



Graph Threading

Erik D. Demaine  

Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, USA

Yael Kirkpatrick  

Department of Mathematics, Massachusetts Institute of Technology, USA

Rebecca Lin  

Computer Science and Artificial Intelligence Lab, Massachusetts Institute of Technology, USA

Abstract

Inspired by artistic practices such as beadwork and himmeli, we study the problem of *threading* a single string through a set of tubes, so that pulling the string forms a desired graph. More precisely, given a connected graph (where edges represent tubes and vertices represent junctions where they meet), we give a polynomial-time algorithm to find a minimum-length closed walk (representing a threading of string) that induces a connected graph of string at every junction. The algorithm is based on a surprising reduction to minimum-weight perfect matching. Along the way, we give tight worst-case bounds on the length of the optimal threading and on the maximum number of times this threading can visit a single edge. We also give more efficient solutions to two special cases: cubic graphs and the case when each edge can be visited at most twice.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases Shortest walk, Eulerian cycle, perfect matching, beading

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.37

Related Version *arXiv Version*: <https://arxiv.org/abs/2309.10122>

Funding *Yael Kirkpatrick*: NSF Graduate Research Fellowship under Grant No. 2141064

Rebecca Lin: MIT Stata Family Presidential Fellowship

Acknowledgements We thank Anders Aamand, Kiril Bangachev, Justin Chen, Alison Martin, Surya Mathialagan, and Zhecheng Wang for insightful discussions. We also thank anonymous reviewers for their helpful comments.

1 Introduction

Various forms of art and craft combine tubes together by threading cord through them to create a myriad of shapes, patterns, and intricate geometric structures. In beadwork [11], artists string together beads with thread or wire. In traditional ‘straw mobile’ crafts [19] — from the Finnish and Swedish holiday traditions of himmeli [4, 13] to the Polish folk art of pajęki [18] — mobile decorations are made by binding straws together with string. Artist Alison Martin has shown experiments where bamboo connected by strings automatically forms polyhedral structures by pulling the strings with a weight [15].

For engineering structures, these techniques offer a promising mechanism for constructing reconfigurable or deployable structures, capable of transforming between distinct geometric configurations: a collection of tubes, loosely woven, can be stored in compact configurations and then swiftly deployed into desired target geometric forms, such as polyhedra, by merely pulling a string taut. Figure 1 shows a prototype of such a structure, illustrating the potential of this approach. The popular ‘push puppet’ toy, originally invented by Walther Kourt Wals in Switzerland in 1926 [17], also embodies this mechanism.

In contrast to related work [12, 14], we study a *theoretical* formulation of these ideas: threading a single string through a collection of tubes to mimic the connectivity of a given



© Erik D. Demaine, Yael Kirkpatrick, and Rebecca Lin;
licensed under Creative Commons License CC-BY 4.0

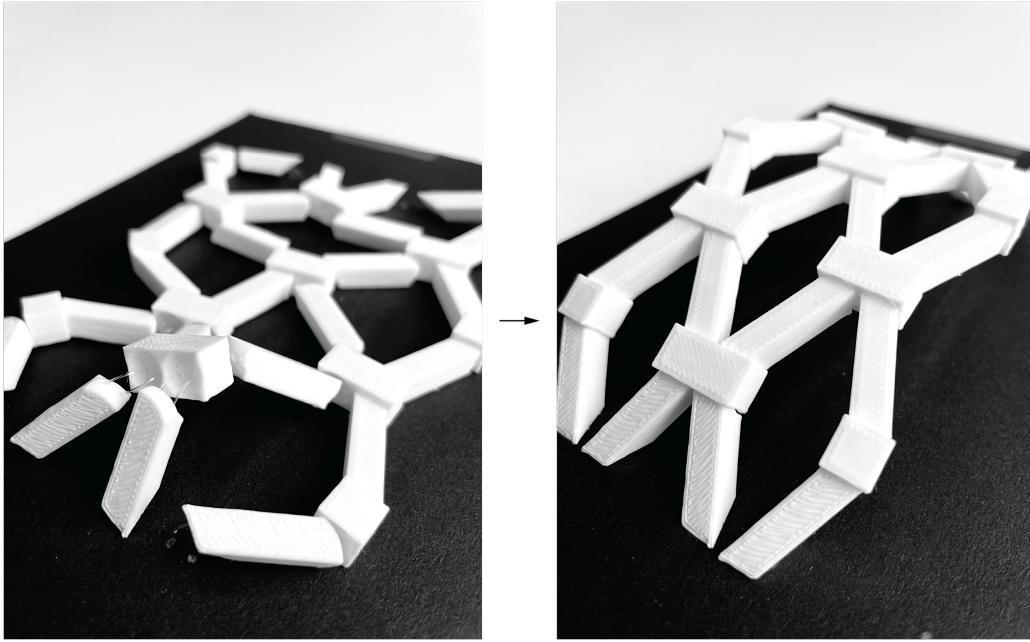
15th Innovations in Theoretical Computer Science Conference (ITCS 2024).

Editor: Venkatesan Guruswami; Article No. 37; pp. 37:1–37:18

Leibniz International Proceedings in Informatics



LIPIC Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



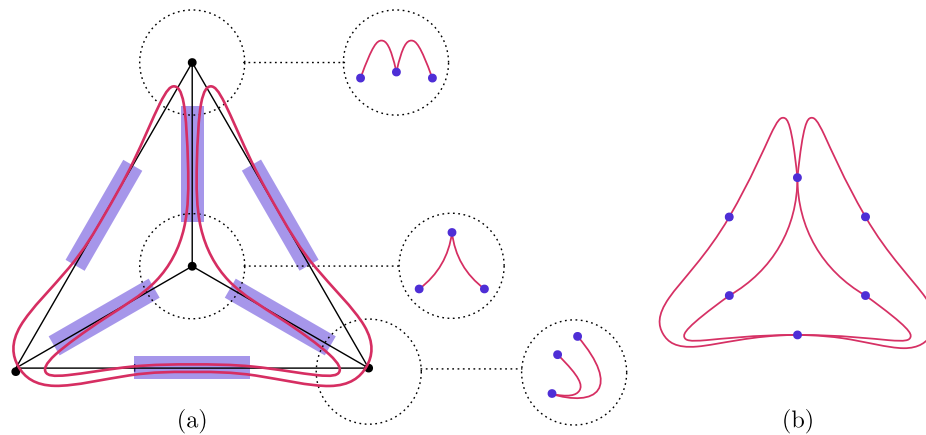
■ **Figure 1** A deployable structure made from disconnected 3D-printed elements (white) connected by string, which automatically shifts between soft (left) and rigid (right) states by pulling on the endpoints of the string beneath the platform (black). This design was developed by the third author in collaboration with Tomohiro Tachi.

44 graph; refer to Figure 2. Consider a connected graph $G = (V, E)$ with minimum vertex
 45 degree 2, where each edge $e \in E$ represents a tube and each vertex $v \in V$ represents the
 46 junction of tubes incident to v . A *graph threading* T of G is a closed walk through G that
 47 visits every edge at least once, induces connected “junction graphs”, and has no ‘U-turns’.
 48 The *junction graph* $J(v)$ of a vertex v induced by a closed walk has a vertex for each
 49 tube incident to v , and has an edge between two vertices/tubes every time the walk visits v
 50 immediately in between traversing those tubes.

51 A threading T of G must have a connected junction graph $J(v)$ for every vertex $v \in V$,
 52 and must have no *U-turns*: when exiting one tube, the walk must next enter a different
 53 tube. Define the *length* $|T|$ of T to be the total length of edges visited by T . For simplicity,
 54 we assume for much of our study that edges (tubes) have unit length — in which case $|T|$
 55 is the number of edge visits made by T — and then generalize to the weighted case with
 56 arbitrary edge lengths.

57 **Our Results.** In this paper, we analyze and ultimately solve the OPTIMAL THREADING
 58 problem, where the goal is to find a minimum-length threading T of a given graph G . Our
 59 results are as follows.

- 60 ■ In Section 2, we give a local characterization of threading in terms of local (per-vertex
 61 and per-edge) constraints that help us structure our later algorithms and analysis.
- 62 ■ In Section 3, we prove tight worst-case bounds on two measures of an optimal threading T .
 63 First, we analyze the minimum length $|T|$ in a graph with unit edge lengths, proving that
 64 $2m - n \leq |T| < 2m$ where m and n are the numbers of edges and vertices, respectively,
 65 and that both of these extremes can be realized asymptotically. Second, we prove that



■ **Figure 2** (a) The closed walk (red) on the graph (black) of a tetrahedron induces junction graphs (circled on the right) that are connected, and so it is a threading. (b) The union of junction graphs is called the *threading graph* (Section 2.2).

66 T traverses any one edge at most $\Delta - 1$ times, where Δ denotes the maximum vertex
 67 degree in G , and that this upper bound can be realized. The second bound is crucial for
 68 developing subsequent algorithms.

69 ■ In Section 4, we develop a polynomial-time algorithm for OPTIMAL THREADING, even
 70 with arbitrary edge lengths, by a reduction to minimum-weight perfect matching.

71 ■ In Section 5, we develop more efficient algorithms for two scenarios: OPTIMAL THREAD-
 72 ING on cubic graphs, and DOUBLE THREADING, a constrained version of OPTIMAL
 73 THREADING where the threading T is allowed to visit each edge at most twice.

