete MCR problem on a graph. Each node is labeled with the subset of obstacles that "cover" the node. The MCR is a minimal subset of obstacles that covers all nodes in a path from the source $s$ to the target $t$ (highlighted). Because the obstacle partition is hard to compute exactly, our planner constructs progressively better approximations by growing a probabilistic roadmap.

- Robots that manipulate obstacles can use explanations to help it decide which obstacles must be moved in order to admit a feasible path.

Recent work has studied excuse-making in the symbolic planning setting by seeking small changes to the initial state that yield a feasible path [5]. A similar notion in the motion planning literature is the problem of finding *disconnection proofs* that certify that no path can be found [1, 2, 16, 11]. Proving infeasibility is computationally challenging and so prior techniques are effectively limited to low-dimensional [16, 11], geometrically simple [1], or algebraically simple [2] settings. Also, these approaches are not constructive in that they do not consider how to change a space in order to yield a feasible path.

I present a new motion planning formulation in which the planner generates explanations for failure by exploring counterfactual scenarios where subsets of constraints are removed from the state space (Figure 1). In particular we are interested in minimizing the number of constraints necessary to admit a solution, because explanations for failure should be parsimonious. This new class of *minimum constraint removal* (MCR) problems differs significantly from prior motion planning problems. It exhibits the curse of dimensionality in continuous spaces along with combinatorial complexity, because subsets of the constraint set (whose size can range into hundreds or thousands) must be enumerated in order to find optimal solutions. As a result, we seek efficient approximate solutions.

I present a sampling-based motion planner that approximately solves the MCR problem in high-dimensional configuration spaces. It incrementally grows a probabilistic roadmap (PRM) in the configuration space that approximates the connectivity of the obstacle partition, and computes the path in the roadmap that violates the fewest constraints. I prove that it is asymptotically optimal, in that as more time is spent planning, it computes the fewest number of obstacles needed to admit a feasi-

ble path with probability approaching 1 under relatively weak conditions. Moreover it is an *any-time* algorithm that can be terminated at any time to yield the best result found so far. A key subroutine is a method to compute the minimum number of constraint violations to each node when restricted to paths on the roadmap. This requires solving the analogous *discrete MCR* problem. I demonstrate several new theoretical results for discrete MCR, including a proof that it is NP-hard by reduction from SET-COVER. I also present two search techniques that work well empirically.

The resulting planner is demonstrated to solve MCR problems in spaces up to 7D with hundreds of constraints. The capability to solve MCR problems appears to be valuable for several applications in robotics, and here we demonstrate its use on the problems of generating human-interpretable explanations of infeasibility, motion planning under environmental uncertainty, and backwards reasoning for manipulation planning.

## 2 The Minimum Constraint Removal Problem

For robotics applications we are interested in solving the following problem:

**Continuous MCR**

**Input.** connected $d$-dimensional configuration space $\mathscr{C} \subseteq \mathbb{R}^d$, $n$ obstacle regions $O_1, \ldots, O_n \subseteq \mathscr{C}$ which are taken to be open sets, and endpoints $q_s, q_g \in \mathscr{C}$.

**Definition.** The *cover* $C(q) : \mathscr{C} \to 2^{\{1,\ldots,n\}}$ determines the set of obstacles violated at configuration $q$: $C(q) = \{i \mid q \in O_i\}$. Likewise define $C(A)$ for any subset $A \subseteq C$ to be the union of $C(q)$ over all points $q \in A$, or equivalently $C(A) = \{i \mid A \cap O_i \neq \emptyset\}$.

**Definition.** For a subset $S \subseteq \{1, \ldots, n\}$, we say that $q'$ is *S-reachable* from $q$ if there exists a continuous path $y(s) : [0,1] \to \mathscr{C}$ starting at $q$ and ending at $q'$ that satisfies $C(y(s)) \subseteq S$ for all $s \in [0,1]$.

**Output.** A *minimum constraint removal* (MCR) $S^\star$ is the cover of a path between $q_s$ and $q_g$ that attains minimum size.

Equivalently, the MCR is the (not necessarily unique) smallest set such that $q_g$ is $S^\star$-reachable from $q_s$. Often we are also interested in producing the connecting path $y$ that achieves the cover $C(y) = S^\star$, which is the *witness* to $S^\star$. We also say that $q'$ is *k-reachable* from $q$ if $q'$ is $S$-reachable for some subset $S$ of size $k$.

MCR generalizes the classical Piano Mover's Problem [13], because $S^\star = \emptyset$ iff there exists a feasible path. Otherwise, $S^\star$ "explains" the infeasibility optimally in the following way. If the constraints indexed by $S^\star$ are removed from the original problem, then $q_s$ can be connected to $q_g$ with a feasible path, and there is no way to connect the two without removing fewer than $|S^\star|$ constraints.

If a planner computes a partition of $\mathscr{C}$, the connectivity among those regions forms a discrete graph. If each element in this partition is connected and has uniform

cover, then Continuous MCR is successfully reduced to a discrete MCR problem on the graph. Such decompositions may be produced in theory, for example, by computing the partition of $\mathscr{C}$ induced by the obstacle boundaries. Although there are methods from computational geometry to compute partitions for lines, planes, and spheres, such decompositions are intractable to compute in general for complex obstacles in high dimensional spaces. Nevertheless Discrete MCR is an essential subproblem for our sample-based approach. It is specified as follows:

**Discrete MCR**

**Input.** graph $G = (V,E)$, cover function $C[v]$, start and terminal vertices $s,t \in V$. $C[v]$ marks each vertex $v$ with a subset of $\{1,\ldots,n\}$ denoting which constraints are violated at $v$.

