Discrete Comput Geom 33:463–481 (2005)
DOI: 10.1007/s00454-004-1148-9

Discrete & Computational
Geometry

A Near-Quadratic Algorithm for Fence Design*

Pankaj K. Agarwal,[1] Robert-Paul Berretty,[2] and Anne D. Collins[3]

[1]Department of Computer Science, Duke University,
Durham, NC 27708-0129, USA
pankaj@cs.duke.edu

[2]Philips Research Laboratories, Building WDC 1-053,
Prof. Holstlaan 4, 5656 AA Eindhoven, The Netherlands
robert-paul.berretty@philips.com

[3]Department of Mathematics, Stanford University,
Bldg. 380, Stanford, CA 94305-2125, USA
collins@math.stanford.edu

Abstract. A part feeder is a mechanism that receives a stream of identical parts in arbitrary orientations and outputs them oriented the same way. Various sensorless part feeders have been proposed in the literature. The feeder we consider consists of a sequence of fences that extend partway across a conveyor belt; a polygonal part P carried by the belt is reoriented by each fence it encounters. We present an $O(m + n^2 \log^3 n)$-time algorithm to compute a sequence of fences that uniquely orients P, if one exists, where m is the total number of vertices and n is the number of stable edges of P. We reduce the problem to searching for a path in a state graph that has $O(n^3)$ edges. By exploiting various geometric properties of this graph, we show that it can be represented implicitly and that a desired path can be computed in $O(m + n^2 \log^3 n)$ time. We believe that our technique is quite general and could be applicable to other part-manipulation problems as well.

1. Introduction

Robotic manipulation deals with various part-manipulation problems in industrial automation [12]. One such problem, which arises in automated assembly, is the so-called

* Research by P.A. was supported by the NSF under Grants CCR-00-86013, EIA-98-70724, EIA-01-31905, and CCR-97-32787, and by a grant from the U.S.–Israel Binational Science Foundation. Research by R.B. was supported by the Dutch Organization for Scientific Research (N.W.O.). Research by A.C. was supported by the NSF under Grants CCR-00-86013, CCR-97-32787, DMS-0107621, and DMS-9983320. Part of this work was done while R.B. was visiting the Department of Computer Science, University of North Carolina, Chapel Hill and A.C. was a Ph.D. student at Duke University.

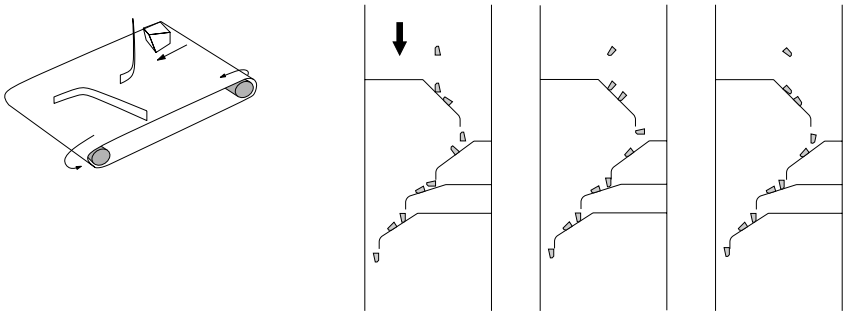

Fig. 1. A polygonal part is moved by a conveyor belt past a series of rigid fences. The fences passively reorient the part, which leaves the last fence in a unique orientation regardless of its initial orientation.

part feeding or *orienting* problem. Many automated manufacturing processes require parts to be oriented prior to assembly. A part feeder receives a stream of identical parts in arbitrary orientations and outputs them oriented the same way. Although many part feeders use some kind of sensors, we are interested in *sensorless* feeders, for which the initial orientation of the part is unknown and parts are oriented using passive mechanical compliance. A variety of sensorless part feeders have been proposed. For example, parts on a conveyor belt can be oriented using a sequence of stationary fences or a single moving fence, using horizontal pins suspended above the belt that can topple a three-dimensional part as it moves by, or using a collection of conveyor belts at varying heights. Parts can be pushed with fences, squeezed or rolled between parallel-jaw grippers, pulled from the inside-out, dropped through traps, tilted on a table, or subjected to vibrating plates with programmable vector fields. See [2], [15], [12], and the references therein.

Part feeders are typically created on a case-by-case basis, and it can take a long time to design a feeder for a single part. Only recently, researchers have begun to focus on automating the design process itself. In this paper we focus on designing a system of fences that uniquely orients a two-dimensional part moving on a conveyor belt.

Our Model. Our feeder consists of a conveyor belt equipped with a series of fences designed to reorient a part passively as it moves by. The fences are rigid, frictionless bars attached to walls on either side of the belt that extend partway across the belt at some fixed angle. See Fig. 1 for an example. We refer to our feeder as a *fence feeder*.

We assume the part to be a planar polygon P, which lies on the conveyor belt and is translated past each fence. Since a fence is only in contact with the boundary of its convex hull, we can assume P to be a convex m-gon. When the part encounters a fence, it simultaneously rotates and slides until one of its stable edges aligns with the fence; we assume that the fences are long enough to allow ample time for this reorientation.[1] Once aligned, the part slides compliantly along the fence until it reaches the end.

[1] We assume that the motion of P follows the pushing model proposed by Mason [11]. See the original paper and [2] for details, which we omit from here.

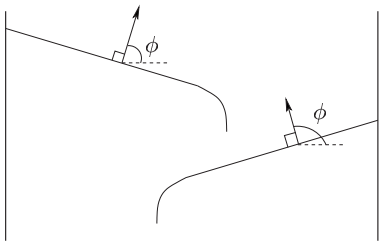

Fig. 2. The fence angle φ is the direction of the upward normal to the fence.

In order to avoid any uncertainty in the orientation of the part as it leaves a fence, we add a carefully curved tail, as in [5]. This ensures that when the part leaves the fence, the aligned edge is parallel to and facing the wall from which the fence emanates.

We work in a frame of reference in which the belt moves downward. A fence is specified by its *fence angle* φ,[2] the direction of the ray normal to the fence with positive vertical component. If the fence is attached on the left side of the belt, then $\varphi \in (0, \pi/2)$, while $\varphi \in (\pi/2, \pi)$ for a right fence (Fig. 2). Note that these are open intervals; a part cannot pass a fence with $\varphi = \pi/2$, and the vertical fences at $\varphi = 0$ and π have no effect.

Given a polygonal part P, the *fence-design problem* is to construct a sequence $\Phi = \langle \varphi_1, \ldots, \varphi_k \rangle$ of fence angles so that, regardless of its initial orientation, P always leaves the last fence in a unique final orientation. We refer to Φ as a *valid fence sequence*.

Related Work. Goldberg [10] showed that it is possible to reorient any polygon in the plane from an unknown initial orientation to a unique final one by a fixed sequence of normal pushes with a straight fence. Each push is in a direction orthogonal to the length of the fence, and the reorientation of the fence between pushes is independent of the orientation of the part. Goldberg's $O(n^2 \log n)$-time algorithm computes a shortest sequence of $O(n^2)$ pushes, although he conjectured that the shortest push sequence was in fact linear. Chen and Ierardi [6] proved the conjecture.

The fence-design problem is closely related to orienting a part by pushing. The main difference is that the fences can no longer be reoriented arbitrarily. Namely, if the belt moves downward, then any fence encountered by a part will affect a push with positive vertical component; thus, only half of the possible push directions are available at a given time. This restriction makes the problem significantly more complicated, and requires a very different approach.

The fence-design problem was first considered by Peshkin and Sanderson [14]. They introduce a graph representation of the state space, in which each node represents a set of possible orientations of the part at a given moment and in which certain paths represent successful fence designs. They discretize the set of allowable fence angles, so their solution is not complete, in the sense that when their algorithm fails to design a feeder, there may still exist a solution that requires an angle not in their set. Later, Brokowski et al. [5] suggested the addition of a curved tail to each fence so that the part

[2] In this paper all angles are represented in the range $[0, 2\pi)$.

is constrained to be in one of $O(m)$ orientations after it leaves a fence. This led to the complete algorithm of Wiegley et al. [16], which allows for all possible fence angles and is guaranteed to find a sequence of fences to orient a part if one exists. However, the running time of their algorithm is exponential, and they conjecture that a polynomial-time fence-design algorithm exists for any polygonal part. Roughly speaking, they construct a *state graph* in which each node corresponds to a subset of states in which P can be at a given moment. Assuming that P is currently in one of the states in the subset I and there is a fence f so that the subset of states in which P can be after it is pushed by f is contained in J, then they add the edge (I, J) to the graph. The fence-design problem then reduces to finding a path in this graph. By taking advantage of certain monotonicity properties of fences it can be shown that it suffices to consider only $O(m^2)$ subsets of states instead of all subsets. Using this observation and a general result by Eppstein [8], an $O(m^4)$-time algorithm for fence design can be obtained. Berretty et al. [3] construct a related graph of size $O(m^3)$, thereby improving the design time to $O(m^3)$. Actually, the running time of their algorithm is $O(m+n^3)$, where n is the number of "stable" edges in P (see Section 2 for the definition), which can be much smaller than m. The interested readers are referred to [2] and the references therein for many other part-feeder problems that have been studied.