74 2 Problem Formulation

75 Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges. Assume until
 76 Section 4.2.2 that G 's edges have unit length. Recall that a *threading* of G is a closed walk
 77 through G that has no U-turns and induces a connected junction graph at each vertex. As
 78 an alternative to this ‘global’ definition (a closed walk), we introduce a more ‘local’ notion
 79 of threading consisting of constraints at each edge and vertex of the graph and prove its
 80 equivalence to threading.

81 Before giving the formal definition of ‘local threading’, we give the intuition. A local
 82 threading assigns a nonnegative integer $x_{uv} \in \mathbb{N}$ for each edge $uv \in E$, which counts the
 83 number of times the threading visits or *threads* edge uv ; we refer to x_{uv} as the *count* of uv .
 84 These integers are subject to four constraints, which we give an intuition for by arguing that
 85 they are necessary conditions for a threading. First, each uv must be threaded at least once,
 86 so $x_{uv} \geq 1$ for all $uv \in E$. Second, a threading increments the count of *two* edges at junction
 87 v every time it traverses v , so the sum of counts for all edges incident to v must be even.
 88 Third, forbidding U-turns implies that, if uv is threaded k times, then the sum of counts for
 89 the remaining edges incident to v must be at least k to supply these visits. Fourth, because
 90 the junction graph $J(v)$ of v is connected, it has at least enough edges for a spanning tree —
 91 $d(v) - 1$ where $d(v)$ denotes the degree of v — so the sum of counts of edges incident to v
 92 must be at least $2(d(v) - 1)$. More formally:

37:4 Graph Threading

93 ► **Definition 1** (Local Threading). Given a graph $G = (V, E)$, a **local threading** of G consists
94 of integers $\{x_{uv}\}_{uv \in E}$ satisfying the following constraints:

- 95 (C1) $x_{uv} \geq 1$ for all $uv \in E$;
96 (C2) $\sum_{u \in N(v)} x_{uv} \equiv 0 \pmod{2}$ for all $v \in V$;
97 (C3) $\sum_{w \in N(v) \setminus \{u\}} x_{wv} \geq x_{uv}$ for all $uv \in E$; and
98 (C4) $\sum_{u \in N(v)} x_{uv} \geq 2(d(v) - 1)$ for all $v \in V$.

99 OPTIMAL LOCAL THREADING (minimizing $\sum_{uv \in E} x_{uv}$) is, in fact, an integer linear
100 program, though this is not helpful algorithmically because integer programming is NP-
101 complete. Nonetheless, local threading will be a useful perspective for our later algorithms.

102 The observations above show that any threading T induces a local threading by setting each
103 count x_{uv} to the number of times T visits edge uv , with the same length: $|T| = \sum_{uv \in E} x_{uv}$.
104 In the following theorem, we show the converse and, thus, the equivalence of threadings with
105 local threadings:

106 ► **Theorem 2.** We can construct a threading T of G from a local threading $\{x_{uv}\}$ of G such
107 that T visits edge uv exactly x_{uv} times. Hence $|T| = \sum_{uv \in E} x_{uv}$.

108 We shall prove this theorem in two parts. First, we show that it is always possible to form
109 a junction graph at every vertex given a local threading (Section 2.1). Then we show that a
110 closed walk can be obtained from the resulting collection of junction graphs (Section 2.2).

111 2.1 Constructing a Connected Junction Graph

112 Forming a junction graph $J(v)$ at vertex v reduces to constructing a connected graph on
113 vertices $t_1, \dots, t_{d(v)}$, where each vertex represents a tube incident with v , with degrees
114 $x_1, \dots, x_{d(v)}$, respectively. We shall construct $J(v)$ in two steps, first in the case where (C4)
115 holds with equality (Lemma 3) and then in the general case (Lemma 4).

116 ► **Lemma 3.** We can construct a tree S consisting of d vertices with respective degrees
117 $x_1, \dots, x_d \geq 1$ satisfying $\sum_{i=1}^d x_i = 2(d - 1)$ in $O(d)$ time.

118 **Proof.** We provide an inductive argument and a recursive algorithm. In the base case, when
119 $d = 2$, $x_1 = x_2 = 1$, and so the solution is a one-edge path. For $d > 2$, the average x_i value is
120 $\frac{2(d-1)}{d}$ which is strictly between 1 and 2. Hence there must be one vertex i satisfying $x_i > 1$
121 and another vertex j satisfying $x_j = 1$. Now apply induction/recursion to x' where $x'_k = x_k$
122 for all $k \notin \{i, j\}$, $x'_i = x_i - 1$, and x_j does not exist (so there are $n - 1 < n$ values), to obtain
123 a tree S' . We can construct the desired tree S from S' by adding the vertex j and edge ij .

124 The recursive algorithm can be implemented in $O(d)$ time as follows. We maintain two
125 stacks: the first for vertices of degree > 1 and the second for vertices of degree 1. In each
126 step, we pop vertex i from the first stack, pop vertex j from the second stack, and connect
127 vertices i and j . We then decrease x_i by 1 and push it back onto one of the stacks depending
128 on its new value. This process continues until the stacks are empty. Each step requires
129 constant time, and we perform at most $\sum_{i=1}^d x_i = O(d)$ steps, so the total running time is
130 $O(d)$. ◀

131 ► **Lemma 4.** Given a local threading $\{x_e\}$ and a vertex $v \in V$, we can construct a connected
132 junction graph $J(v)$ with no self-loops in $O(\sum_{u \in N(v)} x_{uv})$ time.

Algorithm 1 Constructing a Connected Junction Graph $J(v)$

1. $R \leftarrow \emptyset$ ▷ Set of ‘redundant’ edges
 2. $x'_i \leftarrow x_i$ for all $i \in \{1, \dots, d(v)\}$
 3. Repeat until $\sum_{i=1}^{d(v)} x'_i = 2(d(v) - 1)$:
 - a. $x'_\alpha \leftarrow x'_\alpha - 1$ where $x'_\alpha = \max_{i=1}^{d(v)} x'_i$, breaking ties arbitrarily
 - b. $x'_\beta \leftarrow x'_\beta - 1$ where $x'_\beta = \max_{i \in \{1, \dots, d(v)\} \setminus \{\alpha\}} x'_i$, breaking ties arbitrarily
 - c. $R \leftarrow R \cup \{t_\alpha t_\beta\}$
 4. Compute tree S on vertices $t_1, \dots, t_{d(v)}$ with degrees $x'_1, \dots, x'_{d(v)}$ (Lemma 3)
 5. Return $R \cup S$
-

133 **Proof.** We give the construction of a connected junction graph $J(v)$, adopting the notation
 134 introduced at the start of this section. See Algorithm 1 for the corresponding pseudocode.

135 Recall that a local threading $\{x_e\}$ is a set of integers satisfying the constraints specified
 136 in Definition 1. Label the edges incident to vertex v by the integers $1, \dots, \deg(v)$. The goal
 137 is to form a connected graph $J(v)$ having a vertex t_i for each $i \in \{1, \dots, \deg(v)\}$, where the
 138 degree of each t_i is x_i . We begin with an empty graph (Step 1) and initialize $x'_i = x_i$ for
 139 each i (Step 2), where x'_i represents the number of additional edges required for vertex t_i to
 140 achieve its desired degree x_i . While $\sum_{i=1}^{d(v)} x'_i > 2(d(v) - 1)$, we repeat the following steps:
 141 find the two largest values x'_α and x'_β (resolving ties arbitrarily), add an edge between their
 142 corresponding vertices t_α and t_β , and decrement x'_α and x'_β by 1 (Step 3). The resulting x'_i
 143 values sum to $2(d(v) - 1)$, and we prove below that $x'_i \geq 1$ for each i . Next, we construct a
 144 tree with vertex degrees x'_1, \dots, x'_i via the algorithm in Lemma 3 (Step 4). We return the
 145 graph that follows from these two procedures.

146 This graph contains no self-loops because we require $\alpha \neq \beta$ (Step 3b). We further assert
 147 that the graph is connected. To prove this fact, we demonstrate the proper application of
 148 the inductive procedure outlined in the proof of Lemma 3 in forming a tree (Step 4). We
 149 only need to validate that $x'_1, \dots, x'_{d(v)} \geq 1$, as $\sum_{i=1}^{d(v)} x'_i = 2(d(v) - 1)$ is guaranteed upon
 150 the termination of the loop (Step 3). Suppose for contradiction that $x'_k < 1$. It follows that
 151 $x'_k = 1$ at the start of some iteration and was subsequently decremented, either via Step 3a
 152 or 3b. We consider these two cases:

- 153 ■ **Case 1** (Step 3a, $k = \alpha$): $x'_k \geq x'_i$ for all $i \in \{1, \dots, d(v)\}$, so

$$154 \quad \sum_{i=1}^{d(v)} x'_i \leq d(v) \cdot x'_k = d(v) \leq 2(d(v) - 1),$$

155 a contradiction for any $d(v) > 1$, which is assumed.

