IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Triangle-Decomposable Graphs for Isoperimetric Robots

Nathan Usevitch[1], Isaac Weaver[1], and James Usevitch[2]

***Abstract*—Isoperimetric robots are large scale, untethered inflatable robots that can undergo large shape changes, but have only been demonstrated in one 3D shape- an octahedron. These robots consist of independent triangles that can change shape while maintaining their perimeter by moving the relative position of their joints. We introduce an optimization routine that determines if an arbitrary graph can be partitioned into unique triangles, and thus be constructed as an isoperimetric robotic system. We enumerate all minimally rigid graphs that can be constructed with unique triangles up to 9 nodes (7 triangles), and characterize the workspace of one node of each of these robots. We also present a method for constructing larger graphs that can be partitioned by assembling subgraphs that are already partitioned into triangles. This enables a wide variety of isoperimetric robot configurations.**

I. Introduction

Robotic systems will be able to perform a wider range of tasks if they can adapt their shape and safely interact with humans. One type of robot with the potential for large shape change and human-safe interaction is the isoperimetric robot, first introduced in [1], and with an example shown in Fig. 1. This robot consists of inflated fabric beams as the primary structural members, with robotic rollers that pinch the tube, reducing the local bending stiffness to create a region of the tube that acts as a rotational joint. These rollers can drive up and down the tube, simultaneously lengthening one edge of the tube and shortening another. The overall internal volume inside the tubes is conserved, meaning that for inflated beams, no source of compressed air is needed. Computationally, the resulting structure approximates a computer mesh defined by edges and nodes, and allows the robot to change shape. The soft structure of the robot also gives it inherent compliance that makes the robot human safe.

The only demonstrated 3D shape of an isoperimetric robot has been an octahedron, composed of 4 individual tubes that form connected triangles [1]. The fundamental building block for these robots in 3 dimensions is the triangle. In a triangle, the axis of rotation formed by each of the roller modules are guaranteed to remain parallel. These robots can be precisely

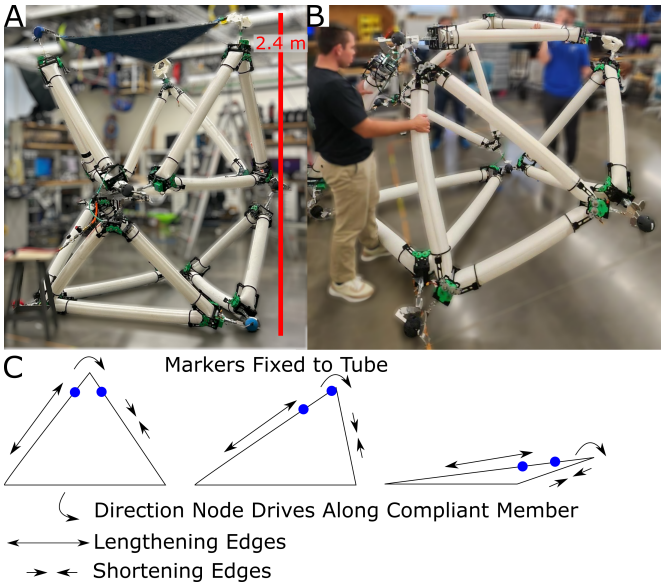

Fig. 1. (A) An isoperimetric robot constructed from 7 triangles (6 inflated triangles, and a rigid triangle that serves as an interaction surface. (B) An isoperimetric robot in the shape of a hexagonal bipyramid, formed from 6 triangles. (C) The operating principle of the isoperimetric robot. Each triangle changes shape as the rollers move along the inflated tube, changing the position of the vertex, but maintaining a constant perimeter.

modeled as graphs composed of interconnected triangles, allowing us to apply tools from graph-theory to systematically generate and analyze new configurations. Robots can be constructed by connecting multiple triangles at spherical joints, but the requirement that each edge belongs to exactly one triangle imposes strict constraints. This constraint raises a fundamental design question: What shapes can be formed using only triangles, such that each edge in the resulting graph belongs to exactly one triangle? This paper introduces theoretical tools to answer that question and identify a wide range of viable robot configurations. Two such robots have been physically constructed and are shown in Fig. 1.

This paper makes the following contributions:

- We present a novel algorithm that determines if a graph can be partitioned into individual triangles, the base unit of an isoperimetric robot, and finds a valid partition.
- We enumerate all possible minimally rigid graphs up to 9 nodes that can be partitioned into individual triangles, along with all possible partitions of those graphs. This enumerates all minimally rigid isoperimetric robots that can be constructed from 7 triangles.
- We present a constructive method for combining partitioned graphs together to form larger structures that themselves are triangle-decomposable, allowing construction of arbitrarily large isoperimetric robots.
- We give an analysis of the reachable set of points by

Manuscript received: March 19, 2025; Revised July 8, 2025; Accepted September 11, 2025.

This paper was recommended for publication by Editor Yong-Lae Park upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the Utah NASA Space Grant Consortium (NASA Grant No. 80NSSC20M0103), the 2024 NASA BIG Idea Challenge, and the U.S. National Science Foundation under Award No. 2501928.)

[1]Nathan Usevitch and Isaac Weaver are with the Ira A. Fulton School of Engineering, Mechanical Engineering Department, Brigham Young University, Provo nathan_usevitch@byu.edu

[2]James Usevitch is with the Ira A. Fulton School of Engineering, Electrical and Computer Engineering Department, Brigham Young University, Provo

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

minimally rigid isoperimetric robots up to 9 nodes.

II. Related Work

A. Modular Robotics

A robot composed of modular and reconfigurable elements would allow robotic systems to adapt to a wide variety of tasks and environments [2], [3]. Often the robots themselves are the primary building blocks of the structure [4], [5], while in other cases robots can form a shape from passive material ([6]).

