INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING, VOL. 28, 1429–1448 (1989)

A PARALLEL MESH GENERATION ALGORITHM BASED ON THE VERTEX LABEL ASSIGNMENT SCHEME

FUHUA CHENG AND JERZY W. JAROMCZYK

Department of Computer Science, University of Kentucky, Lexington, Kentucky 40506-0027, U.S.A.

JUNNIN-REN LIN

Institute of Computer Science, Tsing Hua University, Hsinchu, Taiwan, R.O.C.

SHYUE-SHIAN CHANG AND JEI-YEOU LU

Chung-Shang Institute of Science and Technology, Lungtang, Taiwan, R.O.C.

SUMMARY

In this article a new mesh generation algorithm is presented. The algorithm is based on a new approach called the vertex label assignment scheme to provide the information for the mesh generation so that parallel processing becomes possible. The algorithm generates 2D meshes of quadrilaterals on the basis of individual faces; conformity and smoothness of the resultant mesh are automatically assured. Local and selective mesh-refinements are also supported. A regular quadrilateral network which defines the geometry of the problem and an associated subdivision level assignment which specifies mesh density data on the network are the only input information.

1. INTRODUCTION

Mesh generation is the process of generating finite element (FE) models for simulated structural analysis. Since the accuracy of the FE solution is dependent on the element mesh layout, and the cost of the analysis becomes prohibitively expensive if the number of elements in the mesh is too large, a good mesh generating method should let the user generate a mesh that is just fine enough to give an adequate solution accuracy. Quadrilateral elements are generally preferred over triangles for reasons of accuracy and efficiency. The *conformity* of the mesh should also be assured to avoid gaps in the structure, i.e. the intersection of two non-disjoint, non-identical elements consists of a common vertex or a common edge.[15]

Current mesh generating techniques can be classified into the following categories:

(i) interpolation mesh generation[2,7,9,10,11,23]
(ii) automatic triangulation,[3,4,8,12,15,16,18,20,21]
(iii) quadtree/octree approach,[1,17,22]
(iv) mesh generation based on constructive solid geometry (CSG).[13,14]

The interpolation mesh generation approach partitions the structure into simpler subregions first and then meshes each subregion individually. Though this technique is quite popular because of its ability to produce well-conditioned meshes, it is relatively difficult to carry out local mesh refinement because of the need to fit together in a consistent manner the individual meshes defined on each subregion of the structure.[5]

0029–5981/89/061429–20$10.00
© 1989 by John Wiley & Sons, Ltd.

Received 8 July 1988
Revised 21 October 1988

Automatic triangulation has been a popular method used in 2D mesh generation. However, this technique occasionally produces ill-conditioned meshes and requires extensive checking to ensure that newly formed elements do not pierce or intersect any already defined. Two recently proposed results have been able to overcome these problems with different degrees of success and make this technique more promising.[5,15]

The third approach is based on quadtree encoding of the object in two dimensions and octree encoding in three dimensions.[1,17,22] The object[17,22] or the domain of the object[1] is enclosed in a square universe, which is then recursively subdivided into quadrants. The quadrants that are inside the object[17,22] or the domain of the object[1] are retained and contribute to the elements of the mesh, those outside are discarded, and those intersecting the boundary are further subdivided. The subdivision repeats until a preselected resolution is reached. The quadrants at the last level of subdivision and intersecting the boundary are cut with respect to the boundary, and selective actions are taken to ensure that the result is a set of valid triangular elements. All the 'in' quadrants are also split into elements while ensuring that a valid mesh is maintained.

Mesh generation based on CSG generates meshes for objects which can be represented by a CSG tree. It uses the CSG tree to construct a set of 'well distributed' points within the space of the CSG object and then uses a decision-making process to construct a mesh over this set of points.[14] Point membership classification[19] is extensively used in the point generation algorithm. The efficiency of the mesh construction algorithm depends on the size of the decision tree. This approach can generate a 'good mesh' for a CSG object with constant mesh density throughout the object. However, extension of this approach to generate meshes of variable density needs special care.

In this paper we shall present an algorithm to generate 2D meshes of quadrilaterals in a different fashion. This algorithm uses a vertex label assignment scheme to provide the information for mesh generation so that parallel mesh generation on the basis of individual faces becomes possible. No checking on the conformity of the resultant mesh is required; the conformity of the resultant mesh is automatically assured. The algorithm allows both global and selective mesh-refinement. Complexity and optimality issues are discussed; the number of quadrilaterals generated by the algorithm in the output mesh is small and the performance of the algorithm is extremely efficient. The geometry of the object and mesh density data are the only inputs. Although we present only the algorithm for networks of quadrilaterals, the method can also be used to generate meshes of quadrilaterals for parametrically defined piecewise polynomial surfaces (such as B-spline surfaces or composite Bézier surfaces).

The remainder of this paper is organized as follows. In Section 2 we shall formally define the problem and give definitions of related terms and concepts used in this paper. The main algorithm is presented in Section 3. The construction of an admissible vertex label assignment is shown in Section 4. Correctness of the algorithm is shown in Section 5. In Section 6 we present the time complexity and the size of the resulting mesh. Finally, in Section 7, we shall make concluding remarks, discuss implementation issues and present a potential direction of future research. Some figures demonstrating the output produced by our algorithm are included in the paper.

