Technische Universität Berlin

FACHBEREICH 3
MATHEMATIK

High Quality Quadrilateral Surface Meshing Without Template Restrictions: A New Approach Based on Network Flow Techniques

by

Matthias Müller–Hannemann

No. 561/1997

HIGH QUALITY QUADRILATERAL SURFACE MESHING WITHOUT TEMPLATE RESTRICTIONS: A NEW APPROACH BASED ON NETWORK FLOW TECHNIQUES *

MATTHIAS MÜLLER–HANNEMANN
Fachbereich Mathematik, Technische Universität Berlin, Sekr. MA 6-1,
Straße des 17. Juni 136, D 10623 Berlin, Germany,
e-mail: mhannema@math.tu-berlin.de;
URL: http://www.math.tu-berlin.de/˜mhannema

June, 1997, revised December 1998
to appear in
International Journal of Computational Geometry & Applications

ABSTRACT

We investigate a purely combinatorial approach to the following mesh refinement problem: Given a coarse mesh of polygons in three-dimensional space, find a decomposition into well-shaped quadrilaterals such that the resulting mesh is conforming and satisfies prescribed local density constraints.
We present a new approach based on network flow techniques. In particular, we show that this problem can efficiently be solved by a reduction to a minimum cost bidirected flow problem, if the mesh does not contain branching edges, that is, edges incident to more than two polygons. This approach handles optimization criteria such as density, angles and regularity. In our model we get rid of restrictions on the set of feasible solutions imposed by templates. On the other hand, we still use advantages of general templates with respect to mesh quality for the individual refinement of the mesh polygons.
For meshes with branchings, the problem is feasible if and only if a certain system of linear equations over $GF(2)$ has a solution. To enhance the mesh quality for meshes with branchings, we introduce a two-stage approach which first decomposes the whole mesh into components without branchings, and then uses minimum cost bidirected flows on the components in a second phase. We report on our computational results which indicate that this approach usually leads to a very high mesh quality.

Keywords: Quadrilateral surface meshes, non-manifold surfaces, mesh decomposition, bidirected flows, b-matchings, mesh smoothing

1. Introduction

Mesh refinement has gained much attention in recent years because of its increasing role as a bottleneck in the finite element modeling and analysis process. In the field of computer-aided design (CAD), engineers often model their workpieces first in form of a coarse mesh of convex polygons in three-dimensional space which

*A preliminary version of this paper appeared in the Proceedings of the Sixth International Meshing Roundtable, Park City, Utah, 1997, pp. 293–307.

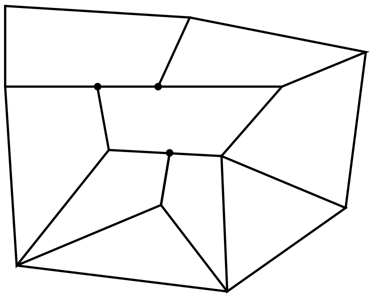

Figure 1: A small, planar artificial mesh.

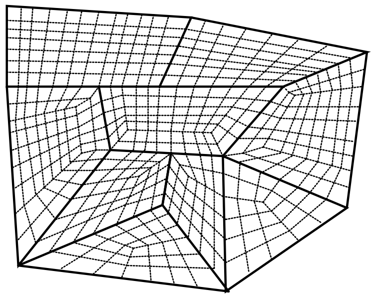

Figure 2: A conformal refinement.

approximates the object's surface. However, in order to make a numerical analysis applicable, a suitable refinement of the coarse mesh is necessary.

A large amount of research has been done in the area of mesh refinement into triangles, see [18], [6] and [7] for surveys. In contrast, there is much less work on quadrilaterals, although meshes which consist *solely* of quadrilaterals are more appropriate in many applications, such as torsion problems and crash simulations [32], [10]. This is the background of our work, and therefore, in this paper, *refinement* of a mesh means decomposing each polygon into strictly convex quadrilaterals.[a]

The input: a coarse mesh. A *polygon* is a region in the plane or, more generally, of a smooth surface in the three–dimensional space, bounded by a finite, closed sequence of straight line (or curved) segments (the so-called *edges*). The endpoints of the line segments or curves are the *vertices*. A polygon is *simple* if its edges do not cross each other, and *convex* if the internal angle at each vertex is at most π. A vertex of a convex polygon is a *corner* if its internal angle is strictly less than π. An *interval* of a polygon P is a path of edges on its boundary. A *segment* S is an interval between two successive corners of P.

A *mesh* is a set of openly disjoint, convex and simple polygons, the so-called *macro elements*. The macro elements are convex, but not necessarily strictly convex. Let $\mathcal{M} = \{P_1, P_2, \ldots, P_q\}$ be the set of polygons of the mesh. Two macro elements are *neighbored* if they have points of the boundary in common which are not corners. These neighborhood relationships induce an undirected graph $G = (V, E)$, which is embedded on the surface approximated by the mesh. More precisely, V consists of the vertices of the polygons. If a vertex of a polygon also belongs to the interior of a side of another polygon, it subdivides this side. Hence, we may identify common intervals of neighbored sides of polygons with each other, and E consists of these intervals after identification. For an edge $e_i \in E$, let E_i be the set of all those polygons which contain e_i.

A *combinatorial description* of a mesh consists of the graph G and the hypergraph $H = (\mathcal{M}, \{E_1, \ldots, E_m\})$ with vertex set $\mathcal{M}$ and hyperedge set $\{E_1, \ldots, E_m\}$.

[a]The work arose from a cooperation with a CAD software company, Dr. Krause Software GmbH, Berlin, Germany, which has developed the finite element preprocessor ISAGEN.

We will often identify a mesh with its combinatorial description.

In a *conformal* refinement of a mesh, any two distinct quadrilaterals which are not completely disjoint either share exactly one whole edge, or they have a single common vertex.

Two-phase approach. Conformal mesh refinement can be achieved in a two-phase approach. First determine the additional mesh vertices located on the individual edges of the input mesh. Then refine each macro element separately such that, for the boundary of each macro element, the vertices are exactly those determined in the first stage.

Templates. Work on conformal refinements in the literature often relies on a few classes of *templates* (see [28,23] and Fig. 7). A *template* (sometimes also called *meshing primitive* [28]) is a pattern which describes how a single polygon can be decomposed into quadrilaterals. The most prominent template is the $n \times m$ grid. Not every mesh allows for a conformal refinement under template restrictions (examples are in [24,20]). We get rid of such template restrictions by using only the evenness condition from the following well-known, but important characterization of those polygons which can be decomposed into strictly convex quadrilaterals [31,19]:

Lemma 1 *A simple polygon P admits a conformal refinement into strictly convex quadrilaterals (without placing additional vertices on the boundary of P) if and only if the number of vertices of P is even.*

In fact, *any* algorithm for conformal mesh refinement has to respect this necessary evenness condition either implicitly or explicitly. Observe that all standard templates certainly do so, but they impose further, non-necessary restrictions on the refinement.

An important feature of our approach is that we can guarantee to fulfill the evenness condition for all polygons simultaneously (if such a solution exists). This property of our approach is also valuable for other methods such as advancing front [31,27] or paving [8,29]. In particular, there is no need to use a few triangular elements as in Rees [27].

Branchings. Note that the graph G of a mesh need not be planar; for example, a mesh approximating the surface of a torus has genus one. Even more, the approximated surface need not be a two-manifold, i. e. the corresponding mesh model may contain *branching edges*, that is, edges incident to more than two polygons (Figure 3). We call a mesh *homogeneous* if it does not contain branching edges. Coping with branchings is an issue of crucial importance, as they appear in many practical examples. Quadrilateral surface meshing can be seen as a first step in hexahedral volume meshing (which still is an only partially solved problem with respect to both theory and practice). At least, it seems to be a promising approach to start a decomposition into hexahedra from a high quality quadrilateral surface mesh [20]. Here, we want to point out that it might be advisable to decompose complicated solid models first into smaller, preferably convex, subdomains by in-

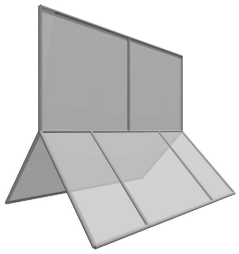

Figure 3: A small mesh with four branching edges.

sertion of internal polygons. This can be done either explicitly or implicitly. The latter approach, so-called *meshing by virtual decomposition*, has been introduced by White et al. [30]. In both variants, these additional polygons induce a number of branchings. However, this fits perfectly into our approach which mainly abstracts from geometry and essentially solves a combinatorial problem. In fact, we will introduce a simple and elegant method to ensure conformity between subdomains resulting from a branching.

Local mesh density control. The *subdivision number* of edge $e \in E$, denoted by x_e, is the number of additional vertices which are placed on edge e in a refinement. To ensure that the refinement of a mesh is fine enough for the numerical analysis to achieve the required accuracy, but not too fine for reasons of efficiency, the mesh density has to be controlled. Depending on the application, it is often crucial to have a local density control on a per edge basis (derived from an error estimation in the numerical analysis). Hence, we usually have for each edge $e \in E$ a desired subdivision number d_e. Moreover, we use *density constraints* for the purpose of density control: For an edge $e \in E$, the subdivision number x_e is at least ℓ_e and at most u_e: $\ell_e \leq x_e \leq u_e$ (*lower and upper edge capacities*, (ℓ_e, u_e)-capacities, for short).

