ELSEVIER

Computational Geometry 13 (1999) 3–19

Computational Geometry
Theory and Applications

www. elsevier.nl/locate/comgeo

Rotational polygon containment and minimum enclosure using only robust 2D constructions ☆

Victor J. Milenkovic *,1

Department of Mathematics and Computer Science, University of Miami, P.O. Box 249085, Coral Gables, FL 33124, USA

Communicated by J. Rossignac; submitted 1 November 1998; accepted 4 January 1999

Abstract

An algorithm and a robust floating point implementation is given for *rotational polygon containment*: given polygons $P_1, P_2, P_3, \ldots, P_k$ and a container polygon C, find rotations and translations for the k polygons that place them into the container without overlapping. A version of the algorithm and implementation also solves *rotational minimum enclosure*: given a class $\mathcal{C}$ of container polygons, find a container $C \in \mathcal{C}$ of minimum area for which containment has a solution. The minimum enclosure is approximate: it bounds the minimum area between $(1-\varepsilon)A$ and A. Experiments indicate that finding the minimum enclosure is practical for $k = 2, 3$ but not larger unless optimality is sacrificed or angles ranges are limited (although these solutions can still be useful). Important applications for these algorithm to industrial problems are discussed. The paper also gives practical algorithms and numerical techniques for robustly calculating polygon set intersection, Minkowski sum, and *range intersection*: the intersection of a polygon with itself as it rotates through a range of angles. In particular, it introduces *nearest pair rounding*, which allows all these calculations to be carried out in rounded floating point arithmetic.

Keywords: Layout; Packing or nesting of irregular polygons; Containment; Minimum enclosure; Robust geometry; Geometric rounding

1. Introduction

A number of industries generate new parts by cutting them from stock material: cloth, leather (hides), sheet metal, glass, etc. These industries need to generate dense non-overlapping layouts of polygonal shapes. Because fabric has a grain, apparel layouts usually permit only a finite set of orientations.

☆ Expanded version of a paper presented at the 14th ACM Symposium on Computational Geometry, June 1998.

* E-mail: vjm@cs.miami.edu; http://www.cs.miami.edu

1 This research was funded by the Textile/Clothing Technology Corporation from funds awarded by the Alfred P. Sloan Foundation, by NSF grants CCR-91-157993 and NSF-CCR-97-12401, and by a subcontract of a National Textile Center grant to Auburn University, Department of Consumer Affairs.

0925-7721/99/$ – see front matter
PII: S0925-7721(99)00006-1

Stripes, plaids, or other patterns on the fabric can further limit the allowed orientations and translations. Nevertheless, apparel manufacturers often allow small "invisible" rotations (called "tilting"), typically not more than 3 degrees, but still large enough to make a significant difference in cloth utilization. Glass and sheet metal (and sometimes leather) have no grain (or stripes or plaids), and therefore layout on these materials allows arbitrary orientations for the parts.

The problem these manufacturers need to solve is either *containment* or *minimum enclosure*.

Containment Given polygons $P_1, P_2, \ldots, P_k$ and a fixed container C, place the polygons into C without overlapping. As in the case of overlap minimization, P_0 denotes $\overline{C}$, in which case the goal becomes to find a non-overlapping layout of $P_0, P_1, \ldots, P_k$.

Minimum enclosure Find a non-overlapping layout of $P_1, P_2, \ldots, P_k$ that minimizes some measure of the container. The most common version in the textile industry is *strip packing*: minimize the length of a rectangular container R of fixed width. It is also useful to minimize the area of R.

The values of k one sees is often in the dozens or even hundreds. Unfortunately, even the translational versions of containment and minimum enclosure are NP-hard, and therefore one has to expect the running time of a containment or minimum enclosure algorithm to be exponential in k. In the apparel industry, no layout software has yet replaced a human. All hope is not lost: one has to use a heuristic or meta-heuristic of which there are many for the layout problem.

In the translational case, we have demonstrated [8] that a containment *algorithm* for modest k ($k = 5$ or even perhaps $k = 2$) is an excellent *tool* for the creation of containment *heuristics* for larger values of k, even exceeding expert human performance in some cases. This work is based on packing one column at a time, but it is only one among many ways a "large k" heuristic can use a "small k" algorithm. For instance, subsets of the polygons might be packed together tightly and then the union treated as a single polygon.

This paper examines the following question: for what values of k is it possible to practically solve rotational containment and minimum enclosure problems *algorithmically*? In other words, for what value of k is it necessary to switch to heuristics and give up on performance guarantees? The two main issues are running time and numerical robustness.

1.1. New results

This paper presents new algorithms and robust floating point implementations of these algorithm for rotational polygon containment and minimum enclosure. As far as we know, there are no other *algorithms* for $k > 2$ or implementations for $k \geqslant 2$ for multiple non-convex polygons. Minimum enclosure algorithms are given for the following classes: (1) rectangles of fixed width, (2) scaled copies of a fixed convex polygon, (3) arbitrary rectangles. The minimum enclosure is approximate: it bounds the minimum area between $(1 - \varepsilon)A$ and A. Experiments are done to determine the largest practical value of k for minimum enclosure of type (1). Since the minimum enclosure algorithm seeks the smallest possible container, solving minimum enclosure also gives the running time for the "hardest" instances of containment.

The property that distinguishes an *algorithm* from a *heuristic* is the ability to say "no". If a *heuristic* fails to place polygons into a container, it just says that it cannot do it, not that the problem itself is infeasible. An *algorithm*, by definition, must be able to detect infeasibility. In our algorithms, the key to

detecting infeasibility is the *range intersection* $P(\alpha, \beta)$, of a polygon P: the intersection of P with itself as it rotates through the ranges of angles from α to β about the origin. The new containment/enclosure algorithms integrate the range intersection into the framework we developed originally for solving translational containment/minimum enclosure [28,33].

This paper also shows that the range intersection has linear complexity, although it may have circular arcs in its boundary. It gives practical algorithms for computing the range intersection and for approximating it by a polygon in a manner that is suitable for the containment/enclosure algorithms.

Like our earlier translational algorithms, our new containment/enclosure algorithms are very numerically demanding. They apply polygon set intersection and Minkowski sum over and over to the same collection of polygons. *Cascading*, the repeated use of the output as the next input, cannot be implemented using "pure" exact arithmetic because the number of bits required to represent coordinates can grow exponentially [31]. Cascading also quickly crashes any naive rounded arithmetic implementation. This paper presents a new hybrid technique *nearest pair rounding* for implementing polygon set operations in floating point arithmetic. Currently, nearest pair rounding must use some exact arithmetic to ensure correctness, but we have created a version that uses no exact arithmetic at all and which, so far, has been numerically robust, even under the cascading "torture test".