The class of modular robots to which the Isoperimetric robots most closely belongs is truss robots, often referred to as variable geometry trusses. Truss robots consist of rigid links that can change their length connected at passive joints. The tetrobot project used as its base unit a tetrahedron, and developed hardware and dynamic controllers for this class of robots [7], [8], [9], [10]. The tetrobot project enumerates a number of different shapes that the tetrahedral robots can take, including an algorithm for chaining tetrahedon elements together to create arbitrarily long chains. Hardware construction and controller design of these tetrahedral based structures has also been considered for lunar operations [11], [12]. There have been several other hardware constructions of these truss style robots [13], [14]. In [15], the authors determine which shapes the truss robot can take while it reconfigure itself without user intervention. Truss robots are also closely related to tensegrity robots [16], which have been proposed for space exploration [17], [18], climbing through ducts [19], and other tasks. Determining how to combine tensegrity elements into large numbers of shapes has also been a topic of research [20], [21]. In the case of the isoperimetric robot, the fundamental element is a triangle, which poses the problem of what shapes can be formed from triangles. While other similar other works that have described and enumerated tensegrity, tetrahedral, and truss configurations, this work describes and enumerates isoperimetric robots.

B. Graph Decomposition

This work also draws on prior contributions in graph theory and discrete optimization. The problem of partitioning a graph into triangles is a special case of an edge partition problem, which has been shown to be NP-complete [22]. While extensive research has focused on characterizing rigid graphs, a full combinatorial characterization of rigidity in three dimensions is an open problem. In [23], the authors examine the number of embedding of minimally rigid graphs, and in doing so, enumerate all minimally rigid graphs up to 8 nodes. In constructing the graphs, the authors note that two operations, dubbed H1 and H2 steps and shown in Fig. 2, are sufficient to construct all Geiringer graphs with ≤ 12 vertices.

III. Mathematical Definition of an Isoperimetric Robot

We define the state of an isoperimetric robot using information about the connectivity of the edges (an undirected graph), the position of the vertices (the embedding) and how the edges are divided into triangles (the triangle partition). We denote the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of vertices or nodes and $\mathcal{E}$ is the set of edges. We define the embedded graph as a framework $(\mathcal{V}, \mathcal{E}, \boldsymbol{p})$ where $(\mathcal{V}, \mathcal{E})$ is the graph and $\boldsymbol{p} \triangleq \begin{bmatrix}\boldsymbol{p}_1^\intercal & \boldsymbol{p}_2^\intercal & \cdots & \boldsymbol{p}_n^\intercal\end{bmatrix}^\intercal$, $\boldsymbol{p}_i \in \mathbb{R}^3 \ \forall i$, is a set of points in Euclidean 3-dimensional space.

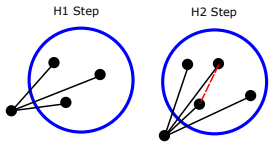

Fig. 2. Two different Hennenberg steps utilized to construct graphs. In an H1 step, a new node is added with 3 connecting edges. For an H2 step, a new node with four new edges is added, and an edge between the connecting nodes is deleted.

Assumption 1: All graphs considered in this paper do not have self-loops and do not have duplicate edges. More precisely, $(i,j) \in \mathcal{E} \implies i \neq j$ and any (undirected) edge $(i,j) \in \mathcal{E}$ is unique.

1) Triangle Partition: The goal is to partition the set of edges $E(\mathcal{G})$ into subsets, where each subset induces a subgraph isomorphic to K_3 (a triangle). To do this, we define the notions of binary-valued *triangle indicator matrices* and *triangle partition matrices*. For this definition, M^j indicates the jth column of the matrix M.

Definition 1: The matrix $T \in \{0,1\}^{|E| \times N_T}$, $N_T \in \mathbb{Z}_{\geq 0}$ is a triangle indicator matrix for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if every column T^j is the indicator vector for the edge set of a subgraph of $\mathcal{G}$ isomorphic to K_3.

Definition 2: The matrix $T \in \{0,1\}^{|E| \times |E|/3}$ is a triangle partition matrix for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ if all of the following conditions are satisfied:

1) T is a triangle indicator matrix, and
2) $T\vec{\mathbf{1}} = \vec{\mathbf{1}}$. Equivalently, the edge sets associated with each column of T form a partition of $\mathcal{E}$.[1]

Definitions 1, 2 imply that the entries of a triangle indicator or triangle partition matrix can be defined as

$$T_i^j = \begin{cases} 1 & \text{if edge } i \text{ is in triangle } j, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Each column T^j of T represents a different triangle and each row T_i of T represents an edge of the graph $\mathcal{G}$. Note that Definitions 1, 2 trivially imply that $T^\intercal \vec{\mathbf{1}} = 3\vec{\mathbf{1}}$, meaning that each triangle contains exactly three edges.

The kinematics of an isoperimetric robot relating the changes in the edge lengths with motions of the nodes are described in detail in [1]. For each graph, we can construct a rigidity matrix, $R(x) \in \mathbb{R}^{|\mathcal{E}| \times 3|\mathcal{V}|}$ can be used to determine whether or not the framework is infinitesimally rigid using the following result:

Lemma 1 ([24]): The framework $(\mathcal{V}, \mathcal{E}, \boldsymbol{p})$ is infinitesimally rigid in $\mathbb{R}^m$ if and only if its rigidity matrix $R(x)$ has rank $3n - 6$, where n is the number of nodes.

[1]The dimensions $T \in \{0,1\}^{|E| \times |E|/3}$ for a triangle partition matrix follow from every edge being included in exactly one triangle (with three edges per triangle).

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

IV. Partitioning a Graph into distinct Triangles

We present three separate algorithms to partition a graph into distinct triangles: an exhaustive search algorithm, an integer programming routine with a pre-enumeration step, and an integer programming algorithm.

For an arbitrary graph to be decomposed into unique triangles, it must satisfy the following necessary (but not sufficient) conditions:

- The degree of each node in the graph must be even
- The total number of graph edges is a multiple of 3

A. Exhaustive Search Routine

If a graph meets the necessary conditions, a naïve approach to determining if a graph can be partitioned into triangles is the following:

- Identify all triangles in the graph
- Select all possible combinations of triangles
- Evaluate each combination of triangles to determine if each edge in the graph appears exactly once.

