Article

The International Journal of Robotics Research
2014, Vol 33(1) 5–17

DOI: 10.1177/0278364913507795
ijr.sagepub.com

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. It describes a probabilistic roadmap motion planner for MCR in continuous configuration spaces that 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. Two search algorithms are described 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. It is demonstrated on three example applications: generating human-interpretable excuses for failure, motion planning under uncertainty, and rearranging movable obstacles.

Keywords

Motion planning, computational complexity, configuration space, robot manipulators

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 a 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.
- 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 (Göbelbecker et al., 2010). A similar notion in the motion planning literature is the problem of finding *disconnection proofs* that certify that no path can be found (Basch et al., 2001; Bretl et al., 2004; Zhang et al., 2008; McCarthy et al., 2012). Proving infeasibility is computationally challenging and so prior techniques are effectively limited to low-dimensional (Zhang et al., 2008; McCarthy et al., 2012), geometrically simple (Basch et al., 2001), or algebraically simple (Bretl et al., 2004) settings. Also, disconnection proof approaches are not constructive in that they do not consider how to change a space in order to yield a feasible path.

This paper presents 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 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.

School of Informatics and Computing, Indiana University, USA

Corresponding author:
Kris Hauser, School of Informatics and Computing, Indiana University, Informatics East, Room 257, 919 E. 10th St, Bloomington, IN 47408-3912, USA.
Email: hauserk@indiana.edu

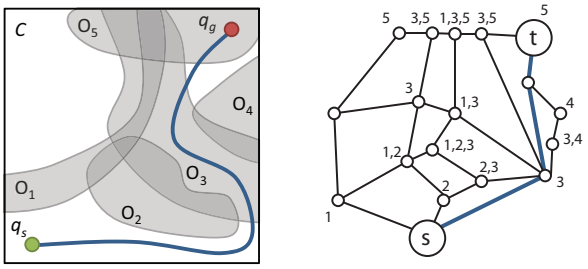

Fig. 1. MCR illustrated in a 2D configuration space. At least two obstacles (O_3 and O_5) need to be removed for a path to connect the start and goal. On the right, a partition of the continuous space yields a discrete MCR problem on a graph. Each node is labeled with the subset of obstacles that 'cover' 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).

We 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. We 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 feasible 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. This paper demonstrates several new theoretical results for discrete MCR, including a proof that it is NP-hard by reduction from SET-COVER. It also presents two search techniques that work well empirically in the vast majority of typical problem instances.

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 for 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:

Equivalently, the *minimum constraint removal* (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 $C(y) = S^\star$,

Continuous MCR

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

Definition. The *cover* $C(q)\colon \mathcal{C} \rightarrow 2^{\{1,\ldots,n\}}$ is defined as the set of obstacles violated at a configuration q: $C(q) = \{i \mid q \in O_i\}$. Likewise the cover $C(A)$ of a subset of configuration space $A \subseteq \mathcal{C}$ is defined as 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 $y(s)\colon [0,1] \rightarrow \mathcal{C}$ 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.

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 (Reif, 1979), because $S^\star = \emptyset$ if 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.

It is also a simple matter to consider *weighted* MCR problems with non-unit obstacle weights $w_1, \ldots, w_n$, and ask it to find a subset of minimum total weight rather than obstacle count. In this way the estimated cost of removing an obstacle (e.g. based on distance from the robot, size, or weight) can be incorporated into the optimization. For example, it is notationally convenient to treat static obstacles as having infinite weight rather than a separate forbidden region. Our software implements the weighted variant, but for simplicity, this paper will primarily discuss the unit-weight case. Results extend to the weighted case in a straightforward manner.

If a planner computes a *partition* of $\mathcal{C}$, the connectivity among those regions forms a discrete graph. By partition we mean an enumeration of each connected region with uniform obstacle cover – in other words, $\mathcal{C}$ is dissected along obstacle boundaries. With such a decomposition, the continuous MCR problem is successfully reduced to a discrete MCR problem on the graph. Although there are methods from computational geometry to compute partitions for collections of lines, planes, and spheres, partitions 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 exact. 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-hard

This section proves the following theorem, which implies that the optimization version of MCR is NP-hard.

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.*

Proof. 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-completeness 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. 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. □

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). 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 (Vazirani, 2001), we suspect that MCR is actually harder to approximate. In Section 3.4 we will show that the natural greedy MCR algorithm has the worst case $O(n)$ error.