2. DEFINITIONS AND NOTATIONS

In this section basic definitions, notions, and notations will be presented. We will start with a standard definition of polyhedral networks.

Consider a surface in 3 dimensional space. A *network* on the surface is a finite set of points (called *nodes*) and curve segments such that

(i) each node is an endpoint of a curve segment,
(ii) each endpoint of a curve segment is a node,
(iii) two curve segments intersect only at their endpoints.

A network divides the surface into *regions*; each region is bounded by curve segments. If the curve segments are segments of straight lines then the corresponding network is called a *polyhedral network*; note that a surface of a polyhedron in R^3 can be naturally associated with such a network. Polyhedral networks can be represented in a plane by planar graphs. A straight line segments embedding of a polyhedral network into a plane will also be called a *polyhedral network*. A standard graph terminology (such as vertices, faces, edges) will be used to refer to objects of polyhedral networks. If all of the bounded faces of a polyhedral network are convex quadrilaterals then the network is referred to as a *quadrilateral network*. In addition, such a network is called a *regular quadrilateral network* if the degree of each vertex (in the corresponding planar graph) belonging only to bounded regions is equal to four. An example of a regular quadrilateral network is shown in Figure 1. Two faces f_1 and f_2 are said to be *adjacent* to each other if they share a common edge.

Let F be the set of all faces of a regular quadrilateral network P. A *subdivision level assignment* S of P is a function defined on F, S: $F \rightarrow N \cup \{0\}$, where N is the set of all positive integers. $S(f)$ is called the *subdivision level* of f for $f \in F$. For a given subdivision level assignment S, a face f is called a *specified face* with respect to S if $S(f) > 0$.Otherwise, it is called a *transitioning face* with respect to S.

Given a regular quadrilateral network P, a quadrilateral network P^* is called a *subdivision mesh* of P if each face of P^* is a subset of a face of P and each face of P is the union of finite faces of P^*. The problem we will study can be formulated as follows.

Given a regular quadrilateral network P and a subdivision level assignment S on P, generate a subdivision mesh P^* of P such that:

(R1) Each specified face f of P is subdivided into at least $4^{S(f)}$ subquadrilaterals.
(R2) The shape of faces generated in P^* is regular, i.e. faces of P^* are not too long or too narrow.

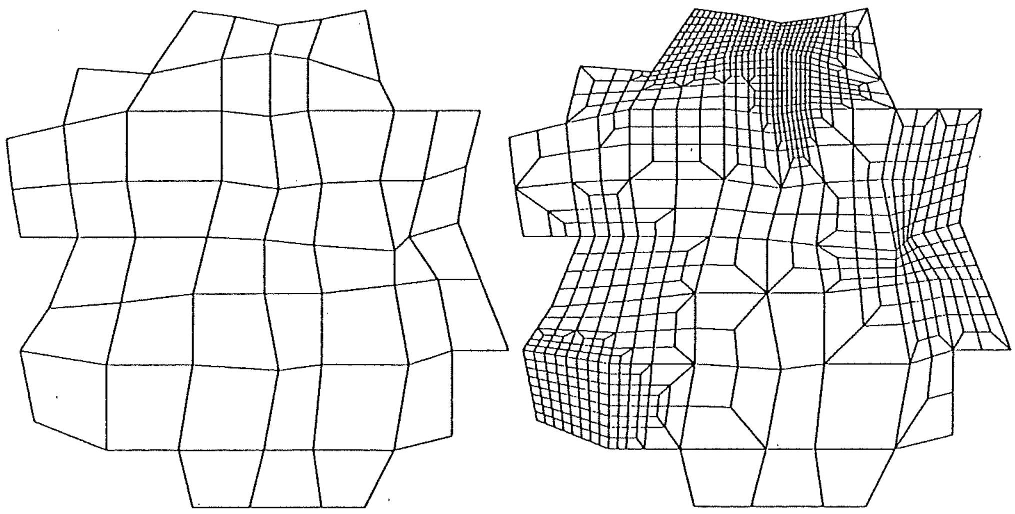
Figure 1. A regular quadrilateral network and a subdivision mesh

(R3) The resultant subdivision mesh P^* is amenable to local modification, i.e. changing the size or shape of some of the faces without affecting the remainder.

(R4) The number of faces generated in P^* is minimal over all subdivision meshes of P satisfying the goal (R1).

In the next section we will present a solution to this problem that meets the goals (R1)–(R3) for a broad class of subdivision level assignments. An efficient algorithm approximating the optimal solution satisfying (R4) will be given also.

3. PARALLEL MESH SUBDIVISION ALGORITHM

The main algorithm of this paper is based on some new concepts listed below:

- vertex label assignment scheme
- admissible label assignment
- balanced subdivision
- unbalanced subdivision

These concepts will be defined in sequence.

Vertex label assignment scheme

The vertex label assignment scheme is the key idea in making parallel mesh generation possible; it allows us to carry out the process of mesh generation by controlling the labels assigned to vertices rather than the subdivision levels assigned to faces of a regular quadrilateral mesh. Therefore, mesh generation can be performed on the basis of individual faces without the danger of violating the conformity requirement.

