A characterization of the reconfiguration space of self-reconfiguring robotic systems

Robotica (2011) volume 29, pp. 73–85. © Cambridge University Press 2011
doi:10.1017/S0263574710000718

A characterization of the reconfiguration space of self-reconfiguring robotic systems

Tom Larkworthy* and Subramanian Ramamoorthy

Institute of Perception, Action and Behaviour, School of Informatics,University of Edinburgh, 10 Crichton Street, Edinburgh EH8 9AB email: tom.larkworthy@gmail.com (corresponding), s.ramamoorthy@ed.ac.uk

(Received in Final Form: October 28, 2010)

SUMMARY

Motion planning for self-reconfiguring robots can be made efficient by exploiting potential reductions to suitably large subspaces. However, there are no general techniques for identifying suitable restrictions that have a positive effect on planning efficiency. We present two approaches to understanding the structure that is required of the subspaces, which leads to improvement in efficiency of motion planning. This work is presented in the context of a specific motion planning procedure for a hexagonal metamorphic robot. First, we use ideas from spectral graph theory – empirically estimating the algebraic connectivity of the state space – to show that the HMR model is better structured than many alternative motion catalogs. Secondly, using ideas from graph minor theory, we show that the infinite sequence of subspaces generated by configurations containing increasing numbers of subunits is well ordered, indicative of regularity of the space as complexity increases. We hope that these principles could inform future algorithm design for many different types of self-reconfiguring robotics problems.

KEYWORDS: Self-reconfiguring robotics; Motion planning; Graph theory.

1. Introduction

Self-reconfiguring systems (SRSs) are robots comprised of a collection of robotic subunits that can physically connect and disconnect from one another. Through collaboration, the aggregate is capable of changing its morphology on demand. Such systems offer versatility unparalleled by monolithic robot solutions. However, one can only exploit the flexibility offered by SRSs if algorithms exist that can efficiently synthesize plans to change from one configuration to another. So far, developing efficient algorithms has proved difficult, particularly when there are many subunits to coordinate, and intricate local constraints to consider.

In this paper, we consider a specific reconfiguration architecture, the hexagonal metamorphic robot (HMR). This architecture is simple to describe, yet captures many of the difficulties in planning for an SRS. We present an algorithm that, for a specific subspace of the Claytronics HMR state space (to be explained later) called the Surface space, is capable of solving tasks in near linear time on average. A goal is to synthesize plans for the difficult Claytronics HMR state space. In prior work, the planner presented here was combined with an additional planner, the combination of which solved up to 95% of shape reconfiguration tasks in linear time, on average, for tasks involving up to 20,000 units.[12]

This paper aims to explore why some reconfiguration state spaces are easier to plan within, and in particular, how these easy planning spaces can be found contained within harder state spaces. We will use the Surface space as an example of an "easy" state space that can be found within a number of possible "hard" state spaces of the HMR. We demonstrate that the subspace is well connected (in a sense to be made precise), which is why planning tasks can be solved efficiently using greedy methods with a low probability of failure. We test this hypothesis by utilizing a sampling-based method to estimate quantitative descriptors of the algebraic connectivity of the state space. We compare the results from this specialized subspace against a more general model of HMR reconfiguration, and discover a striking qualitative difference in the behavior of the algebraic connectivity as the number of subunits in the configuration grows. The implication is that the Surface space contains few bottlenecks, even when there are high numbers of subunits.

A second desirable property of the Surface space is that the different instances of the reconfiguration space, corresponding to incremental addition of a module, are well ordered in a specific sense. Specifically, we prove that the reconfiguration graphs at increasing levels of complexity are ordered by the graph minor relation, in a way that seems to extend the notion of metamodularization. Ordering by graph minors explains why certain SRS models can be solved recursively in a particularly simple and efficient way. We hope that these ideas might inspire further analysis of the global structure of reconfiguration spaces and algorithm designs.

While the specific results of this paper are phrased in the context of the study of a specific algorithm for a specific model of a SRS, the quantitative and analytical tools can be applied to any SRS, to explain when and why a subspace of a reconfiguration space for an SRS may be good to plan within, providing tools for characterizing and evaluating a subspace's suitability for efficient planning. In future work, we hope that these tools can be utilized to develop automated methods for identification of useful subspaces and other abstractions, to seed the development of SRS planning algorithms for different SRS architectures.

* Corresponding author. E-mail: tom.larkworthy@gmail.com

2. Preliminaries

Let $\mathbb{P}$ denote the set of points on a hexagonal lattice, $\mathcal{L}$. The metric $d : \mathbb{P} \times \mathbb{P} \mapsto \mathbb{Z}$ is defined as the Manhattan hex distance (see ref. [3] for details). We say two locations, $x_1 \in \mathbb{P}$ and $x_2 \in \mathbb{P}$ are adjacent as $isAdj(x_1, x_2) \Leftrightarrow d(x_1, x_2) = 1$. The undirected connectivity graph, $\mathcal{G}_{\text{conn}}$, of a set of locations, $V \in \mathcal{P}(\mathbb{P})$ ($\mathcal{P}$ denotes the power set function) is the graph constructed from $\mathcal{G}_{\text{conn}}(V) = G(V, \{(e_1, e_2)|isAdj(e_1, e_2)\})$.

In all models of the HMR described here, a configuration, c, is a connected set of robotic subunit locations, $c \subset \mathcal{P}(\mathbb{P})$ where $\forall x, y \in c$ there exists a path in $\mathcal{G}_{\text{conn}}(c)$. There are often further constraints to the admissible set of configurations depending on the HMR model.

A move, m, is an ordered pair of positions, $m \in \mathbb{M} = \mathbb{P} \times \mathbb{P}$. A single-move plan, is an ordered sequence of moves. Whether a move is admissible depends on the motion catalog, which is different for different models of the HMR.

We specialize the general definition of a metamorphic system by Ghrist *et al.*[1,6] for describing HMR motion catalogs here. Ghrist *et al.* permitted an arbitrary alphabet of symbols to label an arbitrary embedding space to describe a specific state of the system. A "state" in Ghrist *et al.* is a configuration of subunits for our purposes. Our alphabet for labelling the hexagonal lattice, then, is simply $A = \{OCCUPIED, EMPTY\}$. By the Ghrist *et al.* definition, a local metamorphic system's permissible state transitions are completely described by a motion catalog, C, which is a collection of generators. A generator describes which labels may change (the trace) when a given context is present (the support). Specifically, a generator, $\phi \in C$ consists of a support, $SUP(\phi) \subset \mathcal{P}(\mathbb{P})$, a trace, $TR(\phi) \subset SUP(\phi)$ and an unordered pair of labeled local states, $\hat{U}_{0,1} : SUP(\phi) \mapsto A$ satisfying

$$\hat{U}_0|_{SUP(\phi)-TR(\phi)} = \hat{U}_1|_{SUP(\phi)-TR(\phi).}$$

In other words, the labeling of $\hat{U}_0$ and $\hat{U}_1$ are equal over the support locations, but may differ in the trace. For the HMR motion catalogs described here, the trace consists of two adjacent locations, and the local states are labeled to reflect that a single unit moves from an *OCCUPIED* location to an *EMPTY* location.

Generators describe move classes, but an actual movement is carried out at a specific location in the embedding space. Ghrist *et al.* define an action of a generator $\phi \in C$ as a rigid body translation, $\Phi : SUP(\phi) \mapsto \mathcal{L}$, thus providing information as to where the generator was applied and in what direction. Given a state $U : \mathcal{L} \mapsto A$, the action is admissible if $\forall x . \hat{U}_0(x) = U(\Phi(x))$. The result of the action on the state is

$$\Phi[U] := \begin{cases} U & : on\ \mathcal{L} - \Phi(TR(\phi)) \\ \hat{U}_1(\Phi^{-1}) & : on\ \Phi(TR(\phi)). \end{cases}$$

In all the specific catalogs used here, the trace consists of two adjacent locations labeled *EMPTY* and *OCCUPIED* in $\hat{U}_0$ which are swapped for $\hat{U}_1$, representing a subunit moving to an adjacent location. So we can work out the rigid transform Φ from $m \in \mathbb{M}$. The Ghrist *et al.* notation is very general, and in the algorithms presented in this paper, we only need to know whether a move is admissible or not, so for convenience we define the function $match_C : \mathbb{M} \mapsto boolean$ to return true if the move is admissible for the given catalog, C.

