

# Finding a Shortest Curve that Separates Few Objects from Many

Therese Biedl  


University of Waterloo, Ontario, Canada

Éric Colin de Verdière 

LIGM, CNRS, Univ Gustave Eiffel, F-77454 Marne-la-Vallée, France

Fabrizio Frati  

Roma Tre University, Rome, Italy

Anna Lubiw  

University of Waterloo, Ontario, Canada

Günter Rote  

Freie Universität Berlin, Institut für Informatik, Takustraße 9, 14195 Berlin, Germany

---

## Abstract

We present a fixed-parameter tractable (FPT) algorithm to find a shortest curve that encloses a set of  $k$  required objects in the plane while paying a penalty for enclosing unwanted objects.

The input is a set of interior-disjoint simple polygons in the plane, where  $k$  of the polygons are *required* to be enclosed and the remaining *optional* polygons have non-negative penalties. The goal is to find a closed curve that is disjoint from the polygon interiors and encloses the  $k$  required polygons, while minimizing the length of the curve plus the penalties of the enclosed optional polygons. If the penalties are high, the output is a shortest curve that separates the required polygons from the others. The problem is NP-hard if  $k$  is not fixed, even in very special cases. The runtime of our algorithm is  $O(3^k n^3)$ , where  $n$  is the number of vertices of the input polygons.

We extend the result to a graph version of the problem where the input is a connected plane graph with positive edge weights. There are  $k$  required faces; the remaining faces are optional and have non-negative penalties. The goal is to find a closed walk in the graph that encloses the  $k$  required faces, while minimizing the weight of the walk plus the penalties of the enclosed optional faces. We also consider an inverted version of the problem where the required objects must lie outside the curve. Our algorithms solve some other well-studied problems, such as geometric knapsack.

**2012 ACM Subject Classification** Theory of computation  $\rightarrow$  Computational geometry; Theory of computation  $\rightarrow$  Parameterized complexity and exact algorithms

**Keywords and phrases** Enclosure, curve, separation, weakly simple polygon, Euler tour

**Funding** *Therese Biedl*: Natural Science and Engineering Research Council of Canada.

*Fabrizio Frati*: Supported by the European Union, Next Generation EU, Mission 4, Component 1, CUP J53D23007130006 PRIN proj. 2022ME9Z78 “NextGRAAL: Next-generation algorithms for constrained GRAPH visuALization”.

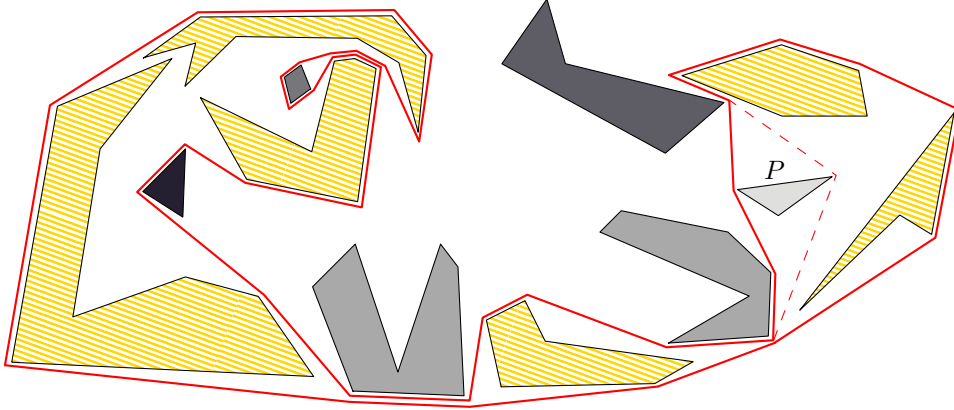
*Anna Lubiw*: Supported by the Natural Sciences and Engineering Research Council of Canada.

**Acknowledgements** This work was initiated at the Dagstuhl seminar 24072 “Triangulations in Geometry and Topology” in February 2024. We thank the organizers, Maike Buchin, Jean Cardinal, Arnaud de Mesmay, and Jonathan Spreer, as well as all participants, for the stimulating atmosphere. We also thank Alexander Wolff for significant contributions, Saul Schleimer for useful discussions, and the anonymous reviewers for their constructive feedback.

---

<sup>0</sup> This paper will appear, without appendices, in the Proceedings of the 41st International Symposium on Computational Geometry (SoCG 2025), Kanazawa, Japan; ed. Oswin Aichholzer and Haitao Wang; Leibniz International Proceedings in Informatics (LIPIcs), Vol. 332, pp. 15:1–15:16.

— Table of Contents —	
A42	
A43	<b>1 Introduction</b> <span style="float: right;"><b>3</b></span>
A44	1.1 Related work . . . . . 5
A45	<b>2 Preliminaries</b> <span style="float: right;"><b>6</b></span>
A46	2.1 Weakly simple polygons . . . . . 6
A47	2.2 Winding number and winding parity . . . . . 7
A48	<b>3 Our Common Framework: Enclosure-with-Penalties</b> <span style="float: right;"><b>8</b></span>
A49	<b>4 Dynamic Programming Algorithm</b> <span style="float: right;"><b>8</b></span>
A50	4.1 Dynamic programming recursion . . . . . 9
A51	4.2 Extracting the solution . . . . . 10
A52	<b>5 Uncrossing Algorithm and Final Output <math>W_{\text{ALG}}</math></b> <span style="float: right;"><b>11</b></span>
A53	<b>6 Correctness Proof</b> <span style="float: right;"><b>13</b></span>
A54	<b>7 Reducing the Runtime with a Dijkstra-Style Algorithm</b> <span style="float: right;"><b>14</b></span>
A55	<b>8 The Inverted Problem</b> <span style="float: right;"><b>15</b></span>
A56	<b>9 Applications, Extensions, and Open Problems</b> <span style="float: right;"><b>17</b></span>
A57	– <b>References</b> <span style="float: right;"><b>17</b></span>
A58	<b>A Details for Section 2: Weakly Simple Polygons or Walks</b> <span style="float: right;"><b>18</b></span>
A59	<b>B Details for Section 3: Common Framework</b> <span style="float: right;"><b>21</b></span>
A60	<b>C Details for Section 4: Dynamic Programming Algorithm</b> <span style="float: right;"><b>22</b></span>
A61	C.1 Runtime of the dynamic programming algorithm . . . . . 22
A62	C.2 Details for Section 4.2: Extracting the solution . . . . . 22
A63	<b>D Details for Section 5: Uncrossing Algorithm</b> <span style="float: right;"><b>25</b></span>
A64	<b>E Details for Section 6: Correctness Proof</b> <span style="float: right;"><b>26</b></span>
A65	<b>F Details for Section 7: Reducing the Runtime</b> <span style="float: right;"><b>30</b></span>
A66	F.1 Setting up a modified system of equations . . . . . 30
A67	F.2 The algorithm . . . . . 31
A68	F.3 Runtime analysis . . . . . 33
A69	F.4 Preprocessing for quickly determining the penalty of a triangle . . . . . 33
A70	<b>G Details for Section 8: Adaptations for the Inverted Problem</b> <span style="float: right;"><b>34</b></span>
A71	G.1 The dynamic programming recursion . . . . . 34
A72	G.2 Adapting the correctness proof for the inverted problem . . . . . 35
A73	<b>H Illustration for Section 9: Splitting a Surface by a Curve</b> <span style="float: right;"><b>36</b></span>
A74	<b>I Point Objects</b> <span style="float: right;"><b>36</b></span>
A75	<b>J Negative Penalties, or Rewards</b> <span style="float: right;"><b>39</b></span>
A76	<b>K Exponential Lower Bounds</b> <span style="float: right;"><b>39</b></span>
A77	<b>L Weakly Simple Immersed Polygons</b> <span style="float: right;"><b>42</b></span>
A78	<b>M The Geometric Knapsack Algorithm</b> <span style="float: right;"><b>44</b></span>



**Figure 1** The GEOMETRIC-ENCLOSURE-WITH-PENALTIES problem. The weakly simple polygon  $W$  (in red) encloses the required polygons  $R$  (yellow hatched) while paying a penalty for enclosing any of the optional polygons  $O$  (in gray, darker for larger penalties). Overlapping paths are slightly displaced for visibility. Changing  $W$  via the dashed path to put polygon  $P$  outside would be preferable if the increase in length is less than the penalty of  $P$ .

## 1 Introduction

We investigate the separation problem of finding a shortest curve that encloses a subset of objects while excluding other objects. A very basic setting is for points in the plane: given  $n$  points in the plane and a subset of size  $k$ , find a minimum-perimeter polygon containing the specified  $k$  points and excluding the other  $n - k$  points. This problem is NP-hard when  $k$  may be large, as proved by Eades and Rappaport [13] for the case  $k = n/2$  via a simple reduction from the Travelling Salesman Problem.

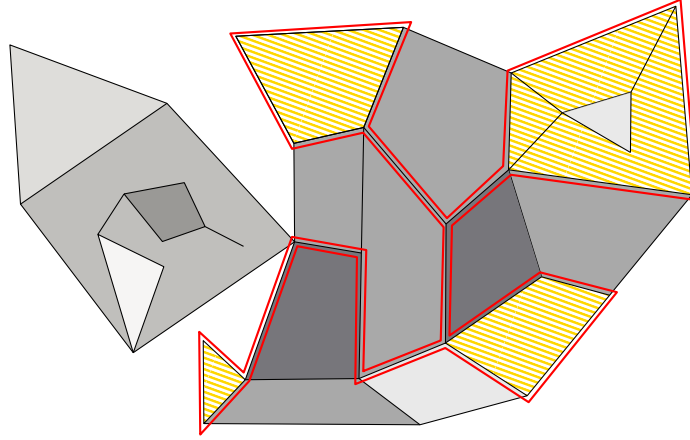
As a special case of our main result, we give the first algorithm for this problem that is fixed-parameter tractable (FPT) in  $k$ . Our result is far more general and applies in two settings, a geometric setting and a graph-theoretic setting.

**Geometric-Enclosure-with-Penalties.** Here we generalize from objects that are points to objects that are interior-disjoint simple polygons in the plane, and we generalize to a weighted form of exclusion.

**Input.** The input is a set of simple interior-disjoint polygons partitioned into a set  $R$  of  $k$  *required polygons* and the remaining set  $O$  of *optional polygons*. Each optional polygon  $P \in O$  comes with a non-negative *penalty*  $\pi_P$  where we allow  $\pi_P = +\infty$ .

**Output.** The goal is to find a weakly simple polygon  $W$  that does not intersect the interior of any input polygon and encloses all polygons of  $R$  while minimizing the *cost*  $c(W)$ , which is defined to be the Euclidean length of  $W$  plus the penalties of the polygons of  $O$  that are inside  $W$ . See Figure 1 for an example. A polygon with penalty  $+\infty$  must be excluded. A polygon with penalty 0 may be included or excluded without making a difference, so it only acts as an obstacle to the solution curve. As Figure 1 illustrates, the problem would be ill-defined if we required the solution curve  $W$  to be a simple polygon. The natural condition is that  $W$  should be a *weakly simple polygon*, whose boundary may touch or overlap itself but not cross itself. We give a precise definition in Section 2.1. An important property is that a weakly simple polygon encloses a well-defined region.

Our first main result is:



■ **Figure 2** The GRAPH-ENCLOSURE-WITH-PENALTIES problem. The colors have the same meaning as in Figure 1. A weakly simple closed walk  $W$  which is a solution for the instance is red (bold). The edge weights, which are not shown, are unrelated to the Euclidean lengths.

A106 ▶ **Theorem 1.** GEOMETRIC-ENCLOSURE-WITH-PENALTIES for  $k$  required polygons can be  
 A107 solved in  $O(3^k n^3)$  time and  $O(2^k n^2)$  space, if the input polygons have  $n$  vertices in total.

A108 If all objects are points, this can be handled by approximating each point by a small triangle.  
 A109 An alternative non-approximate solution for an arbitrary mix of point and polygon objects  
 A110 appears in Appendix I.

A111 **Graph-Enclosure-with-Penalties.** In this setting, the objects are faces of a plane graph.

A112 **Input.** The input is a simple connected plane graph  $G$  and positive edge weights. The  
 A113 bounded faces of  $G$  are partitioned into a set  $R$  of  $k$  **required faces** and the remaining set  
 A114  $O$  of **optional faces**. Each optional face  $F$  has a penalty  $\pi_F$  from  $\mathbb{R}_{\geq 0} \cup \{+\infty\}$ .

A115 **Output.** The goal is to find a weakly simple closed walk  $W$  in  $G$  such that faces of  $R$   
 A116 are inside  $W$  while minimizing the cost  $c(W)$  which is defined to be the sum of the weights  
 A117 of the edges of  $W$  plus the penalties of the faces of  $O$  that are inside  $W$ . See Figure 2 for an  
 A118 example. Intuitively, a **weakly simple** closed walk is one without crossings; we give a more  
 A119 precise definition in Appendix A. For a weakly simple closed walk, the notions of inside and  
 A120 outside are well-defined.

A121 Our second main result is:

A122 ▶ **Theorem 2.** GRAPH-ENCLOSURE-WITH-PENALTIES can be solved in  $O(3^k n^3)$  time and  
 A123  $O(2^k n^2)$  space, where  $k$  is the number of required faces and  $n$  is the number of vertices of  $G$ .

A124 **A common framework.** Although the two settings described above seem different, we resolve  
 A125 them into a common geometric framework, which we call ENCLOSURE-WITH-PENALTIES. Our  
 A126 algorithm applies to this general problem. The basic idea is to transform the graph problem  
 A127 into a geometric problem by taking a straight-line embedding of the graph. The bounded faces  
 A128 of the graph become polygons slightly more general than simple polygons. We also consider  
 A129 the outer face as an unbounded polygon. Then the “free space” between the polygons consists  
 A130 only of the graph edges. This gives us a geometric problem, albeit with arbitrary positive  
 A131 edge weights defined on edges that have a polygon on each side. In Section 3 we define the  
 A132 ENCLOSURE-WITH-PENALTIES problem by generalizing the GEOMETRIC-ENCLOSURE-WITH-  
 A133 PENALTIES problem to include these instances.

A134 We remark that the resulting algorithm for Theorem 2 makes essential use of the straight-  
 A135 line embedding of the input graph. In particular, the subproblems that we solve depend on  
 A136 the embedding. This imposition of geometry seems artificial, but oddly enough, we do not  
 A137 know how to formulate our algorithm in a purely combinatorial setting.

A138 **Our approach.** We use dynamic programming (Section 4) to build a polygon  $W$  that is  
 A139 locally correct—we use segments that do not intersect the interior of any object and we  
 A140 account for required objects and tally the penalties as we add triangles to  $W$ . We will prove  
 A141 that the cost computed by the algorithm is correct, but this is tricky because  $W$  itself will  
 A142 not necessarily be weakly simple. “Inside” is no longer well-defined. Instead, we use winding  
 A143 numbers to give a measure of the cost of  $W$  that matches the cost computed by the algorithm.

A144 In Section 5 we give an algorithm to *uncross*  $W$  to a weakly simple polygon without  
 A145 increasing its cost, which provides our final output. Correctness of the whole algorithm is  
 A146 proved in Section 6.

A147 The run-time for the Uncrossing Algorithm is dominated by the run-time for the dynamic  
 A148 program. To obtain our claimed run-time we speed up the dynamic program in Section 7.

A149 **Lower bounds.** To complement our algorithms we prove that, under the Exponential Time  
 A150 Hypothesis (ETH), the GEOMETRIC- and GRAPH-ENCLOSURE-WITH-PENALTIES problems  
 A151 cannot be solved in  $2^{o(k)} \cdot n^{O(1)}$  time, implying that the linear dependence on  $k$  in the  
 A152 exponent of the running time of our algorithms is the best possible assuming ETH. The  
 A153 proof is a reduction from unweighted PLANAR STEINER TREE, which admits a lower bound  
 A154 by a result by Marx, Pilipczuk, and Pilipczuk [20, Theorem 1.2]. See Appendix K.

A155 **Swapping the inside with the outside.** We extend our algorithm to an *inverted* version of  
 A156 the ENCLOSURE-WITH-PENALTIES problem where the required objects have to be *outside*  $W$ ,  
 A157 and the objective is to minimize the length of  $W$  plus the penalties of the polygons of  $O$   
 A158 that are *outside*  $W$ . The runtime remains the same, see Section 8. This algorithm provides a  
 A159 new faster solution to the geometric knapsack problem discussed below.

A160 **Negative penalties.** We can allow some number  $\ell$  of objects with negative penalties  
 A161 (rewards); in this case, the runtime is increased by a factor of  $3^\ell$ . See Appendix J.

## A162 1.1 Related work

A163 Cut problems and separator problems in graphs have a long history, and separation problems  
 A164 in geometric settings are a natural and well studied counterpart.

A165 **Geometric knapsack problem.** Geometric separation problems were first explored by  
 A166 Eades and Rappaport [13] (as discussed above) and by Arkin, Khuller, and Mitchell, who  
 A167 introduced the *geometric knapsack problem* [4], which corresponds to the *inverted* version of  
 A168 the GEOMETRIC-ENCLOSURE-WITH-PENALTIES problem in the special case where there are  
 A169 no required objects. (In their equivalent formulation, each object has a finite nonnegative  
 A170 value, and the goal is to compute a curve that maximizes the total value of the enclosed  
 A171 objects minus its length.) They gave an algorithm with running time  $O(n^4)$  [4, Theorem 6].  
 A172 In Appendix M, we note some issues regarding weak simplicity and interior/exterior of their  
 A173 output. Since there are no required objects, our algorithm for the inverted problem solves  
 A174 the geometric knapsack problem in time  $O(n^3)$ .

A175 **Relation to homotopy and homology.** Our problem has a topological flavor and is therefore,  
A176 in principle, amenable to homotopy and homology techniques. However, these techniques  
A177 are unlikely to lead to algorithms that are FPT in  $k$ , even assuming only infinite penalties.  
A178 In particular, for the GRAPH-ENCLOSURE-WITH-PENALTIES problem, enumerating a set of  
A179 candidate homotopy classes, i.e., the possible ways how a solution winds around the objects  
A180 to enclose the required objects and avoid the most undesirable ones, is possible using a  
A181 technique by Chambers, Colin de Verdière, Erickson, Lazarus, and Whittlesey [7] (specifically,  
A182 because the exchange argument [7, Proposition 4.2] is also valid for our problem), but this  
A183 results in an algorithm with runtime  $K^{O(K)} \cdot n \log n$ , where  $K$  is the number  $k$  of required  
A184 objects *plus* the number of objects with nonzero penalty. Similarly, the technique of *homology*  
A185 *covers*, by Chambers, Erickson, Fox, and Nayyeri [8], is applicable, but yields algorithms with  
A186 runtime  $K^{O(K)} \cdot n \log \log n$  or  $2^{O(K)} \cdot n \log n$ , thus again with an exponential dependence  
A187 on  $K$ . If there are many objects with nonzero penalty, our algorithm with runtime  $O(3^k n^3)$   
A188 is faster.

A189 **Specifying only the number of objects to be enclosed.** If we are just given a set of  $n$   
A190 points in general position and the exact *number*  $k \leq n$  of points to be enclosed, a minimum-  
A191 perimeter polygon enclosing at least  $k$  points is convex, contains exactly  $k$  points, and can be  
A192 found in polynomial time by an algorithm of Eppstein, Overmars, Rote, and Woeginger [14,  
A193 Corollary 5.3, Case 3]. This algorithm could for example be used to identify an unusual  
A194 cluster in an otherwise uniformly distributed point set. However, if the input consists of  
A195 polygons instead of points, we are not aware of a better method than guessing the  $k$  polygons  
A196 to be enclosed and applying our main result, resulting in an algorithm of running time  
A197  $O(\binom{N}{k} 3^k n^3) = O(N^k n^3)$  if there are  $N$  objects.

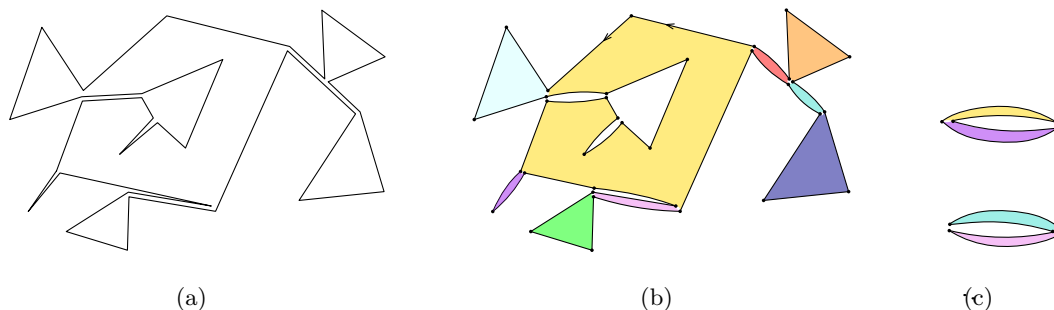
A198 **More variations.** Some related separation and enclosure problems have been recently studied.  
A199 Abrahamsen, Giannopoulos, Löffler, and Rote [1] investigated the problem of separating two  
A200 or more groups of polygonal objects by a shortest system of fences. These fences may form  
A201 an arbitrary plane graph, not necessarily a cycle. For separating *two* groups of polygons by  
A202 a shortest fence, there is an algorithm of running time  $O(n^4 \log^3 n)$ , based on minimum-cut  
A203 techniques [1]. For separating more than two groups, only approximation algorithms are  
A204 known. While we study a problem in the same spirit, a key difference is that we require a  
A205 single (weakly) simple cycle, which makes the techniques of this article not applicable for us.

A206 Chan, He, and Xue [9] studied the problem of choosing a smallest subset of objects  
A207 from a given set whose union encloses a given point set. For this problem, there are only  
A208 approximation algorithms. Klost, van Kreveld, Perz, Rote, and Tkadlec [18] have recently  
A209 investigated optimum “blob-trees”, where the objective is to *connect* a set of points by a  
A210 combination of curves and edges.

## A211 2 Preliminaries

### A212 2.1 Weakly simple polygons

A213 A polygon is *weakly simple* if it has fewer than three vertices, or it has at least three vertices  
A214 and for any  $\varepsilon > 0$ , the vertices can be perturbed by at most  $\varepsilon$  to yield a simple polygon [2, 10].  
A215 We traverse a weakly simple polygon counterclockwise, i.e., with the interior to the left of  
A216 each edge. Our proof uses a combinatorial characterization of a weakly simple polygon in  
A217 terms of a non-crossing Euler tour in a plane multigraph (Lemma 16 in Appendix A). This



■ **Figure 3** (a) A weakly simple polygon drawn via its  $\varepsilon$ -approximation. (b) The edges, after subdividing at interior vertices (forks), are partitioned into interior faces. Four faces are corridors and five are chambers. The largest chamber (in yellow) is almost-simple but not simple. (c) Non-uniqueness of the faces for a weakly simple polygon that traverses a line segment four times. In the top figure the two vertices on the left are transition vertices; this is reversed in the bottom figure.

A218 allows us to partition the edges of a weakly simple polygon into boundary walks of interior  
A219 faces, see Figure 3. Note that we first subdivide an edge when a vertex lies in its interior.

A220 A vertex of a weakly simple polygon with incoming edge  $e$  and outgoing edge  $f$  is a  
A221 *transition vertex* if  $e$  and  $f$  belong to different interior faces. An interior face of two edges  
A222 is a *corridor* and an interior face of more than two edges is a *chamber*. A chamber is not  
A223 necessarily a simple polygon, but it is almost simple (see Figure 3(b)). More formally, a  
A224 *bounded almost-simple polygon* is the boundary walk of an interior face of a connected  
A225 straight-line graph drawing in the plane. We also allow an *unbounded almost-simple*  
A226 *polygon* by traversing the boundary of the outer face clockwise. An almost-simple polygon  
A227 has a connected interior and a bounded almost-simple polygon can be triangulated. Almost-  
A228 simple polygons play two roles: the bounded ones arise as chambers; and our general  
A229 ENCLOSURE-WITH-PENALTIES problem allows almost-simple input polygons (including a  
A230 single unbounded one).

