Polygon Simplification by Minimizing Convex Corners

Yeganeh Bahoo[1], Stephane Durocher[1*], J. Mark Keil[2], Saeed Mehrabi[3], Sahar Mehrpour[1], and Debajyoti Mondal[1]

[1] Department of Computer Science, University of Manitoba, Winnipeg, Canada
[2] Department of Computer Science, University of Saskatchewan, Saskatoon, Canada
[3] Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada
{bahoo,durocher,mehrpour,jyoti}@cs.umanitoba.ca,
keil@cs.usask.ca, smehrabi@uwaterloo.ca

Abstract. Let P be a polygon with $r > 0$ reflex vertices and possibly with holes. A subsuming polygon of P is a polygon P' such that $P \subseteq P'$, each connected component R' of P' subsumes a distinct component R of P, i.e., $R \subseteq R'$, and the reflex corners of R coincide with the reflex corners of R'. A subsuming chain of P' is a minimal path on the boundary of P' whose two end edges coincide with two edges of P. Aichholzer et al. proved that every polygon P has a subsuming polygon with $O(r)$ vertices. Let $\mathcal{A}_e(P)$ (resp., $\mathcal{A}_v(P)$) be the arrangement of lines determined by the edges (resp., pairs of vertices) of P. Aichholzer et al. observed that a challenge of computing an optimal subsuming polygon P'_{min}, i.e., a subsuming polygon with minimum number of convex vertices, is that it may not always lie on $\mathcal{A}_e(P)$. We prove that in some settings, one can find an optimal subsuming polygon for a given simple polygon in polynomial time, i.e., when $\mathcal{A}_e(P'_{min}) = \mathcal{A}_e(P)$ and the subsuming chains are of constant length. In contrast, we prove the problem to be NP-hard for polygons with holes, even if there exists some P'_{min} with $\mathcal{A}_e(P'_{min}) = \mathcal{A}_e(P)$ and subsuming chains are of length three. Both results extend to the scenario when $\mathcal{A}_v(P'_{min}) = \mathcal{A}_v(P)$.

1 Introduction

Polygon simplification is well studied in computational geometry, with numerous applications in cartographic visualization, computer graphics and data compression [8, 9]. Techniques for simplifying polygons and polylines have appeared in the literature in various forms. Common goals of these simplification algorithms include to preserve the shape of the polygon, to reduce the number of vertices, to reduce the space requirements, and to remove noise (extraneous bends) from the polygon boundary (e.g., [2, 4, 5]). In this paper we consider a specific version of polygon simplification introduced by Aichholzer et al. [1], which keeps reflex corners intact, but minimizes the number of convex corners. Aichholzer et al. showed that such a simplification can help achieve faster solutions for many geometric problems such as answering shortest path queries, computing Voronoi diagrams, and so on.

Let P be a polygon with r reflex vertices and possibly with holes. A *reflex corner* of P consists of three consecutive vertices u, v, w on the boundary of P such that the

* Work of the author is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).

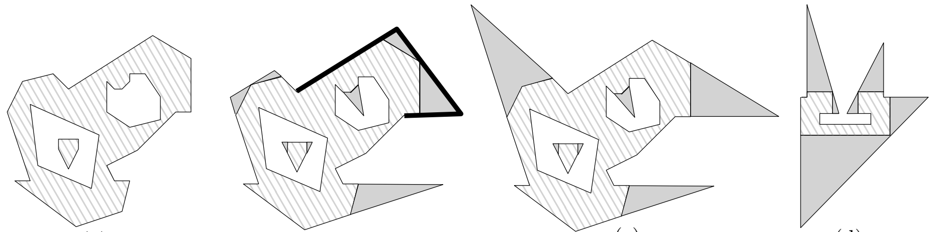

Fig. 1. (a) A polygon P, where the polygon is filled and the holes are empty regions. (b) A subsuming polygon P', where P' is the union of the filled regions. A subsuming chain is shown in bold. (c) A min-convex subsuming polygon P'_{min}, where $\mathcal{A}_e(P'_{min}) = \mathcal{A}_e(P)$. (d) A polygon P such that for any min-convex subsuming polygon P'_{min}, $\mathcal{A}_e(P) \neq \mathcal{A}_e(P'_{min})$.

angle $\angle uvw$ inside P is more than $180°$. We refer the vertex v as a *reflex* vertex of P. The vertices of P that are not reflex are called *convex* vertices. By a *component* of P, we refer to a connected region of P. A polygon P' *subsumes* P if $P \subseteq P'$, each component R' of P' subsumes a distinct component R of P, i.e., $R \subseteq R'$, and the reflex corners of R coincide with the reflex corners of R'. A *k-convex subsuming polygon P'* contains at most k convex vertices. A *min-convex subsuming polygon* is a subsuming polygon that minimizes the number of convex vertices. Figure 1(a) illustrates a polygon P, and Figures 1(b) and (c) illustrate a subsuming polygon and a min-convex subsuming polygon of P, respectively. A *subsuming chain* of P' is a minimal path on the boundary of P' whose end edges coincide with a pair of edges of P, as shown in Figure 1(b).