An edge with equal upper and lower capacities is a *fixed edge*, otherwise it is *free*. A conformal mesh refinement is *feasible* if it respects the density constraints.

Fixed edges may appear in particular if certain parts of the mesh are refined in advance, and other parts have to be meshed afterwards (or remeshed) in such a way that the predetermined parts remain unchanged. Or they appear if a hybrid approach for the meshing is used where different algorithms are used for meshing subdomains, for example an advancing front based approach or paving in combination with a template based approach [20].

We also note that it is sometimes desired (for symmetry reasons, for example) to enforce that certain edges (which might be geometrically far away from each other) are subdivided by exactly the same number of subdivision points. Such an additional restriction can easily be represented in our combinatorial model by an identification of the corresponding single edges to a new branching edge.

Angle control. The quality of a refinement largely depends on the shape of its quadrilaterals. In Section 2 we explain how our bidirected flow model can be used to control the quality of interior angles already in the first stage of the refinement process (which determines the combinatorial structure of the refinement). We use mesh smoothing by local optimization for the final embedding phase (see Section 4).

Overview. In Section 2 we will show that the feasible conformal mesh refinement problem *without* branchings can efficiently be solved by a reduction to a *single* minimum cost bidirected flow problem (or, equivalently, to a weighted b–matching problem). This bidirected flow model captures the refinement problem without adding additional restrictions. In contrast to previous work [28,23], we get rid of restrictions imposed by standard templates. See the seminal paper of Edmonds [13] or the monograph by Derigs [12] for an introduction to bidirected flows and b–matchings. Within this model, certain optimization criteria such as mesh density, interior angles and mesh regularity can be handled.

Then, in Section 3, we show that the general refinement problem is feasible if and only if a certain system of linear equations over $GF(2)$ has a solution. (GF(2) denotes the general field of two elements.) Such systems can be solved efficiently by using standard Gaussian elimination (and numerical stability is no problem over $GF(2)$). From any solution to this system of linear equations we easily derive a feasible conformal mesh refinement. However, there seems to be no way to incorporate mesh quality optimization directly into this approach. This is not surprising as even optimizing the mesh density on the edges has been shown to be $\mathcal{NP}$–hard for meshes with branchings [22].

For that reason, we use mesh decomposition into homogeneous components and combine both approaches: In a first phase, we determine the subdivision numbers for all branching edges by solving a system of linear equations over $GF(2)$. Afterwards, in a second phase, we solve a minimum cost bidirected flow problem for each homogeneous component with fixed subdivision numbers on the branching edges. For real-world instances the number of branching edges is usually relatively small in comparison with the number of polygons. Hence, the proposed combined approach is likely to achieve a reasonably good overall mesh quality.

The main contribution of this paper is to elaborate in detail on the mesh quality which can be achieved by this approach (see Section 4). For the detailed proofs of our theoretical results we refer to [22].

2. Conformal mesh refinement without branchings

In this section we consider the first phase of the refinement process (where the number of additional vertices are determined for each edge of the input mesh) for the case of meshes *without* branchings.

We present a model which guarantees the evenness condition for all polygons of a mesh simultaneously. It turns out that the feasible conformal mesh refinement problem can efficiently be solved by a reduction to a bidirected flow problem.

Bidirected flows. Bidirected flow problems can be defined in several (equivalent) ways (see [13,12]). We will henceforth use the following setting.

Let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be an undirected graph (loops and parallel edges allowed), and for $\tilde{e} \in \widetilde{E}$ let $u_{\tilde{e}} \geq \ell_{\tilde{e}} \geq 0$ be the *upper* and *lower capacity* of edge $\tilde{e}$. [b]

For each vertex $\tilde{v} \in \widetilde{V}$, the set of all incident edges is partitioned into two parts, $A_1(\tilde{v})$ and $A_2(\tilde{v})$, where $A_2(\tilde{v})$ may be the empty set. We define $\delta(A_i(\tilde{v}))$ as the set of non-loop edges in $A_i(\tilde{v})$, and $\gamma(A_i(\tilde{v}))$ as the set of loops within $A_i(\tilde{v})$, for $i = 1, 2$. An integer weighting $x \in \mathbb{Z}^{\widetilde{E}}$ of all edges in $\widetilde{E}$ is a *feasible bidirected flow* if and only if

(1) $\ell_{\tilde{e}} \leq x_{\tilde{e}} \leq u_{\tilde{e}}$ for each edge $\tilde{e} \in \widetilde{E}$ (*edge capacity constraints*),

(2) $$\sum_{\tilde{e} \in \delta(A_1(\tilde{v}))} x_{\tilde{e}} + 2 \sum_{\tilde{e} \in \gamma(A_1(\tilde{v}))} x_{\tilde{e}} - \sum_{\tilde{e} \in \delta(A_2(\tilde{v}))} x_{\tilde{e}} - 2 \sum_{\tilde{e} \in \gamma(A_2(\tilde{v}))} x_{\tilde{e}} = b_{\tilde{v}}$$

for each vertex $\tilde{v} \in \widetilde{V}$ (*flow conservation constraints*).

Vertices with $b_{\tilde{v}} = 0$ are called *transshipment vertices*. Note that a bidirected flow is a proper generalization of the usual flow definition for a directed graph, where the orientation of the edges induces the bipartition $A_1(\tilde{v}), A_2(\tilde{v})$ in a simple way, namely into incoming and outgoing edges for each vertex $\tilde{v}$.

Single polygons. For a single polygon P with p segments we now define a small bidirected flow problem on a graph $\widetilde{G}_P = (\widetilde{V}_P, \widetilde{E}_P)$. See Figure 4 for examples of polygons with three, four and five segments. Let us suppose that we are given some nonnegative integer N_i for each segment S_i, such that the segment S_i has to be subdivided into $N_i + 1$ edges in the refinement.

The vertex set $\widetilde{V}_P$ consists of vertices $\tilde{v}_i$ and $\tilde{w}_i$ corresponding to each segment S_i, and of one additional vertex $\tilde{v}_c$ (c for central), if p is odd.
The edge set $\widetilde{E}_P = \widetilde{E}^{in} \cup \widetilde{E}^{seg} \cup \widetilde{E}^{par}$ consists of

- the set $\widetilde{E}^{in}$ which contains all pairs $(\tilde{v}_i, \tilde{v}_j)$, for $i, j = 1, \ldots, p$, including the loops,
- the set $\widetilde{E}^{seg}$ which contains an edge $(\tilde{v}_i, \tilde{w}_i)$ for each segment S_i, and
- the set $\widetilde{E}^{par}$ which contains an edge $(\tilde{v}_i, \tilde{v}_c)$ for each segment S_i, if p is odd, and is empty otherwise.

The edges in $\widetilde{E}^{in} \cup \widetilde{E}^{seg}$ are "internal" edges of the polygon, and edges in $\widetilde{E}^{par}$ are used to ensure the "correct parity" of the subdivision if the polygon is initially odd. We assign the following capacities to the edges:

- For an edge $\tilde{e} \in \widetilde{E}^{in} \cup \widetilde{E}^{seg}$, we set $\ell_{\tilde{e}} := 0$ and $u_{\tilde{e}} := +\infty$.
- For an edge $\tilde{e} \in \widetilde{E}^{par}$, we set $\ell_{\tilde{e}} := 0$ and $u_{\tilde{e}} := 1$.

[b] The symbol '$\sim$' is used to distinguish the graph on which a flow problem is defined from the graph associated to a mesh.

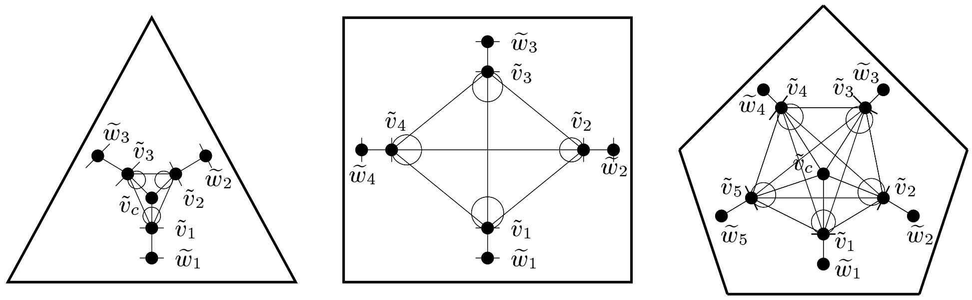

Figure 4: Examples: the bidirected flow graphs for a triangle, a quadrilateral and a pentagon. A short solid line through a vertex is used to indicate a non-trivial bipartition of its adjacency list into two parts.

The bipartition of the vertices is as follows: For each vertex $\tilde{v}_i$, the incident edges are partitioned into the sets $\widetilde{E}^{in} \cup \widetilde{E}^{par}$ and $\widetilde{E}^{seg}$. For each vertex $\tilde{w}_i$ and for $\tilde{v}_c$ (if the latter vertex exists), all incident edges are in $A_1(\cdot)$, and $A_2(\cdot)$ is empty.