- 156 ■ **Case 2** (Step 3b, $k = \beta$): As $x'_k \geq x'_i$ for all $i \in \{1, \dots, d(v)\} \setminus \{\alpha\}$, so

$$157 \quad \sum_{i \in \{1, \dots, d(v)\} \setminus \{\alpha\}} x'_i \leq (d(v) - 1) \cdot x'_k = d(v) - 1.$$

158 Recall that $\sum_{i=1}^{d(v)} x'_i = x'_\alpha + \sum_{i \in \{1, \dots, d(v)\} \setminus \{\alpha\}} x'_i \geq 2d(v)$ is required to enter the loop.
 159 Hence, applying the above deduction, $x'_\alpha > \sum_{i \in \{1, \dots, d(v)\} \setminus \{\alpha\}} x'_i$, contradicting the below
 160 invariant (Equation 1) of the loop in Step 3.

161 **Loop Invariant:** The following invariant is maintained by the algorithm's loop (Step 3),
 162 established on initialization via (C3):

$$163 \quad x'_i \leq \sum_{j \in \{1, \dots, d(v)\} \setminus \{i\}} x'_j \text{ for all } i \in \{1, \dots, d(v)\} \quad (1)$$

164 We observe that $\sum_{i=1}^{d(v)} x_i$ decreases by 2 with every iteration: either both sides of Equation 1
 165 are reduced by 1, thereby maintaining the inequality, or the left-hand side remains unchanged
 166 while the right-hand side is reduced by 2. In the latter scenario, counts $x'_\alpha, x'_\beta \geq x'_i$ are
 167 updated in Steps 3a and 3b. Observe that $x'_\alpha \geq 2$ because $\sum_{i=1}^{d(v)} x'_i \geq 2n$ is a prerequisite
 168 for loop entry. Letting x''_i denote the value of x'_i at the beginning of the next iteration, we
 169 arrive at the desired conclusion:

$$170 \quad x''_i = x'_i \leq (x'_\alpha - 2) + x'_\beta \leq \sum_{j \in \{1, \dots, d(v)\} \setminus \{i\}} x'_j - 2 = \sum_{j \in \{1, \dots, d(v)\} \setminus \{i\}} x''_j.$$

171 **Running-Time Analysis:** To perform Steps 3a and 3b efficiently, we maintain the x' values
 172 in a monotone priority queue, specifically, an array A of lists $A[0], A[1], \dots, A[\max_{i=1}^{d(v)} x_i]$,
 173 where each list L_j maintains the indices i for which $x'_i = j$. We can initialize this data
 174 structure in $O(d(v) + \max_{i=1}^{d(v)} x_i)$ time, which is $O(\sum_{i=1}^{d(v)} x_i)$ because each $x_i \geq 1$. We also
 175 maintain the largest array index j for which $A[j]$ is nonempty, and the second-largest array
 176 index k for which $A[k]$ is nonempty. To find and decrement the maximum value in the
 177 priority queue (as in Step 3a), we extract an index α from list $A[j]$, decrement x'_α , and then
 178 append α to list $A[x'_\alpha] = A[j - 1]$. If $A[j]$ is now empty, we also decrement j ; the new $A[j]$ is
 179 guaranteed to be nonempty. To find and decrement the second-largest value in the priority
 180 queue (as in Step 3b), we extract β from $A[j]$ if $A[j]$ has an index other than α (i.e., has
 181 length > 1), and otherwise extract from $A[k]$; then we decrement x'_β , move β to the correct
 182 list, and optionally decrement either j or k as before. Each of these steps takes constant
 183 time, so the overall running time is $O(\sum_{i=1}^{d(v)} x_i) = O(\sum_{u \in N(v)} x_{uv})$. ◀

184 2.2 Obtaining a Closed Walk

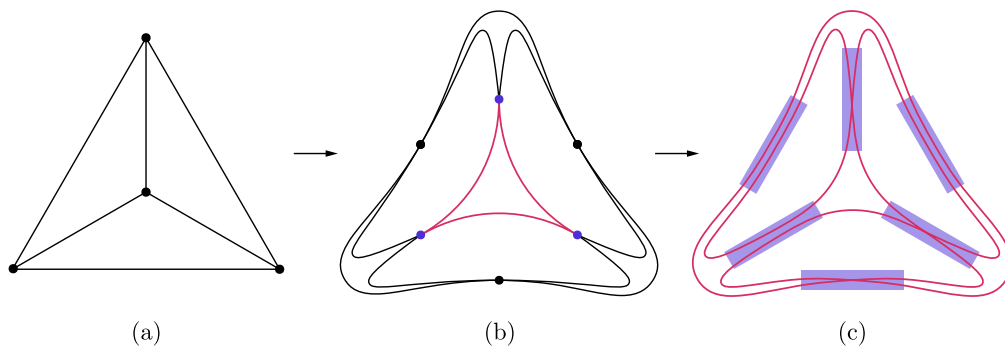
185 Now suppose we have a junction graph $J(v)$ for every vertex v , obtained by repeatedly
 186 applying Lemma 4 to a given local threading. Our goal is to find a closed walk in G that has
 187 no U-turns and corresponds to these junction graphs.

188 Define the **threading graph** to be the graph whose vertices correspond to tubes and
 189 whose edges are given by the union of all junction graphs (joining at vertices corresponding
 190 to the same tube). See Figures 2 and 3 for examples.

191 In this threading graph, we find an **Euler tour**: a closed walk that visits each edge of the
 192 graph exactly once. The presence of an Euler tour through a threading graph is guaranteed
 193 because each vertex has even degree [2], specifically twice the count x_e for vertex t_e . The
 194 tour can be computed in time linear in the number of edges of the input graph [9], which is
 195 $O(\sum_{i=1}^n x_i)$.

196 To ensure that U-turns are avoided in the threading, we enforce that the Euler cycle does
 197 not consecutively traverse two edges of the same junction graph, which can be done in linear
 198 time by a reduction to forbidden-pattern Euler tours [3].

199 Combining our results, we can convert a local threading $\{x_e\}$ of G to a corresponding
 200 threading of G in time $O(\sum_{v \in V} \sum_{u \in N(v)} x_{uv} + \sum_{i=1}^n x_i) = O(\sum_{i=1}^n x_i)$. Later in Section 3.1,
 201 we will show that the optimal threading satisfies $\sum_{i=1}^n x_i = O(m)$, in which case our running
 202 time simplifies to $O(m)$.



■ **Figure 3** The target model, a threading graph featuring junction graphs as cycles, and a threading of the input model following an Eulerian cycle of the threading graph.

203 ▶ **Theorem 5.** *We can convert a local threading solution of G into a threading of G in*
 204 *$O(\sum_{i=1}^n x_i)$ time, which for an optimal threading is $O(m)$.*

205 3 Worst-Case Bounds

206 In this section, we prove tight worse-case upper and lower bounds on the total length of
 207 an optimal threading (Section 3.1) and on the most times one edge may be visited by an
 208 optimal threading (Section 3.2).

209 3.1 Total Length

210 Every graph G with minimum degree ≥ 2 has a **double threading** defined by assigning each
 211 junction graph $J(v)$ to be a cycle of length $d(v)$, as depicted in Figure 3. This threading
 212 results in each tube being traversed exactly twice, totaling a length of $2m$. Thus an optimal
 213 threading has length at most $2m$. We can approach this upper bound up to an additive
 214 constant by considering graphs with long sequences of bridges, such as the graph illustrated
 215 in Figure 4a. We shall later tighten this upper bound by considering graph properties
 216 (Lemma 9).

217 Now we establish a lower bound on the total length of any threading:

218 ▶ **Lemma 6.** *Any threading must have length at least $2m - n$.*

219 **Proof.** Each junction graph $J(v)$ is connected, so it contains at least $d(v) - 1$ edges, and
 220 every edge $t_i t_j$ in $J(v)$ necessitates visits to two tubes, t_i and t_j . By summing these visits
 221 across all junctions, we double-count visits to tubes. Thus, any threading $\{x_{uv}\}$ has length

$$222 \quad \sum_{uv \in E} x_{uv} = \frac{1}{2} \sum_{v \in V} \sum_{u \in N(v)} x_{uv} \geq \frac{1}{2} \sum_{v \in V} 2(d(v) - 1) \stackrel{\text{(handshaking)}}{=} 2m - n.$$

223 From the perspective of local threading, the inequality step follows from constraint (C4). ◀

224 This lower bound is sometimes tight, such as in Figure 2a, which we give a special name:

225 ▶ **Definition 7.** *A **perfect threading** is a graph threading of length $2m - n$.*

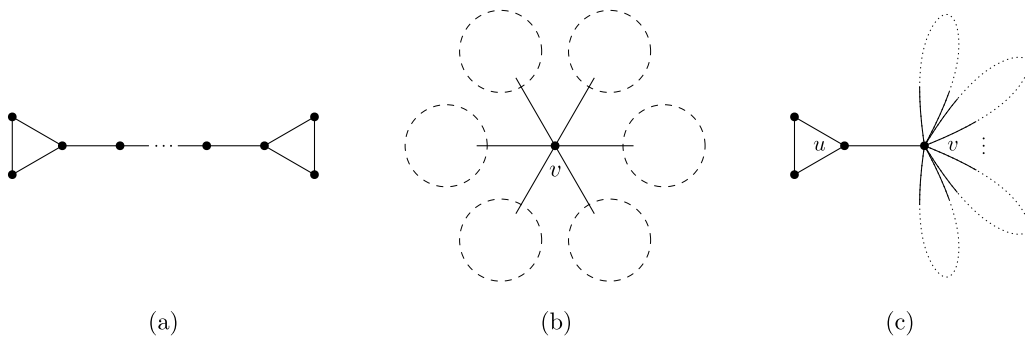
226 By the analysis in the proof of Lemma 6, we obtain equivalent definitions:

227 ► **Lemma 8.** *The following are equivalent for a graph threading $\{x_{uv}\}$:*

- 228 1. $\{x_{uv}\}$ is a perfect threading.
- 229 2. Every junction graph $J(v)$ is a tree, i.e., has exactly $d(v) - 1$ edges.
- 230 3. Inequality (C4) holds with equality.