3.2. Exact search algorithm

The 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 $S_w = C[w] \cup S_v$. 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 the search 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 exact search generates $O(|E|2^n)$ states. More precisely, for any vertex v reached by the search, the algorithm generates no more than

$$\binom{n}{\min(n/2, |S^\star|)} \tag{1}$$

irreducible covers of v. This is maximized with $|S^\star| \geq n/2$, and in this case Stirling's approximation shows that the

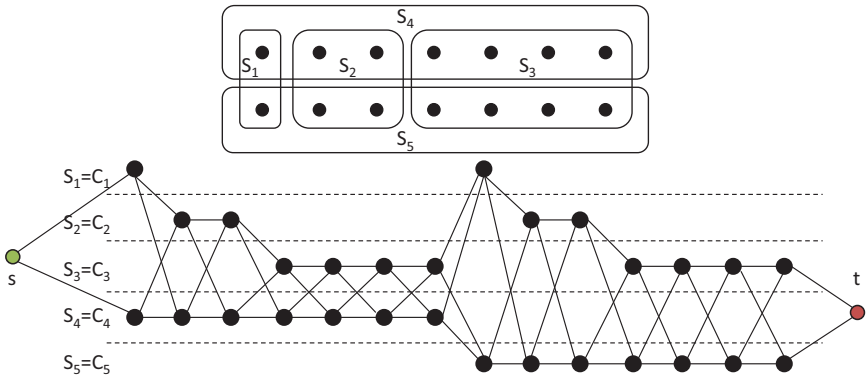

Fig. 2. The reduction from SET-COVER to discrete MCR for a 14-element, 5-set example problem. A graph is constructed on a grid so that the optimal set cover $\{S_4, S_5\}$ corresponds precisely to the MCR $\{C_4, C_5\}$. Each column of the grid corresponds to an element, while each row defines an obstacle corresponding to a set. A path through the graph is in one-to-one correspondence to a cover of all of the elements, and in this case the lower route is optimal.

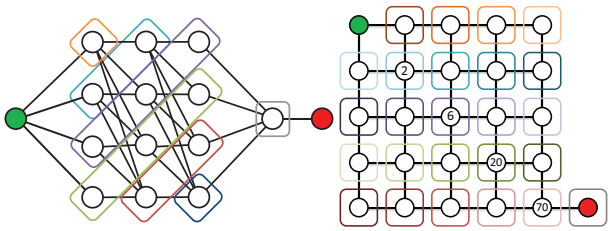

Fig. 3. Left: an example that achieves near worst-case performance for the exact search. The construction can be extended to n obstacles in a $(n+1)/2 \times (n-1)/2$ grid so that the search generates $\binom{n-1}{(n-1)/2}$ covers at the second-to-last vertex. Right: a less contrived example that generates $\binom{2\sqrt{n}-2}{\sqrt{n}-1}$ covers at the lower-right vertex. Diagonal vertices are marked by the number of times they are visited during the exact search.

binomial coefficient is bounded from below by a slightly sub-exponential number:

$$\binom{n}{n/2} \geq \frac{2^{n-1}}{\sqrt{n/2}}. \tag{2}$$

This bound can nearly be achieved using the example shown on the left in Figure 3. A less contrived scenario is the grid given on the right, in which $\binom{2\sqrt{n}-2}{\sqrt{n}-1}$ covers are generated at a single vertex. Even though this is sub-exponential, it is still intractable: in a 20×20 grid, this amounts to over 35 billion covers.

Despite this poor worst-case performance, in practice the pruning step eliminates a large number of states and hence the algorithm is practical for many MCR instances. The following section demonstrates that the *distribution* of obstacles is an essential factor in the running time of this algorithm.

3.3. Empirical results on random graphs

We evaluated the exact 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 4, 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*.

For the rectangles model, running times are roughly linear in the number of obstacles (Figure 4, 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 4, 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 (Prosser, 1996).

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 the 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 algorithm

A greedy search is quite similar to an 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, a greedy search runs in $O(|E|n)$ time. A drawback is that the approximation factor can be arbitrarily

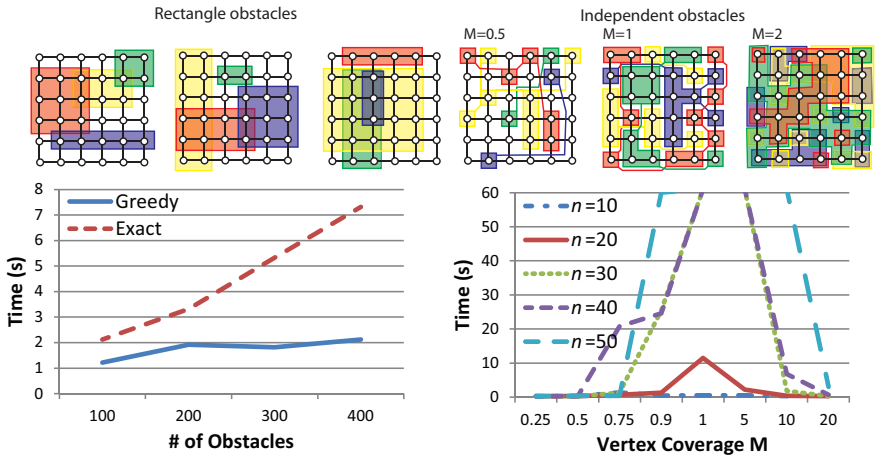

Fig. 4. The obstacle distribution dramatically affects the empirical performance of search. Left: running times for the exact and greedy search on a 100 × 100 grid with n random rectangular obstacles. Right: running times for the exact search on a 10 × 10 × 10 grid with n total obstacles and each vertex is covered by M obstacles chosen independently at random. The greedy search is not shown because it solves each problem in less than 0.3 s. (This figure is best viewed online in color.)