Aichholzer et al. [1] showed that for every polygon P with n vertices, $r > 0$ of which are reflex, one can compute in linear time a subsuming polygon P' with at most $O(r)$ vertices. Note that although a subsuming polygon with $O(r)$ vertices always exists, no polynomial-time algorithm is known for computing a min-convex subsuming polygon. Finding an optimal subsuming polygon seems challenging since it does not always lie on the arrangement of lines $\mathcal{A}_e(P)$ (resp., $\mathcal{A}_v(P)$) determined by the edges (resp., pairs of vertices) of the input polygon. Figure 1(c) illustrates an optimal polygon P'_{min} for the polygon P of Figure 1(a), where $\mathcal{A}_e(P'_{min}) = \mathcal{A}_e(P)$. On the other hand, Figure 1(d) shows that a min-convex subsuming polygon may not always lie on $\mathcal{A}_e(P)$ or $\mathcal{A}_v(P)$. Note that the input polygon of Figure 1(d) is a *simple polygon*, i.e., it does not contain any hole. Hence determining min-convex subsuming polygons seems challenging even for simple polygons. In fact, Aichholzer et al. [1] posed an open question that asks to determine the complexity of computing min-convex subsuming polygons, where the input is restricted to simple polygons.

Let P be a simple polygon. In this paper we show that if there exists a min-convex subsuming polygon P'_{min} such that $\mathcal{A}_e(P'_{min}) = \mathcal{A}_e(P)$ and the subsuming chains of P'_{min} are of constant length, then one can compute such an optimal subsuming polygon in polynomial time. In contrast, if P contains holes, then we prove the problem to be NP-hard. The hardness result holds even when the min-convex subsuming polygon P'_{min} lies on the arrangement $\mathcal{A}_e(P)$, and the length of every subsuming chain of P'_{min} is three. Both results extend to the scenario when $\mathcal{A}_v(P'_{min}) = \mathcal{A}_v(P)$.

The rest of the paper is organised as follows. In Section 2 we describe the techniques for computing subsuming polygons. Section 3 includes the NP-hardness result. Finally, Section 4 concludes the paper discussing directions to future research.

2 Computing Subsuming Polygons

In this section we show that for any simple polygon P, if there exists a min-convex subsuming polygon P_{min} such that $\mathcal{A}_e(P) = \mathcal{A}_e(P'_{min})$ and the subsuming chains are of length at most t, then one can compute an optimal polygon in $O(t^{O(1)}n^{f(t)})$ time. Therefore, if $t = O(1)$, then the time complexity of our algorithm is polynomial in n. We first present definitions and preliminary results on outerstring graphs, which will be an important tool for computing subsuming polygons.

2.1 Independent Set in Outerstring Graphs

A graph G is a *string graph* if it is an intersection graph of a set of simple curves in the plane, i.e., each vertex of G is a mapped to a curve (string), and two vertices are adjacent in G if and only if the corresponding curves intersect. G is an *outerstring graph* if the underlying curves lie interior to a simple cycle C, where each curve intersects C at one of its endpoints. Figure 2(a) illustrates an outerstring graph and the corresponding arrangement of curves. Later in our algorithm, the polygon will correspond to the cycle of an outerstring graph, and some polygonal chains attached to the boundary of the polygon will correspond to the strings of that outerstring graph.

A set of strings is called *independent* if no two strings in the set intersect, the corresponding vertices in G are called an independent set of vertices. Let G be a weighted outerstring graph with a set $\mathcal{T}$ of weighted strings. A *maximum weight independent set* $\texttt{MWIS}(\mathcal{T})$ (resp., $\texttt{MWIS}(G)$) is a set of independent strings $T \subseteq \mathcal{T}$ (resp., vertices) that maximizes the sum of the weights of the strings in T. Observe that $\texttt{MWIS}(\mathcal{T})$ is also a maximum weight independent set $\texttt{MWIS}(G)$ of G. By $|\texttt{MWIS}(G)|$ we denote the weight of $\texttt{MWIS}(G)$.

Let $\Gamma(G)$ be the arrangement of curves that corresponds to G, e.g., see Figure 2(a). Let R be a geometric representation of $\Gamma(G)$, where C is represented as a simple polygon P, and each curve is represented as a simple polygonal chain inside P such that one of its endpoints coincides with a distinct vertex of P. Keil et al. [6] showed that given a geometric representation R of G, one can compute a maximum weight independent set of G in $O(s^3)$ time, where s is the number of line segments in R.