A231 All these concepts are made rigorous in Appendix A. We note that the partition of the  
A232 edges of a weakly simple polygon into interior faces (and hence the definition of corridors and  
A233 transition vertices) is not unique, see Figure 3(c). This non-uniqueness, which is inherent in  
A234 the  $\varepsilon$ -approximation definition of weakly simple polygons, does not affect our proofs.

## A235 2.2 Winding number and winding parity

A236 Our algorithm will construct intermediate polygons that are not necessarily weakly simple,  
A237 so we will find it useful to generalize “enclosed by” in terms of winding numbers. Let  $W$  be  
A238 a polygon and let  $x$  be a point not lying on a vertex or edge of  $W$ . The *winding number*  
A239  $\text{wind}(W, x)$  of  $x$  with respect to  $W$  is defined as follows. Take a ray  $\rho$  from  $x$  that avoids  
A240 vertices of  $W$ . If an edge of  $W$  crosses  $\rho$  from right to left, we count this as  $+1$ ; a crossing  
A241 from left to right is counted as  $-1$ , and the total count gives the winding number. This is  
A242 well-defined independent of the choice of  $\rho$ . The winding number is undefined for points  $x$   
A243 on  $W$ . Observe that, for a weakly simple polygon  $W$  traversed counterclockwise, point  $x$  lies  
A244 in the interior of  $W$  if and only if  $\text{wind}(W, x) = 1$ . The *winding parity* of  $x$  with respect  
A245 to  $W$  is  $\text{wind}(W, x) \bmod 2$ .

A246

### 3 Our Common Framework: Enclosure-with-Penalties

A247

In this section we formally define the ENCLOSURE-WITH-PENALTIES problem that provides a common framework for both the geometric and graph settings.

A248

#### Input:

A249

A250

- A set of interior-disjoint almost-simple polygons in the plane. We allow a single polygon to be unbounded. We subdivide polygon edges to ensure that no polygon vertex lies in the interior of an edge of another polygon. The *free space* is the plane minus the interiors of the polygons.

A251

A252

A253

A254

- A partition of the input polygons into a set  $R$  of  $k$  *required* polygons and the remaining set  $O$  of *optional* polygons. If there is an unbounded polygon, it must lie in  $O$ .

A255

A256

- For each polygon  $P \in O$ , a *penalty*  $\pi_P \in \mathbb{R}_{\geq 0} \cup \{+\infty\}$ .

A257

A258

A259

A260

A261

- The weight  $w_{ab}$  of a line segment  $ab$  in the free space is its Euclidean length, except for *squeezed edges*. A squeezed edge is a polygon edge that is incident to polygons on both sides. We may specify an *arbitrary* positive weight for a squeezed edge. Subsegments of a squeezed edge get proportional weight, and combinations of different squeezed or non-squeezed segments have their weights added.

A262

**Output:** A weakly simple polygon  $W$  that lies in the free space and contains all polygons of  $R$  while minimizing the *cost*  $c(W)$ , which is defined as

A263

A264

$$c(W) := w(W) + \pi(W), \tag{1}$$

A265

where  $w(W)$  is the sum of the weights of the edges of  $W$ , and  $\pi(W)$  is the sum of the penalties of the polygons of  $O$  that are inside  $W$ . Our main result is:

A266

A267

► **Theorem 3.** ENCLOSURE-WITH-PENALTIES for  $k$  required polygons can be solved in  $O(3^k n^3)$  time and  $O(2^k n^2)$  space, if the input polygons have  $n$  vertices in total.

A268

A269

Theorem 1 is an immediate consequence of Theorem 3. Theorem 2 follows from Theorem 3 via a straight-line embedding of the graph, as outlined in Section 1 and detailed in Appendix B.

A270

A271

The ENCLOSURE-WITH-PENALTIES problem as defined above does not allow point objects. (They are not almost-simple.) Appendix I shows how to deal with point objects.

A272

A273

### 4 Dynamic Programming Algorithm

A274

The algorithm builds a polygon composed of free-space edges, where a *free-space edge* is an inclusionwise minimal line segment in the free space whose endpoints are vertices of the input polygons. (Equivalently, a free-space edge is a visibility edge that contains no polygon vertex in its interior.) We prove in Section 6 that this restriction to free-space edges is valid. We refer to a solution interchangeably as a polygon or as a closed walk in the graph of free-space edges.

A275

A276

A277

A278

A279

A280

The intuition for the algorithm is based on the decomposition of a weakly simple polygon  $W$  into corridors and chambers joined at “cutpoints”, see Figure 3. A cutpoint separates  $W$  into subpolygons and partitions the set of enclosed objects. Our first type of subproblem finds polygons that enclose a specified subset of  $R$  and go through a specified vertex.

A281

A282

A283

A284

A285

A corridor is a digon, and a chamber can be triangulated by adding chords, where a chord may cut through polygons. We therefore use digons and triangles as the basic building blocks to construct our solutions. A chord cuts off part of the solution. Our second type of subproblem finds polygons that use a walk of free-space edges between two given vertices  $p$  and  $q$  together with the chord  $pq$  (called the *mouth*) to enclose a specified subset of  $R$ .

A286

A287

A288

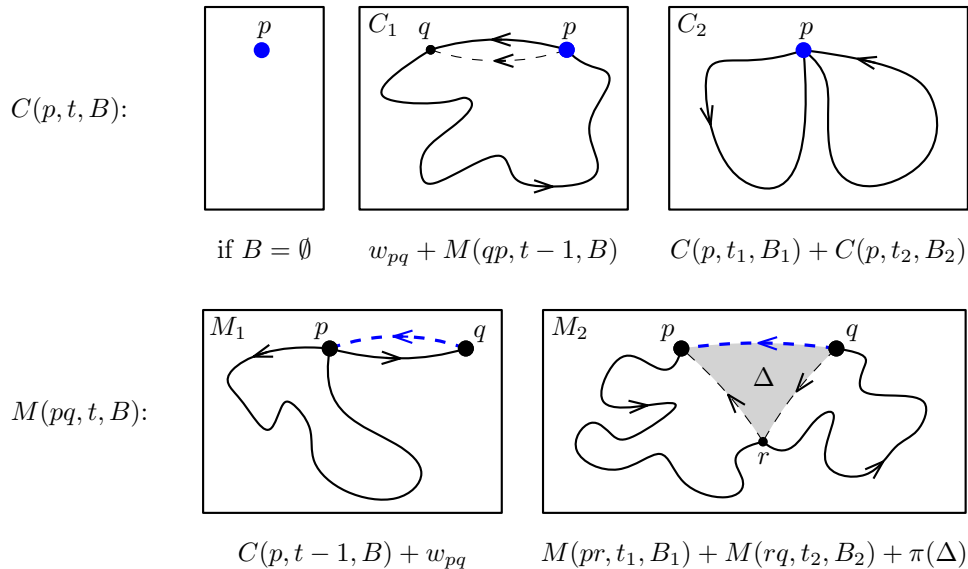
A289 Since a mouth may cut through polygons, we choose a *reference point*  $r_P$  in the interior  
A290 of every input polygon  $P$ , and aim to enclose  $r_P$  for  $P \in R$ . Observe that a weakly simple  
A291 polygon  $W$  in the free space encloses  $P$  if and only if  $r_P$  lies in the interior of  $W$ . (For  
A292 convenience, we could insist that the reference point avoids all potential mouth edges, but  
A293 in fact, we will be able to properly resolve the ambiguities resulting from such degenerate  
A294 choices of reference points.)

A295 The dynamic program explicitly keeps track of the subset of required objects that are  
A296 enclosed by partial solutions ( $2^k$  possibilities). However, when combining two partial solutions,  
A297 the algorithm does not have enough information to check whether they cross. Thus, we allow  
A298 self-crossing solutions. In particular, our use of the word “enclosing” is aspirational, and  
A299 will only be made precise in terms of winding numbers, see Section 4.2. When we state the  
A300 algorithm, we invite the reader to think of a weakly simple solution without crossings.

A301 **Types of subproblems.** A subproblem of type  $C$  (“closed”) is rooted at a vertex  $p$ , and we  
A302 build a closed walk that goes through  $p$  and is composed of free-space edges. A subproblem  
A303 of type  $M$  (“mouth”) is rooted at a segment  $pq$  between vertices of input polygons, called the  
A304 *mouth*, and we build an open walk of free-space edges from  $p$  to  $q$ ; adding segment  $qp$  closes  
A305 the walk. Note that the mouth need not be a free-space edge; it may cut through objects. In  
A306 addition to the root, each subproblem has two more parameters,  $B$  and  $t$ : The set  $B \subseteq R$   
A307 specifies the precise subset of required objects that must be enclosed, and the integer  $t \geq 0$   
A308 is an upper bound on the number of edges of the walk. In each case, the subproblem looks  
A309 for a *minimum-cost* solution with the required properties.

#### A310 4.1 Dynamic programming recursion

A311 We now give recursive formulas for  $C$  and  $M$ , preceded in each case by an explanation of the  
A312 formulas. The formulas with the respective partitions of the walk are illustrated in Figure 4.



■ **Figure 4** Cases of the recursion. Solid edges are free-space edges; dashed edges are mouths.

A313 For  $C(p, t, B)$  we have two base cases: if  $B = \emptyset$ , then the shortest closed walk is just  
A314 the point  $p$  and its cost is 0 (Equation (2)); and if  $B \neq \emptyset$  and  $t \leq 1$ , there is no solution,

A315 and we set the cost to  $\infty$  (Equation (3)). Otherwise we have two general cases: the closed  
A316 walk uses an edge  $pq$  of the free space (for some  $q$ ) plus a solution  $M(qp, t-1, B)$  (Equation  
A317 (4)); or the closed walk is composed of two smaller closed walks that both go through  $p$   
A318 (Equation (5)). The notation  $\sqcup$  means disjoint union: we partition the objects in  $B$  into two  
A319 sets, each “enclosed” by exactly one of the two closed walks.

A320 The base cases define  $C(p, t, B)$  for  $B = \emptyset$  and for  $t \leq 1$ :

$$A321 \quad C(p, t, \emptyset) := 0, \text{ for } t \geq 0 \quad (2)$$

$$A322 \quad C(p, t, B) := \infty, \text{ for } B \neq \emptyset \text{ and } t \leq 1 \quad (3)$$

A323 In the general case, for  $B \neq \emptyset$  and  $t \geq 2$ , we set

$$A324 \quad C(p, t, B) := \min\{C_1, C_2\}, \text{ where}$$

$$A325 \quad C_1 := \min\{w_{pq} + M(qp, t-1, B) \mid pq \text{ is a free-space edge}\} \quad (4)$$

$$A326 \quad C_2 := \min\{C(p, t_1, B_1) + C(p, t_2, B_2) \mid t = t_1 + t_2; B = B_1 \sqcup B_2; B_1, B_2 \neq \emptyset\} \quad (5)$$

A327 For  $M(pq, t, B)$  there are two possibilities: if  $pq$  is a free-space edge, we can use a closed  
A328 walk at  $p$  plus the edge  $pq$  (Equation (6)); or we can attach a triangle  $\Delta = prq$  to the  
A329 mouth  $pq$  (Equation (7)). In the first case we add the weight of the edge  $pq$ . In the triangle  
A330 case we take into account the polygons  $R(\Delta)$  with reference points in  $\Delta$ , where we consider  $\Delta$   
A331 to be closed on  $pr$  and  $rq$  and open on  $pq$ , and we define  $\pi(\Delta)$  to be the sum of the penalties  
A332 of polygons of  $O$  with reference points in  $\Delta$ .

A333  $M(pq, t, B)$  is defined only for  $t \geq 1$ :

$$A334 \quad M(pq, t, B) := \min\{M_1, M_2\}, \text{ where}$$

$$A335 \quad M_1 := \begin{cases} C(p, t-1, B) + w_{pq} & \text{if } pq \text{ is a free-space edge} \\ \infty, & \text{otherwise} \end{cases} \quad (6)$$

$$A336 \quad M_2 := \min\{M(pr, t_1, B_1) + M(rq, t_2, B_2) + \pi(\Delta) \mid \quad (7)$$

$$A337 \quad \Delta = prq \text{ is a counterclockwise triangle,}$$

$$A338 \quad t = t_1 + t_2, t_1 \geq 1, t_2 \geq 1, B = B_1 \sqcup B_2 \sqcup R(\Delta)\}$$

A339 As we shall see later in Lemma 13, the optimal walk has at most  $6n$  edges. Thus, we define  
A340 the solution to the whole problem as

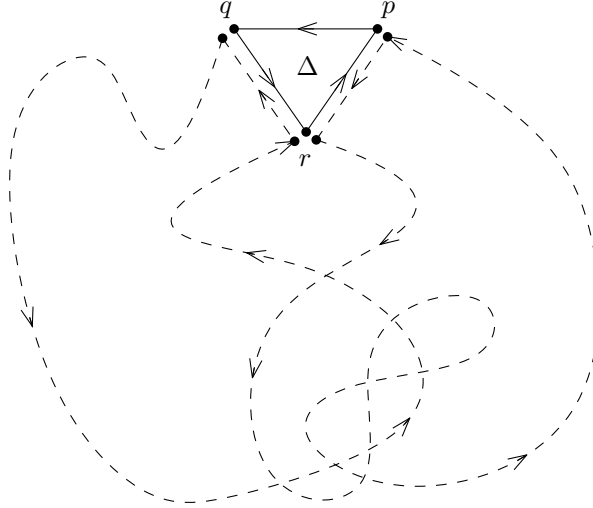
$$A341 \quad c_{\text{DP}} := \min\{C(p, 6n, R) \mid p \text{ a vertex}\}. \quad (8)$$

A342 When we allow point objects, the algorithm needs a few refinements, see Appendix I. In  
A343 the following sections we prove that  $c_{\text{DP}}$  is the correct value. Although not required by our  
A344 proof, we note for completeness in Appendix L that the class of polygons over which the  
A345 algorithm optimizes is the class of *immersed* or *self-overlapping* weakly simple polygons.

A346 **Runtime and Space.** A routine analysis shows that the runtime of the dynamic program  
A347 is  $O(3^k n^5)$ , see Appendix C.1. A more efficient version of the dynamic program, given in  
A348 Section 7, eliminates the parameter  $t$  and runs in time  $O(3^k n^3)$ .

## A349 4.2 Extracting the solution

A350 With every finite value computed in the dynamic program we can naturally associate an open  
A351 or closed walk of free-space edges that has this value as its cost (with “partially enclosed”



■ **Figure 5** Gluing together closed walks, which may cross each other and are possibly self-crossing.

A352 objects taken into account according to their reference points): Figure 4 illustrates how  
 A353 each value is computed from the values of subproblems; the walks corresponding to those  
 A354 subproblems are composed accordingly. (For more details, see Appendix C.2.) We will prove  
 A355 in Section 6 that  $c_{DP}$  is finite; so the associated closed walk exists:

A356 ▶ **Definition 4.**  $W_{DP}$  is the polygon associated with the optimum solution value  $c_{DP}$  in (8).

A357 The polygon  $W_{DP}$  uses free-space edges, but it might not be weakly simple, and there is  
 A358 no notion of enclosed objects. Instead, we use winding numbers: we show that the reference  
 A359 point of any object in  $R$  has winding number 1 in  $W_{DP}$ , and we define a cost measure for  
 A360  $W_{DP}$  in terms of winding numbers and prove equality with  $c_{DP}$ .

A361 ▶ **Definition 5.** For any polygon  $W$  in the free space, define the **cost** to be

A362 
$$c(W) := w(W) + \sum_{P \in O} \text{wind}(W, r_P) \cdot \pi_P.$$

A363 When  $W$  is a counterclockwise weakly simple polygon, this matches the previous definition  
 A364  $c(W) = w(W) + \pi(W)$ , see Equation (1). The solution  $W_{DP}$  produced by the algorithm has  
 A365 the following properties:

A366 ▶ **Lemma 6.**

- A367 (A)  $c_{DP} = c(W_{DP})$ ;  
 A368 (B) for all  $P \in R$ ,  $\text{wind}(W_{DP}, r_P) = 1$ ;  
 A369 (C) for all points  $x$  that do not lie on  $W_{DP}$ ,  $\text{wind}(W_{DP}, x) \geq 0$ .

A370 Lemma 6 is proved in Appendix C.2 by induction as the dynamic program builds solutions  
 A371 to subproblems by gluing together open/closed walks. The induction must apply also to  
 A372 open walks, and we use the fact that winding numbers add when gluing walks together, see  
 A373 Figure 5.

A374 **5 Uncrossing Algorithm and Final Output  $W_{ALG}$**

A375 The final step of our algorithm “uncrosses” the closed walk  $W_{DP}$  produced by the dynamic  
 A376 program and turns it into a weakly simple polygon  $W_{ALG}$  without increasing the cost. To do

A377  
A378  
A379  
A380  
A381  
A382  
A383

so, we cut it into subpaths, eliminate some, and reorder the rest.

Our algorithm uses a known result about taking a plane multigraph (specified via its rotation system) and finding a *non-crossing Euler tour* in which successive visits to a vertex do not cross each other. (See Appendix A for more detailed definitions.) The existence of such a tour in an Eulerian plane multigraph is an easy exercise, see [22] or [24, Lemma 3.1], and a linear-time algorithm was given by Akitaya and Tóth [3]. We summarize it in the following proposition, and give a self-contained proof in Appendix D.

A384  
A385  
A386

► **Proposition 7** (Uncrossing Eulerian plane multigraphs). *Given a plane connected Eulerian multigraph  $H$  with  $m$  edges, specified by its combinatorial map, we can, in  $O(m)$  time, compute a non-crossing Euler tour of  $H$ .*

A387  
A388  
A389

We now sketch our algorithm to uncross any polygon  $W$  to a weakly simple polygon  $W'$ . An *interior crossing* is a point that is in the relative interiors of two (or more) edges that are not collinear. A *fork* (or *interior vertex*) is a vertex in the relative interior of an edge.

A390

► **Algorithm 8** (Uncrossing Algorithm).

A391  
A392  
A393  
A394  
A395  
A396

1. *Subdivide every edge of  $W$  at every fork and interior crossing.*
2. *In the resulting multiset of edges (line segments in the plane) reduce multiplicities to 1 or 2 by repeatedly discarding pairs of equal line segments. The result is a plane connected Eulerian multigraph.*
3. *Apply Proposition 7 to find a non-crossing Euler tour. This corresponds to a weakly simple polygon  $W'$ .*

A397  
A398  
A399  
A400

In Appendix D we give further details of the algorithm and an implementation with runtime  $O(t \log t + s)$  where  $t$  is the number of edges of  $W$  and  $s \in O(t^2)$  is the number of interior crossings of  $W$  (with multiple crossings counted only once). For input  $W_{\text{DP}}$  we show that there are no interior crossings, so the runtime is  $O(n \log n)$ .

A401  
A402

We use the following important property of the Uncrossing Algorithm. Recall the definition of winding parity from Section 2.2.

A403  
A404

► **Lemma 9.** *Every point  $x$  in the plane that does not lie on  $W$  has the same winding parity in  $W$  and in  $W'$ .*

A405  
A406  
A407  
A408

**Proof.** For any ray  $r$  from  $x$  to infinity that avoids vertices of  $W$ , the parity of the number of edges it crosses is the same for  $W$  and  $W'$  since we have discarded pairs of equal line segments. The direction in which  $W'$  traverses an edge does not affect the winding parity. (+1 and  $-1$  have the same parity.) ◀

A409  
A410

► **Definition 10.**  $W_{\text{ALG}}$  is the output of the Uncrossing Algorithm on input  $W_{\text{DP}}$ , oriented in counterclockwise direction.

A411  
A412

► **Lemma 11.**  $W_{\text{ALG}}$  is a weakly simple polygon in the free space.  $W_{\text{ALG}}$  encloses  $R$  and  $c(W_{\text{ALG}}) \leq c_{\text{DP}}$ .

A413  
A414  
A415  
A416

**Proof.** Consider a polygon  $P \in R$ . By Lemma 6(B),  $\text{wind}(W_{\text{DP}}, r_P) = 1$ . By Lemma 9,  $r_P$  has the same winding parity in  $W_{\text{ALG}}$ . Since  $W_{\text{ALG}}$  is weakly simple, every point has winding number 0 or 1. Thus  $\text{wind}(W_{\text{ALG}}, r_P) = 1$  and  $W_{\text{ALG}}$  encloses  $P$ .

Next we consider costs. The definition of the costs in Equation (1) gives

A417

$$c(W_{\text{ALG}}) = w(W_{\text{ALG}}) + \pi(W_{\text{ALG}}),$$

A418

where  $\pi(W_{\text{ALG}})$  is the sum of the penalties of objects of  $O$  enclosed by  $W_{\text{ALG}}$ .

A419 By Lemma 6(A) and the definition of  $c(W_{\text{DP}})$ ,

$$A420 \quad c_{\text{DP}} = c(W_{\text{DP}}) = w(W_{\text{DP}}) + \sum_{P \in O} \text{wind}(W_{\text{DP}}, r_P) \cdot \pi_P.$$

A421 The Uncrossing Algorithm ensures that  $w(W_{\text{ALG}}) \leq w(W_{\text{DP}})$ . It remains to compare the  
 A422 penalties. Let  $P$  be a polygon of  $O$  enclosed by  $W_{\text{ALG}}$ , i.e., with  $\text{wind}(W_{\text{ALG}}, r_P) = 1$ . By  
 A423 Lemma 9, the reference point  $r_P$  has the same winding parity in  $W_{\text{DP}}$ , and by Lemma 6(C),  
 A424  $\text{wind}(W_{\text{DP}}, r_P) \geq 0$ . Thus  $1 \leq \text{wind}(W_{\text{DP}}, r_P)$  and

$$A425 \quad \pi(W_{\text{ALG}}) = \sum_{P \in O} \text{wind}(W_{\text{ALG}}, r_P) \cdot \pi_P \leq \sum_{P \in O} \text{wind}(W_{\text{DP}}, r_P) \cdot \pi_P$$

A426 Therefore  $c(W_{\text{ALG}}) \leq c_{\text{DP}}$ . (In fact, equality holds, as shown in the next section.) ◀

## A427 **6 Correctness Proof**

A428 In defining  $W_{\text{ALG}}$ , we relied on the assumption that  $c_{\text{DP}}$  is finite. In this section we prove  
 A429 this fact, which implies that  $W_{\text{ALG}}$  exists, and we prove our main correctness result:

A430 ▶ **Theorem 12.**  $W_{\text{ALG}}$  is an optimum solution to the ENCLOSURE-WITH-PENALTIES problem.

A431 We defined the ENCLOSURE-WITH-PENALTIES problem over the continuous space of all  
 A432 weakly simple polygons, but our algorithm only explores the discrete space of weakly simple  
 A433 polygons composed of at most  $6n$  free-space edges. So we first prove that there is an optimum  
 A434 solution in this discrete space. A *feasible* solution is a weakly simple polygon that lies in the  
 A435 free space and encloses  $R$ .

A436 ▶ **Lemma 13.** For the ENCLOSURE-WITH-PENALTIES problem, there exists an optimum  
 A437 solution  $W_{\text{OPT}}$  of finite cost that consists of at most  $6n$  free-space edges.

A438 **Proof idea (Details in Appendix E).** Let  $\mathcal{S}$  be the discrete set of feasible solutions that  
 A439 consist of free-space edges each traversed at most twice. Because a planar graph on  $n$  vertices  
 A440 has at most  $3n$  edges, any solution in  $\mathcal{S}$  has at most  $6n$  free-space edges.

A441 We next prove that  $\mathcal{S}$  contains a feasible solution that encloses  $R$  and excludes  $O$ , and  
 A442 thus has finite cost. The idea is to take the cycle boundaries of polygons in  $R$  and join them  
 A443 by paths traversed twice.

A444 Since  $\mathcal{S}$  is finite and nonempty, this implies that, among the solutions in  $\mathcal{S}$ , there is a  
 A445 solution  $W^*$  of minimum cost.