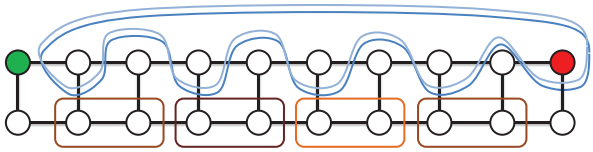

Fig. 5. An example for which the greedy discrete MCR algorithm achieves $O(n)$ error. The case where $n = 6$ is shown. To connect the start (left) and the goal (right), two overlapping obstacles block the top path, while $n - 2$ disjoint obstacles block the bottom. The greedy search produces the suboptimal bottom path even though the top path is optimal. The example generalizes to larger n by repeating the pattern on a wider graph.

bad. A counterexample, shown in Figure 5, demonstrates the possibility of achieving $O(n)$ error.

In experiments, the greedy search is very fast and can solve much larger problems than an exact search. Problems with $|S^\star|$, n, and $|V|$ on the order of thousands, and even the pathological cases of Figure 4 are solved in fractions of a second. Perhaps surprisingly, it also produces high-quality covers in practice. It computes an optimal solution on the examples of Figure 2 and Figure 3. In our tests in rectangles problems it seems to have 0 approximation error, and in independent vertex problems its error was at most 2. This suggests that the 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 search 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 search is no more than*

$$|C[v_0]| + \sum_{i=1}^{t} |C[v_i] \setminus C[v_{i-1}]|. \tag{3}$$

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

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. □

Theorem 2. *The solution computed by the greedy search 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 non-decreasing 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 a 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. □

A similar line of reasoning leads to the conclusion that the approximation error of the greedy search is bounded by the minimal number of times that obstacles must be reentered.

3.5. Complexity analysis for problem subclasses

Based on Theorem 2, one may speculate that there exist subclasses of MCR problems — for example, those in which obstacles are connected, convex, or take on particular shapes — that are solvable in polynomial time. Unfortunately, this section proves that this is not the case: continuous MCR is in NP even for relatively simple obstacle classes, and the only classes that we have so far been able to prove are in P are probably too simple to characterize problems of interest.

Theorem 3. *The following statements hold:*

1. *Discrete MCR is in P if removing t (or s) from G yields a forest.*
2. *Discrete MCR with disjoint and simply connected obstacles is in P.*
3. *Continuous MCR in Cartesian spaces with half plane obstacles is in P.*
4. *Discrete MCR with planar graphs is NP-hard.*
5. *Discrete MCR with simply connected obstacles is NP-hard.*
6. *Continuous MCR in Cartesian spaces of dimension 3 and higher is NP-hard, even when restricted to obstacles that are axis-aligned boxes.*

In a discrete problem, a *simply connected* obstacle O is one for which the subgraph induced by vertices with $O \in C[v]$ is connected and whose removal from G does not introduce new disconnected components.

Proof. Statement 1 holds because such problems can easily be solved by computing the covers of all paths from s to t (resp. t to s), which incurs $O(n|V|)$ time. Statement 2 holds because for such problems, optimal paths do not need to exit or enter an obstacle more than once, and hence the greedy search generates an optimal solution by Theorem 2. Statement 3 holds because a straight line path yields the optimal cover.

To prove Statement 4, we consider a variant of the proof of Theorem 1 via reduction from SET-COVER. As illustrated in Figure 6, we construct a graph that avoids crossing edges by adding an additional vertical stripe to the right of each element-vertex; this stripe allows free movement from any element-vertex to the next. By inspection the solution to this MCR problem is also a solution to SET-COVER.

We will now prove Statement 6, which will imply Statement 5. Given an instance of SET-COVER, we construct in polynomial time a three-dimensional continuous MCR problem with axis-aligned obstacles whose solution will be in one-to-one correspondence with the solution to SET-COVER (Figure 7).

Recall the definitions from the proof of Theorem 1. Now construct a $m \times n$ table in which elements of U are indexed in the columns, and the subsets in F are indexed in the rows. Mark entry (i, j) with a 1 if the element in column j appears in subset i. Now construct for each of the m subsets a *primary* box with width w, depth d/m, and height $h/2$, stacked together along the depth axis. Construct for each 0 entry a *blocking* box with width $w/2n$, depth d/m, and height $h/2$ in the appropriate cell. Below this whole construction, create n blocking boxes with width $w/2n$, height $h/2$ and spanning the full depth d. The non-blocked cells act as 'gates' so that a path from left to right can pass through a gate in row i and column j if and only if element i is part of subset j; doing so 'chooses' subset j to cover element i. Note that the stacked construction is necessary so that after passing a gate in column j, a path is given free choice of any gate in column $j + 1$. Without this construction, a move from one row to another would need to pass through each intermediate primary obstacle.