We define $b_{\tilde{v}_c} = 1$, if p is odd, and all other vertices $\tilde{v}_i$ are transshipment vertices. Finally, let $b_{\tilde{w}_i} = N_i$ for the given nonnegative integers N_i. This completes the definition of a bidirected flow problem for the polygon P.

Meshes without branchings. We now extend our definition of a bidirected flow instance for a single polygon to an instance for a whole mesh. Let $G = (V, E)$ be the undirected graph of a mesh without branchings.

The underlying graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ of the bidirected flow instance is built up using the graphs $\widetilde{G}_P = (\widetilde{V}_P, \widetilde{E}_P)$ as defined for single polygons as subgraphs. Roughly speaking, two such subgraphs are connected by a “dual edge” if the corresponding polygons share an edge in G.

The vertex set $\widetilde{V}$ is the union of

- the vertex sets $\widetilde{V}_P$ of all polygons P,
- all vertices among V which are non-corners of some polygon, and
- one special vertex $\tilde{v}_{out}$. This vertex corresponds to the region “outside” the mesh.

The edge set $\widetilde{E}$ contains the following edges:

- We take the union of the edges sets $\widetilde{E}_P$ over all polygons P, and the edge capacities as in $\widetilde{G}_P$.
- Let $e \in E$ be an edge which belongs to two polygons, say to P and Q, and suppose that $\tilde{w}_i \in \widetilde{V}_P$ and $\tilde{w}_j \in \widetilde{V}_Q$ are the vertices which correspond to the segments of P and Q to which e belongs. For each such edge e, we introduce a “dual” edge $\tilde{e} = (\tilde{w}_i, \tilde{w}_j)$, with edge capacities equal to that of e in G.
- If an edge $e \in E$ belongs only to one polygon, say to P, and $\tilde{w}_i \in \widetilde{V}_P$ is the vertex which corresponds to the segment of P to which e belongs, then we introduce a “dual” edge $\tilde{e} = (\tilde{w}_i, \tilde{v}_{out})$, with edge capacities equal to that of e in G.

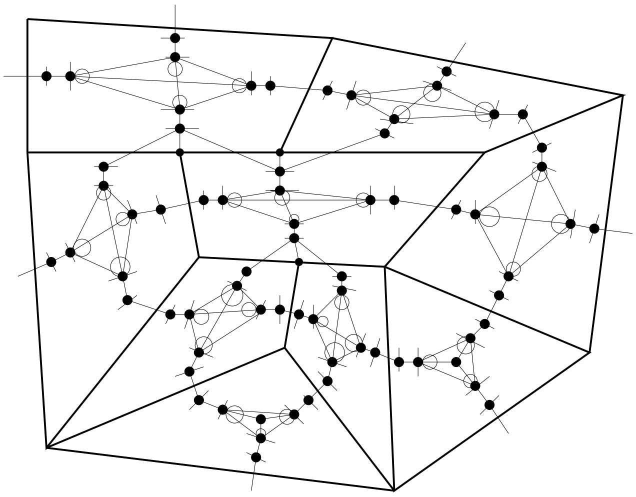

Figure 5: The graph $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ of the bidirected flow problem for an artificial instance. (The special vertex $\tilde{v}_{out}$ is omitted in this figure. The vertex corresponds to the unbounded region around the mesh.)

- For each polygon P and each vertex $v \in V \cap P$ which is not a corner of P, we introduce an edge $\tilde{e} = (\tilde{w}_i, v)$, where v lies on the segment of P to which $\tilde{w}_i \in \widetilde{V}_P$ corresponds. Such an edge gets identical lower and upper capacities $\ell_{\tilde{e}} := u_{\tilde{e}} := 1$.
- A loop $\tilde{e} = (\tilde{v}_{out}, \tilde{v}_{out})$ with capacities $\ell_{\tilde{e}} := 0$ and $u_{\tilde{e}} := +\infty$.

The bipartition of the vertices is as follows: For each vertex $\tilde{w}_i \in \widetilde{V}_P$, we put exactly all edges $\tilde{e} \in \widetilde{E} \setminus \widetilde{E}_P$ incident to $\tilde{w}_i$ into $A_2(\tilde{w}_i)$. For each vertex $v \in V$, all incident edges are in $A_1(v)$, and $A_2(v)$ is empty. All incident edges to $\tilde{v}_{out}$ are in $A_1(\tilde{v}_{out})$ except $(\tilde{v}_{out}, \tilde{v}_{out})$ which belongs to $A_2(\tilde{v}_{out})$.

All vertices $\tilde{w}_i \in \widetilde{V}_P$ which correspond to segments of P become transshipment vertices (in contrast to the case of single polygons). For each vertex $\tilde{v} \in \widetilde{V}$, let $b_{\tilde{v}}$ be equal to the number of incident edges in $\widetilde{G}$. Finally, let $b_{\tilde{v}_{out}} := 0$ if $\sum_{\tilde{v} \in \widetilde{V} \setminus \tilde{v}_{out}} b_{\tilde{v}}$ is even, and $b_{\tilde{v}_{out}} := 1$, otherwise.

Theorem 1 [22] *There exists a feasible conformal refinement for the homogeneous mesh G if and only if the bidirected flow problem as defined above has a feasible bidirected flow.*

2.2. Optimizing the mesh quality

In the preceding section we have seen the correspondence of conformal mesh refinements and feasible solutions of certain bidirected flow problems. We continue with the more ambitious task to find special solutions which fulfill certain optimization criteria instead of just feasible solutions. This leads to *minimum cost bidirected flow problems*. Let $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ be the graph of a bidirected flow instance where a cost $c_{\tilde{e}}$ is associated with every edge $\tilde{e}$. Then the *minimum cost bidirected flow*

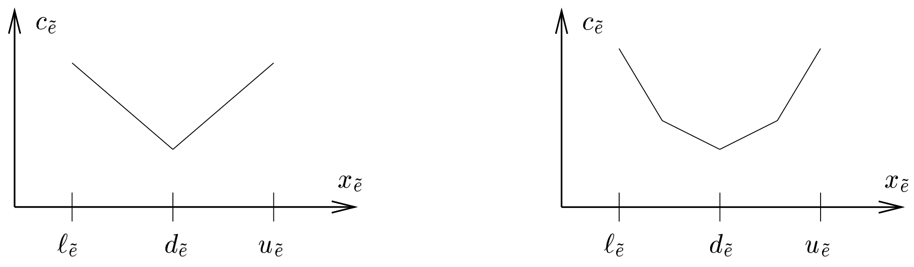

Figure 6: Piecewise linear convex cost functions for density control.

problem seeks for a feasible bidirected flow with minimum cost $\sum_{\tilde{e}\in\widetilde{E}} c_{\tilde{e}} x_{\tilde{e}}$. We will consider three kinds of mesh quality criteria:

- control over the mesh density,
- avoidance of too small or too large angles, and
- "regularity" of the overall mesh structure.

Density control. Probably, the most natural way to get control over the density of a mesh is to use a convex cost function for each dual edge with a minimum at the desired density $d_{\tilde{e}}$. As we allow only integer flows, the cost functions can be assumed to be piecewise linear. For practical purposes, it will often suffice to use piecewise linear functions with only few different slopes, as depicted in Figure 6. The slopes should be chosen such that the relative change from the desired density is taken into account, for example by choosing them inversely proportional to the desired subdivision number. Then the objective is to minimize the weighted sum of deviations over all edges.

This defines a minimum convex cost bidirected flow problem. The standard way to transform such a problem to an ordinary linear minimum cost bidirected flow problem is to replace each edge $\tilde{e}$ by as many copies as there are slopes (cf. [1], pages 551ff.). So if there are p slopes with cost coefficients $c^k_{\tilde{e}}$ and breakpoints at $d^k_{\tilde{e}}$, the k-th copy $\tilde{e}^k$ gets edge capacities $[0, d^k_{\tilde{e}} - d^{k-1}_{\tilde{e}}]$ and a cost coefficient of $c^k_{\tilde{e}}$. (Here, we assume $d^0_{\tilde{e}} = \ell_{\tilde{e}}$ and $d^p_{\tilde{e}} = u_{\tilde{e}}$.)

Very recently, Mitchell [20,21] proposed to minimize the maximum weighted deviation from the desired subdivision numbers. Note that such a bottleneck-type objective can be solved via a combination of our bidirected flow method with binary search on the value of the maximum deviation. Hence, Mitchell's objective is somewhat more expensive than ours but can still be solved with a combinatorial approach for meshes without branchings.

For ease of exposition, we will discuss the next two optimization criteria only for polygons with four corners.

Angle control and mesh structure. It is easy to see that, for a polygon with four segments, we may assume that we have a feasible flow with at most four edges with a non-zero flow within the set $\widetilde{E}^{in}$. Furthermore, there is a conformal refinement such that exactly $x_{(\tilde{v}_i,\tilde{v}_j)}$ disjoint paths go from the interior of segment S_i to the interior of segment S_j. Hence, conformal subdivisions of polygons with four segments can be assumed to be of the form as shown in Figure 7. The first three possibilities are the *standard templates* used in [23,28], whereas the last template is new and generalizes all standard templates.