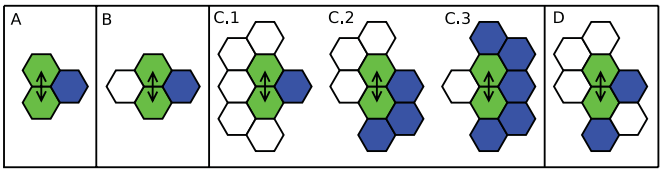

Fig. 1. Previous motion catalogs modulo isomorphisms. Green denotes the trace, where a subunit can move between. Blue and white respectively denote where subunits must be/must not be in the local context for the move to be admissible. (A) The original motion catalog for the hexagonal metamorphic robot by Chirikjian.[3] (B) The motion catalog for the Claytronics prototype.[9] (C) The three generators comprising of Ghrist's example motion catalog.[1] (D) A move permitted by the Claytronics model but not Ghrist's as it changes the gross topology of the aggregate.

3. Background

Chirikjian originally proposed the HMR.[3] This robot was one of the earliest proposed SRSs and remains a prototypical example of a lattice-based SRS. The state of the SRS is entirely specified by the locations of all subunits on the lattice. This is in contrast to chain type SRSs and unit compressible SRSs where subunits may have further internal states, such as joint angles.

In the HMR model of motion, a subunit may move each time iteration. In single-move planning, only one subunit is permitted to change lattice location per time step, and in multimove planning, several may change position per time step. Whether a subunit can move or not is dependent on its local context, as well as the global requirement that all units remain connected. A number of different local motion catalogs have been developed, giving rise to several different versions of the HMR model. Chrikjian's original definition was the least restrictive, locally requiring only that a robotic neighbor was present for the moving subunit to pivot around (Fig. 1A). However, the additional global requirement that the subunits of the configuration remain connected requires explicit checking before any move can be considered admissible. This global check requires $O(n)$ time to run.

Pamecha *et al.*[16] were able to develop an algorithm capable of solving Chirikjian reconfiguration tasks using a greedy simulated annealing procedure. Since similar approaches fail for other similar models,[11] we conclude that their success was perhaps due to the relatively unrestricted motion catalog. In general, for simulated annealing (and other randomized approaches) to be successful on a problem, the planning space must not contain deep local minima or related constraints.[13]

A mechanical prototype of the HMR has been developed, capable of moving according to the Chirikjian motion catalog.[15] This motion catalog implies that a moving subunit could be potentially completely enveloped by robotic neighbors, save for the empty space it is moving into. This puts strong requirements on the geometry of the subunit. If we

denote the hexagonal space as six equilateral triangles with sides of length r, then a stationary unit must span a distance of $2r$ in order to physically connect to its six neighbors. However, when the subunit moves, it must squeeze through a space as little as r, or alternatively, its neighbors must be compliant in some way. In order to meet these strict requirements, the prototype was constructed from six rotary actuators in a six-bar linkage mechanism. Thus, each subunit was quite complex, making it a rather undesirable platform for many practical applications.

A mechanically simpler physical realization of a hexagonal metamorphic robot was developed by the Claytronics team.[9] This motion catalog was more restrictive, requiring empty space to be present opposite the subunit around which the moving subunit was pivoting (Fig. 1B). The extra pivoting space permitted the physical units to be circles and no deformation of geometry was necessary in the physical realization, making it desirable from the pragmatic viewpoint of manufacture. Unfortunately, the requirement of extra space in the motion catalog makes planning much harder. The simulated annealing technique successfully used by Pamecha *et al.* for the Chirikjian catalog requires exponential time as the number of modules increases.[11] As was the case for the Chirikjian motion catalog, the Claytronics motion catalog requires an explicit connectivity check to ensure the aggregate remains connected during moves, at linear cost.

In an attempt to develop a cross-model theory of self-reconfiguration, Ghrist *et al.* developed a general definition of a *local* metamorphic system.[1,6] Ghrist required, for all local metamorphic systems, that the motion catalog should completely describe the local and global motion constraints. Neither of the above-mentioned catalogs satisfy this definition, as the requirement that the configurations do not become disconnected must be enforced explicitly, and *s*eparately, from the local motion catalog. Ghrist suggested a new motion catalog for the HMR (Fig. 1) as an example of a local version of a similar metamorphic system. Ghrist's new HMR motion catalog prevents gross topological changes to occur during motion, such as introduction of enclosed space into the configuration, or disconnection between subgroups of subunits.

The theoretical analysis by Ghrist *et al.* yields useful implications for all HMR reconfiguration state spaces. Ghrist represented the reconfiguration state space as a cubical complex. Each cube grouped moves that were commutative. That is, a sequence of moves drawn from a cube could be admissibly applied in any temporal permutation, i.e., the local contexts defining the applicability of the moves were not overlapping. Further, this state complex was of non-positive curvature, with the immediate result that any multimove plan (including single-move plans) could be deformed locally into an optimal version of the same homotopy class in $O(n^2)$*.

The state description of a configuration for a HMR is the same for all models discussed. It is only the admissible motions that are different. One can see immediately from the motion catalogs (Fig. 1), that Ghrist's motion constraints are more restrictive than the Claytronics motion catalog which is more restrictive than the Chirikjian motion catalog. Clearly, all less constrained models can execute plans applicable for more constrained models.

Determining admissibility of moves in Ghrist's HMR model can be done in $O(1)$, or in $O(log(n))$ persistently.[11] In general, local metamorphic systems are computationally attractive because only local contexts need to be considered. However, in practice, even with the additional cost of a linear cost connectivity check, the Claytronics model is quicker to plan for than Ghrist's HMR motion constraints. This has been found to be true on a broad range of general planning strategies:[11] simulated annealing,[10] greedy search,[20] rapidly-exploring random trees[13] and probabilistic roadmap planning.[8]

A different form of HMR is presented in ref. [18] with $O(\sqrt{n})$ movement time performance. We consider this to be different from the other HMRs discussed so far in two respects. First, it uses an additional empty third dimension for units to move into during planning, which simplifies planning considerably by providing a large empty space for units to utilize. Secondly, each unit is given a momentum, which allows a single unit to move faster than one lattice location per time unit. The computation effort required to form plans is still $O(n)$, but for a less constrained model than the original Chirikjian HMR.

In this paper, we present another subspace, called the Surface space, that can be viewed as a further constrained version of Ghrist's HMR motion catalog. It inherits Ghrist's locality and admits a simple planning algorithm to solve Surface-to-Surface reconfiguration tasks efficiently. This Surface state space has been used in other related work to form long distance plans within the Claytronics HMR reconfiguration state space, which allowed Claytronics-to-Claytronics reconfiguration tasks to be solved, empirically, on average, in linear time[12] for 97% of the state space. An arbitrary Claytronics HMR configuration, on average, is only a few moves away from a nearby Surface adhering configuration using the Claytronics motion catalog. It is thus tractable to compute a motion trajectory in the more general, and computationally less efficient, Claytronics space only a small distance to find a nearby Surface configuration. The previous work[12] found a pair of nearby Surface configurations corresponding to the Claytronics start and end configurations in the reconfiguration task, and found a trajectory between them using a more efficient version of the Surface-to-Surface planner presented here. While the algorithm was highly efficient and operated on a large fraction of the Claytronics state space, the Claytronics-to-Surface planning step was essentially a heuristic method that had a failure rate proportional to problem complexity. In the interest of clarity, all the analysis in the following sections is restricted to the properties of the Surface state space, for which a well-behaved motion planner is presented.

So it is the nature of the reconfiguration state space of the Surface HMR motion model we wish to understand further. It is worth noting that the motivation for the Surface state space definition is removal of the planning difficulties associated with the Claytronics motion catalog requiring empty space

* Non-positive curvature in simple terms means that there are "no fat triangles" in the space. So, the optimal shortest path can be found with a series of local improvements from *any* starting path within the same homotopy class.

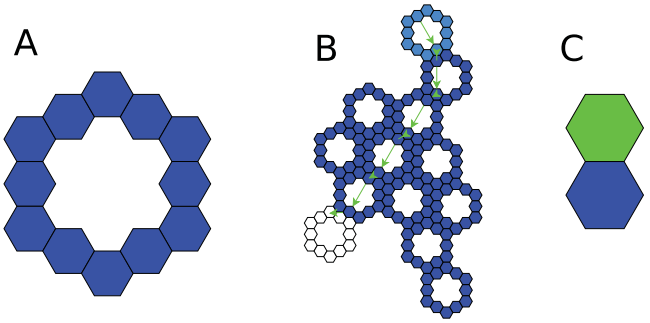

Fig. 2. (A) A potential metamodularization of the HMR that permits mobility of the subunits found in the center of each hexagon edge using the Claytronics motion catalog, in particular, providing empty space opposite the pivot locations. (B) A configuration built out of 12 such metamodules, and an example of how local metamodule motion primitives can be daisy-chained together to the effect of moving one metamodule to an empty location adjacent to the perimeter. (C) Expressing the local support required for a subunit to enter or leave a location.