Our Results. The main result of this paper is an $O(m + n^2 \log^3 n)$ algorithm for the fence-design problem, where n is the number of stable edges of the convex m-gon P. By exploiting the geometry of the state graph, we show that the graph can be represented implicitly, as the union of a family of complete bipartite subgraphs, and a desired path can be found in this implicit representation. Although similar techniques have been used in the past in other contexts [1], [9], we believe this is the first application of this approach for a manipulation problem. We believe that our approach is versatile and will find other applications in those part-manipulation problems that can be formulated as searching in a graph.

2. Fence Function and Fence Graph

We first describe the push mechanism and the motion of the part when it encounters a fence. Then we explain how the fence-design problem can be formulated as a graph-searching problem.

Radius and Push Functions. Let c be the center of mass of P. We attach a local frame of reference to P with c as its origin. The *radius function* $\rho\colon \mathbb{S}^1 \to \mathbb{R}^+$ of P is defined as follows: Let ℓ be the line tangent to P whose inward (toward P) normal points in the direction α, as measured in the frame attached to P. Then $\rho(\alpha)$ is the (shortest) distance from c to ℓ; see Fig. 3.

The radius function is piecewise sinusoidal, and its local maxima and minima occur when a ray originating from a contact point and normal to ℓ passes through c. Let γ_i and ε_i be the angles where ρ attains its local maxima and minima, respectively, ordered in the counterclockwise direction with $\gamma_0 < \varepsilon_0 < \gamma_1 < \ldots < \varepsilon_{n-1}$. An edge e of P is called *stable* if the normal direction of e is a local minima of ρ; if e is stable, then the

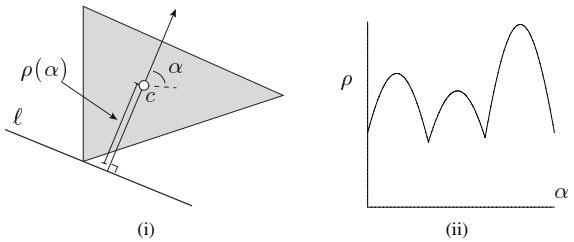

Fig. 3. The radius function $\rho : \mathbb{S}^1 \to \mathbb{R}^+$: (i) α is the direction of the ray normal to ℓ that passes through c; (ii) ρ is a piecewise-sinusoidal function of α.

ray from c normal to e intersects e. Let $\langle e_0, \ldots, e_{n-1} \rangle$ be the sequence of n stable edges of P; they can be computed in time $O(m)$.

Now, suppose that we push P with a fence in the direction normal to the fence. As described in [11], during the push, P first rotates in the direction that decreases ρ until a stable edge of P aligns with the fence, and then P translates in the direction normal to the fence. The *push function* p: $\mathbb{S}^1 \to \mathbb{S}^1$ for P is defined as follows: If a fence applies a normal push to P in the direction α, then $p(\alpha)$ is the final orientation of the fence, where both α and $p(\alpha)$ are measured in P's coordinate frame; see Fig. 4. That is,

$$p(\alpha) = \begin{cases} \varepsilon_i & \text{if} \quad \gamma_i < \alpha < \gamma_{i+1}, \\ \gamma_i & \text{if} \quad \alpha = \gamma_i. \end{cases} \tag{1}$$

We avoid pushing in the directions γ_i, as these are unstable equilibria.

Note that the belt does not translate the part in a direction normal to the fence; in general, the motion of the part when pushed in such a fashion is quite complicated, even unpredictable [12]. However, the component of the push force along the length of the fence is due to friction alone, so our assumption that the contact is frictionless implies that the force of the push felt by the part is always orthogonal to the fence. Thus the push function p correctly predicts the resulting orientation of P when it encounters a fence on the belt.

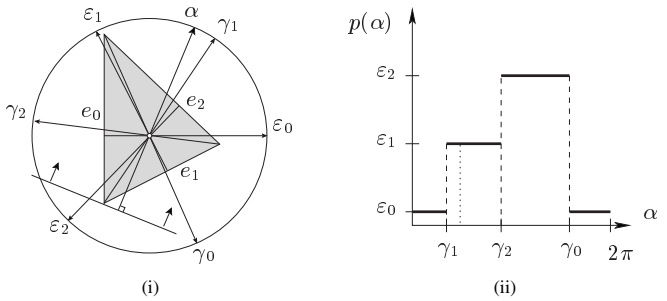

Fig. 4. The push function $p : \mathbb{S}^1 \to \mathbb{S}^1$. (i) A push in direction α with $p(\alpha) = \varepsilon_1$. (ii) $p(\alpha)$ is a step function. ε_i and γ_i are the local minima and maxima of ρ.

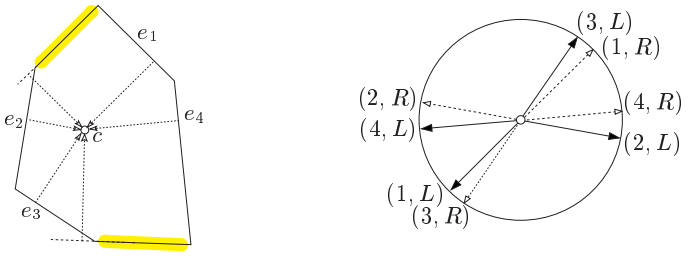

Fig. 5. A hexagon P with four stable edges (shaded edges are unstable) and the stable orientations of P in its local frame.

Fence Function. Finally, we turn to the fence function, which describes the action of a fence, with fence angle φ, on the orientation of the part. Recall that once P is aligned with a fence, the curved tail reorients P so that the aligned edge faces the left side of the belt if $\varphi \in (0, \pi/2)$, or the right side of the belt if $\varphi \in (\pi/2, \pi)$. This ensures that P is in one of only $2n$ orientations as it travels between fences. We denote by (i, L) and (i, R) the orientation of P with stable edge e_i parallel to and facing the left and right walls of the belt, respectively. Note that as viewed in the frame of reference attached to P, the $(+x)$-axis of the belt points in the direction ε_i when P is in orientation (i, L), and in the direction $\varepsilon_i + \pi$ when P is in orientation (i, R). See Fig. 5.

Suppose P is initially in orientation (i, L). When P encounters a fence with fence angle φ, it feels a push in direction $\varepsilon_i + \varphi$ in its own frame. If $\gamma_j < \varepsilon_i + \varphi < \gamma_{j+1}$, then the push function $p(\varepsilon_i + \varphi) = \varepsilon_j$ dictates that edge e_j aligns with the fence. If, on the other hand, P is initially aligned to the right, say in orientation (i, R), then it feels a push in the $\varepsilon_i + \varphi + \pi$ direction, and the final orientation depends on $p(\varepsilon_i + \varphi + \pi)$. See Fig. 6.

The equivalent action in planar pushing is to reorient the fence by θ. The difference here is that, although values of θ can in general lie anywhere in $[0, 2\pi)$, we are restricted to $\theta = \varphi$ or $\theta = \varphi + \pi$ for any particular push, where $\varphi \in (0, \pi)$. Thus, only half of the possible reorientations are available at a given time, which implies that some orientations

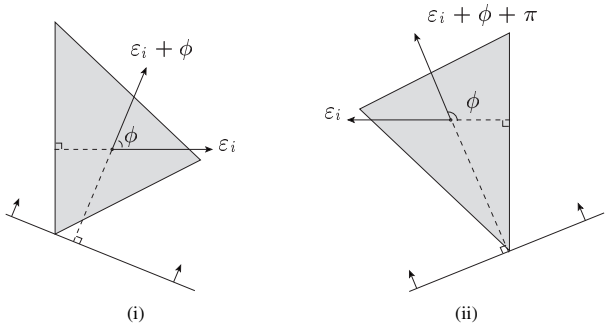

Fig. 6. The action of a fence: (i) If P is initially in orientation (i, L), it feels a push in direction $\alpha = \varepsilon_i + \varphi$. (ii) If P is initially in orientation (i, R), it feels a push in direction $\alpha = \varepsilon_i + \varphi + \pi$.

are unattainable from others. It is precisely this fact that makes fence design harder than pushing.

Definition 2.1. Given polygon P with n stable edges, define its *fence function*

$$\mathcal{F}\colon \{0,\ldots,n-1\}\times\{L,R\}\times(0,\pi)\to\{0,\ldots,n-1\}\times\{L,R\}$$

so that if P is initially in position (i,s) when it encounters a fence with fence angle $\varphi\in(0,\pi)$, the resulting orientation is $\mathcal{F}(i,s,\varphi)=(j,t)$, where

$$\varepsilon_j=\begin{cases}p(\varepsilon_i+\varphi) & \text{if}\quad s=L,\\ p(\varepsilon_i+\varphi+\pi) & \text{if}\quad s=R,\end{cases}\tag{2}$$

and

$$t=\begin{cases}L & \text{if}\quad \varphi\in(0,\pi/2),\\ R & \text{if}\quad \varphi\in(\pi/2,\pi).\end{cases}\tag{3}$$

The following lemma was observed in [3]:

Lemma 2.2. *$\mathcal{F}$ is monotonic, i.e., $i_1\le i_2\le i_3$ implies that $\mathcal{F}(i_1,s,\varphi)\le\mathcal{F}(i_2,s,\varphi)\le\mathcal{F}(i_3,s,\varphi)$, for any pair $(s,\varphi)\in\{L,R\}\times(0,\pi)$.*

The monotonicity property was crucial for the algorithm in [3], and it will be crucial for ours as well.