Apart from the mesh density, mesh quality criteria depend upon the shape of the quadrilaterals. For numerical reasons in the finite element analysis, interior angles of quadrilaterals should neither be too small nor too large. There is no generally accepted, precise threshold, but one usually aims at generating quadrilaterals with no angles smaller than some given α and no angles larger than some β. (In practice, one often uses as a rule of thumb values of $\alpha = 30°$ and $\beta = 150°$ [31].)

The shape of the quadrilaterals in the refinement is closely related to the choice of the template which determines the refinement of the macro element. As a rule of thumb, the more the polygon looks like a trapezoid, the better template (2) will be, the more it looks like a kite, the better template (3) will be. In all other cases, template (1) is likely to be the best.

Moreover, template (1) tends to produce a fairly regular mesh. "Regularity of the mesh structure" is a mesh quality criterion which seemingly cannot be fully formalized. But the heuristical rule to prefer template (1) often achieves practical results which reflects such a goal quite satisfactorily [23].

Hence, we would like to refine as many polygons by template (1) as possible (among those polygons which have no very small or large angle), or more generally, we would like to maximize the number of macro elements which are refined to some preferred template, for a given preference order for each individual polygon of our instance. Unfortunately, such a goal is intractable.

Theorem 2 [22] *Given a feasible homogeneous mesh instance, it is strongly $\mathcal{NP}$-hard to find a solution where the number of macro elements with four corners which are refined according to the $(m \times n)$–grid template (i.e. template (1) in Figure 7) is maximized.*

So what can we hope for? As the mesh refinement problem only allows to insert new vertices and edges, no sharp input angle can be erased. Hence, we can only try to avoid the creation of *new* angles smaller than α. On the other hand, we can try to enforce the splitting of an angle larger than β. Observe that, for the given templates in Figure 7, a non-zero flow on edge $(\tilde{v}_1, \tilde{v}_2)$ induces the splitting of the angle γ_4, and a non-zero flow on the loop $(\tilde{v}_1, \tilde{v}_1)$ induces the splitting of both γ_4 and γ_3.

There are two possibilities to modify our bidirected flow instance for these purposes: First, we can change the edge capacities of "internal edges" $\widetilde{E}^{in}$ of a polygon: In the pure feasibility problem all these edges have a lower capacity of zero and a large upper bound ("plus infinity"). Hence, we can enforce to use an edge, if we set the lower capacity to one, or we can forbid an edge completely, if we set the

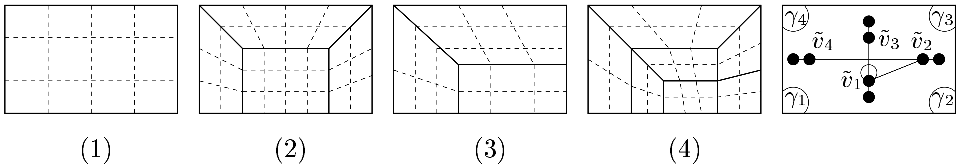

Figure 7: Illustration of the four templates for polygons with four corners. Solid lines are mandatory for each template (up to rotation and symmetry), whereas the number of dashed lines may vary. The rightmost figure shows the corresponding subgraph of the bidirected flow instance with only those edges displayed which may have a non-zero flow value.

upper capacity to zero. The disadvantage of this approach is, that such a modification may cause the infeasibility of the bidirected flow problem. A second (and not so restrictive) way is to assign costs to these edges in order to make them more attractive or unattractive.

Using these ideas we can express our preferences for the choice of the chosen template for each polygon. So if we prefer a realization of template (1) for some polygon, we assign high cost coefficients to all internal edges but $(\tilde{v}_1, \tilde{v}_3)$ and $(\tilde{v}_2, \tilde{v}_4)$, which get zero cost. Or, for example, if the angle γ_4 should be split by an application of template (3), we either raise the lower bound for edge $(\tilde{v}_1, \tilde{v}_2)$ to one or make the corresponding cost coefficient negative.

Hence, in summary, we can use local information about the geometry of the unrefined polygon and so can model our preferences of the choice of an appropriate template and enforce or forbid the splitting of macro element angles by these modifications.

3. Conformal mesh refinement with branchings

A simple approach for meshes with branchings. As we know from Lemma 1, it suffices to determine the subdivision numbers such that all polygons become even. Then we can always complete the quadrangulation of all polygons.

For each edge $e \in E$, let $\tilde{x}_e$ be some fixed integer in the interval $[\ell_e, u_e]$. Then define for each polygon $P \in \mathcal{M}$ the "parity number" $b_P := \sum_{e \in P}(\tilde{x}_e + 1) \bmod 2$.

For each edge $e \in E$, let y_e be a 0/1-integer variable, but fix $y_e \equiv 0$, if $u_e - \ell_e = 0$.

Now, if we take for each polygon P an equation of the form $\sum_{e \in P} y_e = b_P$, this defines a system of linear equations over $GF(2)$ which one can easily solve. Clearly, any solution $\tilde{y}$ to this system immediately makes all polygons even, if we set

$$x_e := \begin{cases} \tilde{y}_e + \tilde{x}_e & \text{if } \tilde{x}_e < u_e \\ -\tilde{y}_e + \tilde{x}_e & \text{otherwise} . \end{cases}$$

Note that the solvability of this system of equations over $GF(2)$ does not depend on the choice of $\tilde{x}$.

This shows that finding subdivision numbers such that all polygons become even is not harder than solving a system of linear equations over $GF(2)$.

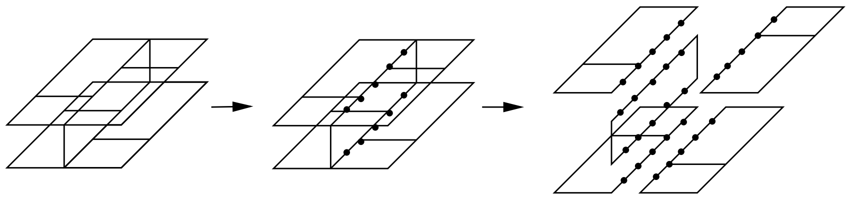

Figure 8: An input mesh with branching edges (left), the placement of additional vertices on branching edges after solving the linear equations over GF(2) (middle), the decomposition into five homogeneous components (right).

Note that the subsequent embedding phase can be done in linear time (linear in the number of subdivision points) using the decomposition algorithm in [25]. The obvious advantage of this approach is its simplicity. The disadvantage, however, is that we cannot incorporate optimization criteria as we can in the homogeneous case. Angle control and a preference of mesh regularity seem to be impossible by this simple approach.

Density control can still be achieved to a certain extent, if we initially choose $\tilde{x}_e$ as the desired subdivision number. But we will be forced to change the subdivision number by one (but not by more than one) for all those edges which have a non-zero entry in the solution of our system of equations. We also note that we cannot hope to find a solution of such a system with a minimum number of non-zero entries, as this problem is $\mathcal{NP}$–hard.

Theorem 3 [22] *For a mesh with branchings and desired subdivision numbers it is strongly $\mathcal{NP}$–hard to find a feasible refinement such that the weighted sum of deviations from the desired subdivision numbers over all edges is minimized.*

A decomposition approach for meshes with branchings. For these reasons, we introduce a mesh decomposition into homogeneous components. This allows us to combine the different approaches for branchings and homogeneous components into a two-phase approach: In a first phase, we determine the subdivision numbers for all branching edges by solving a modified system of linear equations over $GF(2)$ which we explain below. Afterwards, in a second phase, we solve a minimum cost bidirected flow problem for each homogeneous component where the flow values for all branching edges are predetermined from the first phase.

Recall from the Introduction that a combinatorial description of a mesh consists of a graph $G = (V, E)$ and a hypergraph $H = (\mathcal{M}, \{E_1, \ldots, E_m\})$. Depending on the number of polygons they belong to, the edges of G can be partitioned into sets E^1, E^2, and $E^{\geq 3}$. The set E^1 contains the *boundary edges*, i.e. edges which belong to exactly one polygon, the set $E^{\geq 3}$ contains all branching edges, and E^2 all remaining edges. A *non-branching path* in H is a path between two polygons $P_1, P_2 \in \mathcal{M}$ which contains only hyperedges of cardinality two, i.e. hyperedges corresponding to edges in E^2. Being connected by a non-branching path whose edges are all free is an equivalence relation on the set of polygons. Its equivalence classes are exactly the *homogeneous components* of the mesh decomposition.

Such a *decomposition into homogeneous components* of a mesh with branchings can be obtained by the following procedure which "splits" all branching edges and fixed edges: A branching edge which is incident to p polygons is replaced by p copies, once for each polygon. The copied edges are treated as boundary edges, and they get the same capacities as the original ones. All fixed edges in E^2 are replaced in the same way by two copies, once for each incident polygon. Let $G_1, \ldots, G_k$ be the resulting homogeneous components, and $\mathcal{M}_1, \ldots, \mathcal{M}_k$ the sets of polygons contained in these components.
Let $H' = (\mathcal{M}', \{E'_1, \ldots, E'_{m'}\})$ be the hypergraph which we obtain from $H = (\mathcal{M}, \{E_1, \ldots, E_m\})$ if we

(1) delete all hyperedges E_i where e_i is fixed, then afterwards

(2) contract all those hyperedges of degree two which correspond to free edges in E^2, and finally,

(3) identify all parallel hyperedges, i.e. we keep only one hyperedge for all edges $e \in E$ which are incident to exactly the same set of polygons.