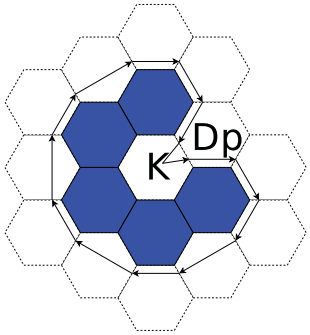

Fig. 3. Wrapping a tour around a configuration. This configuration is not a valid Surface configuration because it contains a kink violation (K) and a dual path violation (Dp).

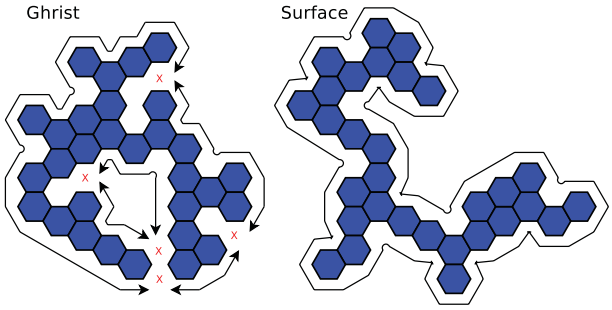

Fig. 4. Examples of a valid Ghrist configuration and a valid Surface configuration. Ghrist configurations may contain narrow intrusions of space, which prevent subunits on the perimeter from crossing. Surface configurations, by construction, do not.

opposite the pivot location. Previous attempts at abstracting away troublesome constraints in other SRS models have centered around metamodularization of the state space.[19] In metamodularization, the atomic planning unit is actually a collection of SRS subunits (Fig. 2A) with predefined sequences of moves that permit the metamodules to move with fewer motion constraints than the underlying subunits (Fig. 2B). The planning task is thus simplified, but at the cost of coarsening the embedding lattice significantly. The drawback of the methodology becomes apparent when one considers the proportion of configurations in the underlying SRS state space that have a representation in the metamodule state-space. Metamodule conforming configurations occupy an almost negligible proportion of the overall general state space. Metamodularization, then, does not lend itself well to being used as an intermediate path through a more general configuration space (for example, there are no metamodule configurations for 13 subunits for the example in Fig. 2). Both a metamodularization subspace and the Surface subspace achieve similar qualitative behavior, simplifying planning, by adding additional motion constraints to an underlying motion model. The Surface space achieves a similar result to metamodularization, but by sacrificing fewer configurations.

So the motivation for this work is to shed light on the question: why is it that adding extra constraints can sometimes make planning harder, as in adding constraints to the Chirikjian catalog to create the Claytronics catalog, and yet sometimes easier, as in adding constraints to the Ghrist catalog to create the Surface catalog, or by applying a metamodularization strategy? By identifying general principles that describe when and how adding constraints can simplify planning, we expect advances in reconfiguration algorithms for much harder models of SRS whose motion state space is difficult to mentally visualize, e.g., M-TRAN.[14] Furthermore, adding artificial constraints to a model reduces the number of states the augmented system can express, so our work will aid in constructing constrained motion catalogs that sacrifice the minimum state space volume for a gain in planning efficiency.

4. A Surface-to-Surface Planner

A Surface adhering configuration, $c \in \mathbb{S}$, is defined as a configuration that permits a Hamiltonian path to be wrapped around the adjacent external locations ($adj(c)$). This requirement is compromised by two classes of violation. A kink violation is present when the peripheral tour leaves through the same edge it enters from (Fig. 3), and a dual path violation is where the tour traverses through the same location more than once (Fig. 3).

A valid Hamiltonian path implies several properties relevant to motion under the Claytronics and Ghrist motion catalogs. If an extra subunit is added, the subunit can move in a complete loop around the entire perimeter of the configuration. The lack of dual path violations implies there is always open space above the additional subunit for pivotal space. Absence of kink violations implies there cannot be a change of gross topology, specifically an introduction of enclosed space, caused by a subunit bridging the empty space adjacent to the kink (Fig. 1D).

Note however that while the added subunit is able to move around the configuration freely using Ghrist's motion catalog, it may not be able to stop anywhere on this path and still result in a valid Surface configuration. The moving subunit may itself cause kink or dual path violations. The Surface model's constraints merely imply that if a unit can be removed from one perimeter location and placed at another valid location, then a sequence of Ghrist motion moves will exist to link them (although violations may transiently be generated when executing the underlying Ghrist sequence).

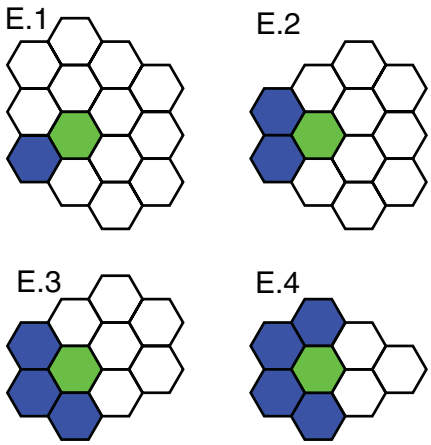

Fig. 5. There are four different generators for adding/removing a subunit from a Surface configuration.

Whilst the Hamiltonian path constraint is a useful description of the Surface model's restrictions, both for implementation and visualization of the path around the configuration subunits take, we can rewrite this functionality in terms of a new set of local contexts for the motion catalog. This proves that the Surface HMR model is also a *l*ocal metamorphic system by Ghrist's definition. A major difference with the motion catalog for the Surface model compared to the other HMR models is that the start and end locations for a move may not be adjacent. So a Surface plan is a set of moves that relocate individual subunits from one location on the perimeter to another, with the guarantee that a detailed sequence of consecutive Ghrist moves will exist that pass through the Surface plan way-points.

Figure 5 shows the generators where a unit can be added or removed from a Surface configuration to generate another valid Surface configuration. If local states at the single trace location, tr, satisfy $\hat{U}_0(tr) = EMPTY$ and $\hat{U}_1(tr) = OCCUPIED$ then we say the generator is an *ADD*, otherwise it is a *REMOVE*. A move for the Surface model is a *REMOVE* followed by an *ADD* elsewhere. Technically, in Ghrist parlance, the catalog for the Surface model is the (infinite) union of all possible relative arrangements of a *REMOVE* and an *ADD* whose labeled local states agree.

Figure 5 was not generated by hand. All valid Surface configurations containing eight subunits were enumerated using the Hamiltonian path constraint description above. The relative local contexts of adjacent empty space was stored and annotated with a label describing whether a subunit could be added or not. This set of annotated local contexts was processed by the C4.5 algorithm[17] found in the Weka[7] data mining library to produce a shallow decision tree. The decision tree had 100% accuracy at determining what necessary local context was present to add a subunit, and was optimized upon valid configurations only. As a side effect, the data mining tool identified that a subunit could not be added if it created the patterns shown in Fig. 6.

Lemma 1. *For any given Surface adhering configuration, an additional module can move around the perimeter in a complete loop using the Ghrist motion catalog.*

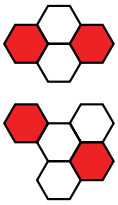

Fig. 6. A Surface configuration never contains the above patterns of empty space (white) and robotic subunits (red).

Proof. By construction, discussed above. □

The Surface-to-Surface planning algorithm finds an admissible sequence of Surface moves in order to change one Surface adhering configuration into another. From Lemma 1 the resulting plan can be executed by a HMR constrained by the Ghrist, Claytronics or Chirikjian motion constraints. Each single move in the Surface HMR, however, is a concatenation of several single moves by the other catalogs (a so-called, long-move) on account of the start and end location decoupling. The high level Algorithm is outlined in algorithm 1. For clarity, the version presented here does not include a number of additional optimizations (see ref. [12]). So this specific algorithm does not really run in linear time. However, the salient features relevant for the present discussion have been preserved.

The algorithm's main loop incrementally changes the current configuration (which is initially the start configuration) toward the goal configuration by applying valid Surface moves determined by *improve*. The algorithm tracks a set P which represents placed subunits. Once a unit is placed, it is no longer considered by the *improve* subroutine to be a possible subunit that can be moved. P is updated incrementally by *updateP* (Algorithm 2); a subunit is considered placed if it is adjacent to an already placed unit, and the goal contains a unit at the same location. It has been empirically determined that, on average, only $O(\sqrt{n})$ long-moves are required to transform a start configuration into the goal configuration.[12]

The subroutine *improve* finds a valid Surface move that moves a subunit not in P to a location that would lead to an addition to P. The *improve* subroutine is highly optimized elsewhere to maximize the chances that only a few moves need be considered,[12] but these optimizations are not included here. There is a small chance that no move will improve P in which case the planner fails to find a solution for the reconfiguration task at hand. Empirically, this seems to happen rarely. In fact, the probability of failure tends toward zero as n tends to infinity (Table I).

The result of the Surface-to-Surface planner is a sequence of long-moves, representing location-to-location traversals

Table I. Probability of *StoS* failing to find an improvement in random tasks decreases as the number of units in the random configurations increase.