A446 Finally, we prove that any feasible solution outside  $\mathcal{S}$  can be homotopically shortened  
 A447 and then uncrossed to get a solution in  $\mathcal{S}$  of no greater cost. Thus  $W^*$  is an optimum  
 A448 solution. ◀

A449 We prove that the solution  $W_{\text{OPT}}$  from Lemma 13 is one of the candidate solutions over  
 A450 which the dynamic program optimizes. As a consequence:

A451 ▶ **Lemma 14.**  $c_{\text{DP}} \leq c(W_{\text{OPT}})$ .

A452 Theorem 12 then follows: Lemmas 13 and 14 establish that  $c_{\text{DP}}$  is finite. Thus  $W_{\text{ALG}}$   
 A453 exists. By Lemma 11,  $W_{\text{ALG}}$  is a feasible solution and  $c(W_{\text{ALG}}) \leq c_{\text{DP}}$ . Combining with  
 A454 Lemma 14 yields  $c(W_{\text{OPT}}) \leq c(W_{\text{ALG}}) \leq c_{\text{DP}} \leq c(W_{\text{OPT}})$ . Thus  $W_{\text{ALG}}$  is optimal.

A455 In the remainder of the section we say a bit more about the proof of Lemma 14. By the  
 A456 definition of  $c_{\text{DP}}$ , it suffices to show that  $C(p, 6n, R) \leq c(W_{\text{OPT}})$  for a vertex  $p$  on  $W_{\text{OPT}}$ .

A457 We give an inductive proof of the more general statement that  $C(p, t, B)$  is at most the  
A458 cost of any weakly simple polygon  $W$  with at most  $t$  free-space edges that encloses  $B$  and  
A459 goes through  $p$ . Since  $W_{\text{OPT}}$  has at most  $6n$  edges, this implies Lemma 14. We make an  
A460 analogous inductive statement for  $M(pq, t, B)$ , with a suitable definition of the cost  $c(W_0)$   
A461 of an open walk  $W_0$ . It refers to transition vertices, which were defined in Section 2.1.

A462 ► **Lemma 15.** (A) *Let  $W$  be a weakly simple polygon that consists of  $\ell$  free-space edges. Let*  
A463  *$p$  be a vertex of  $W$  and let  $B$  be the objects of  $R$  that are enclosed by  $W$ . Then, for all  $t \geq \ell$ ,*  
A464  *$C(p, t, B) \leq c(W)$ .*

A465 (B) *Let  $W_0$  be an open walk with  $\ell$  free-space edges from vertex  $p$  to vertex  $q$  such that*  
A466 *the polygon  $W = W_0 + qp$  is weakly simple and  $q$  is not a transition vertex of  $W$ . Let  $B$*   
A467 *be the objects of  $R$  whose reference points lie inside  $W$  and not on  $pq$ . Then, for all  $t \geq \ell$ ,*  
A468  *$M(pq, t, B) \leq c(W_0)$ .*

A469 We prove the lemma by induction on  $\ell$  in Appendix E.

## 7 Reducing the Runtime with a Dijkstra-Style Algorithm

A470 The runtime of our algorithm to solve the ENCLOSURE-WITH-PENALTIES problem can be  
A471 reduced by a  $\Theta(n^2)$  factor, leading to the bound of Theorem 3.

A472 The dynamic programming algorithm in Section 4 is guided by a parameter  $t$ , which  
A473 limits the number of edges of the walk. We have proved (Lemma 13) that there is an optimal  
A474 solution with at most  $6n$  edges. Hence, the solution cannot be improved by allowing larger  
A475 values of  $t$ ; the iteration stabilizes, and the algorithm can stop when  $t$  reaches  $6n$ . The  
A476 parameter  $t$  has been useful for ensuring that the quantities in the dynamic programming  
A477 algorithm are well-defined, and it is essential as an induction variable for the proofs. We will  
A478 now eliminate  $t$  and solve the recursion in the style of Dijkstra’s algorithm for shortest paths.  
A479 In particular, we use a generalization of Dijkstra’s algorithm as proposed by Knuth [19].  
A480

A481 More specifically, we define  $C(p, B) := C(p, 6n, B)$  and  $M(pq, B) := M(pq, 6n, B)$  in  
A482 terms of the quantities from Section 4. By the above observations,  $C(p, B) = C(p, t, B)$  and  
A483  $M(pq, B) = M(pq, t, B)$  for all  $t \geq 6n$ . Therefore, the limit quantities  $C(p, B)$  and  $M(pq, B)$   
A484 fulfill a variation of the recursions (2–7) where the parameter  $t$  is eliminated:

$$A485 \quad C(p, \emptyset) = 0 \tag{9}$$

$$A486 \quad C(p, B) = \min\{C_1(p, B), C_2(p, B)\} \text{ for } B \neq \emptyset, \text{ where} \tag{10}$$

$$A487 \quad C_1(p, B) = \min\{w_{pq} + M(pq, B) \mid pq \text{ is a free-space edge}\} \tag{11}$$

$$A488 \quad C_2(p, B) = \min\{C(p, B') + C(p, B'') \mid B = B' \sqcup B''; B', B'' \neq \emptyset\} \tag{12}$$

$$A489 \quad M(pq, B) = \min\{M_1(pq, B), M_2(pq, B)\}, \text{ where} \tag{13}$$

$$A490 \quad M_1(pq, B) = \begin{cases} w_{pq} + C(q, B), & \text{if } pq \text{ is a free-space edge} \\ \infty, & \text{otherwise} \end{cases} \tag{14}$$

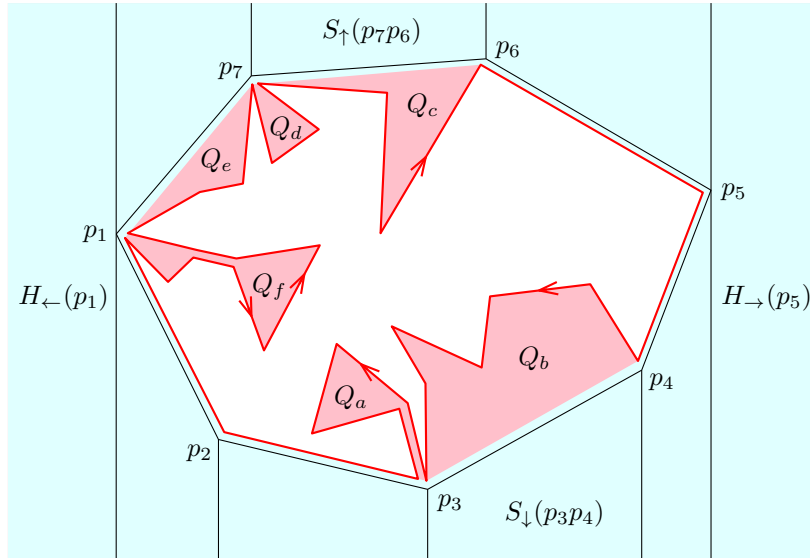
$$A491 \quad M_2(pq, B) = \min\{M(pr, B') + M(rq, B'') + \pi(\Delta) \mid \tag{15}$$

$$A492 \quad \quad \quad prq = \Delta \text{ is a counterclockwise triangle;}$$

$$A493 \quad \quad \quad B = B' \sqcup B'' \sqcup \{P \in R \mid r_P \in \Delta\} \}$$

A494 As in Equation (8), we define the solution to the whole problem as

$$A495 \quad c_{\text{DP}} := \min\{C(p, R) \mid p \text{ is a vertex}\}. \tag{16}$$



■ **Figure 6** Partition of the outside into pockets  $Q_a, \dots, Q_f$ , two half-planes, and seven planks.

A496 By Lemma 13 and by the definition of  $C(p, B)$  and  $M(pq, B)$ , the value of  $c_{\text{DP}}$  resulting  
A497 from (16) is the same as the one resulting from (8).

A498 The system (9–15) is a *system of equations* that involves cyclic dependencies. Nevertheless,  
A499 we can show that it has a unique solution (Lemma 22 in Appendix F.1). The reason is  
A500 that on the right-hand side of the equations, the result of any expression combining some  
A501 quantities of the form  $C(p, B)$  and  $M(pq, B)$  is always *larger* than these quantities.

A502 Similar to Dijkstra’s shortest-path algorithm, our algorithm maintains tentative values  
A503  $C(p, B)$  and  $M(pq, B)$ . The smallest of the tentative values is made permanent, and all  
A504 right-hand side expressions where this value appears are evaluated and used to update the  
A505 corresponding tentative left-hand side values. The algorithm that carries out this idea is  
A506 shown in Appendix F.2 (Algorithm 1).

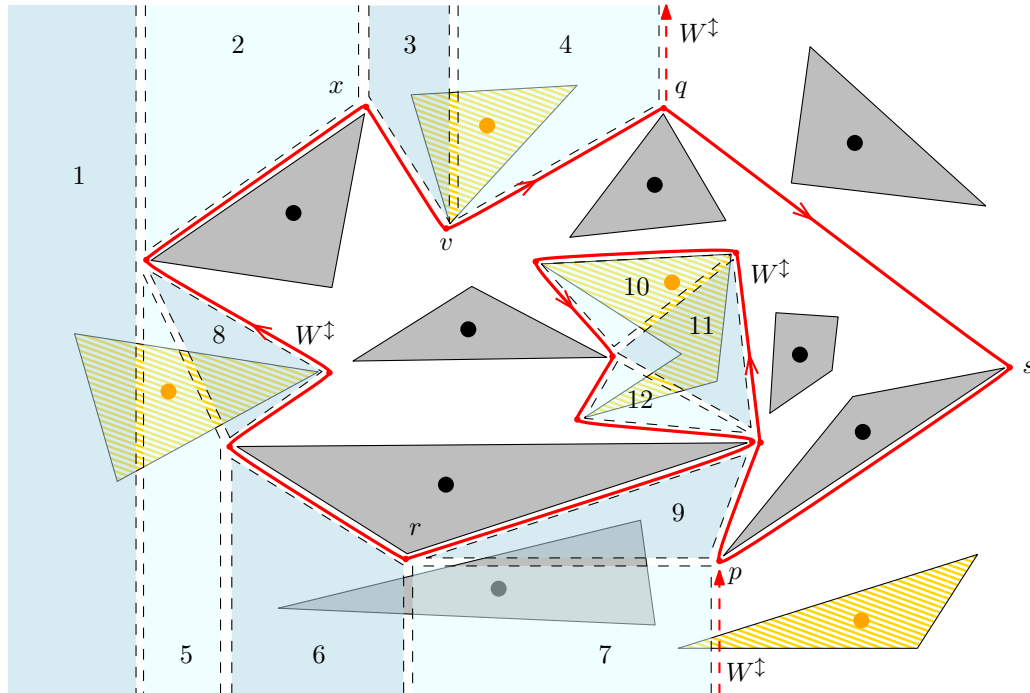
A507 The most numerous quantities are the  $O(2^k n^2)$  values  $M(pq, B)$ , and hence the space  
A508 complexity is  $O(2^k n^2)$ . Compared to the running time for the dynamic programming  
A509 algorithm in Section 4, we save a factor  $n^2$ : The elimination of  $t$  reduces the number of  
A510 recursions by a factor  $\Theta(n)$ , and we save another factor  $\Theta(n)$  because we need not go through  
A511 all decompositions  $t = t_1 + t_2$  on the right-hand side. The analysis of the full algorithm is  
A512 given in Appendix F.3. The total running time is  $O(3^k n^3)$ , as claimed in Theorem 3.

## A513 8 The Inverted Problem

A514 In the inverted problem, the required objects have to be outside the weakly simple polygon  $W$ ,  
A515 and the penalties of the polygons that are outside  $W$  are counted (see the introduction).  
A516 The approach for the original problem has to be adapted accordingly. To derive a suitable  
A517 dynamic programming formulation, we decompose the *outside* of  $W$  into elementary pieces,  
A518 as shown in Figure 6: We form the convex hull of  $W$  and extend vertical rays upward and  
A519 downward from the convex hull vertices. This leads to two additional types of regions:

- A520 ■ one left half-plane and one right half-plane, each bounded by a vertical line through an  
A521 object vertex;

A522 ■ vertical *planks*, that is, regions bounded by a line segment and two vertical upward rays  
 A523 or two vertical downward rays. We discuss such regions in Appendix F.4.



■ **Figure 7** A weakly simple polygon  $W$  (red/bold) that has the four required objects (yellow/hatched) on the outside. The penalties of two optional (grey) objects is added to the length when the cost is computed. We grow the outer region triangle by triangle, considering also “triangles” that extend to  $-\infty$  or  $+\infty$  (vertical planks), or both, like the left half-plane (number 1). The union of the shaded blue regions is bounded by vertical rays from  $p$  downward and from  $q$  upward plus a weakly simple walk between  $p$  and  $q$ . The extended walk  $W^\dagger$  is a candidate solution considered for the subproblem  $U(p, q, B, t)$ , when  $t \geq 13$  and  $B$  consists of the three leftmost required objects. (The fourth required object has its reference point (thick dot) outside the region.) Two more planks, spanned by  $ps$  and  $qs$ , plus the right half-plane through  $s$ , would complete the outside region of  $W$ .

A524 We stick to the convention that the interior of the region of interest lies on the left side of  $W$ .  
 A525 Accordingly, the solution polygon  $W$  is now oriented clockwise.

A526 In the algorithm, we build the region of interest outside-in, see Figure 7, starting from  
 A527 a left half-plane bounded by two vertical rays through an object vertex. We add planks  
 A528 from left to right along common rays, and, as in Section 4, we may also attach triangles  
 A529 along common edges and digons along common vertices. In addition to the usual bounded  
 A530 walks, we now also consider polygonal walks  $W^\dagger$  that start from the endpoint of a vertical  
 A531 downward ray and end at a vertical upward ray. More precisely, for each pair of vertices  $p, q$ ,  
 A532 we consider a subproblem of type  $U$  (“unbounded”), which considers regions bounded by a  
 A533 vertical ray down from  $p$ , a walk  $W$  from  $p$  to  $q$ , and a vertical ray up from  $q$ , see Figure 7  
 A534 for an example;  $p = q$  is allowed. Accordingly, the algorithm computes quantities  $U(p, q, t, B)$   
 A535 for all  $B \subseteq R$  and  $t \leq 6n$ . The two unbounded rays jointly play the role of the mouth.

A536 Appendix G describes how the dynamic programming recursion and the correctness proof  
 A537 of Section 6 for the non-inverted problem is adapted for the inverted problem.

## 9 Applications, Extensions, and Open Problems

A538

**Splitting a surface.** Our result may shed some light on the following open problem by Bulavka, Colin de Verdière, and Fuladi [6, Conclusion]: given an orientable combinatorial surface of genus  $g$ , and an integer  $g'$ ,  $1 \leq g' < g$ , is it FPT in  $g'$  to compute a shortest weakly simple closed curve that cuts off a surface of genus  $g'$ ? The problem is FPT in  $g$  [7, Theorem 6.1]. Our algorithm for GRAPH-ENCLOSURE-WITH-PENALTIES shows that the answer is yes when restricting to some (admittedly very special) instances, see Appendix H, and thus provides some hope for a positive answer in general, although this remains open.

**Curved objects and line segments.** We believe that our approach carries over to more general objects. Curved objects that are sufficiently well behaved can be treated by considering all bitangents as free-space edges. We can already handle point objects, as described in Appendix I; line segments without other vertices in their interior should also be doable. However, extending to weakly simple polygon objects seems difficult. Even for a path object consisting of two line segments joined at point  $p$  it is a challenge to prevent a solution from cutting through the path at  $p$ .

**Recognizing weakly simple self-overlapping polygons.** As mentioned, our dynamic program optimizes over the class of weakly simple self-overlapping polygons, see Appendix L. Weakly simple polygons can be recognized in  $O(n \log n)$  time [2], and self-overlapping polygons in time  $O(n^3)$  [21]. Can weakly simple self-overlapping polygons be recognized efficiently?

A557

---

### References

- A558 1 M. Abrahamsen, P. Giannopoulos, M. Löffler, and G. Rote. Geometric multicut: shortest  
A559 fences for separating groups of objects in the plane. *Discrete Comput. Geom.*, 64:575–607,  
A560 2020. doi:10.1007/s00454-020-00232-w.
- A561 2 H. A. Akitaya, G. Aloupis, J. Erickson, and C. D. Tóth. Recognizing weakly simple polygons.  
A562 *Discrete & Computational Geometry*, 58(4):785–821, 2017. doi:10.1007/S00454-017-9918-3.
- A563 3 H. A. Akitaya and C. D. Tóth. Reconstruction of weakly simple polygons from their edges.  
A564 *International Journal of Computational Geometry & Applications*, 28(02):161–180, 2018.  
A565 doi:10.1142/S021819591860004X.
- A566 4 E. M. Arkin, S. Khuller, and J. S. Mitchell. Geometric knapsack problems. *Algorithmica*,  
A567 10(5):399–427, 1993. doi:10.1007/BF01769706.
- A568 5 T. Biedl. Small drawings of outerplanar graphs, series-parallel graphs, and other planar graphs.  
A569 *Discret. Comput. Geom.*, 45(1):141–160, 2011. doi:10.1007/S00454-010-9310-Z.
- A570 6 D. Bulavka, É. Colin de Verdière, and N. Fuladi. Computing shortest closed curves on non-  
A571 orientable surfaces. In W. Mulzer and J. M. Phillips, editors, *40th International Symposium*  
A572 *on Computational Geometry (SoCG 2024)*, volume 293 of *Leibniz International Proceedings in*  
A573 *Informatics (LIPIcs)*, pages 28:1–28:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik,  
A574 2024. arXiv:2403.11749, doi:10.4230/LIPIcs.SoCG.2024.28.
- A575 7 E. W. Chambers, É. Colin de Verdière, J. Erickson, F. Lazarus, and K. Whittlesey. Splitting  
A576 (complicated) surfaces is hard. *Comput. Geom.*, 41(1–2):94–110, 2008. doi:10.1016/j.comgeo.  
A577 2007.10.010.
- A578 8 E. W. Chambers, J. Erickson, K. Fox, and A. Nayyeri. Minimum cuts in surface graphs. *SIAM*  
A579 *J. Comput.*, 52(1):156–195, 2023. doi:https://doi.org/10.1137/19M1291820.
- A580 9 T. M. Chan, Q. He, and J. Xue. Enclosing points with geometric objects. In W. Mulzer and  
A581 J. M. Phillips, editors, *40th International Symposium on Computational Geometry (SoCG*

- A582 2024), volume 293 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–  
 A583 35:15. Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. [arXiv:](#)  
 A584 2402.17322, doi:10.4230/LIPIcs.SoCG.2024.35.
- A585 10 H.-C. Chang, J. Erickson, and C. Xu. Detecting weakly simple polygons. In *Proceedings of*  
 A586 *the 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1655–1670,  
 A587 2015. [arXiv:1407.3340](#), doi:10.1137/1.9781611973730.110.
- A588 11 T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT  
 A589 Press, 3rd edition, 2009.
- A590 12 G. Di Battista and R. Tamassia. Algorithms for plane representations of acyclic digraphs.  
 A591 *Theor. Comput. Sci.*, 61:175–198, 1988. doi:10.1016/0304-3975(88)90123-5.
- A592 13 P. Eades and D. Rappaport. The complexity of computing minimum separating polygons.  
 A593 *Patt. Recog. Lett.*, 14(9):715–718, 1993. doi:10.1016/0167-8655(93)90140-9.
- A594 14 D. Eppstein, M. Overmars, G. Rote, and G. Woeginger. Finding minimum area  $k$ -gons.  
 A595 *Discrete Comp. Geom.*, 7:45–58, 1992. doi:10.1007/BF02187823.
- A596 15 P. Evans and C. Wenk. Combinatorial properties of self-overlapping curves and inter-  
 A597 rior boundaries. *Discrete & Computational Geometry*, 69(1):91–122, 2023. doi:10.1007/  
 A598 s00454-022-00416-6.
- A599 16 M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network  
 A600 optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.
- A601 17 S. K. Ghosh and D. M. Mount. An output-sensitive algorithm for computing visibility graphs.  
 A602 *SIAM Journal on Computing*, 20(5):888–910, 1991. doi:10.1137/0220055.
- A603 18 K. Klost, M. van Kreveld, D. Perz, G. Rote, and J. Tkadlec. Minimum spanning blob-  
 A604 trees. In J. Kratochvíl and G. Liotta, editors, *Abstracts of the 41st European Workshop on*  
 A605 *Computational Geometry (EuroCG 2025), April 9–11, Liblice, Czech Republic*, pages 23:1–  
 A606 23:8, 2025. URL: <https://kam.mff.cuni.cz/conferences/eurocg2025/booklet2025.pdf>,  
 A607 [arXiv:2503.02439](#).
- A608 19 D. E. Knuth. A generalization of Dijkstra’s algorithm. *Information Processing Letters*, 6(1):1–5,  
 A609 1977. doi:10.1016/0020-0190(77)90002-3.
- A610 20 D. Marx, M. Pilipczuk, and M. Pilipczuk. On subexponential parameterized algorithms for  
 A611 Steiner tree and directed subset TSP on planar graphs. In *Proc. 59th Ann. IEEE Symp.*  
 A612 *Foundat. Comput. Sci. (FOCS)*, pages 474–484, 2018. doi:10.1109/FOCS.2018.00052.
- A613 21 P. W. Shor and C. J. Van Wyk. Detecting and decomposing self-overlapping curves. *Comput.*  
 A614 *Geom.: Theory and Applications*, 2(1):31–50, 1992. doi:10.1016/0925-7721(92)90019-0.
- A615 22 D. Singmaster and J. W. Grossman. Solution to problem E2897: An Eulerian circuit with  
 A616 no crossings. *The American Mathematical Monthly*, 90(4):287–288, 1983. URL: [http://www.](http://www.jstor.org/stable/2975767)  
 A617 [jstor.org/stable/2975767](http://www.jstor.org/stable/2975767), doi:10.2307/2975767.
- A618 23 R. Tamassia and I. G. Tollis. A unified approach a visibility representation of planar graphs.  
 A619 *Discret. Comput. Geom.*, 1:321–341, 1986. doi:10.1007/BF02187705.
- A620 24 M.-T. Tsai and D. B. West. A new proof of 3-colorability of Eulerian triangulations. *Ars*  
 A621 *Mathematica Contemporanea*, 4:73–77, 2011. doi:10.26493/1855-3974.193.8e7.
- A622 25 J. H. van Lint and R. M. Wilson. *A course in combinatorics*. Cambridge University Press,  
 A623 second edition, 2001.

A624

## **A** Details for Section 2: Weakly Simple Polygons or Walks

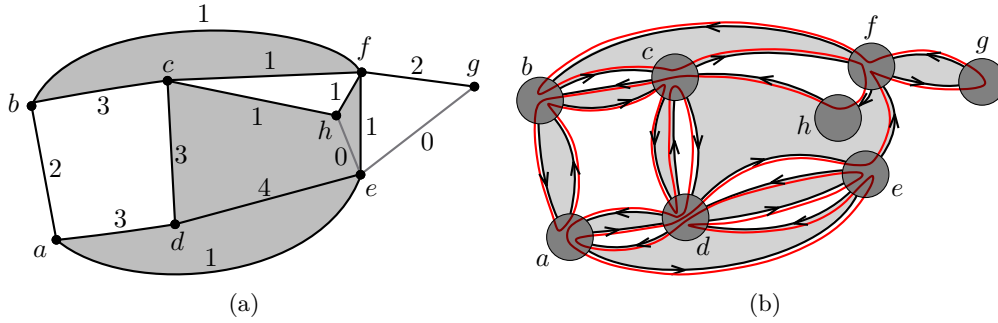
A625

We characterize weakly simple polygons/walks in terms of non-crossing Euler tours.

A626

A connected plane graph or multigraph is specified via its combinatorial map (or rotation system) that specifies the counterclockwise cyclic order of edges around each vertex. If there

A627



■ **Figure 8** (a) A simple plane graph  $G$ . The edges of a closed walk  $W = abcdcdaededadedefgfbfchcba$  are drawn in black and labeled with their multiplicities. (b) A non-crossing Euler tour in an expansion  $M(G, W)$  of  $W$ . Vertices of  $M(G, W)$  are represented as large disks. The Euler tour is shown in red and certifies that the walk  $W$  from (a) is weakly simple.