**Output.** A subset $S^\star$ of $\{1,\ldots,n\}$ of minimum size such that there exists a path $P$ in $G$ from $s$ to $t$ for which $C[v] \subseteq S^\star$ for all vertices $v \in P$.

The definitions provided above for Continuous MCR extend directly to the analogous concepts for Discrete MCR.

## 3 Analysis and Algorithms for Discrete MCR

Before proceeding to an algorithm for the continuous case, we will first prove some theoretical results about the MCR problem on graphs. Two practical search algorithms will be presented: one greedy and one optimal. Both of these will be used later in our planner to solve MCR on roadmap discretizations of the continuous space.

### *3.1 Discrete MCR is NP-Complete*

This section proves the following theorem.

**Theorem 1.** *The decision version of MCR (i.e., "is there a set S of k constraints such that t is S-reachable from s?") is NP complete.*

The proof that MCR is in NP is simple. Given a certificate in the form of an explanation $S$, $S$-reachability can be computed in polynomial time by selecting the subgraph of vertices $v$ with $C[v] \subseteq S$ and computing the shortest path from $s$ to $t$. To prove NP-hardness we perform a polynomial-time reduction from SET-COVER.

The input to SET-COVER is a universe $U = \{1,\ldots,m\}$ and a family $F = \{S_1,\ldots,S_n\}$ of subsets of $U$. A cover is a subset of $F$ whose union covers $U$. SET-COVER asks whether there exists a cover that contains at most $k$ elements of $F$. An instance of SET-COVER can be reduced to an instance of MCR on a graph constructed as follows (Figure 2). Let $V' = \{(e,S) \text{ for all } e \in U \text{ and } S \in F \mid e \in S\}$ be vertices indicating element-subset pairs for which the subset covers the element.

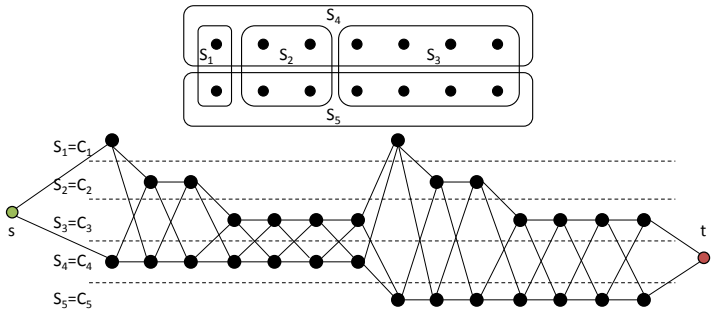


**Fig. 2** The reduction from SET-COVER to discrete MCR for a 14-element, 5-set example problem. A graph is constructed so that the optimal set cover $\{S_4, S_5\}$ corresponds precisely to the MCR $\{C_4, C_5\}$.

Let $E' = \{(e,S) \rightarrow (e',S') \text{ for all } ((e,S),(e',S')) \in V' \times V' \mid e' = e+1\}$ connect all vertices with numerically subsequent elements. Finally, construct $G = (V,E)$ by augmenting $(V',E')$ with starting node $s$ and terminal node $t$, where $s$ is connected to all vertices with $e = 1$, and $t$ is connected to all vertices with $e = m$. Clearly the size of this graph is polynomial in $m$ and $n$. Now construct obstacles $C[(e,S_i)] = \{i\}$ such that each vertex in $V'$ is covered by exactly the index of its subset, and let $C[s] = C[t] = \{\}$. Note that any path in $G$ from $s$ to $t$ passes exactly once through all elements of $U$, and the union of $C[v]$ for all $v$ along this path is a cover of $U$. Hence if $MCR(G,C,s,t,k)$ is true, then there exists an explanation $S$ of size $k$ that corresponds to a cover of size $k$. QED.

A consequence of this reduction is that the optimization version of $MCR(G,C,s,t)$ cannot be approximated in polynomial time within a factor of $1/4\log_2 n$ unless NP is in $DTIME(n^{\text{poly } \log n})$ (Lund and Yannakakis, 1994 [10]). The reduction does not apply in the converse, because solutions to SET-COVER cannot be easily converted into solutions for MCR. Although SET-COVER has a greedy polynomial time approximation with $O(\log n)$ error [15], we suspect that MCR is actually harder to approximate. In Section 3.4 we will show that the natural greedy MCR algorithm has worst case $O(n)$ error.

## *3.2 Optimal Search Algorithm*

Best-first search can be employed to solve Discrete MCR exactly. This formulation explores a state space in which states are vertex/subset pairs $(v,S_v)$ in which $S_v$ is the cover of some path $P$ leading from $s$ to $v$. Note that each vertex can be reached via many paths, each of which could be explained by a distinct subset, and hence a given vertex may potentially appear in $2^n$ search states. The search proceeds in best-first fashion in order of increasing $|S_v|$ from the initial state $(s,C[s])$, and generates children by enumerating edges $(v,S_v) \rightarrow (w,S_w)$ with $w \in Adj(v)$ and $S_w = C[w] \cup$