Units, n	fails	trials	95% C.I. of P(fail)	
250	264	10,000	0.0233	0.0297
500	7	10,000	0.0003	0.0014

Algorithm 1 The 𝕊urface-to-𝕊urface finds a set of admissible moves to change a configuration c into a configuration c_{goal}. A set of placed subunits, P, is incrementally grown by calls to *improve*. *improve* searches for admissible moves to improve P.

$improve : \mathbb{S} \times \mathbb{S} \times \mathcal{P}(\mathbb{P}) \mapsto M$
$improve(c,\ c_{goal},\ P) \triangleq$
 for$(\forall s.s \notin P \wedge s \in c, \forall e.e \notin P \wedge e \in c_{goal})$
  if $(match(c,\ s,\ REMOVE)))$
   if $(match(c - \{s\},\ e,\ ADD))$
    return (s, e)
throw error
$StoS$: $\mathbb{S} \times \mathbb{S} \mapsto M^k$
$StoS(c, c_{goal}) \triangleq M^k$
$P \leftarrow updateP(\emptyset, ORIGIN, c, c_{goal})$
while$(|P| < |c|)$
 $m \leftarrow improve(c, P, c_{goal})$
 $c \leftarrow c \cup \{m_2\} - \{m_1\}$
 $P \leftarrow updateP(P, m_2, c, c_{goal})$
 $append(m, M)$

Algorithm 2 The set of placed units, P, is updated recursively. If a location, x, is in the current and goal configuration but not in P it is placed in P and *updateP* is called on its neighbors.

$updateP : \mathcal{P}(\mathbb{P}) \times \mathbb{P} \times \mathbb{S} \times \mathbb{S} \mapsto \mathcal{P}(\mathbb{P})$
$updateP(P_{prev}, x, c, c_{goal}) \triangleq P$
$P \leftarrow P_{prev}$
 if$(x \in c \wedge x \in c_{goal} \wedge x \notin P)$
  $P \leftarrow P \cup \{x\}$
  for$(\forall q.isAdj(q, x))$
   $P \leftarrow updateP(P,\ q,\ c, c_{goal})$

round the perimeter of the intermediate configurations. Unwrapping the long-moves into a sequence of short-moves, compatible with other HMR models, can be done in near linear time.[12] This is possible because, on average, the perimeter distance for each long-move scales as $O(\sqrt{n})$, and the number of long-moves required to reconfigure also scales as $O(\sqrt{n})$. Thus, it appears empirically justified that the average asymptotic performance of the Surface-to-Surface planner is nearly linear (subject to how close to constant time *improve* can be implemented) with an insignificant failure rate for large n.

We wish to understand several things about this algorithm. Why does the algorithm asymptotically fail less as $n \to \infty$, even though it is essentially a local heuristic? Why can the task be solved incrementally by growing a placed set, P? Also, why cannot the Ghrist HMR reconfiguration tasks be solved in a similar fashion, i.e., what makes this particular HMR motion catalog *special*?

5. The Surface Space is Highly Connected

In general terms, a planning algorithm's task is to find paths through some space, C_{free}, for multiple start and end points. The difficulty of the task, and therefore the minimal complexity of a planning algorithm, is inherently coupled to the properties of the configuration space. Typically, for computational reasons, one approximates a continuous or

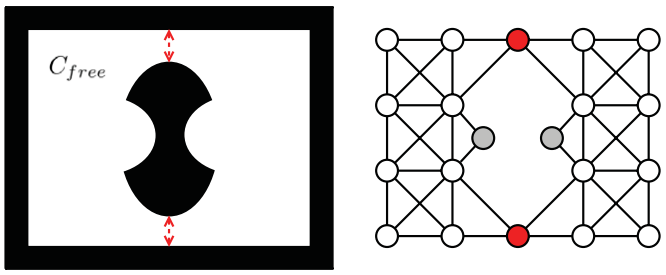

Fig. 7. Left: a simple example of a planning space. The obstacle in the center causes bottlenecks in C_{free} (shown in red). Right: a graph approximation of the same space.

otherwise complex, C_{free}, by discretization or sampling-based methods to yield a graph whose vertices are states in C_{free}. For HMR planning, C_{free} is naturally discrete; vertices of the space represent configurations, and edges represent admissible moves (or sets of moves in multimove planning, but not considered here).

Bottlenecks in C_{free} are well known complications for planners.[13] A bottleneck is a suitably narrow subset within C_{free} that constrains different possible solution paths to go through it in order to traverse between much larger subsets of the space (Fig. 7, in red). Iterative or sampling-based planning algorithms that must 'discover' such bottlenecks computationally can face serious challenges as they may be naively expending precious computational time exploring irrelevant areas of C_{free} (Fig. 7, in gray).

Well-founded graph-theoretic measures can concisely express what we mean by a bottleneck. Let $G = (V, E)$ be a simple unweighted graph. A cut, $W \subset V$, separates the graph into two sets of vertices, W and $V - W$. Let $deg(x \in V)$ be the degree of a vertex. Then the volume of a set of vertices $W \subset V$ is defined as $vol(W) \triangleq \sum_{i \in W} deg(i)$. The cost of a cut on the graph is $cut(W) = |\{x = (u, v) | x \in E \wedge u \in W \wedge v \notin W\}|$, in other words, the number of edges crossing the cut.

The Cheeger constant of a graph, h_G, is a measure of "bottleneckedness" which finds a small cut that separates the graph into two large volumes.[4] It is defined as

$$h_G = \min_S \frac{cut(S)}{\min\{vol(S), vol(V - S)\}}.$$

Directly measuring the Cheeger constant for a graph is computationally intractable. It is, however, bounded by the second smallest eigenvalue of the Laplacian matrix (which is also known as the algebraic connectivity of the graph[4]).

The Laplacian L matrix of a graph is

$$l_{i,j} := \begin{cases} 1 & \text{if } i = j \text{ and } deg(v_i) \neq 0 \\ -\frac{1}{\sqrt{deg(v_i)deg(v_j)}} & \text{if } i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{otherwise.} \end{cases}$$

If L has the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ then the Cheeger constant h_G is bounded by

$$\sqrt{2\lambda_2} > h_G \geq \frac{\lambda_2}{2},$$

Table II. The number of vertices, edges and measure of algebraic connectivity (λ_2) for the Ghrist and Surface model reconfiguration graphs generated by different numbers of subunits (n). The states spaces are identical up to $n = 4$, and only differ marginally at $n = 7$.

	Ghrist			Surface		
n	$\|V\|$	$\|E\|$	λ_2	$\|V\|$	$\|E\|$	λ_2
2	6	15	1.2000	6	15	1.2000
3	33	168	1.1429	33	168	1.1429
4	176	1431	1.1186	176	1431	1.1186
5	930	10836	1.1033	900	10332	1.1111
6	4878	75945	1.1	4482	67725	1.0978
7	25,480	506,394	1.0872	21,910	417,042	1.0909

λ_1 is 0 for all Laplacians.[4] λ_2 is known as the algebraic connectivity of the graph. If a graph has a low Cheeger constant, this implies there are small cuts that can separate large volumes of the graph. This captures the essence of bottlenecks; paths between the two volumes must be routed through a small corridor.

Our hypothesis is that the configuration space of the Surface model has fewer bottlenecks compared to the Ghrist model of the HMR. We will use algebraic connectivity and its relation to the Cheeger constant to measure the severity of bottlenecks in C_{free} for the Ghrist model and the Surface model. However, first note that the atomic moves in the Ghrist motion model represent a single subunit moving an adjacent location, whereas Surface moves represent a single subunit moving a number of lattice locations. To compare the models fairly, we created an analogous definition of a long-move for the Ghrist model to be a sequence of consecutive admissible moves applied to a single subunit, i.e., all the locations that a single subunit can reach while the other subunits remain fixed. In addition, because both model catalogs do not permit gross topological changes in the morphology, we only study configurations that do not contain enclosed space.

For the configuration graphs containing up to seven subunits, it is possible to construct the Laplacian and calculate the algebraic connectivity directly. The results are shown in Table II. However, the reconfiguration graphs differ very little at such low numbers of subunits for the two models. With few subunits, there are not enough permutations of possible local contexts to differentiate the models. The difference between models only becomes apparent at higher complexity levels.

Unfortunately, expanding the configuration graphs containing larger number of subunits quickly becomes intractable. So in order to estimate the properties of the Cheeger constant at higher complexity levels, i.e., subunit numbers, we use a sampling methodology. First, we generated a random Surface adhering configuration containing n units by iteratively uniformly selecting an ADD action to a growing configuration (starting from the ORIGIN). Second, we applied 1000 random moves from the respective motion catalog (long-moves for the Ghrist model), so that the initial Surface configuration diffuses into a model-specific area of the configuration space. Finally, the model-specific configuration reached is used as a starting location for taking a sample. Examples of configurations generated by this procedure are shown in Fig. 4.