A628 are parallel edges, they have distinct identities and must be explicitly ordered in the rotation  
 A629 system. One face is designated as the outer face.

A630 A **non-crossing Euler tour** of a plane multigraph is a closed walk that traverses each  
 A631 edge exactly once and has no **vertex crossing**. A **vertex crossing** occurs when the tour  
 A632 visits some vertex  $v$  twice, entering once on edge  $e$  and leaving on edge  $f$ , and entering again  
 A633 on edge  $g$  and leaving on edge  $h$ , such that  $e, f$  and  $g, h$  interleave in the cyclic ordering of  
 A634 edges around  $v$ , i.e., they appear in the order  $e, g, f, h$  or  $e, h, f, g$ .

A635 Let  $G$  be a plane multigraph with a non-crossing Euler tour  $T$ . Then the vertices of  $G$   
 A636 have even degrees, so the faces of  $G$  can be 2-colored such that the two faces incident to an  
 A637 edge have different colors [25, Theorems 34.2 and 34.4]. Suppose the two colors are grey and  
 A638 white with the outer face colored white. We will traverse  $T$  so that a grey face lies to the  
 A639 left of the first edge of the tour. Then, because the tour is non-crossing, every edge of the  
 A640 tour has a grey face to the left. We call this a **counterclockwise traversal** of  $T$ , and we  
 A641 define the **interior** faces of  $T$  to be the grey faces. Note that the interior faces determine a  
 A642 partition of the edges of  $G$ .

A643 **Weakly simple walks in plane graphs.** A walk  $W$  of length  $n$  in a simple plane graph  $G$   
 A644 is a sequence  $(v_0, v_1, \dots, v_n)$  of vertices, such that each  $v_i v_{i+1}$  is an edge of the graph. A  
 A645 vertex/edge of  $G$  may appear multiple times in the sequence. If  $v_0 = v_n$  this is a **closed**  
 A646 **walk**; otherwise it is an **open walk**.

A647 Intuitively, a closed walk  $W$  is weakly simple if multiple traversals of an edge of  $G$  can be  
 A648 resolved to avoid vertex crossings. We make this more formal by way of a non-crossing Euler  
 A649 tour that provides a certificate that  $W$  is weakly simple.

A650 For edge  $e$  of  $G$ , define the **multiplicity**  $m(e)$  of  $e$  in  $W$ , to be the number of times  $W$   
 A651 traverses  $e$  (in either direction). An **expansion** of  $W$  is a plane multigraph  $M(W, G)$  that  
 A652 replaces each edge  $e = ab$  of  $G$  by a **bundle** of  $m(e)$  parallel edges, each identified with a  
 A653 unique edge of  $W$ , and replaces  $e$  in the rotation systems of  $a$  and  $b$  by an ordered sequence  
 A654 of the edges in the bundle. Then  $W$  corresponds to an Euler tour in  $M(W, G)$ .

A655 A closed walk  $W$  in a plane graph  $G$  is **weakly simple** if it has an expansion  $M(W, G)$  in  
 A656 which  $W$  corresponds to a non-crossing Euler tour. We call such an  $M(W, G)$  a **certificate**  
 A657 that  $W$  is weakly simple. Note that certificates are not unique; in particular they can have  
 A658 different rotation systems. For example, see Figure 3(c) when  $G$  is a single edge and  $W$   
 A659 traverses it four times.

A660 Let  $M(W, G)$  be a certificate that  $W$  is weakly simple. Some faces of  $M(W, G)$  are digons  
A661 between parallel edges. Each remaining face is a union of faces of  $G$ . (If an edge of  $G$   
A662 has multiplicity 0, then the incident faces are merged in  $M(W, G)$ .)  $W$  corresponds to a  
A663 non-crossing Euler tour of  $M(W, G)$ , which determines the interior faces of  $M(W, G)$ . We  
A664 say that a face of  $G$  is *interior* to  $W$  if it corresponds to an interior face of  $M(W, G)$ . See  
A665 Figure 8. (Observe that the interior faces of  $G$  are well-defined independent of choice of  
A666 certificate because, in a 2-coloring of the faces of  $M(W, G)$ , the color of a face of  $G$  does not  
A667 depend on the choice of rotation system for  $M(W, G)$ —the two faces incident to edge  $e$  of  $G$   
A668 have the same color if  $m(e)$  is even, and opposite colors if  $m(e)$  is odd.)

A669 **Weakly simple polygons.** A polygon  $P$  is a sequence  $(p_0, p_1, \dots, p_{n-1})$  of points (the  
A670 *vertices* of  $P$ ) together with the line segments  $p_i p_{i+1 \bmod n}$  (the *edges* of  $P$ ). We do not  
A671 allow edges of length 0. As a degenerate case, we allow a polygon with a single vertex and  
A672 no edges. A polygon is *simple* if the vertices are distinct points and no two edges intersect  
A673 except that consecutive edges intersect at their common vertex.

A674 For a general non-simple polygon, a point of the plane may correspond to multiple polygon  
A675 vertices, and polygon edges may overlap or cross. Recall that an *interior crossing* of  $P$  is  
A676 a point that is in the relative interiors of two (or more) edges that are not collinear, and  
A677 a *fork* is a vertex that lies in the relative interior of an edge. Both interior crossings and  
A678 forks can be eliminated by subdividing edges, albeit possibly with a quadratic blow-up in  
A679 the number of vertices of the polygon.

A680 The standard definition [2, 10] is that a polygon is *weakly simple* if it has fewer than  
A681 three vertices, or it has at least three vertices and for any  $\varepsilon > 0$ , the vertices can be perturbed  
A682 by at most  $\varepsilon$  to yield a simple polygon. The intuition is that a weakly simple polygon is one  
A683 without crossings, but it is tricky to define crossings, since they need not be local, see the  
A684 discussion by Chang, Erickson, and Xu [10].

A685 In our proofs we find it useful to characterize weakly simple polygons in terms of the  
A686 purely combinatorial notion of non-crossing Euler tours in an associated multigraph. Let  $P$   
A687 be a polygon without interior crossings. Expanding on definitions from [2, 10], we define the  
A688 *image graph* of  $P$  to be a plane straight-line graph  $G$  formed as follows. First subdivide  
A689 edges of  $P$  at forks. Next replace every set of coincident vertices of  $P$  by a single vertex of  
A690  $G$ , and replace every set of equal line segments of  $P$  by a single edge of  $G$  (these are called  
A691 “segments” in [2, 10]). Then  $P$  corresponds to a closed walk  $W_P$  in the plane graph  $G$  and we  
A692 can apply the concept of a certificate for a weakly simple walk from above. In this context  
A693 we call an expansion  $M(W_P, G)$  an *image multigraph* of  $P$ . We use the notation  $M(P)$   
A694 for an image multigraph of  $P$ , since it depends only on  $P$ .

A695 ► **Lemma 16.** *A polygon  $P$  is weakly simple if and only if it has no interior crossings and it*  
A696 *has an image multigraph in which  $P$  corresponds to a non-crossing Euler tour.*

A697 An image multigraph in which  $P$  corresponds to a non-crossing Euler tour is called a  
A698 *certificate* that  $P$  is weakly simple. Again, note that a certificate is in general not unique.

A699 Before turning to the proof of the lemma, we discuss equivalent notions of the interior of  
A700 a weakly simple polygon. The definition of the interior of a non-crossing Euler tour gives  
A701 one definition of the interior of a weakly simple polygon. This is equivalent to the definition  
A702 of interior in terms of winding numbers. As seen in Figure 3, the edges of a weakly simple  
A703 polygon can be partitioned into boundary walks of the interior faces.

A704 The  $O(n \log n)$  time algorithm to recognize weakly simple polygons by Akitaya, Aloupis,  
A705 Erickson, and Toth [2] implicitly proves Lemma 16 (as does the earlier algorithm by Chang,

A706 Erickson, and Xu [10]). Akitaya et al. use *strip systems* as their certificates of weak simplicity.  
A707 A strip system is more geometric in nature, but has the advantage of being linear size, which  
A708 is important for their fast algorithm. Our image multigraphs have quadratic size, but more  
A709 immediately give the properties we need.

A710 We give a direct proof of Lemma 16 that depends only on the characterization of a weakly  
A711 simple polygon as a limit of simple curves as Fréchet distance goes to zero [10, Theorem 2.1],  
A712 which allows adding new vertices to the polygon.

A713 **Proof.** Suppose  $P$  is weakly simple. By definition,  $P$  has no interior crossings. Let  $P'$  be the  
A714 result of subdividing  $P$  at forks. By definition, for any  $\varepsilon > 0$  there is a simple  $\varepsilon$ -approximation  
A715 of  $P$ , and this determines a simple  $\varepsilon$ -approximation of  $P'$ , call it  $P'_\varepsilon$ . Any set  $U$  of coincident  
A716 vertices of  $P'$  lies in a disc  $D$  of radius  $\varepsilon$  in  $P'_\varepsilon$ , and the edges incident to  $U$  leave  $D$  at  
A717 distinct points. From  $P'_\varepsilon$  we construct a plane multigraph  $M$  by contracting each set  $U$  to a  
A718 single vertex, and ordering the incident edges according to their order around  $D$ . Then  $M$  is  
A719 a plane Eulerian multigraph that expands the image graph of  $P$ , and  $P$  corresponds to a  
A720 non-crossing Euler tour of  $M$ .

A721 For the other direction, suppose  $P$  has no interior crossings and suppose  $P$  has an image  
A722 multigraph  $M(P)$  in which  $P$  corresponds to a non-crossing Euler tour  $W$ . We use the result  
A723 that a polygon is weakly simple if it is a limit of simple polygons (possibly with more vertices)  
A724 as Fréchet distance goes to zero [10, Theorem 2.1]. For  $\varepsilon$  small enough, we construct a simple  
A725 polygon  $P_\varepsilon$  within Fréchet distance  $\varepsilon$  of  $P$ . Polygon  $P_\varepsilon$  will have more vertices than  $P$ . In  
A726 particular, our construction will subdivide edges of  $P$  at forks, and then replace each vertex  
A727 by two vertices and add vertices in the middle of edges. The coordinates of  $P$ 's vertices  
A728 determine a straight-line drawing of  $P$ 's image graph  $G$  in the plane. We expand this to a  
A729 1-bend drawing of  $M(P)$  in which the edges in each bundle are spread apart. In more detail,  
A730 each edge  $e$  of  $G$  corresponds to a bundle of  $m(e)$  edges in  $M(P)$ . We add a vertex in the  
A731 middle of every edge of the bundle and space these vertices along a small line segment drawn  
A732 perpendicular to  $e$  at its midpoint using the ordering of the edges in the rotation system of  
A733  $M(P)$ .

A734 We complete the construction of  $P_\varepsilon$  by altering the drawing of  $M(P)$  to spread apart  
A735 the coincident vertices of  $P'$ . For each vertex  $v$  of  $M(P)$ , construct a small disc  $D$  of radius  
A736  $\varepsilon$  centered at  $v$  in the drawing. The edges that enter  $D$  are incident to  $v$ , and they cross  
A737 the boundary of  $D$  in rotation system ordering. Let  $D_e$  be the point where edge  $e$  enters  
A738 disc  $D$ . Suppose the Eulerian tour  $W$  visits  $v$ , entering on edge  $e$  and leaving on edge  $f$ . In  
A739 the drawing and in  $W$  replace segments  $D_e v$  and  $v D_f$  by the chord  $D_e D_f$ . If two of these  
A740 chords of  $D$  cross, they would correspond to a vertex crossing in  $W$ . Thus the result is a  
A741 simple polygon  $P_\varepsilon$ . ◀

## A742 **B** Details for Section 3: Common Framework

A743 **Proof of Theorem 2, assuming Theorem 3.** Consider an instance of GRAPH-ENCLOSURE-  
A744 WITH-PENALTIES, with simple connected plane graph  $G$ , required faces  $R$ , and optional  
A745 faces  $O$ . We reduce to an instance of ENCLOSURE-WITH-PENALTIES.

A746 Find a straight-line plane embedding  $G'$  of  $G$  with the same combinatorial map (i.e.,  
A747 preserving the rotation system). The faces of  $G'$ , including the outer face, become the  
A748 polygons for our new instance. Observe that all these polygons are almost-simple. For the  
A749 bounded faces of  $G'$  we preserve the partition into  $R$  and  $O$  and the penalties. The outer  
A750 face of  $G'$  becomes an unbounded polygon. We put it in the set  $O$  with a penalty of 0 (the  
A751 penalty is irrelevant, since no weakly simple polygon  $W$  can contain the unbounded polygon).

A752 The free space of this set of polygons has no interior; it consists only of the edges of  $G'$ .  
 A753 Each such edge lies between two faces (polygons) so it is a squeezed edge and we assign it  
 A754 the weight of the corresponding edge of  $G$ .

A755 This completes the reduction. For a graph on  $n$  vertices, the reduction produces a set  
 A756 of polygons with  $O(n)$  vertices. The number  $k$  of required objects remains the same. The  
 A757 reduction takes  $O(n)$  time. The runtime claim in Theorem 2 follows.

A758 There is a one-to-one correspondence between weakly simple polygons in the free space and  
 A759 weakly simple closed walks in  $G$  (as defined in Appendix A), and the interior and the cost are  
 A760 preserved. Therefore a solution to the resulting instance of the ENCLOSURE-WITH-PENALTIES  
 A761 problem provides a solution to the original graph problem. ◀

## A762 **C** Details for Section 4: Dynamic Programming Algorithm

### A763 **C.1** Runtime of the dynamic programming algorithm

A764 The number of subproblems of type  $M$  is  $O(2^k n^3)$ : there are  $O(n^2)$  choices for the mouth  $pq$ ,  
 A765  $2^k$  choices for the set  $B$ , and  $O(n)$  choices for the parameter  $t$ . The number of subproblems  
 A766 of type  $C$  is only  $O(2^k n^2)$ , by the same analysis. Thus, the space requirement is  $O(2^k n^3)$ .

A767 The recursion that dominates the runtime is (7) for  $M_2$ . For fixed parameters  $pq$ ,  $t$ ,  
 A768 and  $B$ , there are at most  $n$  choices for the point  $r$ , at most  $t = O(n)$  possibilities for  $t_1$  and  
 A769  $t_2$ , and at most  $2^{|B|}$  choices for  $B_1$  and  $B_2$ . We run through the possible choices of  $r$  in an  
 A770 outer loop. Then, for each triangle  $\Delta = prq$ , we can determine the reference points that lie  
 A771 in  $\Delta$  in a straightforward way in  $O(n)$  time, and this runtime will be dominated by the inner  
 A772 loop. This leads to an overall runtime of

$$A773 \quad O(n^3) \times \sum_{B \subseteq R} n \times (O(n) + O(n) \times 2^{|B|}) = O(n^5) \sum_{B \subseteq R} 2^{|B|} = O(n^5) \sum_{i=0}^k \binom{k}{i} 2^i = O(3^k n^5).$$

A774 We assume that we can access each of the  $O(2^k n^3)$  entries of the dynamic programming  
 A775 table in constant time. In particular, a memory word contains at least  $k$  bits, and hence set  
 A776 operations on subsets of  $R$  take constant time.

A777 The above computation assumes that we can determine in  $O(1)$  time, given two input  
 A778 vertices  $p$  and  $q$ , whether  $pq$  is a free-space edge. For this purpose, we precompute a Boolean  
 A779 array of size  $n \times n$ , with rows and columns indexed by the vertices, storing this information.  
 A780 The array can be determined in  $O(n^2)$  time from the visibility graph of the input polygons,  
 A781 which can be computed in  $O(n \log n + e) = O(n^2)$  time for a visibility graph with  $e$  edges [17].

### A782 **C.2** Details for Section 4.2: Extracting the solution

A783 **Defining  $W_{\text{DP}}$ .** With each finite value  $C(p, t, B)$  that is computed in the recursions (2)–(5),  
 A784 we can naturally associate a polygon  $W = W(p, t, B)$ , a closed walk of free-space edges that  
 A785 goes through  $p$ . Similarly, with each finite value  $M(pq, t, B)$  computed in the recursions  
 A786 (6)–(7), we can associate an open walk of free-space edges  $W = W(pq, t, B)$  that goes from  $p$   
 A787 to  $q$ . For example, in (7), where we form the sum  $M(pr, t_1, B_1) + M(rq, t_2, B_2)$ , the open  
 A788 walk  $W$  is obtained by concatenating the open walks associated with  $M(pr, t_1, B_1)$  and  
 A789  $M(rq, t_2, B_2)$ .

A790 ▶ **Definition 17.** For an open walk  $W$  from  $p$  to  $q$ , we define its closure  $\overline{W}$  to be the polygon  
 A791  $W + qp$ . For a closed walk  $W$ , we define  $\overline{W}$  to be the polygon  $W$  itself.

A792 By remembering for each recursion the values  $t_1, t_2, B_1, B_2$ , etc. from which the minimum  
A793 was obtained, we can recursively reconstruct the associated open/closed walks  $W$  and the  
A794 polygons  $\overline{W}$  in  $O(t)$  time.

A795 This formalizes the definition of  $W_{\text{DP}}$  (Definition 4).

A796 **Proving Lemma 6 about the Properties of  $W_{\text{DP}}$ .** We must extend the definition of cost  
A797 to open walks:

A798 ► **Definition 18.** *For an open or closed polygon  $W$  in the free space,*

$$A799 \quad c(W) := w(W) + \sum_{P \in O} \text{wind}(\overline{W}, r_P) \cdot \pi_P. \quad (17)$$

A800 *In particular, if  $W$  is an open walk, we take winding numbers with respect to its closure  $\overline{W}$ .*

A801 This definition agrees with the previous Definition 5 when  $W$  is closed. For a reference point  
A802  $r_P$  lying on the mouth  $qp$  of an open walk  $W$  from  $p$  to  $q$ , we compute  $\text{wind}(\overline{W}, r_P)$  as if  $r_P$   
A803 were slightly moved to the right of the segment  $qp$ , i.e., in the direction where the outside  
A804 would normally be in case of a counterclockwise simple polygon. In case of a weakly simple  
A805 polygon  $\overline{W}$ , this means that points on the mouth are not considered to be enclosed.

A806 To prove Lemma 6, we need results on winding numbers as walks are glued together. We  
A807 first define gluing more precisely. Two closed walks  $W_1$  and  $W_2$  can be glued together at  
A808 a common vertex, or along a common edge that is traversed in opposite directions by  $W_1$   
A809 and  $W_2$ .

A810 More formally: If  $W_1 = (p, q_1, q_2, \dots, q_n)$  and  $W_2 = (p, p_1, p_2, \dots, p_m)$ , then the result of  
A811 gluing the walks along the common point  $p$  is  $P = (p, q_1, q_2, \dots, q_n, p, p_1, p_2, \dots, p_m)$ .

A812 If  $P_1 = (p, q, q_2, \dots, q_n)$  and  $P_2 = (q, p, p_2, \dots, p_m)$  both use the edge  $pq$ , but in opposite  
A813 directions, then  $P = (q, q_2, \dots, q_n, p, p_2, \dots, p_m)$  is the result.

A814 We have the following easy but key property:

A815 ► **Lemma 19 (Additivity of Winding Numbers).** *The winding number is additive with respect  
A816 to the gluing operation: If  $P$  is a closed walk obtained by gluing two closed walks  $P_1$  and  $P_2$   
A817 along a common edge or vertex, then*

$$A818 \quad \text{wind}(P, x) = \text{wind}(P_1, x) + \text{wind}(P_2, x)$$

A819 *for all points  $x$  that do not lie on  $P_1$  or  $P_2$ .*

A820 **Proof.** Let  $\rho$  be any ray from  $x$  to the unbounded face that avoids the vertices of  $P$  and  
A821 intersects the edges of  $P_1$  and  $P_2$  transversally.

A822 Assume first that  $P$  results from gluing  $P_1$  and  $P_2$  at a common vertex; then the multiset  
A823 of the directed edges of  $P$  is exactly the union of the directed edges of  $P_1$  and of  $P_2$  (counting  
A824 multiplicities). Let  $r^+$ ,  $r_1^+$ , and  $r_2^+$  be the number of times an edge of  $P$ ,  $P_1$ , and  $P_2$ ,  
A825 respectively, crosses  $\rho$  from right to left; we have  $r^+ = r_1^+ + r_2^+$ . Similarly, with the analogous  
A826 notations  $r^-$ ,  $r_1^-$ , and  $r_2^-$  counting the number of crossings from left to right, we have  
A827  $r^- = r_1^- + r_2^-$ . Summing up, we obtain the result.

A828 If  $P$  results from gluing  $P_1$  and  $P_2$  at a common edge  $pq$ , the effects of the two oppositely  
A829 oriented edges  $pq$  and  $qp$  cancel out when the winding number is computed. (They contribute  
A830 to  $r_1^+$  and  $r_2^-$ , or to  $r_1^-$  and  $r_2^+$ , or not at all.) The proof for the first case carries over. ◀

A831 We now restate and prove Lemma 6.

A832 ► **Lemma 6.**

- A833 (A)  $c_{\text{DP}} = c(W_{\text{DP}})$ ;  
A834 (B) for all  $P \in R$ ,  $\text{wind}(W_{\text{DP}}, r_P) = 1$ ;  
A835 (C) for all points  $x$  that do not lie on  $W_{\text{DP}}$ ,  $\text{wind}(W_{\text{DP}}, x) \geq 0$ .

A836 **Proof.** We prove by induction that the properties hold more generally for all subproblems  
A837 solved in the dynamic programming algorithm. To be precise, consider a finite value  $C(p, t, B)$   
A838 or  $M(pq, t, B)$  computed in the recursions (2)–(7). Let  $W = W(p, t, B)$  or  $W = W(pq, t, B)$   
A839 be the closed or open walk associated with the solution and let  $\overline{W}$  be its closure (Definition 17).  
A840 We prove by induction on  $t$  that:

- A841 (i)  $C(p, t, B) = c(W(p, t, B))$ ,  $M(pq, t, B) = c(W(pq, t, B))$ ;  
A842 (ii) for all  $P \in B$ ,  $\text{wind}(\overline{W}, r_P) = 1$  and for all  $P \in R \setminus B$ ,  $\text{wind}(\overline{W}, r_P) = 0$ ;  
A843 (iii) for all points  $x$  that do not lie on  $\overline{W}$ ,  $\text{wind}(\overline{W}, x) \geq 0$ .

A844 These properties hold in the base case (2), where  $C(p, t, \emptyset) = 0$  and  $W$  is the single  
A845 point  $p$ . For the general formulas we heavily rely on the additivity of the winding number  
A846 with respect to gluing, Lemma 19. The cases are as follows, numbered by the equation  
A847 numbers; it may help to refer to Figure 4.