Let P be a regular quadrilateral mesh, and V and F be the sets of vertices and faces of P, respectively. Consider a subdivision level assignment, S, of P. A *vertex label assignment*, L, of P with respect to S is a function, L: $V \rightarrow N \cup \{0\}$, such that $L(v) = \max\{S(f) | f \in F$ *and* v *is a vertex of* $f\}$.

Figure 2 shows a subdivision level assignment and the corresponding vertex label assignment. $L(v)$ is called the *label* of v with respect to L. A vertex v is called a *supporting vertex* of L if $L(v) = 0$.

Admissible label assignment

Owing to some geometric and algorithmic constraints, which will be discussed in Section 4, the algorithm solving the mesh generation problem will construct a subdivision mesh of P starting

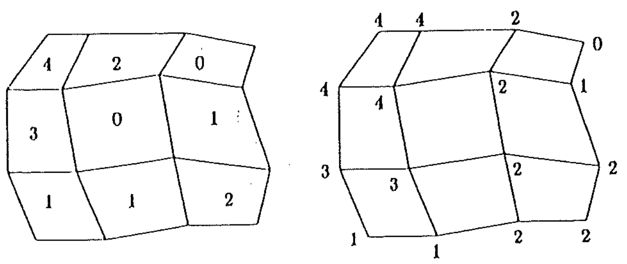

Figure 2. Example of subdivision level assignment and vertex label assignment

from a special type of vertex label assignment, called the *admissible label assignment*. This type of vertex label assignment is defined as follows.

Let L and G be vertex label assignments on the vertices of P. G is called an *extension* of L if the values of G on the complement of the *supporting vertices* of L are the same as those of L, i.e. $G(v) = L(v)$ if $L(v) > 0$. Therefore, if G is an extension of L then $G(v) \geqslant L(v)$ for each vertex v of P. An extension G of L is called an *admissible label assignment* if the following condition holds for every face f of P: if the values of G at two adjacent vertices of f are non-zero then at least one of its values at the two remaining vertices must be non-zero. The problem of constructing the admissible label assignments will be discussed in Section 4.

Balanced and unbalanced subdivisions

The main algorithm will be based on two elementary subdivision procedures.

The procedure *balanced-sub* (f, f_1, f_2, f_3, f_4) will perform a so-called *balanced subdivision* on the quadrilateral $f = v_1 v_2 v_3 v_4$ having at least two non-zero labels assigned to its vertices. This procedure generates four subquadrilaterals $f_1 = q_1 q_2 q_3 q_4$, $f_2 = r_1 r_2 r_3 r_4$, $f_3 = s_1 s_2 s_3 s_4$ and $f_4 = t_1 t_2 t_3 t_4$, and assigns a label to each of their vertices (see Figure 3).

The new vertices are defined in the following fashion.

$$q_1 = v_1, \quad r_2 = v_2, \quad s_3 = v_3, \quad t_4 = v_4$$
$$q_2 = r_1 = (v_1 + v_2)/2, \quad s_2 = r_3 = (v_2 + v_3)/2$$
$$t_3 = s_4 = (v_3 + v_4)/2, \quad t_1 = q_4 = (v_1 + v_4)/2$$
$$q_3 = r_4 = s_1 = t_2 = (v_1 + v_2 + v_3 + v_4)/4$$

Labels assigned to the new vertices are defined as follows.

$$\mathrm{LABEL}(q_1) = \max\{0, \mathrm{LABEL}(v_1) - 1\}$$
$$\mathrm{LABEL}(r_2) = \max\{0, \mathrm{LABEL}(v_2) - 1\}$$
$$\mathrm{LABEL}(s_3) = \max\{0, \mathrm{LABEL}(v_3) - 1\}$$
$$\mathrm{LABEL}(t_4) = \max\{0, \mathrm{LABEL}(v_4) - 1\}$$
$$\mathrm{LABEL}(q_2) = \mathrm{LABEL}(r_1) = \min\{\mathrm{LABEL}(q_1), \mathrm{LABEL}(r_2)\}$$
$$\mathrm{LABEL}(r_3) = \mathrm{LABEL}(s_2) = \min\{\mathrm{LABEL}(r_2), \mathrm{LABEL}(s_3)\}$$
$$\mathrm{LABEL}(s_4) = \mathrm{LABEL}(t_3) = \min\{\mathrm{LABEL}(s_3), \mathrm{LABEL}(t_4)\}$$
$$\mathrm{LABEL}(t_1) = \mathrm{LABEL}(q_4) = \min\{\mathrm{LABEL}(t_4), \mathrm{LABEL}(q_1)\}$$
$$\mathrm{LABEL}(q_3) = \mathrm{LABEL}(r_4) = \mathrm{LABEL}(s_1) = \mathrm{LABEL}(t_2)$$
$$= \begin{cases} 0 & \textit{if } q_2, r_3, s_4 \textit{ and } t_1 \textit{ are assigned zero label} \\ \min\{\mathrm{LABEL}(v) \mid v \in \{q_2, r_3, s_4, t_1\}, \mathrm{LABEL}(v) > 0\} & \textit{otherwise} \end{cases}$$