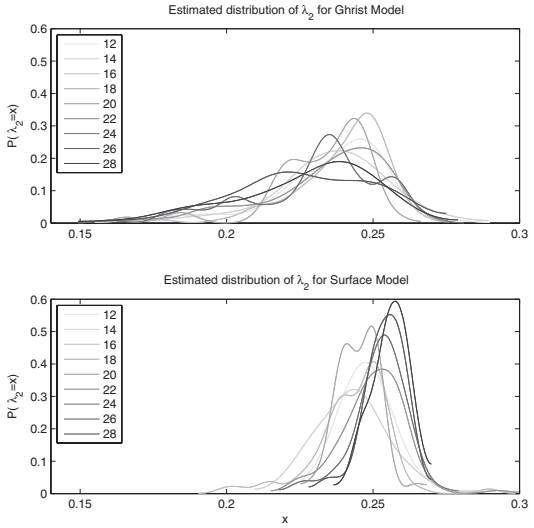

Fig. 8. The estimated densities of λ_2 of the Laplacian after sampling 100 subgraphs from the reconfiguration spaces of the Ghrist model and the Surface model.

A sample subgraph is generated by performing a breadth first search to a depth of two from the sample location (the sample location, its neighbors, and its neighbor's neighbors). The algebraic connectivity may be computed for this subgraph, and should correlate to the global algebraic connectivity. This procedure was applied to 100 samples for each complexity level under study. The smoothed results are shown in Fig. 8.

For the Ghrist model, Fig. 8 shows that as the number of units in the configuration increases, so the spread of λ_2 increases, and the mean diminishes. This suggests that our sampled subgraphs are increasingly likely to contain bottlenecks. For higher numbers of subunits this suggests that the Cheeger constant is tending toward 0. For the Surface model the reverse seems to be true. The spread of λ_2 is decreasing, and the mean increasing. Bottlenecks seem to be sparser as we add more units to the Surface configurations.

Our interpretation for the Surface model is that when there are very large numbers of subunits, the movement of a particular subunit is relatively unrestricted; it is able to move anywhere on the surface. Interaction between subunits mainly occurs at the local level, whose importance diminishes as the number of units grows. Thus, local interactions that cause bottlenecks become less likely, and the mobility of subunits on the perimeter increases. For the Ghrist model, it only takes two kinks on the surface to divide the perimeter into two classes that units cannot move between (see the H configuration in Fig. 17 in next section). As the number of units grow, so the probability of two or more kinks being found somewhere on the perimeter tends toward certainty.

The implication of Fig. 8 is that the Surface model has fewer bottlenecks compared to the Ghrist model. The sparsity of bottlenecks in the Surface model explains why a greedy planning procedure, such as the one employed in the S-to-S

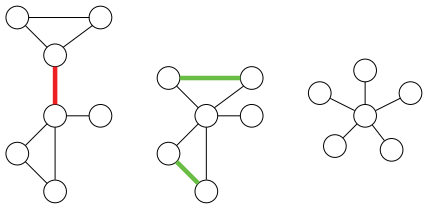

Fig. 9. Graph minor operations, graphs to the right are minors of those to the left. Red denotes an edge contraction operation, and green an edge deletion.

planner, suffices in an increasing proportion of cases as n grows.

6. Graph Minor Substructure

The previous section uses Spectral Graph Theory in order to explain when greedy planning methods suffice in a reconfiguration state space. Within this next section we introduce the use of the Graph Minor Theory to the analysis of SRS state spaces, first as a compact, precise notation for representing that one state space is a constrained version of another, and as a tool that reveals startling differences between the easy and hard planning spaces as subunits are added. This last point in particular partially explains why efficient planning methods may only exist for some planning state spaces.

Definition 2. A graph, H, is said to be a *minor* of a graph, G, denoted $H \leq G$ if there exists a sequence of edge deletions, contractions and vertex deletions to change G into H.[5]

Figure 9 illustrates the basic graph minor modification operations. The graph minor relation is a compact notation for describing when one state space can be executed upon another model; for describing when one SRS motion model is a constrained version of another. Consider the similar subgraph relation.

Definition 3. A graph, H, is said to be a *subgraph* of a graph, G, if there exists a sequence of edge deletions and vertex deletions to change G into H.[5]

The subgraph relation lacks edge contractions. Now consider the metamodule reconfiguration state space graph containing k metamodules, $\mathbb{M}_k$, and the Claytronics state space which contains 12 subunits for every metamodule, $\mathbb{C}_{12k}$ (Fig. 10). Metamodule movements are built from sequences of underlying motion primitives, so although each $\mathbb{M}_k$ vertex is present in the $\mathbb{C}_{12k}$ graph, each edge of the $\mathbb{M}_k$ graph represents a *sequence* of underlying $\mathbb{C}_{12k}$ edges. Thus $\mathbb{M}_k$ is not a subgraph of $\mathbb{C}_{12k}$, yet it is a graph minor, i.e., $\mathbb{M}_k \leq \mathbb{C}_{12k}$. Similarly, we can summarize all the discussed HMR state spaces using graph minor nomenclature as $\mathbb{J}_k \geq \mathbb{C}_k \geq \mathbb{G}_k \geq \mathbb{S}_k$, where $\mathbb{J}_k$, $\mathbb{C}_k$, $\mathbb{G}_k$, and $\mathbb{S}_k$ stand for the Chirikjian, Claytronics, Ghrist, and Surface model reconfiguration state spaces containing k subunits, respectively.

The minor relation is a compact, precise notation for expressing what we mean by one state space is a constrained version of another. Butler *et al.*[2] present an argument that their cubic SRS model is generic because it can be

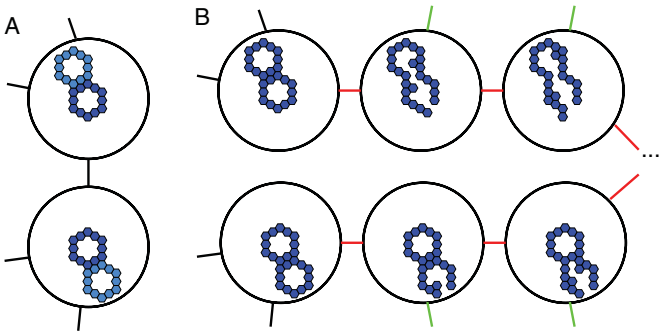

Fig. 10. A. A metamodule can tunnel through another, represented by a single edge in the planning state space. This single move in the metamodule state space is implemented using a sequence of moves from the underlying state space. The transient moves used in the Claytronics state space have used the edge contraction operation to form atomic moves in the metamodule state space (B, red). The edge contractions, plus pruning of non-conforming metamodule states and moves (green) show that metamodularization corresponds to a graph minor of the Claytronics state space.

instantiated by various existing SRS motion catalogs. If we denote their cubic SRS reconfiguration state space containing k subunits as $\mathbb{B}_k$, the metamodularized M-Tran state space as $\mathbb{T}_k$, we can rewrite one of the instantiations described in the work as $\mathbb{B}_k \leq \mathbb{T}_{4k}$, as four M-Tran units were required to cooperate in order to achieve the minimum motion requirements of their model.

While the graph minor relation is a useful notation for describing relationships between different SRS motion catalogs, we now look at the family of state space graphs of an individual SRS catalog generated by different numbers of subunits, e.g., $\mathbb{S}_n$, for $n > 1$; to understand how the planning problem changes as more subunits are added. Intuitively, one might presume that the state space graph for a HMR model containing i subunits will share similarities with the state space containing $i + 1$ subunits. We address this question formally and find a significant difference between the state spaces generated by models that are hard to plan for, e.g., the Ghrist catalog, and models that have efficient solutions in existence, e.g., the Surface catalog or a metamodularized state space.

For the Surface model, the i reconfiguration graph is a graph minor of the $i + 1$ reconfiguration graph, $\mathbb{S}_1 \leq \mathbb{S}_2 \leq \ldots$. This does not appear to be true of the configuration graphs generated by the Ghrist motion catalog. In fact, the counter-examples for the Ghrist case are caused by the very cases where bottlenecks are found. Similar to the $\mathbb{S}$ case, the HMR metamodularization example is also well ordered by the minor relation, $\mathbb{M}_1 \leq \mathbb{M}_2 \leq \ldots$. As will be discussed further later, graph minor ordering in the reconfiguration state spaces has significant implications for the motion planning problem, and is likely to be the mechanism for explaining why one motion model admits efficient planning and others do not.