Theorem 1 (Keil et al. [6]). *Given the geometric representation R of a weighted outerstring graph G, there exists a dynamic programming algorithm that computes a maximum weight independent set of G in $O(s^3)$ time, where s is the number of straight line segments in R.*

Figure 2(b) illustrates a geometric representation R of some G, where each string is represented with at most 4 segments. Keil et al. [6] observed that any maximum weight independent set of strings can be triangulated to create a triangulation P_t of P, as shown in Figure 2(c). Let $\mathcal{T}$ be the strings in R. Then the problem of finding $\texttt{MWIS}(\mathcal{T})$ can be

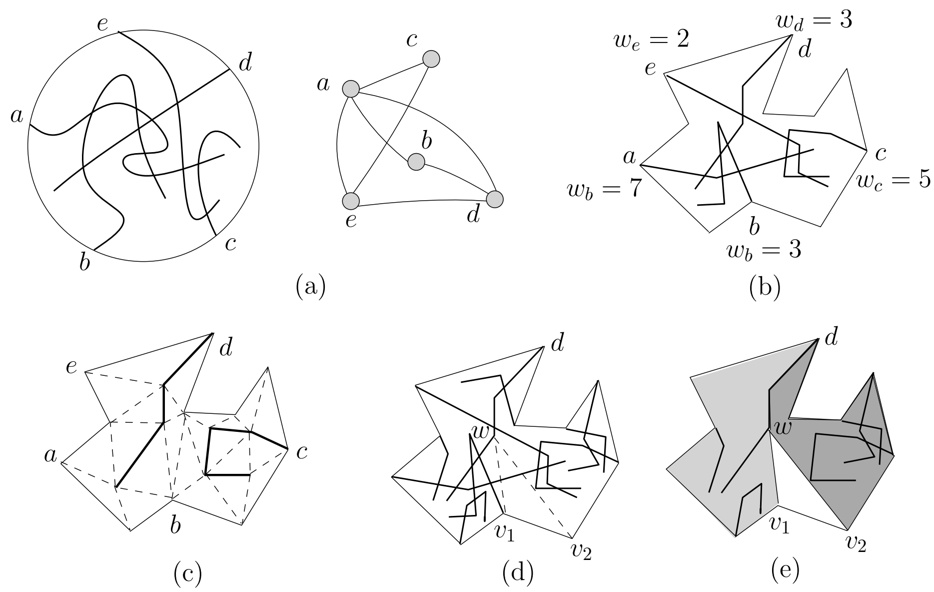

Fig. 2. (a) Illustration for G and $\Gamma(G)$. (b) A geometric representation R of G. (c) A triangulated polygon obtained from an independent set of G. (d)–(e) Dynamic programming to find maximum weight independent set.

solved by dividing the problem into subproblems, each described using only two points of R. We illustrate how the subproblems are computed very briefly using Figure 2(d). Let $P(v_1, v_2)$ be the problem of finding $\texttt{MWIS}(\mathcal{T}_{v_1,v_2})$, where $\mathcal{T}_{v_1,v_2}$ consists of the strings that lie to the left of v_1v_2. Let wv_1v_2 be a triangle in P_t, where w is a point on some string d inside $P(v_1, v_2)$; see Figure 2(d). Since P_t is a triangulation of the maximum weight string set, d must be a string in the optimal solution. Hence $P(v_1, v_2)$ can be computed from the solution to the subproblems $P(v_1, w)$ and $P(w, v_2)$, as shown in Figure 2(e). Keil et al. [6] showed that there are only a few different cases depending on whether the points describing the subproblems belong to the polygon or the strings. We will use this idea of computing $\texttt{MWIS}(\mathcal{T})$ to compute subsuming polygons.

2.2 Subsuming Polygons via Outerstring Graphs

Let $P = (v_0, v_1, \ldots, v_{n-1})$ be a simple polygon with n vertices, $r > 0$ of which are reflex vertices. A *convex chain* of P is a path $C_{ij} = (v_i, v_{i+1}, \ldots, v_{j-1}, v_j)$ of strictly convex vertices, where the indices are considered modulo n.

Let $P' = (w_0, w_1, \ldots, w_{m-1})$ be a subsuming polygon of P, where $\mathcal{A}_e(P') = \mathcal{A}_e(P)$, and the subsuming chains are of length at most t. Let $C'_{qr} = (w_q, \ldots, w_r)$ be a subsuming chain of P'. Then by definition, there is a corresponding convex chain C_{ij} in P such that the edges (v_i, v_{i+1}) and (v_{j-1}, v_j) coincide with the edges (w_q, w_{q+1}) and (w_{r-1}, w_r). We call the vertex v_i the *left support* of C'_{qr}. Since $\mathcal{A}_e(P') = \mathcal{A}_e(P)$, the