For a quadrilateral $f = v_1 v_2 v_3 v_4$ with exactly one non-zero label, the procedure *unbalanced-sub* (f, f_1, f_2, f_3) performs a so-called *unbalanced subdivision* with respect to the vertex whose label is non-zero to create three quadrilaterals $f_1 = q_1 q_2 q_3 q_4$, $f_2 = r_1 r_2 r_3 r_4$ and $f_3 = s_1 s_2 s_3 s_4$, and assign labels to their vertices. For instance, if v_1 is the vertex with a non-zero label then the vertices and their labels are defined in the following way (see Figure 4). Other cases can be defined

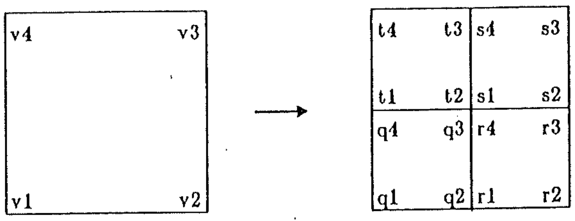

Figure 3. Balanced subdivision

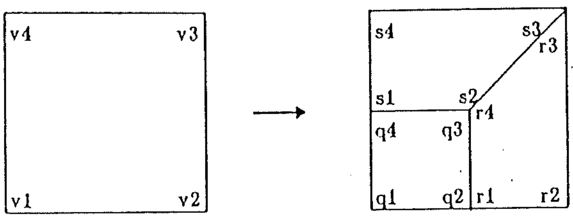

Figure 4. Unbalanced subdivision

similarly.

$$q_1 = v_1, \quad r_2 = v_2, \quad r_3 = s_3 = v_3, \quad s_4 = v_4$$

$$q_2 = r_1 = (v_1 + v_2)/2, \quad s_1 = q_4 = (v_1 + v_4)/2$$

$$s_2 = q_3 = r_4 = (v_1 + v_2 + v_3 + v_4)/4$$

$$\text{LABEL}(q_1) = \text{LABEL}(v_1) - 1$$

$$\text{LABEL}(q_i) = 0, \quad i = 2, 3, 4$$

$$\text{LABEL}(r_j) = 0, \quad j = 1, 2, 3, 4$$

$$\text{LABEL}(s_k) = 0, \quad k = 1, 2, 3, 4.$$

The main algorithm

The parallel mesh generation algorithm is presented in this subsection. The algorithm uses two types of subdivision scheme defined in the previous section to subdivide the given regular quadrilateral network into a subdivision mesh. The sub-division process is entirely driven by the values of labels assigned to the vertices of the input network. After two preprocessing phases, which are responsible for the construction of an admissible label assignment, the algorithm performs the subdivision for all the faces simultaneously; one processor for one face. Each face of the network represents the root of a *quadtree*, and is subdivided into three or four quadrilaterals, depending on the labels of its vertices. New labels are assigned to the vertices of the quadrilaterals. These quadrilaterals are then recursively subdivided into subquadrilaterals and labels are also assigned to the vertices of these subquadrilaterals. This process is continued for each quadrilateral until all labels are equal to zero.

The overall structure of the parallel algorithm is given in Figure 5, parallel mesh geneation. In this algorithm and hereafter, a block of instructions will be called a *paralell step* if it is enclosed by the keywords **PARDO** and **DOPAR**. Any instructions contained in a parallel step are supposed

```
Algorithm PMG: Parallel mesh generation
{input: a regular quadrilateral network $P$ and a subdivision level assignment $S$ on $P$}
{output: a subdivision mesh $P^*$ of $P$}
Phase 1: [Construct the vertex label assignment $L$ of $P$ with respect to $S$.]
         PARDO for each vertex $v$ of $P$ do
               $L(v) := \max\{S(f) | f \in F,\ v$ is a vertex of $f\}$
         DOPAR
Phase 2: [Construct an admissible extension $G$ of $L$ and define the LABEL for each vertex $v$ of $P$]
         Construct the admissible extension $G$ of $L$; {see Section 4}
         PARDO for each vertex $v$ of $P$ do
               LABEL $(v) := G(v)$
         DOPAR
Phase 3: [Subdivide the faces of $P$ in parallel]
         PARDO for each face $f$ of $P$ do
               push $(f, ST(f))$;
               while not empty $(ST(f))$ do
                     begin
                           $g := pop(ST(f))$;
                           subdivide$(g)$;
                     end;
         DOPAR
```

Figure 5. Parallel mesh generation

to be executed in parallel for all faces or vertices of the input regular quadrilateral network. Instructions not enclosed by the keywords **PARDO** and **DOPAR** are assumed to be executed in sequential order. *push* $(f, ST(f))$ is a procedure which puts the face f on top of the stack $ST(f)$ defined for the face f; $pop(ST(f))$ is a function which returns the quadrilateral on top of the stack $ST(f)$. *subdivide*(g), a procedure which performs a balanced or unbalanced subdivision on the input quadrilateral g depending on the LABELs of its vertices, is defined in Figure 6, subdivide procedure.

Phase 3 of the parallel mesh generation algorithm can also be stated in a recursive fashion, which is more suitable for space and time complexity analysis. The recursive version is given below.

```
PARDO for each face $g$ of $P$ do
      subdivide1$(g)$;
DOPAR
```

where *subdivide1*(g) is defined in Figure 7, procedure subdivide1.

The construction of the admissible extension G of the vertex label assignment L will be shown in Section 4.