231 Not every graph has a perfect threading (Figure 4b). A key observation is that bridges
 232 must be threaded at least twice. If we were to remove a bridge, the graph would have two
 233 connected components, and any closed walk on the entire graph would have to enter and
 234 exit each component at least once. Because the only way to pass between the two connected
 235 components is through the bridge, the walk would have to traverse the bridge at least twice.

236 Hence, vertices whose incident edges are all bridges must have junction graphs containing
 237 at least $d(v)$ edges. We call these vertices **London** vertices. A tighter lower bound is
 238 $2m - n + |L|$ where L is the set of London vertices in G . Note that this bound is determined
 239 by the number of London vertices rather than the number of bridges — a London vertex
 240 connected to multiple bridges only increases the bound by 1.



■ **Figure 4** (a) A graph with a minimum threading length of $2m - 6$. (b) A vertex connected to 6 disjoint parts of the graph (denoted as dashed circles). Each bridge incident to vertex v is at least double-threaded, and hence (C4) holds at v as strict inequality, so the graph has no perfect threading. (c) The vertex v has degree Δ and is connected to $\frac{\Delta-1}{2}$ loops (dotted) of length > 5 . In an optimal threading, the edge uv is threaded $\Delta - 1$ times.

241 Next, we consider an improved upper bound on the length of an optimal threading. While
 242 $2m$ edge visits always suffice to thread a graph, the following lemma demonstrates that this
 243 number is never necessary, as any graph without vertices of degree 1 contains a cycle.

244 ► **Lemma 9.** *Let C be a set of vertex-disjoint simple cycles in G , and let $|C|$ denote the
 245 total number of edges in its cycles. In an optimal threading of G , at most $2m - |C|$ edge
 246 visits are needed.*

247 **Proof.** We use $e \in C$ to denote edge e participating in some cycle in C . Define the set of
 248 integers $\{x_e\}$ where $x_e = 1$ if $e \in C$ and $x_e = 2$, otherwise. By design, $\sum_{e \in E} x_e = 2m - |C|$,
 249 and so it suffices to show that $\{x_e\}$ is a valid threading of G , i.e., $\{x_e\}$ satisfies constraints
 250 (C1)–(C4). Observe that each vertex v is either (1) covered once by a single cycle in C ,
 251 meaning that two of its incident edges are single-threaded while the others are threaded
 252 twice, or (2) left uncovered, in which all of its incident edges are double-threaded. In both
 253 scenarios, all constraints are clearly met. Note that (C4) holds as an equality in a vertex
 254 covered once by a cycle in C . ◀

255 In Section 5.2, we provide an efficient algorithm for computing a threading that achieves
 256 the above bound by reduction to finding the largest set of vertex-disjoint cycles.

3.2 Maximum Visits to One Edge

Each edge is threaded at least once in a graph threading, but what is the maximum number of times an *optimal* solution can thread an edge? In this section, we establish that no optimal threading exceeds $\Delta - 1$ visits to a single edge. This upper bound is tight, as demonstrated by edge uv in Figure 4c: Constraint (C4) requires multiple visits to at least one edge connected to v , and revisiting uv is the most economical when the loops incident to v are long. It is worth noting that bounding the visits to an edge by the maximum degree of its endpoints may not suffice for an optimal solution, as in the case of the left-most edge in Figure 4c, which is traversed $\frac{\Delta-1}{2} > 2$ times despite both its endpoints having a degree of 2.

► **Lemma 10.** *An optimal threading visits a single edge at most $\Delta - 1$ times.*

Proof. If $\Delta = 2$, then G is a cycle, in which case the optimal threading traverses every edge once. Hence, for the remainder of this proof, we may assume $\Delta \geq 3$.

Suppose $\{x_e\}$ is an optimal threading of a graph G . Let $uv = \arg \max_{e \in E} x_e$ denote the edge with the highest count and assume for a contradiction that $x_{uv} \geq \Delta$. For simplicity, we first assume that $d(u), d(v) \geq 3$ and handle the case where $d(u) = 2$ or $d(v) = 2$ at the end. We shall show that we can remove two threads from uv without violating the problem constraints. That is, the set $\{\hat{x}_e\}$ is a valid threading when defined as $\hat{x}_e = x_{uv} - 2$ if $e = uv$ and $\hat{x}_e = x_e$, otherwise. This conclusion contradicts our assumption that $\{x_e\}$ is optimal. The key to this proof is the following:

(C4): Because $\{x_e\}$ satisfies (C3), $\sum_{i=1}^{d(v)-1} x_{u_i v} \geq x_{uv} \geq \Delta$, and so

$$\sum_{w \in N(v)} \hat{x}_{wv} = \hat{x}_{uv} + \sum_{i=1}^{d(v)-1} x_{u_i v} \geq (\Delta - 2) + \Delta \geq 2(d(v) - 1).$$

By symmetry, u also satisfies (C4), and therefore (C4) is met by all vertices of G . We are left to show that $\{\hat{x}_e\}$ satisfies (C1)–(C3).

(C1): $\hat{x}_{uv} > \Delta - 2 \geq 1$. For any other edge $\hat{x}_e = x_e \geq 1$.

(C2): Constraint (C2) is met as we do not modify the parity of any count.

(C3): We now show (C3) is satisfied for v and by symmetry, u , and therefore met by all vertices of G . Let us denote the neighbors of v by $u, u_1, \dots, u_{d(v)-1}$. We have

$$\sum_{w \in N(v) \setminus \{u\}} \hat{x}_{wv} = \sum_{w \in N(v) \setminus \{u\}} x_{wv} \geq x_{uv} > \hat{x}_{uv},$$

so (C3) is satisfied for uv . We now demonstrate (C3) also holds for the remaining $u_i v$'s. If $d(v) \geq 4$, because $x_{uv} \geq x_{u_i v} = \hat{x}_{u_i v}$ by our choice of uv , we have

$$\sum_{w \in N(v) \setminus \{u_i\}} \hat{x}_{wv} \stackrel{(C1)}{\geq} \hat{x}_{uv} + \underbrace{d(v) - 2}_{\geq 2} \geq (x_{uv} - 2) + 2 = x_{uv} \geq \hat{x}_{u_i v},$$

as desired. Otherwise, $d(v) = 3$. Without loss of generality, we want to show that

$$x_{u_1 v} \leq \hat{x}_{uv} + \hat{x}_{u_2 v} = x_{uv} + x_{u_2 v} - 2.$$

290 Because $x_{uv} \geq x_{u_1v}$ (by choice of uv) and $x_{u_2v} \geq 1$ (from (C1)), this inequality holds in all
 291 cases except when $x_{u_1v} = x_{uv}$ and $x_{u_2v} = 1$. However, in this particular scenario, the sum of
 292 counts surrounding v amounts to $2x_{uv} + 1$, which contradicts (C2).

293 If either endpoint of uv has degree 2, then we instead consider the maximal path w_1, \dots, w_ℓ
 294 including uv such that all intermediate vertices have degree 2: $d(w_2) = \dots = d(w_{\ell-1}) = 2$.
 295 Thus $d(w_1), d(w_\ell) \geq 3$ (as we are in the case $\Delta \geq 3$) and $uv = w_i w_{i+1}$ for some i . Because
 296 $\{x_e\}$ is a valid threading, we must have $x_{w_1 w_2} = \dots = x_{w_{\ell-1} w_\ell} = x_{uv} \geq \Delta$. Now we modify
 297 the threading $\{x_e\}$ by removing two threads from each $x_{w_i w_{i+1}}$ to obtain $\{\hat{x}_e\}$. Constraints
 298 (C1)–(C4) remain satisfied at the degree-2 vertices $w_2, \dots, w_{\ell-1}$. Finally, we can apply the
 299 proof above to show that the constraints remain satisfied at the end vertices w_1 and w_ℓ of
 300 degree at least 3. ◀

301 4 Polynomial-Time Algorithm via Perfect Matching

302 In this section, we present our main result: a polynomial-time algorithm for computing
 303 an optimal threading of an input graph G . Our approach involves reducing OPTIMAL
 304 THREADING to the problem of min-weight perfect matching, defined as follows.

305 A *matching* in a graph is a set of edges without common vertices. A *perfect matching*
 306 is a matching that covers all vertices of the graph, i.e., a matching of cardinality $\frac{n}{2}$. If the
 307 graph has edge weights, the *weight* of a matching is the sum of the weights of its edges, and
 308 a *min-weight perfect matching* is a perfect matching of minimum possible weight.

309 We begin by constructing a graph that possesses a perfect matching if and only if G has
 310 a *perfect* threading (Definition 7). This construction gives a reduction from determining
 311 the existence of a perfect threading to the perfect matching problem. Next, we extend this
 312 construction to ensure perfect matching always exists. In this extended construction, a
 313 perfect matching of weight W corresponds to a threading of length $W + m$, giving a reduction
 314 from OPTIMAL THREADING to finding a min-weight perfect matching.

