A Near-Quadratic Algorithm for Fence Design*

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

[1] Duke University, Box 90129, Durham, NC 27708-0129. 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] Stanford University, Bldg. 380, Stanford, CA 94305-2125 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. As in [3], 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 the state graph, we show that it can be represented implicitly and a desired path can be found in $O(m + n^2 \log^3 n)$ time. Our technique is quite general and is applicable to other part manipulation problems.

1 Introduction

Robotic manipulation deals with various part handling problems in industrial automation [11]. One such problem, which arises in automated assembly, is the so-called *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* orientation of parts, 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 3-dimensional part as it moves by, or using a collection of conveyor belts at varying heights. Parts can be pushed with fences, squeezed between parallel-jaw grippers, pulled from the inside-out, dropped

* Research by P.A. is supported by the NSF under grants CCR-00-86013 EIA-98-70724, EIA-01-31905, ITR–333–1050, 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 ITR–333–1050, CCR–97-32787, DMS–0107621, and DMS–9983320. Part of this work was done while the second author was visiting Department of Computer Science, University of North Carolina, Chapel Hill.

through traps, tilted on a table, or subjected to vibrating plates with programmable vector fields. See [2,14,11] 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 2-dimensional part moving on a conveyor belt.

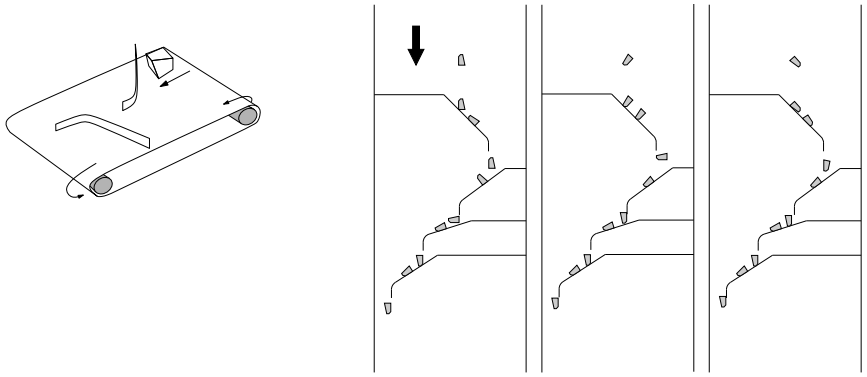

Fig. 1. A polygonal part is moved by a conveyor belt past a series of rigid fences. The fences passively reorient the part, which leaves the last fence in a unique orientation regardless of its initial orientation.

Our model. Our feeder consists of a conveyor belt equipped with a series of fences designed to passively reorient a part 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 Figure 1 for an example.

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.

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 [4]. This ensures that when the part leaves the fence, the aligned edge is parallel to and facing the wall from which the fence emanates.

[1] We assume that the motion of P follows the pushing model proposed by Mason [10]. See the original paper and [2] for details, which we omit from here.

We work in a frame of reference in which the belt moves downward. A fence is specified by its *fence angle* ϕ, 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 $\phi \in (0, \pi/2)$, while $\phi \in (\pi/2, \pi)$ for a right fence (Figure 2). Note that these are open intervals; a part cannot pass a fence with $\phi = \pi/2$, and the vertical fences at $\phi = 0$ and π have no effect.

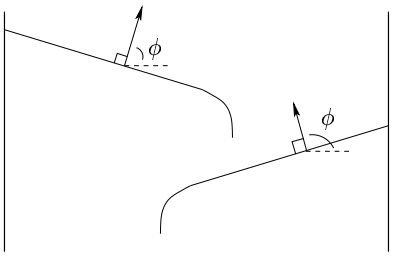

Fig. 2. The fence angle ϕ is the direction of the upward normal to the fence.

Given a polygonal part P, the *fence-design problem* is to construct a sequence $\Phi = \langle \phi_1, \ldots, \phi_k \rangle$ of fence angles so that, regardless of its initial orientation, P always leaves the last fence in a unique final orientation.