Observe that there is a one-to-one correspondence between the vertices of H' and the homogeneous components $G_1, \ldots, G_k$ of G.

We will now define a smaller system of equations over $GF(2)$ based on H'. We start similar as in the first approach above. Again, let $\tilde{x}_e$ be some fixed integer in the interval $[\ell_e, u_e]$, for each edge $e \in E$, and define for each polygon $P \in \mathcal{M}$ the parity number $b_P := \sum_{e \in P}(\tilde{x}_e + 1) \mod 2$. These parity numbers are aggregated to parity numbers for each homogeneous component, by setting $b_{G_i} := \sum_{P \in \mathcal{M}_i} b_P \mod 2$, for $i = 1, \ldots, k$.

For each hyperedge $E'_j \in H'$, let $y_{E'_j}$ be a 0/1-integer variable. For each homogeneous component G_i which is not incident to a free boundary edge in G, we take an equation of the form

$$\sum_{E'_j:\, |E_j \cap G_i| \text{ odd}} y_{E'_j} = b_{G_i},$$

that is, we sum over those hyperedges which are derived from a branching edge with an odd number of incidences with the homogeneous component G_i. Let us call a homogeneous component *active* if it is not incident to a free boundary edge in G, and *inactive* otherwise. Notice, that it might happen that some set $\{E'_j : |E_j \cap G_i| \text{ odd}\}$ is empty (because of the deletion of fixed edges, for example). In that case, we define the sum over the empty set to be zero. This defines our system of linear equations over $GF(2)$.

Suppose that this system has a solution $\tilde{y}$. For each hyperedge E'_i with $\tilde{y}_{E'_i} = 1$, take one corresponding branching edge $e = e_i$ (there may be a choice if E'_i resulted from parallel hyperedges of a branching), and set $x_e := \tilde{x}_e \pm 1$ (such that the edge capacities remain fulfilled). For all other branching edges, set $x_e := \tilde{x}_e$. This completes the first phase. In the second phase, we fix the subdivision values x_e for all branching edges and then solve the bidirected flow problems for each homogeneous component separately. We claim that each bidirected flow problem is feasible if we have a feasible solution $\tilde{y}$ in the first phase. This is summarized in the following theorem:

Theorem 4 [22] *There exists a feasible conformal refinement for a mesh G with branchings if and only if the two-phase approach as described above yields a feasible solution.*

4. Computational Experiences

In this section we report on our experiences with an implementation of the algorithms presented in this paper. We have implemented our algorithm on a Sun Sparc station under SunOS 5.1, the programming language is C++, and our front end is ISAGEN. [c] ISAGEN accepts input from a number of standard CAD formats (IGES, VDAFS, DXF, and some others), and provides routines for an automatic conversion of each of these formats into a macro element model consisting of quadrilaterals and triangles. We got the latter as input for our algorithms.

The goal of this section is to investigate two main questions which immediately arise from the proposed algorithms:

1. How well does the decomposition approach work for meshes with branchings?
2. What mesh quality can be achieved?

Having refined each macro element as a planar graph, we are faced with the important and difficult task of finding a "nice" embedding of the refinement onto the surface defined by the macro element. ISAGEN provides subroutines to find an embedding only for the standard templates. In the following two subsections, we first report on the mesh decomposition approach and then elaborate on mesh quality.

4.1. Mesh Decomposition and Bidirected Flows

The decomposition into homogeneous components can be easily implemented by a straightforward depth or breadth first search. Solving the system of linear equations over $GF(2)$ is also simply done by Gaussian elimination (note that over $GF(2)$ no care has to be taken for numerical stability). The running time for this phase is negligible.

Our implementation has been applied to a variety of problem instances from practice, listed in Table 1. For these instances, some global mesh density has been given. The goal was to find refinements with uniform edge lengths. The results obtained by our decomposition approach are very appealing. Most remarkably, the decomposition approach always yields a feasible solution if one exists. "Emergency" templates (using triangles) are not necessary nor do we leave unrefined "holes." Table 1 shows the distribution of realized templates for quadrangles. However, more expressive quantities for the mesh quality will be given in the following subsection. The number of homogeneous components after the decomposition is given in column

[c] ISAGEN is a trademark of Dr. Krause Software GmbH, Berlin.

instance	♯ macro elem.	♯ bra. edges	des. len.	C	T	♯ realized templates 1	2	3	4	sec. CPU CPLEX	BLO
1) axle	34	0	30	1 (0)	6	24	2	2	0	0.2	0.6
2) bowl	24	0	20	1 (0)	2	21	0	1	0	0.1	0.2
3) wing	131	0	20	1 (0)	16	106	5	4	0	2.6	9.6
4) casing	237	2	10	1 (0)	26	167	25	14	5	225.3	20.5
5) wh-cap	377	0	10	1 (0)	66	247	30	30	4	244.4	15.4
6) rack	68	0	30	1 (0)	2	64	0	2	0	0.4	0.4
7) chassis	158	38	20	5 (1)	9	127	6	16	0	45.8	10.2
8) coolsys	530	115	5	13 (6)	31	375	40	74	10	19.0	7.1
9) pump	161	7	10	12 (4)	7	141	5	7	1	3.5	2.9
10) bend	608	21	10	11 (10)	6	559	5	38	0	244.2	34.9
11) vw-p4	188	116	20	90 (28)	55	93	9	30	1	2.5	1.8
12) tub	150	68	30	26 (26)	8	126	16	0	0	1.3	0.9
13) frp	171	62	4	19 (19)	0	128	20	17	6	241.1	5.6

Table 1: Results of the decomposition approach for real-world instances. The third column shows the number of branching edges, the fourth column contains the desired edge length, the fifth column gives the number of triangles (T) among the macro elements. (C) refers to the number of homogeneous components in the mesh decomposition, the number of active components is given in brackets. Templates (1), (2), and (3) are the standard templates, whereas template (4) is the general template for quadrangles. The last two columns give the CPU times (in seconds) to solve the minimum cost bidirected flow instances with CPLEX and our implementation of the blossom algorithm (BLO), respectively.

(C), the number of active components is given in brackets. The first six instances consist of just one single component, which implies that the corresponding linear systems of equations over $GF(2)$ are empty. The distinction between active and inactive homogeneous components is useful for instances (7)–(11), whereas instances (12) and (13) stem from solid models without boundary. The latter rules out the possibility of inactive homogeneous components.

Minimum cost bidirected flow problems are equivalent to minimum cost perfect b-matchings and can, therefore, be solved in strongly polynomial time, as shown by Anstee [3] and Edmonds (cf. Gerards' survey on matching [17]). The asymptotic running time of strongly polynomial algorithms for these problems are dominated by strongly polynomial algorithms for ordinary minimum cost flow problems. Thus, the best strongly polynomial time bound for minimum cost bidirected flow is $O((m \log n)(m + n \log n))$, where m denotes the number of edges and n the number of vertices of the underlying graph [1]. The number of edges and vertices in the auxiliary graph of the bidirected flow model is linear in the number of original polygons and mesh edges.

We explored two implementations for the solution of the bidirected flow problems, an approach using a general purpose integer programming solver and a purely combinatorial implementation of the primal-dual blossom algorithm for weighted b-matching as described by Pulleyblank [26]. As the problem remains unchanged, the optimal solutions, i. e. the mesh quality, is certainly not affected by a change in the algorithmic approach.

Using integer linear programming. Our first implementation uses the callable library of the linear and integer programming solver CPLEX 4.0 [d] with its Mixed Integer Solver option (MIP). (Note that the minimum cost bidirected flow problem has the form of a pure integer programming problem.) CPLEX [11] provides a number of algorithmic parameters which can be used to improve the performance of the MIP solver. In our experiments we obtained the overall best solution times for our particular class of problem instances by changing the default parameters for the branch and bound process in two ways. First, we used the "best estimate node selection strategy," and second, variables are selected with so-called "strong branching." See the CPLEX manual [11] for an informal description of these heuristics.

We set a time limit of 300 seconds to solve a minimum cost bidirected flow instance, however, in all cases the actual solution time for a near-optimal feasible solution was less than 25 seconds. The time to compute the optimal solution for these problems is given in Table 1. The running time ranges from less than a second to 244 seconds of CPU time for the instances given there.

Using weighted b–matching. Implementing matching algorithms is tedious. Nevertheless, we decided to implement a combinatorial alternative to the general purpose integer programming solver CPLEX. In our project, we implemented Pulleyblank's description of a primal-dual algorithm for weighted b-matching [26] and combined it with heuristic ideas of Applegate and Cook [4], and Ball and Derigs [5]. A detailed description of the implementation details goes beyond the scope of this article. However, this effort gives several advantages: first, our implementation becomes independent of a commercial third party product. Second, the running times to solve all instances to optimality reduced significantly (see the last column in Table 1). Third, this variant has much lower storage requirements.

For this algorithm the running time ranges from less than a second to 35 seconds of CPU time for our test set. The execution time is more than acceptable in view of the other far more expensive steps in a numerical analysis.