315 4.1 Determining Existence of a Perfect Threading

316 By Lemma 8, a threading $\{x_{uv}\}$ of a graph G is a perfect threading if and only if it satisfies
 317 inequality (C4) with equality:

$$318 \text{(C*4)} \quad \sum_{u \in N(v)} x_{uv} = 2(d(v) - 1) \text{ for all } v \in V.$$

319 In fact, most of the other constraints become redundant in this case:

320 ▶ **Lemma 11.** $\{x_{uv}\}$ is a perfect threading if and only if it satisfies (C1) and (C*4).

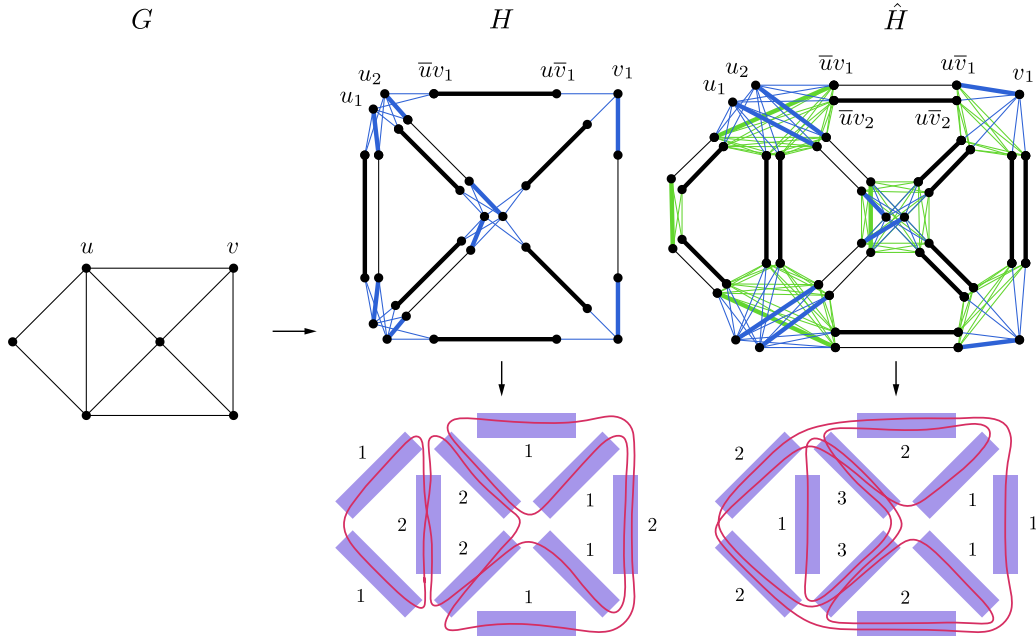
321 **Proof.** If $\{x_{uv}\}$ satisfies (C*4), then it satisfies constraint (C2), because $2(d(v) - 1) \equiv 0$
 322 (mod 2). (C*4) can be rewritten as $x_{uv} + \sum_{w \in N(v) \setminus \{u\}} x_{wv} = 2(d(v) - 1)$, and by (C1),
 323 $\sum_{w \in N(v) \setminus \{u\}} x_{wv} \geq d(v) - 1$, so (C3) also holds. ◀

324 Consider a vertex v and its neighbors $u_1, \dots, u_{d(v)}$. We can think of constraint (C*4) as
 325 allocating $2(d(v) - 1)$ units among $x_{u_1 v}, \dots, x_{u_{d(v)} v}$. First, we must allocate one unit to each
 326 $x_{u_i v}$ in order to satisfy (C1). This leaves $d(v) - 2$ units to distribute among the edges.

327 We show how to simulate this distribution problem by constructing a graph H that has a
 328 perfect matching if and only if, for every vertex v , we can distribute $d(v) - 2$ units among its
 329 neighboring $x_{u_i v}$. Thus H has a perfect matching if and only if G has a perfect threading.

330 Given a graph G , define the graph H as follows; refer to Figure 5. For each edge
 331 $uv \in E(G)$, create a perfect matching of $d_{uv} := \min\{d(u), d(v)\} - 2$ disjoint edges $(\bar{u}v_i, u\bar{v}_i)$,

332 among $2d_{uv}$ created vertices $\bar{u}v_1, \bar{u}v_1, \dots, \bar{u}v_{d_{uv}}, \bar{u}v_{d_{uv}}$.¹ For each vertex v , create $d(v) - 2$
 333 vertices labeled $v_1, \dots, v_{d(v)-2}$. For every edge uv incident to v , add an edge between vertices
 334 v_i and $\bar{u}v_j$ for all $1 \leq i \leq d(v) - 2$ and $1 \leq j \leq d_{uv}$ (forming a biclique). Note that any
 335 vertex of degree 2 disappears in this construction because of the -2 in each creation count.



■ **Figure 5** Construction of H and \hat{H} from G , each with some matching in bold and a corresponding threading to the matching labeled with counts.

336 ▶ **Theorem 12.** G has a perfect threading if and only if H has a perfect matching.

337 To prove Theorem 12, we will show how to translate between a perfect threading of G
 338 and a perfect matching of H . Given a matching $M \subseteq E(H)$ of H , define a possible threading
 339 solution $\varphi(M) = \{x_{uv}\}$ by taking x_{uv} to be 1 plus the number of edges $(\bar{u}v_i, \bar{u}v_i)$ that are
 340 not included in M : $x_{uv} := 1 + |\{(\bar{u}v_i, \bar{u}v_i) : 1 \leq i \leq d_{uv}\} \setminus M|$.

341 ▷ **Claim 13.** If M is a perfect matching in H , then $\varphi(M)$ is a perfect threading of G .

342 **Proof.** By Lemma 11, it suffices to prove that $\varphi(M)$ satisfies (C1) and (C*4). The 1+ in
 343 the definition of $\varphi(M)$ satisfies (C1). For every vertex $v \in V$, the vertices $v_1, \dots, v_{d(v)-2}$ are
 344 all matched to vertices of the form $\bar{u}v_i$; for each such matching pair, the edge $(\bar{u}v_i, \bar{u}v_i) \notin M$.
 345 Conversely, for any vertex $\bar{u}v_i$ that is not matched to any v_j , the edge $(\bar{u}v_i, \bar{u}v_i)$ must be
 346 part of the matching. Hence, for each vertex v , the number of edges of the form $(\bar{u}v_i, \bar{u}v_i)$
 347 that are not included in M is exactly $d(v) - 2$. The sum $\sum_{u \in N(v)} x_{uv}$ includes this count
 348 and $d(v)$ additional 1s, so equals $(d(v) - 2) + d(v) = 2(d(v) - 1)$, satisfying (C*4). ◀

349 ▷ **Claim 14.** For any perfect threading $\{x_{uv}\}$ of G , there exists a perfect matching M of H
 350 such that $\varphi(M) = \{x_{uv}\}$.

¹ In the same way that uv and vu denote the same edge, we treat labels $\bar{u}v$ and $\bar{v}u$ as the same. Thus, the notation $\bar{u}v_i$ and $\bar{v}u_i$ refers to the same vertex.

37:12 Graph Threading

351 **Proof.** Given a perfect threading $\{x_{uv}\}$ of G , we construct a perfect matching of H as follows.
 352 First, for every $uv \in E(G)$, we match the edges $(\bar{u}v_1, \bar{u}v_1), \dots, (\bar{u}v_{d_{uv}-x_{uv}+1}, \bar{u}v_{d_{uv}-x_{uv}+1})$.
 353 We show that index $d_{uv} - x_{uv} + 1$ is always non-negative; when it is zero, we match no such
 354 edges. By constraint (C*4), $x_{uv} = 2(d(v) - 1) - \sum_{w \in N(v) \setminus \{u\}} x_{wv}$. By constraint (C1), each
 355 term in the sum is at least 1, so $x_{uv} \leq d(v) - 1$. Thus $x_{uv} \leq d_{uv} + 1$, i.e., $d_{uv} - x_{uv} + 1 \geq 0$.
 356 With our matching so far, the number of unmatched vertices of the form $\bar{u}v_i$ at each
 357 vertex v is $\sum_{u \in N(v)} (x_{uv} - 1)$. By (C*4), this count is exactly $2(d(v) - 1) - d(v) = d(v) - 2$.
 358 Thus we can match each of these unmatched vertices to a unique vertex v_j to complete our
 359 perfect matching. ◀

360 Claims 13 and 14 complete the proof of Theorem 12.

361 4.1.1 Running-Time Analysis

362 First, let us calculate the sizes of $V(H)$ and $E(H)$. Recall that H has $d(v) - 2$ vertices
 363 corresponding to every vertex $v \in V(G)$, and up to $2(\min\{d(u), d(v)\} - 2) \leq 2\Delta$ vertices
 364 corresponding to every edge $uv \in E(G)$. Therefore, the maximum number of vertices in H is

$$365 \sum_{v \in V} (d(v) - 2) + 2 \sum_{uv \in E} \Delta \leq 2m - 2n + 2m\Delta = O(m\Delta).$$

366 Now recall that H has $\min\{d(u), d(v)\} - 2 \leq \Delta$ edges for every uv and at most Δ^3 edges for
 367 every v . Thus, the total number of edges in H is upper-bounded by

$$368 2 \sum_{uv \in E} \Delta + \sum_{v \in V} \Delta^3 \leq 2m \cdot \Delta + n\Delta^3 = O(n\Delta^3).$$

369 We conclude that H can be constructed in $O(n\Delta^3 + m\Delta)$ time.

370 Micali and Vazirani [16] gave an algorithm that computes the maximum matching of
 371 a general graph in $O(\sqrt{nm})$ time, thereby enabling us to verify the existence of a perfect
 372 matching. It follows that we can determine a perfect matching of H in time

$$373 O(\sqrt{|V(H)|} \cdot |E(H)|) = O(\sqrt{m\Delta} \cdot n\Delta^3) = O(n\sqrt{m} \cdot \Delta^{3.5}).$$

374 This running time exceeds the construction time of H , and so it is the final running time of
 375 our algorithm.

376 Note that we can improve the bound on the size of H by considering the *arboricity*
 377 of G . The arboricity of a graph $\alpha(G)$ is defined as the minimum number of edge-disjoint
 378 spanning forests into which G can be decomposed [5]. This parameter is closely related to
 379 the degeneracy of the graph and is often smaller than Δ . Chiba and Nishizeki [5] show
 380 that $\sum_{uv \in E} \min\{d(u), d(v)\} \leq 2m\alpha(G)$, which would give us a tighter bound on the size of
 381 $V(H)$.