Related work. Goldberg [9] 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. His algorithm requires $\Omega(m^2)$ pushes in the worst-case. Chen and Ierardi [5] constructed a linear sequence of such pushes.

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 [13]. They introduce a graph representation of the state space, in which each node represents a set of orientations in which the part can be at a given moment and in which certain paths represent successful fence designs. They discretize the set of allowable fences, 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. [4] constrained the part to one of $O(m)$ orientations by adding a curved tail to each fence. This led to the complete algorithm of Wiegley et al. [15], which allows for all possible fence angles and is guaranteed to find a

sequence of fences to orient a part if one exists. They conjecture that a polynomial-time fence-design algorithm exists for any polygonal part.

Application of a general result by Eppstein [7] immediately yields an $O(m^4)$-time algorithm; successful fence designs again correspond to certain paths in a state graph that requires only $O(m^2)$ of the $O(2^m)$ possible states of the part. 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.

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 [1,8], we believe this is the first application of this approach for a manipulation problem.

We believe that our approach is versatile and will find many other applications in those part manipulation problems that can be formulated as searching in a graph. For example, the same technique can give an improved $O(n^2 \log^3 n)$ algorithm for the problem of orienting micro-scale parts using squeeze and roll primitives, studied by Moll et al. [12].

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* $\rho : \mathbb{S}^1 \to \mathbb{R}^+$ of P is defined as follows: Let ℓ be the line tangent to P whose inward (towards 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 Figure 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 counter-clockwise direction with $\gamma_0 < \varepsilon_0 < \gamma_1 < \cdots < \varepsilon_{n-1}$. 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 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.

Now, suppose that we push P with a fence, in the direction normal to the fence. As described in [10], During the push, P rotates in the direction that decreases ρ until a stable edge of P aligns with the fence, and then P translates in the direction

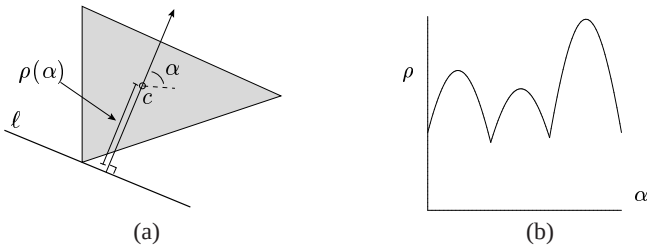

Fig. 3. The radius function $\rho : \mathbb{S}^1 \to \mathbb{R}^+$. (a) α is the direction of the ray normal to ℓ that passes through c. (b) ρ is a piecewise-sinusoidal function of α.

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; Figure 4. That is,

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

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

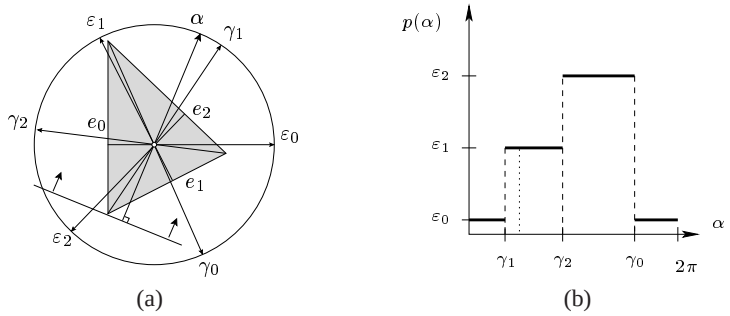

Fig. 4. The push function $p : \mathbb{S}^1 \to \mathbb{S}^1$. (a) A push in direction α with $p(\alpha) = \varepsilon_1$. (b) $p(\alpha)$ is a step function. ε_i and γ_i are the local minima and maxima of ρ.

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 [11]. 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 behavior of P when it encounters a fence on the belt.

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 $\phi \in (0, \pi/2)$, or the right side of the belt if $\phi \in (\pi/2, \pi)$. This ensures that P is in one of only $2n$ orientations as it travels between fences. We will 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.