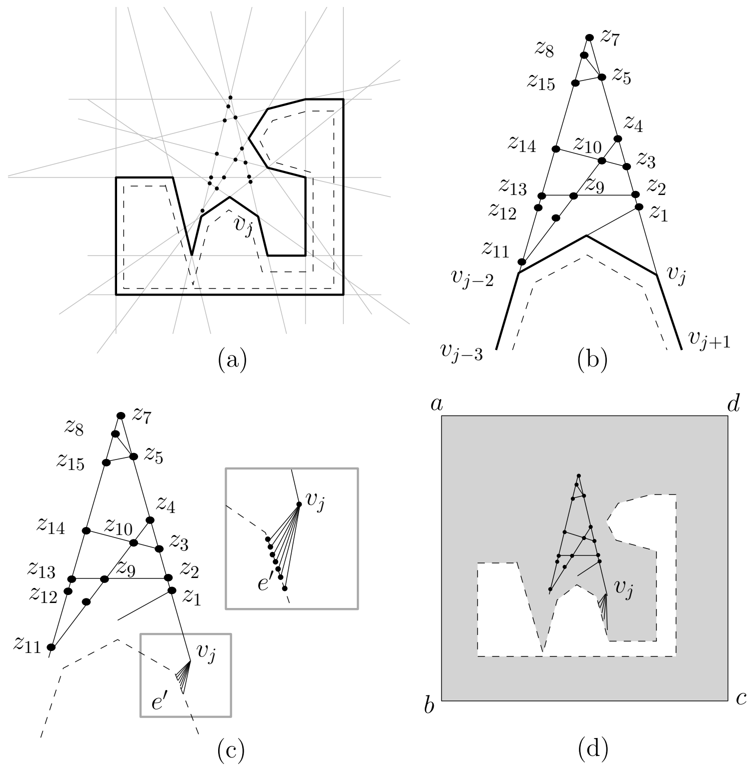

Fig. 3. (a) Illustration for the polygon P (in bold), $\mathcal{A}_e(P)$ (in gray), and Q (in dashed lines). (b) Chains of v_j. (c) Attaching the strings to Q. (d) Dynamic programming inside the gray region.

chain C'_{qr} must lie on $\mathcal{A}_e(P)$. Moreover, since P' is a min-convex subsuming polygon, the number of vertices in C'_{qr} would be at most the number of vertices in C_{ij}.

We claim that the number of paths in $\mathcal{A}_e(P)$ from v_i to v_j is $O(n^t)$. Since t is an upper bound on the length of the subsuming chains, any subsuming chain can have at most $(t-1)$ line segments. Since there are only $O(n)$ straight lines in the arrangement $\mathcal{A}_e(P)$, there can be at most n^j paths of j edges, where $1 \leq j \leq t-1$. Consequently, the number of candidate chains that can subsume C_{ij} is $O(n^t)$.

Lemma 1. *Given a simple polygon P with n vertices, every convex chain C of P has at most $O(n^t)$ candidate subsuming chains in $\mathcal{A}_e(P)$, each of length at most t.*

In the following we construct an outerstring graph using these candidate subsuming chains. We first compute a simple polygon Q interior to P such that for each edge e in P, there exists a corresponding edge e' in Q which is parallel to e and the perpendicular distance between e and e' is ϵ, as shown in dashed line in Figure 3(a). We choose ϵ sufficiently small[4] such that for each component w of P, Q contains exactly one component inside w. We now construct the strings. Let v_j be a convex corner of P. Let S_j

[4] Choose $\epsilon = \delta/3$, where δ is the distance between the closest visible pair of boundary points.

be the set of candidate subsuming chains such that for each chain in S_j, the left support of the chain appears before v_j while traversing the unbounded face of P in clockwise order. For example, the subsuming chains that correspond to v_j are (v_{j-2}, z_1, v_{j+1}), $(v_{j-3}, z_{13}, z_2, v_{j+1})$, $(v_{j-3}, z_{14}, z_3, v_{j+1})$, $(v_{j-3}, z_{11}, z_4, v_{j+1})$, $(v_{j-3}, z_{15}, z_5, v_{j+1})$, $(v_{j-3}, z_8, z_5, v_{j+1})$, (v_{j-3}, z_7, v_{j+1}), as shown in Figure 3(b). For each of these chains, we create a unique endpoint on the edge e' of Q, where e' corresponds to the edge $v_j v_{j+1}$ in P, as shown in Figure 3(c). We then attach these chains to Q by adding a segment from v_j to its unique endpoint on Q.

We attach the chains for all the convex vertices of P to Q. Later we will use these chains as the strings of an outerstring graph. We then assign each chain a weight, which is the number of convex vertices of P it can reduce. For example in Figure 3(b), the weight of the chain $(v_{j-3}, z_8, z_5, v_{j+1})$ is one.

Although the strings are outside of the simple cycle, it is straightforward to construct a representation with all the strings inside a simple cycle Q: Consider placing a dummy vertex at the intersection points of the arrangement, and then find a straight-line embedding of the resulting planar graph such that the boundary of Q corresponds to the outerface of the embedding. Consequently, Q and its associated strings correspond to an outerstring graph representation R. Let G be the underlying outerstring graph. We now claim that any $\texttt{MWIS}(G)$ corresponds to a min-convex subsuming polygon of P.

Lemma 2. *Let P be a simple polygon, where there exists a min-convex subsuming polygon that lies on $\mathcal{A}_e(P)$, and let G be the corresponding outerstring graph. Any maximum weight independent set of G yields a min-convex subsuming polygon of P.*

Proof. Let T be a set of strings that correspond to a maximum weight independent set of G. Since T is an independent set, the corresponding subsuming chains do not create edge crossings. Moreover, since each subsuming chain is weighted by the number of convex corners it can remove, the subsuming chains corresponding to T can remove $|\texttt{MWIS}(G)|$ convex corners in total.