1.2. Related work

In the literature, there are essentially three approaches to layout of non-convex polygons: (1) heuristics and meta-heuristics (neural nets, simulated annealing, genetic algorithms), (2) "classical" computational/combinatorial geometry, (3) a combination of computational geometry and mathematical programming (which we use here).

Heuristics and meta-heuristics are limited in theory because they cannot say "no" if there is no solution. According to our sources in the apparel industry, available layout software is limited in practice to falling about 5% behind humans in cloth utilization. There are a number of surveys of packing/nesting heuristics [7,10–12,36,38]. Recent work has used simulated annealing [21], boundary matching [22], grouping of polygons into sub-rectangles [13], genetic algorithms [5], database driven layout [23,24], or a hybrid approach [18,19,34,37]. Heuristic approaches have been tailored to sheet metal [35] and leather [20], both of which permit rotations.

Chazelle [6] introduced the single-polygon containment problem: place m-gon P into n-gon container Q. For convex Q, the running time bound is $O(mn^2)$, and for non-convex P and Q, $O(m^3n^3(m+n)\log(m+n))$. Avnaim et al. [3,4] improve this to $O(m^3n^3\log(m+n))$. Most recent work deals with finding the largest copy of a convex P that can be placed, which is equivalent to finding the minimum (scaled) enclosure. For convex Q the best running time is $O(mn^2\log n)$ [1], and for non-convex Q, $O(m^2n^2)$ [2]. Grinde and Cavalier use an extensive case analysis and linear programming to place a single convex P [15] or convex P_1 and P_2 [16]. The running time of the first algorithm appears to be $O(m^2n^3)$, and it is not clear what the running time of the second is, but for one of its cases they are able to use parametric programming and find a solution by solving $O(m^4n^4)$ linear programs.

For multi-polygon *translational* layout, we have had considerable theoretical and practical success using a combination of computational geometry and mathematical programming (CG/MP) [9,28,29]. These algorithms can practically solve containment for up to ten polygons and minimum enclosure for at least five. We have also proved a number of theoretical running time bounds including $O((m^2+mn)^{2k}\log n)$ for placing k non-convex m-gons into a non-convex n-gon and $O(m^{4k-4}\log m)$ for placing

them into a minimum area, fixed orientation rectangle [27] (this paper also surveys other results in multi-polygon translational layout).

1.3. Relation of new algorithm to previous work

Avnaim and Boissonnat gave a formula for placing two polygons in a container. This formula works for both the translational and the rotational case. Our previous translational algorithm generalizes this work to $k > 2$ polygons through the use of a branch and bound paradigm. It is possible to generalize our translational algorithm to handle the rotational case in the same manner. One might call this the *true generalization* of our translational algorithm.

The algorithm presented here is *not* the true generalization. The true generalization would require robust set operations on subsets of $\mathbb{R}^2 \times S^1$. These subsets would either represent either (a) valid configurations (translation plus rotation) of a single polygon to place it in the container, or (b) the set of valid (non-overlapping) relative positions of one polygon with respect to another. Unfortunately, the configuration sets are three dimensions with curved surface boundary. As yet, there are no numerically robust ways to implement the necessary set operations. As we have previously stated, the branch and bound paradigm cascades these set operations in a manner that is extremely stressful numerically.

The algorithm presented here only requires set operations on polygons in two dimensions, which we know how to do robustly. It is likely that the true generalization would run faster than the current algorithm—if and when it could be implemented robustly. The algorithm presented here represents a practical tradeoff of running time for numerical robustness.

1.4. Geometric rounding

An *arrangement* of line segments subdivides the plane into a disjoint union of point *vertices*, open line segment *edges*, and open polygonal region *faces*: the connected components of the complement of the vertices and edges. *Geometric rounding* reduces the precision of the vertices in a subdivision while changing the topology (combinatorial structure) in a consistent way. Each edge becomes a polygonal chain, and so forth. The most recent versions of geometric rounding do not introduce new vertices, although they may increase the degree of existing. *Snap rounding* [14,17] can round vertices to the integer grid. *Shortest path rounding* [25,26,30,31] can also round to the non-uniform lattice of points with floating point coordinates and it has less rounding error on average even for the integer grid. Rounding to floating point rather than just integers is nice because the resulting algorithm is much more *scale invariant* in practice. [2]

We could have used snap rounding or shortest path rounding for the rotational containment algorithm, but we chose to use the algorithm as a test-bed for a new geometric rounding method: *shortest path rounding*. Shortest path rounding rounds vertices to the floating point grid, and unlike previous rounding methods, it ensures a minimum separation criterion in the result. This minimum separation allows us to avoid the use of exact arithmetic. All calculations are carried out in rounded floating point arithmetic. The downside of nearest pair rounding is that it has no proven bound on the rounding error. The other methods have constant error. In future work, we plan to investigate the error introduced by shortest path

[2] Of course, floating point arithmetic is only truly scale invariant under multiplication by powers of two, and even then only if the calculation avoids overflowing or underflowing the exponent field.

rounding, but the current work is only well suited to testing its robustness. Again, we emphasize that we could have used the other rounding methods if we wanted a guaranteed error bound on each operation. (It still would not guarantee an error bound for the entire algorithm.)

Of course, any rounding technique we choose will introduce rounding error, and rounding means uncertainty. Does that make this algorithm into a heuristic? There are many established algorithms for global minimization: linear programming, quadratic programming, and so forth. Floating point implementations of these algorithms are not called heuristics. They are just called "work for numerical analysts". For the faint of heart, we make two observations. First, each output of the containment algorithm is a local as well as a global minimum, and we compute the local minima from the original input polygons using an accurate commercial optimization library. Second, rounding can cause the algorithm to miss a local minimum entirely (by shrinking a "face" to nothing), but if this turns out to be a practical problem, one can always run the algorithm on slightly shrunken copies of the polygons and then restore them to their original size for the local minimization.

Outline. Section 2 gives the algorithms for rotational containment and minimum enclosure using the range intersection. Section 3 analyzes the range intersection and gives algorithms for constructing it and useful polygonal approximations to it. Section 4 gives the nearest pair rounding algorithm. Finally, Section 5 gives results.

2. Containment/enclosure algorithms

The rotational containment/enclosure algorithm uses a "branch and bound" approach which generalizes our previous practical translational containment algorithm [28,33]. However, this is also essentially the approach used by many mathematical programming algorithms.