Suppose P is initially in orientation (i, L). When it encounters a fence with fence angle ϕ, P feels a push in direction $\varepsilon_i + \phi$ in its own frame. If $\gamma_j < \varepsilon_i + \phi < \gamma_{j+1}$, then the push function $p(\varepsilon_i + \phi) = \varepsilon_j$ 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 $\varepsilon_i + \phi + \pi$ direction, and the final orientation depends on $p(\varepsilon_i + \phi + \pi)$. See Figure 5.

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 $\theta = \phi$ or $\theta = \phi + \pi$ for any particular push, where $\phi \in (0, \pi)$. Only half of the possible reorientations are available at a given time, which implies that some orientations are unattainable from others. It is precisely this fact that makes fence design harder than pushing.

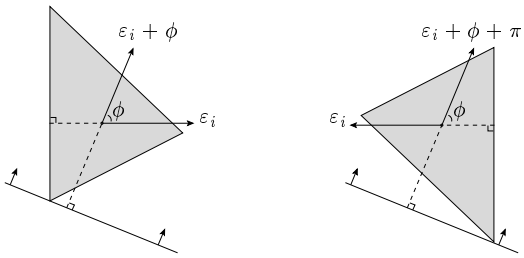

Fig. 5. The action of a fence: (i) If P is initially in orientation (i, L), it feels a push in direction $\alpha = \varepsilon_i + \phi$. (ii) If P is initially in orientation (i, R), it feels a push in direction $\alpha = \varepsilon_i + \phi + \pi$.

Definition 1 Given polygon P with n stable edges, define its *fence function*

$$\mathcal{F} : \{1, .., n\} \times \{L, R\} \times (0, \pi) \to \{1, .., n\} \times \{L, R\}$$

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

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

and

$$t = \begin{cases} L \text{ if } \ \phi \in (0, \pi/2), \\ R \text{ if } \ \phi \in (\pi/2, \pi). \end{cases}$$

As observed in [3], $\mathcal{F}$ is *monotonic* in the sense that $i_1 \leq i_2 \leq i_3$ implies that $\mathcal{F}(i_1, s, \phi) \leq \mathcal{F}(i_2, s, \phi) \leq \mathcal{F}(i_3, s, \phi)$, for fixed s and ϕ. This monotonicity was crucial for their algorithm, and it will be crucial for ours as well.

A convenient representation of the fence function is by a collection of n circles, with one circle C_i for each stable edge e_i of P. C_i is marked with the intervals $\chi_{ij} = (\gamma_j - \varepsilon_i, \gamma_{j+1} - \varepsilon_i)$, for every $0 \leq j < n$; imagine rotating the circle clockwise by ε_i, Figure 6. The interval χ_{ij} on C_i represents all tripls s, t, ϕ for which $\mathcal{F}(i, s, \phi) = (j, t)$.

Suppose that $\theta \in \chi_{ij} \cap (0, \pi)$. Then $\gamma_j < \varepsilon_i + \theta < \gamma_{j+1}$ implies that $p(\varepsilon_i + \theta) = \varepsilon_j$; 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 χ_{ij} intersects the lower half of C_i, then there exists a fence which takes the right orientation (i, R) to one of the orientations (j, t). Suppose $\theta \in \chi_{ij} \cap (\pi, 2\pi)$. Of course, θ itself is not a valid fence angle, but $\phi = \theta - \pi$ is, and $p(\varepsilon_i + \phi + \pi) = \varepsilon_j$. Therefore, $\theta \in (\pi, 3\pi/2)$ implies $\mathcal{F}(i, R, \theta - \pi) = (j, L)$, and $\theta \in (3\pi/2, 2\pi)$ implies $\mathcal{F}(i, R, \theta - \pi) = (j, R)$.

Consider the circle in Figure 6. χ_{ij} intersects both quadrants I and II, which implies that there exist fence angles ϕ_1 and ϕ_2 with $\mathcal{F}(i, L, \phi_1) = (j, L)$ and $\mathcal{F}(i, L, \phi_2) = (j, R)$. Note that χ_{ij} does not intersect the lower semicircle, and $\mathcal{F}(i, R, \phi) \neq (j, t)$ for any $\phi \in \chi_{ij}$ and $t \in \{L, R\}$. But χ_{ij-1} intersects the fourth quadrrant so there exists $\phi \in \chi_{ij-1}$ for which $\mathcal{F}(i, R, \phi) = (j - 1, R)$.