The fence function can be represented by a family of n partitions of $\mathbb{S}^1$: For each stable edge e_i of P, let C_i denote the unit circle marked with the intervals $\chi_{ij}=(\gamma_j-\varepsilon_i,\gamma_{j+1}-\varepsilon_i)$, for $0\le j<n$; imagine rotating the circle in the clockwise direction by ε_i. See Fig. 7. The intervals χ_{ij} on C_i specify all triples s,t,φ for which $\mathcal{F}(i,s,\varphi)=(j,t)$. More precisely, the upper semicircle of C_i represents the orientation (i,L) of P and the lower represents (i,R), in the following sense. Suppose $\theta\in(0,\pi)\cap\chi_{ij}$. Then $\gamma_j<\varepsilon_i+\theta<\gamma_{j+1}$, and, by (1), $p(\varepsilon_i+\theta)=\varepsilon_j$; thus, the fence at angle θ takes the left orientation (i,L) to one of the orientations (j,t). Specifically, if $\theta\in(0,\pi/2)$, then $\mathcal{F}(i,L,\theta)=(j,L)$, while if $\theta\in(\pi/2,\pi)$, then $\mathcal{F}(i,L,\theta)=(j,R)$. Similarly, if $\theta\in(\pi,2\pi)\cap\chi_{ij}$, then there exists a fence that takes the right orientation (i,R) to one of the orientations (j,t). Of course, θ itself is not a valid fence angle, but $\varphi=\theta-\pi$ is, and $p(\varepsilon_i+\varphi+\pi)=\varepsilon_j$. Thus, by (2) and (3), a fence at angle $\theta-\pi$ takes the orientation (i,R) to (j,t) where $t=L$ if $\theta\in(\pi,3\pi/2)$, and $t=R$ if $\theta\in(3\pi/2,2\pi)$.

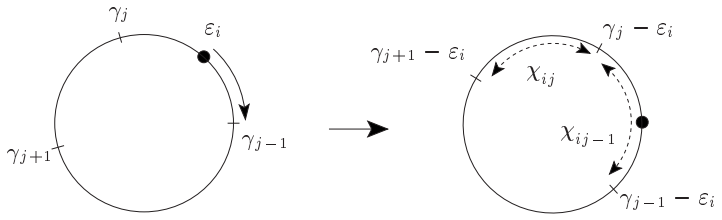

Fig. 7. The fence circle C_i is obtained by rotating the circle clockwise by ε_i.

For example, consider the right circle in Fig. 7. χ_{ij} intersects the first and second quadrants, so there exist fence angles φ_1 and φ_2 with $\mathcal{F}(i, L, \varphi_1) = (j, L)$ and $\mathcal{F}(i, L, \varphi_2) = (j, R)$. Since χ_{ij} does not intersect the lower semicircle, $\mathcal{F}(i, R, \varphi) \neq (j, t)$ for any $\varphi \in (0, \pi)$ or $t \in \{L, R\}$. On the other hand, χ_{ij-1} does intersect the fourth quadrant, so $\mathcal{F}(i, R, \theta - \pi) = (j - 1, R)$ for some $\theta \in \chi_{ij-1} \cap (3\pi/2, 2\pi)$.

Note that when choosing φ for our push plans, we avoid the discrete set of values $\gamma_j - \varepsilon_i, \pi + \gamma_j - \varepsilon_i$, for $0 \leq i, j < n$, since each can lead to an unstable equilibrium.

Lemma 2.3. *There exists a family $\Theta \subset (0, \pi)$ of $O(n^2)$ orientations so that if there is a valid fence sequence $\Phi = \langle \varphi_1, \ldots, \varphi_k \rangle$, then there is another valid fence sequence $\Phi' = \langle \varphi'_1, \ldots, \varphi'_k \rangle$ in which $\varphi'_i \in \Theta$ for all $i \leq k$.*

Proof. Let $\mathcal{I} = \langle I_1, \ldots, I_u \rangle$ be the sequence of open intervals in

$$(0, \pi) \backslash (\{\gamma_j - \varepsilon_i \pmod{\pi} \mid 0 \leq i, j < n\} \cup \{\pi/2\}).$$

By the above discussion, for any pair $(i, s) \in \{0, \ldots, n-1\} \times \{L, R\}$ and for any $1 \leq j \leq u$, $\mathcal{F}(i, s, \varphi)$ is the same for all $\varphi \in I_j$. Let θ_j be the midpoint of (the closure of) the interval I_j. Then $\Theta = \theta_1 < \theta_2 < \cdots < \theta_u$ is the desired family of fence orientations. Indeed, suppose there is a valid fence sequence $\Phi = \langle \varphi_1, \ldots, \varphi_k \rangle$. If φ_l lies in $I_{j_l} \in \mathcal{I}$, then $\Phi' = \langle \theta_{j_1}, \ldots, \theta_{j_k} \rangle$ is also a valid fence sequence since $\mathcal{F}(i, s, \varphi_l) = \mathcal{F}(i, s, \theta_{j_l})$ for all $0 \leq i < n$, $s, t \in \{L, R\}$, and $1 \leq l \leq k$. This completes the proof of the lemma. □

Fence Graph. Next, we show how the fence-design problem can be formulated as a connectivity problem in a graph. The graph defined here is slightly different from the one constructed in [3]. Not only does the modified definition provide an almost identical $O(m + n^3)$ solution to the fence-design problem, it also allows us to *compress* the graph to obtain a faster $O(m + n^2 \log^3 n)$ algorithm. By Lemma 2.3, we have a family $\Theta = \theta_1 < \cdots < \theta_u$ of orientations, with $\theta_1 \in (0, \pi/2)$ and $\theta_u = (\pi/2, \pi)$, which are sufficient to represent all valid fence sequences. From here on, we assume that all fence angles are selected from Θ.

Definition 2.4. Given a polygon P with n stable edges and fence function $\mathcal{F}$, we define the *fence graph* G as follows:

- The nodes of G are sets of the form $[i, j]^s = \{(i, s), (i + 1, s), \ldots, (j, s)\}$, where $0 \leq i, j < n$, and $s = L$ or R. Note that $[i, j]^s \neq [j, i]^s$.
- For $i' \neq j'$, there is an edge $e = ([i, j]^s, [i', j']^t)$ whenever there exists a fence angle $\varphi \in \Theta$ such that $\mathcal{F}(i, s, \varphi) = (i', t)$ and $\mathcal{F}(j, s, \varphi) = (j', t)$. We refer to these edges as *non-sink* edges.
- There is an edge $e = ([i, j]^s, [i', i']^t)$ if there exists a fence angle $\varphi \in \Theta$ with $\mathcal{F}(k, s, \varphi) = (i', t)$ for *all* $i \leq k \leq j$. We refer to these edges as *sink* edges.

In both cases we refer to φ as the *witness* of edge e and denote it by $\omega(e)$.

Lemma 2.5. *Every node in the fence graph G has out-degree $O(n)$, and the outgoing edges from a vertex can be computed in $O(n)$ time.*

Proof. By Definition 2.4, the neighbors of $[i, j]^s$ can be determined by overlaying the fence circles C_i and C_j. Indeed, if $[i', j']^t$ is a neighbor of $[i, j]^s$, then $\chi_{ii'} \cap \chi_{jj'} \neq \emptyset$. The overlay of C_i and C_j induces a partition of the unit circle into $2n$ intervals whose endpoints are the endpoints of intervals in C_i and C_j, i.e., $\gamma_k - \varepsilon_i$, $\gamma_k - \varepsilon_j$, for $0 \leq k < n$. The same argument as in the proof of Lemma 2.3 implies that each of these intervals corresponds to at most one edge of G, so there are at most $2n$ outgoing edges from the node $[i, j]^s$. Since we can overlay C_i and C_j in $O(n)$ time, the outgoing edges of $[i, j]^s$ can be computed in $O(n)$ time. □

Lemma 2.6. *Every valid fence sequence of P corresponds to a path in the fence graph G from a node of the form $[i, i-1]^s$ to a sink node $[j, j]^t$, and vice versa.*

Proof. The monotonicity of $\mathcal{F}$ (see Lemma 2.2) and Definition 2.4 imply that if G has an edge $e = ([i, j]^s, [i', j']^t)$, then $\mathcal{F}(k, s, \omega(e)) \in [i', j']^t$ for all $k \in [i, j]$. Let $e_1, \ldots, e_k$ be a path in G from $[i, i-1]^s$ to $[j, j]^t$. For $1 \leq i \leq k$, let $\varphi_i = \omega(e_i)$, and let $\varphi_0 = \theta_1$ (resp. $\varphi_0 = \theta_u$) if $s = L$ (resp. $s = R$). Then $\Phi = \langle \varphi_0, \varphi_1, \ldots, \varphi_k \rangle$ is a valid fence sequence. Indeed, the fence in orientation φ_0 ensures that the orientation of P after it passes through this fence is in the interval $[i, i-1]^s$. By the definition of the edge witnesses, the final orientation of P is (j, t), regardless of its initial orientation.