4.2. Graph Embeddings and Mesh Smoothing

Given the subdivision numbers on the edges of the input mesh, it is easy to create for each macro element the corresponding planar graph of the decomposition. It is also no problem just to find a preliminary geometric embedding of such a graph on the corresponding surface. However, it is usually hard to find an embedding which leads to a good mesh quality. A common approach for *mesh improvement*, namely *mesh smoothing*, uses local optimization to move the vertex positions while preserving the combinatorial embedding of the graph.

Laplacian smoothing might be the most frequently used smoothing technique [7]. This method iterates over the vertex set several times, repeatedly moving each adjustable vertex to the (weighted) barycenter of the vertices adjacent to it. The method is computationally inexpensive. However, in general, the barycenter of a

[d] CPLEX is a registered trademark of CPLEX Optimization, Inc.

number of vertices need not lie on the surface anymore. Hence, a projection back onto the given surface is necessary. Even in the plane the method does not guarantee an improvement in mesh quality. Moreover, it can happen that quadrilaterals become inverted, i.e. non-convex, unless the algorithm performs an explicit check before moving a vertex.

Freitag, Jones, and Plassmann [14] proposed an alternative to Laplacian smoothing for triangulations. They compute for each vertex a new placement that maximizes the minimum angle in adjacent triangles by use of an iterative steepest-descent algorithm to solve this optimal placement problem. Recent theoretical work by Amenta, Bern, and Eppstein [2] on triangulated, plane meshes shows that optimization-based smoothing can be performed in linear time for a number of quality measures. However, it remains open to which extent these results can be extended to quadrilateral meshes and to curved surfaces. Moreover, in spite of its theoretical linear time complexity these mesh improvement procedures turn out to be very expensive.

We tried to combine several quality measures simultaneously. Let θ, with $0 \leq \theta \leq 360$, be the measure of an angle in degrees. The *quality of an angle* $\phi(\theta)$ ranges between 0 and 90 and is defined by

$$\phi(\theta) = \begin{cases} \theta & \text{if } 0 \leq \theta \leq 90, \\ 180 - \theta & \text{if } 90 < \theta \leq 180, \\ 0 & \text{if } \theta > 180. \end{cases}$$

The *angle quality of a quadrilateral* Q with interior angles α, β, γ, δ is defined as

$$aq(Q) := \min\{\phi(\alpha),\, \phi(\beta),\, \phi(\gamma),\, \phi(\delta)\,\}.$$

Quadrilaterals with an angle quality below the threshold of 30 are marked as *distorted.*

Roughly speaking, we wanted to avoid very large and very small angles and to create uniform edge lengths. More precisely, our primary objective was to maximize the (weighted) average angle quality over all quadrilaterals. In the weighted version of this objective, distorted quadrilaterals are assigned a higher weight than non-distorted ones. Our secondary objective was to create uniform edge lengths and to maximize the (minimum or average) aspect ratio. These goals are combined in form of a weighted sum.

In our implementation, we used the general framework of local optimization, but did not solve the optimization problems exactly. Instead we iterate over the vertex set a fixed number of times and strive only for an approximation of the optimal placement problem. For each adjustable vertex we first determine a search direction and calculate a step length which gives a proposal for a new vertex position. Then we check whether the new position is feasible, i.e. it preserves planarity of the graph and convexity of the quadrilaterals, and whether it would yield an improvement in local mesh quality. In the affirmative case, the new position is accepted, otherwise we try a new search direction or step length. Moreover, additional side constraints in the position check come from our secondary objective to have more or less uniform

		angle quality								
instance	Q	> 80	> 70	> 60	> 50	> 40	> 30	≤ 30	DA	AQ
1) axle	216	87	34	17	28	24	6	20	6	66.28
2) bowl	351	277	35	10	13	6	7	3	0	83.04
3) wing	3955	1798	852	552	334	248	143	28	7	73.26
4) casing	12083	5727	2144	1468	1172	817	518	237	20	72.32
5) wh-cap	9747	6963	776	564	405	475	369	195	47	78.61
6) rack	2030	1600	186	73	53	62	52	4	0	81.85
7) chassis	1981	877	395	343	191	122	45	8	8	73.42
8) coolsys	13885	7658	2765	1518	869	655	373	47	8	76.39
9) pump	11105	7977	1294	928	352	269	208	77	3	80.62
10) bend	3045	2014	428	176	153	137	113	24	4	78.01
11) vw-p4	992	370	165	157	108	104	68	20	31	68.88
12) tub	1332	936	234	33	52	62	11	4	0	82.03
13) frp	7385	3305	1567	1080	665	503	242	23	0	73.54
$\sum$	68107	39589	10875	6919	4395	3484	2155	690	134	76.34

Table 2: Evaluation of the angle quality for our set of real-world instances. The second column gives the number of quadrilaterals (Q) in the refinement. The set of quadrilaterals is partitioned into seven classes of different angle quality, ordered from left to right with decreasing quality. The third column contains the number of quadrilaterals with an angle quality above 80, the forth column the corresponding number for quadrilaterals with a quality in the range from 70 to 80, and so on. In particular, the number of distorted elements is given in the third but last column. (DA) refers to the number of input angles with a bad quality. (AQ) denotes the average angle quality.

edge lengths. In particular, very short edges are always rejected, even if the angle quality could otherwise be further improved.

Certainly, there is a lot of freedom in fine-tuning the choice of several parameters. Most probably, our empirical results can still be improved by a more clever choice of these parameters. In Table 2 we show the results of our current implementation applied to our set of real-world instances. See also the histogram in Figure 9 for the distribution of the angle quality with respect to classes of different angle quality.

Statistically, our refinements contain few quadrilaterals with "bad angles" (less than 1%). Experiments by Freitag and Ollivier-Gooch [16] showed that for linear finite elements on triangular meshes the number of iterations required for convergence is not significantly affected by a small number of poor-quality elements. Such a result seems likely to hold in some form for quadrilateral meshes, too. If so, the number of distorted elements in our results is not a critical issue.

The average angle quality for all instances is 76.34. This seems to be a significant improvement over results obtained for *free-form mesh generation* (in *free-form mesh generation*, there is no prepartition of the whole meshing domain into macro elements). Unfortunately, we are not aware of a computational study which is directly comparable with ours. Borouchaki, Frey and George [9] report results for free-form meshing where they use the same measure for the angle quality (up to a constant scalar), but calculate angles with respect to a metric induced by an isotropic field. For this measure they obtain a mean angle quality of approximately

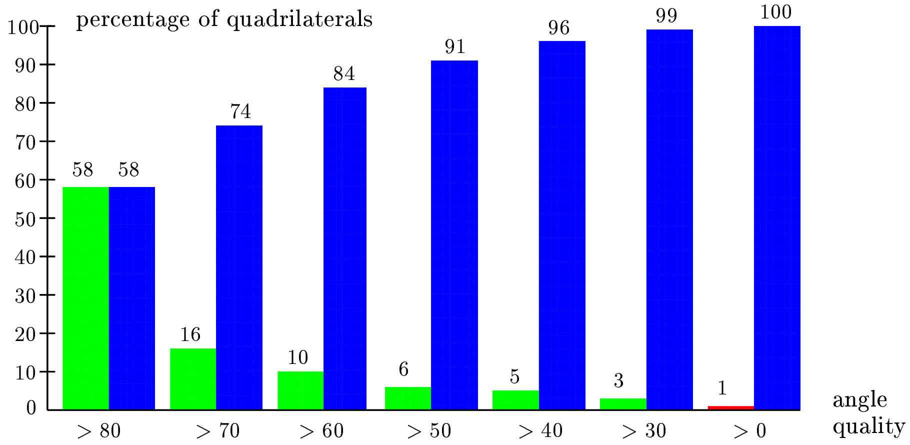

Figure 9: The average angle quality: for each quality class, the left columns gives the percentage of quadrilaterals falling into it, whereas the right columns give the cumulative percentage (with decreasing quality).

61.2 (this number is already scaled up to our measure, but still cannot be used for a fair comparison).

Note that certain input instances contain a significant number of macro elements with a bad angle quality (column (DA) in Table 2). As the geometry of the macro elements themselves is not subject to change in the mesh smoothing process, this implies that a number of quadrilaterals has to be distorted in any refinement. In particular, very small angles cannot be removed. On the other hand, very large angles can possibly be split in two acceptable smaller ones. This explains why the number of bad input angles (DA) can be larger than the number of distorted elements in the refinement (which happens for our instance (11) in Table 2).

Our mesh smoothing technique is computationally very expensive. In fact, it usually consumes the major portion of the whole run time. For our examples, it took more than fifty percent of the overall run time.

Fortunately, this run time bottleneck can be reduced if the smoothing is performed in parallel. The crucial observation is that our macro elements induce in a natural way a fast mechanism for determining independent sets of vertices that can be manipulated simultaneously on different processors. Freitag, Jones and Plassmann [14] showed how to do this distribution on processors with graph coloring techniques, in general, and they reported successfully results of experiments run on a network of workstations. Moreover, a further speed-up can be achieved by a combined smoothing approach in which Laplacian smoothing is always done, but followed by optimization based smoothing when quality is low [15].