- A848 (4)  $C = C_1 = w_{pq} + M(qp, t - 1, B)$  where  $pq$  is a free-space edge.  
A849 By induction, the properties hold for the open walk  $W_0 = W(qp, t - 1, B)$ . Let  $W$  be the  
A850 polygon associated with  $C$ , i.e.,  $W = pq + W_0$ . Observe that  $W$  is the same polygon as  $\overline{W}_0$ .  
A851 This takes care of properties (ii) and (iii). For property (i), note that  $c(W) = w_{pq} + c(W_0)$ .  
A852 By induction,  $c(W_0) = M(qp, t - 1, B)$ . Thus  $c(W) = w_{pq} + M(qp, t - 1, B) = C$ , which  
A853 proves property (i).  
A854 (5)  $C = C_2 = C(p, t_1, B_1) + C(p, t_2, B_2)$  where  $t = t_1 + t_2$ ,  $B = B_1 \sqcup B_2$ ,  $B_1, B_2 \neq \emptyset$ .  
A855 By induction, the properties hold for the polygons  $W_1 = W(p, t_1, B_1)$  and  $W_2 =$   
A856  $W(p, t_2, B_2)$ . The polygon  $W$  associated with  $C$  is formed by gluing  $W_1$  and  $W_2$   
A857 at the common point  $p$ . The weights are additive by definition:  $w(W) = w(W_1) + w(W_2)$ ,  
A858 and by additivity of winding numbers,  $\text{wind}(W, x) = \text{wind}(W_1, x) + \text{wind}(W_2, x)$  for all  
A859 points  $x$  not on  $W$ . Property (iii) follows immediately, and property (i) follows by the  
A860 definition of the cost,  $c(W) = w(W) + \sum_{P \in O} \text{wind}(W, r_P) \cdot \pi_P$ .  
A861 Property (ii) propagates from  $B_1$  and  $B_2$  to their disjoint union  $B$  by the additivity  
A862 of winding numbers. More precisely, consider first some  $P \in B$ . Since  $B = B_1 \sqcup B_2$ ,  
A863 the polygon  $P$  is in exactly one of these sets. Suppose without loss of generality that  
A864  $P \in B_1$ . By induction,  $\text{wind}(W_1, r_P) = 1$  and  $\text{wind}(W_2, r_P) = 0$ . Thus  $\text{wind}(W, r_P) = 1$ .  
A865 Finally, if  $P \in R \setminus B$ , then by induction  $\text{wind}(W_1, r_P) = 0$  and  $\text{wind}(W_2, r_P) = 0$ , so  
A866  $\text{wind}(W, r_P) = 0$ , as required.  
A867 (6)  $M = M_1 = C(p, t - 1, B) + w_{pq}$  where  $pq$  is a free-space edge.  
A868 By induction, the properties hold for the polygon  $W_0 = W(p, t - 1, B)$ . The open walk  
A869  $W$  that is associated with  $M$  starts at  $p$ , traverses the polygon  $W_0$  and then the edge  
A870  $pq$ , ending at  $q$ .  $\overline{W}$  is formed by gluing the doubled edge  $qp$  to the polygon  $W_0$ . Thus,  
A871 winding numbers with respect to  $\overline{W}$  are the same as for  $W_0$ , except that they become  
A872 undefined for points  $x$  on  $pq$ . This proves properties (ii) and (iii), and also that the  
A873 penalty term in the cost (17) for  $W$  is the same as for  $W_0$ . Since  $w(W) = w(W_0) + w_{pq}$ ,  
A874 property (i) follows.  
A875 (7)  $M = M_2 = M(pr, t_1, B_1) + M(rq, t_2, B_2) + \pi(\Delta)$  where  $\Delta = prq$  is a counterclockwise  
A876 triangle,  $t = t_1 + t_2$ ,  $t_1 \geq 1$ ,  $t_2 \geq 1$ ,  $B = B_1 \sqcup B_2 \sqcup R(\Delta)$ .  
A877 By induction, the properties hold for the open walks  $W_1 = W(pr, t_1, B_1)$  and  $W_2 =$   
A878  $W(rq, t_2, B_2)$ . Let  $W$  be the open walk associated with  $M$ . Then  $w(W) = w(W_1) + w(W_2)$ .  
A879  $\overline{W}$  is formed by gluing  $\overline{W}_1$  and  $\overline{W}_2$  to  $\Delta$  on the common edges  $pr$  and  $rq$ , respectively.

A880 The argument is analogous to the treatment of (5) above, except that we form the  
 A881 combination of *three* areas, and two gluings are performed, along common edges instead  
 A882 of common vertices.

A883 An important point is therefore the treatment of reference points that lie *on* these edges:  
 A884 By the convention established in connection with Definition 18, the points on the mouths  
 A885  $pr$  and  $rq$  are not considered to be enclosed by  $\overline{W}_1$  and  $\overline{W}_2$ , both for determining  $R(\overline{W}_i)$   
 A886 and for computing  $\pi(\overline{W}_i)$ . However, when determining  $R(\Delta)$  and  $\pi(\Delta)$ , these edges are  
 A887 considered to be part of  $\Delta$ , by our conventions of Section 4.1 (in the paragraph before (6)).  
 A888 Thus the points on the mouth are neither overcounted nor undercounted.

A889 The edge  $pq$  is *not* considered as part of  $\Delta$ . This is in line with the convention of  
 A890 Definition 18 that the mouth  $pq$  should not be counted as enclosed by  $\overline{W}$ . ◀

## A891 **D** Details for Section 5: Uncrossing Algorithm

A892 We first give more details about the following proposition (restated).

A893 ▶ **Proposition 7** (Uncrossing Eulerian plane multigraphs). *Given a plane connected Eulerian*  
 A894 *multigraph  $H$  with  $m$  edges, specified by its combinatorial map, we can, in  $O(m)$  time,*  
 A895 *compute a non-crossing Euler tour of  $H$ .*

A896 As noted in the main text, a linear-time algorithm for constructing such an Euler tour was  
 A897 given by Akitaya and Tóth [3, Corollary 1]. Their algorithm makes the unstated assumption  
 A898 that the combinatorial map is given. Their terminology differs from ours, e.g., their input is  
 A899 geometric, and their output is a simple polygon that  $\varepsilon$ -approximates a non-crossing Euler  
 A900 tour. We outline the idea of the algorithm using our terminology. For the more general  
 A901 setting of graphs on arbitrary surfaces, a linear time algorithm was recently described by  
 A902 Bulavka, Colin de Verdière, and Fuladi [6, Lemma 4.2 of the full version on arXiv], expressed  
 A903 in the framework of cross-metric surfaces.

A904 **Idea of the proof of Proposition 7.** Take a 2-coloring (white and grey) of the faces of  $H$ ,  
 A905 with the outer face colored white. Traverse every grey face counterclockwise to obtain a set  
 A906 of edge-disjoint cycles without vertex crossings. The plan is to stitch together these cycles to  
 A907 form a non-crossing Euler tour. Initialize  $T$  to one of the cycles. While there are other cycles,  
 A908 find a vertex  $v$  where an edge of  $T$  and an edge of another cycle  $C$  appear consecutively in  
 A909 the cyclic order of edges around  $v$ , and merge  $C$  into  $T$  at this point. This does not create  
 A910 vertex crossings in  $T$ . The algorithm can be implemented to run in  $O(m)$  time. ◀

A911 We next give more details of our Uncrossing Algorithm, restated here.

A912 ▶ **Algorithm 8** (Uncrossing Algorithm).

- A913 1. *Subdivide every edge of  $W$  at every fork and interior crossing.*
- A914 2. *In the resulting multiset of edges (line segments in the plane) reduce multiplicities to 1*  
 A915 *or 2 by repeatedly discarding pairs of equal line segments. The result is a plane connected*  
 A916 *Eulerian multigraph.*
- A917 3. *Apply Proposition 7 to find a non-crossing Euler tour. This corresponds to a weakly*  
 A918 *simple polygon  $W'$ .*

A919 The definitions and results of Appendix A allow us to clarify this. For Step 1 we generalize  
 A920 the notion of an image graph to a polygon that may have interior crossings: first subdivide  
 A921 edges at interior crossings and then apply the previous definition of an image graph. In  
 A922 Step 1 we compute this image graph together with the multiplicity function. Step 2 simply

A923 modifies the multiplicities. In Step 3, the claim that a non-crossing Euler tour corresponds  
 A924 to a weakly simple polygon  $W'$  is justified by Lemma 16.

A925 **► Lemma 20.** *The Uncrossing Algorithm can be implemented to run in time  $O(t \log t + s)$   
 A926 where  $t$  is the number of edges of  $W$  and  $s \in O(t^2)$  is the number of interior crossings of  $W$ .  
 A927 (Multiple crossings at the same point are counted only once.) For input  $W_{\text{DP}}$  the runtime is  
 A928  $O(n \log n)$ .*

A929 **Proof.** We first show that the image graph  $G$  and multiplicities  $m(e)$  can be computed in  
 A930 time  $O(t \log t + s)$ . We must be careful to avoid the quadratic blow-up that results if we  
 A931 construct  $G$  in the obvious way by first subdividing edges of  $W$  at forks.

A932 One approach is to perform a plane sweep and represent overlapping segments in terms  
 A933 of multiplicities to avoid explicitly subdividing all edges in an overlapping bundle when one  
 A934 of those edges ends at a vertex.

A935 Another approach (following ideas in [2, 10]) is to compute multiplicities before running  
 A936 a plane sweep. Sort by slope to partition the edges into collinear groups. If  $\ell$  is a line that  
 A937 contains  $m$  edges, we can sort their endpoints along  $\ell$  in time  $O(m \log m)$  and output a  
 A938 corresponding set of  $O(m)$  interior-disjoint edges with multiplicities. We then run plane  
 A939 sweep on the new edges to compute  $G$  and its multiplicities in time  $O(t \log t + s)$ .

A940 Note that the image graph  $G$  (which is a simple plane graph) has at most  $t + s$  vertices,  
 A941 hence  $O(t + s)$  edges. The plane connected Eulerian multigraph  $M$  created in Step 2 (with  
 A942 edge multiplicities 1 or 2) has  $O(t + s)$  edges, and by Proposition 7, a weakly simple Euler  
 A943 tour of  $M$  can be found in time  $O(t + s)$ .

A944 Finally, consider running the algorithm on  $W_{\text{DP}}$ .  $W_{\text{DP}}$  has at most  $6n$  edges which gives  
 A945 an immediate runtime bound of  $O(n^2)$ . In fact, the runtime is less (though note that the  
 A946 runtime of the dynamic program dominates in any case). As a consequence of the optimality  
 A947 of  $W_{\text{ALG}}$  we prove (in Corollary 21) that  $W_{\text{DP}}$  has no interior crossing points. Thus the  
 A948 algorithm to uncross  $W_{\text{DP}}$  runs in time  $O(n \log n)$ . ◀

A949 We note that Akitaya and Tóth [3, Theorem 4] prove a related uncrossing result. Their  
 A950 input polygon has  $t$  edges and no interior crossings and they “uncross” to a weakly simple  
 A951 polygon with the same multiplicities as  $W$  and with  $O(t)$  edges, rather than the obvious  
 A952 quadratic number. They do not give a runtime. By contrast, we allow interior crossings (at a  
 A953 quadratic cost), and we escape the quadratic blow-up due to forks in a simpler way because  
 A954 we only care about parity.

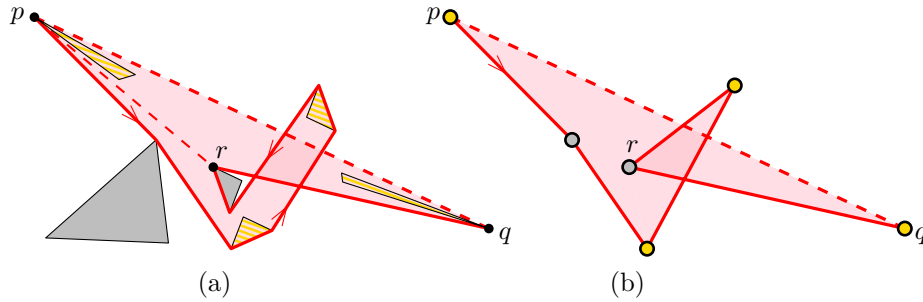
A955 The example of Figure 9 shows that self-crossings are not just a theoretical possibility;  
 A956 they actually can occur in an optimal solution to a type- $M$  subproblem. The solutions in  
 A957 this example are *weakly simple immersed polygons*, see Appendix L.

## A958 **E** Details for Section 6: Correctness Proof

A959 We restate and prove Lemma 13.

A960 **► Lemma 13.** *For the ENCLOSURE-WITH-PENALTIES problem, there exists an optimum  
 A961 solution  $W_{\text{OPT}}$  of finite cost that consists of at most  $6n$  free-space edges.*

A962 **Proof.** Recall that  $\mathcal{S}$  is the discrete set of feasible solutions that consist of free-space edges  
 A963 each traversed at most twice. There are two parts of the proof that warrant more detail  
 A964 than was given in the main text: (Part 1)  $\mathcal{S}$  contains a feasible solution that encloses  $R$  and  
 A965 excludes  $O$ ; and (Part 2) if  $W$  is a finite cost feasible solution outside  $\mathcal{S}$ , then there is a  
 A966 solution  $W'$  in  $\mathcal{S}$  of no greater cost.



■ **Figure 9** (a) The optimum solution with given mouth  $pq$  can indeed self-overlap. The yellow objects are required, and the grey objects have high penalties. The solution with mouth  $pr$  is a simple polygon. Attaching the triangle  $prq$  to it yields a self-overlapping polygon. (b) The same example with point objects. Observe that the solution covers more than  $360^\circ$  around  $r$ , although this is not apparent locally from looking at the boundary near  $r$ .

A967 **Part 1.** We begin with the boundaries of the polygons in  $R$  traversed counterclockwise.  
A968 These form a collection of  $k$  weakly simple polygons in the free space such that each free-space  
A969 edge is used at most twice. As long as there is more than one polygon, combine polygons as  
A970 follows. If two polygons share an edge, join them by removing that edge. Otherwise, if there  
A971 are polygons that share a vertex, find a vertex  $v$  where two polygons appear consecutively in  
A972 the cyclic order of edges around  $v$ , and merge the polygons at  $v$ . Otherwise, find a shortest  
A973 path among all paths in the free space that connect two vertices of different polygons, and  
A974 combine the two polygons into a single polygon by traversing the path once in each direction.  
A975 After  $k - 1$  steps, the process stops with a single weakly simple polygon, and this polygon  
A976 has the desired properties.

A977 **Part 2.** We now prove that if  $W$  is a finite cost feasible solution outside  $\mathcal{S}$ , then there is a  
A978 solution  $W'$  in  $\mathcal{S}$  of no greater cost. The idea is to construct a weakly simple polygon  $W'$  in  
A979  $\mathcal{S}$  that encloses the same objects as  $W$  and does not increase the sum of edge weights.

A980  $W$  may have vertices that are not object vertices. Let  $W_h$  be the result of homotopically  
A981 shortening (i.e., in the free space) every subpath of  $W$  that goes from one object vertex to  
A982 another. (If  $W$  contains no object vertex, we homotopically shorten all of  $W$ .) Then  $W_h$   
A983 is composed of free-space edges and  $w(W_h) \leq w(W)$ —this holds for edges with Euclidean  
A984 weights and also for squeezed edges. Although  $W_h$  need not be weakly simple, every object  
A985 has the same winding number (1 or 0) in  $W$  and  $W_h$ .

A986 Let  $W'$  be the result of applying the Uncrossing Algorithm 8 to  $W_h$ . Then  $W'$  is a weakly  
A987 simple polygon composed of free-space edges each used at most twice, so  $W'$  lies in  $\mathcal{S}$ . By  
A988 Lemma 9,  $W'$  preserves the winding numbers so  $W'$  encloses the same objects as  $W$ . Finally,  
A989  $w(W') \leq w(W_h)$ . ◀

A990 We note the following consequence of the above proof. It is used to analyze the runtime  
A991 of the Uncrossing Algorithm but nowhere else.

A992 ▶ **Corollary 21.**  $W_{\text{DP}}$  has no interior crossing.

A993 **Proof.** If  $W_{\text{DP}}$  had an interior crossing point, then the Uncrossing Algorithm (Algorithm 8)  
A994 would produce a walk  $W_{\text{ALG}}$  with a vertex at that crossing point, which is not a vertex of an  
A995 input polygon. The homotopic shortening step of Part 2 above would then strictly decrease  
A996 the weight, and thus the cost, of the solution, a contradiction to the optimality of  $W_{\text{ALG}}$ . ◀

A997 Finally we restate and prove Lemma 15. Recall the concepts of an open walk  $W_0$ , its  
 A998 closure  $\overline{W_0}$  and cost  $c(W_0)$  from Definitions 17 and 18.

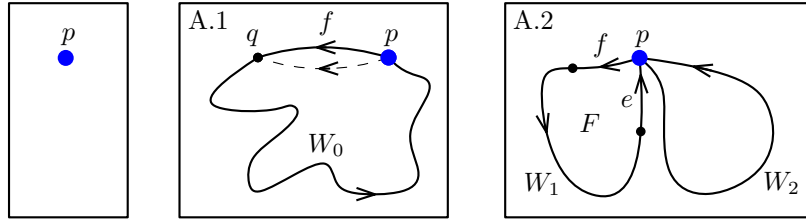
A999 ► **Lemma 15.** (A) Let  $W$  be a weakly simple polygon that consists of  $\ell$  free-space edges. Let  
 A1000  $p$  be a vertex of  $W$  and let  $B$  be the objects of  $R$  that are enclosed by  $W$ . Then, for all  $t \geq \ell$ ,  
 A1001  $C(p, t, B) \leq c(W)$ .

A1002 (B) Let  $W_0$  be an open walk with  $\ell$  free-space edges from vertex  $p$  to vertex  $q$  such that  
 A1003 the polygon  $W = W_0 + qp$  is weakly simple and  $q$  is not a transition vertex of  $W$ . Let  $B$   
 A1004 be the objects of  $R$  whose reference points lie inside  $W$  and not on  $pq$ . Then, for all  $t \geq \ell$ ,  
 A1005  $M(pq, t, B) \leq c(W_0)$ .

A1006 **Proof.** Let  $M(W)$  be a certificate that  $W$  is weakly simple, i.e.,  $M(W)$  is an image multigraph  
 A1007 in which  $W$  corresponds to a non-crossing Euler tour, see Appendix A. Via this correspondence,  
 A1008 each edge of  $W$  has an interior face of  $M(W)$  to its left, which provides a partition of the  
 A1009 edges of  $W$  into faces of  $M(W)$ . We use the cyclic order of edges around faces in the proof.  
 A1010 The reader may find it helpful to refer to Figure 3.

A1011 As in the proof of Lemma 6, we go through each case of the dynamic program recursion.  
 A1012 The difference is that in Lemma 6 we analyze, in terms of winding numbers, the cost of  
 A1013 any polygon constructed by the dynamic program (potentially not weakly simple), whereas  
 A1014 here we will deconstruct any weakly simple polygon into smaller pieces as defined by the  
 A1015 appropriate recursion formula, and winding numbers do not come into play.

A1016 We prove claims (A) and (B) simultaneously by induction on  $\ell$ . For part (A), see Figure 10.



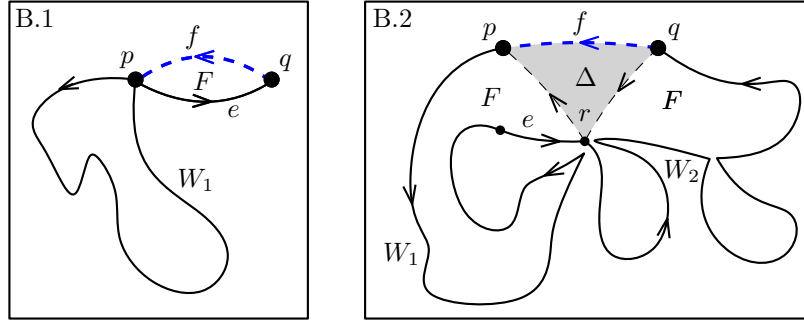
■ **Figure 10** Cases for statement (A) of Lemma 15.

A1017 In the base case,  $\ell = 0$ , polygon  $W$  degenerates to a single point, so only case (A)  
 A1018 applies. For objects with interior,  $W$  cannot enclose any objects, so  $B = \emptyset$ . By Equation (2),  
 A1019  $C(p, t, B) = 0 \leq c(W)$ .

A1020 For part (A), the case  $B = \emptyset$  was just dealt with, and for  $B \neq \emptyset$  we distinguish two cases  
 A1021 depending whether  $p$  is a transition vertex in  $W$ . Let  $f = pq$  be the edge of  $W$  that follows  $p$ .

A1022 **Case A.1.  $p$  is not a transition vertex.** Let  $W_0$  be the open walk from  $q$  to  $p$  formed by  
 A1023 removing the edge  $f$  from  $W$ . Then  $\overline{W_0} = W$  and  $W_0$  has  $\ell - 1 \leq t - 1$  free-space edges. By  
 A1024 Equation (4),  $C(p, t, B) \leq w_{pq} + M(qp, t - 1, B)$ , and by induction  $M(qp, t - 1, B) \leq c(W_0)$ .  
 A1025 Thus  $C(p, t, B) \leq w_{pq} + c(W_0) = c(W)$ , where the last equality comes from the definition of  
 A1026 the cost of the open walk  $W_0$ .

A1027 **Case A.2.  $p$  is a transition vertex.** Let  $F$  be the face of  $M(W)$  incident to edge  $f = pq$   
 A1028 and let  $e$  be the edge of  $W$  that precedes  $f$  around  $F$ . Suppose edge  $e$  enters vertex  $r$  of  $W$ ;  
 A1029 then vertices  $r$  and  $p$  are coincident. Cut  $W$  into two polygons, where  $W_1$  traverses  $W$  from  
 A1030  $p$  to  $r$  and  $W_2$  traverses  $W$  from  $r$  to  $p$ . Observe that  $W_1$  and  $W_2$  are both weakly simple,  
 A1031 and that no point is interior to both. For  $i = 1, 2$ , let  $\ell_i$  be the number of edges of  $W_i$ . Then  
 A1032  $\ell_i > 0$  and  $\ell_1 + \ell_2 = \ell$ . Let  $B_i = \{P \in O \mid r_P \text{ lies inside } W_i\}$ . Then  $B = B_1 \sqcup B_2$ .



■ **Figure 11** Cases for statement (B) of Lemma 15.

A1033 Suppose that neither  $B_1$  nor  $B_2$  is empty. Since  $t = \ell_1 + (t - \ell_1)$ , Equation (5) yields  
A1034  $C(p, t, B) \leq C(p, \ell_1, B_1) + C(p, t - \ell_1, B_2)$ . By induction,  $C(p, \ell_1, B_1) \leq c(W_1)$  and  $C(p, t -$   
A1035  $\ell_1, B_2) \leq c(W_2)$ . Thus  $C(p, t, B) \leq c(W_1) + c(W_2) = c(W)$ , where the last equality is because  
A1036  $W_1$  and  $W_2$  partition the edges and the interior of  $W$ .

A1037 On the other hand, if some  $B_i$ , say  $B_2$ , is empty, then  $B_1 = B$ . By induction,  $C(p, t, B) \leq$   
A1038  $c(W_1)$  since  $W_1$  has  $\ell_1 < t$  edges and contains  $B$ . Thus  $C(p, t, B) \leq c(W_1) \leq c(W)$ , where  
A1039 the last inequality is because  $W_1$ 's edges and interior are contained in those of  $W$ .

A1040 For part (B), we distinguish two cases depending whether the interior face  $F$  of  $M(W)$   
A1041 incident to edge  $f = qp$  of  $W$  is a corridor or a chamber, see Figure 11. Note that  $B = \emptyset$   
A1042 is allowed.

A1043 **Case B.1.  $F$  is a corridor.** Suppose  $q$  has incoming edge  $e$  and outgoing edge  $f$ . By  
A1044 assumption,  $q$  is not a transition vertex. Thus  $e$  is incident to face  $F$ , and forms the other  
A1045 side of the corridor. Since  $e$  is a free-space edge, so is  $f$ . By Equation (6),  $M(pq, t, B) \leq$   
A1046  $C(p, t - 1, B) + w_{pq}$ . Let  $W_1$  be the polygon formed by deleting edges  $e$  and  $f$  from  $W$ .  
A1047 Then  $W_1$  is a weakly simple polygon of  $\ell - 1$  free-space edges that encloses  $B$ . By induction,  
A1048  $C(q, t - 1, B) \leq c(W_1)$ . Thus  $M(pq, t, B) \leq c(W_1) + w_{pq} = c(W_0)$ , where the last equality is  
A1049 by definition of the cost of the open walk  $W_0$ .

A1050 **Case B.2.  $F$  is a chamber.** Take a triangulation of  $F$  (which exists because a chamber is  
A1051 an almost-simple polygon) and consider the triangle incident to edge  $f = pq$ . The triangle  
A1052 lies inside face  $F$ . Let  $v$  be the vertex of  $M(W)$  that forms the third corner of the triangle.  
A1053 Vertex  $v$  may correspond to more than one polygon vertex, but we choose the “right” one as  
A1054 follows. Let  $e$  be the edge of face  $F$  incoming to  $v$ . In  $W$ , suppose edge  $e$  enters vertex  $r$ .  
A1055 Name the triangle  $\Delta = pqr$ .