The input to the containment algorithm is a set of polygons $P_1, P_2, P_3, \ldots, P_k$ and a container polygon C. In the case of minimum enclosure, the container is replaced by a set $\mathcal{C}$ of containers. A *configuration* is an assignment of translations $t_1, t_2, \ldots, t_k$ and rotations $\theta_1, \theta_2, \ldots, \theta_k$ to the polygons. To simplify the notation, it is convenient to replace the container by an additional polygon $P_0 = \overline{C}$ which is the complement of the container and to fix its translation t_0 and rotation θ_0 at $(0, 0)$ and 0, respectively.

In the case of containment, the goal is to find all configurations (or at least one configuration) with zero overlap among the polygons. Since the complement of the container is P_0, this means that $P_1, P_2, \ldots, P_k$ lie inside the container. In what follows these configurations are called the *solutions*. In the case of minimum enclosure, the goal is to find a non-overlapping layout which fits in the minimum area container $C \in \mathcal{C}$.

2.1. Abstract algorithm

We use the same language as in our translational work. A *hypothesis* is a set of constraints on the configuration. *Restriction* adds to this set of constraints in a way that is guaranteed not to eliminate any solutions. *Evaluation* attempts to find a solution within the hypothesis (that satisfies its constraints). *Subdivision* splits a hypothesis H into two sub-hypotheses H' and H'' such that all solutions in H reside in (satisfy all the constraints of) either H' or H''.

The containment algorithm first generates a root hypothesis: a set of constraints which all solutions must satisfy. It restricts this hypothesis and evaluates it. If it cannot either (a) restrict it to the empty set, or (b) find a solution, it subdivides the hypothesis and recurses on the two sub-hypotheses. A hypotheses for which evaluation is successful is called a *solution hypothesis*. If only one solution is desired, the algorithm can stop when it finds the first one. Otherwise, it generates a set of solution hypotheses. Note that we only find one solution for each solution hypothesis, not all. In the translational case, it is possible to find all solutions [27], but we have not done this for the rotational case.

The minimum enclosure algorithm acts the same as the containment algorithm, except that every time it finds a solution hypothesis, it establishes a new upper bound on the area of the minimum container. It uses this upper bound as a constraint to restrict the previously discovered solution hypotheses, and it recurses on these. Its output is a solution hypothesis with minimum area (or possibly several solution hypothesis whose solutions have the same area). To establish that the area is indeed minimal, it is necessary to run the algorithm on its output hypotheses with a slightly smaller upper bound. We first diminish the upper bound by 1%. If no new solution is found, then we roll back and try diminishing the upper bound by 0.01%, and so forth. By these means, the upper bound can be established to any degree of numerical accuracy. In theory, it might be possible to apply symbolic perturbation to the solution to establish that there is no solution for any smaller area. In practice, this is not possible since much of our algorithm is numerical.

2.2. Root hypothesis

The hypothesis for the containment algorithm takes the following form:

$$\alpha_i \leqslant \theta_i \leqslant \beta_i, \quad 0 \leqslant i \leqslant k, \tag{1}$$

$$t_j - t_i \in U_{ij}, \quad 0 \leqslant i < j \leqslant k, \tag{2}$$

where α_i and β_i are bounds on the angle and U_{ij} is a (usually non-convex) planar region. Think of each U_{ij} as a non-convex-polygon-valued variable which is initialized when the hypothesis is created and possibly changed later by restriction.

In the case of translational containment [28,33] all solutions must satisfy

$$t_j - t_i \in \overline{P_i \oplus P_j}, \quad 0 \leqslant i < j \leqslant k, \tag{3}$$

where $\oplus$ is the Minkowski sum,

$$A \oplus B = \{a + b \mid a \in A \text{ and } b \in B\}.$$

Hence, we set $U_{ij} = \overline{P_i \oplus P_j}$ initially in the root hypothesis. However, this does not work for rotational containment.

Define the *range intersection*

$$P(\alpha, \beta) = \bigcap_{\alpha \leqslant \theta \leqslant \beta} P(\theta), \tag{4}$$

where $P(\theta)$ is P rotated by angle θ.[3] The range intersection is the intersection of all copies of P as it is rotated from α to β. The following lemma provides a way of constructing a root hypothesis for rotational containment.

[3] All rotations are about the origin. It is assumed that all polygons contain the origin.

Lemma 1. *There is a solution to rotational containment for* $\alpha_i \leqslant \theta_i \leqslant \beta_i$, $0 \leqslant i \leqslant k$ *only if* (*but not if, alas*) *there is a solution to translational containment for* $P(\alpha_i, \beta_i)$, $0 \leqslant i \leqslant k$.

Proof. If $\alpha_i \leqslant \theta_i \leqslant \beta_i$, then $P(\alpha_i, \beta_i) \subseteq P(\theta_i)$. Given the solution to rotational containment, just use the same translations. □

In the case of rotational containment, we initially set

$$\alpha_i = 0, \quad \beta_i = 2\pi, \qquad 0 \leqslant i \leqslant k,$$

(which is a disk) in the root hypothesis, and set

$$U_{ij} = \overline{P_i(0, 2\pi) \oplus P_j(0, 2\pi)}, \quad 0 \leqslant i < j \leqslant k$$

(which are the complements of disks). By Eqs. (1) and (3), all solutions to containment will satisfy the constraint equations (1) and (2) for this hypothesis.

In the case of minimum enclosure, if the set $\mathcal{C}$ of containers is either (1) a set of rectangles with fixed width, or (2) a set of similar (scaled) copies of a convex polygon, then we set C equal to the element whose size (length or scale) equals a known upper bound. (For sufficiently large size, it is trivial to find a solution.) We postpone the discussion of more general sets, such as (3) *all rectangles*, until Section 2.6.

2.3. Restriction

Restriction. In our previous work on translational containment, we established two types of restriction: *geometric restriction* [7] and *linear programming restriction* [27]. These restrictions were both derived from Eq. (2), and therefore they apply in the case of rotational containment as well.

Repeatedly, for all triples h, i, j, geometric restriction sets U_{ij} to be a subset,[4] $U_{ij} \cap (U_{ih} \oplus U_{hj})$. This corresponds to the rule that "a valid placement of P_j relative to P_i must also be a valid placement of P_j relative to P_h plus a valid placement of P_h relative to P_i". It stops when no polygon diminishes in area by more than a threshold fraction. This repetition (cascading) can lead to numerical difficulties which we deal with using nearest pair rounding (Section 4). Note: for $k = 2$, geometric restriction is an exact algorithm. In this special case, it becomes essentially the same as Avnaim and Boissonnat's exact formula for this case [3,4]. If either geometric restriction or linear programming restriction generates the empty set, then there is no solution in the current hypothesis.