To show that a reconfiguration state space, $\mathbb{X}_i$ is a minor of $\mathbb{X}_{i+1}$, we utilize the fact that each vertex of the reconfiguration graphs is labeled by the arrangement of subunits on a common embedding lattice. This labeling scheme permits a vertex, v_i of the $\mathbb{X}_i$ graph to be associated

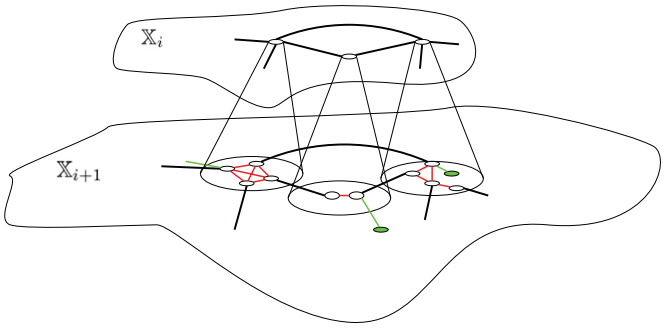

Fig. 11. To globally show $\mathbb{X}_i \leq \mathbb{X}_{i+1}$ we can define a local relationship between the graphs, and show that this relationship locally adheres to the minor relationship. Red denotes edge contractions, and green, edge and vertex deletions. The global minor can be proved by stitching together the local minors.

with a group of vertices, $\bar{\mathbf{v}}_{i+1}$, in the $\mathbb{X}_{i+1}$ graph that corresponds to possible locations a subunit can be added to the v_i configuration to generate a configuration within $\mathbb{X}_{i+1}$. This observation implies that a local area of the $\mathbb{X}_i$ graph has a corresponding local area in the $\mathbb{X}_{i+1}$ graph.

To show globally that the $\mathbb{X}_i$ graph is a minor of the $\mathbb{X}_{i+1}$ graph, it is sufficient to prove that: for every vertex, v_i, in the $\mathbb{X}_i$, that v_i's local graph neighborhood is a minor of $\bar{\mathbf{v}}_{i+1}$ graph neighborhood, and that these local neighborhoods are connected in the same topology. This is summarized diagrammatically in Fig. 11.

The sketch of the proof to show that $\mathbb{S}_i \leq \mathbb{S}_{i+1}$ is as follows. Any configuration adhering to the Surface model implies that if a module can move at all, then it is free to move in a complete loop around the exterior. The S_{i+1} space contains one extra subunit. We show that the subunit can always move out of the way, in order to let any move that existed in the S_i graph take place. Prevention of a move in the $\mathbb{S}_i$ state space for the $\mathbb{S}_{i+1}$ state space can only occur if the extra subunit in the $\mathbb{S}_{i+1}$ state space interferes with the local support that determined the move's admissibility. We argue geometrically that for large configurations, there always exists a potential location for the added subunit that lies outside of the local support locations of the $\mathbb{S}_i$ move. The finite size of the motion catalog's local supports, plus the total mobility of the additional subunit, implies that "get out of the way" moves always exist in the $\mathbb{S}_{i+1}$ state space. The "get out of the way" move edges can be contracted to generate the $\mathbb{S}_i$ graph, thus showing that $\mathbb{S}_i \leq S_{i+1}$. This argument does not follow for the Ghrist model because, in general, the extra subunit does not always have enough freedom to "get out of the way" of the local supports that determined the admissibility of a move.

Concretely, we introduce the notions of local structure in a reconfiguration graph around some vertex, and an inherited local structure which represents the analogous locale in a reconfiguration graph generated by adding a unit.

Definition 4. The *local structure* for a configuration, v, is all configurations reachable by a single Surface move (remember a move is a REMOVE followed by an ADD from the Surface catalog, Fig. 5).

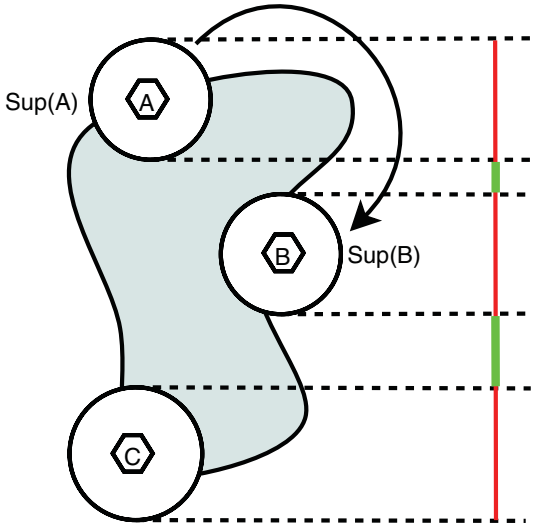

Fig. 12. Whether or not a subunit exists at location C does not affect a move between A and B because its support does not intersect A or B's. Rather than showing this in 2-dim, we project the support areas onto a line parallel to the widest diameter of the shape. Showing the supports do not intersect is simplified to showing the projected support intervals do not overlap.

Definition 5. The *inherited local structure* for a configuration, v, is all possible configurations generated by applying an ADD from the Surface catalog to v. (Fig. 5)

Lemma 6. *Any two configurations belonging to a vertex's inherited structure have a valid move between them.*

Proof. This follows from Lemma 1. Thus the inherited local structure forms a clique of configurations connected by moves.

To show that the *inherited local structure* preserves analogous moves that existed in the *local structure*, we first show that an extra subunit can always be added at a location that is far enough away from the start and end of the move so that it does not affect the local support that determined the moves admissibility (sketched in Fig. 12). If a move is between position A to position B, we need to show that $\forall A \forall B \exists C.(sup(A) \cap sub(C)) \cup (sup(B) \cap sub(C)) = \emptyset$. There are a variety of ways to show this, for simplicity, in the following proof we project the support areas onto a line parallel to the widest diameter of the configuration. □

Lemma 7. *All generators of the Surface catalog have a width of less than 5.*

Proof. See Fig. 5. □

Lemma 8. *A connected configuration containing 631 or more subunits has a large diameter of at least 29.*

Proof. The configuration with the smallest large diameter occurs when subunits are arranged into a perfect hexagon. 631** subunits can be arranged into a large hexagon of diameter 29. Moving any subunit, or adding more subunits, will only increase the large diameter. □

**A Hexagonal Number

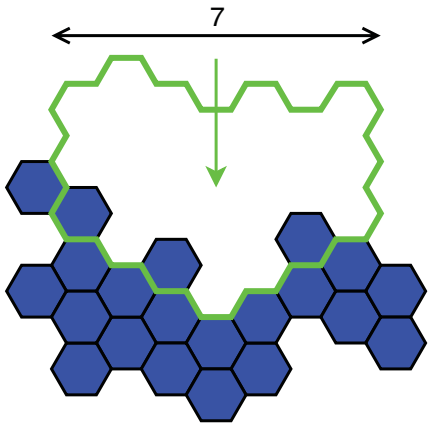

Fig. 13. Lemma 9 is shown by sliding the shape shown in green (of width 7) toward the configuration until it overlaps one or more subunits on its lower edge.

Lemma 9. *For a Surface configuration, within an columnar interval of width 7, a valid ADD location and its complete support is contained.*

Proof. First, a shape of width 7 is slid over empty space toward the configuration (Fig. 13) until intersecting a subunit. The lower row of hexagons comprising of the shape will then contain between one and seven subunits and the remainder shall be empty (by construction). We will now consider different cases of how the bottom row can be occupied in order to show that regardless of how, there is always a location where an extra subunit can be added.

A and B of Fig. 14 reflect the cases of when the bottom row contains only one subunit. In each case a subunit can be added using Surface ADD E.1 (Fig. 5). When two subunits are present and adjacent, Fig. 14C and its generalizations demonstrate Surface ADD E.2 can be used to add a subunit. When the two subunits are a distance of one from each other, case D is relevant. Case D can only occur if additional subunits are found adjacent to the empty location (light blue), because otherwise the configuration would be invalid (Fig. 6). With the implied extra subunits included, Surface ADD E.4 (Fig. 5) is applicable. Another possibility when a pair of lower row subunits are at a distance of one is case E, this however, is an impossible Surface configuration (Fig. 6), but regardless, an applicable ADD location exists. When two subunits are at a distance greater than two, such as in the case F, it is clear one subunit no longer becomes relevant to determining an ADD applicability. For cases with more subunits, the above arguments are trivial to extend (Surface ADD E.3 is used when case B is extended to three subunits). Therefore, within an interval of 7, a valid ADD location can always be found, regardless of the specifics of the Surface configuration. □

Lemma 10. *On a line of length 29 or greater, if two intervals of width 5 are present, then an interval of width 7 can be found which intersects neither.*

Proof. The worst placement of the intervals of width five are shown in Fig. 15 for a line of width 28. Clearly extending

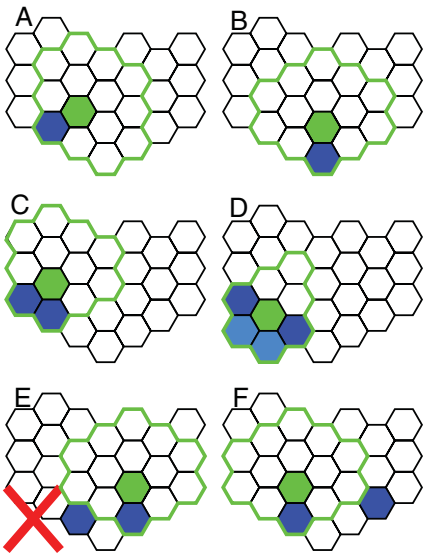

Fig. 14. The major cases for consideration of how the shape in Fig. 13 can be occupied with subunits. The area marked with a green perimeter labels the location of an applicable support for some Surface ADD generator. The location of where the subunit can be added is shown in green.