Assume now that there exists a min-convex subsuming polygon that can remove at least k convex corners. The corresponding subsuming chains would correspond to an independent set T' of strings in G. Since each string is weighted by the number of convex corners the corresponding subsuming chain can remove, the weight of T' would be at least k. □

2.3 Time Complexity

To construct G, we first placed a dummy vertex at the intersection points of the chains, and then computed a straight-line embedding of the resulting planar graph such that all the vertices of Q are on the outerface. Therefore, the geometric representation used at most nt edges to represent each string. Since each convex vertex of P is associated with at most $O(n^t)$ strings, there are at most $n \times O(n^t)$ strings in G. Consequently, the total number of segments used in the geometric representation is $O(tn^{2+t})$. A subtle point here is that the strings in our representation may partially overlap, and more than three strings may intersect at one point. Removing such degeneracy does not increase the asymptotic size of the representation. Finally, by Theorem 1, one can compute the optimal subsuming polygon in $O(t^3 n^{6+3t})$ time.

The complexity can be improved further to as follows. Let $abcd$ be a rectangle that contains all the intersection points of $\mathcal{A}_e(P)$. Then every optimal solution can be extended to a triangulation of the closed region between $abcd$ and Q. Figure 3(d) illustrates this region in gray. We now can apply a dynamic programming similar to Section 2.1 to compute the maximum weight independent string set, where each subproblem finds a maximum weight set inside some subpolygon. Each such subpolygon can be described using two points v_1, v_2, each lying either on Q or on some string, and a subset of $\{a, b, c, d\}$ that helps enclosing the subpolygon.

Since there are $n \times O(n^t)$ strings, each containing at most t points, the number of vertices in the geometric representation is $O(tn^{1+t})$. Therefore, the size of the dynamic programming table is $O(tn^{1+t}) \times O(tn^{1+t}) \times O(1)$. Since there can be at most $O(tn^{1+t})$ candidate triangles v_1v_2w, we take $O(tn^{1+t})$ time to fill an entry of the table. Hence the dynamic program takes at most $O(t^3n^{3+3t})$ time in total.

Theorem 2. *Given a simple polygon P with n vertices such that there exists a min-convex subsuming polygon that lie on $\mathcal{A}_e(P)$ and the subsuming chains are of length at most t, one can compute such a min-convex subsuming polygon in $O(t^3n^{3+3t})$ time.*

2.4 Generalizations

We can generalize the results for any given line arrangements. However, such a generalization may increase the time complexity. For example, consider the case when the given line arrangement is $\mathcal{A}_v(P)$, which is determined by the pairs of vertices of P. Since we now have $O(n^2)$ lines in the arrangement $\mathcal{A}_v(P)$, the time complexity increases to $O(t^3(n^2)^{3+3t})$, i.e., $O(t^3n^{6+6t})$.

3 NP-hardness of Min-Convex Subsuming Polygon

In this section we prove that it is NP-hard to find a subsuming polygon with minimum number of convex vertices. We denote the problem by MIN-CONVEX-SUBSUMING-POLYGON. We reduce the NP-complete problem monotone planar 3-SAT [3], which is a variation of the 3-SAT problem as follows: Every clause in a monotone planar 3-SAT consists of either three negated variables (*negative clause*) or three non-negated variables (*positive clause*). Furthermore, the bipartite graph constructed from the variable-clause incidences, admits a planar drawing such that all the vertices corresponding to the variables lie along a horizontal straight line l, and all the vertices corresponding to the positive (respectively, negative) clauses lie above (respectively, below) l. The problem remains NP-hard even when each variable appears in at most four clauses [7].

The idea of the reduction is as follows. Given an instance of a monotone planar 3-SAT I with variable set X and clause set C, we create a corresponding instance $\mathcal{P}_I$ of MIN-CONVEX-SUBSUMING-POLYGON. Let λ be the number of convex vertices in $\mathcal{P}_I$. The reduction ensures that if there exists a satisfying truth assignment of I, then $\mathcal{P}_I$ can be subsumed by a polygon with at most $\lambda - |X||C|^2 - 3|C|$ convex vertices, and vice versa.

Given an instance I of monotone planar 3-SAT, we first construct an orthogonal polygon P_o with holes. We denote each clause and variable using a distinct axis-aligned