4. THE CONSTRUCTION OF AN ADMISSIBLE EXTENSION

The fact that the construction of the subdivision mesh in algorithm PMG should be based on the admissible label assignment is implied by the conformity requirement for the resultant mesh (see Section 1) and the desire to organize the subdivision process so that it is driven solely by labels assigned to the vertices of the quadrilateral network. To see this, consider the subnetwork depicted in Figure 8(a) where labels of the vertices form a non-admissible label assignment. The

```
subdivide (g:quadrilateral);
    begin
        if(LABEL(v) > 0 for more than one vertex v of g) then
            begin
                balanced-sub(g, g_1, g_2, g_3, g_4);
                push (g_1, g_2, g_3, g_4, ST(f));
            end
        else if (LABEL(v) > 0 for only one vertex v of g) then
            begin
                unbalanced-sub(g, g_1, g_2, g_3);
                push(g_1, g_2, g_3, ST(f));
            end;
    end {subdivide};
```

Figure 6. Subdivide procedure

```
subdivide1(g:quadrilateral);
    begin
        if (LABEL(v) > 0 for more than one vertex v of g) then
            begin
                balanced-sub(g, g_1, g_2, g_3, g_4);
                subdivide1(g_1);
                subdivide1(g_2);
                subdivide1(g_3);
                subdivide1(g_4);
            end
        else if (LABEL(v) > 0 for only one vertex v of g) then
            begin
                unbalanced-sub(g, g_1, g_2, g_3);
                subdivide1(g_1);
                subdivide1(g_2);
                subdivide1(g_3);
            end
    end; {subdivide1}
```

Figure 7. Procedure subdivide1

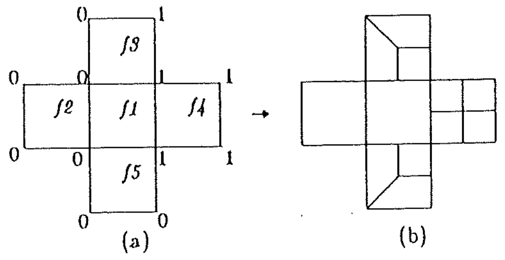

Figure 8

application of the procedure *subdivide*(f) to the faces f_i, $i=2, 3, 4, 5$ would generate three extra vertices on the boundaries of face f_1 (Figure 8(b)). Under this circumstance, it is not possible to subdivide face g_1 into subquadrilaterals without violating the conformity requirement. Assume, for example, that f_1 was subdivided into S quadrilaterals. Since each subquadrilateral has four edges and each edge, except the seven edges on the boundary of f_1, is shared by two subquadrilaterals, there should be $E=(4S+7)/2$ edges in the subdivision mesh of f_1. This leads to a contradiction because the number of edges should clearly be an integer.

A label assignment is non-admissible if it contains one of the four cases shown in Figure 9. The preprocessing phase of our algorithm is responsible for removing such cases (called *illegal* cases). This can always be achieved by changing some 0 labels to strictly positive labels. Note, however, that the space complexity (and, consequently, the time complexity) of the subdivision procedure depends on current values of the labels. In fact, the overall cost of the algorithm depends on two factors: where the 0 labels are located and how many of them are to be changed.

It turns out that the admissible extension construction is relatively difficult owing to its global character; the modification of labels of vertices directly related to the illegal case can create new illegal cases.

In this section we will study the problem of constructing admissible extensions effectively. We will limit our attention to minimizing the number of 0 labels to be changed in order to remove all forbidden situations in a given vertex label assignment. In fact, we will develop a simple theory of admissible extensions which will enable us to design an efficient algorithm in that direction.

Let us assume, without loss of generality, that the label assignments considered hereafter are defined on a regular quadrilateral network whose vertices form an $m \times n$ rectangular grid $V = \{v_{ij} | 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\}$. In fact, if the vertices of the given quadrilateral network do not form a rectangular grid then we can simply add some extra vertices to make one and set labels of these vertices to zero; these vertices will eventually be deleted.

Let us introduce some definitions. A face $f_{i,j}$ is defined by four vertices $v_{i,j}$, $v_{i+1,j}$, $v_{i+1,j+1}$ and $v_{i,j+1}$ where $1 \leqslant i \leqslant m-1$ and $1 \leqslant j \leqslant n-1$. For any vertex $v_{i,j}$ of the given network, $v_{i+1,j}$, $v_{i,j+1}$, $v_{i-1,j}$ and $v_{i,j-1}$ (if they exist) are called the *adjacent vertices* of $v_{i,j}$. The pairs of vertices, $(v_{i,j}, v_{i+1,j+1})$ and $(v_{i+1,j}, v_{i,j+1})$, are called *opposite vertices* of $f_{i,j}$.

Let L be a label assignment on V, i.e. $L: V \rightarrow N \cup \{0\}$. Furthermore, let $L(v_{i,j})$ denote the label of $v_{i,j}$. The *supporting set* of L, denoted by V_0^L, is the set of all vertices v_{ij} in V such that $L(v_{ij}) = 0$. The complement of V_0^L in V, $V - V_0^L$, is referred to as V_1^L. According to our definition, L is an *admissible* label assignment of V if and only if the following condition is satisfied: for any face $f_{i,j}$, if the value of L is equal to zero for exactly two vertices then these vertices are the opposite vertices of $f_{i,j}$.