A1056 Break  $W$  into two open walks  $W_1$  from  $q$  to  $r$  and  $W_2$  from  $r$  to  $p$ . Observe that  $\overline{W_1}$  and  
A1057  $\overline{W_2}$  are weakly simple polygons and that  $r$  is not a transition vertex of  $\overline{W_1}$ . For  $i = 1, 2$ , let  
A1058  $\ell_i$  be the number of edges of  $W_i$ . Then  $\ell_i > 0$  and  $\ell_1 + \ell_2 = \ell$ . Let  $B_1$  be the set of polygons  
A1059  $P \in R$  with  $r_P$  in  $\overline{W_1}$  but not on  $qr$ , and let  $B_2$  be the set of polygons  $P \in R$  with  $r_P$  in  
A1060  $\overline{W_2}$  but not on  $rp$ . These sets may be empty. Let  $R(\Delta)$  be the set of polygons  $P \in R$  with  
A1061  $r_P$  inside  $\Delta$  where we regard  $\Delta$  as being closed on edges  $pr$  and  $qr$  and open on edge  $pq$ .  
A1062 Then  $B = B_1 \sqcup B_2 \sqcup R(\Delta)$ .

A1063 By Equation (7),  $M(pq, t, B) \leq M(pr, \ell_1, B_1) + M(rq, t - \ell_1, B_2) + \pi(\Delta)$ . By induction,  
A1064  $M(pr, \ell_1, B_1) \leq c(W_1)$  and  $M(rq, t - \ell_1, B_2) \leq c(W_2)$ . Thus  $M(pq, t, B) \leq c(W_1) + c(W_2) +$   
A1065  $\pi(\Delta) = c(W_0)$  where the last equality is because we have partitioned the edges of  $W_0$  and  
A1066 the interior of  $W$ . ◀

## F

 Details for Section 7: Reducing the Runtime

### F.1 Setting up a modified system of equations

For the original recursions (2)–(7), the absence of a cyclic dependence between the quantities  $C(p, t, B)$  and  $M(pq, t, B)$  is guaranteed by the second parameter  $t$ , which is always smaller on the right-hand side than on the left side. For the system (9)–(15) that was introduced in Section 7, we need to argue differently. In terms of the parameter representing the set of required objects, the parameter  $B$ ,  $B'$ , or  $B''$  on the right-hand side is always a subset of the parameter  $B$  on the left-hand side. Whenever it is a strict subset, the corresponding equation cannot be part of a cyclic dependence. The recursion is more delicate when the same set  $B$  appears on the right-hand side. To separate these cases, we split  $M_2$  into three parts  $M_3$ ,  $M_4$  and  $M_5$  consisting of those compositions where  $B' = B$ , where  $B'' = B$ , and where both  $B'$  and  $B''$  are strict subsets of  $B$ . Thus, equation (15) becomes:

$$M_2(pq, B) = \min\{M_3(pq, B), M_4(pq, B), M_5(pq, B)\}, \text{ where} \quad (18)$$

$$M_3(pq, B) = \min\{ M(pr, B) + M(rq, \emptyset) + \pi(\Delta) \mid \Delta = prq \text{ is a counterclockwise triangle, } R(\Delta) = \emptyset \} \quad (19)$$

$$M_4(pq, B) = \min\{ M(pr, \emptyset) + M(rq, B) + \pi(\Delta) \mid \Delta = prq \text{ is a counterclockwise triangle, } R(\Delta) = \emptyset \} \quad (20)$$

$$M_5(pq, B) = \min\{ M(pr, B') + M(rq, B'') + \pi(\Delta) \mid \Delta = prq \text{ is a counterclockwise triangle; } B = B' \sqcup B'' \sqcup R(\Delta); B', B'' \subsetneq B \} \quad (21)$$

For  $B = \emptyset$ , the equations (19) and (20) coincide, but this redundancy is no problem.

Equations (12) and (21), defining the quantities  $C_2(p, B)$  and  $M_5(pq, B)$ , have on the right-hand side quantities whose parameter,  $B'$  or  $B''$ , is a strict subset of  $B$ . Thus they cannot be involved in cyclic dependencies. On the other hand, equations (11), (14), (19), and (20), defining the quantities  $C_1(p, B)$ ,  $M_1(pq, B)$ ,  $M_3(pq, B)$ , and  $M_4(pq, B)$ , respectively, have on the right-hand side quantities whose parameter  $B$  is the same as the one on the left-hand side. By inspecting these equations, one can see that the left-hand quantity that is computed is always strictly bigger than the ingredients on the right-hand side, and hence the recurrences behave like a *superior context-free grammar* which uses strictly superior functions; see Knuth [19, Section 5]. Knuth proved that, for such a grammar, the minimum value of a string (representing a composition of functions) derived from each terminal symbol exists, is unique, and can be computed efficiently. In our problem, this translates to the fact that all the values  $C(p, B)$  and  $M(pq, B)$ , where  $p$  and  $q$  are vertices and  $B$  is any subset of the input set of required polygons  $R$ , exist, are unique, and can be computed efficiently.

Knuth's setup does not directly apply to our problem as far as uniqueness is concerned, because our functions are not *strictly* superior functions. This is compensated by having positive additive terms in the recursion. Our uniqueness proof below (Lemma 22) is a straightforward adaptation of Knuth's proof to our situation.

We show that the system of  $O(2^k n^2)$  equations (9)–(14) and (18)–(21) has a unique solution  $S = (C, M)$ . (We do not consider the auxiliary quantities  $C_1, C_2, M_1, M_2, M_3, M_4, M_5$  as part of the solution  $S$ , because they can be directly expressed in terms of  $C$  and  $M$ ). This, together with the fact that the solution of equations (2)–(7) is a solution to the system, implies that the solution  $S$  is the same as the one coming from equations (2)–(7).

A1106 **► Lemma 22.** *The system of equations (9)–(14) and (18)–(21) has a unique solution  $S =$*   
A1107  *$(C, M)$  with  $C(p, B) \in \mathbb{R}_{\geq 0} \cup \{\infty\}$  and  $M(pq, B) \in \mathbb{R}_{> 0} \cup \{\infty\}$ .*

A1108 **Proof.** The existence of a solution follows by substituting the limiting solution of the equations  
A1109 (2)–(7) for large enough  $t$ . All quantities  $M(pq, t, B)$  in those equations are positive because  
A1110 the quantity  $M_1$  (see equation (6)) is at least equal to the weight  $w_{pq}$  of the mouth, which is  
A1111 positive, and the other term  $M_2$  (see equation (7)) involves the addition of two quantities  
A1112  $M(pr, t_1, B_1)$  and  $M(rq, t_2, B_2)$  whose second parameter,  $t_1$  or  $t_2$ , is smaller than  $t$ .

A1113 We now prove uniqueness. The crucial fact that allows us to exclude a cyclic dependency  
A1114 is that in the equations (9)–(14) and (18)–(21), the quantities  $M$  and  $C$  on the right-hand  
A1115 side that could cause such a cyclic dependency (because they use the same set parameter  $B$ )  
A1116 must be strictly smaller than the quantities on the left side that are defined through them.

A1117 Assume, for contradiction, that there are two different solutions  $S = (C, M)$  and  $S' =$   
A1118  $(C', M')$ . Among the quantities where the two solutions differ, select the ones for which  
A1119 the parameter  $B$  is minimal, and among those, consider a pair with the smallest value  
A1120  $T = \min\{C(p, B), C'(p, B)\}$  or  $T = \min\{M(pq, B), M'(pq, B)\}$ .

A1121 Let us first deal with the case that the smallest difference occurs for  $C(p, B) \neq C'(p, B)$ .  
A1122 Assume without loss of generality that  $T = C(p, B) < C'(p, B)$ . By the minimality of  $B$ , we  
A1123 have that  $C(p, B') = C'(p, B')$  and  $C(p, B'') = C'(p, B'')$ , for any strict subsets  $B'$  and  $B''$   
A1124 of  $B$ . Hence, equation (12) gives us that  $C_2(p, B) = C'_2(p, B)$  and thus  $T = C_1(p, B) <$   
A1125  $C'_1(p, B)$ . By equation (11), we have that  $C_1(p, B)$  is equal to  $w_{pq} + M(qp, B)$  for some  
A1126 free-space edge  $pq$ . Since the weight  $w_{pq}$  is positive,  $M(qp, B)$  is strictly smaller than  $T$ , and  
A1127 hence, by the minimality of  $T$ , we have  $M(qp, B) = M'(qp, B)$ . It follows that

$$A1128 \quad C_1(p, B) = w_{pq} + M(qp, B) = w_{pq} + M'(qp, B) \geq C'_1(p, B),$$

A1129 a contradiction.

A1130 The same argument works for the case in which  $T = \min\{M(pq, B), M'(pq, B)\}$ . Here it  
A1131 is necessary to use the fact that all values  $M(pq, B)$  are positive. (Without this assumption,  
A1132 the identically zero solution  $M(pq, B) \equiv 0$  might be an alternative solution, for example.) ◀

## A1133 F.2 The algorithm

A1134 We now describe the algorithm to compute the values  $C(p, B)$  and  $M(pq, B)$  for all vertices  
A1135  $p$  and  $q$  and all the subsets  $B \subseteq R$  of required polygons. The algorithm has an outer loop  
A1136 that goes through all subsets  $B \subseteq R$  in order of increasing size  $|B|$ , or in any other order  
A1137 that is compatible with set inclusion.

A1138 When the algorithm needs to compute the values  $M(pq, B)$  and  $C(p, B)$  for a certain  $B$ ,  
A1139 the values  $M(rs, B')$  and  $C(r, B')$  have already been computed for all strict subsets  $B' \subset B$ ,  
A1140 all vertex pairs  $rs$  and all vertices  $r$ . This allows us to compute the values  $C_2(p, B)$  for all  
A1141 vertices  $p$ , via equation (12), and  $M_5(pq, B)$  for all vertex pairs  $pq$ , via equation (21).

A1142 The algorithm maintains a set  $F_1$  of vertices  $p$  for which the value  $C(p, B)$  has been  
A1143 determined, and a set  $F_2$  of vertex pairs  $pq$  for which the value  $M(pq, B)$  has been determined.  
A1144 Initially,  $F_1$  and  $F_2$  are empty. The algorithm also maintains *tentative* values  $M(pq, B)$  and  
A1145  $C(p, B)$ , which are upper bounds on their final values. When they become final, the corre-  
A1146 sponding item  $pq$  or  $p$  is added to  $F_2$  or  $F_1$ . Actually, what the algorithm maintains are tenta-  
A1147 tive values for  $M_1(pq, B)$ ,  $M_3(pq, B)$ ,  $M_4(pq, B)$  and  $C_1(p, B)$ , which are initialized to  $\infty$ . The  
A1148 values of  $C(pq, B)$  and  $M(pq, B)$  are kept up-to-date via  $C(p, B) = \min\{C_1(p, B), C_2(p, B)\}$   
A1149 and  $M(pq, B) = \min\{M_1(pq, B), M_3(pq, B), M_4(pq, B), M_5(pq, B)\}$ .

A1150 The core of the algorithm consists in making a tentative value final and adding the  
A1151 corresponding vertex or vertex pair to  $F_1$  or  $F_2$ . The strategy to do so is akin to the strategy  
A1152 of Dijkstra's algorithm for computing shortest paths: We pick the smallest tentative value  
A1153 and make it final. Then we look at all equations where this value appears on the right-hand  
A1154 side, and update the left-hand side. The pseudo-code in Algorithm 1 implements this in a  
A1155 straightforward way. (The only challenge is the confusion caused by the necessary renaming  
A1156 of the vertices  $p, q, r, s$ .)

■ **Algorithm 1** Computation of the values  $M(pq, B)$  and  $C(p, B)$  for fixed  $B$

---

**Input** : Set  $R$  of required polygons, set  $O$  of optional polygons, set  $B \subseteq R$   
**Output** : Values  $M(pq, B)$  for every pair of vertices  $pq$  and  $C(p, B)$  for every vertex  $p$

- 1 // For every strict subset  $B' \subset B$ , the values  $M(rs, B')$  for every pair of vertices  $rs$   
and  $C(r, B')$  for every vertex  $r$  have already been computed.
- 2 **for each** vertex  $p$  **do**
- 3   | Set  $C_1(p, B) := \infty$ ; compute  $C_2(p, B)$  by equation (12); set  $C(p, B) := C_2(p, B)$
- 4 **for each** vertex pair  $pq$  **do**
- 5   | Set  $M_1(pq, B) := M_3(pq, B) := M_4(pq, B) := \infty$ , and compute  $M_5(pq, B)$  by  
| equation (21); set  $M(pq, B) := M_5(pq, B)$  according to (13) and (18)
- 6 Set  $F_1 := F_2 := \emptyset$  //  $F_1$  contains the vertices  $p$  for which  $C(p, B)$  has been computed,  
and  $F_2$  the vertex pairs  $pq$  for which  $M(pq, B)$  has been computed.
- 7 **while** there are vertices not in  $F_1$  or vertex pairs not in  $F_2$  **do**
- 8   | Find the smallest value  $D$  among the tentative values  $C(p, B)$  with  $p \notin F_1$  and  
| the tentative values  $M(pq, B)$  with  $pq \notin F_2$ ; ties are broken arbitrarily.
- 9   | **if**  $D$  is  $C(p, B)$  **then**
- 10   |   | Set  $F_1 := F_1 \cup \{p\}$  // make  $C(p, B)$  permanent
- 11   |   | **for each** free-space edge  $sp$  incident to  $p$  **do**
- 12   |   |   | Set  $M_1(sp, B) := \min\{M_1(sp, B), w_{sp} + C(p, B)\}$  // by equation (14)
- 13   |   |   | Update  $M(sp, B) = \min\{M_1(sp, B), M_3(sp, B), M_4(sp, B), M_5(sp, B)\}$
- 14   | **if**  $D$  is  $M(pq, B)$  **then**
- 15   |   | Set  $F_2 := F_2 \cup \{pq\}$  // make  $M(pq, B)$  permanent
- 16   |   | **if**  $pq$  is a free-space edge **then**
- 17   |   |   | Set  $C_1(q, B) := \min\{C_1(q, B), w_{qp} + M(pq, B)\}$  // by equation (11)
- 18   |   |   | Update  $C(q, B) := \min\{C_1(q, B), C_2(q, B)\}$
- 19   |   | **for each** counterclockwise triangle  $\Delta = psq$  with  $R(\Delta) = \emptyset$  **do**
- 20   |   |   | Set  $M_3(ps, B) := \min\{M_3(ps, B), M(pq, B) + M(qs, \emptyset) + \pi(\Delta)\}$  // by (19)
- 21   |   |   | Update  $M(ps, B) := \min\{M_1(ps, B), M_3(ps, B), M_4(ps, B), M_5(ps, B)\}$ ;
- 22   |   |   | Set  $M_4(sq, B) := \min\{M_4(sq, B), M(sp, \emptyset) + M(pq, B) + \pi(\Delta)\}$  // by (20)
- 23   |   |   | Update  $M(sq, B) := \min\{M_1(sq, B), M_3(sq, B), M_4(sq, B), M_5(sq, B)\}$

---

A1157 Correctness is established in the same way as for Dijkstra's algorithm. The smallest  
A1158 tentative value  $D$  that is determined at the beginning of each iteration of the main loop is  
A1159 simultaneously a lower bound on all tentative values and an upper bound on all permanent  
A1160 values. The algorithm ensures that every tentative value always fulfills its corresponding  
A1161 equation (10) or (13). Thus, whenever a value is finalized, the equation is fulfilled. The  
A1162 values on the right-hand side on which it depends do not change any more because they are

A1163 smaller than  $D$ , and hence they have already been finalized.

A1164 Thus, when the algorithm terminates, the computed values  $C(p, B)$  and  $M(pq, B)$  fulfill  
 A1165 (10) and (13). The correct values  $C(p, 6n, B)$  and  $M(pq, 6n, B)$  also fulfill these equations,  
 A1166 and by Lemma 22 the solution of (10) and (13) is unique, and hence the computed values  
 A1167 agree with the correct values.

### A1168 F.3 Runtime analysis

A1169 Clearly, there are  $O(2^k n^2)$  values  $M(pq, B)$  and  $C(p, B)$ , and this defines the space complexity.

A1170 We first analyze the computations of the quantities  $C_2(p, B)$  and  $M_5(pq, B)$ , which are  
 A1171 computed directly by equations (12) and (21), respectively. The dominating term for the  
 A1172 runtime comes from the computation of the  $O(2^k n^2)$  quantities  $M_5(pq, B)$ . For each of them,  
 A1173 we have to run through all points  $r$  and check each counterclockwise triangle  $\Delta = prq$ : We  
 A1174 have to find the set

$$A1175 \quad R(\Delta) := \{P \in R \mid r_P \in \Delta\}, \quad (22)$$

A1176 and compute the sum  $\pi(\Delta)$  of the penalties of the polygons in  $O$  whose reference point  
 A1177 is in  $\Delta$  and, in case  $R(\Delta) \subseteq B$ , run through all partitions of  $B - R(\Delta)$  into two sets  $B'$   
 A1178 and  $B''$ . We describe below a preprocessing step that allows us to obtain the quantity  $\pi(\Delta)$   
 A1179 in constant time. The set  $R(\Delta)$  can be trivially computed in  $O(k)$  time. Thus the total  
 A1180 running time for computing the quantities  $M_5(pq, B)$  is

$$A1181 \quad n^2 \sum_{B \subseteq R} \left( n \times (O(k) + 2^{|B|}) \right) = O(n^3 2^k k) + O(n^3) \sum_{B \subseteq R} 2^{|B|} = O(3^k n^3). \quad (23)$$

A1182 Let us now look at the running time of the core of the algorithm, in which, repeatedly,  
 A1183 a tentative value is made final. Consider a fixed subset  $B \subseteq R$  (there are  $2^k$  such sets).  
 A1184 We need to maintain a priority queue for the  $O(n^2)$  tentative values for the quantities  
 A1185  $C(p, B)$  and  $M(pq, B)$ . Each of the  $O(n^3)$  expressions on the right-hand side of any of the  
 A1186 equations (11), (12), (14), and (19)–(21) is evaluated exactly once (when the corresponding  
 A1187 quantity becomes final) or twice (in case  $B = \emptyset$ , for the expression  $M(pq, \emptyset) + M(pq, \emptyset)$ ). The  
 A1188 evaluation potentially triggers an update to the priority queue, which takes  $O(1)$  amortized  
 A1189 time with Fibonacci heaps [11, 16]. We need to extract the minimum  $O(n^2)$  times, at an  
 A1190 amortized cost of  $O(\log n)$  per operation. The overall runtime for the heap operations is then  
 A1191  $O(n^2 \log n + n^3) = O(n^3)$ . In summary, the overall runtime for this part of the algorithm is  
 A1192  $2^k \cdot O(n^3)$ , which is dominated by (23).

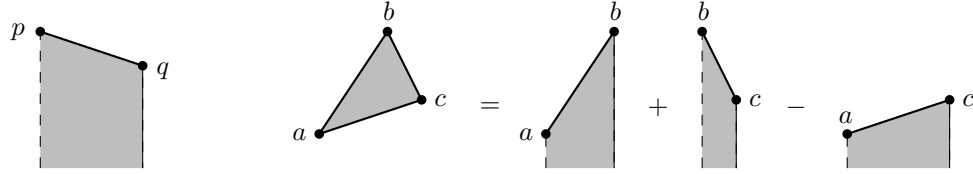
A1193 Thus, up to showing how  $\pi(\Delta)$  can be determined in constant time (which we will do  
 A1194 below), we have established Theorem 3. ◀

### A1195 F.4 Preprocessing for quickly determining the penalty of a triangle

A1196 One can set up a table with  $O(n^2)$  entries from which, for any triangle  $\Delta$  whose vertices are  
 A1197 vertices of the input polygons, the sum of the penalties of the polygons whose reference point  
 A1198 is in  $\Delta$  can be obtained in constant time. This is a standard technique in this area, see for  
 A1199 example [14, Section 2]. We give some details.

A1200 A *plank* is a region bounded by a nonvertical line segment  $pq$  and two vertical upward  
 A1201 rays or two vertical downward rays originating from  $p$  and  $q$ . We consider the right boundary  
 A1202 ray and the open line segment  $pq$  to be part of the plank, but not the left boundary ray  
 A1203 including the point  $p$ . Figure 12 shows some downward planks (“bottomless trapezoids”,  
 A1204 so-to-speak). Upward planks (or “topless trapezoids”) are used in Section 8.

A1205 We store for each vertex pair  $pq$ , the sum of the penalties in the downward plank below  
 A1206 the segment  $pq$ . From this, the same data  $\pi(\Delta)$  can then be computed for any triangle  
 A1207  $\Delta = abc$  in constant time by addition and subtraction from three planks, see Figure 12 for  
 A1208 an example. The table can be computed in  $O(n^2)$  time and  $O(n^2)$  space [14, Theorem 2.1].  
 A1209 Even a straightforward  $O(n^3)$  preprocessing would be acceptable for us, as the running time  
 A1210 is dominated by the cost of other parts of the algorithm.



■ **Figure 12** Left: a plank under the segment  $pq$ . Right: A triangle area  $abc$  is obtained by addition and subtraction of planks.

A1211 Reference points of objects with infinite penalties are handled separately, in the same  
 A1212 way: Instead of storing the sum of their penalties, they are merely *counted*, to determine  
 A1213 whether the triangle in question contains at least one of them or none. Finally, recall that  
 A1214  $\Delta = prq$  was defined to be open on segment  $pq$ . To handle this, we also calculate the sum the  
 A1215 penalties of the reference points *on* each segment  $pq$  (as well as the number of infinite-penalty  
 A1216 points). These have then to be added or subtracted as appropriate. Triangles with vertical  
 A1217 edges have to be handled specially, but this poses no challenge.

## G Details for Section 8: Adaptations for the Inverted Problem

### G.1 The dynamic programming recursion

A1220 For simplicity, we assume that distinct vertices and distinct reference points have distinct  
 A1221  $x$ -coordinates; this can be achieved by a rotation. We denote by  $H_{\leftarrow}(q)$  and  $H_{\rightarrow}(q)$  the  
 A1222 left and right half-plane bounded by the vertical line through  $q$ .  $S_{\downarrow}(pq)$  and  $S_{\uparrow}(pq)$  are the  
 A1223 planks with boundary segment  $pq$ . By convention,  $p$  is always left of  $q$ .

A1224 The recursion considers three cases, as illustrated in Figure 13. The easy case ( $U_{\leftarrow}$ ) is a  
 A1225 left half-plane, which applies only for  $p = q$ . The other two cases are symmetric to each other;  
 A1226 we discuss here only  $U_{\downarrow}$ . This is similar to the term  $M_2$  for  $M(pq, t, B)$  in recursion (7),  
 A1227 except that the plank  $S_{\downarrow}(rp)$  plays the role of the triangle  $\Delta = prq$ . One of the subproblems,  
 A1228 with mouth  $pr$ , is an “ordinary” subproblem of type  $M$ , the other subproblem is of type  $U$ .