Acknowledgments

The author wishes to thank R. H. Möhring and K. Weihe for many fruitful discussions, and G. Krause for providing us with the finite element preprocessor ISAGEN and instances from the automobile industry. Finally, we thank A. Schwartz for his help in implementing our algorithm. The author was partially supported by the special program "Efficient Algorithms for Discrete Problems and Their Applications" of the Deutsche Forschungsgemeinschaft (DFG) under grant Mo 446/2-2.

References

1. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*, Prentice Hall, 1993.
2. N. Amenta, M. Bern, and D. Eppstein, *Optimal point placement for mesh smoothing*, Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms, 1997, pp. 528–537.
3. R. P. Anstee, *A polynomial algorithm for b-matching: An alternative approach*, Information Processing Letters **24** (1987), 153–157.
4. D. Applegate and W. Cook, *Solving large-scale matching problems*, Network Flows and Matching, DIMACS Series in Discrete Mathematics and Theoretical Computer Science (D. S. Johnson and C. C. McGeoch, eds.), vol. 12, 1993, pp. 557–576.
5. M. O. Ball and U. Derigs, *An analysis of alternative strategies for implementing matching algorithms*, Networks **13** (1983), 517–549.
6. M. Bern and D. Eppstein, *Mesh generation and optimal triangulation*, Computing in Euclidean Geometry, 2nd Edition (D.-Z. Du and F. Hwang, eds.), World Scientific, Singapore, 1995, pp. 47–123.
7. M. Bern and P. Plassmann, *Mesh generation*, Handbook of Computational Geometry (J. Sack and J. Urrutia, eds.), Elsevier Science, 1997, to appear.
8. T. D. Blacker and M. B. Stephenson, *Paving: A new approach to automated quadrilateral mesh generation*, Int. J. Numer. Methods in Eng. **32** (1991), 811–847.
9. H. Borouchaki, P. J. Frey, and P. L. George, *Unstructured triangular-quadrilateral mesh generation. Application to surface meshing*, Proceedings of the 5th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, USA, 1996, pp. 229–242.
10. J. R. Brauer, ed., *What every engineer should know about finite element analysis*, Marcel Decker Inc., 1993.
11. CPLEX Optimization Inc., *Using the CPLEX callable library, version 4.0*, 1995.
12. U. Derigs, *Programming in networks and graphs*, Lecture Notes in Economics and Mathematical Systems, vol. 300, Springer-Verlag, Berlin, 1988.
13. J. Edmonds, *An introduction to matching*, Lecture notes, University of Michigan, Ann Arbor, 1967.
14. L. Freitag, M. Jones, and P. Plassmann, *An efficient parallel algorithm for mesh smoothing*, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, USA, 1995, pp. 47–58.
15. L. Freitag, C. Ollivier-Gooch, *Tetrahedral mesh improvement using swapping and smoothing*, Int. J. Numer. Methods in Eng. **40** (1997), 3979–4002.
16. L. Freitag, C. Ollivier-Gooch, *A cost/benefit analysis of simplicial mesh improvement techniques as measured by solution efficiency*, Technical report ANL/MSC-P722-6598, Argonne National Laboratory, 1998.

17. A. M. H. Gerards, *Matching*, (M. O. Ball et al., ed.), Handbooks in Operations Research and Management Science, vol. 7, North-Holland, 1995, pp. 135–224.

18. K. Ho-Le, *Finite element mesh generation methods: a review and classification*, Computer-Aided Design **20** (1988), 27–38.

19. B. Joe, *Quadrilateral mesh generation in polygonal regions*, Computer-Aided Design **27** (1995), 209–222.

20. S. A. Mitchell, *Choosing corners of rectangles for mapped meshing*, Proceedings of the 13th Annual ACM Symposium on Computational Geometry, Nice, France, ACM, 1997, pp. 87–93.

21. S. A. Mitchell, *High fidelity interval assignment*, Proceedings of the 6th International Meshing Roundtable, Park City, Utah, Sandia National Laboratories, Albuquerque, USA, 1997, pp. 33–44.

22. R. H. Möhring and M. Müller-Hannemann, *Complexity and modeling aspects of mesh refinement into quadrilaterals*, Proceedings of the 8th Annual International Symposium on Algorithms and Computation, ISAAC'97, Singapore, Lecture Notes in Computer Science **1350**, Springer-Verlag, 1997, pp. 263–273, journal version to appear in Algorithmica.

23. R. H. Möhring, M. Müller-Hannemann, and K. Weihe, *Mesh refinement via bidirected flows: Modeling, complexity, and computational results*, Journal of the ACM **44** (1997), 395–426.

24. M. Müller-Hannemann, *On the generation of finite element meshes with graph theoretical methods*, Diploma thesis, Technische Universität Berlin, 1994.

25. M. Müller-Hannemann and K. Weihe, *Minimum strictly convex quadrangulations of convex polygons*, Proceedings of the 13th Annual ACM Symposium on Computational Geometry, Nice, France, ACM, 1997, pp. 193–202.

26. W. R. Pulleyblank, *Faces of matching polyhedra*, Ph.D. thesis, Faculty of Mathematics, University of Waterloo, 1973.

27. M. Rees, *Combining quadrilateral and triangular meshing using the advancing front approach*, Proceedings of the 6th International Meshing Roundtable, Park City, Utah, Sandia National Laboratories, Albuquerque, USA, 1997, pp. 337–348.

28. T. K. H. Tam and C. G. Armstrong, *Finite element mesh control by integer programming*, Int. J. Numer. Methods in Eng. **36** (1993), 2581–2605.

29. D. R. White and P. Kinney, *Redesign of the paving algorithm: Robustness enhancements through element by element meshing*, Proceedings of the 6th International Meshing Roundtable, Park City, Utah, Sandia National Laboratories, Albuquerque, USA, 1997, pp. 323–335.

30. D. R. White, L. Mingwu, S. E. Benzley, and G. D. Sjaardema, *Automated hexahedral mesh generation by virtual decomposition*, Proceedings of the 4th International Meshing Roundtable, Sandia National Laboratories, Albuquerque, USA, 1995, pp. 165–176.

31. J. Z. Zhu, O. C. Zienkiewicz, E. Hinton, and J. Wu, *A new approach to the development of automatic quadrilateral mesh generation*, Int. J. Numer. Methods in Eng. **32** (1991), 849–866.

32. O. C. Zienkiewicz and R. L. Taylor, *The finite element method*, McGraw Hill, London, 1989.

Appendix A: Examples

We provide some illustrations of instances which have been refined by our algorithm (instances (7), (9), (10), (12), and (13) in Table 1).

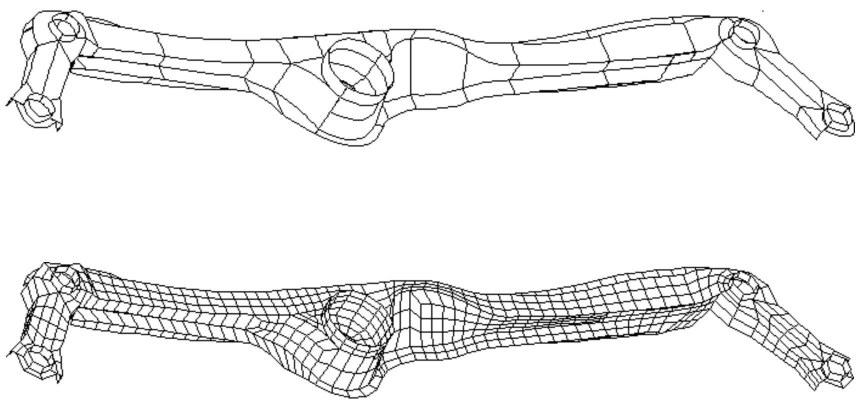

Figure A.1: Model of a chassis and its refinement. (Instance (7) in Tab. 1.)

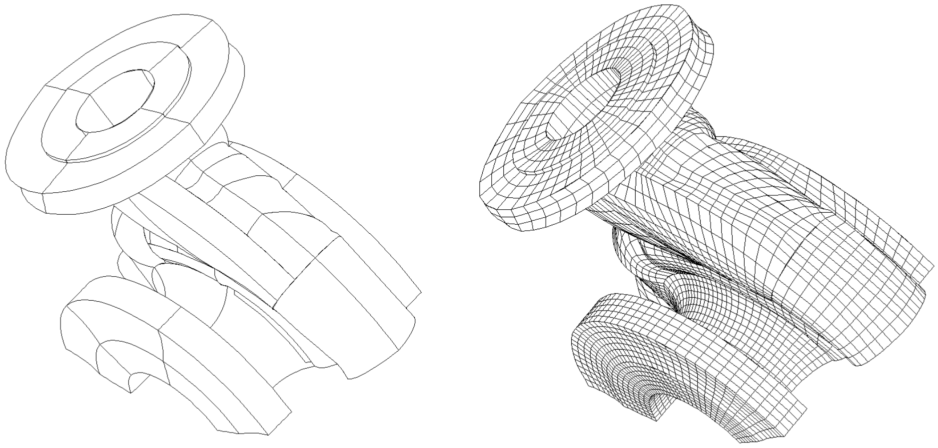

Figure A.2: Model of a pump and its refinement. (Instance (9) in Tab. 1.)