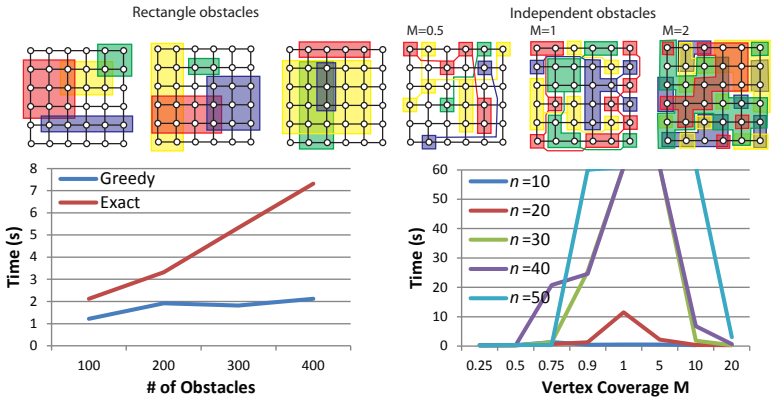


**Fig. 3** Left: running times for exact and greedy search with the on a $100\times100$ grid with $n$ random rectangular obstacles. Right: running times for exact search on a $10\times10\times10$ grid with $n$ total obstacles and each vertex is covered by $M$ obstacles chosen independently at random. (Greedy search solves each problem in less than 0.3 s).

$S_v$. It is complete and optimal because it maintains the following invariant: when a state $(v, S_v)$ is expanded at node $v$, $S_v$ is the minimum size cover over all paths from $s$ to $v$.

Revisited and suboptimal states are pruned according to the following definition:

**Definition.** An *irreducible constraint removal* (ICR) is a set $S$ such that $t$ is $S$-reachable from $s$ but not $S'$-reachable for any strict subset $S' \subset S$.

The search can safely prune non-ICR sets at a given vertex, i.e., a state $(v, S_v)$ can be pruned if we have previously visited a state $(v, S'_v)$ with $S'_v \subseteq S_v$. These pruned states do not affect the optimal explanation because that path to each descendant in the search tree under $(v, S_v)$ is covered by a set no smaller than the cover of the path under $(v, S'_v)$ that visits the same vertices.

In the worst-case the search generates $O(|G|2^{|S^\star|})$ states. However, in practice the pruning step eliminates a large number of states and hence the algorithm is practical for many MCR instances.

## *3.3 Empirical Results on Random Graphs*

We evaluated the optimal search on 2D and 3D grids of varying resolution and with varying numbers of random obstacles. It appears that the spatial coherence of obstacles is more relevant to performance than their number. We consider two random obstacle models (Figure 3, top):

1. Rectangles: rectangular obstacles are sampled randomly from the grid.
2. Independent Vertex: coverage sets $C[v]$ are drawn independently at random *per vertex*.

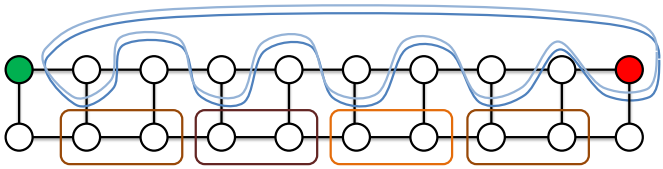

**Fig. 4** An example for which the greedy dicrete MCR algorithm achieves $O(n)$ error. The case where $n = 6$ is shown. To connect the start (left) and the goal (right), two overlapping obstacles block the top path, while $n-2$ disjoint obstacles block the bottom. Greedy search produces the suboptimal bottom path even though the top path is optimal. The example generalizes to larger $n$ by repeating the pattern on a wider graph.

For the Rectangles model, running times are roughly linear in the number of obstacles (Figure 3, left). The Independent Vertex model gives very different behavior. Running times are somewhat dependent on $n$, but curiously they rather rise dramatically for intermediate levels of vertex coverage (Figure 3, right). These obstacles realize the potential for worst-case exponential growth, as instances around $n = 30$ and vertex coverage 3 exhaust our test computer's 2Gb of RAM. Note that for these tests we implemented MCR in the interpreted Python language, which is orders of magnitude slower than the implementation used in our motion planner. Interestingly, as coverage grows higher, running time becomes more manageable. This type of transition from easy-to-hard-to-easy problem instances has been observed in many NP-complete problems [12].

The per-vertex random obstacle model is a pathological case that is atypical of problems encountered in practice. Yet because discrete MCR searches are used often in planning it may be prudent to avoid rare case of exponential growth at the cost of optimality. This motivates our development and analysis of a greedy MCR algorithm.

### *3.4 Greedy Search*

Greedy search is quite similar to the exact search but it prunes more aggressively. At each vertex it keeps only the subset $S_v$ with minimum size and prunes any states $(v, S'_v)$ with $|S'_v| \geq |S_v|$. Because it is guaranteed to only expand each node once, greedy search runs in $O(|G|n)$ time. A drawback is that the approximation factor can be arbitrarily bad. A counterexample, shown in Figure 4, demonstrates the possibility of achieving $O(n)$ error.

In experiments, greedy search is very fast and can solve much larger problems than the exact search. Problems with $|S^\star|$, $n$, and $|V|$ on the order of thousands, and even the pathological cases of Figure 3 are solved in fractions of a second. Perhaps surprisingly, it also produces high-quality paths in practice. In our tests on Rectangles problems it seems to have 0 approximation error, and in Independent Vertex problems its error was at most 2. This suggests that greedy search is not likely

to introduce severe approximation errors in practice, particularly on the spatially coherent obstacles typical of continuous MCR problems.

The next theorem helps explain why the greedy algorithm performs well in practice. First consider the following lemma.