2.4. Evaluation

We have previously developed a practical algorithm for *rotational overlap minimization* [32]: find translations and rotations that minimizes the sum of the overlaps among $P_0, P_1, P_2, \ldots, P_k$. This algorithm can also perform *rotational compaction*: given a non-overlapping layout of $P_1, P_2, \ldots, P_k$ inside a rectangular container C, find a layout which minimizes (local minimum) of the container while keeping the width fixed. In the full paper we show how to generalize compaction to minimize the area of a scaled convex enclosure or even the area of an arbitrary rectangular container. (The latter problem is non-linear.)

[4] Setting U_{ij} to be a subset is equivalent to adding more constraints since the previous constraints must still hold.

Evaluation for the containment algorithm simply consists of running rotational overlap minimization with a set of additional constraints derived from the current hypothesis. Since compaction is based on linear programming, the additional constraints must be convex. The additional constraints are (a) the angle constraints, Eq. (1), and (b) the "relaxed" translation constraints, Eq. (2), with each U_{ij} replaced by its convex hull. [5]

For the minimum enclosure algorithm, if the algorithm finds a solution hypothesis, it compacts the non-overlapping layout into a local minimum area container. This establishes an upper bound on the area of the container which is also a local minimum.

2.5. Subdivision

Subdivision has two cases based on the output of evaluation. If the translations satisfy Eq. (2), then the output is a non-overlapping layout of the range intersections $P_i(\alpha_i, \beta_i)$ (Lemma 1). It is necessary in this case to subdivide an angle range. If the output does not satisfy Eq. (2), then the algorithms use essentially the same subdivision algorithm as the translational containment algorithm.

To subdivide an angle range, the algorithm selects the polygon P_i whose total overlap with the other polygons is maximal at its current translation and rotation (the output of evaluation). It then cuts the angle range $[\alpha_i, \beta_i]$ at its midpoint, creating two new angle ranges $[\alpha_i', \beta_i']$ and $[\alpha_i'', \beta_i'']$. To generate sub-hypothesis H', it generates the root hypothesis for $[\alpha_i, \beta_i]$ replaced by $[\alpha_i', \beta_i']$ and adds these constraints to H. Generating H'' is analogous.

If Eq. (2) is not satisfied, then the subdivision algorithm picks the pair i, j such that $t_j - t_i$ is farthest from U_{ij} (Euclidean distance). If U_{ij} has more than one component, then it sets U_{ij}' equal to the component of U_{ij} nearest to $t_j - t_i$ and it sets U_{ij}'' to the U_{ij} minus this component. If U_{ij} has only one component, then subdivision calculates the line L which is tangent to U_{ij} at the point p of its boundary nearest to $t_j - t_i$: L is perpendicular to the line from $t_j - t_i$ to p. Using line L, the algorithm cuts U_{ij} into U_{ij}' and U_{ij}''. [6] This is different (and somewhat simpler) than what we do for our translational containment algorithm [33] but this case is rare enough that it should not make much of a difference in practice. To generate sub-hypotheses H' and H'', subdivision replaces U_{ij} by either U_{ij}' or U_{ij}''.

2.6. Minimum area rectangle enclosure

The preceding sections have described the whole of the containment and minimum enclosure algorithms for the case of a rectangle with fixed width and variable length or the case of scaled copies of a fixed convex polygon. The case of minimum area rectangle with arbitrary length and width requires a more sophisticated hypothesis with $k + 2$ polygons. Given polygons $P_1, P_2, \ldots, P_k$ as input, set $P_0 = \{(x, y) \mid x \leqslant 0 \text{ or } y \leqslant 0\}$ and $P_{k+1} = \{(x, y) \mid x \geqslant \text{ or } y \geqslant 0\}$. In the root hypothesis, set $U_{0,k+1} = \{(x, y) \mid xy \leqslant A\}$; otherwise, set $U_{ij} = \overline{P_i \oplus P_j}$ as usual. Any solution under these constraints corresponds to a layout in an axis-parallel rectangle with diagonal $t_0 t_{k+1}$ and with area no greater than A. To keep everything polygonal, we approximate the hyperbola $xy = A$ by a polygon.

[5] This replacement is a means to generate a set of constraints. The values of the U_{ij} "variables" are not changed by this step.

[6] Since p lies inside the convex hull of U_{ij} (from the constraints), L cannot extend a convex hull edge, and therefore it must cut the interior of U_{ij}.

3. Range intersection

Section 2.2 gave a definition for the *range intersection* $P(\alpha, \beta)$ of a polygon P: the intersection of all copies of P as it is rotated from angle α to angle β about the origin. The range intersection is a *circular polygon*: a planar region bounded by line segments and circular arcs. In particular, these circular arcs are concentric with the origin.

The set intersection, convex hull, and even Minkowski sum of circular polygons is a circular polygon, and therefore it is possible in principle to implement all the operations used in Section 2 on circular polygons. However, generalizing the overlap minimization/compaction algorithm to circular polygons would require a switch from linear program to quadratic or convex programming. Also, it is difficult to handle the intersection of circular arcs in a numerically robust fashion, which would make it difficult to implement the intersection and Minkowski sum of circular polygons.

This section establishes that the complexity of the range intersection $P(\alpha, \beta)$ is linear in the complexity of P. It gives a way to approximate circular polygons by polygons in a way that is suitable for the containment/enclosure algorithms. In particular, the approximation is (a) a subset (b) whose error is proportional to $(\alpha - \beta)^2$ and (c) whose complexity is also linear. Finally, it gives the "practical" implementation we use to construct approximate range intersections.

3.1. Theoretical complexity

Lemma 2. *The range intersection* $P(\alpha, \beta)$ *has complexity* $O(|P|)$.

Proof. Construct a *concentric trapezoidalization* of P which is analogous to the standard trapezoidalization of P. From each vertex v of P extend an arc (centered at the origin) in each direction (clockwise and counter-clockwise) that goes into the interior of P. Terminate the arc when it hits the boundary of P. This procedure cuts P into a linear number of *concentric trapezoids*: regions bounded by two line segments and two concentric circular arcs.