The triangles in a graph are precomputed using algorithm 1. In this algorithm, we loop over each node in the graph, and look for nodes that have an intersection in their neighbor sets, where the neighbor set of node i is denoted $N(i)$

Algorithm 1 Triangle Pre-Enumeration

```
triangles ← ∅
for i = 0 : n do                                  ▷ Loop over each node
    for v ∈ N(i) do                               ▷ Loop over the nodes neighbors
        if v < i then                             ▷ Avoid duplicate checks
            for w ∈ N(i) ∩ N(v) do                ▷ Shared neighbors
                if w > v then                     ▷ Avoid duplicate triangles
                    triangles ← triangles ∪ {(i, v, w)}
return triangles
```

The computational complexity of this algorithm, and the upper bound on the number of triangles is given in [25] as

$$N_{Triangles} = \frac{1}{3}\sum_{i=1}^{n} \frac{d_i(d_i-1)}{2} \tag{2}$$

where d_i is the degree of node i. The number of possible combinations depends on the number of triangles in the graph. If there are m candidate triangles, and $3k$ edges in the graph, the number of possible combinations to check is given by the binomial coefficient:

$$N_{combinations} = \frac{n!}{(n-k)!k!}. \tag{3}$$

Thus the number of triangles in a graph grows quadratically with the degree of the nodes of the graph, but the number of possible combinations grows with the factorial of the number of triangles in the graph. This procedure is tractable for small graphs, but exhaustively enumerating and searching the possible combinations rapidly exceeds our computational resources for large, dense graphs. For this reason, we pursue an integer programming approach.

B. Integer Programming Formulations for Determining Triangulated Rigidity

We present two integer programming techniques for decomposing a graph into triangles. In the first, each triangle in the graph is identified as an initial step. In the second, the decomposition is solved end-to-end by an integer convex program without pre-enumeration.

1) Triangle Pre-Enumeration ILP: The triangle pre-enumeration technique consists of two steps. The first step precomputes all possible triangles within the input graph, using the procedure in Algorithm 1. The second step uses an integer linear programming solver to partition edges into a valid graph decomposition.

For the pre-enumeration technique, we encode all possible triangles with a triangle indicator matrix T as per Definition 1. Since the matrix T contains all possible triangles within the graph and is not guaranteed to be a triangle partition matrix, it follows that some triangles within T may have intersecting edge sets.

To obtain a triangle partition matrix from T as per Definition 2, all edges must be included in exactly one triangle. Mathematically this can be expressed as the constraint $Tx^* = 1$, where the jth entry of the indicator vector $x^* \in \{0,1\}^{N_T}$ is 1 if triangle j is included in the partition and 0 if it is excluded. The final partition matrix is then $T^* \triangleq \begin{bmatrix} T^{j_1} & T^{j_2} & \cdots & T^{j_m} \end{bmatrix}$ where $x_{j_k} = 1 \ \forall k = 1, \ldots, m$.

Finding the indicator vector x^* can be posed as an integer linear programming (ILP) feasibility problem as follows:

$$x^* = \underset{x \in \{0,1\}^{N_T}}{\arg\min} \quad 0 \quad \text{s.t.} \quad Tx = \mathbf{1} \tag{4}$$

Any feasible point to this ILP is a valid indicator vector specifying which columns of T form a triangle partition matrix.

The total runtime of the Triangle Pre-Enumeration ILP method is the runtime of Algorithm 1 added with the runtime of the ILP in (4). The ILP in (4) can be solved by any standard optimization solver such as Gurobi [26], MOSEK [27], HiGHS [28], or SCIP [29].

2) End-to-End IQCQP: The second integer programming technique involves computing a triangle partition matrix directly from the graph incidence matrix without a pre-enumeration step. Let $d = \begin{bmatrix} d_1 & \cdots & d_{|E|} \end{bmatrix}$ be the vector of degrees of the line graph $\mathcal{L}(\mathcal{G})$. Let D denote the incidence matrix of $\mathcal{G}$, and let $|D|$ denote the entrywise absolute value of the matrix D. Let $\mathcal{L}(\mathcal{G})$ denote the line graph of $\mathcal{G}$, and let L be the Laplacian matrix of $\mathcal{L}(\mathcal{G})$. We will show in this section that this problem can be expressed as the following integer quadratically-constrained quadratic program (IQCQP):

$$\begin{aligned}
T^* = \underset{T \in \{0,1\}^{|E| \times |E|/3}}{\arg\min} \quad & 0 \\
\text{s.t.} \quad & T^\intercal x = 3\vec{\mathbf{1}} && (5)\\
& Tx = \vec{\mathbf{1}} && (6)\\
& (T^j)^\intercal L(T^j) - d^\intercal T^j \leq -6 \quad \forall j = 1, \ldots, N_T^* && (7)\\
& |D|T^j \leq 2 \quad \forall j = 1, \ldots, N_T^* && (8)
\end{aligned}$$

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

The optimization variables of this IQCQP are the entries of the matrix $T \in \{0,1\}^{|E|\times|E|/3}$. The first constraint (eq. (5)) ensures the columns of T have exactly 3 non-zero entries, and the second constraint enforces each edge to be present in exactly 1 column. The third and fourth sets of constraints enforce each column of T^* to represent an indicator vector for an edge set of a subgraph of $\mathcal{G}$ isomorphic to K_3. The third set of constraints ((7) enforces that all edges in each triangle are adjacent to each other, but on its own is not sufficient to ensure that a given triangle T^j is isomorphic to K_3. [2] Eq. (8) enforces that no single node is part of all three edges.

Lemma 2: An induced triangle subgraph T^j with edges $j_1, j_2, j_3 \in E(\mathcal{G})$ is isomorphic to K_3 if and only if inequalities (7) and (8) are simultaneously satisfied.