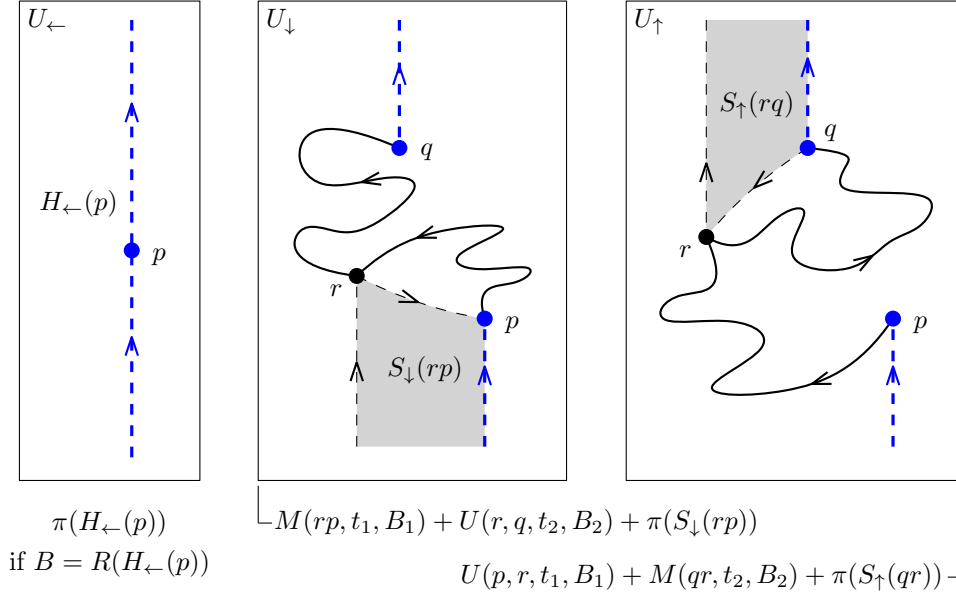
$$U(p, q, t, B) = \min\{U_{\leftarrow}, U_{\downarrow}, U_{\uparrow}\}, \text{ where}$$

$$U_{\leftarrow} = \begin{cases} \pi(H_{\leftarrow}(p)), & \text{if } p = q \text{ and } B = R(H_{\leftarrow}(p)) \\ \infty, & \text{otherwise} \end{cases} \quad (24)$$

$$U_{\downarrow} = \min\{ M(pr, t_1, B_1) + U(r, q, t_2, B_2) + \pi(S_{\downarrow}(rp)) \mid \\ r \text{ left of } p, t = t_1 + t_2, B = R(S_{\downarrow}(rp)) \sqcup B_1 \sqcup B_2 \} \quad (25)$$

$$U_{\uparrow} = \min\{ U(p, r, t_1, B_1) + M(rq, t_2, B_2) + \pi(S_{\uparrow}(rq)) \mid \\ r \text{ left of } q, t = t_1 + t_2, B = R(S_{\uparrow}(rq)) \sqcup B_1 \sqcup B_2 \} \quad (26)$$

A1236  $R(\Omega)$  and  $\pi(\Omega)$  (for some region  $\Omega$ ) generalize the earlier notations  $R(\Delta)$  and  $\pi(\Delta)$  and  
 A1237 denote the required objects and the sum of penalties of optional objects whose reference  
 A1238 point lies inside  $\Omega$ . We clarify that reference points that lie *on* a segment  $pq$  are treated as



■ **Figure 13** The dynamic programming recursion for the inverted problem. Free-space edges are solid; mouths are dashed.

A1239 belonging to  $S_{\downarrow}(pq)$  and  $S_{\uparrow}(pq)$ . No reference points lie on other boundaries of planks and  
 A1240 half-planes since they do not have the same  $x$ -coordinates as vertices.

A1241 The overall solution is

A1242 
$$c'_{\text{DP}} := \min\{U(p, p, R(H_{\leftarrow}(p)), 6n) + \pi(H_{\rightarrow}(p)) \mid p \text{ is a vertex}\}$$

A1243 The first term, with  $B = R(H_{\leftarrow}(p))$ , makes sure that all required objects with the reference  
 A1244 point to the left of  $p$  are covered. The remaining objects are then automatically covered by  
 A1245 the right half-plane  $H_{\rightarrow}(p)$ .

A1246 **G.2 Adapting the correctness proof for the inverted problem**

A1247 To apply the arguments from Section 6, we have to extend the notion of winding number  
 A1248 to regions  $W^{\uparrow}$  whose boundary includes an upward and a downward vertical ray. We pick  
 A1249 an anchor point  $X_-$  to the left of all object vertices. We define  $\text{wind}(W^{\uparrow}, X_-) = 1$ , and  
 A1250 define the winding number for other points relative  $X_-$  by connecting them by a curve  
 A1251 to  $X_-$  and counting signed intersections. It follows that  $\text{wind}(W^{\uparrow}, x) = 0$  for any point  $x$   
 A1252 that is sufficiently far to the right. The winding number of planks must also be defined  
 A1253 appropriately: The winding number of  $S_{\downarrow}(pq)$  or  $S_{\uparrow}(pq)$  is 1 between the two rays, on the  
 A1254 “correct” side of the segment  $pq$ , and 0 otherwise.

A1255 When the region is finished off by adding a right halfplane,  $W$  becomes a closed clockwise  
 A1256 cycle. By the usual conventions, the interior has winding number  $-1$  and the exterior has  
 A1257 winding number 0. The winding number that we are using has an additive offset of 1,  
 A1258 due to the stipulation that the point  $X_-$  has winding number 1. Thus, according to our  
 A1259 convention, the winding number is 1 outside  $W$  and 0 inside  $W$ , which is precisely what we  
 A1260 need because the objects whose presence is checked and whose penalties are added are the  
 A1261 objects outside  $W$ .

A1262 The correctness proof in Lemma 15 must be adapted as follows. In the inductive step,  
 A1263 we have a solution  $W^{\uparrow}$  consisting of a finite walk  $W$  from  $p$  to  $q$  and two vertical rays. The

case that  $p = q$  and the walk  $W$  is trivial is handled by formula (24) for  $U_-$ . Suppose that  $p \neq q$  and, w.l.o.g.,  $p$  is not the leftmost point of  $W$ . Then we take the edge  $pr$  on the lower convex hull of  $W$ . The formula (25) for  $U_\downarrow$  shows that  $W^\uparrow$  can be reduced to smaller pieces.

There is another issue that we have to address, namely the partial solutions of type  $C$  (“closed”) that are incident to the convex hull, such as the pieces  $Q_a, Q_d, Q_f$  in Figure 6, are properly handled. Every convex hull edge, such as the edge  $p_6p_7$ , appears once as a mouth in the recursion. As can be checked in Figure 13, it appears always in counterclockwise direction along the boundary. For example, the edge  $p_6p_7$  appear in a subproblem of the form  $M(p_6p_7, t, B)$ , for appropriate parameters  $B$  and  $t$ . According to Lemma 15, this problem will consider partial solutions in which  $p_6$  is not a transition vertex. However, there is no such restriction on  $p_7$ . Thus we assign all type- $C$  pieces hanging off  $p_7$  to this subproblem. (In the example, there is only one such piece,  $Q_d$ .)

The general strategy is as follows. We cut the solution walk  $W$  into pieces at the convex hull vertices, and assign a piece to each convex hull edge, which acts as the mouth of the piece. Pieces of type  $C$  that start and end at a hull vertex  $p_i$  are assigned to the hull edge  $p_{i-1}p_i$  clockwise from  $p_i$ . In this way, the pieces are uniquely defined, and we have ensured that for each mouth  $p_i p_{i-1}$ ,  $p_{i-1}$  is never a transition vertex of the respective piece. Thus, by Lemma 15, the weight of the corresponding piece is an upper bound on  $M(p_i p_{i-1}, t, B)$  with the appropriate parameters  $t$  and  $B$ . The remainder of the proof, regarding the coverage of outside the convex hull by adding halfspaces and planks, is straightforward.

The example of Figure 7 illustrates that the region covered by the planks need not actually be the outside of the convex hull of the solution polygon: Regions 3 and 4 form an indentation in the convex hull. In fact, any “ $x$ -monotone hull” of the solution can be taken.

## H Illustration for Section 9: Splitting a Surface by a Curve

Figure 14 illustrates the problem of splitting off a piece of given genus from a surface, which was mentioned in the conclusion, Section 9.

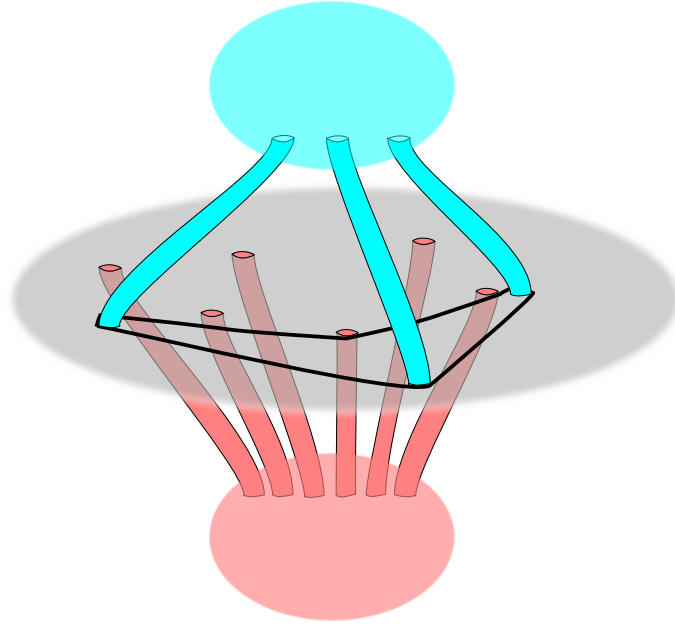
## I Point Objects

In this section we modify our algorithm to handle input objects that are a mix of points and almost-simple polygons. The condition that polygons have disjoint interiors is replaced by the condition that the interior of an object may not intersect another object. So a point object is either disjoint from all polygon objects, or it lies on the boundaries of some polygon objects. We subdivide polygon edges to ensure that no point object lies in the interior of a polygon edge.

Point objects disjoint from polygons could be approximated by tiny polygons, but point objects at polygon vertices cannot be dealt with so cavalierly. Instead, we show how to modify our algorithm to deal directly with point objects.

We must clarify the output requirements in case a point object  $p$  lies on the boundary of a solution  $W$ . The only reasonable way to decide the matter in this case without making the problem ill-posed is to consider  $p$  to be enclosed or not at our discretion, since, by an arbitrarily small perturbation of  $W$  in the vicinity of  $p$ , either outcome can be achieved. This agrees with the convention used by Eades and Rappaport [13]. More precisely, we adapt the notion of a feasible solution  $W$  as follows:

For each point object  $p \in R$ , we require that  $p$  lies in the interior or on the boundary of  $W$  (possibly several times).



■ **Figure 14** Any instance of GRAPH-ENCLOSURE-WITH-PENALTIES with infinite penalties only, and with parameter  $k$ , can be recast as an instance of the problem of splitting off a surface of genus  $k - 1$ . In this example, the graph  $G$  that is an instance of GRAPH-ENCLOSURE-WITH-PENALTIES is embedded in the middle gray disk; each of the  $k = 3$  required faces is connected by a tube (in cyan) to the top sphere; each of the 6 optional faces, with infinite penalty, is connected by a tube (in red) to the bottom sphere. Finally, the gray disk is extended to a sphere (not shown), to obtain a surface  $S$  without boundary. Any weakly simple closed walk in  $G$  separating the required faces from the optional ones corresponds to a weakly simple closed walk splitting off a surface of genus  $k - 1 = 2$ , and conversely. (The graph  $G$  is not cellularly embedded on  $S$ , but can be made so by adding edges of large weight.) Figure inspired by [7].

A1308 The penalty of an optional point object  $p \in O$  that lies on the boundary of  $W$  is not counted  
A1309 towards the cost in formula (1).

A1310 We use a reference point  $r_P$  for each object  $P$ . For a point object, the reference point  
A1311 must be that point itself. A reference point lying on a mouth in a subproblem  $M(pq, t, B)$   
A1312 was already handled by the algorithm. What is new is the possibility that a vertex is a point  
A1313 object, either required or optional.

A1314 We make two changes to the dynamic programming algorithm:

A1315 1. If  $p \in R$  is a required point object, we add an extra possibility to equation (3) for the  
A1316 case  $B = \{p\}$  as follows:

$$A1317 \quad C(p, t, \{p\}) := 0 \text{ for } t \geq 0 \tag{27}$$

A1318 2. For equation (7) we used a triangle  $\Delta = prq$  that was defined to be open on edge  $pq$   
A1319 and closed on edges  $pr$  and  $qr$ . We redefine  $\Delta$  to exclude its corners  $p, q, r$ , i.e., the only  
A1320 boundary points of  $\Delta$  that are included are the interiors of edges  $pr$  and  $qr$ . This affects  
A1321 both  $\pi(\Delta)$  and  $R(\Delta)$ .

A1322 We explain the effect of these changes informally, and then outline how our proofs of  
A1323 correctness must be modified.

A1324 First observe that the changes to  $\Delta$  mean that  $\pi(\Delta)$  does not count penalties of optional  
A1325 point objects at the corners of  $\Delta$  in equation (7), which is the correct thing to do. This is

the only place in the equations where penalties are added.

Consider now a required point object  $p$ . At the top level,  $p$  lies in  $R$ , and this is passed to the disjoint sets  $B$  used in the recursions. If ever  $p$  is contained in  $R(\Delta)$ , then this is where  $p$  is considered to be enclosed (and it can only be enclosed once in this way). At the bottom of the recursion, rule (27) permits us to consider  $p$  as enclosed, and rule (2) permits us to consider  $p$  as not enclosed. Thus, we can “catch” the object  $p$  on the boundary if we have not done so already in a triangle. If the boundary goes through  $p$  several times, we can catch it on one occasion and pass over it on the other occasions.

To make this more formal, we adapt Lemma 15 in the following way:

► **Lemma 23.** (A) *Let  $W$  be a weakly simple polygon that goes through some vertex  $p$  and consists of  $\ell$  free-space edges. Let  $B_{\text{in}}$  be the objects of  $R$  that are enclosed by  $W$ , and let  $B_{\text{point}}$  be the point objects of  $R$  that coincide with vertices of  $W$ . Then, for all  $t \geq \ell$  and for all  $B$  with  $B_{\text{in}} \subseteq B \subseteq B_{\text{in}} \cup B_{\text{point}}$ ,  $C(p, t, B) \leq c(W)$ .*

(B) *Let  $W_0$  be an open walk with  $\ell$  free-space edges from vertex  $p$  to vertex  $q$  such that the polygon  $W = W_0 + qp$  is weakly simple. Let  $B_{\text{in}}$  be the objects of  $R$  whose reference points lie inside  $W$  and not on  $pq$ , and let  $B_{\text{point}}$  be the point objects of  $R$  that coincide with vertices of  $W$ , excluding  $p$ .*

*In addition, assume that  $q$  is not a transition vertex of  $W$ . Then, for all  $t \geq \ell$  and for all  $B$  with  $B_{\text{in}} \subseteq B \subseteq B_{\text{in}} \cup B_{\text{point}}$ ,  $M(pq, t, B) \leq c(W_0)$ . ◀*

Note in particular that, in the statement of part (B), we have added the condition that if  $p$  is a required point in  $R$ , it cannot be in  $B$  (in addition to requiring that  $p$  is not a transition vertex).

The induction basis, treating the trivial polygons with  $t = 0$  edges, is covered by (2) and (27).

For the closed walks in statement (A), when  $p$  is a point object in  $B$ , we cannot apply the strategy for case A.1, because it would lead to the subproblem  $M(qp, \dots)$  in which  $p$  is a point object, for which the extra requirement for (B) does not hold. Thus, in this case, if  $p$  is not a transition vertex anyway, we make a degenerate split as in case A.2, with an empty walk  $W_2$  and  $B_2 = \{p\}$ , and consequently  $W_1 = W$ , see Figure 10. Equation (5) yields

$$C(p, t, B) \leq C(p, t, B \setminus \{p\}) + C(p, 0, \{p\}) \leq C(p, t, B \setminus \{p\}) + 0.$$

To the subproblem  $C(p, t, B_1)$  with  $B_1 = B \setminus \{p\}$ , case A.1 applies, since  $p$  is no longer an element of  $B_1$  for this subproblem. This leads to

$$C(p, t, B_1) \leq w_{pq} + M(qp, t - 1, B_1) \leq c(W)$$

and hence to  $C(p, t, B) \leq c(W)$ .

In case B.2, where we split the set  $B$  into  $B_1 \sqcup B_2 \sqcup R(\Delta)$ , the splitting is clear: we have to assign any point objects on  $W_1$  to  $B_1$  and any point objects on  $W_2$  to  $B_2$ . If the vertex  $r$  is a required point and belongs to  $B$ , we use the freedom of choice to put it in  $B_2$  (the subproblem belonging to the mouth  $rp$ ) and not in  $B_1$ , ensuring that the inductive hypothesis can be applied to the first subproblem  $M(pr, t_1, B_1)$ .

Case B.1 does not require any changes.

The other part of the correctness proof is based the properties of  $W_{\text{DP}}$  proved in Lemma 6. Lemma 6(B) claims that for  $P \in R$ ,  $\text{wind}(W_{\text{DP}}, r_P) = 1$ . But for a point object  $p \in R$  that lies on  $W_{\text{DP}}$ , the winding number is undefined. Thus, we have to restrict this claim to required point objects that do not lie on  $W_{\text{DP}}$  (and ditto in the claims for the subproblems

A1370 in the inductive proof). For a point object  $p \in R$  lying on  $W_{\text{DP}}$ , we simply observe that it  
 A1371 will also lie on  $W_{\text{ALG}}$  because the Uncrossing Algorithm does not remove points from the  
 A1372 polygon boundary. Hence  $p$  fulfills the adapted requirements of a feasible solution.

## A1373 **J** Negative Penalties, or Rewards

A1374 We now consider the extension of GEOMETRIC-ENCLOSURE-WITH-PENALTIES and GRAPH-  
 A1375 ENCLOSURE-WITH-PENALTIES in which we allow objects with negative penalties. In order to  
 A1376 have a balanced and general statement, we use a greater variety of types of polygons/faces:  
 A1377 there is a set  $R^-$  of **required** polygons/faces, a set  $R^+$  of **forbidden** polygons/faces  
 A1378 (the notation suggests that these sets correspond to objects with penalty  $-\infty$  and  $+\infty$ ,  
 A1379 respectively), and a set  $O^+ \sqcup O^0 \sqcup O^-$  of **optional** polygons/faces, whose penalties are  
 A1380 finite and positive for the objects in  $O^+$ , zero for the objects in  $O^0$ , and finite and negative  
 A1381 for the objects in  $O^-$ . The goal is to find a weakly simple closed curve/walk disjoint from  
 A1382 the objects, enclosing  $R^-$ , excluding  $R^+$ , and minimizing the length of the curve plus the  
 A1383 penalties of the objects in  $O^+ \cup O^0 \cup O^-$  that are enclosed by the curve.

A1384 **► Theorem 24.** *We can solve these generalized problems in time  $O(3^k n^3)$  time and  $O(2^k n^2)$   
 A1385 space, where  $k = \min\{|R^-| + |O^-|, |R^+| + |O^+|\}$ .*

A1386 **Proof.** Let us first consider the case where  $k = |R^-| + |O^-|$ . The idea is to try all subsets  
 A1387 of  $O^-$  that can be enclosed in an optimal solution.

A1388 We run the dynamic program with set of required objects  $R := R^- \cup O^-$  and set of  
 A1389 optional objects  $O := R^+ \cup O^+ \cup O^0$ , where the objects in  $R^+$  have infinite penalties and  
 A1390 those in  $O^+ \cup O^0$  keep their original nonnegative penalties; this takes  $O(3^{|R|} n^3) = O(3^k n^3)$   
 A1391 time. As part of the dynamic programming recursion, the algorithm determines  $C(p, B)$  for  
 A1392 all subsets  $B \subseteq R$  and all vertices  $p$ . We therefore have available all quantities that enter the  
 A1393 following formula for the optimum solution:

$$A1394 \quad C_{\text{final}}^* := \min_{O_1^- \subseteq O^-} \left( \left( \min_{p \text{ a vertex}} C(p, R^- \cup O_1^-) \right) + \sum_{P \in O_1^-} \pi_P \right) \quad (28)$$

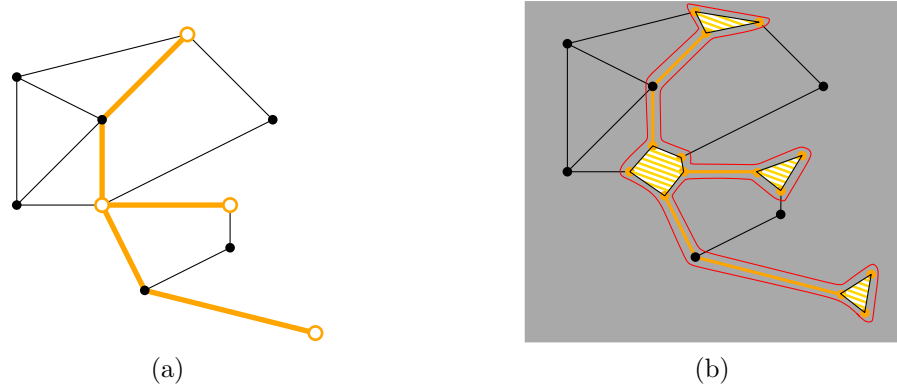
A1395 This formula is justified as follows: The solutions considered for  $\min_p C(p, R^- \cup O_1^-)$  are  
 A1396 those solutions that, among the objects in  $R = R^- \cup O^-$ , enclose precisely the objects of  
 A1397  $R^- \cup O_1^-$  and exclude the remaining objects of  $O^-$ . In this way, we consider all solutions  
 A1398 that enclose  $R$  plus some arbitrary subset  $O_1^- \subseteq O^-$  of the negative-weight objects. The  
 A1399 negative weights of the objects in  $O_1^-$  are explicitly added in (28) to get the correct value of  
 A1400 the objective function.

A1401 The time  $O(2^{|O^-|} n)$  for calculating  $C_{\text{final}}^*$  by (28) is dominated by the runtime  $O(3^k n^3)$   
 A1402 of the dynamic programming algorithm.

A1403 In the other case where  $k = |R^+| + |O^+|$ , we invoke the inverted algorithm (Section 8)  
 A1404 instead of the algorithm of Theorem 1 or 2. This swaps  $R^+$  with  $R^-$  and  $O^+$  with  $O^-$ . The  
 A1405 rest of the argument is identical. ◀

## A1406 **K** Exponential Lower Bounds

A1407 We first prove a (conditional) exponential lower bound for the GRAPH-ENCLOSURE-WITH-  
 A1408 PENALTIES problem. The proof consists of a simple reduction to our problem from the  
 A1409 PLANAR STEINER TREE problem.



■ **Figure 15** (a) An instance of the PLANAR STEINER TREE problem. Terminals are large empty disks. Edges of a tree  $S$  connecting the terminals are thick and yellow. (b) The corresponding instance of the GRAPH-ENCLOSURE-WITH-PENALTIES problem. Faces in  $R$  are yellow/hatched, faces in  $O$  (including the unbounded face) are gray. The weakly simple closed walk  $W$  constructed from  $S$  is represented by a red curve.

A1410 ► **Theorem 25.** *Assuming the Exponential Time Hypothesis, the GRAPH-ENCLOSURE-WITH-*  
A1411 *PENALTIES problem cannot be solved in  $2^{o(k)} \cdot n^{O(1)}$  time, even when all the weights are 1,*  
A1412 *and all penalties are  $\infty$ .*

A1413 **Proof.** The proof consists of a reduction from the PLANAR STEINER TREE problem, whose  
A1414 input is an edge-weighted planar graph  $G$  with  $n$  vertices and a set  $T$  of  $k$  vertices of  $G$ ,  
A1415 usually called *terminals*. The problem asks for a minimum-weight tree in  $G$  connecting  
A1416 all terminals. Marx, Pilipczuk, and Pilipczuk [20, Theorem 1.2] proved that the PLANAR  
A1417 STEINER TREE problem cannot be solved in  $2^{o(k)} \cdot n^{O(1)}$  time, assuming the Exponential  
A1418 Time Hypothesis, even if the input graph is unweighted (that is, all the edge weights are 1).  
A1419 We now describe the reduction.