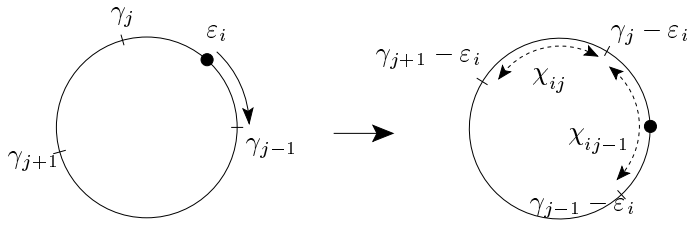

Fig. 6. The fence circle C_i is obtained by rotating the circle clockwise by ε_i.

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

Fence graph. Next, we show how the fence-design problem can be formulated as a path search in a graph. The graph defined here is slightly different from the one constructed in [3]. While the modified definition provides 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.

Definition 2 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 subsets 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 from $[i, j]^s$ to $[i', j']^t$ whenever there exists a fence angle ϕ such that both $\mathcal{F}(i, s, \phi) = (i', t)$ and $\mathcal{F}(j, s, \phi) = (j', t)$. We refer to these edges as *non-sink* edges.

- There is an edge from $[i, j]^s$ to $[i', i']^t$ if there exists a fence angle ϕ with $\mathcal{F}(k, s, \phi) = (i', t)$ for *all* $i \leq k \leq j$. We refer to these edges as *sink* edges.

Lemma 1. *Every node in G has degree $O(n)$.*

Proof: The neighbors of $[i, j]^s$ can be determined by overlaying the fence circles C_i and C_j. If $[i', j']^t$ is a neighbor, then $\chi_{ii'} \cap \chi_{jj'} \neq \emptyset$. The overlay of C_i and C_j is partitioned into $2n$ intervals by the endpoint of the χ's, and each interval corresponds to at most one edge of G. □

Lemma 2. *Every successful fence design corresponds to a path in G from a node of the form $[i, i - 1]^s$ to a singleton node $[j, j]^t$, and vice versa.*

Proof: The monoticity of $\mathcal{F}$ implies that G has an edge $[i, j]^s \to [i', j']^t$ if and only if there exists a single fence angle ϕ so that $\mathcal{F}(k, s, \phi) \in [i', j']^t$ for all $(k, s) \in [i, j]^s$. Then given a path in G from $[i - 1, i]^s$ to $[j, j]^t$, we construct a fence design by choosing one such fence angle for each edge in the path.

The proof of the converse, that every succesful fence design corresponds to such a path, is a bit more subtle but follows easily from a more general result by Eppstein [7] relating feeder design problems to monotonic automata. We omit the details from this version of the paper.

□

By Lemmas 1 and 2,

Theorem 3. *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 fence graph compression, we pause to consider the sink edges, i.e., the edges of the form $([i, j]^s, [i', i']^t)$, in more detail. Suppose $\mathcal{F}(i, s, \phi) = (i', t)$ and $\mathcal{F}(j, s, \phi) = (j', t)$. If $i' \neq j'$, the monotonicity of $\mathcal{F}$ ensures that $\mathcal{F}(k, s, \phi) \in [i', j']^t$ for all $(k, s) \in [i, j]^s$, so we automatically add an edge to G. However, this is not guaranteed when $i' = j'$, since it is unclear whether $[i', i']^t$ should represent the set of all orientations or just the singleton node $\{(i', t)\}$. We therefore need to take extra care with the sink edges when we compress G in the next section. To begin, we note that the solution is trivial for certain parts.

Lemma 3. *If $\gamma_{i+1} - \gamma_i > \varepsilon_j - \varepsilon_{j+1}$ for some i and j, then two fences are sufficient to orient P.*

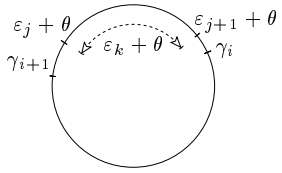

Fig. 7. For $\gamma_{i+1} - \gamma_i > \varepsilon_j - \varepsilon_{j+1}$, there exists θ s.t. $p(\varepsilon_k + \theta) = \varepsilon_i$ for all k.