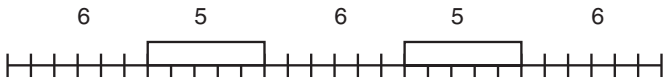

Fig. 15. On a line of length 28, two intervals of width 5 can be placed such that an addition interval of 7 cannot be placed without intersecting one of them.

the length of line by one will permit space for an interval of width 7 to be inserted without overlap. □

Lemma 11. *For any move between location A to location B on a Surface configuration containing 631 subunits or greater, there exists a location C where a subunit can be added, whereby the support of A and B do not intersect the support of C, i.e.,* $\forall A \forall B \exists C.(sup(C) \cap sup(A)) \cup (sup(C) \cap sup(B)) = \emptyset$.

Proof. Projecting the supports of A and B onto a line parallel to the line defining the large diameter of the 631 subunit sized configuration yields two intervals of size 5 (by Lemma 7) on a line of length 29 (Lemma 8). An interval of width 7 shall exist on this line that does not intersect either of the intervals of size 5 (Lemma 10). Somewhere within the area of the configuration that would project to the interval of size 7 exists and ADD location C (Lemma 9) which cannot intersect the supports of A or B. □

Remark. This also holds for configurations of any size, but the only proof we are aware of involves cumbersome enumeration of cases.

Lemma 12. *For every move* $A \rightarrow B$ *in a local structure between configurations* v *and* u*, there exists at least one pair of configurations* v' *and* u' *in the inherited structure between the same locations.*

Proof. A configuration can be represented as a set of subunit locations. Let $v = X \cup \{A\}$ and $u = X \cup \{B\}$. We

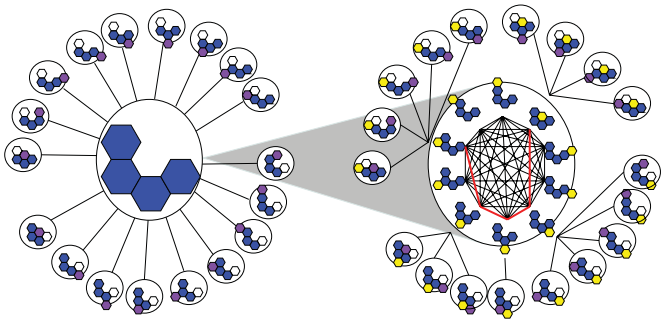

Fig. 16. A local structure (left) is a minor of the inherited local structure (right). The left central vertex is surrounded by all configurations reachable by a move (its local structure). The right central vertex contains the inherited structure for that vertex (a clique), yellow denoting where an additional subunit has been added. For every move in the local structure (white to purple), a comparable move can be found in the inherited structure with an addition subunit added, denoted by the verteces joining the central vertex. The red lines within the inherited structure shows which moves are required to move the additional subunit around to "get out of the way" so that all analogous moves can execute. Deleting all black edges in the inherited clique followed by contracting the red edges reproduces the local structure.

simply need to find an ADD location C that can be added to v and u such that it does not interfere with the support of A and B that enabled the move $A \rightarrow B$ to take place. Lemma 11 shows such a C exists when there are more than 631 subunits in the configuration. Thus a C always exists such that $v' = X \cup \{A, C\}$, $u' = X \cup \{B, C\}$ and the move $A \rightarrow B$ is still valid. □

Lemma 13. *The local structure of a vertex, v, is a graph minor of its inherited local structure v′.*

Proof. Each neighbor, u_i, in the local structure of v, represents a valid move between the configurations v and u_i. By Lemma 12 $\forall u_i$, there exists a location z_i that permits a move between $v \cup \{z_i\}$ and $u_i \cup \{z_i\}$. By definition, the configuration $v \cup \{z_i\}$ is in the *inherited structure*. By Lemma 6 there exists a move between all $v \cup \{z_x\}$ configurations. If all moves between $v \cup \{z_x\}$ are contracted and all edges not $v \cup \{z_i\} \rightarrow u_i \cup \{z_i\}$ in the inherited structure are deleted, the remaining edges are the local structure (see Fig. 16) □

Theorem 14. *The state space of the Surface model containing i subunits is a graph minor of the reconfiguration space containing* $i + 1$ *units,* $\mathbb{S}_i \leq \mathbb{S}_{i+1}$ *for* $i \geq 631$.

Proof. By Lemma 13 every vertex in the i graph is a minor of the inherited graph. For a pair of configurations in the i graph, $u = X \cup \{x\}$, $v = X \cup \{y\}$ with a move between them, $x \rightarrow y$, Lemma 13 states an ADD location on each, $z_u$and z_v exists such that the same move can take place in the inherited structure, $X \cup \{x, z_u\} \rightarrow X \cup \{y, z_u\}$ and $X \cup \{y, z_v\} \rightarrow X \cup \{x, z_v\}$. By Lemma 6, a connecting move between the local minors exists between $X \cup \{x, z_v\}$ and $X \cup \{x, z_u\}$ and thus we can compose all the local minors of Lemma 13 into a graph and edge contracting the connecting moves to produce the reconfiguration graph containing i subunits. □

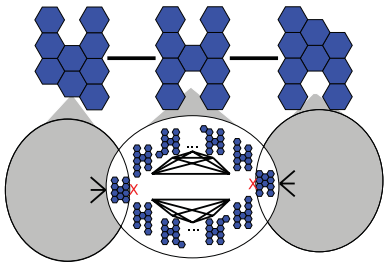

Fig. 17. A counterexample case for the Ghrist model. Two neighbors in the local structure of the central H configuration are shown (top). The induced local structure of the H configuration is divided into four connected components, two cliques and two unconnected vertices. The local structure cannot be reconstructed from edge deletions and contractions of the inherited structure, and thus is not a graph minor.

Remark 15. For an alternate viewpoint on the same result, we could entirely skip the composition of local graphs. An extra subunit can be added, and moved out of the way in order to realize all sequences of realizable moves (Lemma 12). However, this loses sight that there is a notion of locality relating the local structure to the inherited local structure through the embedding space. This becomes important when we consider the counter example for the Ghrist model.

Conjecture 16. *The Ghrist reconfiguration graph containing i units is not a graph minor of* $i + 1$ *when i is greater than some constant.*

Our above construction of graph minor for the Surface model does not hold for the Ghrist configuration graph because an additional subunit in the inherited local structures does not, in general, form a clique structure. Thus, while a location may exist for every local move that permits the move to take place in the inherited structure, there may not be connections between these locations. Figure 17 shows a counterexample where the inherited structure is disconnected. In these cases, the local structures are not minors of the inherited structures, and so a global minor cannot be constructed from a composition of local minors.

Interestingly, if the H configuration counterexample in Fig. 17 is used as a starting point for a subgraph sample for the procedure in Section 5, then the resulting subgraph yields an λ_2 of just 0.03. This classifies the configuration as the most bottlenecked configuration encountered. It appears that the areas of the reconfiguration space where the graph minor relation breaks down is also where bottlenecks appear.

Theorem 17. *For the metamodule state space,* $\mathbb{M}_1 \leq \mathbb{M}_2 \leq \ldots$.

Proof. Omitted for brevity, but the proof largely follows the logic for the Surface model. □

Decoupling the start and end positions of moves is the primary reason why minor ordering is found in the reconfiguration graphs of the easy planning spaces studied here. It must be noted though, that the minor ordering is a global structural property of the reconfiguration graphs, and not a consequence of the representation used to describe the motion catalogs.

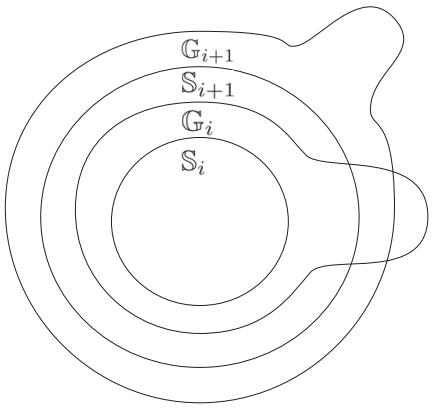

Fig. 18. Each Surface state space can be nested within the next. While a Surface state space is always found within a Ghrist state space, each Ghrist state space contains an area that is not contained with its child. It is for this reason that Ghrist problems cannot be solved efficiently over the entirety of the state space.