The proof of the converse, that every valid fence sequence corresponds to a path of the above form, is a bit more subtle. We adapt a result by Eppstein [8] relating some of the feeder-design problems to finite monotonic automata. For the sake of completeness, we sketch the proof in the Appendix. □

We can compute the stable of edges of P in $O(m)$ time, and, by Lemmas 2.5 and 2.6, we can compute a desired path in G in $O(n^3)$ time. Hence, we obtain the following result, which gives a simpler proof of the same result in [3].

Theorem 2.7. *Given a convex polygonal part P with m vertices and n stable edges, a fence design can be constructed in $O(m + n^3)$ time.*

Sink Edges. Before proceeding with the compression of the fence graph, we consider the sink edges in more detail, i.e., edges of the form $([i, j]^s, [i', i']^t)$. Since we are dealing with circular intervals, $[i', i']^t$ poses an ambiguity—whether it represents the singleton orientation (i', t) or the set of *all* orientations aligned with side t. We therefore need to take extra care with the nodes $[i', i']^t$ when we compress G in the next section because we want to ensure that they represent sink nodes, i.e., singletons. To circumvent this problem, we first dispose of a class of parts for which there is a trivial solution, and then prove a property of the fence graph which applies to the remaining parts. In the next two lemmas all arithmetic and logical operations on angles are performed modulo 2π.

The majority of polygonal parts have the following property:

$$(*) \quad \varepsilon_{j+1} - \varepsilon_j \leq \gamma_i - \gamma_{i+1} \qquad \text{for all} \quad 0 \leq i, j < n.$$

Parts that do *not* have property $(*)$ have a relatively large gap between two successive stable edges, which allows for the following simple two-fence solution described in Lemma 2.8; see Fig. 8 for an example.

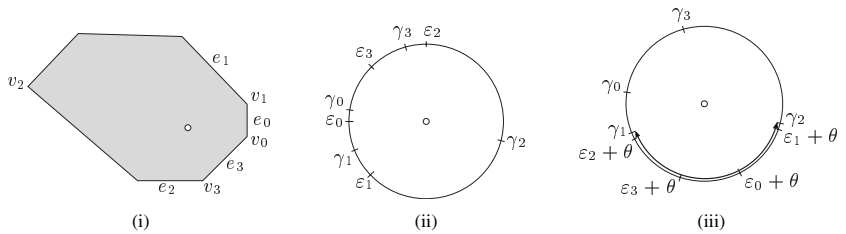

Fig. 8. (i) A part that does not satisfy $(*)$, with stable edges and vertices labeled. (ii) The angles corresponding to local extrema of the radius function. (iii) In this case, $\varepsilon_2 - \varepsilon_1 > \gamma_1 - \gamma_2$, so there exists θ such that $p(\varepsilon_k + \theta) = \varepsilon_1$ for all k, as $\varepsilon_k + \theta \in (\gamma_1, \gamma_2)$ for all k.

Lemma 2.8. *Two fences are sufficient to orient any part that does not satisfy* $(*)$.

Proof. For some i and j, we have $\varepsilon_{j+1} - \varepsilon_j > \gamma_i - \gamma_{i+1}$. Then there exists a $\theta \in [0, 2\pi)$ so that $\gamma_i < \varepsilon_{j+1} + \theta < \varepsilon_j + \theta < \gamma_{i+1}$; see Fig. 8. Since $\varepsilon_k \in (\varepsilon_{j+1}, \varepsilon_j)$ for all k, we have $\varepsilon_k + \theta \in (\varepsilon_{j+1} + \theta, \varepsilon_j + \theta) \subseteq (\gamma_i, \gamma_{i+1})$. Therefore, by (1), $p(\varepsilon_k + \theta) = \varepsilon_i$ for all k. A reorientation by θ can be accomplished with exactly two fences, as follows. If $\theta \in (0, \pi)$, then we set $\varphi_1 = \pi/4$ and $\varphi_2 = \theta$; if $\theta \in (\pi, 2\pi)$, then we set $\varphi_1 = 3\pi/4$ and $\varphi_2 = \theta - \pi$. The first fence aligns the part with the left (resp. right) side of the belt if $\theta \in (0, \pi)$ (resp. $\theta \in (\pi, 2\pi)$), and the second fence orients the part in direction ε_i. □

For the rest of this paper we assume that P satisfies $(*)$. The next lemma proves a property of sink edges that facilitates the compression of the fence graph in the next section.

Lemma 2.9. *Suppose P satisfies* $(*)$, $\mathcal{F}(i, s, \varphi) = \mathcal{F}(j, s, \varphi) = (i', t)$ *for some* $\varphi \in (0, \pi)$, *and* $\gamma_{i'+1} - \gamma_{i'} \leq \pi$. *Then there is a sink edge* $[i, j]^s \to [i', i']^t$ *in G if and only if* $\varepsilon_j - \varepsilon_i \leq \pi$.

Proof. Set $\theta = \varphi$ if $s = L$ and $\theta = \varphi + \pi$ if $s = R$. Suppose first that $\varepsilon_j - \varepsilon_i \leq \pi$. Since $\varepsilon_i + \theta, \varepsilon_j + \theta \in (\gamma_{i'}, \gamma_{i'+1})$, $\varepsilon_j - \varepsilon_i \leq \pi$, and $\gamma_{i'+1} - \gamma_{i'} \leq \pi$, we must have $\gamma_{i'} < \varepsilon_i + \theta < \varepsilon_j + \theta < \gamma_{i'+1}$; see Fig. 9(i). This implies that $\mathcal{F}(k, s, \varphi) = (i', t)$ for all $i \leq k \leq j$, so G has a sink edge $[i, j]^s \to [i', i']^t$.

On the other hand, if $\varepsilon_j - \varepsilon_i > \pi$, then $\varepsilon_i + \theta < \gamma_{i'+1} < \gamma_{i'} < \varepsilon_j + \theta$; Fig. 9(ii). If $[i, j]^s \to [i', i']^t$ is an edge of G, i.e., $p(\varepsilon_k + \theta) \in (\gamma_{i'}, \gamma_{i'+1})$, for all $i \leq k \leq j$, then there must be some k with $i \leq k < j$ such that $\varepsilon_k + \theta < \gamma_{i'+1} < \gamma_{i'} < \varepsilon_{k+1} + \theta$; but this implies that $\varepsilon_{k+1} - \varepsilon_k > \gamma_{i'} - \gamma_{i'+1}$, violating $(*)$. □

3. Fence Graph Compression

In this section we describe how to compute a compressed representation of the fence graph for a polygon that satisfies $(*)$. The compressed representation needs considerably

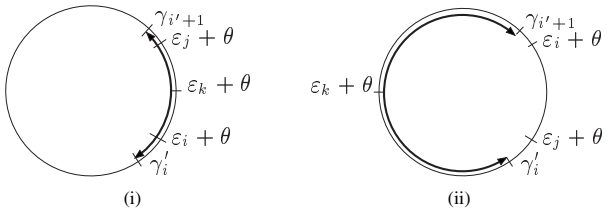

Fig. 9. For parts which satisfy $(*)$, the sink edge $[i, j]^s \to [i', i']^t$ exists if and only if $\varepsilon_j - \varepsilon_i \leq \pi$. (i) If $\varepsilon_j - \varepsilon_i \leq \pi$, then $\varepsilon_k + \theta \in (\gamma_{i'}, \gamma_{i'+1})$ for all $i \leq k \leq j$, (ii) but if $\varepsilon_j - \varepsilon_i > \pi$, at least one $\varepsilon_k + \theta \in (\gamma_{i'+1}, \gamma_{i'})$ as P satisfies $(*)$.

less space but still allows us to compute a path between two vertices quickly. We first explain the compression scheme in general and then apply it to the fence graph.

3.1. *Bipartite Clique Covers*

Given a directed graph $G = (V, E)$, a *bipartite clique cover* of G is a collection

$$\mathcal{G} = \{(A_1, B_1), \ldots, (A_k, B_k)\}$$

such that

(i) $A_i, B_i \subseteq V$,
(ii) $E_i = A_i \times B_i \subseteq E$,
(iii) $E = \bigcup_i E_i$,
(iv) $E_i \cap E_j = \emptyset$, if $i \neq j$.

Conditions (i) and (ii) imply that each bipartite clique $(A_i \cup B_i, E_i)$ is a complete bipartite subgraph of G, while (iii) and (iv) imply that every edge of E is represented exactly once. The *size* of $\mathcal{G}$ is $|\mathcal{G}| = \sum_{i=1}^{k} |A_i| + |B_i|$.