Now, formally, an admissible label assignment G on V is called an *admissible extension* of L if

$$G(v) = \begin{cases} L(v), & v \in V_1^L \\ 0 \text{ or } 1, & v \in V_0^L \end{cases}$$

Note that the extension G of L such that $G(v) = 1$ for all $v \in V_0^L$ is an admissible extension of L. G is called a *direct admissible extension* (or DAE, for short) of L if there is no admissible extension H of L such that $V_0^G \subset V_0^H \subset V_0^L$.

Ideally, given a label assignment L of V we would like to find an admissible extension G of L such that $|V_0^G| \geqslant |V_0^H|$ for each admissible extension H of L. Note that such an extension is necessarily direct and always exists (the set of admissible extensions of a label assignment L is non-empty and finite). Unfortunately, we are not in a position to find such an extension

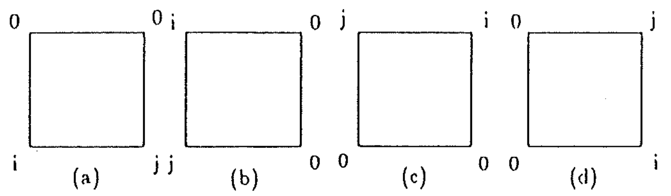

Figure 9. Illegal vertex label assignments

efficiently. We will present, instead, a simple approximation algorithm running in linear time with respect to the number of vertices; the admissible extension constructed by this algorithm will approximate the optimal solution by a factor of two. More precisely, if *opt* is the optimal direct admissible extension of a given label assignment L and G is the admissible extension constructed by our algorithm then $|V_0^G| \geqslant \frac{1}{2}|V_0^{opt}|$.

Toward this end, an operation $\oplus$ on label assignments is required. For technical reasons we shall allow hereafter $\frac{1}{2}$ to be a value of label assignments.

Let two label assignments F and G be defined on V such that $F(v_{ij})$ and $G(v_{ij})$ are either 0, $\frac{1}{2}$ or any positive integer. The $\oplus$-product of F and G, $F \oplus G: V \to N \cup \{0, \frac{1}{2}\}$, is defined as follows:

$$F \oplus G(v_{ij}) = \max\{F(v_{ij}), G(v_{ij})\}, \quad v_{ij} \in V$$

Two special label assignments L_o and L_e are defined as follows:

$$L_o(v_{ij}) = \begin{cases} \frac{1}{2} & \text{if } i+j \textit{ is odd} \\ 0 & \text{if } i+j \textit{ is even} \end{cases}$$

$$L_e(v_{ij}) = \begin{cases} \frac{1}{2} & \text{if } i+j \textit{ is even} \\ 0 & \text{if } i+j \textit{ is odd} \end{cases}$$

Note that L_o and L_e are both admisssible label assignments and $L_o \oplus L$ and $L_e \oplus L$ are admissible extensions of L.

Now we are ready for the approximation algorithm which is presented in Figure 10, algorithm AEC.

```
Algorithm AEC: Admissible extension construction
{input: a label assignment $L$ on $V$}
{output: an admissible extension $G$ of $P$}

1. [Construct $G_e$]
  1.1. [Set $G_e := L \oplus L_e$]
       PARDO for each vertex $v_{ij}$ of $P$ do
           $G_e(v_{ij}) := L \oplus L_e(v_{ij})$
       DOPAR
  1.2. PARDO for each $v_{ij}$ such that $G_e(v_{ij}) = \frac{1}{2}$ do
           if $G_e(v) > 1/2$ for at least one adjacent vertex $v$ of $v_{ij}$
               then $G_e(v_{ij}) := 1$
               else $G_e(v_{ij}) := 0$;
       DOPAR

2. [Construct $G_o$]
  2.1. [Set $G_o := L \oplus L_o$]
       PARDO for each vertex $v_{ij}$ of $P$ do
           $G_o(v_{ij}) := L \oplus L_o(v_{ij})$
       DOPAR

  2.2. PARDO for each $v_{ij}$ such that $G_o(v_{ij}) = 1/2$ do
           if $G_o(v) > 1/2$ for at least one adjacent vertex $v$ of $v_{ij}$
               then $G_o(v_{ij}) := 1$
               else $G_o(v_{ij}) := 0$;
       DOPAR

3. [Construct $G$]
       if $|V_0^{G_o}| > |V_0^{G_e}|$
           then return $G := G_o$
           else return $G := G_e$
```

Figure 10. Algorithm AEC

The time complexity of algorithm AEC is proportional to the number of vertices in the input network because Step 3 requires a linear scan of the vertices. The correctness of the algorithm, i.e. it finds an admissible (although not necessarily direct) extension of the input label assignment, is proved in the following theorem.

Theorem 4.1

The label assignment G returned by algorithm AEC is an admissible extension of the input label assignment L.

Proof. It is easy to see that the label assignment G_e constructed in Step 1.1 is an admissible extension of L. Then in Step 1.2 the value of G_e at a vertex v_{ij} is changed from $\frac{1}{2}$ to zero only if $G_e(v) = 0$ for all adjacent vertices v of v_{ij}; it does not affect the admissibility of G_e. Therefore, after Step 1.2, G_e is again an admissible extension of L. A similar argument shows that G_o is an admissible extension of L as well. Hence, the returned label assignment is an admissible extension of L. □

The significance of algorithm AEC follows from the following fact.