Proof: Since $\gamma_{i+1} - \gamma_i > \varepsilon_j - \varepsilon_{j+1}$, we have $\gamma_i < \varepsilon_{j+1} + \theta < \varepsilon_j + \theta < \gamma_{i+1}$ for some $\theta \in [0, 2\pi)$. Since $\varepsilon_k \in (\varepsilon_{j+1}, \varepsilon_j)$, for all k, $p(\varepsilon_k + \theta) = \varepsilon_i$ for all k (see Figure 7). A reorientation by θ can be accomplished with exactly two fences, as follows. If $\theta \in (0, \pi)$, then we set $\phi_1 = \pi/4$ and $\phi_2 = \theta$; if $\theta \in (\pi, 2\pi)$, then we set $\phi_1 = 3\pi/4$ and $\phi_2 = \theta - \pi$. The first fence align 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 has the following property:

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

Lemma 4. *Suppose that P satisfies $(*)$, $\mathcal{F}(i, s, \phi) = \mathcal{F}(j, s, \phi) = (i', t)$ for some $\phi \in (0, \pi)$, 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 $\theta = \phi$ if $s = L$ and $\theta = \phi + \pi$ if $s = R$. Suppose first that $\varepsilon_j - \varepsilon_i \leq \pi$. Since $\varepsilon_i + \theta, \varepsilon_j + \theta \in (\gamma_{i'}, \gamma_{i'+1})$ and $\gamma_{i'+1} - \gamma_{i'} \leq \pi$, we must have $\gamma_{i'} < \varepsilon_i + \theta < \varepsilon_j + \theta < \gamma_{i'+1}$; Figure 8(a). This implies that $\mathcal{F}(k, s, \phi) = (i', t)$ 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 $\varepsilon_i + \theta < \gamma_{i'+1} < \gamma_{i'} < \varepsilon_j + \theta$; Figure 8(b). If $[i, j]^s \to [i', i']^t$ is an edge of G, 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 less space and still allows us to quickly compute a path between two vertices. We first explain the compression scheme in general and then apply it to the fence graph.

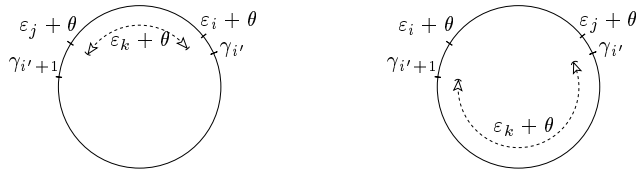

Fig. 8. For most parts, $[i,j]^s \to [i',i']^t$ exists if and only if $\varepsilon_j - \varepsilon_i \leq \pi$. (a) If $\varepsilon_j - \varepsilon_i \leq \pi$, then $\varepsilon_k + \theta \in (\gamma_{i'}, \gamma_{i'+1})$ for all $i \leq k \leq j$, (b) but not if $\varepsilon_j - \varepsilon_i > \pi$.

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

$$\text{(i) } A_i, B_i \subseteq V, \qquad \text{(iii) } E = \bigcup_i E_i,$$

$$\text{(ii) } E_i = A_i \times B_i \subseteq E, \qquad \text{(iv) } E_i \cap E_j = \emptyset, \text{ if } i \neq j.$$

Conditions (i) and (ii) imply that each 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* $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ G by setting

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

See Figure 9.

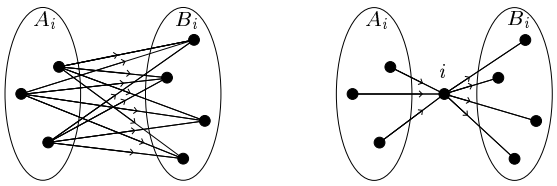

Fig. 9. G and $\widetilde{G}$ store the same connectivity information, but $\widetilde{G}$ has fewer edges.