Given a bipartite clique cover $\mathcal{G} = \{(A_1, B_1), \ldots, (A_k, B_k)\}$ of G, we can generate a *compressed representation* $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ of G by setting

$$\widetilde{V} = V \cup \{1, \ldots, k\}, \quad \text{and}$$
$$\widetilde{E} = \bigcup_{i=1}^{k} \{(a, i) \mid a \in A_i\} \cup \{(i, b) \mid b \in B_i\}.$$

See Fig. 10.

There is an edge $(a, b) \in A_i \times B_i$ in G if and only if $(a, i), (i, b) \in \widetilde{E}$, which immediately implies the following.

Lemma 3.1. *There is a path of length l in G from a vertex μ to another vertex ν if and only if there is a path of length $2l$ from μ to ν in $\widetilde{G}$.*

We now present a method for generating a bipartite clique cover for $G = (V, E)$. Suppose that we have a family $\mathcal{N} = \{B_1, \ldots, B_k\}$ of subsets $B_i \subseteq V$, with the property

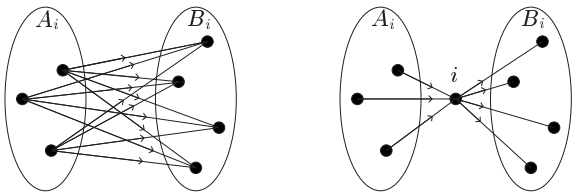

Fig. 10. G and $\widetilde{G}$ store the same connectivity information, but $\widetilde{G}$ has fewer edges.

that the set of neighbors $N(\mu) = \{\nu \in V \mid (\mu, \nu) \in E\}$ of a vertex μ can be expressed as a disjoint union of subsets of $\mathcal{N}$. That is, for each $\mu \in V$, there is a subfamily $\mathcal{N}_\mu \subseteq \mathcal{N}$ such that

(i) $N(\mu) = \bigcup_{B_i \in \mathcal{N}_\mu} B_i$, and
(ii) $B_i \cap B_j = \emptyset$ if $B_i, B_j \in \mathcal{N}_\mu$ and $i \neq j$.

We refer to $\mathcal{N}$ as a family of *canonical* subsets. For each $1 \leq i \leq k$, define

$$A_i = \{\mu \in V \mid B_i \in \mathcal{N}_\mu\},$$

and set $\mathcal{G} = \{(A_1, B_1), \ldots, (A_k, B_k)\}$.

Lemma 3.2. *$\mathcal{G}$ is a bipartite clique cover for G.*

Proof. It is clear that (A_i, B_i) is a complete bipartite subgraph of G. Indeed, $A_i, B_i \subseteq V$, and $(\mu, \nu) \in A_i \times B_i$ implies that $\nu \in B_i \subseteq N(\mu)$, i.e., $(\mu, \nu) \in E$. Conversely, if $(\mu, \nu) \in E$, then there exists a subset $B_i \in \mathcal{N}_\mu$ such that $\nu \in B_i$, implying $(\mu, \nu) \in A_i \times B_i$. Finally, if there are two indices i, j such that $(\mu, \nu) \in A_i \times B_i \cap A_j \times B_j$, then $\nu \in B_i \cap B_j$ and both $B_i, B_j \in \mathcal{N}_\mu$, contradicting the assumption (ii) of $\mathcal{N}_\mu$. Hence, (i)–(iv) are satisfied, and $\mathcal{G}$ is a bipartite clique cover of G. □

3.2. *Compressing the Fence Graph*

We now describe how we compress the fence graph G for a polygon that satisfies (∗), i.e., $\varepsilon_{j+1} - \varepsilon_j \leq \gamma_i - \gamma_{i+1}$ for all i, j. Recall that the nodes of G are of the form $[i, j]^s = \{(i, s), \ldots, (j, s)\}$, where (i, s) is the orientation of the part with stable edge e_i aligned with side s of the belt, and if there is an edge $[i, j]^s \to [i', j']^t$, then $\mathcal{F}(i, s, \varphi) = (i', t)$ and $\mathcal{F}(j, s, \varphi) = (j', t)$ for some $\varphi \in (0, \pi)$. Recall from the discussion in Section 2 that the sink edges need to be handled carefully. If there is an index i' such that $\gamma_{i'+1} - \gamma_{i'} > \pi$ (there is at most one such i'), we check in $O(1)$ time for each vertex $[i, j]^s$ of G and for each pair $s, t \in \{L, R\}$ whether $[i, j]^s \to [i', i']^t$. If the answer is yes, we add the pair $([i, j]^s, [i', i']^t)$ to $\mathcal{G}$. There are only $O(n^2)$ such pairs and we spend $O(n^2)$ time on them. Therefore, from now on we are interested in computing a compressed representation of only those neighbors of a vertex that correspond to either a non-sink edge or a sink edge with $\gamma_{i'+1} - \gamma_{i'} \leq \pi$.

We first describe how we compute the family of canonical subsets and then show how to represent these neighbors of a vertex as a disjoint union of canonical subsets.

Constructing Canonical Subsets. We first give an overview of the construction and postpone description of the necessary data structures until Section 4. Let $[i : j]$ denote the set of integers $\{i, i+1, \ldots, j\}$, mod n.

The construction of $\mathcal{N}$ proceeds in two phases. In the first phase we construct a family $\mathcal{I}$ of $O(n)$ canonical subsets of $[0 : n-1]$ with the following properties:

(F.1) Each $I_a \in \mathcal{I}$ is of the form $I_a = [l_a : r_a]$.
(F.2) $|\mathcal{I}| = \sum_{I_a} n_a = O(n \log n)$, where $n_a = r_a - l_a + 1$.
(F.3) For any interval $A = (\alpha_1, \alpha_2)$ of push angles, there is a subfamily $\mathcal{I}_A \subseteq \mathcal{I}$ of $O(\log n)$ disjoint subsets such that $p(A) = \bigcup_{I_a \in \mathcal{I}_A} \bigcup_{i \in I_a} \varepsilon_i$.

We show in Section 4 how we compute in $O(n \log n)$ time the family $\mathcal{I}$, and in $O(\log n)$ time the subfamily $\mathcal{I}_A$ for an arbitrary angular interval $A \subseteq \mathbb{S}^1$.

For each pair of canonical subsets $I_a, I_b \in \mathcal{I}$, we define the function $V_{ab} : \mathbb{S}^1 \to 2^{I_a \times I_b}$ as follows: For a given *shift* angle δ, $V_{ab}(\delta)$ is the set of pairs $(i, j) \in I_a \times I_b$ such that the angular intervals (γ_i, γ_{i+1}) and $(\gamma_j + \delta, \gamma_{j+1} + \delta)$ intersect on $\mathbb{S}^1$. We also define another function $\bar{V}_{ab}$, which is the same as V_{ab} except that we remove all nodes that would represent sink edges; that is, $\bar{V}_{ab}(\delta) = V_{ab}(\delta) \backslash \{[i, i] \mid 1 \leq i < n\}$.

The second phase of the algorithm constructs a family $\mathcal{N}^{ab}$ of canonical subsets of $I_a \times I_b$, for each pair $I_a, I_b \in \mathcal{I}$, with the following properties:

(S.1) $|\mathcal{N}^{ab}| = \sum_{N \in \mathcal{N}^{ab}} |N| = O(n_a n_b \log n)$.
(S.2) Given a shift angle δ, there are two subfamilies $\mathcal{N}^{ab}_\delta, \bar{\mathcal{N}}^{ab}_\delta \subseteq \mathcal{N}^{ab}$ of $O(\log n)$ disjoint canonical subsets each such that

$$V_{ab}(\delta) = \bigcup_{N \in \mathcal{N}^{ab}_\delta} N \quad \text{and} \quad \bar{V}_{ab}(\delta) = \bigcup_{\bar{N} \in \bar{\mathcal{N}}^{ab}_\delta} \bar{N}.$$

Again, we describe in Section 4 how to compute the families $\mathcal{N}^{ab}$.

Computing the Neighbors. With these families of canonical subsets in hand, we are now ready to compute the neighbor sets of each vertex in the fence graph G. For a vertex $[i, j]^L \in G$, let $N_L([i, j]^L)$ be the set of neighbors $[i', j']^L$ of $[i, j]^L$ such that either $i' \neq j'$, or $i' = j'$ and $\gamma_{i'+1} - \gamma_{i'} \leq \pi$. We refer to N_L as the set of *left neighbors*. Recall that if there is an edge $[i, j]^L \to [i', j']^L$ in E, then we must have $\mathcal{F}(i, L, \varphi) = (i', L)$ and $\mathcal{F}(j, L, \varphi) = (j', L)$ for some $\varphi \in (0, \pi/2)$. Furthermore, $[i, j]^L \to [i', i']^L$ with $\gamma_{i'+1} - \gamma_{i'} \leq \pi$ is a sink edge if and only if $\mathcal{F}(i, L, \varphi) = \mathcal{F}(j, L, \varphi) = (i', L)$ and $\varepsilon_j - \varepsilon_i \leq \pi$, by Lemma 2.9.