**Lemma 1.** *Let $P = (v_0, \ldots, v_t)$ be any path in $G$ starting at $v_0 = s$. Then the size of the subset at $v_t$ computed by the greedy algorithm is no more than*

$$|C[v_0]| + \sum_{i=1}^{t} |C[v_i] \setminus C[v_{i-1}]|. \tag{1}$$

**Proof.** Let $S_0, \ldots, S_t$ denote the subsets obtained at $v_0, \ldots, v_t$ by greedy search. Define the partial sums $k_0 = |C[v_0]|, k_1 = k_0 + |C[v_1] \setminus C[v_0]|, \ldots, k_t = k_{t-1} + |C[v_t] \setminus C[v_{t-1}]|$.

We will prove that $|S_i| \leq k_i$ by induction on $i$. The base case with $i = 0$ is true by definition. Now consider the subset $S_{i-1}$ reached at $v_{i-1}$ and make the inductive assumption $|S_{i-1}| \leq k_{i-1}$. Because $S_{i-1}$ also contains $C[v_{i-1}]$, the search would add precisely $|C[v_i] \setminus C[v_{i-1}]|$ elements to $S_{i-1}$ if it took the candidate edge from $(v_{i-1}, S_{i-1})$ to $(v_i, S_i)$. The greedy search has the option of traversing the edge $v_{i-1} \to v_i$, and will not choose an edge into $v_i$ that produces a larger subset. So, we have that $|S_i| \leq |S_{i-1}| + |C[v_i] \setminus C[v_{i-1}]| \leq k_{i-1} + |C[v_i] \setminus C[v_{i-1}] = k_i$ as desired. By induction this inequality holds for all $i$. QED

**Theorem 2.** *The solution computed by the greedy algorithm is optimal if there exists an s-t path $P$ with cover $S^\star$ such that each obstacle in $S^\star$ enters into $P$ at most once. Here, "entering" means that for any $i \in S^\star$, the set of vertices along $P$ for which $i$ lies in their cover form a connected subsequence.*

**Proof.** Number the vertices of such a path $P = (v_0, \ldots, v_t)$. The size of the covers of each prefix of $P$ form a nondecreasing sequence $(k_0, \ldots, k_t)$ for which $k_0 = |C[s]|$ and $k_t = |S^\star|$. The single entry assumption shows that $k_{i+1} - k_i = |C[v_{i+1}] \setminus C[v_i]|$. Now consider the $S_t$ obtained at $v_t$ by greedy search. By the lemma, $|S_t| \leq |C[v_0]| + \sum_{i=1}^{t} |C[v_i] \setminus C[v_{i-1}]| = k_t = |S^\star|$. Since $|S^\star|$ is optimal, $|S_t| \geq |S^\star|$, and hence $|S_t| = |S^\star|$ as desired. QED.

A similar line of reasoning leads to the conclusion that the approximation error of the greedy algorithm is bounded by the minimal number of times that obstacles must be reentered. Given the difficulty involved in constructing problem instances in which this number is high, it would stand to reason that the greedy algorithm almost always has low error in nonadversarial settings.

## 4 Continuous MCR Motion Planner

Let us now return to the continuous setting. Our motion planner grows a probabilistic roadmap (PRM) that progressively approximates the true connectivity of the obstacle partition. As the approximation improves, a discrete MCR query restricted

to the roadmap will approach the true MCR. The resulting algorithm is any-time, in that it can be queried at any time to produce an increasingly accurate series of MCR estimates. We also introduce exploration strategies that help improve the planner's convergence rate.

## *4.1 Summary*

The algorithm maintains a probabilistic roadmap $G$, which is a network of configurations (known as *milestones*) in the configuration space that are connected by straight-line paths. The planner is specialized to a problem instance by providing by two problem-specific subroutines: Cover$(q)$, which computes the set of constraints violated at configuration $q$; and EdgeCover$(q,q')$, which computes the set of constraints violated along the straight-line path between $q$ and $q'$.

We are primarily concerned with how well reachability on the roadmap approximates the true reachability in the continuous space, so we define some new terminology. Given $G$ and an exploration limit $k$, we say that a node $q$ in $G$ is $(G,k)$-*reachable* if there exists a path in $G$ from $q_s$ to $q$ with a cover of size no more than $k$. A good exploration strategy constructs $G$ so that $k$-reachable nodes are likely to be $(G,k)$-reachable.

$(G,k)$-reachability is calculated using Discrete MCR search with the cover function taken to be $C[q] \equiv \text{Cover}(q)$. This works directly when each edge $q \to q'$ satisfies the condition $\text{EdgeCover}(q,q') = \text{Cover}(q) \cup \text{Cover}(q')$. For edges that violate this condition, we treat the edge as a "virtual node" to propagate edge-wise constraint violations appropriately.

Our planner grows $G$ by expanding from $(G,k)$-reachable nodes for some *exploration limit* $k$. In a manner reminiscent of the RRG planner [8], an RRT-like [9] strategy encourages fast exploration of the $(G,k)$ reachable space, while a PRM-like strategy connects new nodes to their nearby neighbors. These connections improve the connectivity of the roadmap and the quality of the $(G,k)$ approximation. The choice of the limit $k$ is another important facet of the exploration strategy. We vary $k$ during planning using a strategy governed by two principles. First, the cover of any given path to the goal is an upper bound on the true $|S^\star|$. So, $k$ should never be raised above $|S_{min}|-1$, where $S_{min}$ is the cover of the best path found so far. Second, it is more important to begin exploring at low $k$ because undersampling regions that are reachable with low $k$ will handicap the planners ability to identify regions that are reachable with high $k$. So, we use an *incremental raising* strategy.