Sufficiency: Suppose T^j is the indicator vector for the edge set of a triangle isomorphic to K_3. T^j can be written as $T^j = I^{j_1} + I^{j_2} + I^{j_3}$ where I^j represents the jth column of the identity matrix. Observe that

$$(I^i)^\intercal L I^k = L_i^k = \begin{cases} d_i & \text{if } i = k, \\ -1 & \text{if } i \neq k \text{ and edge } i \text{ is adjacent} \\ & \text{to edge } k, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Also observe that $d^\intercal T^j = d^\intercal(I^{j_1} + I^{j_2} + I^{j_3}) = d_{j_1} + d_{j_2} + d_{j_3}$. From the symmetry of L (recall that we are considering undirected graphs) it follows that

$$(T^j)^\intercal L(T^j) - d^\intercal T^j = 2(L_{j_1}^{j_2} + L_{j_1}^{j_3} + L_{j_2}^{j_3}). \tag{10}$$

Since T^j is the indicator vector for the edge set of a triangle isomorphic to K_3, all three edges j_1, j_2, j_3 are adjacent to each other and the RHS of (10) is equal to -6, satisfying constraint (7).

Next, since the triangle represented by T^j is isomorphic to K_3, there exist three nodes n_1, n_2, n_3 such that $|D|I^{j_1} = I^{n_1} + I^{n_2}$, $|D|I^{j_2} = I^{n_2} + I^{n_3}$, and $|D|I^{j_3} = I^{n_3} + I^{n_1}$. This follows from the definition of the incidence matrix D. We therefore have $|D|T^j = |D|(I^{j_1} + I^{j_2} + I^{j_3}) = 2(I^{n_1} + I^{n_2} + I^{n_3})$, which implies that $|D|T^j \leq 2\vec{\mathbf{1}}$ for all j. Constraint (8) is therefore satisfied.

Necessity: We prove the contrapositive. Suppose that T^j is the indicator vector for the edge set of a subgraph that is *not* isomorphic to K_3. The objective is to show that either constraint (7) or (8) is not satisfied. Since the subgraph represented by T^j contains three edges due to the constraint in eq. (5), the subgraph not being isomorphic to K_3 implies that the number of nodes within the subgraph is either 4, 5, or 6.[3]

Case 1: Suppose there exists a node n^* in the subgraph represented by T^j that belongs to all edges. Since there are only 3 edges, this implies that the number of nodes in the subgraph is 4. An example is given by the subgraph with edges (1,2), (1,3), (1,4) with $n^* = 1$. These graphs are clearly non-isomorphic to K_3 since they must contain four nodes. It follows that for the three edges j_1, j_2, j_3 in T^j we have $|D|I^{j_1} = I^{n_1} + I^{n_2}$, $|D|I^{j_2} = I^{n_1} + I^{n_3}$, and $|D|I^{j_3} = I^{n_1} + I^{n_4}$. Therefore $|D|T^j = 3I^{n_1} + I^{n_2} + I^{n_3} + I^{n_4}$, which implies that $\max\left(|D|T^j\right) = 3$ and therefore constraint (8) is not satisfied.

Case 2: Suppose that T^j represents a subgraph not isomorphic to K_3 and $\max\left(|D|T^j\right) < 3$. Since constraint (8) is feasible, we demonstrate that constraint (7) is not satisfied. Observe that, for this case, there must exist two non-adjacent edges j_1, j_2 in the subgraph. This can be easily verified by noting that the only possible subgraphs where all three edges are adjacent are K_3 or a graph matching the conditions of Case 1; i.e., $\max\left(|D|T^j\right) = 3$. Since there exist two non-adjacent edges, by (9) we have $(I^{j_1})^\intercal L I^{j_2} = L_{j_1}^{j_2} = 0$. By (10) we therefore have

$$\begin{aligned}(T^j)^\intercal L(T^j) - d^\intercal T^j &= 2(L_{j_1}^{j_2} + L_{j_1}^{j_3} + L_{j_2}^{j_3}) \\ &= 2(0 + L_{j_1}^{j_3} + L_{j_2}^{j_3}) > -6.\end{aligned} \tag{11}$$

Therefore constraint (7) is not satisfied, which concludes the proof.

V. Composition Algorithms

A. Enumeration from minimally rigid graphs

In this section we address the problem of enumeration: what are all of the minimally rigid graphs, up to a given number of nodes, that can be decomposed into triangles? This allows us to determine which graphs we can use to construct isoperimetric robots.

In [23], the authors provide an enumeration of all minimally rigid graphs based on Hennenberg steps up to 8 nodes. Extending this enumeration to an exhaustive enumeration of graphs with 9 nodes requires a prohibitive amount of computational resources. However, the only graphs that can have valid partitions are those that have all nodes with even degree. If the final step is an H1 step, the last node will always have had odd degree, and cannot be partitioned. The only viable partitions for 9 nodes graphs will be encountered if the degree of each node is even. Therefore, we extend their enumeration to 9 nodes by taking all known graphs with 8 nodes, and evaluating all 9 node graphs that result from taking an H2 step.

Using the enumeration of minimally rigid graphs, we then apply the exhaustive search algorithm from the previous section to identify which are decomposable. The total numbers of the candidate graphs are given in table V-A for each number of nodes. Fig. 3 shows the graphs for which a successful partition was found. We also list the workspace of one node of each of these graphs, which we will describe in VII. In order to describe these graphs, we utilize a convention used by other researchers to describe arbitrary graphs. We take the upper triangular portion of the adjacency matrix, interpret that string of 1's and 0's as a binary number, and label the graph with the number that corresponds to the

[2] As a counterexample, consider a 4-node graph with edges $(1,2),\ (1,3),\ (1,4)$.