We compute $N_L([i, j]^L)$, the set of left neighbors of $[i, j]^L$, as follows. Set $A_i = (\varepsilon_i, \varepsilon_i + \pi/2)$, $A_j = (\varepsilon_j, \varepsilon_j + \pi/2)$, and $\delta_{ij} = \varepsilon_i - \varepsilon_j$. Using the family $\mathcal{I}$ of canonical subsets, we compute subfamilies $\mathcal{I}_{A_i}$ and $\mathcal{I}_{A_j}$. For each pair $I_a \in \mathcal{I}_{A_i}$, $I_b \in \mathcal{I}_{A_j}$, we compute the subfamily $\mathcal{N}^{ab}_\delta \subseteq \mathcal{N}^{ab}$, representing the pairs $[i', j'] \in V_{ab}(\delta_{ij})$ if $\varepsilon_j - \varepsilon_i \leq \pi$; and the subfamily $\bar{\mathcal{N}}^{ab}_\delta \subseteq \mathcal{N}^{ab}$, representing the non-sink pairs $[i', j'] \in \bar{V}_{ab}(\delta_{ij})$ if $\varepsilon_j - \varepsilon_i > \pi$.

Lemma 3.3. *The set of left neighbors of a left node* $[i, j]^L \in V$ *is*

$$N_L([i,j]^L) = \begin{cases} \bigcup_{I_a \in \mathcal{I}_{A_i}} \bigcup_{I_b \in \mathcal{I}_{A_j}} \bigcup_{N \in \mathcal{N}_\delta^{ab}} N & \text{if} \quad \varepsilon_j - \varepsilon_i \le \pi, \\ \bigcup_{I_a \in \mathcal{I}_{A_i}} \bigcup_{I_b \in \mathcal{I}_{A_j}} \bigcup_{N \in \bar{\mathcal{N}}_\delta^{ab}} N & \text{if} \quad \varepsilon_j - \varepsilon_i > \pi. \end{cases}$$

Proof. We assume that $\varepsilon_j - \varepsilon_i \le \pi$. Since P satisfies $(*)$, by Lemma 2.9, the left-neighbor set of $[i, j]^L$ is the set of all $[i', j']^L$ such that $\chi_{ii'} \cap \chi_{jj'} \cap [0, \pi/2] \neq \emptyset$ in the overlay of the fence circles C_i and C_j.

The family $\mathcal{I}_{A_i}$ partitions the first quadrant of C_i into $O(\log n)$ intervals, and similarly for C_j. For each pair $I_a \in \mathcal{I}_{A_i}$ and $I_b \in \mathcal{I}_{A_j}$, we want to list the pairs $[i', j']$ such that $(\gamma_{i'} - \varepsilon_i, \gamma_{i'+1} - \varepsilon_i)$ and $(\gamma_{j'} - \varepsilon_j, \gamma_{j'+1} - \varepsilon_j)$ intersect for some $i' \in I_a$ and $j' \in I_b$, which is equivalent to saying that $(\gamma_{i'}, \gamma_{i'+1})$ and $(\gamma_{j'} + (\varepsilon_i - \varepsilon_j), \gamma_{j'+1} + (\varepsilon_i - \varepsilon_j))$ overlap. This is equivalent to determining $V_{ab}(\delta)$, where $\delta = \varepsilon_i - \varepsilon_j$, which we can express as the disjoint union of $O(\log(m_a m_b))$ canonical subsets in the subfamily $\mathcal{N}_\delta^{ab} \subseteq \mathcal{N}^{ab}$. The argument for the case $\varepsilon_j - \varepsilon_i > \pi$ is the same, except that $[i, j]$ does not have any outgoing sink edges, and we use $\bar{V}_{ab}$ and $\bar{\mathcal{N}}_\delta^{ab}$. □

In the preceding argument, we focused on edges $[i, j]^s \to [i', j']^t$ with $s = t = L$. The three other cases can be handled similarly. For $s = L, t = R$, we set $A_i = (\varepsilon_i + \pi/2, \varepsilon_i + \pi)$; for $s = R, t = L$, we set $A_i = (\varepsilon_i + \pi, \varepsilon_i + 3\pi/2)$; and for $s = R, t = R$, we set $A_i = (\varepsilon_i + 3\pi/2, \varepsilon_i + 2\pi)$. We do the same for A_j in each case, and then follow the above procedure.

Theorem 3.4. *Given a convex* m*-gon* P *with* n *stable edges, we can determine in* $O(m + n^2 \log^3 n)$ *time whether a valid fence sequence exists for* P.

Proof. We first compute the families $\mathcal{I}$ and $\mathcal{N}^{ab}$ of canonical subsets of nodes of the fence graph G, as described above. Next, define the bipartite clique cover $\mathcal{G} = \{(A_1, B_1), \ldots, (A_k, B_k)\}$, where the B_i are the canonical subsets that make up the $\mathcal{N}^{ab}$. Then the compressed graph $\widetilde{G}$, described in Section 3.1, has $|\widetilde{V}| = |V| + k$ nodes and $|\widetilde{E}| = \sum_i |A_i| + |B_i|$ edges.

We initially added $O(n^2)$ pairs in $\mathcal{G}$ corresponding to the sink edges $[i, j]^s \to [i', i']^t$ such that $\gamma_{i'+1} - \gamma_{i'} > \pi$. Next, for each pair I_a, I_b, there are $O(n_a n_b)$ subsets in $\mathcal{N}^{ab}$, so the total number of canonical subsets, using (F.2), is

$$k = O(n^2) + \sum_{I_a, I_b \in \mathcal{I}} O(n_a n_b) = \sum_{I_a \in \mathcal{I}} O(n_a n \log n) = O\left(n^2 \log^2 n\right).$$

Hence $|\widetilde{V}| = O(n^2 \log^2 n)$.

Recall that $\mu \in A_i$ if and only if $B_i \in \mathcal{N}_\mu$, so $\sum_i |A_i| = \sum_{\mu \in V} |\mathcal{N}_\mu|$. For $\mu = [i, j]$, we first compute $\mathcal{I}_{A_i}$ and $\mathcal{I}_{A_j}$. Each contains $O(\log n)$ canonical subsets I_a, but only $O(\log n)$ pairs will overlap at a given shift. For each overlapping pair, we collect $O(\log n)$

canonical subsets from $\mathcal{N}^{ab}$, so $\mathcal{N}_\mu$ contains $O(\log^2 n)$ subsets total. Therefore

$$\sum_{i=1}^{k} |A_i| = \sum_{\mu \in V} |\mathcal{N}_\mu| = O(n^2 \log^2 n).$$

Moreover, by (S.1),

$$\sum_i |B_i| = \sum_{a,b} \sum_{N \in \mathcal{N}^{ab}} |N| = \sum_{a,b} O(n_a n_b \log(n_a n_b)) = O(n^2 \log^3 n).$$

Hence $|\widetilde{E}| = O(n^2 \log^3 n)$ and $\mathcal{G}$ can be computed in time $O(m + n^2 \log^3 n)$. Finally, once $\widetilde{G}$ is computed, the time to find a path in $\widetilde{G}$ from one of the nodes $[i, i-1]^s$ to a singleton node $[j, j]^t$ is $O(|\widetilde{V}| + |\widetilde{E}|)$, using a simple breadth-first search. □

4. Data Structures

In this section we present the data structures needed to generate the families of canonical subsets. In particular, we introduce two tree data structures, the *fence tree* and the *shift tree*, which generate the canonical families $\mathcal{I}$ and $\mathcal{N}^{ab}$, respectively.

Fence Tree. We describe the construction of a family $\mathcal{I}$ of $O(n)$ canonical subsets that satisfies properties (F.1)–(F.3). Let T be a minimum-height binary tree whose leaves store $\{\gamma_i \mid 0 \le i < n\}$. In order to guide the search, each internal node of T stores the value of the leftmost leaf of its right subtree. We associate the *canonical* interval $I_a = [l_a : r_a]$ with each node a of T, where $\gamma_{l_a}, \ldots, \gamma_{r_a}$ are the leaves in the subtree rooted at a. We set $\mathcal{I} = \{I_a \mid a \in T\}$. Since each index is stored in exactly one I_a at each level of T and the height of T is $O(\log n)$, we have $|\mathcal{I}| = \sum_a |I_a| = O(n \log n)$. Hence (F.1) and (F.2) hold.

Let $A = (\alpha_1, \alpha_2)$ be a push-angle interval. If $0 \in (\alpha_1, \alpha_2) \pmod{2\pi}$, we split A into two intervals $(\alpha_1, 2\pi]$ and $[0, \alpha_2)$, and work with each of them separately, so we assume that $0 \notin (\alpha_1, \alpha_2)$. Recall that $p(\alpha) = \varepsilon_i$ if and only if $\alpha \in (\gamma_i, \gamma_{i+1})$. Then $p(A) = \{\varepsilon_l, \ldots, \varepsilon_r\}$, where l and r are the maximum values for which $\gamma_l \le \alpha_1$ and $\gamma_r < \alpha_2$.