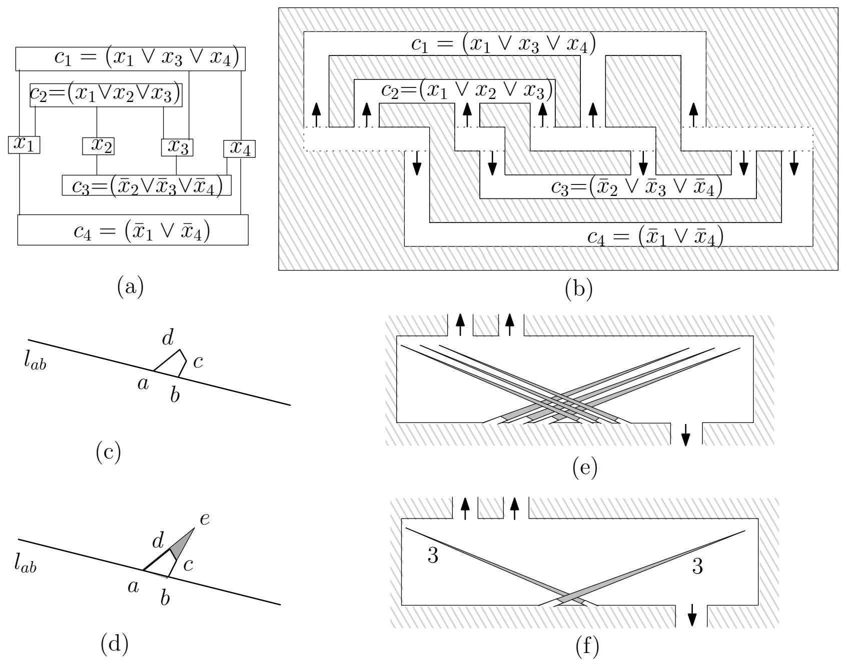

Fig. 4. (a) An instance I of monotone planar 3-SAT. (b) The orthogonal polygon P_o corresponding to I. (c)–(f) Illustration for the variable gadget.

rectangle, which we refer to as the *c-rectangle* and *v-rectangle*, respectively. Each edge connecting a clause and a variable is represented as a thin vertical strip, which we call an *edge tunnel*. Figures 4(a) and (b) illustrate an instance of monotone planar 3-SAT and the corresponding orthogonal polygon, respectively. While adding the edge tunnels, we ensure for each v-rectangle that the tunnels coming from top lie to the left of all the tunnels coming from the bottom. Figure 4 (b) marks the top and bottom edge tunnels by upward and downward rays, respectively. The v-rectangles, c-rectangles and the edge tunnels may form one or more holes, whereas the polygon is shown in diagonal line pattern. We now transform P_o to an instance $\mathcal{P}_I$ of MIN-CONVEX-SUBSUMING-POLYGON.

We first introduce a few notations. Let $abcd$ be a convex quadrangle and let l_{ab} be an infinite line that passes through a and b. Assume also that l_{bc} and l_{ad} intersect at some point e, and c, d, e all lie on the same side of l_{ab}, as shown in Figures 4(c)–(d). Then we call the quadrangle $abcd$ a *tip* on l, and the triangle cde a *cap* of $abcd$.

3.1 Variable Gadget

We construct variable gadgets from the v-rectangles. We add some top-right (and the same number of top-left) tips at the bottom side of the v-rectangle, as show in Figure 4(e). There are three top-right and top-left tips in the figure. For convenience we show only one top-left and one top-right tip in the schematic representation, as shown in Figure 4(f). However, we assign weight to these tips to denote how many tips there should be in the exact construction. We will ensure a few more properties: (I) The caps do not intersect the boundary of the v-rectangle, (II) no two top-left caps (or, top-right caps) intersect, and (III) every top-left (resp., top-right) cap intersects all the top-right (resp., top-left) caps.

Observe that each top-left tip contributes to two convex vertices such that covering them with a cap reduces the number of convex vertices by 1. The peak of the cap reaches very close to the top-left corner of the v-rectangle, which will later interfere with the clause gadget. Specifically, this cap will intersect any downward cap of the clause gadget coming through the top edge tunnels. Similarly, each top-right tip contributes to two convex vertices, and the corresponding cap intersects any upward cap coming through the bottom edge tunnels.

Note that the optimal subsuming polygon P cannot contain the caps from both the top-left and top-right tips. We assign the tips with a weight of $|C|^2$. In the hardness proof this will ensure that either the caps of top-right tips or the caps of top-left tips must exist in P, which will correspond to the true and false configurations, respectively.

3.2 Clause Gadget

Without loss of generality assume that each clause is incident to three edge tunnels, otherwise, we can create necessary multi-edges to satisfy this constraint. Figure 5(a) illustrates the transformation for a c-rectangle. Here we describe the gadget for the positive clauses, and the construction for negative clauses is symmetric. We add three downward tips incident to the top side of the c-rectangle, along its three edge tunnels. Each of these downward tip contributes to two convex vertices such that covering the tip with a cap reduces the number of convex vertices by 1. Besides, the corresponding caps reach almost to the bottom side of the v-rectangles, i.e., they would intersect the top-left caps of the v-rectangles. Let these tips be t_1, t_2, t_3 from left to right, and let $\gamma_1, \gamma_2, \gamma_3$ be the corresponding caps.

We then add a down-left and a down-right tip at the top side of the c-rectangle between t_i and t_{i+1}, where $1 \leq i \leq 2$, as shown in Figure 5(a). Let the tips be $t'_1, \ldots, t'_4$ from left to right, and let the corresponding caps be $\gamma'_1, \ldots, \gamma'_4$. Note that the caps corresponding to t'_j and t'_{j+1}, where $1 \leq j \leq 4$, intersect each other. Therefore, at most two of these four caps can exist at the same time in the solution polygon. Observe also that the caps corresponding to t_1, t_2, t_3 intersect the caps corresponding to $\{t'_2\}, \{t'_1, t'_4\}, \{t'_3\}$, respectively. Consequently, any optimal solution polygon containing none of $\{\gamma_1, \gamma_2, \gamma_3\}$ have at least 12 convex vertices along the top boundary of the c-rectangle, as shown in Figure 5(b).