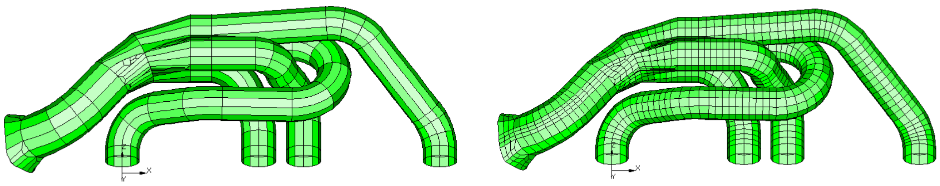

Figure A.3: Model of a bend and its refinement. (Instance (10) in Tab. 1.)

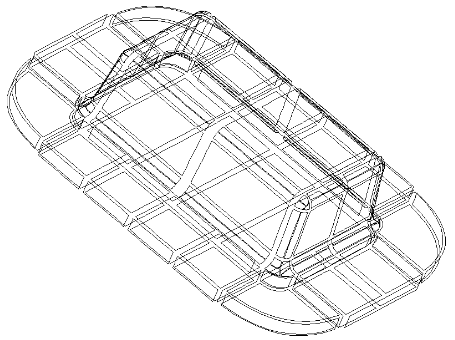

Figure A.4: Wire-frame model of a tub-like workpiece (instance (12) in Tab. 1). The polygons have been shrunk in this visualization to show the hyperplanes inside the model which partition the solid tub into convex parts.

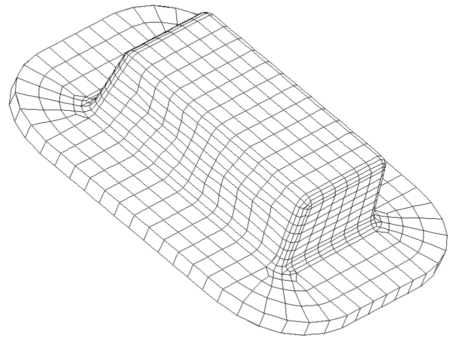

Figure A.5: The refinement produced by our algorithm for the mesh in Fig. A.4.

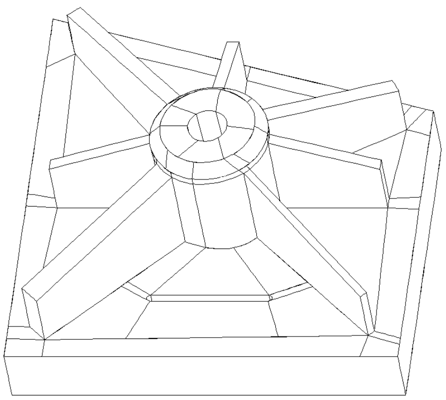

Figure A.6: This solid model (instance (13) in Tab. 1) is partitioned into convex subdomains by the use of internal polygons (which are not visible in this hidden surface representation).

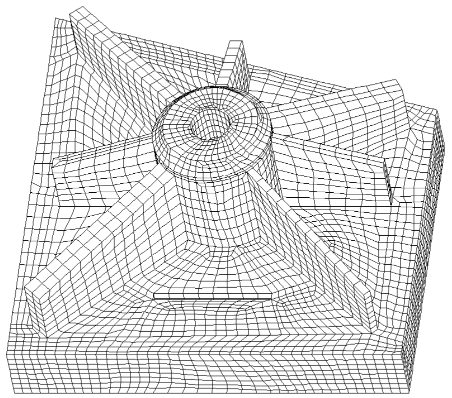

Figure A.7: The refinement produced by our algorithm.

Reports from the group

"Algorithmic Discrete Mathematics"

of the Department of Mathematics, TU Berlin

605/1998 *Friedrich Eisenbrand:* A Note on the Membership Problem for the Elementary Closure of a Polyhedron

596/1998 *Rolf H. Möhring and Frederik Stork and Marc Uetz:* Resource Constrained Project Scheduling with Time Windows: A Branching Scheme Based on Dynamic Release Dates

595/1998 *Rolf H. Möhring and Andreas S. Schulz and Marc Uetz:* Approximation in Stochastic Scheduling: The Power of LP-based Priority Rules

591/1998 *Matthias Müller–Hannemann and Alexander Schwartz:* Implementing Weighted b-Matching Algorithms: Towards a Flexible Software Design

590/1998 *Stefan Felsner and Jens Gustedt and Michel Morvan:* Interval Reductions and Extensions of Orders: Bijections to Chains in Lattices

577/1998 *Martin Skutella:* Semidefinite Relaxations for Parallel Machine Scheduling

566/1997 *Jens Gustedt:* Minimum Spanning Trees for Minor-Closed Graph Classes in Parallel

565/1997 *Andreas S. Schulz, David B. Shmoys, and David P. Williamson:* Approximation Algorithms

564/1997 *Uta Wille:* On Extending Closure Systems to Matroids

561/1997 *Matthias Müller–Hannemann:* High Quality Quadrilateral Surface Meshing Without Template Restrictions: A New Approach Based on Network Flow Techniques

559/1997 *Matthias Müller–Hannemann and Karsten Weihe:* Improved Approximations for Minimum Cardinality Quadrangulations of Finite Element Meshes

554/1997 *Rolf H. Möhring and Matthias Müller–Hannemann:* Complexity and Modeling Aspects of Mesh Refinement into Quadrilaterals

551/1997 *Hans Bodlaender, Jens Gustedt and Jan Arne Telle:* Linear-Time Register Allocation for a Fixed Number of Registers and no Stack Variables

550/1997 *Karell Bertet, Jens Gustedt and Michel Morvan:* Weak-Order Extensions of an Order

549/1997 *Andreas S. Schulz and Martin Skutella:* Random-Based Scheduling: New Approximations and LP Lower Bounds

542/1996 *Stephan Hartmann:* On the NP–Completeness of Channel and Switchbox Routing Problems

536/1996 *Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Bounds on Relaxations of a Parallel Machine Scheduling Problem

535/1996 *Rainer Schrader, Andreas S. Schulz, and Georg Wambach:* Base Polytopes of Series-Parallel Posets: Linear Description and Optimization

533/1996 *Andreas S. Schulz and Martin Skutella:* Scheduling–LPs Bear Probabilities: Randomized Approximations for Min–Sum Criteria

530/1996 *Ulrich H. Kortenkamp, Jürgen Richter-Gebert, Aravamuthan Sarangarajan, and Günter M. Ziegler:* Extremal Properties of 0/1-Polytopes

524/1996 *Elias Dahlhaus, Jens Gustedt and Ross McConnell:* Efficient and Practical Modular Decomposition

523/1996 *Jens Gustedt and Christophe Fiorio:* Memory Management for Union-Find Algorithms

520/1996 *Rolf H. Möhring, Matthias Müller-Hannemann, and Karsten Weihe:* Mesh Refinement via Bidirected Flows: Modeling, Complexity, and Computational Results

519/1996 *Matthias Müller-Hannemann and Karsten Weihe:* Minimum Strictly Convex Quadrangulations of Convex Polygons

517/1996 *Rolf H. Möhring, Markus W. Schäffter, and Andreas S. Schulz:* Scheduling Jobs with Communication Delays: Using Infeasible Solutions for Approximation

516/1996 *Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein:* Scheduling to Minimize Average Completion Time: Off-line and On-line Approximation Algorithms

515/1996 *Christophe Fiorio and Jens Gustedt:* Volume Segmentation of 3-dimensional Images

514/1996 *Martin Skutella:* Approximation Algorithms for the Discrete Time-Cost Tradeoff Problem

509/1996 *Soumen Chakrabarti, Cynthia A. Phillips, Andreas S. Schulz, David B. Shmoys, Cliff Stein, and Joel Wein:* Improved Scheduling Algorithms for Minsum Criteria

508/1996 *Rudolf Müller and Andreas S. Schulz:* Transitive Packing

506/1996 *Rolf H. Möhring and Markus W. Schäffter:* A Simple Approximation Algorithm for Scheduling Forests with Unit Processing Times and Zero-One Communication Delays

505/1996 *Rolf H. Möhring and Dorothea Wagner:* Combinatorial Topics in VLSI Design: An Annotated Bibliography

504/1996 *Uta Wille:* The Role of Synthetic Geometry in Representational Measurement Theory

502/1996 *Nina Amenta and Günter M.Ziegler:* Deformed Products and Maximal Shadows of Polytopes

500/1996 *Stephan Hartmann and Markus W. Schäffter and Andreas S. Schulz:* Switchbox Routing in VLSI Design: Closing the Complexity Gap

498/1996 *Ewa Malesinska, Alessandro Panconesi:* On the Hardness of Allocating Frequencies for Hybrid Networks

496/1996 *Jörg Rambau:* Triangulations of Cyclic Polytopes and higher Bruhat Orders

Reports may be requested from:

S. Marcus
Fachbereich Mathematik, MA 6–1
TU Berlin
Straße des 17. Juni 136
D-10623 Berlin – Germany

e-mail: Marcus@math.TU-Berlin.DE

Reports are available via anonymous ftp from:

ftp.math.tu-berlin.de
cd pub/Preprints/combi
file Report–<number>–<year>.ps.Z