The intersection $P(\alpha, \beta)$ is a rotated copy of $P(0, \gamma)$, where $\gamma = \beta - \alpha$. The set $P(0, \gamma)$ equals the set of counter-clockwise endpoints of arcs of angle γ which lie inside P. It is clear that each such arc lies in a single concentric trapezoid. Therefore, the range intersection of each concentric trapezoid can be computed independently. For a concentric trapezoid, the range intersection is simply the intersection of the two extremes. There are a number of cases to consider (see Fig. 1), but the intersection can be constructed in constant time, and it is also a concentric trapezoid. The range intersections of the concentric trapezoids meet only at their boundaries and in a nice fashion. It is easy to show that the complexity of the boundary of their union is also linear: a linear number of arcs and line segments. □

3.2. Polygonal approximations

In order to compute a polygonal approximation to $P(\alpha, \beta)$, it is necessary to perform a *radial trapezoidalization*. From each vertex v of $P(\alpha, \beta)$, extend a line towards the origin and/or away from the origin, in each direction that goes into the interior of $P(\alpha, \beta)$. Terminate a line when it hits the boundary.

Fig. 2 shows the three types of non-polygonal radial trapezoids and how to form inner polygonal approximations. The most complex case has an inner arc and outer line segment. We approximate the

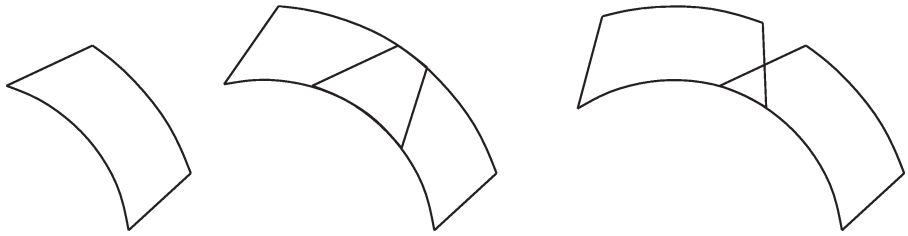

Fig. 1. Possible range intersection of a concentric trapezoid.

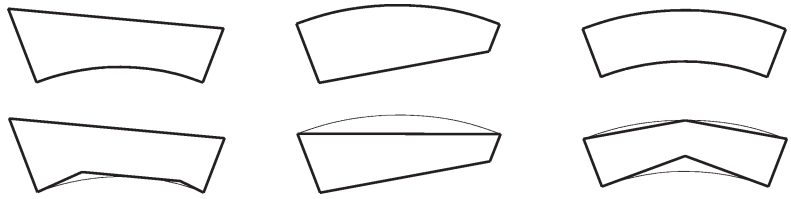

Fig. 2. Radial trapezoids and their inner polygonal approximations.

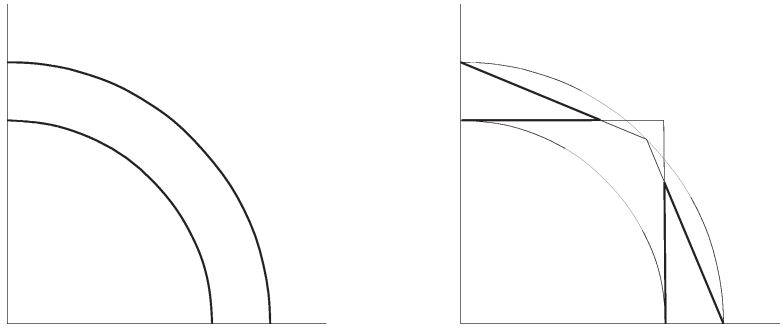

Fig. 3. Polygonal approximation of two-arc trapezoid can have different topology.

inner arc by three tangent line segments: two of them tangent at the endpoints and the middle one parallel to the outer bounding line segment. In the case of two arcs, the polygonal approximations can intersect. In this case, we lose the middle part of the arc, and the topology changes as illustrated in Fig. 3.

Lemma 3. *Let $B(r)$ be the ball of radius r. Let $Q(\alpha,\beta)$ be the polygonal approximation to the range intersection $P(\alpha,\beta)$. For $|\gamma|$ sufficiently small ($\gamma=\beta-\alpha$),*

$$Q(\alpha,\beta)\subseteq P(\alpha,\beta)\subseteq Q(\alpha,\beta)\oplus B\big(\mathrm{O}(\gamma^2)\big).$$

Proof. Clearly the approximation is a subset. It can be shown that $P(\alpha,\beta)$ can have no arc in its boundary longer than $|\gamma|$. Basic trigonometry shows that a chordal approximation to an outer arc has error at most $1-\cos(\gamma/2)\approx\gamma^2/4$ times the radius. A tangential approximation to an inner arc has error at most $1/\cos(\gamma/2)-1\approx\gamma^2/4$ times its radius.

The difference between the inner and the outer radius is at least the thickness of the thinnest concentric trapezoid of the original trapezoidalization of P. This value is independent of γ. Therefore,

for sufficiently small γ, the topological change of Fig. 3 does not occur. Each individual radial trapezoid approximation satisfies the condition of the lemma, and therefore their union does too. □

Lemma 3 is exactly the property we need to ensure that we can substitute the polygonal approximation to the range intersection for the actual range intersection in the containment/enclosure algorithms. As the containment algorithm subdivides the angle ranges, the accuracy of the approximation increases with the square of the angle range. Thus we expect to have to divide a given angle range no more than b times to obtain accuracy at least $\varepsilon = 2^{-b}$.

3.3. Practical algorithm

We briefly sketch here a "practical algorithm" we use in lieu of constructing a full circular trapezoidalization. We decompose the complement $\overline{P}$ into *angularly monotonic* regions: each ray out of the origin intersects the region in a single line segment, a points, or the empty set. This implies that a monotone component M is bounded by two functions of angle:

$$M = \{(r, \theta) \mid f(\theta) \leqslant r \leqslant g(\theta)\},$$

in polar coordinates. We compute the minimum of $f(\,)$ over an angle "window" of size α as the window rotates about the origin. Similarly, we compute the maximum of $g(\,)$ over a sliding window of angle α. We compute an inner polygonal approximation to the minimum curve and an outer polygonal approximation to the maximum curve. The two curves bound an approximation to the range union $M(0, \gamma)$ of the component M. Finally, we take the complement of the union of the range unions.

4. Nearest pair rounding

As stated in the introduction (Section 1.4), *nearest path* rounding alters planar subdivisions. After rounding, the new subdivision has a guaranteed minimum separation condition: each vertex/vertex pair and each vertex/edge pair is farther apart than a specified minimum (unless, of course, the two vertices are identical or the vertex is an endpoint of the edge). This minimum separation in turn permits the use of rounded floating point arithmetic for all operations. In particular, if vertex c lies near to edge ab, it still must lie sufficiently far away to permit us to determine on which side it lies, using only floating point arithmetic.