## *4.2 Algorithm*

The planner takes as input a problem description and two parameters: $N_{raise}$, which dictates how many expansions are performed before raising $k$; and $\delta$, an RRT-like

step-size that governs the maximum length of edges in the roadmap. The algorithm is as follows:

**Continuous-MCR**:
1. $S_{min} \leftarrow$ EdgeCover$(q_s, q_g)$
2. $k \leftarrow |\text{Cover}(q_s) \cup \text{Cover}(q_g)|$
3. $G \equiv (V,E) \leftarrow (\{q_s, q_g\}, \{q_s \rightarrow q_g\})$
4. For $N = 1, 2, \ldots$ repeat:
5. Expand-Roadmap$(G,k)$
6. Compute the minimum explanations $S_G(q)$ for all $q \in V$
7. $S_{min} \leftarrow \arg\min_{S \in S_G(q_g)} |S|$
8. Every $N_{raise}$ steps, raise $k$.
9. If $k \geq |S_{min}|$, set $k \leftarrow |S_{min}| - 1$.

Lines 1-3 initialize the roadmap $G$ to a single edge from $q_s$ to $q_g$, and sets the exploration limit $k$ to a known lower bound, which is the union of constraints violated at the start and goal. Line 5 expands the roadmap $G$ using random sampling, and respects the exploration limit. Line 6 computes for each milestone $q$ the minimum explanation sets $S_G(q)$ from $q_s$ to $q$, restricted to edges of the roadmap $G$. Here we have the option of using the exact Discrete-MCR search, in which will $S_G(q)$ is set of one or more irreducible explanation sets; or using the greedy search, in which $S_G(q)$ will just consist of a single explanation set that may not be minimal. Line 7 updates the best explanation, and lines 8-9 update the exploration limit. The operation of Continuous MCR is illustrated in Figure 5.

The Expand-Roadmap procedure grows the roadmap while limiting the domain of exploration such that each added node is guaranteed to be $(G,k)$-reachable. It operates very much like RRG in that it expands the roadmap toward a configuration drawn at random from $\mathscr{C}$, but then it also adds connections to neighbors as well.

**Expand-Roadmap**(G,k)
1. $q_d \leftarrow$Sample()
2. Let $q_n \leftarrow$ Closest$(G,k,q_d)$
3. $q \leftarrow$ Extend-Toward$(q_n, q_d, \delta, k)$
4. Let $\{q_1, \ldots, q_m\} \leftarrow$ Neighbors$(G,q)$
5. For $i = 1, \ldots, m$, do:
6. If $d(q_i, q) < \delta$, then add $q_i \rightarrow q$ to $E$

It operates by generating a random sample $q_d$ (Line 1) and extends an edge from the closest $(G,k)$-reachable milestone toward $q_d$ (Lines 2–3). The resulting leaf node $q$ is then connected to nearby milestones in the roadmap (Lines 4-6). Each new extension and connection is limited to the maximum step $\delta$.

Expand-Roadmap uses several subroutines, which are implemented as follows.

**Closest.** Closest$(G,k,q)$ finds the closest $(G,k)$-reachable node to $q$ according to the distance metric $d(q,q')$.

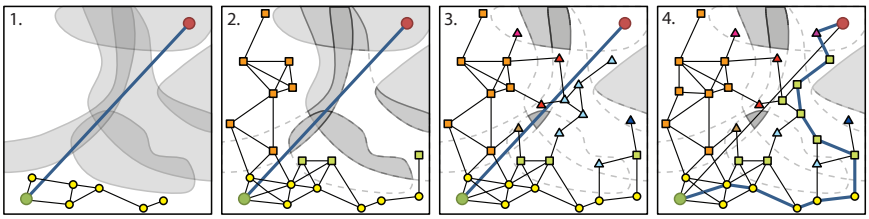


**Fig. 5** Illustrating the planner on the example of Figure 1. Circles, squares, and triangles indicate $(G,0)$-, $(G,1)$-, and $(G,2)$-reachable nodes, respectively. 1) The best path is initialized to a straight line, setting $|S_{min}| = 4$. Exploration begins in the feasible subset of the C-space. 2) Exploration after one constraint violation $k = 1$ is allowed. 3) $k$ is eventually raised to 2. A connection between the upper right milestone and the goal is not permitted because such a path would accumulate 3 violations. 4) After further roadmap refinement at $k = 2$, an alternate cover-1 path to the upper right milestone is found, which can then be connected to the goal via a cover-2 path.

**Neighbors.** Neighbors$(G,q)$ returns a set of milestones $q_1,\ldots,q_m$ that are close to $q$. As usual in sample based planners, the notion of "close" is defined either by selecting all roadmap nodes within a ball of radius $r$ centered at $q$, or by picking the $m$ nearest neighbors in the roadmap. Our experiments use the latter approach with $m = 10$.

**Extend-Toward.** Extend-Toward$(q_i,q,\delta,k)$ extends the roadmap with a new edge from $q_i$ to a configuration $q'$ in the direction of $q$. Like RRT, the step is limited to some maximum value $\delta$ by taking $q' = q_i + \min(\frac{\delta}{d(q_i,q)},1)(q-q_i)$. We also ensure that for some explanation set $S \in S_G(q_i)$, the resulting path to $q'$ has no more than $k$ violations; that is, $|S \cup EdgeCover(q_i,q')| \leq k$. If this is not satisfied, then we set $q'$ to the midpoint of $q \rightarrow q'$ and repeat the test. This continues until an acceptable point is found or some fixed number of bisections is reached (4 in our implementation).