Finally, we set $\mathcal{C}$ to be a box of width $w + \epsilon$, depth d, and height h with space on either side of the table to hold the start and goal configurations. We must enforce that blocking boxes are never part of a solution, and can do so by setting the cost of the box to infinity, if weights are allowed, or by duplicating each blocking box. This construction is $O(mn)$ time. With this construction, a solution to the continuous MCR problem is a certificate S of k or fewer primary boxes whose removal admits a feasible connecting path. These boxes are in one-to-one correspondence with a solution to SET-COVER.

To prove Statement 5, observe that this continuous construction is in one-to-one correspondence with a discrete MCR problem on a 3D grid placed at the cell centers of this construction. Statement 4 holds because all obstacles induce connected subgraphs. □

Given the difficulty involved in constructing counterexamples that exhibit worst-case complexity, it would stand to reason that the greedy search usually has low error in non-adversarial settings. Our implementations provide the user with a choice between the exact and greedy algorithms: the greedy search can be used to prioritize consistently low running times, while the exact search can be used to prioritize accuracy.

4. Continuous MCR motion planner

Let us now return to the question of practical planning in the continuous setting. Our motion planner grows a 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

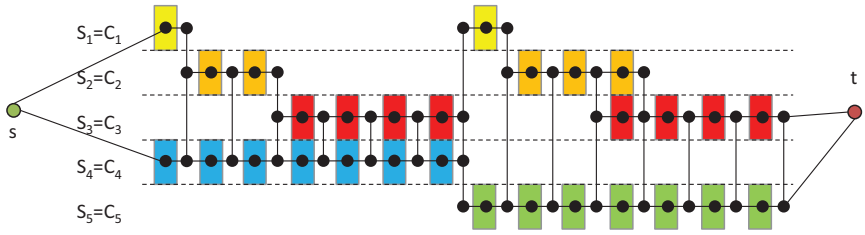

Fig. 6. Illustrating the SET-COVER reduction for planar MCR based on the example of Figure 2.

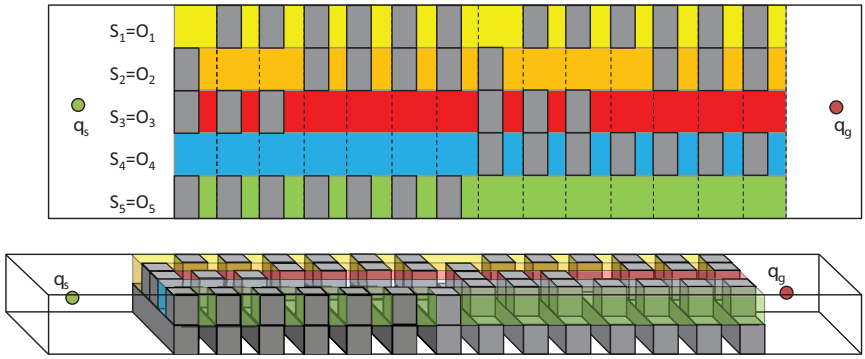

Fig. 7. Illustrating the SET-COVER reduction for continuous MCR in a 3D space with axis-aligned box obstacles.

by providing 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.

The (G, k)-reachability is calculated using a discrete MCR search with the cover function taken to be $C[q] \equiv$ Cover(q). This works directly when each edge $q \rightarrow q'$ satisfies the condition EdgeCover(q, q') = Cover(q) $\cup$ 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 (Karaman and Frazzoli, 2010), an RRT-like (LaValle and Kuffner, Jr., 2001) 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 δ, 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. $k \leftarrow |$Cover(q_s) $\cup$ 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 $S_G(q)$ 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 8.

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 $\mathcal{C}$, 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,δ,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 δ.

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 $d(q,q')$.

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,δ,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 δ 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

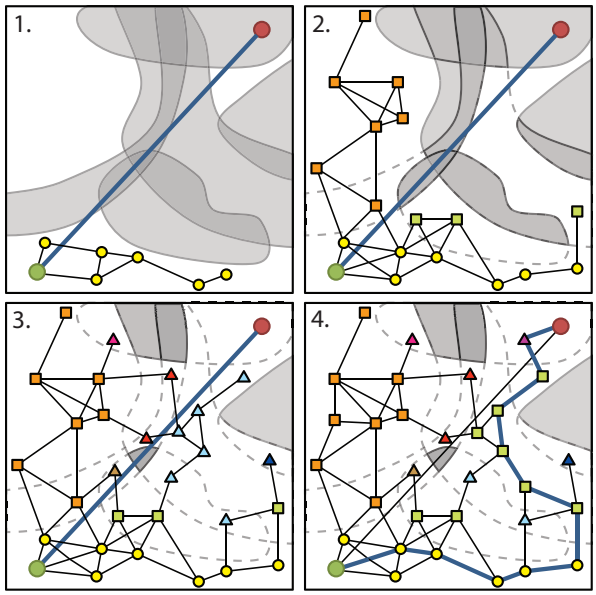

Fig. 8. Illustrating the planner on the example of Figure 1. Circles, squares, and triangles indicate $(G,0)$-, $(G,1)$-, and $(G,2)$-reachable nodes, respectively. (1) The best path is initialized to a straight line, setting $|S_{min}|=4$. Exploration begins in the feasible subset of the C-space. (2) 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.