Theorem 4.2

$$|V_0^G| \geqslant \tfrac{1}{2}|V_0^L|.$$

Proof. Let L be the given label assignment. Observe that $V_0^{L \oplus L_e} \cup V_0^{L \oplus L_o} = V_0^L$ and $V_0^{L \oplus L_e} \cap V_0^{L \oplus L_o} = \phi$. Hence, at least one of $V_0^{L \oplus L_e}$ and $V_0^{L \oplus L_o}$ covers one half of V_0^L. Consequently, G returned by the algorithm satisfies the theorem. □

The above theorem says that not 'too many' 0's are changed to 1's by algorithm AEC. Specifically, at most half of them. It should be pointed out, however, that the label assignment returned by algorithm AEC is not guaranteed to be a direct admissible extension of the input label assignment; thus, the returned extension is not necessarily optimal. The following corollary is an immediate consequence of Theorem 4.2 and the inequality $|V_0^L| \geqslant |V_0^{opt}|$.

Corollary 4.3

An admissible extension G of L such that $|V_0^{opt}|/|V_0^G| \leqslant 2$, where *opt* is an optimal direct admissible extension of L, can be constructed in linear time with respect to the number of vertices.

5. CORRECTNESS OF THE ALGORITHM

Note first that each step of the parallel mesh generation algorithm decreases at least one label. Hence, in finite number of steps all the labels become zero and the algorithm halts.

In order to prove that the resultant mesh is a subdivision mesh in our sense we need to show that the conformity condition is satisfied, i.e. no element of the generated mesh has more than four vertices on its boundary. It follows from the following observations.

Lemma 5.1

The condition that no two adjacent vertices of a quadrilateral are labelled with 0 while the remaining two vertices have positive labels is an invariant of the algorithm,i.e. it is satisfied in each step of the algorithm.

Proof. The second phase of the algorithm generates an admissible label assignment; therefore no label assignment violating the condition may occur on the faces of the input network. The lemma will be proved if we can show that none of the illegal cases shown in Figure 9 can occur in Phase 3. Assume, for example, that the case shown in Figure 9(a) occurred. Then it must be a subquadrilateral of one of the quadrilaterals shown in Figure 11 (the hatched ones). However, it is not possible because of the following arguments:

(i) the label of the midpoint of a boundary (if it is split after a subdivision) is less than or equal to the new labels of its endpoints and this excludes, therefore, cases (a), (d), (e), (f), (g) and (h);
(ii) the label of the inner vertex generated after a balanced subdivision is equal to the minimum of the non-zero labels of the four midpoints, and will equal zero only if the labels of the midpoints are all zero; therefore, cases (b) and (c) can not happen either.

Other cases in Figure 9 can be proved in a similar way. □

Now consider an edge e of some quadrilateral; the labels of the endpoints of e are i and j, respectively. Let $V(i,j)$ denote the number of vertices created by the algorithm between these endpoints. It turns out that this number depends on i and j only.

Theorem 5.1

If $j \geqslant i$ then $V(i,j) = 2^i + j - i - 1$.

Proof. The proof is carried out by induction on i and j. For $i = j = 0$ by virtue of Lemma 5.1 either no subdivision or an unbalanced subdivision with respect to one of the remaining vertices is performed; the edge will not be subdivided, and therefore the number of the generated vertices is 0. Assume that $V(0,j) = j$ for some $j > 0$. When the endpoints of the edge are labelled by 0 and $j+1$ the algorithm performs a (balanced or unbalanced) subdivision and creates a new vertex, in the middle of the edge, labelled by 0. The label $j+1$ is decreased by 1. Hence, by induction hypothesis, $V(0,j+1) = 1 + V(0,0) + V(0,j) = j+1$. Assume now that $V(i',j') = 2^{i'} + j' - i' - 1$ for all of the pairs (i',j'), where $0 \leqslant i' \leqslant j' \leqslant j$ for some $j \geqslant 0$. Consider an edge labelled by i and $j+1$ with $i \leqslant j+1$. The first application of the (balanced or unbalanced) subdivision to this edge creates a new vertex, in the middle of the edge, labelled by $\max\{0, i-1\}$. The new labels of the endpoints are $\max\{0, i-1\}$ and j, respectively. Denote $\max\{0, i-1\}$ by r.

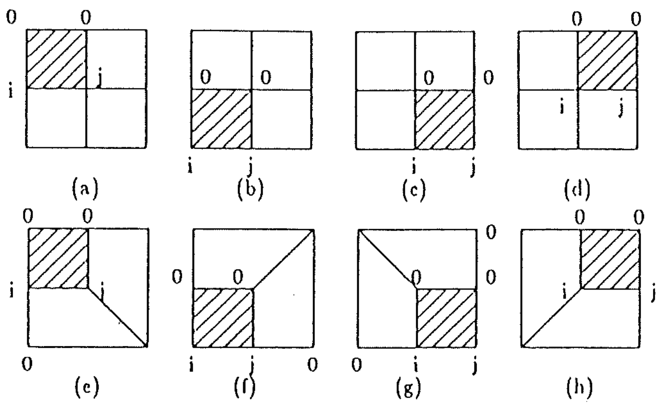

Figure 11. Quadrilaterals which might contain a subquadrilateral of the type shown in Figure 9(a)

Then by induction hypothesis $V(i, j+1) = 1 + V(r, r) + V(r, j) = 2^i + (j+1) - i - 1$. Note also that the distribution of the created vertices depends on the labels of the endpoints only. This ends the proof.