We now show that any optimal solution polygon P containing at least $\alpha > 0$ caps from $\Gamma = \{\gamma_1, \gamma_2, \gamma_3\}$ have exactly 11 convex vertices along the top boundary of the c-rectangle. We consider the following three cases:

Case 1 ($\alpha = 1$)**:** If γ_1 (resp., γ_3) is in P, then P must contain $\{\gamma'_1, \gamma'_3\}$ (resp., $\{\gamma'_2, \gamma'_4\}$). Figure 5(c) illustrates the case when P contains γ_1. If γ_2 is in P, then P must contain $\{\gamma'_2, \gamma'_3\}$. In all the above scenario the number of convex vertices along the top boundary of the c-rectangle is 11.

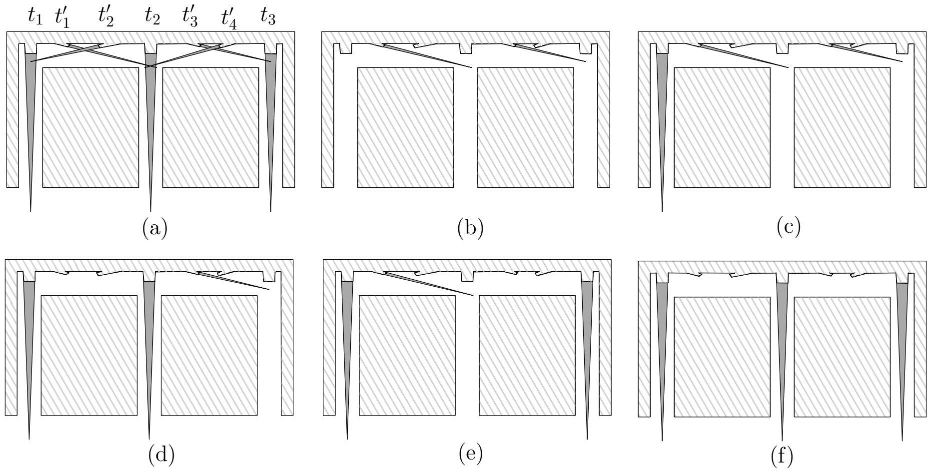

Fig. 5. Illustration for the clause gadget.

Case 2 ($\alpha = 2$)**:** If P contains $\{\gamma_1, \gamma_3\}$, then either γ'_1 or γ'_4 must be in P. Otherwise, P contains either $\{\gamma_1, \gamma_2\}$ or $\{\gamma_2, \gamma_3\}$. If that P contains $\{\gamma_1, \gamma_2\}$, as in Figure 5(d), then γ'_3 must lie in P. In the remaining case, γ'_2 must lie in P. Therefore, also in this case the number of convex vertices along the top boundary of the c-rectangle is 11.

Case 3 ($\alpha = 3$)**:** In this scenario P cannot contain any of $\gamma'_1, \ldots, \gamma'_4$. Therefore, as shown in Figure 5(e), the number of convex vertices along the top boundary of the c-rectangle is 11.

As a consequence we obtain the following lemma.

Lemma 3. *If a clause is satisfied, then any optimal subsuming polygon reduces exactly three convex vertex from the corresponding c-rectangle.*

3.3 Reduction

Although we have already described the variable and clause gadgets, the optimal subsuming polygon still may come up with some unexpected optimization that interferes with the convex corner count in our hardness proof. Figure 6(left) illustrates one such example. Therefore, we replace each convex corner that does not correspond to the tips by a small polyline with alternating convex and reflex corners, as shown Figure 6(right).

We now prove the NP-hardness of computing optimal subsuming polygon.

Theorem 3. *Finding an optimal subsuming polygon is NP-hard.*

Proof. Let $I = (X, C)$ be an instance of the monotone planar 3-SAT and let $\mathcal{P}_I$ be the corresponding instance of MIN-CONVEX-SUBSUMING-POLYGON. Let λ be the number of convex vertices in $\mathcal{P}_I$. We now show that I admits a satisfying truth assignment if and only if $\mathcal{P}_I$ can be subsumed using a polygon having at most $\lambda - |X||C|^2 - 3|C|$ convex vertices.

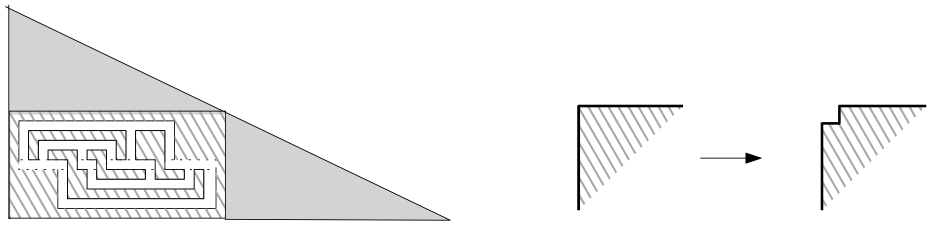

Fig. 6. Refinement of $\mathcal{P}_I$.