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 (Frigioni et al., 2000). In our experiments these reduce the overhead of discrete MCR updates to less than 5% of the overall running time amongst our example problems.

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 $\mathcal{C} \setminus \bigcup_{i \in S^\star} O_i$. 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 $\mathcal{F}$ — the complement of the obstacle region. Given

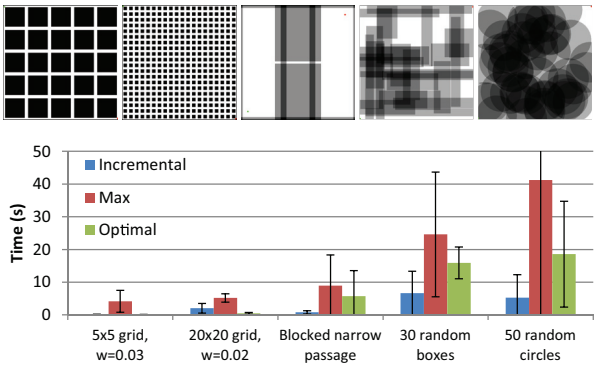

Fig. 9. Comparing the performance of strategies for choosing the exploration limit on five example problems (top row). Average time and standard deviation to find an optimal MCR over 10 runs is plotted for an incremental raising strategy with $N_{raise} = 1000$ (Incremental), a fixed limit equal at $|S_{min}| - 1$ (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.

certain *visibility* characteristics of $\mathcal{F}$, 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 (Hsu et al., 1997; LaValle and Kuffner, Jr., 2001). Let us denote the domain $\mathcal{C} \setminus \bigcup_{i \notin S^\star} O_i$ as $\mathcal{F}^\star$. 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 $\mathcal{F}^\star$. Hence the probabilistic completeness of RRT (LaValle and Kuffner, Jr., 2001) extends to our planner as follows:

Theorem 4. *If there exists a path in $\mathcal{F}^\star = \mathcal{C} \setminus \bigcup_{i \notin S^\star} O_i$ 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 $\mathcal{F}^\star$, 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 $\mathcal{F}^\star$. Of course, the planner does not know the shape of $\mathcal{F}^\star$ 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 $\mathcal{F}^\star$ will remain completely unexplored, but if it is too high, then $\mathcal{F}^\star$ will be a small fraction of the explored space.

We tested the effects of the exploration limit on several benchmark problems. Figure 9 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 $\mathcal{F}^\star$ 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 3 and 5 times slower than a standard RRT, while the incremental strategy was approximately 4 and 15 times 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 an exact discrete MCR search is used. For this reason we typically use a 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

5.1. 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.

Figure 10 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

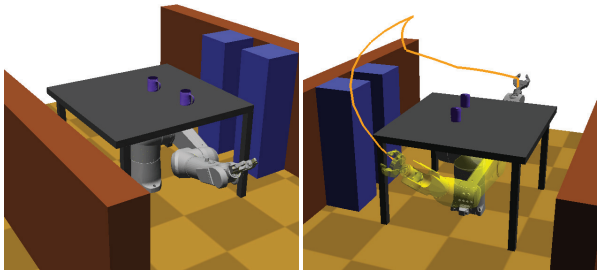

Fig. 10. A robot arm in a highly cluttered environment (left) is asked to reach a goal configuration (right, foreground). The MCR states that the goal could be reached but for collision with the table. The end effector trajectory for the witness path is drawn at right.

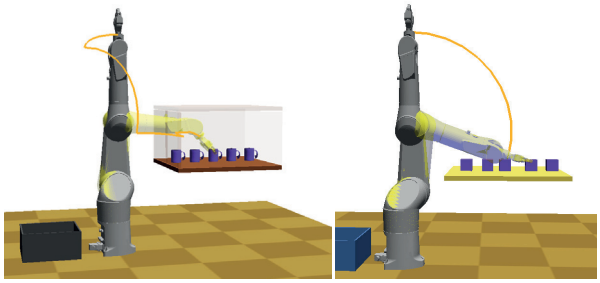

Fig. 11. Left: The planner computes that the two cupboard doors (drawn transparently) prevent reaching the cup. Right: for simple collisions at the query endpoints, MCR terminates with the exact solution almost immediately.

as 'the table is in the way'. Figure 11 illustrates two scenarios where the robot is asked to grasp an object in a cluttered environment. At the top, the robot does not know a priori that it must open the cupboard doors in order to reach the cup. The MCR planner computes a path that violates only those collision avoidance constraints in approximately 13 s on a 2.8GHz laptop PC. At the bottom, the robot's gripper interferes with other nearby objects. Here, planning is nearly instantaneous because the straight-line path between the start and goal collide with the same objects as the endpoints themselves (Figure 10, bottom). After a single edge collision check, the planner finds that the upper and lower bounds on the exploration limit are equal. Such cases are relatively common, for example, when grasping objects surrounded only locally by clutter.