382 In summary, we can find a perfect threading of G , if one exists, by determining a perfect
 383 matching in H in $O(n\sqrt{m} \cdot \Delta^{3.5})$ time.

384 4.2 Finding an Optimal Threading

385 Now we examine the general scenario where a perfect threading may not exist, i.e., (C4) may
 386 hold with a strict inequality for some vertex. The graph H constructed in Section 4.1 permits
 387 exactly $2(d(v) - 1)$ visits to vertex v . We aim to allow more visits to v while satisfying
 388 constraints (C2) and (C3).

389 In a general threading, $x_{uv} \leq \min\{d(u), d(v)\} - 1$ (as argued in Claim 14) is not necessarily
 390 true. However, Lemma 10 gives us a weaker upper bound, $x_{uv} \leq \Delta - 1$, for any optimal
 391 threading. We therefore modify the construction from Section 4.1 in two ways. First, we
 392 generate $\Delta - 2$ copies of every edge, regardless of the degree of its endpoints. Second, for
 393 every pair of edges uv and wv meeting at vertex v , we introduce an edge between $u\bar{v}_i$ and
 394 $w\bar{v}_j$ for all $1 \leq i, j \leq \Delta - 2$. Intuitively, these edges represent threads passing through v ,
 395 going from uv to wv , after having met the lower bound of $2(d(v) - 1)$ visits.

396 More formally, we define a weighted graph \hat{H} from G as follows; refer to Figure 5. For
 397 each edge $uv \in E(G)$, create a weight-0 perfect matching of $\Delta - 2$ disjoint weight-0 edges
 398 $(\bar{u}v_i, u\bar{v}_i)$, among $2(\Delta - 2)$ created vertices $\bar{u}v_1, u\bar{v}_1, \dots, \bar{u}v_{\Delta-2}, u\bar{v}_{\Delta-2}$; these edges are black
 399 in Figure 5.

400 For every vertex v , create $d(v) - 2$ vertices $v_1, \dots, v_{d(v)-2}$, and add a weight- $\frac{1}{2}$ edge
 401 $(v_i, u\bar{v}_j)$ for every $u \in N(v)$ and $1 \leq i \leq d(v) - 2, j \leq \Delta - 2$; these edges are blue in Figure 5.
 402 Finally, for each pair of edges uv and wv incident to v , create a weight-1 edge $(u\bar{v}_i, w\bar{v}_j)$ for
 403 every $1 \leq i, j \leq \Delta - 2$; these edges are green in Figure 5.

404 ► **Theorem 15.** G has a threading of length $W + m$ with $\max_{uv \in E(G)} x_{uv} \leq \Delta - 1$ if and
 405 only if \hat{H} has a perfect matching of weight W .

406 To prove Theorem 15, we again show how to translate between a threading of G and
 407 a perfect matching of \hat{H} . Given a matching $M \subseteq E(\hat{H})$ of \hat{H} , define a possible threading
 408 solution $\psi(M) = \{x_{uv}\}$ by taking x_{uv} to be 1 plus the number of copies of uv not matched
 409 in M : $x_{uv} := 1 + |\{(\bar{u}v_i, u\bar{v}_i) : 1 \leq i \leq \Delta - 2\} \setminus M|$.

410 ▷ **Claim 16.** If M is a perfect matching in \hat{H} of weight W , then $\psi(M) = \{x_{uv}\}$ is a threading
 411 of G of length $W + m$ with $\max_{uv \in E(G)} x_{uv} \leq \Delta - 1$.

412 **Proof.** By definition of $\psi(M)$, every x_{uv} satisfies $1 \leq x_{uv} \leq \Delta - 1$. Thus, $\{x_{uv}\}$ satisfies
 413 (C1) and $\max_{uv \in E(G)} x_{uv} \leq \Delta - 1$.

414 Let $a_v(uv)$ denote the number of vertices $u\bar{v}_i$ (for $1 \leq i \leq \Delta - 2$) matched with some
 415 vertex v_j , i.e., the number of blue edges incident to a vertex $u\bar{v}_i$ that appear in M . Let
 416 $b_v(uv)$ denote the number of vertices $u\bar{v}_i$ (for $1 \leq i \leq \Delta - 2$) matched with some vertex $w\bar{v}_j$,
 417 i.e., the number of green edges incident to a vertex $u\bar{v}_i$ that appear in M . Any other vertex
 418 $u\bar{v}_i$ (not incident to either a blue or green edge in M) must be matched to its corresponding
 419 vertex $\bar{u}v_i$, which does not contribute to x_{uv} . Hence, $x_{uv} = 1 + a_v(uv) + b_v(uv)$.

420 Next we prove that $\{x_{uv}\}$ satisfies constraint (C4). For every vertex v , we have
 421 $\sum_{u \in N(v)} a_v(uv) = d(v) - 2$, which implies $\sum_{u \in N(v)} (x_{uv} - 1) \geq d(v) - 2$, which is equi-
 422 valent to (C4).

423 Next consider (C2). Any edge $(u\bar{v}_i, w\bar{v}_j)$ present in M adds 1 to both $b_v(uv)$ and $b_v(wv)$,
 424 thereby ensuring $\sum_{u \in N(v)} b_v(uv) \equiv 0 \pmod{2}$. Consequently,

$$425 \sum_{u \in N(v)} x_{uv} \equiv \sum_{u \in N(v)} (a_v(uv) + 1) = 2(d(v) - 1) \equiv 0 \pmod{2}.$$

426 Finally, consider (C3). Given that $a_v(uv) \leq d(v) - 2$, we infer $\sum_{w \in N(v) \setminus \{u\}} a_v(uw) +$
 427 $d(v) - 1 \geq a_v(uv) + 1$. Additionally, for each vertex contributing to $b_v(uv)$, its matched
 428 vertex contributes to some $b_v(wv)$, so $\sum_{w \in N(v) \setminus \{u\}} b_v(wv) \geq b_v(uv)$. Hence, we have

$$429 \sum_{w \in N(v) \setminus \{u\}} x_{uw} = \sum_{w \in N(v) \setminus \{u\}} (a_v(uw) + b_v(wv) + 1) \geq (a_v(uv) + 1) + b_v(uv) = x_{uv}.$$

430 We conclude that $\{x_{uv}\}$ is a threading of G .

37:14 Graph Threading

431 Lastly, we compute its length.

432 The weight of M is determined by the number of blue and green edges it contains because
 433 the edges $(\bar{u}v_i, \bar{u}v_j)$ have zero weight. Each of its blue edges of the form $(v_i, \bar{u}v_j)$ has weight
 434 $\frac{1}{2}$ and is accounted for once in $a_v(uv)$, for a total weight of $a_v(uv)/2$. Each of its green edges
 435 of the form $(\bar{u}v_i, \bar{w}v_j)$ has weight 1 and is counted twice — once in $b_v(uv)$ and once more in
 436 $b_v(uv)$ — for a total weight of $b_v(uv)/2$. Hence, the weight W of the matching M is given by

$$437 \quad W = \sum_{v \in V} \sum_{u \in N(v)} \left(\frac{a_v(uv)}{2} + \frac{b_v(uv)}{2} \right) = 2 \sum_{uv \in E} \frac{x_{uv} - 1}{2} = \sum_{uv \in E} x_{uv} - m.$$

438 Therefore $\{x_{uv}\}$ is a threading of G of length $W + m$. ◀

439

440 \triangleright **Claim 17.** For every threading $\{x_{uv}\}$ of G such that $\max_{uv \in E(G)} x_{uv} \leq \Delta - 1$, \hat{H} has a
 441 perfect matching M such that $\psi(M) = \{x_{uv}\}$.

442 **Proof.** Let $\{x_{uv}\}$ be a threading of G satisfying $x_{uv} \leq \Delta - 1$ for every edge $uv \in E$. Recall
 443 Lemma 4, where we demonstrate the construction of a junction graph $J(v)$ for vertex v .