First assume that I admits a satisfying truth assignment. For each variable x, we choose either the top-right caps or the top-left caps depending on whether x is assigned true or false. Consequently, we save at least $|X||C|^2$ convex vertices. Consider any clause $c \in C$. Since c is satisfied, one or more of its variables are assigned true. Therefore, for each positive (resp., negative) clause, we can have one or more downward (resp., upward) caps that enter into the v-rectangles. By Lemma 3, we can save at least three convex vertices from each c-rectangle. Therefore, we can find a subsuming polygon with at most $\lambda - |X||C|^2 - 3|C|$ convex vertices.

Assume now that some polygon P with at most $\lambda - |X||C|^2 - 3|C|$ convex vertices can subsume $\mathcal{P}_I$. We now find a satisfying truth assignment for I. Note that the maximum number of convex vertices that can be reduced from the c-rectangles is at most $3|C|$. Therefore, P must reduce at least $|C|^2$ convex vertices from each v-rectangle. Recall that in each v-rectangle, either the top-right or the top-left caps can be chosen in the solution, but not both. Therefore, the v-rectangles cannot help reducing more than $|X||C|^2$ convex vertices. If P contains the top-right caps of the v-rectangle, then we set the corresponding variable to true, otherwise, we set it to false. Since P has at most $\lambda - |X||C|^2 - 3|C|$ convex vertices, and each c-rectangle can help to reduce at most 3 convex vertices (Lemma 3), P must have at least one cap from $\gamma_1, \gamma_2, \gamma_3$ at each c-rectangle. Therefore, each clause must be satisfied. Recall that the downward (resp., upward) caps coming from edge tunnels are designed carefully to have conflict with the top-left (resp., top-right) caps of v-variables. Since top-left and top-right caps of v-variables are conflicting, the truth assignment of each variable is consistent in all the clauses that contains it. $\square$

4 Conclusion

In this paper we have developed a polynomial-time algorithm that can compute optimal subsuming polygons for a given simple polygon in restricted settings. On the other hand, if the polygon contains holes, then we show the problem of computing an optimal subsuming polygon is NP-hard. Therefore, the question whether the problem is polynomial-time solvable for simple polygons, remains open.

Our algorithm can find an optimal solution if the optimal subsuming polygon lies on some prescribed arrangement of lines, e.g., $\mathcal{A}_e(P)$ or $\mathcal{A}_v(P)$. The running time of our algorithm depends on the length of the subsuming chains, i.e., the running time is polynomial if the subsuming chains are of constant length. However, there exist polygons

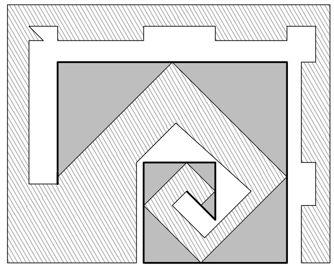

Fig. 7. Illustration for the case when the optimal subsuming polygon contains a subsuming chain of length $\Omega(n)$. The subsuming chain is shown in bold.

whose optimal subsuming polygons contain subsuming chains of length $\Omega(n)$. Figure 7 illustrates such an example optimal solution that is lying on $\mathcal{A}_e(P)$. Therefore, it would be interesting to find algorithms whose running time is polynomial in the size of $\mathcal{A}_e(P)$ or $\mathcal{A}_v(P)$.

Another interesting research direction would be to examine whether there exists a good approximation algorithm for the problem.

References

1. O. Aichholzer, T. Hackl, M. Korman, A. Pilz, and B. Vogtenhuber. Geodesic-preserving polygon simplification. *Int. J. Comp. Geom. & Appl.*, 24(4):307–324, 2014.
2. L. Arge, L. Deleuran, T. Mølhave, M. Revsbæk, and J. Truelsen. Simplifying massive contour maps. In *Proc. ESA*, volume 7501 of *LNCS*, pages 96–107. Springer, 2012.
3. M. de Berg and A. Khosravi. Optimal binary space partitions for segments in the plane. *Int. J. Comp. Geom. & Appl.*, 22(3):187–206, 2012.
4. D. H. Douglas and T. K. Peucker. Algorithm for the reduction of the number of points required to represent a line or its caricature. *The Canadian Cartographer*, 10(2):112–122, 1973.
5. L. J. Guibas, J. Hershberger, J. S. B. Mitchell, and J. Snoeyink. Approximating polygons and subdivisions with minimum link paths. *Int. J. Comp. Geom. & Appl.*, 3(4):383–415, 1993.
6. J. M. Keil, J. S. B. Mitchell, D. Pradhan, and M. Vatshelle. An algorithm for the maximum weight independent set problem on outerstring graphs. In *Proc. CCCG*, pages 2–7, 2015.
7. D. Kempe. On the complexity of the "reflections" game, 2003. http://www-bcf.usc.edu/ dkempe/publications/reflections.pdf.
8. W. A. Mackaness, A. Ruas, and L. T. Sarjakoski. *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier, 2011.
9. H. Ratschek and J. Rokne. *Geometric Computations with Interval and New Robust Methods: Applications in Computer Graphics, GIS and Computational Geometry*. Horwood Publishing, 2003.