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 5. *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 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

$$\text{(i) } N(\mu) = \bigcup_{B_i \in \mathcal{N}_\mu} B_i, \text{ and}$$

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

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

Proof: Clearly, (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 thus $\mathcal{G}$ is a bipartite clique 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., $\gamma_i - \gamma_{i+1} \geq \varepsilon_{j+1} - \varepsilon_j$ 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 $\mathcal{F}(i, s, \phi) = (i', t)$ and $\mathcal{F}(j, s, t) = (j', t)$ for some $\phi \in (0, \pi)$. 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 $[i, j]$ 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. Hence, nowonwards we are interested in computing a compressed representation of only those neighbors neighbors of a vertex that correspond to either a non-sink edge or a sink edge $[i', i']$ 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 will show in Section 4 how we compute in $O(n \log n)$ time $\mathcal{I}$ 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 $\bar{V}_{ab}(\delta) = V_{ab}(\delta) \setminus \{(i, i) \mid 1 \le i \le n\}$, i.e., $\bar{V}_{ab}$ is the same as V_{ab} except that we remove all nodes that would represent sink edges.

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) $|\mathcal{N}^{ab}| = \sum_{N \in \mathcal{N}^{ab}} |N| = O(n_a n_b \log n)$, and
(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

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

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 $\gamma_{i'+1} - \gamma_{i'} \le \pi$. 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 $\mathcal{F}(i, L, \phi) = (i', L)$ and $\mathcal{F}(j, L, \phi) = (j', L)$ for some $\phi \in (0, \pi/2)$. Furthermore if $[i, j]^L \to [i', i']^L$ with $\gamma_{i'+1} - \gamma_{i'} \le \pi$ is a sink edge if and only if $\mathcal{F}(i, L, \phi) = \mathcal{F}(j, L, \phi) = (i', L)$ and $\varepsilon_j - \varepsilon_i \le \pi$.

We compute 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 $\varepsilon_j - \varepsilon_i \le \pi$; 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 7. *The set of left neighbors of a left node $[i,j]^L \in V$ is*

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

Proof: Let us assume that $\varepsilon_j - \varepsilon_i \le \pi$. Since P satisfies $(*)$, by Lemma 4, 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 circle C_i and C_j.

The family $\mathcal{I}_{A_i}$ partitions the first quadrants of C_i and C_j each 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 $j' \in I_b$. 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. □

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 $A_i = (\varepsilon_i + \pi/4, \varepsilon_i + \pi/2)$; for $s = R, t = L$, we set $A_i = (\varepsilon_i + \pi/2, \varepsilon_i + \pi)$; and for $s = R, t = R$, we set $A_i = (\varepsilon_i + 3\pi/2, \varepsilon_i)$. We do the same for A_j in each case, and then follow the above procedure.

Theorem 4. *Given a convex m-gon P with n stable edges, we can compute a fence design in $O(m + n^2 \log^3 n)$ time.*

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 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

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

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 $\sum_i |A_i| = \sum_{\mu \in V} |N_\mu|$. 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

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

Moreover,

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

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, called *fence tree* and *shift tree*, which generate $\mathcal{I}$ 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 $\{\gamma_i \mid i \in [0 : n-1]\}$. Each internal node of T stores the leftmost leaf of its right subtree to guide the search. 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 $0 \in (\alpha_1, \alpha_2)$ (mod 2π), we split A into two intervals $(\alpha_1, 2\pi]$ and $[0, \alpha_2)$, and work with each of them separately, so let us assume that $0 \notin (\alpha_1, \alpha_2)$. In order to compute $\mathcal{I}_A$, we regard T as a 1-dimensional range tree [6], we query T with the interval (α_1, α_2), and report the γ_i's lying in the query interval as a disjoint union of $O(\log n)$ canonical intervals. The query procedure finds the leftmost leaf γ_l such that $\gamma_l > \alpha_1$ and the rightmost leaf $\gamma_r < \alpha_2$. Let u be their least common ancestor of γ_l, γ_r. For any node on the path from u to γ_l (resp. γ_r) if its right (resp. left) child v is not on the path, we add $[l_v : r_v]$ to $\mathcal{I}_A$.