5.2. 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

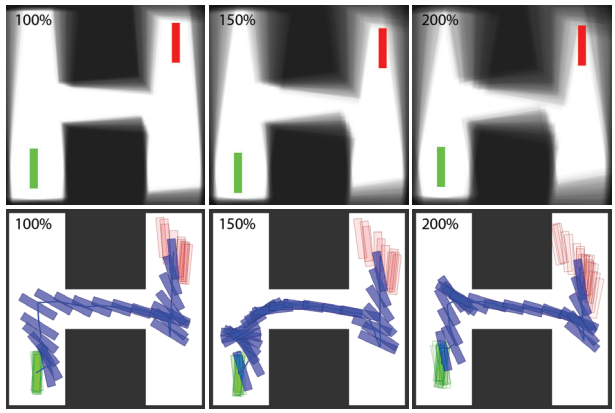

Fig. 12. A bar must move through the corridor from the lower left to upper right of the environment, without sensing feedback. 100 hypothetical samples indicate uncertainty in the world's position and orientation relative to the robot. At the original level of uncertainty (100%), there exists a feasible 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.

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.

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 12). The uncertain start location is represented by 100 particles sampled from a uniform distribution over a box in the (x, y, θ) 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 12. 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).

5.3. 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 (2011) consider *backwards reasoning* techniques for planning sequences of manipulation actions in the presence of cluttered movable obstacles. 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.

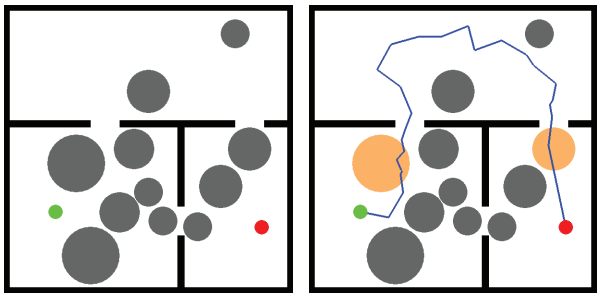

Fig. 13. A circular robot must move some obstacles (circles) out of the way in order to move from the lower right room to the lower left room. A naive direct plan would need to move three obstacles. The MCR planner identifies a path through the third room that requires moving only two obstacles. This path was computed in 2.4 s.

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 13 illustrates an example in which moving directly to the room on the right would require the robot to move three objects aside, while the detour through the upper room only requires moving two. Figure 14 illustrates a more complex example with 72 objects, where the planner incrementally improves the best set of objects found so far. Note however that MCR does not dictate where the obstacles should be moved aside in order to enable free passage. Nevertheless MCR is a useful first pass to a more sophisticated planner that does consider out-of-the-way locations. One possible implementation would call for a second planner that searches for out-of-the-way locations for each object, and if the planner fails, to re-run MCR with additional costs for removing those objects.

6. Extensions and open problems

A number of extensions and variants of the basic MCR problem appear to be useful for various robotics applications.

6.1. Multi-goal variants

An *all-pairs* variant would remove a subset of obstacles so that a collection of configurations are mutually reachable. This problem might have applications in modifying buildings or road networks for accessibility, or removing obstructions for robot swarms. An *any-goal* variant would find an MCR from the start to any configuration in some collection of goals, and would be useful for manipulation in clutter where an object can be grasped at multiple candidate configurations.

6.2. Path costs, dynamics, and obstacle movement

Another direction might optimize both path and constraint removal costs. One natural approach might integrate

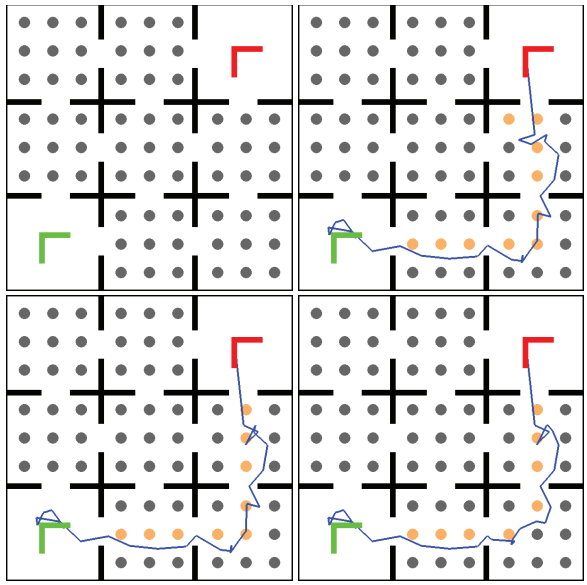

Fig. 14. Incremental improvement of an object removal set, while moving an L-shaped robot that can translate and rotate from the lower left to the upper right room. Circles are removable objects. After 74 s, a 10 object path is found. After 102 s, the planner yields a 9 object path, and after 198 s, the planner finds an 8 object path.