**Reducing Discrete MCR Overhead.** We reduce the number of nodes in the Discrete-MCR graph by clustering connected milestones in each region in the obstacle partition using a union-find data structure. The results of Discrete-MCR on the clustered roadmap are equivalent to the full roadmap but the graph is usually much smaller. Also, we observe that Extend-Toward operations simply add new leaf nodes to $G$ and induce no changes to $S_G$ at existing milestones, so, $S_G(q')$ can be computed quickly. Moreover, many new edges constructed in Line 6 do not affect $S_G$ because they do not induce a better path to the neighbors of $q'$. So, we test whether edges out of $q'$ improve $S_G$ at each of its neighbors. If none are improved, then we avoid recalculating $S_G$. Finally, we reduce overhead even further by recalculating only local modifications to $S_G$ in the manner of dynamic shortest paths algorithms [4]. In our experiments these reduce the overhead of Discrete-MCR updates to be less than 5% of overall running time amongst our example problems.

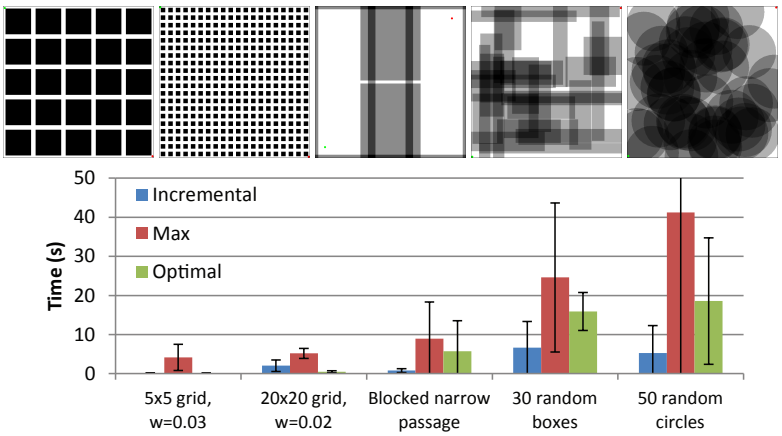


**Fig. 6** Comparing the performance of strategies for choosing the exploration limit on five example problems (top row). Average time and standard deviation to find an optimal MCR over 10 runs is plotted for an incremental raising strategy with $N_{raise} = 1000$ (Incremental), a fixed limit equal at $|S_{min}| - 1$ (Max), and a fixed limit equal to the size of the optimal MCR $|S^\star|$ (Optimal). MCRs for each problem have size 0, 0, 2, 9, and 12, respectively.

## *4.3 Rate of Convergence to the Minimum*

The MCR planner will produce a true MCR $S^\star$ as long as its roadmap contains a path that passes through $\mathscr{C} \setminus \bigcup_{i \in S^\star} O_i$. This section applies a theoretical result about traditional PRMs to demonstrate that the MCR planner does indeed converge, and it compares strategies for improving the convergence rate.

Traditional sample-based planners operate in the free space $\mathscr{F}$ — the complement of the obstacle region. Given certain *visibility* characteristics of $\mathscr{F}$, the probability of failing to connect two points that lie in the same connected component of free space is bounded by an exponentially decreasing curve [6, 9]. Let us denote the domain $\mathscr{C} \setminus \bigcup_{i \notin S^\star} O_i$ as $\mathscr{F}^\star$. Observe that once $k \geq |S^\star|$, our planner constructs a graph with at least as many milestones and edges as an RRT restricted to $\mathscr{F}^\star$. Hence the probabilistic completeness of RRT [9] extends to our planner as follows:

**Theorem 3.** *If there exists a path in $\mathscr{F}^\star = \mathscr{C} \setminus \bigcup_{i \notin S^\star} O_i$ between $q_s$ and $q_g$ with clearance at least $\delta > 0$, then MCR is probabilistically complete.*

But observe that our planner generates many milestones that do not lie in $\mathscr{F}^\star$, and do not contribute to the ultimate solution. So, a good expansion strategy should limit the sampling density to regions that are likely to be candidates for an optimal $\mathscr{F}^\star$. Of course, the planner does not know the shape of $\mathscr{F}^\star$ because $|S^\star|$ is unknown, and furthermore cannot even test whether a point lies within it. This motivates our choice for raising the expansion limit $k$ incrementally. If $k$ is set too low, then parts of $\mathscr{F}^\star$ will remain completely unexplored, but if it is too high, then $\mathscr{F}^\star$ will be a small fraction of the explored space.

We tested the effects of the exploration limit on several benchmark problems. Figure 6 shows results. Setting $k$ to the size of the optimal MCR $|S^\star|$ (Optimal) is certainly better than simply keeping $k$ one below the size of the current best cover (Max). But it is surprising that incrementally raising $k$ performs many times better than the Optimal strategy on the infeasible problems 3–5. This suggests that while certainly the volume of $\mathscr{F}^\star$ is an important factor, at least two other factors are at play:

- Better roadmaps in regions reachable under low $k$ reduce the errors of $(G,k)$-reachability estimates, which subsequently leads to better roadmaps in regions reachable with high $k$.
- The overhead involved in calculating $S_G$ is proportional to $k$.