[3] This follows because a subgraph with three nodes and three edges must trivially be isomorphic to K_3.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

# of Nodes	MR Graphs	Satisfy NC	Partitions
6	4	1	2
7	26	2	1
8	374	6	3
9	-	60	13

TABLE I

The number of minimally rigid graphs for different number of nodes, how many of those graphs satisfy the necessary conditions, and the total number of partitions (note that some graphs that satisfy the necessary conditions can be partitioned in multiple ways).

value of the binary number converted to base 10. Thus the number assigned to each graph uniquely describes the graph's adjacency matrix.

Interestingly, there are some graphs for which multiple partitions of the same graph are possible. Graph 26622, an octahedron, has two possible partitions, one with a labeled triangle on the top, and another with a labeled triangle as the base. There are three possible partitions of two stacked octahedrons (graph 60243677150). Although the graphs are structurally identical, the robot moves differently depending on how the graph is partitioned into triangles, due to the constraints those partitions impose. This is evidenced by the different sizes of their workspaces.

a) Embedding the Graphs: Once the graphs are constructed, we must then find an embedding (assign values to the nodes coordinates of the graph) so that we can examine them as robotic systems. In this study we find embeddings using a multidimensional scaling [30], as implemented by the Matlab function "mdscale."

One challenge is that the embedding that results from the multidimensional scaling for some of these graphs occur at configurations where the graph is not infinitesimally rigid. Mathematically, the robot is not infinitesimally rigid when the rigidity matrix, R, loses row rank. We quantify the rigidity of a graph with the worst case rigidity index, λ_{WCR} formulated in [31], [32]:

$$\lambda_{WCR} = \frac{\lambda_7}{\sum_{i=1}^{3n} \lambda_i} = \frac{\lambda_7}{\text{tr}(R(x)^T R(x))} = \frac{\lambda_7}{\sum_{i=1}^{N_L} (L(x)_i)^2} \tag{12}$$

where λ_7 is the seventh smallest eigenvalue of the matrix $R(x)^T R(x)$. In order to find embeddings that are infinitesimally rigid, we use the algorithm in V-A0a. With this approach, we generate a large number of embeddings of the graph with some induced randomness, and then select the one that has the largest worst-case rigidity index.

Algorithm 2 Determine a Embedding

```
WCR ← 0
for i = 0 : n_trials do          ▷ Number of Embeddings to Try
    D = 0                        ▷ Initialize Distance Matrix
    for i ∈ 1 : n do
        for j ∈ [j : n] do
            if A(i, j) == 1 then ▷ Connected nodes
                D(i, j) = rand()
            else
                D(i, j) = 10     ▷ Disjoint nodes
    x=mdscale(D)                 ▷ Compute embedding
    WCR=wcr(x)                   ▷ Eq. 12
    if WCR > WCR_max then
        x_max = x
return x_max
```

B. Constructive Methods for Combining Partitioned Graphs

We present a method for combining two infinitesimally rigid graphs that are partitioned into edge-disjoint triangles, into a single graph that is also infinitesimally rigid and decomposes into unique triangles. Conceptually, this involves deleting a labeled triangle from one graph, and merging the 3 nodes of that triangle with three nodes of the other graph

Let $\mathcal{G}_1 = (\mathcal{U}_1, \mathcal{E}_1)$, with node locations $x_1 = (p_1, ..., p_{n_1})$ and $\mathcal{G}_2(\mathcal{V}_1, \mathcal{E}_2)$ with node positions $x_2 = (q_1 ... q_{n_2})$ be infinitesimally rigid frameworks in $\mathbb{R}^3$. Let there be a triangle (a subgraph isomorphic to K_3) in ($\mathcal{G}_1$ composed of nodes u_1, u_2, u_3, and edges e_{12}, e_{23}, e_{13}. Let $\mathcal{G}_2$ have 3 nodes positioned at the same relative distances as the triangle in $\mathcal{G}_1$, meaning that a homogeneous transform M exists such that $[p_1, p_2, p_3] = M[q_1, q_2, q_3]^T$. We define the graph $\mathcal{G}_{1'} = \mathcal{G}_1 - \{e_{12}, e_{23}, e_{13}\}$, meaning that the graph $\mathcal{G}_{1'}$ is the graph resulting from deleted the connecting edges of the triangle.

Lemma 3: Let $\mathcal{G}_{combine}$ be the graph formed by joining graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, i.e., $\mathcal{G}_{combine} = \mathcal{G}_{1'} \cup \mathcal{G}_2$ with $\{v_1 \sim u_1, v_2 \sim u_2, v_3 \sim u_3\}$, where $\sim$ denotes combining the nodes of the two graphs. Let $x_c = [p_1 ... p_{n_1}, Mq_4 ... Mq_{n_2}]$ be the node positions for $\mathcal{G}_{combine}$. Then the framework $(\mathcal{G}_{combine}, x_c)$ is infinitesimally rigid.

The rigidity matrix of ($\mathcal{G}_1$, x_1) is

$$R_1 = \begin{bmatrix} R_C & 0 \\ R_{1C} & R_{11} \end{bmatrix}. \tag{13}$$

where R_C is the rigidity matrix of the deleted triangle. Let R_2 be the rigidity matrix of $(\mathcal{G}_2, x_2)$ The rigidity matrix of $(\mathcal{G}_{combine}, x_c)$ is

$$R_{combine} = \begin{bmatrix} R_2 & 0 \\ R_{1C} & R_1 \end{bmatrix} \tag{14}$$

The deleted edges corresponding to the triangle R_c are linearly dependent with the rows of R_2, because the framework $(\mathcal{G}_2, x_2)$ is infinitesimally rigid. Thus the rank of $R_{combine}$ is $(3n_1 - 6 + 3n_2 - 6 - 3)$. As $n_C = n_1 + n_2 - 3$, the rank of $R_{combine}$ is $3n_C - 6$. Thus the framework $(\mathcal{G}_{combined}, x_c)$ is infinitesimally rigid by Lemma 1.

If the triangle at which the two graphs are joined is part of the partition in each subgraph, then the overall graph can also be partitioned. Using this constructive algorithm, we can combine any of the graphs presented in Fig.3. We have constructed several different graphs using this constructive algorithm, with the results shown in Fig. 4.