techniques from the recent PRM*/RRG* motion planners (Karaman and Frazzoli, 2010) into our roadmap planner. Dynamic motion constraints are another direction that can be easily taken into account by our method as long as a steering function — a method to construct a dynamically-feasible path between two given states — is available. Another formulation of dynamic constraints is to consider them as obstacles that can be removed in similar fashion to static obstacles, so that a result of a dynamic MCR query may be 'cannot brake fast enough to avoid collision'. Finally, it may be useful to study a *minimum constraint displacement* problem in which the planner moves obstacles as little as possible in order to yield a feasible path. This formulation would be particularly valuable for manipulation tasks.

6.3. Logical structure

MCR is a simple instance of a class of motion planning problems in which constraints are specified as logical statements involving obstacle membership. Different logical statements can be used to encode other interesting problems. For example, suppose that removing an obstacle has the effect of 'carving out' a section of the obstacle region, so that only one obstacle at a configuration needs to be removed in order to make it feasible. This *minimum freespace addition* problem can specify the minimum number of sensor readings in an unknown environment needed to navigate from point A to point B, and the logical constraint can be encoded as a disjunction rather than a conjunction.

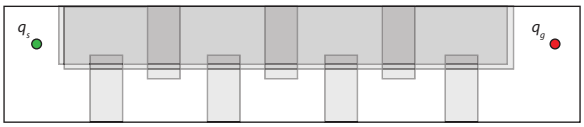

Fig. 15. A planar example with n connected rectangular obstacles in which the greedy search exhibits an error of $(n-1)/2-2$. The greedy solution passes through all lower obstacles while the optimal path weaves through the two long obstacles.

Another option is to encode *priority constraints* that enforce that region A must be touched before passing through region B. Such constraints could be useful to encode a switch that opens a door, or a region in which an agent can launch an attack on an adversary guarding a blocked region. Similar problems have been considered using task planning formulations (Plaku and Hager, 2010) and linear temporal logic specifications (Plaku, 2012). It is an open question whether the MCR computational complexity analysis and sampling strategies can be extended to this more general case.

6.4. Large obstacle counts

Although we have so far considered hundreds of obstacles, an interesting question is whether MCR methods can scale to massive numbers of obstacles. For example, obstacle grids, particularly in 3D, may consist of a thousands or even millions of cells. Finding the minimum number of cells to carve out may be an important problem in excavation, mining, and search and rescue in damaged buildings. It would also be useful in CAD/CAM to help redesign parts so that they are accessible during assembly and maintenance. With regularly-structured obstacles it may be possible to devise efficient methods with running time sub-linear in the number of obstacles, e.g., using hierarchical methods such as quadtrees/octrees.

6.5. Open theoretical questions

This work also raises a number of theoretical questions regarding MCR complexity. In particular, it would be useful to identify problem subclasses that have polynomial worst-case running time and that admit efficient approximation algorithms. It is interesting to note that the continuous MCR reduction relies on the use of the third dimension, which is reminiscent of shortest-paths problems among polyhedral obstacles, which are known to be in P in 2D but NP-hard in 3D (Canny, 1988). Since the initial writing of this paper, recent work (Erickson and LaValle, 2013) proved that MCR is NP-hard when obstacles are planar, polygonal, and simply connected, via reduction from Horn clause satisfiability. It would appear that NP-hardness is caused when a single obstacle is disconnected into $O(n)$ pieces or overlaps with $O(n)$ other obstacles, but MCR is in P when each obstacle is connected and non-overlapping. These observations suggest two open complexity questions that may be interesting to study in future work:

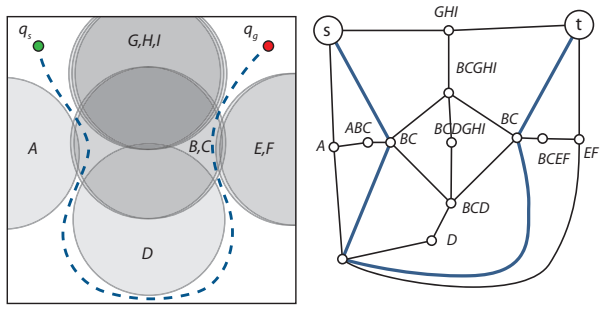

Fig. 16. A planar example with 9 circular obstacles of the same shape in which the greedy search exhibits an error of 1. The partition graph and optimal path is drawn on the right. The greedy search will compute a path to the lower left vertex through A, resulting in a suboptimal solution of size 3.

1. Is MCR in P on problems where obstacles have fixed shape and variable position?
2. Is MCR in P on problems in which each obstacle is connected and intersects no more than k obstacles (k held constant)?