444 For every vertex $v \in V$, we know by (C2) and (C4) that $\sum_{u \in N(v)} x_{uv} = 2(d(v) - 1) + 2k$
 445 for some integer k . Note that $J(v)$ has $d(v)$ vertices and $d(v) - 1 + k$ edges. Because $J(v)$ is
 446 connected, we can thus select k edges from $J(v)$ such that removing them will leave behind a
 447 tree. Denote these edges by $(u^1, w^1), \dots, (u^k, w^k)$ where $u^1, \dots, u^k, w^1, \dots, w^k \in N(v)$. For
 448 each edge (u^ℓ, w^ℓ) , match a green edge of the form $(u^\ell \bar{v}_i, w^\ell \bar{v}_j)$. For every edge uv connected
 449 to v , denote by $b_v(uv)$ the number of vertices of the form $\bar{u}v_i$ currently matched, i.e., the
 450 number of times u appears as an endpoint among the k edges selected from $J(v)$.

451 Because the edges remaining in $J(v)$ after removing $(u^1, w^1), \dots, (u^k, w^k)$ form a tree,
 452 every neighbor of v must have at least one incident edge in $J(v)$ that is *not* selected. Because
 453 the degree of t_{uv} in $J(v)$ is x_{uv} , the number of matched vertices must satisfy $b_v(uv) \leq x_{uv} - 1$.²

454 For each $u \in N(v)$, let $a_v(uv) = x_{uv} - b_v(uv) - 1$. It is clear from our above observation
 455 that $a_v(uv) \geq 0$. Given $\sum_{u \in N(v)} b_v(uv) = 2k$, we have $\sum_{u \in N(v)} a_v(uv) = d(v) - 2$. It
 456 follows that we can match $a_v(uv)$ vertices in $\bar{u}v_1, \dots, \bar{u}v_{\Delta-2}$ to an equal number of vertices
 457 in $v_1, \dots, v_{d(v)-2}$ using blue edges. After executing this procedure, all vertices of the form
 458 $v_1, \dots, v_{d(v)-2}$ will have been matched. Furthermore, the number of matched vertices of the
 459 form $\bar{u}v_i$ is exactly $a_v(uv) + b_v(uv) = x_{uv} - 1$. We repeat this procedure for all vertices.

460 Now, for every edge uv , there are two sets of unmatched vertices, each of size $\Delta - 2 -$
 461 $(x_{uv} - 1) = \Delta - x_{uv} - 1$ $\bar{u}v_i$, of the form $\bar{u}v_i$ and $\bar{u}v_j$, respectively. By rearranging the existing
 462 matches, we can ensure these vertices are exactly $\bar{u}v_1, \dots, \bar{u}v_{\Delta-x_{uv}-1}, \bar{u}v_1, \dots, \bar{u}v_{\Delta-x_{uv}-1}$.
 463 Then we can proceed to match every pair $(\bar{u}v_i, \bar{u}v_i)$, for $i \leq \Delta - x_{uv} - 1$, using a black edge.

464 The above process results in a perfect matching M from the threading $\{x_{uv}\}$. The number
 465 of edges of the form $(\bar{u}v_i, \bar{u}v_i)$ included in the matching is precisely $\Delta - x_{uv} - 1$. Hence,
 466 $\psi(M) = \{x_{uv}\}$. ◀

467 The above two claims complete the proof of Theorem 15. Lemma 10 establishes that
 468 an optimal threading visits an edge no more than $\Delta - 1$ times, so \hat{H} must have a perfect
 469 matching. Furthermore, if M is the min-weight perfect matching of \hat{H} , then $\psi(M)$ is the
 470 optimal threading of G . We can therefore find the optimal threading of G by finding the
 471 min-weight perfect matching of \hat{H} and applying the reduction of Claim 16.

² Here t_{uv} is a vertex representing the tube uv . See the notation in Section 2.1.

472 Note that the solution presented in this section can be readily adapted to address a
 473 constrained variant of OPTIMAL THREADING, where each edge is allowed to be traversed
 474 only a limited number of times by imposing limits on the number of vertex and edge copies
 475 created during the construction of \hat{H} . This scenario arises, for example, when dealing with
 476 tubes of restricted diameter.

477 4.2.1 Running-Time Analysis

478 First, let us analyze the size of \hat{H} : the graph contains $\Delta - 2$ vertices for each vertex $v \in V(G)$
 479 and $2(\Delta - 2)$ vertices for each edge $uv \in E(G)$. Hence, the total number of vertices in \hat{H} is
 480 $O(m\Delta)$. In terms of edges, \hat{H} includes $\Delta - 2$ edges for each edge $uv \in E(G)$ and no more
 481 than Δ^4 edges for each vertex $v \in V(G)$. Therefore, the total edge count in \hat{H} is $O(n\Delta^4)$.
 482 As a result, the construction of \hat{H} requires $O(m\Delta + n\Delta^4)$ time.

483 Next, we use the algorithm of Galil, Mical, and Gabow [10] to find a minimum weight
 484 perfect matching of \hat{H} . This algorithm has time complexity $O(nm \log n)$, and so on \hat{H} it
 485 runs in time

$$486 \quad O(|V(H)||E(H)| \log(|V(H)|)) = O(m\Delta \cdot n\Delta^4 \cdot \log(m\Delta)) = O(nm \cdot \Delta^5 \log n).$$

487 As this term dominates the time for constructing \hat{H} , we conclude that our algorithm for
 488 OPTIMAL THREADING runs in time $O(nm \cdot \Delta^5 \log n)$.

489 4.2.2 Extension to Weighted Graphs

490 In this section, we adapt our OPTIMAL THREADING algorithm to weighted graphs representing
 491 structures whose edges have varying lengths. Specifically, we introduce a weight function
 492 $\ell : E \rightarrow \mathbb{R}^+$, where $\ell(e)$ represents the length of tube e . The goal of OPTIMAL THREADING
 493 is now to minimize the **total length** of a threading T , defined as $\sum_{e \in T} \ell(e)$. This problem
 494 is equivalent to the weighted version of OPTIMAL LOCAL THREADING where we seek to
 495 minimize $\sum_{e \in E} \ell(e) x_e$ subject to constraints (C1)–(C4).

496 Our OPTIMAL THREADING algorithm hinges upon Lemma 10. Fortunately, this result
 497 holds for weighted graphs. In the proof of the lemma, we demonstrated that if any threading
 498 $\{x_e\}$ has $x_e \geq \Delta$ for some $e \in E$, then we can construct a strictly shorter threading
 499 $\{x'_e\}$ that remains consistent with constraints (C1)–(C4). Specifically, $x'_e \leq x_e$ for all
 500 $e \in E$ and $x'_e < x_e$ for at least one $e \in E$. Therefore, even in the weighted case we have
 501 $\sum_{e \in E} \ell(e) x'_e < \sum_{e \in E} \ell(e) x_e$ for any weight function $\ell : E \rightarrow \mathbb{R}^+$. Hence, an optimal
 502 threading never traverses an edge more than $\Delta - 1$ times as desired.

503 To adapt our OPTIMAL THREADING algorithm for the weighted scenario, we construct a
 504 graph similar to \hat{H} in Section 4.2, but with modified edge weights: a blue edge $(v_i, u\bar{v}_j)$ now
 505 has weight $\frac{1}{2}\ell(uv)$ instead of weight $\frac{1}{2}$, and a green edge $(u\bar{v}_i, w\bar{v}_j)$ has weight $\frac{1}{2}(\ell(uv) + \ell(wv))$
 506 rather than weight 1. The black edges continue to have zero weight. Denote this new graph
 507 by \tilde{H} .

508 By a similar proof to that of Theorem 15, we obtain a reduction from weighted OPTIMAL
 509 THREADING to minimum-weight perfect matching:

510 ► **Theorem 18.** *G has a threading of length $W + \sum_{e \in E(G)} \ell(e)$ with $\max_{e \in E(G)} x_e \leq \Delta - 1$
 511 if and only if \tilde{H} has a perfect matching of weight W .*

512 As before, an edge uv traversed by a threading corresponds to an edge $(u\bar{v}_i, \bar{v}v_i)$ that is
 513 *not* part of the perfect matching of \tilde{H} . Both endpoints of this edge must be matched with
 514 either a green or blue edge. Each such matching contributes $\frac{\ell(uv)}{2}$ to the matching's total

515 weight. Thus, we can show that a perfect matching in \tilde{H} with weight W corresponds to a
 516 threading of G of length $W + \sum_{e \in E} \ell(e)$.

517 **5 Special Cases**

518 Here we focus on two scenarios: OPTIMAL THREADING on cubic graphs and DOUBLE
 519 THREADING, where each edge can be traversed at most twice.

520 **5.1 Cubic Graphs**

521 If graph G is cubic, then by Lemma 10, an optimal threading of G visits each edge at most
 522 twice. Furthermore, in a perfect threading of G , if it exists, exactly one edge incident to each
 523 vertex is double-threaded due to constraint (C*4). Hence, it follows that G has a perfect
 524 threading if and only if G has a perfect matching. A perfect matching of G gives the set
 525 of edges to be double-threaded in a perfect threading. Every bridgeless cubic graph has a
 526 perfect matching [6]—it can be computed in $O(n \log^4 n)$ time [1]. In fact, if all bridges of a
 527 connected cubic graph G lie on a single path of G , then G has a perfect matching [8].