Graph Minor Theory is a powerful, modern mathematical tool. Many properties are preserved or bounded by taking minors. If a graph $H \leq G$ and G can be drawn in some topological space without edge crossings (e.g., a planar graph, or a generalization thereof) then H can be too. H is no more complex (in a topological sense) than G.

It may be initially difficult for a reader unfamiliar with graph minor theory to see the consequences of minor ordering. Recall the easier-to-grasp concept introduced first that if $H_k \leq G_k$, and H and G are distinct SRS motion models, then plans for the hardware of H can be run on the hardware of G, e.g., H could be a metamodularization of G. Generalizing this, we can now see that if $H_i \leq H_{i+1}$ that plans for the H_i reconfiguration graph can be instantiated on the same hardware, differing only in that an extra subunit has been added (H_{i+1}). Thus, plans in H_i can be reused, and augmented, to form plans in H_{i+1}, so planning in these well-ordered spaces can be achieved in an incremental, local and recursive manner.

In contrast, the hard planning spaces, like the Ghrist model, do not permit this efficient strategy. As $\mathbb{G}_i$ is not a minor of $\mathbb{G}_{i+1}$ this implies that both edge deletions and additions must be used to modify $\mathbb{G}_{i+1}$ into $\mathbb{G}_i$ (and in fact, vice versa). So, a plan that worked for a $\mathbb{G}_i$ planning task may not always operate in an analogous manner for the $\mathbb{G}_{i+1}$ case, because it may of utilized an edge in the reconfiguration graph that no longer exists.

Like the subset relation, the graph minor relation induces a partial order on a set of elements. Partial orders can be summarized graphically using a nested set notation. The observations in this section about how the Ghrist state spaces relate to the Surface state spaces and between themselves using the graph minor relation are summarized in Fig. 18.

7. Discussion

Self-reconfiguring systems are a desirable future robotic technology. Unfortunately, practical implementations of SRS tend to have awkward motion constraints that make planning computationally difficult. To get the full benefits of SRSs we require efficient motion planning algorithms so that SRS deployments can reconfigure on demand in response to environmental challenges.

So far, efficient motion planning algorithms have been developed on a somewhat *ad hoc* basis, wherein researchers have looked carefully at each instantiation of SRS architectures and carefully chosen motion catalog restrictions. So far, we have lacked a theoretical understanding of why some classes of SRS are good to plan within and some are not. Our work is an attempt to elucidate the structure of SRS reconfiguration spaces, which could be exploited in planning algorithms. We applied graph-theoretic techniques to sample the reconfiguration space in order to quantify the presence of bottlenecks, and we identified a graph property that separated an easy to plan with SRS model from a harder one. These are general methodologies with computational implications for a much larger class of SRS.

Metamodularization has been a common tactic in the SRS community for isolating troublesome motion constraints within an abstraction. Metamodularization often involves the definition of a tunneling procedure that allows a peripheral metamodule to appear anywhere else on the perimeter of the configuration. The unconstrained movement of metamodules around the perimeter using a tunneling procedure is similar to the Surface model's long-move motion primitives (though the Surface model does not permit movements to locations on the perimeter that cause Surface violations). Both metamodularization and the Surface model configuration graphs are well ordered by the graph minor relationship, which we believe goes some way towards explaining why these approaches make planning easier. It is clear that the Surface model's motion catalog is *far less restrictive* than the strategy offered by metamodularization though. The configurations adhering to the Surface model's constraints occupy a much larger volume of HMR configurations, and thus sacrifice less generality in the configurations that can be planned with efficiently than an alternative metamodularization approach would.

The reason why Surface constraints are significantly less restrictive is because they are defined as addition *local* constraints describing where a subunit cannot stop along a motion *path*. There is no restriction that the local constraints to be defined in global terms (e.g., at specific points on a globally defined grid spacing as in metamodularization). It seems entirely plausible that with a set of geometric path primitives (path segments and perhaps more generally branches), and with the insights of this paper (keeping algebraic connectivity high, and looking for graph minor ordering), that a set of *local* constraints that constrain an underlying model only a little, but simplify planning significantly, can be elucidated automatically.

Constraining a motion model only a little implies that only a small volume of the target general state space cannot be represented. In previous work,[12] we utilized the Surface state space as an efficient basis for long range planning across the more general Claytronics motion catalog, with occasional recourse to more expensive but general search methods. Although the overall algorithm targeted the Claytronics motion catalog, by finding a large subspace that was efficient to work within, the size of the "difficult" part of the remaining

space was greatly reduced. Thus, overall, the algorithm could achieve near linear performance, as empirically demonstrated using a large number of randomly generated configurations, over a large proportion of the target state space.

8. Conclusions

SRSs need efficient motion planning algorithms, but developing them has been difficult because of the inherent high dimensionality and complexity (due to motion and shape constraints) of the problem. An efficient SRS motion planning algorithm must exploit local and global structure. In this work we have shown that even in cases where the basic state space of a planning problem may be complex, specific subspaces may admit much more interesting structure that can be gainfully utilized for planning. We have made precise what structure is required of the subspace, and moreover, we have shown how one can characterize this structure using general and powerful mathematical tools that are applicable to a large class of SRS problems. One promising direction for future work is to try to utilize these conceptual ideas to develop techniques that automatically discover efficient subspaces from more complex self-reconfiguration models.

References

1. A. Abrams and R. Ghrist, "State complexes for metamorphic robot systems," *Int. J. Robot. Res.* **23**(7), 809–824 (2004).
2. Z. Butler, K. Kotay, D. Rus and K. Tomita, "Generic decentralized control for a class of self-reconfigurable robots," *Proceedings of the 2002 International Conference on Robotics and Automation (ICRA 2002)*, pp. 809–816.
3. G. S. Chirikjian, "Kinematics of a metamorphic robotic system," *Robotics and Automation, Proceedings., 1994 IEEE International Conference on*, vol. 1, pp. 449–455.
4. F. Chung, *Spectral Graph Theory* (American Mathematical Society, Providence, RI, 1997).
5. R. Diestel, *Graph Theory*, vol. 173 of *Graduate Texts in Mathematics*, 3rd. ed. (Springer-Verlag, Heidelberg, 2005).
6. R. Ghrist and V. Peterson, "The geometry and topology of reconfiguration," *Adv. Appl. Math.* **38**, 302–323 (2007).
7. M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.* **11**(1), 10–18 (2009).
8. L. E. Kavraki, P. Svestka, J.-C. Latombe and M. H. Overmars, In *Robotics and Automation, Proceedings., IEEE, International Conference on Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*, pp. 566–580 (1997).
9. B. T. Kirby, B. Aksak, J. D. Campbell, J. F. Hoburg, T. C. Mowry, P. Pillai and S. C. Goldstein, "A modular robotic system using magnetic force effectors," *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, pp. 2787–2793 (29 2007-Nov. 2 2007).
10. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science* **220**, 671–679 (1983).
11. T. Larkworthy, G. Hayes and S. Ramamoorthy, "General motion planning methods for self-reconfiguration planning," *Towards Auton. Robot. Sys.* (2009).
12. T. Larkworthy and S. Ramamoorthy, "An effecient algorithm for self-reconfiguration planning in a modular robot," In *Robotics and Automation, Proceedings., IEEE International Conference on*, pp. 5139–5146 (2010).
13. S. M. LaValle, *Planning Algorithms* (Cambridge University Press, Cambridge, U.K., 2006).
14. S. Murata, E. Yoshida, A. Kamimura, H. Kurokawa, K. Tomita and S. Kokaji, "M-tran: Self-reconfigurable modular robotic system," *Mechatron., IEEE/ASME Trans.* **7**(4), 431–441 (2002).
15. A. Pamecha, C. Chiang, D. Stein and G. Chirikjian, "Design and implementation of metamorphic robots," In *The ASME Design Engineering Technical Conference and Computers in Engineering Conference* (1996).
16. A. Pamecha, I. Ebert-Uphoff and G. Chirikjian, "Useful metrics for modular robot motion planning," *Robot. Automat., IEEE Trans.* **13**(4), 531–545 (Aug 1997).
17. R. J. Quinlan, *C4.5: Programs for Machine Learning* (Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993).
18. J. Reif and S. Slee, "Optimal kinodynamic motion planning for 2d reconfiguration of self-reconfigurable robots," *Robot. Sci. Syst.* (2007).
19. D. Rus and M. Vona, "Crystalline robots: Self-reconfiguration with compressible unit modules," *Auton. Robots* **10**(1), 107–124 (2001).
20. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995).