The parameter $N_{raise}$ must be tuned to achieve a good schedule of exploration limit raises. If there is prior knowledge that $S^\star$ is small, then $N_{raise}$ should be large to prevent overexploration. For example, on the feasible benchmark problems #1 and #2, the Incremental strategy performs better as $N_{raise}$ grows larger. If on the other hand $S^\star$ is large, then $N_{raise}$ should be smaller to explore more quickly. However, it should not be too small because the benefits of exploring low $k$ regions will be missed. For all of our examples $N_{raise} = 1000$ seemed to be an acceptable compromise.

MCR appears to have a roughly constant factor overhead above traditional sample-based motion planners in feasible problems. In the feasible problems #1 and #2, the MCR planner with the Optimal strategy was approximately 3x and 5x slower than a standard RRT, while the incremental strategy was approximately 4x and 15x slower. Similar overhead factors were observed across several problem variations. We did observe that in a small fraction of runs the planner spends a long time updating $S_G$ when exact discrete MCR search is used. For this reason we typically use greedy search to reduce extreme variations in running time. Over hundreds of planner runs over dozens of benchmark problem variants we have only observed one instance in which the greedy search converged to a suboptimal MCR, and it only overestimated the MCR by 1.

## 5 Applications

**Parsimonious Excuse-Making.** If a robot were to succeed at a task *but for* some subset of its geometric, dynamic, and operational constraints, then the existence of those constraints can be interpreted as an explanation for failure. Clearly, small, parsimonious explanations of failure are more easily interpretable by a human. MCR, as we have formulated it, solves this problem for kinematic planning problems under point-wise constraints. Collision avoidance, joint limits, static balance, and static actuator limits fall in this category. Extending MCR to handle kinodynamic planning, differential constraints, and resource constraints would be interesting topics for future work.

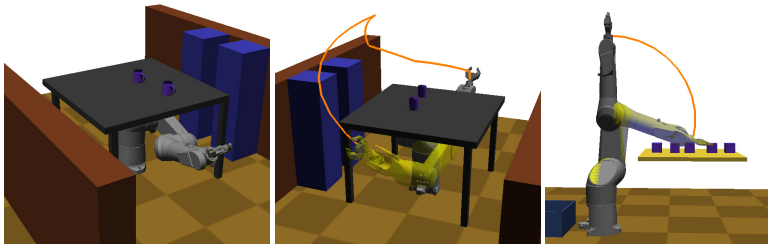

**Fig. 7** Left: a robot arm in a highly cluttered environment is asked to reach a goal configuration (Middle, foreground). The MCR states that the goal could be reached but for collision with the table. Right: for simple failures at the query endpoints, MCR terminates with the exact solution almost immediately. End effector trajectories for witness paths are drawn in orange.

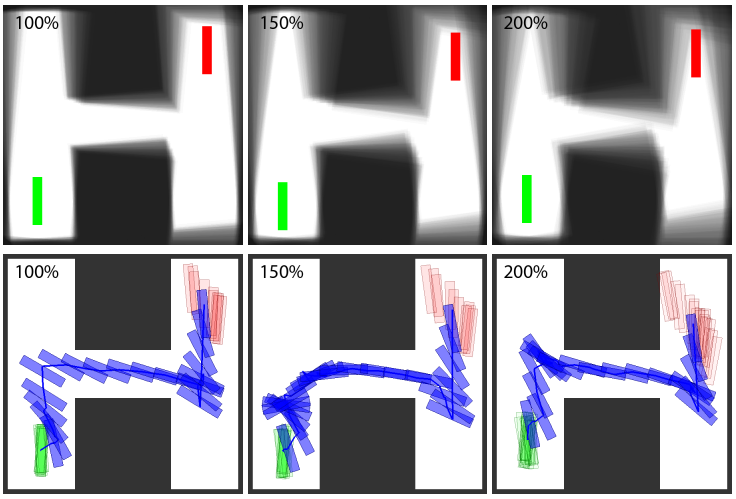


**Fig. 8** A bar must move through the corridor from the lower left to upper right of the environment, without sensing feedback. 100 hypothetical samples indicate uncertainty in the world's position and orientation relative to the robot. At the original level of uncertainty (100%), there exists a feasible open-loop motion that brings the bar from the start to the goal. With increased uncertainty (150%) the planner finds a path that collides in 4/100 samples. At 200% uncertainty the optimized path collides in 36/100 samples.

Figure 7 illustrates an application to excuse-making for a 6DOF robot arm with a 1DOF gripper in a cluttered environment. 99 collision, 55 self-collision, and 7 joint limit constraints are modeled as configuration space obstacles. An MCR of size 1 was computed in 11 s which states that at a minimum the arm must collide with the tabletop for a feasible path to exist. Such an explanation may be reported as "the table is in the way". One common cause of failure is when the start or goal configuration is infeasible. MCR performs very well here; it terminates almost immediately because the upper and lower bounds of the exploration limit are found to be equal.

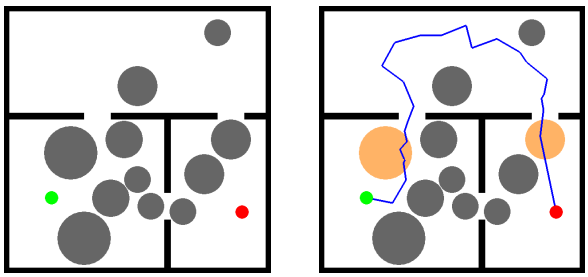

**Fig. 9** A circular robot must move some obtacles (circles) out of the way in order to move from the lower right room to the lower left room. A naive direct plan would need to move three obstacles. The MCR planner identifies a path through the third room that requires moving only two obstacles. This path was computed in 2.4 s.