VI. Results: Computational Timing

We evaluate the computational performance of the three partition algorithms presented in Section IV. We generate a

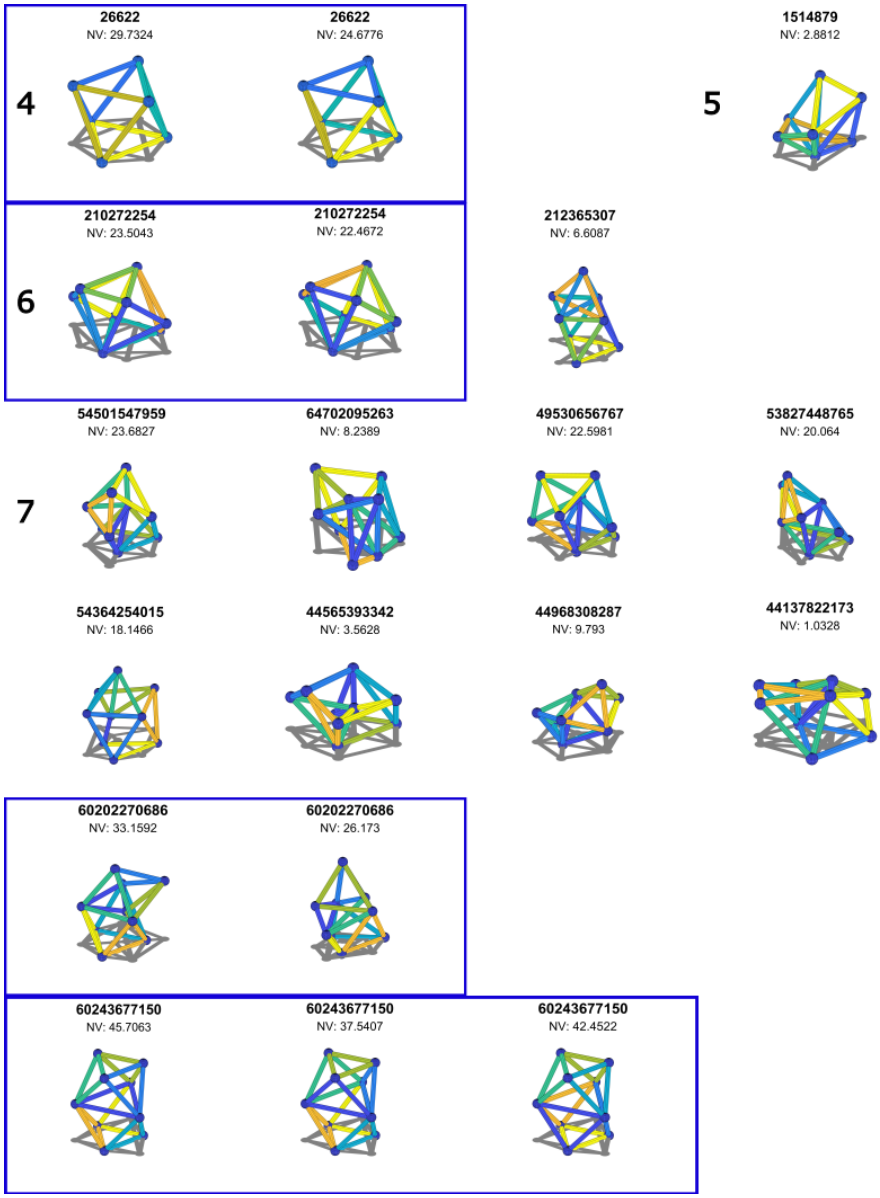

Fig. 3. All partitions of all minimally rigid graphs with 9 or fewer nodes that can be partitioned into triangles. Note that some graphs have multiple possible partitions. Graphs are grouped according to the number of component triangles. Graphs that are identical but have different partitions are grouped in blue boxes. For each graph, the normalized workspace volume NV of the top node is given.

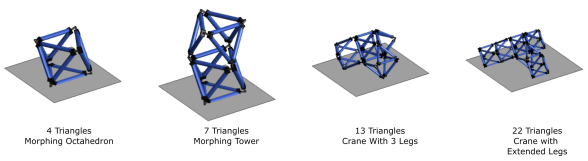

Fig. 4. Shapes formed by combining octahedral units into chains and branching shapes

test set of random graphs by starting with an octahedron, identifying three random nodes in the graph, and merging the base graph with a new octahedron using the procedure in Lemma 3. Denoting the number of nodes in the graph as $|\mathcal{V}|$, 20 random graphs were generated for each value of $|\mathcal{V}|$ in $[9, 15, 21, \ldots, 297]$. For each graph, the Exhaustive Search Routine, Triangle Pre-Enumeration ILP, and End-to-End IQCQP algorithms were run and the runtimes were recorded. Due to computational limitations a 100-second time limit was imposed on the runtimes. Integer programs were solved using the Gurobi optimizer [26]. All experiments were conducted on an AMD Ryzen Threadripper Pro 5975wx workstation processor.

A plot with the results is shown in Figure 5. The exhaustive search reaches the memory limit beginning with graphs of 18 nodes, while the end-to-end IQCQP reaches the time limit beginning at 24 nodes. However, the algorithm with pre-enumeration enables computation up to 297 nodes in approximately 1 second, indicating that this method is tractable for large graphs.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

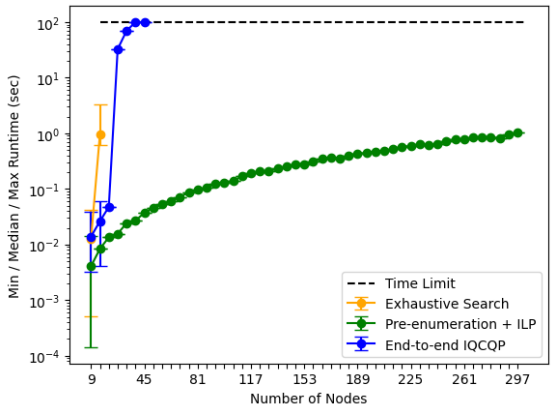

Fig. 5. Plot comparing runtimes for exhaustive search, pre-enumeration ILP, and end-to-end IQCQP. Center dots show the median runtime, while upper and lower whiskers show min / max runtimes respectively. All experiments ran with a time limit of 100 seconds. Only two entries for exhaustive search are shown since the algorithm encountered out-of-memory errors after that point. End-to-end IQCQP runtimes beyond 27 nodes are omitted due to runs hitting the timeout limit.