4.1. Rounding algorithm

Nearest pair rounding repeats a rounding step until no more rounding is necessary. Each step "rounds together" the nearest pair of non-incident features: vertex/vertex or vertex/edge. When a step computes the nearest pair, it "ignores" any vertex/edge pair c, ab if $\angle cab$ or $\angle cba$ is greater than or equal to $45°$.[7] Rounding stops when every pair of non-incident features is farther apart than some threshold and when no two edges are intersecting. We use a threshold equal to 16 times the rounding unit for double precision ($16 \cdot 10^{-52} \approx 2^{-50}$) times the diameter of the set of points and line segments.

[7] This rule prevents rounding from "bending" an edge "too much".

Fig. 4. "Worst case" for nearest pair rounding.

To round a vertex/vertex pair a, b, the algorithm moves both a and b to the midpoint v of segment ab. Any edge which had a or b as an endpoint now has v as that endpoint. Edge ab, if it existed, is deleted. To round a vertex/edge pair c, ab, the algorithm constructs v, the point halfway from c to ab. It moves c to v (altering edges which have c as an endpoint), and it replaces ab by av and vc.

4.2. Proof of correctness

Fig. 4 illustrates a "worst case" for nearest pair rounding. Since $\angle cab = 45°$, the vertex/edge pair c, ab is ignored, and $\text{dist}(c, ab)$ is not measured. Suppose $\text{dist}(a, d)$ is the minimum *measured* separation. (Actually, $\text{dist}(c, d)$ would have to be smaller, as the figure indicates, but we will not bother proving that.) Hence, $\text{dist}(a, d) \leqslant \text{dist}(a, c)$. By the 45° rule, $\text{dist}(a, c) \leqslant \sqrt{2}\,\text{dist}(c, ab)$. Therefore, rounding a to e, the midpoint of a and d, moves no point on ab farther than $(\sqrt{2}/2)\text{dist}(c, ab)$. The distance from c to the rounded segment eb is still definitely positive:

$$\text{dist}(c, eb) \geqslant \left(1 - \sqrt{2}/2\right)\text{dist}(c, ab) > 0.25\,\text{dist}(c, ab).$$

Theorem 4. *Nearest pair rounding converges without introducing any new intersections.*

Proof. As indicated in the discussion above, the key to nearest path rounding is that the perturbation it introduces is at most $3/4$ of the minimum distance between non-incident features. Therefore, applying the rounding step cannot cause other non-incident features to "collide". Because we ignore vertex/edge pairs unless the angles with the endpoints are less than 45°, the measured minimum feature separation may be $\sqrt{2}$ times the actual minimum. However, rounding moves features to a midpoint, and therefore the actual perturbation is at most $\sqrt{2}/2 < 3/4$ times the actual minimum feature separation.

Consider a partial order on sets of line segments, ordered on: (1) number of edge/edge intersections, (2) number of vertices, (3) length of longest edge, (4) length of second longest edge, and so forth. Adding an intersection point decreases (1). Rounding two vertices together decreases (2) without increasing (1). Because of the 45° rule, breaking an edge forms two shorter edges without increasing (1) or (2). □

4.3. Experimental rounded arithmetic arrangement algorithm

In practice, we use floating point arithmetic for all operations and apply nearest pair rounding without using exact arithmetic to measure distances. Because the rounding step has $1 - 3/4 = 25\%$ "leeway" for error, we expect it to be numerically stable. So far, it has been stable even after hundreds of cascaded operations.

Nearest pair rounding can also be applied to a set of intersecting line segments. If, after rounding is complete, any pair of edges are intersecting, a new vertex is created as near as possible to their actual intersection, and rounding resumes. Thus, nearest pair rounding becomes an arrangement algorithm, and that is all one needs to implement set operations on polygonal regions and the Minkowski sum.

We have implemented the arrangement algorithm in a straightforward fashion in rounded double precision arithmetic, and so far this implementation has worked well, even under the containment algorithm “torture test”. This is not a proof of robustness, but it is a very strong experimental result.

5. Results

All algorithms were implemented in C++. Overlap minimization code: 9100 lines. Polygon set operations: 2700 lines. Containment/Enclosure algorithms: 1200 lines. Overlap minimization uses CPLEX 4.0 by CPLEX Optimization, Inc. Experiments were run on an SGI Powerchallenge L which has roughly the same speed as a 400 MHz Intel Pentium II.

In each case, the target accuracy was one part in 2^{11}. Whenever a successful layout was found, it was compacted using rotational compaction. The new target length was set at $1 - 2^{-11}$ times this length. The first set of tests allowed a full 360° rotation for all but the largest polygon, which was limited to 180°. (By symmetry, this restriction will still yield the optimum length.) Figs. 5, 6 and 8 show all the successful layouts that were found, including the optimal one.

The second set of tests limited each range of rotation to 22.5°.

5.1. Full rotation

The 2-polygon test yielded an optimal (to one part in 2^{11}) solution of length 1385.11. The polygons had 71 and 64 vertices. The algorithm visited 30 hypotheses and ran in 3 minutes. See Fig. 5.

The 3-polygon test yielded an optimal solution of length 2081.84. The polygons had 71, 64 and 65 vertices. The algorithm visited 1420 hypotheses and ran in 130 minutes. See Fig. 6.

The 4-polygon test yielded a solution of length 2699. The polygons had 71, 64, 65, 60 vertices. The algorithm visited over 2700 hypotheses in over 21 hours, but it still had not verified this was optimum, and it had only visited fraction of the search space. See Fig. 7.

The industrial data has a large number of vertices. To see if this caused the long running time, we tried an example using polygons with only 7 vertices. We first limited the search to depth 3 (45°). After about 24 hours, the algorithm found the solutions shown in Fig. 8. To verify the third solution, we ran it without a bound on the depth. After a day it had only visited a miniscule fraction of the solution space.

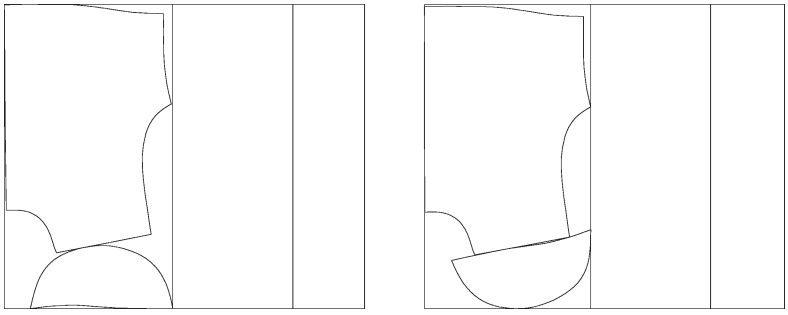

Fig. 5. Lengths: 1403.7 (non-optimal) and 1385.11 (optimal).