**Planning Under Environmental Uncertainty.** Environmental uncertainty is commonly represented with a set of sampled *particles* that represent hypothetical placements of objects and obstacles in the world. Localization uncertainty can be represented in a similar manner: by fixing the reference frame relative to the robot, the uncertainty is captured in the relative transformation of the world with respect to the robot. Each of these particles can be considered as an instantiation of a C-space obstacle. Applied to this setting, MCR finds the path that *minimizes probability of collision*. Huang and Gupta (2009) considered a related problem of chance-constrained optimal planning, and they present an approximate planner for computing the minimum length path on a given roadmap that exceeds a collision probability threshold [7].

We implemented the minimum probability of collision approach for a simple planar robot that can translate and rotate, and is subject to localization uncertainty (Figure 8). The uncertain start location is represented by 100 particles sampled from a uniform distribution over a box in the $(x, y, \theta)$ space. The goal of the robot is to move to the upper right corner of the environment along an open loop path while minimizing the probability of collision. We then ran MCR for 10,000 iterations to generate the optimized paths in Figure 8. On the original feasible problem (100%) MCR converged in 7 s. With higher uncertainty (150% and 200%), MCR spent 68 s and 104 s respectively before progress stalled (as judged when 2,000 iterations passed without finding a better path).

**Backward Reasoning for Manipulation Planning.** Finally, we consider an application to navigation amongst movable obstacles and manipulation planning. Several authors, including Stilman and Kuffner (2005) and Dogar and Srinivasa (2010) consider *backwards reasoning* techniques for planning sequences of manipulation actions in the presence of cluttered movable obstacles [3, 14]. The common strategy is to compute a path to the goal in the absence of movable obstacles and then use the robot's swept volume along this path to determine a set of objects to move. This strategy, however, is only a heuristic and may result in unnecessarily large sets. MCR may lead to more efficient plans that disturb fewer objects (Figure 9).

## 6 Future Work

Ongoing work is investigating a variety of extensions to the basic MCR problem, including non-unit obstacle costs, goal regions instead of goal configurations, and optimizing both path costs and constraint removal costs. These seem to require only modest implementation change; for example, the latter extension might use techniques from the recent PRM*/ RRG* motion planners [8]. On a more theoretical note, upper bounds on an MCR are easy to obtain but lower bounds are not. It may be possible to use a disconnection prover [1] to provide such bounds and provide a "warm start" to the solver. Finally, for manipulation tasks it may be useful to study a *minimum constraint displacement* problem in which the planner moves constraints as little as possible in order to yield a feasible path.

## References

1. J. Basch, L. Guibas, D. Hsu, and A. T. Nguyen. Disconnection proofs for motion planning. In *IEEE Int. Conf. Rob. Aut.*, pages 1765–1772, Seoul, Korea, 2001.
2. T. Bretl, S. Lall, J.-C. Latombe, and S. Rock. Multi-step motion planning for free-climbing robots. In *Workshop on the Algorithmic Foundations of Robotics*, Zeist, Netherlands, 2004.
3. M. R. Dogar and S. S. Srinivasa. A framework for push-grasping in clutter. In *Robotics: Science and Systems*, 2011.
4. D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms*, 34(2):251 – 281, 2000.
5. M. Göbelbecker, T. Keller, P. Eyerich, M. Brenner, and B. Nebel. Coming up with good excuses: What to do when no plan can be found. In *Int. Conf. on Automated Planning and Scheduling*, 2010.
6. D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *IEEE Int. Conf. Rob. Aut.*, pages 2219–2226, 1997.
7. Y. Huang and K. Gupta. Collision-probability constrained prm for a manipulator with base pose uncertainty. In *IEEE/RSJ Int. Conf. Intel. Rob. Sys.*, pages 1426–1432, 2009.
8. S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Robotics: Science and Systems (RSS)*, Zaragoza, Spain, 2010.
9. S. M. LaValle and J. J. Kuffner, Jr. Randomized kinodynamic planning. *Int. J. Rob. Res.*, 20(5):379 – 400, 2001.
10. C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. of the ACM*, 41(5):960–981, 1994.
11. Z. McCarthy, T. Bretl, and S. Hutchinson. Proving path non-existence using sampling and alpha shapes. In *IEEE Int. Conf. Rob. Aut.*, 2012.
12. P. Prosser. An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence*, 81(12):81 – 109, 1996.
13. J. H. Reif. Complexity of the mover's problem and generalizations. In *20th Annual IEEE Symposium on Foundations of Computer Science*, pages 421–427, San Juan, Puerto Rico, 1979.
14. M. Stilman and J. Kuffner. Navigation among movable obstacles: Real-time reasoning in complex environments. *International Journal of Humanoid Robotics*, 2(4), December 2005.
15. V. V. Vazirani. *Approximation Algorithms*. Springer-Verlag, 2001.
16. L. Zhang, Y. Kim, and D. Manocha. A simple path non-existence algorithm using c-obstacle query. In S. Akella, N. Amato, W. Huang, and B. Mishra, editors, *Algorithmic Foundation of Robotics VII*, volume 47 of *Springer Tracts in Advanced Robotics*, pages 269–284. Springer Berlin / Heidelberg, 2008.