It is also unknown whether better polynomial time approximation algorithms exist for certain problem subclasses. Figure 15 demonstrates that the greedy search has $O(n)$ error on the partition graph of a problem with simply-connected planar obstacles. Even more surprising is Figure 16, which demonstrates that the greedy search fails to generate an optimal path even when obstacles are circles of fixed radius. Note that this example can be repeated n times horizontally to achieve a $n/9$ error. Do polynomial-time algorithms with sub-linear error exist for these cases? The answer is currently unknown.

7. Conclusion

This paper presented a new MCR motion planning problem that asks for the minimum number of constraints that need to be removed for a feasible path to exist. The planner is applied to three problems: allowing robots to explain their failures, maximizing safety in uncertain environments, and selecting movable obstacles in cluttered environments. The planner uses a probabilistic roadmap method that builds a graph that increasingly approximates the connectivity of continuous obstacles as more samples are added. This paper analyzes the key subproblem of discrete MCR on a graph, and proves that it is NP-hard. Nevertheless, experimentation suggests that typical instances can be solved tractably: either using an exact algorithm that achieves typically low running time or a greedy algorithm that achieves typically low error.

This work also raises a number of open theoretical questions, as well as directions for extending the concept to

a wider variety of applications, such as multi-goal variants, incorporating movement costs and constraints, and incorporating logical and temporal relationships between active/inactive constraints. At a broader philosophical level, this work argues that there is great value in breaking open the classical 'black-box' collision test. Doing so will make it possible to exploit the inherent structure of constraints to achieve more reliable, more informative, and more versatile motion planning.

Acknowledgement

The author is grateful for discussions with several researchers at WAFR 2012 who suggested new applications and raised insightful questions about the tractability of MCR problem subclasses.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

References

Basch J, Guibas L, Hsu D and Nguyen AT (2001) Disconnection proofs for motion planning. In: *IEEE International conference on robotics and automation*, Seoul, Korea, pp. 1765–1772.

Bretl T, Lall S, Latombe JC and Rock S (2004) Multi-step motion planning for free-climbing robots. In: *Workshop on the Algorithmic Foundations of Robotics*, Zeist, the Netherlands.

Canny J (1988) *Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press.

Dogar MR and Srinivasa SS (2011) A framework for push–grasping in clutter. In: *Robotics: Science and Systems*.

Erickson L and LaValle S (2013) A simple, but np-hard, motion planning problem. In: *Proceedings of the twenty-seventh AAAI conference on artificial intelligence (AAAI-13)*.

Frigioni D, Marchetti-Spaccamela A and Nanni U (2000) Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms* 34(2): 251–281. DOI: 10.1006/jagm.1999.1048.

Göbelbecker M, Keller T, Eyerich P, Brenner M and Nebel B (2010) Coming up with good excuses: What to do when no plan can be found. In: *International conference on automated planning and scheduling*.

Hsu D, Latombe JC and Motwani R (1997) Path planning in expansive configuration spaces. In: *IEEE International conference on robotics and automation* pp. 2219–2226.

Huang Y and Gupta K (2009) Collision-probability constrained PRM for a manipulator with base pose uncertainty. In: *IEEE/RSJ International conference on intelligent robotic systems*, pp. 1426–1432.

Karaman S and Frazzoli E (2010) Incremental sampling-based algorithms for optimal motion planning. In: *Robotics: Science and Systems (RSS)*, Zaragoza, Spain.

LaValle SM and Kuffner, Jr JJ (2001) Randomized kinodynamic planning. *The International Journal of Robotics Research* 20: 379–400.

Lund C and Yannakakis M (1994) On the hardness of approximating minimization problems. *Journal of the ACM* 41: 960–981. DOI: 10.1145/185675.306789.

McCarthy Z, Bretl T and Hutchinson S (2012) Proving path non-existence using sampling and alpha shapes. In: *IEEE International conference on robotics and automation*.

Plaku E (2012) Planning robot motions to satisfy linear temporal logic, geometric, and differential constraints. In: *ICAPS 2012 Workshop on Combining Task and Motion Planning for Real-World Applications*, Sao Paolo, Brazil.

Plaku E and Hager G (2010) Sampling-based motion planning with symbolic, geometric, and differential constraints. In: *IEEE International conference on robotics and automation*, Anchorage, AK.

Prosser P (1996) An empirical study of phase transitions in binary constraint satisfaction problems. *Artificial Intelligence* 81: 81–109.

Reif JH (1979) Complexity of the mover's problem and generalizations. In: *20th annual IEEE symposium on foundations of computer science*, San Juan, Puerto Rico, pp. 421–427.

Stilman M and Kuffner J (2005) Navigation among movable obstacles: real-time reasoning in complex environments. *International Journal of Humanoid Robotics* 2: 479–504.

Vazirani VV (2001) *Approximation Algorithms*. Berlin: Springer-Verlag.

Zhang L, Kim Y and Manocha D (2008) A simple path non-existence algorithm using c-obstacle query. In: Akella S, Amato N, Huang W and Mishra B (eds.), *Algorithmic Foundation of Robotics VII* (*Springer Tracts in Advanced Robotics*, vol. 47). Berlin: Springer, pp. 269–284.