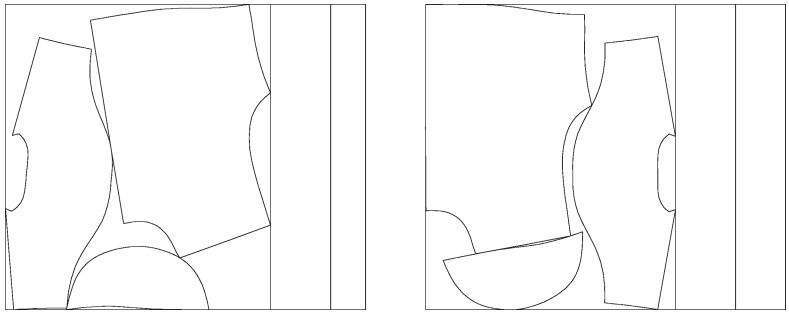

Fig. 6. Lengths: 2208.39 (non-optimal) and 2081.84 (optimal).

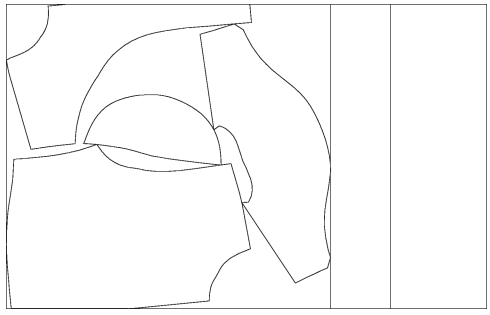

Fig. 7. Length: 2699.

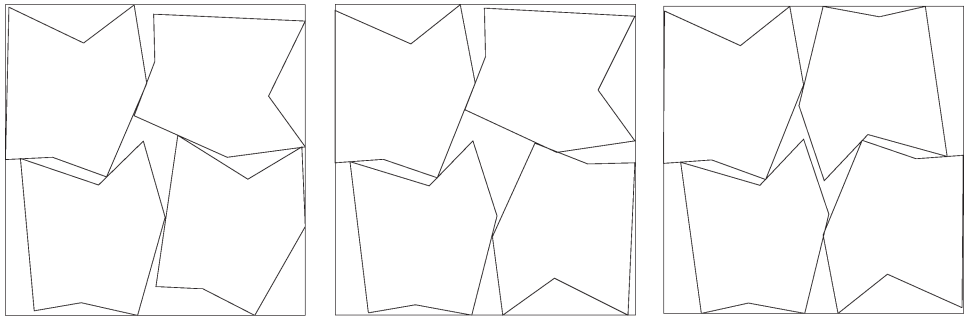

Fig. 8. Heights: 515.54, 515.21, 509.48.

5.2. Limited angle

We ran the same 3 and 4-polygon tests with a limited angle range of 22.5° ($[-11.25, 11.25]$). Fig. 9 shows the output of the 3-polygon and 4-polygon tests. The 3-polygon test yielded an optimal solution of length 2240.72. The algorithm visited 64 hypotheses and ran in 4 minutes. The 4-polygon test yielded an optimal solution of length 2856.65. The algorithm visited 590 hypotheses and ran in 90 minutes.

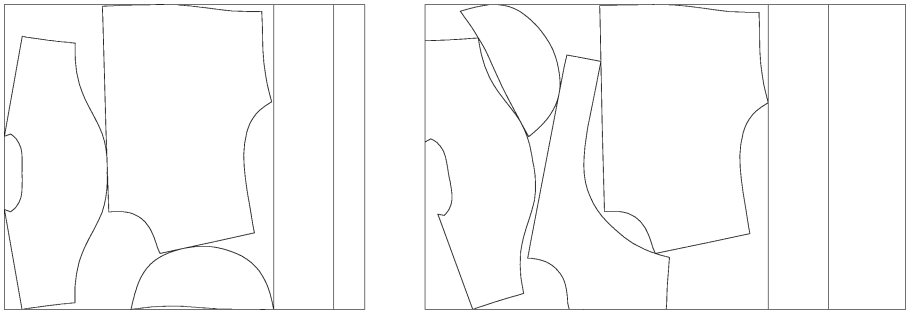

Fig. 9. Limited angle: 3-polygon length 2240.72, 4-polygon length 2856.65.

6. Conclusions

Nearest path rounding appears to be robust, even when implemented in rounded floating point arithmetic. It remains to find an experiment that accurately measures the error that rounding introduces. It is our hope that, unlike shortest path rounding and snap rounding, nearest path rounding will generalize to circular arcs, making it unnecessary to approximate circular arcs.

At present, the range intersection method appears to be practical for two or three-polygon minimum enclosure. It does not appear to be practical for four polygons, unless one is willing to live without the guarantee of optimality. For a limited range of angles, it might be practical for up to the four-polygon case: 90 minutes is a bit slow, but additional tinkering and faster hardware will bring it within the practical range. We did not even try five polygons because it was clear that the time would be much larger.

Some modifications might speed up the algorithm. For instance, it can search the hypothesis space for each individual polygon and then each pair before considering three at time. That might speed up the three-polygon example somewhat because it appeared to sometimes make a bad choice for on polygon's angle and then spend much time investigating all hypotheses with this angle.

Another conclusion is that it is worth pursuing the "true generalization" of translational containment described in Section 1.3. This will involve performing complex set operations on curved three dimensional objects. It will probably be difficult to make these operations robust, but the current results demonstrate that these algorithms would be worth the effort. For instance, in the special case of two polygons, the true generalization would be essentially the same as Avnaim and Boissonnat's formula and thus solve this case exactly. Our experience with translational work would indicate that it could solve at least the three-polygon case and possibly the four or five-polygon case also in a reasonable amount of time.

It is possible that a multidimensional method might be used to narrow the angle range, and then the current algorithm could finish the job. Also, one should not forget that apparel manufacturers only allow small tilts, usually less than 10°. This means that the containment algorithm of this paper might be practical for applications in this industry.

References

[1] P. Agarwal, N. Amenta, M. Sharir, Largest placement of one convex polygon inside another, in: Proceedings of the 2nd Workshop on Algorithmic Foundations of Robotics, July 1996.