In order to compute $\mathcal{I}_A$, we regard T as a one-dimensional range tree [7], and query T with α_1 and α_2 to determine l and r. We then report the set of indices $[l : r]$ as a disjoint union of $O(\log n)$ canonical intervals as follows: Let u be the least common ancestor in T of the leaves z_l and z_r storing γ_l and γ_r, respectively. For any node on the path from u to z_l (resp. z_r), if its right (resp. left) child v is not on the path, we add $[l_v : r_v]$ to $\mathcal{I}_A$. Since all the intervals we add to $\mathcal{I}_A$ lie along two paths of T, $|\mathcal{I}_A| = O(\log n)$.

Lemma 4.1. *Given a polygon P with n stable edges, we can construct in $O(n \log n)$ time a family $\mathcal{I}$ of $O(n)$ canonical subsets that satisfy* (F.1)–(F.3), *and for an interval A, $\mathcal{I}_A$ can be computed in $O(\log n)$ time.*

Shift Trees. For each pair I_a, I_b of canonical subsets in $\mathcal{I}$, we show how to construct a family $\mathcal{N}^{ab}$ of $O(n_a n_b)$ subsets of $I_a \times I_b$ with properties (S.1) and (S.2). Notice that

the angular intervals (γ_i, γ_{i+1}) and $(\gamma_j + \delta, \gamma_{j+1} + \delta)$ intersect if and only if the shift angle δ is in the interval $\Delta_{ij} = (\gamma_i - \gamma_{j+1}, \gamma_{i+1} - \gamma_j)$. Set $\mathcal{D}_{ab} = \{\Delta_{ij} \mid i \in I_a, j \in I_b\}$. The following lemma is straightforward.

Lemma 4.2. *Let $i \in I_a$ and $j \in I_b$. Then $(i, j) \in V_{ab}(\delta)$ if and only if $\delta \in \Delta_{ij}$.*

Thus, computing $V_{ab}(\delta)$ is reduced to the following one-dimensional *stabbing query*: determine which of the intervals in $\mathcal{D}_{ab}$ contain the query point $\delta \in \mathbb{S}^1$.

To this end, we build a segment tree T_{ab} on the set $\mathcal{D}_{ab}$ of intervals; see [7]. As with the fence tree, if Δ_{ij} contains 0, it is split into two intervals and each of them is inserted separately. We associate two subsets $N_c^{ab}, \bar{N}_c^{ab}$ with each node c of T_{ab}. Suppose the interval Δ_{ij} is stored at a node c of T_{ab}. If $i \neq j$, then we add $[i, j]$ to both N_c^{ab} and $\bar{N}_c^{ab}$. If $i = j$ and $\gamma_{i+1} - \gamma_i \leq \pi$, we add $[i, i]$ to N_c^{ab} only. We set $\mathcal{N}^{ab} = \{N_c^{ab}, \bar{N}_c^{ab} \mid c \in T_{ab}\}$. There are $n_a n_b$ intervals in $\mathcal{D}_{ab}$, so there are $O(n_a n_b)$ nodes in T_{ab}, and thus $O(n_a n_b)$ canonical subsets $N_c^{ab}, \bar{N}_c^{ab} \in \mathcal{N}^{ab}$. Since each $[i, j] \in I_a \times I_b$ is stored in $O(\log n)$ canonical subsets, $|\mathcal{N}^{ab}| = O(n_a n_b \log n)$, as claimed in (S.1).

To compute $V_{ab}(\delta)$ (resp. $\bar{V}_{ab}(\delta)$) for a query shift δ, we follow the path from the root to the leaf whose associated interval contains δ and report N_c^{ab} (resp. $\bar{N}_c^{ab}$) for all $O(\log n)$ nodes c on this path. The correctness of the procedure follows from Lemma 4.2. Hence, (S.2) is also satisfied.

Lemma 4.3. *Given a pair $I_a, I_b \in \mathcal{I}$, we can construct in $O(n_a n_b \log n)$ time a family $\mathcal{N}^{ab}$ of $O(n_a n_b)$ canonical subsets that satisfy* (S.1)–(S.2). *For a given shift angle δ, $\mathcal{N}_\delta^{ab}$ and $\bar{\mathcal{N}}_\delta^{ab}$ can be computed in $O(\log n)$ time.*

5. Conclusion

In this paper we presented a new technique for fence design that, given a polygonal part P with m edges and n stable edges, can compute in $O(m + n^2 \log^3 n)$ time a valid fence sequence if one exists. We believe our approach is general and a number of other part-feeder problems can benefit from this approach. We conclude by mentioning two open problems:

- Can the problem of orienting a polygonal part by applying a sequence of pull operations [4] be solved in near-quadratic time using our graph-compression scheme?
- Is there a near-linear algorithm for fence design? Such an algorithm has to compress the vertices of the fence graph as well.

Acknowledgements

The authors thank Ken Goldberg and John Harer for many useful discussions and two referees for their helpful comments.

Appendix. Proof of Lemma 2.6

In this appendix we sketch the proof of the second half of Lemma 2.6, i.e., every valid fence sequence corresponds to a path in the fence graph from a node $[i, i-1]^s$ to $[j, j]^t$, for some $0 \le i, j < n$ and $s, t \in \{L, R\}$. We adapt the proof by Eppstein [8] for a similar claim involving some other part-feeder problems.

Natarajan [13] showed that a part feeder can be viewed as a finite automaton $\mathcal{A} = (S, \Sigma, \delta)$ as follows: The *states* S are the possible orientations of the part, and the *symbols* in the *alphabet* Σ correspond to the possible operations of the feeder. The *transition function* $\delta\colon \Sigma \times S \to S$ describes the action of a symbol on a state; that is, if the feeder applies operation σ to orientation s of the part, the resulting orientation is $\delta(\sigma, s)$. Let Σ^* be the family of all words derived from Σ. We extend the definition of δ to $\tau \in \Sigma^*$: If τ is the empty string, then $\delta(\tau, s) = s$, and if $\tau = \sigma\bar{\tau}$ for $\sigma \in \Sigma$, then $\delta(\tau, s) = \delta(\bar{\tau}, \delta(\sigma, s))$. For $X \subseteq S$, we use $\delta(\tau, X)$ to denote the set $\{\delta(\tau, s) \mid s \in X\}$ and $\delta^{-1}(\tau, X)$ to denote the set $\{s \mid \delta(\tau, s) \in X\}$. Note that for any $\tau \in \Sigma^*$, $X \subseteq \delta^{-1}(\tau, \delta(\tau, X))$. A word $\tau \in \Sigma^*$ is called a *reset sequence* of $X \subseteq S$ with respect to $\mathcal{A}$ if $|\delta(\tau, X)| = 1$. A successful feeder design is a sequence of operations that produces the same final orientation, regardless of the initial orientation of the part, therefore it corresponds to a reset sequence of S with respect to the finite automaton associated with the feeder.

Although computing a reset sequence of a subset $X \subseteq S$ with respect to a general automata is PSPACE-complete [13], polynomial-time algorithms are known for the so-called monotonic automata [8]. $\mathcal{A}$ is called *monotonic* if there exists a cyclic ordering on S so that if $s_1, \ldots, s_k$ appear in this order, then for any $\sigma \in \Sigma$, $\delta(\sigma, s_1), \ldots, \delta(\sigma, s_k)$ also appear in this cyclic order. We also define the notion of a quasi-monotonic automaton. $\mathcal{A} = (S, \Sigma, \delta)$ is called a *quasi-monotonic* finite automaton if the following two properties hold:

(Q1) S can be partitioned into two subsets S_L and S_R, and Σ can be partitioned into two subsets Σ_L and Σ_R so that

$$\delta\colon \Sigma_L \times S \to S_L \quad \text{and} \quad \delta\colon \Sigma_R \times S \to S_R.$$

(Q2) There is a cyclic ordering of each of S_L and S_R so that if $s_1, \ldots, s_k \in S_L$ appear along this cyclic ordering, then for all $\sigma \in \Sigma_L$ (resp. $\sigma \in \Sigma_R$), $\delta(\sigma, s_1), \ldots, \delta(\sigma, s_k)$ appear along the cyclic ordering of S_L (resp. S_R). The same holds if $s_1, \ldots, s_k \in S_R$.