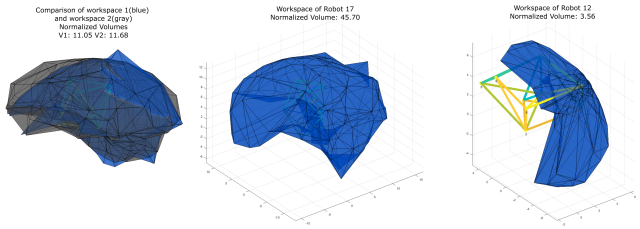

Fig. 6. Visualization of three robot workspaces: (1) Overlap of two robots with identical graphs, (2) largest achievable workspace, and (3) smallest constrained workspace.

VII. Results: Workspace Characterization

We now characterization the motion of the new robot shapes given in Fig. 3 to understand how these new decomposable graphs behave as isoperimetric robots. We provide a quantitative comparison by determining the reachable workspace of a single vertex of each robot, while three other "base nodes" are constrained to be stationary. The support nodes we selected are shown as the base nodes in Fig. 3.

Determining the workspace of each robot requires solving the inverse kinematics, meaning that we specify the motion of the controlled nodes of the graph in operational space, and determine how the robot must actuate to achieve this motion. We phrase the inverse kinematics as the following optimization problem:

$$\min_{\dot{x}_i} \|R(x)\dot{x}\|^2 \tag{15}$$

subject to

$$\begin{bmatrix} A_i \\ C_i \\ T^T R(x) \end{bmatrix} \dot{x}_i = \begin{bmatrix} b_i \\ 0 \\ 0 \end{bmatrix} \tag{16}$$

$$D\dot{x}_i \geq 0 \tag{17}$$

where $\dot{x}_i$ is the velocity of all nodes of the robot at time i. In this case, $C\dot{x} = 0$ constrains the base nodes of the robot to be stationary, $A\dot{x} = b$ constrains one of the nodes to move in a specified target direction. $T^T R(x)\dot{x} = 0$ is the constraint that the perimeter of each triangle remains constant, where T is the triangle indicator matrix of the graph.

We also include the inequality constraint $D\dot{x}_i \geq 0$. We move the robot while enforcing the following constraints:

- Individual edge lengths must remain above a minimum length of 0.2
- The robot's worst case rigidity index must remain above a threshold of 0.005, as defined in eq. 12.

After each time step we evaluate these constraints. If violated, we return to the previous time step and enforce the gradient of the constraint as a linear constraint of the form $D\dot{x}_i \geq 0$, using the same technique used in [33].

To calculate the workspace, we generated a set of 200 points on the surface of a sphere of radius 6, significantly larger than the robot can reach. We compute the convex hull of these points to find a triangulation Each robot was positioned within the sphere and commanded to move its designated movement node in a straight line to each target point on the sphere's surface. For each attempt, the final position reached by the movement node was recorded until the robot could no longer continue without violating the constraints. These final positions, along with the triangulation of the initial point locations, was used to compute the volume of the workspace. This procedure is visualized in the video attachment.

To allow comparisons between robots with different number of nodes and lengths of edges, the workspace volume was normalized by dividing it by the cube of the longest edge length of each robot. This normalization controls for differences in scale and initial edge length, enabling a fair comparison of workspace across robots with differing geometries and triangle counts.

Robots with identical graphs exhibit variability in their workspaces due to differences in how they are partitioned into triangles. This can be seen in the boxed groupings in Figure 3. The most drastic case is that of graph 60243677150 where the workspace ranges from 37.54 to 45.7. Similar differences occur in graph 26622. Robots with edges that span through the middle of the graph tend to have limited ranges of motion. These graphs in their initial configuration are closer to singularities. Graph 26622 has an initial WCRI of 1.67, which is the highest of all the graphs we explored. Graph 44565393342 had the smallest initial WCRI of 0.01. The initial WCRI does not directly correlate to the workspace volume. Robots based on the same graph have the same WCRI regardless of their partition, yet have different workspace volumes. Though 60243677150 had the largest workspace, it has a comparably small initial WCRI of 0.47.

A key design insight for isoperimetric robots is that the arrangement of triangles within a robot affects performance significantly, even for two robots where the graph is the same. For the physically constructed stacked octahedron shown in Fig. 1, we selected an embedding that positioned a triangle at the top to accommodate a solar panel. This specific partitioning of triangles resulted in the lowest workspace volume

for that configuration. In future iterations, it is important to consider how the overall geometry and the configuration of triangular components impact the system's effectiveness.

VIII. Conclusion

In this paper we have expanded the number of isoperimetric robots that can be built by characterizing triangle decomposable graphs. This allows for an increased number of robot shapes and types. These robots may form a viable candidate for space exploration, as they can stow in small volume when deflated, and then inflate to form large structures that can support substantial loads.

This algorithm could also be extended to decompose graphs into shapes beyond triangles. For example, it can partition a graph into sets of more complex base units such as octahedra. However, the current robot configuration requires base units to be planar. In future work, new hardware development could lead to nodes for the isoperimetric robot that allow for the compliant members to bend along multiple axes. This would remove the constraint that each tube must remain a planar triangle, and replace it with the constraint that the graph must be decomposed into Euler paths. This would increase the number of candidate robot configurations.

References

[1] N. S. Usevitch, Z. M. Hammond, M. Schwager, A. M. Okamura, E. W. Hawkes, and S. Follmer, "An untethered isoperimetric soft robot," *Science Robotics*, vol. 5, no. 40, 2020.