[2] P.K. Agarwal, B. Aronov, M. Sharir, Motion planning for a convex polygon in a polygonal environment, in preparation, 1992.
[3] F. Avnaim, Placement et déplacement de formes rigides ou articulées, Ph.D. Thesis, Université de Franche-Comté, France, 1989.
[4] F. Avnaim, J. Boissonnat, Polygon placement under translation and rotation, in: Proceedings of the 5th Annual Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science, Vol. 294, Springer, 1988, pp. 322–333.
[5] C. Bounsaythip, S. Maouche, Irregular shape nesting and placing with evolutionary approach, in: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, IEEE, New York, NY, 12–15 October 1997, p. 5.
[6] B. Chazelle, The polygon containment problem, in: F. Preparata (Ed.), Advances in Computing Research, Volume 1: Computational Geometry, JAI Press, Inc., Greenwich, CT, 1983, pp. 1–33.
[7] K. Daniels, Containment algorithms for nonconvex polygons with applications to layout, Ph.D. Thesis, Harvard University, Cambridge, MA, 1995.
[8] K. Daniels, V. Milenkovic, Column-based strip packing using ordered and compliant containment, in: Proc. 1st ACM Workshop on Appl. Comput. Geom. 1996, pp. 33–38.
[9] K. Daniels, V.J. Milenkovic, Multiple translational containment, Part I: An approximate algorithm, Algorithmica 19 (1997) 148–182.
[10] K.A. Dowsland, W.B. Dowsland, Packing problems, European J. Oper. Res. 56 (1992) 2–14.
[11] K.A. Dowsland, W.B. Dowsland, Solution approaches to irregular nesting problems, European J. Oper. Res. 84 (3) (1995) 506–521.
[12] H. Dyckhoff, A typology of cutting and packing problems, European J. Oper. Res. 44 (1990) 145–159.
[13] R.M.S. Abd El-Aal, A new technique for nesting irregular shapes based on rectangular modules, Current Advances in Mechanical Design and Production 1996 6 (1996) 533–540.
[14] D.H. Greene, F.F. Yao, Finite-resolution computational geometry, in: Proc. 27th Annu. IEEE Sympos. Found. Comput. Sci., 1986, pp. 143–152.
[15] R. Grinde, T. Cavalier, Containment of a single polygon using mathematical programming, Technical Report IMSE Working Paper 92-164, The Pennsylvania State University, Department of Industrial and Management Systems Engineering, 1993.
[16] R.B. Grinde, T.M. Cavalier, A new algorithm for the two-polygon containment problem, Comput. Oper. Res., to appear, 1996.
[17] L. Guibas, D. Marimont, Rounding arrangements dynamically, in: Proc. 11th Annu. ACM Sympos. Comput. Geom., 1995, pp. 190–199.
[18] G.C. Han, S.J. Na, Two-stage approach for nesting in two-dimensional cutting problems using neural network and simulated annealing, J. Engrg. Manufacture 210 (B6) (1996) 509–519.
[19] R. Heckmann, T. Lengauer, Computing upper and lower bounds on textile nesting problems, Lecture Notes in Computer Science, Vol. 1136, 1996, pp. 392–405.
[20] K. Hoang, Aspects in automatic nesting of irregular shapes, in: Proceedings of the SPIE – The International Society for Optical Engineering, Vol. 2589, 23–26 October 1995, pp. 234–241.
[21] P. Jain, P. Fenyes, R. Richter, Optimal blank nesting using simulated annealing, J. Mech. Design 114 (1) (1992) 160–165.
[22] H.J. Lamousin, W.N. Waggenspack, G.T. Dobson, Nesting of complex 2-d parts within irregular boundaries, J. Mech. Design 118 (4) (1996) 615.
[23] Z. Li, Compaction algorithms for nonconvex polygons and their applications, Ph.D. Thesis, Harvard University, Cambridge, MA, 1994.
[24] Z. Li, V. Milenkovic, Compaction and separation algorithms for nonconvex polygons and their applications, European J. Oper. Res. 84 (1995) 539–561.

[25] V. Milenkovic, Double precision geometry: a general technique for calculating line and segment intersections using rounded arithmetic, in: Proc. 30th Annu. IEEE Sympos. Found. Comput. Sci., 1989, pp. 500–505.
[26] V. Milenkovic, Rounding face lattices in the plane, in: Abstracts 1st Canad. Conf. Comput. Geom., 1989, p. 12.
[27] V. Milenkovic, Translational polygon containment and minimal enclosure using linear programming based restriction, in: Proc. 28th Annu. ACM Sympos. Theory Comput. (STOC'96), 1996, pp. 109–118.
[28] V. Milenkovic, K. Daniels, Translational polygon containment and minimal enclosure using geometric algorithms and mathematical programming, Technical Report 25-95, Center for Research in Computing Technology, Division of Applied Sciences, Harvard University, Cambridge, MA, 1995.
[29] V.J. Milenkovic, Multiple translational containment, Part II: Exact algorithms, Algorithmica 19 (1997) 183–218.
[30] V.J. Milenkovic, Practical methods for set operations on polygons using exact arithmetic, in: Proc. 7th Canad. Conf. Comput. Geom., 1995, pp. 55–60.
[31] V.J. Milenkovic, Shortest path rounding, Algorithmica, in press, 1998.
[32] V.J. Milenkovic, Rotational polygon overlap minimization, Computational Geometry 10 (4) (1998) 305–318.
[33] V.J. Milenkovic, K. Daniels, Translational polygon containment and minimal enclosure using mathematical programming, Intern. Trans. Oper. Res., in press, 1999.
[34] Y.K.D.V. Prasad, A set of heuristic algorithms for optimal nesting of two-dimensional irregularly shaped sheet-metal blanks, Computers in Industry 24 (1) (1994) 55–70.
[35] Y.K.D.V. Prasad, S. Somasundaram, K.P. Rao, A sliding algorithm for optimal nesting of arbitrarily shaped sheet metal blanks, Internat. J. Production Res. 33 (6) (1995) 1505–1520.
[36] P.E. Sweeney, E.R. Paternoster, Cutting and packing problems: A categorized, application-oriented research bibliography, J. Oper. Res. Soc. 43 (7) (1992) 691–706.
[37] J.Y. Wang, D.Y. Liu, E.W. Lee, T.H. Koh, An algorithm for nesting patterns in apparel, in: Computer Integrated Manufacturing International Conference, Vol. 1, World Scientific, 1995, pp. 377–384.
[38] P.F. Whelan, B.G. Batchelor, Automated packing systems: review of industrial implementations, in: Proceedings of the SPIE – The International Society for Optical Engineering, Vol. 2064, SPIE, 1993, pp. 358–369.