A1420 Given an unweighted planar graph  $G$  and a set  $T \subseteq V(G)$ , as the one in Figure 15(a),  
A1421 we replace each terminal  $v$  in  $T$  with a corresponding *terminal cycle*  $C_v$ , whose number of  
A1422 edges is equal to  $\max\{3, d_G(v)\}$ , where  $d_G(v)$  denotes the degree of  $v$  in  $G$ ; each vertex of  $C_v$   
A1423 is connected to a different neighbor of  $v$ , and the interior of  $C_v$  is a face, see Figure 15(b).  
A1424 Denote by  $H$  the obtained plane graph and by  $\gamma$  the total number of *terminal edges*, i.e., edges  
A1425 in terminal cycles. Every edge of  $H$  has weight 1. Let  $R$  be the set of faces inside terminal  
A1426 cycles, and let  $O$  be the set of all the other faces. The faces in  $O$  have penalty  $\infty$ . This  
A1427 completes the reduction. We now prove that  $G$  contains a tree with weight  $\leq w$  connecting  
A1428 the terminals in  $T$  if and only if  $H$  has a weakly simple closed walk  $W$  with weight  $\leq 2w + \gamma$   
A1429 that has the faces in  $R$  inside (and the faces in  $O$  outside).

A1430 For the forward implication, given any tree  $S$  in  $G$  with weight at most  $w$  connecting  
A1431 the terminals in  $T$ , one can construct the desired walk as follows. The walk traverses  
A1432 each edge in  $S$  twice (once in each direction), and traverses each terminal cycle once, in  
A1433 counter-clockwise direction.

A1434 For the backward implication, let  $W$  be a weakly simple closed walk in  $H$  with weight  
A1435 at most  $2w + \gamma$  that has the faces in  $R$  inside and the faces in  $O$  outside.  $W$  contains each  
A1436 terminal edge at least once, because  $W$  must separate  $R$  from  $O$ . Moreover,  $W$  uses each  
A1437 non-terminal edge of  $H$  an even number of times, as otherwise one of its incident faces,  
A1438 both of which have penalty  $\infty$ , would be inside  $W$ . Since  $W$  is connected, it contains at  
A1439 least twice each edge of a connected subgraph  $S_H$  spanning the terminal cycles. The simple

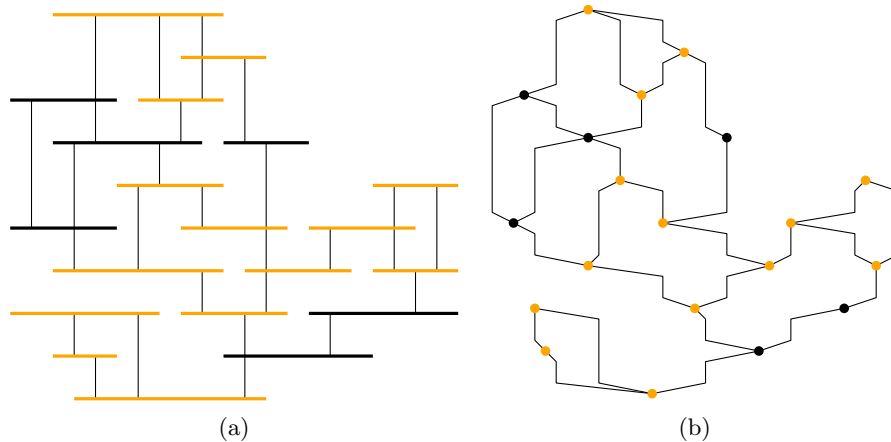
A1440 graph  $S_G$  in  $G$  corresponding to  $S_H$  connects all the terminals. The weight of  $S_G$  is at most  
 A1441  $((2w + \gamma) - \gamma)/2 = w$ , which it is obtained from the weight  $2w + \gamma$  of  $W$  by subtracting  
 A1442 the total weight of the terminal edges, which is at least  $\gamma$ , and by then dividing by two as  
 A1443 each edge of  $S_G$  is used at least twice in  $W$ . We conclude the proof by observing that  $S_G$   
 A1444 contains a tree that spans all terminals and has weight at most  $w$ . ◀

A1445 We now present a reduction similar to, and slightly more technical than, the one of  
 A1446 Theorem 25 for the geometric version of our problem.

A1447 ▶ **Theorem 26.** *Assuming the Exponential Time Hypothesis, the GEOMETRIC-ENCLOSURE-*  
 A1448 *WITH-PENALTIES problem cannot be solved in  $2^{o(k)} \cdot n^{O(1)}$  time, even when all penalties are  $\infty$ .*

A1449 **Proof.** Consider an instance  $(G, T)$  of the PLANAR STEINER TREE problem in which  $G$   
 A1450 has  $n$  vertices and edges with weight 1. We start by constructing the  $O(n)$ -vertex planar  
 A1451 graph  $H$  as in the proof of Theorem 25. We now construct a sequence of representations  
 A1452 of  $H$ , and eventually get the desired instance of GEOMETRIC-ENCLOSURE-WITH-PENALTIES.

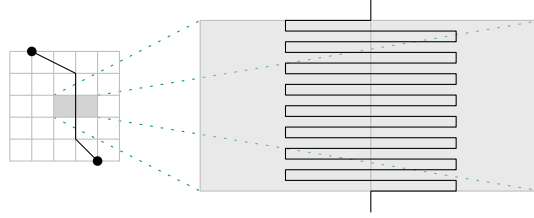
A1453 First, we construct a *visibility representation*  $\Gamma$  of  $H$  on an  $O(n) \times O(n)$  grid [23],  
 A1454 see Figure 16(a). In  $\Gamma$ , vertices are represented by disjoint horizontal segments lying on  
 A1455 grid rows and edges are represented by disjoint vertical segments lying on grid columns.  
 A1456 Each vertical segment representing an edge has its endpoints on the horizontal segments  
 A1457 representing the end-vertices of the edge and otherwise does not cross any horizontal segment  
 A1458 representing a vertex. We scale all the coordinates in the drawing up by a factor of 5.



■ **Figure 16** (a) A visibility representation  $\Gamma$  of the graph  $H$  from Figure 15(b). We stress that the relative interior of a vertical segment representing an edge of  $H$  does not intersect any horizontal segment representing a vertex of  $H$ ; vertical lines may consist of several vertical segments. (b) A poly-line drawing  $\Gamma'$  of  $H$  constructed from  $\Gamma$ . Both representations lie on an  $O(n) \times O(n)$  grid.

A1459 Second, we turn  $\Gamma$  into a poly-line drawing  $\Gamma'$ ; this can be done by modifying  $\Gamma$  only  
 A1460 “close” to its vertices, see [5, 12] and Figure 16(b). Specifically, each horizontal segment  $s_v$   
 A1461 representing a vertex  $v$  is replaced by a grid point  $p_v$  on  $s_v$ . Also, we shorten each vertical  
 A1462 segment representing an edge  $uv$  by one unit at the top and at the bottom and connect the  
 A1463 endpoints to  $p_u$  and  $p_v$ .

A1464 Third, we turn  $\Gamma'$  into a poly-line drawing  $\Gamma''$  in which all edges have “almost” the same  
 A1465 length. Intuitively, we are going to modify the representation of each edge by “orthogonally  
 A1466 zig-zagging” in an intermediate part of the edge, so that the edge has length between  $n^2$  and  
 A1467  $n^2 + 1$ , see Figure 17. As a consequence of the scaling of  $\Gamma$ , the representation of each edge  $e$



■ **Figure 17** The left part of the figure shows an edge  $e$  in the drawing  $\Gamma'$ . The right part shows an enlarged central section of  $e$  in which the drawing of  $e$  is modified in order to transform  $\Gamma'$  into a poly-line drawing  $\Gamma''$  of  $H$  in which  $e$  has length between  $n^2$  and  $n^2 + 1$ . The edge  $e$  is only modified in its portion  $\sigma_e$  inside the two gray grid cells.

A1468 in  $\Gamma'$  contains a vertical segment  $\sigma_e$  between two grid points  $(i, j)$  and  $(i, j + 1)$  such that  $\Gamma'$   
A1469 has no intersection with the two grid cells incident to  $\sigma_e$ , other than at  $\sigma_e$  itself. Let  $\ell_e$  be  
A1470 the length of the polygonal chain representing  $e$  in  $\Gamma'$  and let  $a_e = n^2 - \lfloor \ell_e \rfloor$  be the increase  
A1471 of length that we want for  $e$ . Then we can replace  $\sigma_e$  by the orthogonal line passing through  
A1472 points  $(i, j), (i + \frac{1}{2}, j), (i + \frac{1}{2}, j + \frac{1}{a_e}), (i - \frac{1}{2}, j + \frac{1}{a_e}), (i - \frac{1}{2}, j + \frac{2}{a_e}), (i + \frac{1}{2}, j + \frac{2}{a_e}), \dots, (i, j + 1)$ .  
A1473 The vertical segments of this line have total length 1, while the horizontal segments have  
A1474 total length  $a_e$  (two of them have length  $\frac{1}{2}$ , while the other  $a_e - 1$  have length 1). Hence,  
A1475 the length of  $e$  has increased by  $n^2 - \lfloor \ell_e \rfloor$  and it is now between  $n^2$  and  $n^2 + 1$ .

A1476 In order to get the instance of GEOMETRIC-ENCLOSURE-WITH-PENALTIES, we interpret  
A1477 the faces of  $\Gamma''$  as polygons: those inside the terminal cycles are in  $R$  and those corresponding  
A1478 to faces of  $G$  are in  $O$  and have penalty  $\infty$ . Since all the edges have approximately the same  
A1479 length, between  $n^2$  and  $n^2 + 1$ , the same proof as in Theorem 25 shows that  $G$  contains a tree  
A1480 with weight  $\leq w$  connecting the terminals in  $T$  if and only if there exists a weakly simple  
A1481 closed walk  $W$  with weight  $\leq (2w + \gamma) \cdot (n^2 + 1)$  in the instance of GEOMETRIC-ENCLOSURE-  
A1482 WITH-PENALTIES. Indeed, the “only if” part is easy, and the proof for the “if” part uses the  
A1483 following argument. From the weakly simple closed walk  $W$ , one can extract a simple graph  $S_G$   
A1484 in  $G$  spanning all the terminals whose weight is at most  $\frac{(2w+\gamma) \cdot (n^2+1) - \gamma \cdot n^2}{2n^2} = w + \frac{(2w+\gamma)}{2n^2}$ .  
A1485 Since  $2w + \gamma$  is in  $O(n)$ , we have that  $\frac{(2w+\gamma)}{2n^2}$  is in  $o(1)$ , hence  $S_G$  has at most  $w$  edges  
A1486 provided  $n$  is large enough, which we can obviously assume. The described reduction takes  
A1487 polynomial time, given that all the vertex coordinates in  $\Gamma''$  are rational numbers whose  
A1488 numerators and denominators are polynomially bounded. ◀

A1489 The above proof essentially contains a polynomial, parameter-preserving reduction from  
A1490 GRAPH-ENCLOSURE-WITH-PENALTIES to GEOMETRIC-ENCLOSURE-WITH-PENALTIES. In  
A1491 passing, we mention that there is also a polynomial-time, parameter-preserving reduction in  
A1492 the other direction: Given an instance of GEOMETRIC-ENCLOSURE-WITH-PENALTIES, we  
A1493 know that the output will consist of free-space edges, so one can compute the graph that is  
A1494 the overlay of all free-space edges (equivalently, of the visibility graph of the input vertices),  
A1495 assign each subdivided edge a weight that is its Euclidean length, and assign penalty zero to  
A1496 each face of this arrangement that does not come from an input polygon. This results in an  
A1497 equivalent instance of GRAPH-ENCLOSURE-WITH-PENALTIES.

## L Weakly Simple Immersed Polygons

A1499 We can define the precise class of polygons over which the dynamic program optimizes. They  
A1500 are more general than the weakly simple polygons that we want as a solution, because they  
A1501 can self-cross, but they are not arbitrary polygons.

A1502 It turns out that  $M(pq, t, B)$  and  $C(p, t, B)$  is the minimum cost (with the extended  
A1503 meaning of Definition 5) of a *weakly simple immersed polygon* that satisfies appropriately  
A1504 modified constraints that correspond to the intended constraints regarding the number  
A1505 of edges, the set  $B$  of objects whose reference points are enclosed, and the mouth  $pq$  or  
A1506 startpoint  $p$ , respectively.

A1507 As in Appendix A and Appendix C.2, we describe such a polygon as a sequence of  
A1508 vertices forming its boundary cycle, in the form  $P = (p_1, p_2, \dots, p_n)$ . The polygon runs  
A1509 counterclockwise around its “enclosed region”, with the interior to its left.

A1510 **Weakly simple immersed polygons (WSImP).** A *weakly simple immersed polygon*  
A1511 (WSImP) is obtained by gluing together triangles and digons in a tree-like fashion.

A1512 There are two base cases:

- A1513 ■ a counterclockwise nondegenerate triangle  $(p, q, r)$
- A1514 ■ a digon  $(p, q)$

A1515 The two ways of inductively combining two WSImPs into a larger WSImP are the same  
A1516 combinations that we introduced in Appendix C.2 for arbitrary polygons:

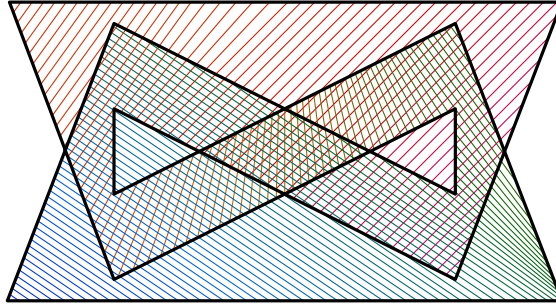
- A1517 ■ Two WSImPs can be glued together along a common *edge*: If  $P_1 = (p, q, q_2, \dots, q_n)$  and  
A1518  $P_2 = (q, p, p_2, \dots, p_m)$  both use the edge  $pq$ , but in opposite directions, then we can form  
A1519 the WSImP  $P = (q, q_2, \dots, q_n, p, p_2, \dots, p_m)$ . (This is the same as gluing together two  
A1520 polygons in the plane along a common edge if they lie on different sides of that edge,  
A1521 except that we do not care whether they overlap.)
- A1522 ■ Two WSImPs  $P_1 = (p, q_1, q_2, \dots, q_n)$  and  $P_2 = (p, q_1, q_2, \dots, q_n)$  can be glued together at  
A1523 a shared *vertex*  $p$ , forming a new WSImP  $P = (p, q_1, q_2, \dots, q_n, p, p_1, p_2, \dots, p_m)$ . (The  
A1524 vertex  $p$  becomes a transition vertex.)

A1525 By construction, our dynamic program computes a WSImP  $W$  that has the correct  
A1526 winding number for all objects in  $R$ . Since WSImPs can be triangulated, the same proof  
A1527 as that of Lemma 15 shows that the cost of  $W$  is optimal. More precisely,  $M(pq, t, B)$  and  
A1528  $C(p, t, B)$  is the minimum cost of a weakly simple immersed polygon  $W$  under the following  
A1529 constraints:

- A1530 1. For  $M(pq, t, B)$ , the walk connects the endpoints  $p$  and  $q$  and is closed by the mouth  $qp$ ;  
A1531 for  $C(p, t, B)$ , it goes through the startpoint  $p$ .
  - A1532 2. The number of free-space edges is at most  $t$ , not counting the mouth in case of  $M(pq, t, B)$ .
  - A1533 3. For each object  $P \in B$ ,  $\text{wind}(W, r_P) = 1$ , and for each object  $P \in R \setminus B$ ,  $\text{wind}(W, r_P) = 0$ .
- A1534 The cost is interpreted with the extended meaning of Definition 5, and the weight  $w_{pq}$  is  
A1535 subtracted in case of  $M(pq, t, B)$ .

A1536 If we restrict the base case to triangles and only allow gluings along edges, we arrive at  
A1537 the subclass of *(simple) immersed polygons* (SImPs). Here, the construction defines a  
A1538 simply connected surface, which is obtained by starting with the triangles and performing  
A1539 the gluing as an identification of common points. The boundary walk of a SImP is known as  
A1540 a *self-overlapping polygon*, see for example Evans and Wenk [15] for a recent discussion.  
A1541 The boundary walks of WSImPs are related to self-overlapping polygons in the same way as  
A1542 *weakly simple* polygons are related to *simple* polygons.

A1543 The relation between a WSImP and its boundary walk is delicate, just as for self-  
A1544 overlapping polygons: It is not the case that the SImP as a boundary determines this surface  
A1545 uniquely. Figure 18 shows the simplest counterexample, which is known under the name  
A1546 *Milnor’s doodle*. By construction, a SImP comes with a triangulation, but the triangulation  
A1547 is obviously not unique. Shor and Van Wyk [21] define a SImP as an equivalence class of



■ **Figure 18** Milnor’s doodle. The hatching indicates one of two symmetric ways of viewing this self-overlapping polygon as the boundary of an immersed surface.

A1548 triangulations of a self-overlapping polygon, and they give an algorithm to count the number  
 A1549 of SImPs for a given self-overlapping polygon [21, Section 6].

A1550 Thus, for specifying a WSimP, the boundary walk is not sufficient: We would have to  
 A1551 specify the sequence of gluings describing how the WSimP was built. For our purposes,  
 A1552 however, the precise SImP or WSimP is irrelevant, and the boundary walk is all we need:  
 A1553 By Lemma 19, we can find out how often a point in the plane is covered by  $P$  by calculating  
 A1554 the winding number.

## A1555 **M** The Geometric Knapsack Algorithm

A1556 In this section we mention some issues with the algorithm by Arkin, Khuller, and Mitchell [4]  
 A1557 for the geometric knapsack problem. The problem was described in Section 1.1. As mentioned  
 A1558 in that section, they solve an inverted problem: to minimize the length of the solution polygon,  
 A1559 which they call a “fence”, plus the values of the objects outside the fence.

A1560 We point out two issues: their algorithm does not deal with the complications that arise  
 A1561 due to the possibility of a weakly simple (but not simple) enclosure; and their local approach  
 A1562 to identifying the outside of the fence can lead to an incorrect output.

A1563 With regard to the first issue, they write, “an optimal enclosure in the presence of  
 A1564 obstacles will not, in general, be convex. It will, however, be a simple polygon . . .” [4,  
 A1565 Section 3, p. 411]. Their algorithm relies on identifying the point where a path exits (i.e.,  
 A1566 crosses) the fence. The condition of being crossed by a path is trickier and less local for  
 A1567 weakly simple polygons than for simple polygons, so their algorithm needs to be augmented  
 A1568 to specify exactly what happens in degenerate situations. We believe that this is a technical  
 A1569 issue that can be overcome by a careful treatment of these degeneracies.

A1570 To explain the second issue, we first summarize their algorithm. It relies on the fact  
 A1571 that the solution area can be assumed to be geodesically convex. The reason is that an  
 A1572 obstacle-avoiding shortcut between two points on the boundary of the fence through the  
 A1573 outside will grow the enclosed area and thus not degrade the solution. (This argument fails  
 A1574 in our more general scenario when some objects are required to lie outside the fence.)

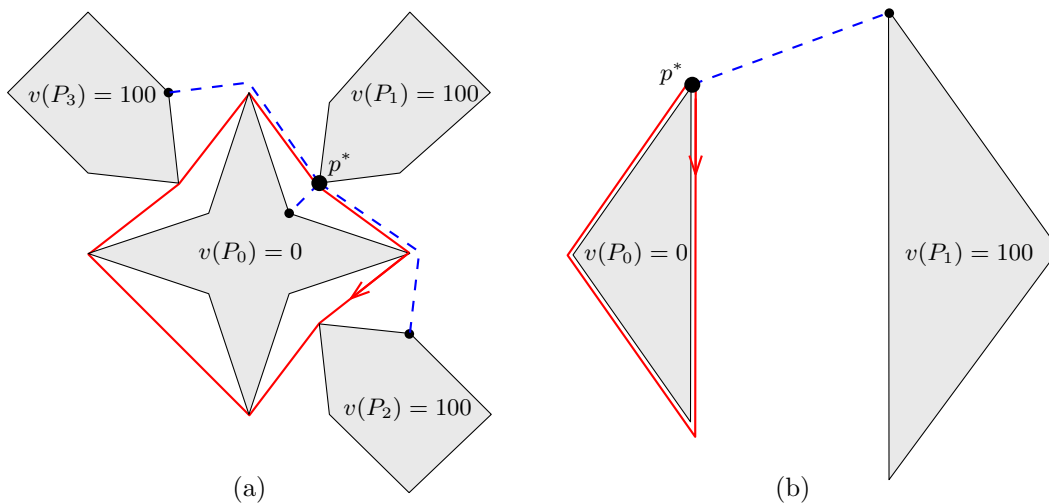
A1575 For consistency, we stick with our convention of surrounding the enclosed area in counter-  
 A1576 clockwise direction by the fence, while the opposite convention is used in [4].

A1577 The solution consists of edges of  $\mathcal{VG}$ , the visibility graph of the object vertices. For  
 A1578 each object vertex  $p^*$ , the algorithm finds an optimal fence that goes through  $p^*$ , and then  
 A1579 optimizes over the choice of  $p^*$ . For given  $p^*$  they find a directed cycle through  $p^*$  using  
 A1580 edges of  $\mathcal{VG}$  with carefully chosen weights (which they call “costs”), which are defined as

A1581 follows. Choose a reference vertex on each object. Consider an object  $O$  and a shortest path  
 A1582  $\pi_O$  from  $p^*$  to the reference vertex of  $O$ . They prove that there is an optimum solution that,  
 A1583 for all  $O$ , intersects  $\pi_O$  in a connected subpath [4, Lemma 3]. This implies that  $O$  is inside  
 A1584 the fence iff the path  $\pi_O$  remains inside the fence, and  $O$  is outside iff  $\pi_O$  exits the fence,  
 A1585 in which case it exits once at a unique point. Thus their idea is to charge the value of an  
 A1586 outside object  $O$  to the point of  $\mathcal{VG}$  where its path  $\pi_O$  exits the fence. For an edge of  $\mathcal{VG}$   
 A1587 that is crossed by a path  $\pi_O$ , the value of  $O$  is simply added to the weight of the edge. If  $\pi_O$   
 A1588 goes through a vertex  $v$ , the question whether it crosses the fence can only be decided by  
 A1589 looking at both edges incident to  $v$ . The value of  $O$  is added to the cost of any triple  $e, v, f$   
 A1590 where the directed edge  $e$  enters  $v$ , the directed edge  $f$  leaves  $v$ ,  $\pi_O$  goes through  $v$  and,  
 A1591 after passing through  $v$  enters the interior of the wedge to the right of  $e, v, f$ . An appropriate  
 A1592 version of this rule should be used when  $v$  is the reference point of  $O$ .

A1593 This declares the outside of the directed cycle to be to its right, implicitly assuming that  
 A1594 the cycle goes in the counterclockwise direction. However, there is no enforcement that the  
 A1595 right side of the cycle will actually be the unbounded part of the plane. Figure 19 shows  
 A1596 two examples where the algorithm mistakenly finds a polygon that counts the values of the  
 A1597 objects in the bounded region determined by the cycle. We think that this can be repaired  
 A1598 by requiring that  $p^*$  is a topmost vertex of the cycle, i.e., by allowing only visibility edges  
 A1599 below  $p^*$  when  $p^*$  is considered as the starting point.

A1600 This modification does not affect the runtime of the algorithm. The runtime of the  
 A1601 algorithm remains at  $O(n^2|\mathcal{VG}|) = O(n^4)$ , where  $|\mathcal{VG}|$  denotes the number of edges of the  
 A1602 visibility graph [4, Theorem 6]. Even if the visibility graph is sparse, i.e.,  $|\mathcal{VG}| = O(n)$ , the  
 A1603 algorithm could not improve over our  $O(n^3)$  approach; hence we did not investigate this  
 A1604 potential fix of the algorithm.



**Figure 19** Bad examples for the geometric knapsack algorithm of [4]. All edge lengths are very small compared to the non-zero object values. Overlapping paths are slightly displaced for visibility. (a) The optimum fence is the counterclockwise convex hull of the four polygons. When the algorithm of [4] considers  $p^*$  and the dashed blue paths to the polygon reference points, it finds the red cycle with “interior” to its left—the length is smaller and the value of “outside” polygons is 0. (b) The optimum fence just hugs  $P_1$  in the counterclockwise direction but the algorithm considers  $p^*$  and finds the red cycle with “interior” to its left. The polygon  $P_0$ , which contains  $p^*$ , is considered to lie “outside” the fence.