[2] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[3] J. Seo, J. Paik, and M. Yim, "Modular reconfigurable robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 63–88, 2019.

[4] M. Rubenstein, C. Ahler, and R. Nagpal, "Kilobot: A low cost scalable robot system for collective behaviors," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 3293–3298.

[5] J. W. Romanishin, K. Gilpin, and D. Rus, "M-blocks: Momentum-driven, magnetic modular robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 4288–4295.

[6] K. H. Petersen, R. Nagpal, and J. K. Werfel, "Termes: An autonomous robotic system for three-dimensional collective construction," *Robotics: Science and Systems VII*, 2011.

[7] G. J. Hamlin and A. C. Sanderson, "Tetrobot modular robotics: Prototype and experiments," *in Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 390–395, 1996.

[8] ——, "Tetrobot: A modular approach to parallel robotics," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 42–50, 1997.

[9] W. H. Lee and A. Sanderson, "Dynamic rolling locomotion and control of modular robots," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 32–41, 2002.

[10] W. H. Lee and A. C. Sanderson, "Dynamics and distributed control of tetrobot modular robots," *IEEE International Conference on Robotics and Automation*, vol. 4, pp. 2704–2710, 1999.

[11] S. Curtis *et al.*, "Tetrahedral robotics for space exploration," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 6, pp. 22–30, 2007.

[12] M. Abrahantes, A. Silver, and L. Wendt, "Gait design and modeling of a 12-tetrahedron walker robot," *Proc. IEEE Southeastern Symposium on System Theory*, pp. 21–25, 2007.

[13] A. Lyder, R. F. M. Garcia, and K. Stoy, "Mechanical design of odin, an extendable heterogeneous deformable modular robot," *IEEE/RSJ Intelligent Robots and Systems*, pp. 883–888, 2008.

[14] J. C. Zagal, C. Armstrong, and S. Li, "Deformable octahedron burrowing robot," *In Proc. Int. Conf. on the Synthesis and Simulation of Living Systems*, pp. 431–438, 2012.

[15] A. Spinos, D. Carroll, T. Kientz, and M. Yim, "Topological reconfiguration planning for a variable topology truss," *Journal of Mechanisms and Robotics*, vol. 13, no. 4, p. 040902, 2021.

[16] D. S. Shah, J. W. Booth, R. L. Baines, K. Wang, M. Vespignani, K. Bekris, and R. Kramer-Bottiglio, "Tensegrity robotics," *Soft robotics*, vol. 9, no. 4, pp. 639–656, 2022.

[17] J. Bruce, A. P. Sabelhaus, Y. Chen, D. Lu, K. Morse, S. Milam, K. Caluwaerts, A. M. Agogino, and V. SunSpiral, "Superball: Exploring tensegrities for planetary probes," in *12th International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, 2014.

[18] A. P. Sabelhaus, J. Bruce, K. Caluwaerts, P. Manovi, R. F. Firoozi, S. Dobi, A. M. Agogino, and V. SunSpiral, "System design and locomotion of superball, an untethered tensegrity robot," in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2867–2873.

[19] J. Friesen, A. Pogue, T. Bewley, M. de Oliveira, R. Skelton, and V. Sunspiral, "Ductt: A tensegrity robot for exploring duct systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 4222–4228.

[20] O. Aloui, D. Orden, and L. Rhode-Barbarigos, "Generation of planar tensegrity structures through cellular multiplication," *Applied Mathematical Modelling*, vol. 64, pp. 71–92, 2018.

[21] Y. Wang, X. Xu, and Y. Luo, "Topology design of general tensegrity with rigid bodies," *International Journal of Solids and Structures*, vol. 202, pp. 278–298, 2020.

[22] I. Holyer, "The np-completeness of some edge-partition problems," *SIAM Journal on Computing*, vol. 10, no. 4, pp. 713–717, 1981.

[23] E. Bartzos, I. Z. Emiris, J. Legerskỳ, and E. Tsigaridas, "On the maximal number of real embeddings of spatial minimally rigid graphs," in *Proceedings of the 2018 ACM International Symposium on Symbolic and Algebraic Computation*, 2018, pp. 55–62.

[24] R. Connelly, "Rigidity," in *Handbook of convex geometry*. Elsevier, 1993, pp. 223–271.

[25] S. Suri and S. Vassilvitskii, "Counting triangles and the curse of the last reducer," in *Proceedings of the 20th international conference on World wide web*, 2011, pp. 607–614.

[26] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[27] M. ApS, *The MOSEK optimization toolbox for MATLAB manual. Version 10.1.*, 2024. [Online]. Available: http://docs.mosek.com/latest/toolbox/index.html

[28] Q. Huangfu and J. J. Hall, "Parallelizing the dual revised simplex method," *Mathematical Programming Computation*, vol. 10, no. 1, pp. 119–142, 2018.

[29] S. Bolusani *et al.*, "The SCIP Optimization Suite 9.0," Optimization Online, Technical Report, February 2024. [Online]. Available: https://optimization-online.org/2024/02/the-scip-optimization-suite-9-0/

[30] T. F. Cox and M. A. Cox, *Multidimensional scaling*. CRC press, 2000.

[31] D. Zelazo, A. Franchi, F. Allgöwer, H. H. Bülthoff, and P. R. Giordano, "Rigidity maintenance control for multi-robot systems," *Robotics: Science and Systems*, pp. 473–480, 2012.

[32] D. Zelazo, A. Franchi, H. H. Bülthoff, and P. Robuffo Giordano, "Decentralized rigidity maintenance control with range measurements for multi-robot systems," *The International Journal of Robotics Research*, vol. 34, no. 1, pp. 105–128, 2015.

[33] N. S. Usevitch, Z. M. Hammond, and M. Schwager, "Locomotion of linear actuator robots through kinematic planning and nonlinear optimization," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1404–1421, 2020.