Lemma A.1. *The fence-design problem can be formulated as finding a reset sequence in a quasi-monotonic finite automaton.*

Proof. Let $S_L = \{(i, L) \mid 0 \le i < n\}$ and $S_R = \{(i, R) \mid 0 \le i < n\}$, $\Sigma = \Theta$ (see Lemma 2.3), and $\delta(\varphi, (i, s)) = \mathcal{F}(i, s, \varphi)$ for any $(i, s) \in S_L \cup S_R$ and $\varphi \in \Sigma$. Then any valid fence sequence is a reset sequence of $\mathcal{A} = (S_L \cup S_R, \Sigma, \delta)$. Let $\Sigma_L = \Sigma \cap (0, \pi/2)$ and $\Sigma_R = \Sigma \cap (\pi/2, \pi)$. By (2) and (3), $\delta(\sigma, (i, s)) \in S_L$ for any $\sigma \in \Sigma_L$ and $\delta(\sigma, (i, s)) \in S_R$ for any $\sigma \in \Sigma_R$. Moreover, Lemma 2.2 implies that there exists a cyclic ordering of each of S_L and S_R that satisfies (Q2). Hence, $\mathcal{A}$ is quasi-monotonic, as desired. □

We define an interval $[s_i, s_j]$ in S_L to be the set of all those states of S_L that lie between s_i and s_j in the cyclic ordering of S_L. We say that an interval $J = [s_i, s_j]$ is contained in another interval $I = [s_g, s_h]$, and denote it by $J \prec I$, if the endpoints of the intervals appear in the cyclic order s_g, s_i, s_j, s_h. Similarly, we define intervals and their containment property for S_R. In order to prove the equivalence between a reset-sequence in the automata induced by an instance of fence design and a path of the desired form in the corresponding fence graph, we introduce a new automaton $\mathcal{A}' = (S', \Sigma, \delta')$, where $S' = \{[s_i, s_j] \mid s_i, s_j \in S_L\} \cup \{[s_i, s_j] \mid s_i, s_j \in S_R\} \cup \{\infty\}$. The new transition function δ': $\Sigma \times S' \to S'$ is defined as follows: Suppose $I = [s_i, s_j]$, $\delta(\sigma, s_i) = t_i$ and $\delta(\sigma, s_j) = t_j$. If $t_i \neq t_j$, then set $\delta'(\sigma, I) = [t_i, t_j]$. If $t_i = t_j$ and $\delta(\sigma, s_k) = t_i$ for all $s_i \leq s_k \leq s_j$, then set $\delta'(\sigma, I) = [t_i, t_i]$. Otherwise, set $\delta'(\sigma, I) = \infty$. If $\sigma \in \Sigma_L$ (resp. Σ_R), then $[t_i, t_j]$ is in S_L (resp. S_R).

We extend Eppstein's argument to show that a reset sequence in a quasi-monotonic automaton $\mathcal{A}$ corresponds to a path in the graph induced by the automaton $\mathcal{A}'$, as follows. The following lemma can be proved by induction, using the quasi-monotonic property of $\mathcal{A}$.

Lemma A.2. *For all $\tau \in \Sigma^*$, for any interval $I \in S' \backslash \{\infty\}$, and for any $p \in \{L, R\}$, $\delta^{-1}(\sigma, I) \cap S_p$ is an interval.*

The next lemma can be proved using the same argument as in [8].

Lemma A.3. *For all $\sigma \in \Sigma_L$, for all intervals $J \subseteq S_L$, and for $p \in \{L, R\}$, if $I = \delta^{-1}(\sigma, J) \cap S_p$ is not all of S_p, then $\delta'(\sigma, I) \neq \infty$ and $\delta'(\sigma, I) \prec J$. The same holds for $\sigma \in \Sigma_R$ and $J \subseteq S_R$.*

Lemma A.4. *Given $\tau \in \Sigma^*$ and $X \subseteq S$, $\delta(\tau, X) = \{s\}$ if and only if for $p \in \{L, R\}$, there is a representation of $\delta^{-1}(\tau, \delta(\tau, X)) \cap S_p$ as an interval I_p such that $\delta'(\tau, I_p) = [s, s]$.*

Proof. For $p \in \{L, R\}$, let I_p be a representation of $\delta^{-1}(\tau, \delta(\tau, X)) \cap S_p$ such that $\delta'(\tau, I_p) = [s, s]$. Since $X \subseteq I_L \cup I_R$, the quasi-monotonicity property implies that $\delta(\tau, X) \subseteq \delta(\tau, I_L \cup I_R) = [s, s]$. Conversely, suppose $\delta(\tau, X) = \{s\}$. We prove the existence of a desired interval by induction on the length of τ. Indeed, if τ is the empty word, then $X = \{s\}$ and $[s, s]$ is the desired interval. Next, let $\tau = \sigma\tau'$. Without loss of generality, assume that $\sigma \in \Sigma_L$. By the induction hypothesis, there is an interval $J \subseteq S_L$ representing $\delta^{-1}(\tau', s) \cap S_L$. If $I_p = \delta^{-1}(\sigma, J) \cap S_p$ is not all of S_p, then Lemma A.3 implies that $\delta(\tau, I_p) = [s, s]$. On the other hand, if I_p is all of S_p, then we follow the same argument as in [8] to prove that there exists an interval $I'_p \subseteq S_p$ such that $\delta(\tau, I'_p) = [s, s]$. We refer the reader to the original paper for further details. □

Putting everything together, we can now prove the main claim of this appendix.

Lemma A.5. *A valid fence sequence corresponds to a path in the fence graph from a vertex $[i, i-1]^s$ to $[j, j]^t$ for some $0 \leq i, j < n$, and $s, t \in \{L, R\}$.*

Proof. By Lemma A.1, there is a quasi-monotonic finite automaton $\mathcal{A} = (S, \Sigma, \delta)$ so that a valid fence sequence for P corresponds to a reset sequence τ of S with respect to $\mathcal{A}$. Since $S_L \subseteq S$, τ is also a reset sequence for S_L. Therefore, by Lemma A.4, there is a representation of $\delta^{-1}(\tau, \delta(\tau, S_L)) = S_L$ as an interval I_p so that $\delta'(\tau, I_p) = [j, j]^t$ for some $0 \leq j < n$. Since any representation of S_L as an interval in S' is of the form $[i, i-1]^L$, for some $0 \leq i < n$, τ corresponds to a path from $[i, i-1]^L$ to $[j, j]^t$ in the graph $G_{\mathcal{A}}$ induced by $\mathcal{A}'$. Note that the fence graph is the same as $G_{\mathcal{A}}$ except that the node ∞ and the edges connected to ∞ have been removed. Since there is no out-going edge from ∞, a path from $[i, i-1]^L$ to $[j, j]^t$ in $G_{\mathcal{A}}$ is also a path in the fence graph. Hence, the lemma is true. □

References

1. P. K. Agarwal and K. R. Varadarajan, Efficient algorithms for approximating polygonal chains, *Discrete & Computational Geometry*, **23** (2000), 273–291.
2. R.-P. Berretty, Geometric Design of Part Feeders, Ph.D. Thesis, Utrecht University, Utrecht, 2000.
3. R.-P. Berretty, K. Goldberg, M. Overmars, and A. F. van der Stappen, Algorithms for fence design, in: *Robotics*: *The Algorithmic Perspective* (P. K. Agarwal, L. E. Kavraki, and M. T. Mason, eds.), AK Peters, Natick, MA, 1998, pp. 279–296.
4. R.-P. Berretty, K. Goldberg, M. Overmars, and A. F. van der Stappen, Orienting parts by inside-out pulling, *Proceedings of the* 2001 *IEEE International Conference on Robotics and Automation*, 2001, pp. 1053–1058.
5. M. E. Brokowski, M. A. Peshkin, and K. Goldberg, Optimal curved fences for part alignment on a belt, *ASME Journal of Mechanical Design*, **117** (1995), 27–34.
6. Y.-B. Chen and D. Ierardi, The complexity of oblivious plans for orienting and distinguishing polygonal parts, *Algorithmica*, **14** (1995), 367–397.
7. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf, *Computational Geometry*: *Algorithms and Applications*, Springer-Verlag, Berlin, 1997.
8. D. Eppstein, Reset sequences for monotonic automata, *SIAM Journal on Computing*, **19** (1990), 500–510.
9. T. Feder and R. Motwani, Clique partitions, graph compression, and speeding-up algorithms, *Proceedings of the* 23*rd ACM Symposium on Theory of Computing*, 1991, pp. 123–133.
10. K. Y. Goldberg, Orienting polygonal parts without sensors, *Algorithmica*, **10** (1993), 210–225.
11. M. T. Mason, Manipulator Grasping and Pushing Operations, Ph.D. Thesis, MIT, 1982.
12. M. T. Mason, *Mechanics of Robotic Manipulation*, MIT Press, Cambridge, MA, 2001.
13. B. K. Natarajan, Some paradigms for the automated design of parts feeders, *International Journal of Robotics Research*, **8** (1989), 98–109.
14. M. A. Peshkin and A. C. Sanderson, Planning robotic manipulation strategies for workpieces that slide, *IEEE Journal of Robotics and Automation*, **4** (1988), 524–531.
15. A. F. van der Stappen, R.-P. Berretty, K. Goldberg, and M. H. Overmars, Geometry and part feeding, in: *Sensor Based Intelligent Robot Systems* (G. D. Hager, H. I. Christensen, H. Bunke, and R. Klein, eds.), Lecture Notes in Computer Science 2238, Springer-Verlag, Berlin, 2002, pp. 259–281.
16. J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski, A complete algorithm for designing passive fences to orient parts, *Assembly Automation*, **17** (1997), 129–136.

Received September 9, 2003, *and in revised form June* 30, 2004. *Online publication January* 21, 2005.