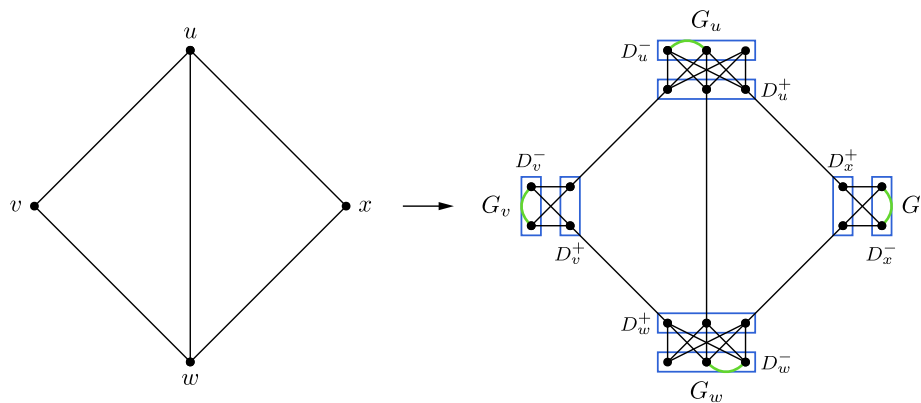
528 **5.2 Double Threading Problem**

529 In DOUBLE THREADING, the goal is to minimize the number of double-threaded edges or,
 530 equivalently, to maximize the number of edges visited only once. A solution to DOUBLE
 531 THREADING on a cubic graph also solves OPTIMAL THREADING on the same graph. This is due
 532 to the observation that either zero or two single-threaded edges are incident to each vertex in
 533 a solution to DOUBLE THREADING, which aligns with the reality of OPTIMAL THREADING on
 534 cubic graphs. By the same observation, a solution to DOUBLE THREADING matches the upper
 535 bound given in Lemma 9 for general graphs. We further note that DOUBLE THREADING may
 536 be reduced to the task of finding vertex-disjoint cycles with maximum collective length,
 537 which we solve below in Algorithm 2.

■ **Algorithm 2** Maximum Length Vertex-Disjoint Cycles

1. Construct a weighted graph G' from G (Figure 6):
 - a. For each vertex $v \in V$, create a complete bipartite graph $G_u = K_{d(v), d(v)}$ with zero-weight edges. Let D_u^- and D_u^+ denote the two disjoint vertex sets of this graph.
 - b. For each edge $uv \in E$, add an edge unit weight between a vertex of D_u^+ and a vertex of D_v^+ such that each vertex of D_u^+ and D_v^+ has exactly one edge incident to it.
 - c. For each subgraph G_v , add a zero-weight edge between any two vertices of D_v^- .
 2. Compute a maximum weight perfect matching M in G' .
 3. Return edge set $S \subseteq E$ of G corresponding to the weighted edges of M .
-

538 We sketch the intuition behind why matching M corresponds one-to-one to vertex disjoint
 539 cycles in G . Observe two cases for each u : (i) If M contains the edge of 1(c), then $d - 2$
 540 vertices in D_u^- match with the vertices in D_u^+ , leaving two vertices in D_u^+ to match with
 541 their neighbors in adjacent subgraphs; (ii) all vertices in D_u^+ are saturated via connections
 542 to D_u^- , otherwise. That is, each vertex u is in exactly one cycle (i) or none at all (ii).



■ **Figure 6** Illustration of constructing G' from G .

543 **Running-Time Analysis:** We begin our analysis of the running time of Algorithm 2 by first
 544 bounding the size of G' . Each subgraph G_v has $2d(v)$ vertices and $d(v)^2 + 1$ edges, and
 545 these subgraphs are connected via m edges. Because $\sum_{v \in V} d(v) = 2m$ and $\sum_{v \in V} d(v)^2 \leq$
 546 $m(2m/(n-1) + n-2)$ [7], we conclude that $V(G') = O(m)$ and $E(G') = O(nm)$.

547 The problems of finding a max-weight perfect matching and a min-weight perfect match-
 548 ing are symmetric: we can multiply edge weights by -1 to switch between the two prob-
 549 lems. It follows that we can apply the min-weight perfect matching algorithm proposed
 550 by Galil, Micali, and Gabow [10] in Step 2 of our algorithm. This procedure runs in
 551 $O(|V(G')||E(G')| \log |V(G')|) = O(nm^2 \log n)$ time, which dominates the $O(nm)$ construction
 552 time of G' in the first step. Hence, the overall running time of Algorithm 2 is $O(nm^2 \log n)$.

553 6 Future Work

554 Potential avenues for future work include developing tighter upper and lower bounds based
 555 on properties of the input graph and devising a more efficient solution to the general problem.

556 Practical challenges associated with the design of reconfigurable structures (Figure 1)
 557 inspire further intriguing problems. For instance, friction plays a central role in the deploy-
 558 ability of such structures — it determines the force required to draw the string through the
 559 system. According to the Capstan equation, friction increases exponentially with the sum of
 560 the absolute values of turning angles in the threading route. Therefore, a logical next step
 561 is to investigate a variant of OPTIMAL THREADING where the focus is on minimizing this
 562 frictional cost instead of the threading length.

563 References

- 564 1 Therese C. Biedl, Prosenjit Bose, Erik D. Demaine, and Anna Lubiw. Efficient algorithms for
 565 Petersen's matching theorem. *Journal of Algorithms*, 38(1):110–134, 2001.
- 566 2 J. A. (John Adrian) Bondy. *Graph Theory with Applications*. North Holland, New York,
 567 1980–1976.
- 568 3 Jeffrey Bosboom, Charlotte Chen, Lily Chung, Spencer Compton, Michael Coulombe, Erik D.
 569 Demaine, Martin L. Demaine, Ivan Tadeu Ferreira Antunes Filho, Dylan Hendrickson, Adam
 570 Hesterberg, Calvin Hsu, William Hu, Oliver Korten, Zhezheng Luo, and Lillian Zhang. Edge
 571 matching with inequalities, triangles, unknown shape, and two players. *Journal of Information*
 572 *Processing*, 28:987–1007, 2020. doi:10.2197/ipsjip.28.987.

- 573 4 Carina Chela. The original Finnish Christmas ornament. *this is FINLAND*, Dec 2013. URL:
574 <https://finland.fi/christmas/the-original-finnish-christmas-ornament/>.
- 575 5 Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM*
576 *Journal on Computing*, 14(1):210–223, 1985. doi:10.1137/0214017.
- 577 6 Maria Chudnovsky and Paul Seymour. Perfect matchings in planar cubic graphs. *Combinat-*
578 *orica*, 32(4):403–424, 2012. doi:10.1007/s00493-012-2660-9.
- 579 7 D. de Caen. An upper bound on the sum of squares of degrees in a graph. *Discrete Mathematics*,
580 185(1):245–248, 1998.
- 581 8 Alfred Errera. Du colorage des cartes. *Mathesis*, 36:56–60, 1922.
- 582 9 Herbert Fleischner. *Eulerian graphs and related topics*. North-Holland, Amsterdam, 1990.
- 583 10 Zvi Galil, Silvio Micali, and Harold Gabow. An $O(EV \log V)$ algorithm for finding a maximal
584 weighted matching in general graphs. *SIAM J. Comput.*, 15:120–130, Feb 1986. doi:10.1137/
585 0215009.
- 586 11 James Green. Beadwork in the arts of Africa and beyond. *The Metropolitan Museum*
587 *of Art*, Jul 2018. URL: [https://www.metmuseum.org/blogs/collection-insights/2018/
588 beadwork-in-arts-of-africa-and-beyond](https://www.metmuseum.org/blogs/collection-insights/2018/beadwork-in-arts-of-africa-and-beyond).
- 589 12 Yuki Igarashi, Takeo Igarashi, and Jun Mitani. Beady: Interactive beadwork design and
590 construction. *ACM Trans. Graph.*, 31(4), Jul 2012. doi:10.1145/2185520.2185545.
- 591 13 Joelle Jackson. Heavenly harmony: The universal language of Finnish himmeli. *Smithso-*
592 *nian Center for Folklife and Cultural Heritage*, Jul 2021. URL: [https://folklife.si.edu/
593 magazine/eija-koski-finnish-himmeli](https://folklife.si.edu/magazine/eija-koski-finnish-himmeli).
- 594 14 Bih-Yaw Jin and Chiachin Tsou. Bead sculptures and bead-chain interlocking puzzles inspired
595 by molecules and nanoscale structure. 2019. URL: [https://api.semanticscholar.org/
596 CorpusID:219636992](https://api.semanticscholar.org/CorpusID:219636992).
- 597 15 Alison Martin. Optimization of threading paths. *Twitter*, Nov 2021. URL: [https://twitter.
598 com/alisonmartin57/status/1461643652946698240](https://twitter.com/alisonmartin57/status/1461643652946698240).
- 599 16 Silvio Micali and Vijay V. Vazirani. An $O(\sqrt{|v|} \cdot |E|)$ algorithm for finding maximum matching
600 in general graphs. In *21st Annual Symposium on Foundations of Computer Science (sfcs 1980)*,
601 pages 17–27, 1980. doi:10.1109/SFCS.1980.12.
- 602 17 Rodakis. Push puppet. URL: <https://rodakis.com/Push-Puppet>.
- 603 18 Saskia Solomon. A vanishing craft reappears. *The New York Times*, Sep 2022. URL:
604 <https://www.nytimes.com/2022/09/28/style/a-vanishing-craft-reappears.html>.
- 605 19 Wikipedia. Straw mobile, Apr 2023. URL: https://en.wikipedia.org/wiki/Straw_mobile.