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 8. *$[i, j] \in V_{ab}(\delta)$ if and only if $\delta \in \Delta_{ij}$.*

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

To this end, we build a segment tree T_{ab} on the set $\mathcal{D}_{ab}$ of *critical* intervals; see [6]. As with the fence tree, if Δ_{ij} contains 0, it is split into two intervals and each of

them is inserted separately. We associated two subsets N_c^{ab}, $\bar{N}_c^{ab}$ with each node c of T_{ab}. Suppose a critical 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}. We set $\mathcal{N}^{ab} = \{N_c^{ab}, \bar{N}_c^{ab} \mid c \in T_{ab}\}$. There are $n_a n_b$ critical 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 $N_c^{ab} \in \mathcal{N}^{ab}$, $\bar{N}_c^{ab}$. 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 point 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 8. Hence, (S.2) is obviously satisfied.

5 Conclusion

In this paper, we presented a new technique for computing a fence design in near-quadratic $O(m+n^2 \log^3 n)$ time, thus improving the previous $O(m+n^3)$ algorithm. Here, the polygonal part to be oriented has m vertices and n stable edges. The approach, based on a graph compression scheme, is quite general. There are a number of other part-feeding problems which can be solved by searching for certain paths in a discrete state space, and our compression technique applies to these as well.

References

1. P. K. Agarwal and K. R. Varadarajan. Efficient algorithms for approximating polygonal chains. *Discrete & Computational Geometry*, 23:273–291, 2000.
2. R.-P. Berretty. *Geometric Design of Part Feeders*. PhD thesis, Utrecht University, Utrecht, The Netherlands, 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. Kavaraki, and M. T. Mason, eds.), AK Peters, Natick, MA, 1998, 279–296.
4. M. E. Brokowski, M. A. Peshkin, and K. Goldberg. Optimal curved fences for part alignment on a belt. *ASME Journal of Mechanical Design*, 117:27, 1995. **Page numbers**
5. Y.-B. Chen and D. Ierardi. The complexity of oblivious plans for orienting and distinguishing polygonal parts. *Algorithmica*, 14(5):367–397, 1995.
6. M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer Verlag, Berlin, 1997.
7. D. Eppstein. Reset sequences for monotonic automata. *SIAM J. Comput.*, 19:500–510, 1990.
8. T. Feder and R. Motwani. Clique partitions, graph compression, and speeding-up algorithms. In *Proc. 23rd Ann. ACM Symp. on Theory of Computing*, pages 123–133, 1991.
9. K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:210–225, 1993.
10. M. T. Mason. Manipulator grasping and pushing operations. Ph.D. Thesis, MIT, 1982. Published in Robot Hands and the Mechanics of Manipulation, MIT Press, Cambridge, 1985, 1982.

11. M. T. Mason. *Mechanics of Robotic Manipulation.* MIT Press, August 2001. Intelligent Robotics and Autonomous Agents Series, ISBN 0-262-13396-2.
12. M. Moll, K. Goldberg, M. Erdmann, and R. Fearing. Orienting micro-scale parts with squeeze and roll primitives. In *IEEE International Conference on Robotics and Automation*, May 2002. **page numbers?**
13. M. A. Peshkin and A. C. Sanderson. Planning robotic manipulation strategies for workpieces that slide. *IEEE Journal of Robotics and Automation*, 4:524–531, 1988.
14. A. van der Stappen, R.-P. Berretty, K. Goldberg, and M. Overmars. Geometry and part feeding. to appear in: Modelling of Sensor-Based Intelligent Robot Systems (H. Bunke, H.I. Christensen, G. Hager, R. Klein Eds.) Lecture Notes in Computer Science, Springer Verlag, Berlin, 2001. **Page numbers??**
15. J. Wiegley, K. Goldberg, M. Peshkin, and M. Brokowski. A complete algorithm for designing passive fences to orient parts. In *IEEE Int. Conf. Robotics and Automation*, 1995. **Page numbers**