The above theorem shows that the number and the distribution of the vertices generated by algorithm PMG between the endpoints of an edge depend only on the labels of the endpoints. Consequently, the conformity of the mesh generated by the algorithm follows.

The subdivision mesh generated by algorithm PMG satisfies requirements (R1), (R2) and (R3). This can be shown as follows.

Let f be a specified face with respect to the given subdivision level assignment S, i.e. $S(f) > 0$. According to the definitions of L (Section 3) and the construction of G (Section 4), the minimal label of the vertices of f assigned by G is greater than or equal to $S(f)$. Hence, a balanced subdivision will be performed on f. On the other hand, the minimal label of each subquadrilateral of this face generated after the balanced subdivision is greater than or equal to $S(f) - 1$, and each of these subquadrilaterals will again be subdivided using the balanced subdivision if $S(f) - 1 > 0$. This process also applies to the subquadrilaterals generated subsequently and will repeat $S(f)$ times. After this point, further subdivision might be needed for some of the newly generated subquadrilaterals. However, by that time, $4^{S(f)}$ subquadrilaterals have already been generated. Consequently, if f is a specified face with respect to S then it will be subdivided into at least $4^{S(f)}$ subquadrilaterals and requirement (R1) is satisfied.

The satisfiability of requirement (R2) follows from the observation that the nature of the subdivision scheme and label assignment scheme used in Phase 3 forbids any further subdivision of any quadrilaterals generated after an unbalanced subdivision whose labels are all zero. Since this is the only occasion when the aspect ratios and sizes of interior angles of the subquadrilaterals would be changed by a factor of approximately 2 (otherwise, they would be approximately the same), it guarantees that no subquadrilaterals which are too long or too narrow will be generated by algorithm PMG.

Furthermore, note that changing the subdivision level of a face or changing the label of a vertex affects the mesh layout on a small area of the mesh only, i.e. the neighbouring faces. Therefore, requirement (R3) is satisfied as well.

As far as requirement (R4) is concerned, since the admissible extension G of the label assignment L constructed in Phase 2 is not guaranteed to be optimal, the number of quadrilaterals generated in the subdivision mesh usually is not minimal over all subdivision meshes of the input quadrilateral network which satisfy requirement (R1). Note that an optimal direct admissible extension would not guarantee the corresponding subdivision mesh to be optimal either. However, since the corresponding subdivision mesh of an optimal direct admissible extension is asymptotically the optimal subdivision mesh for the input subdivision level assignment and the admissible extension G constructed in Phase 2 differs from the optimal direct admissible extension by a factor of at most 2 with respect to the number of supporting vertices of L, the subdivision mesh generated by our algorithm is actually a good approximation to the optimal subdivision mesh of the input regular network. This is especially true when the number of supporting vertices is small or when the values of the subdivision levels are relatively large.

6. COMPLEXITY ANALYSIS OF THE ALGORITHM

This section discusses the space and time complexity issues of algorithm PMG. Note that, since the construction of the label assignment L of the input regular quadrilateral network can be done simultaneously for all the vertices, the time complexity of Phase 1 is constant. On the other hand, although Step 1 and Step 2 of algorithm AEC would each require only constant time to construct G_e and G_o, respectively, since the testing of the condition in Step 3 requires the scanning of all the vertices in sequential order, the time complexity of Phase 2 is $O(|V|)$ where V is the set of vertices of the input regular quadrilateral network.

The time complexity of Phase 3 depends on the number of subquadrilaterals to be generated and on the number of processors. With a full parallelization (i.e. one processor per face) the face with the largest time complexity determines the complexity of Phase 3.

The construction of the subdivision mesh of an individual face can be viewed as the traversal of a quadtree: the root of the tree represents the given face, each other node of the tree represents a subquadrilateral generated after a balanced or an unbalanced subdivision. The number of *leaves* in this tree represents the number of subquadrilaterals generated in the resultant subdivision mesh and the number of *nodes* in the tree represents the number of iterations of the procedure *subdivide*(f). Therefore, studying the time and space complexity of Phase 3 is equivalent to finding the numbers of leaves and nodes in the quadtree for each face of the given regular quadrilateral network.

Let v_1, v_2, v_3 and v_4 be the vertices of a given quadrilateral (in the counterclockwise direction) with labels i, j, k and l, respectively. Without loss of generality we shall assume that $l = \min\{i, j, k, l\}$. Let $N(i, j, k, l)$ denote the number of subquadrilaterls generated in the subdivision mesh of this quadrilateral and $T(i, j, k, l)$ denote the number of iterations of the procedure *subdivide*(f) in the construction of the mesh. Using recursive relations explicit expressions can be found for both $N(i, j, k, l)$ and $T(i, j, k, l)$. In the following, however, we shall present the main results for the most interesting cases only. Details and proofs can be found in the technical report.[6]

Remark: Note that, as long as the labels i, j, k and l are listed in the counterclockwise direction, it does not matter which one is used as the leading parameter in N and T (i.e. N and T are *cyclosymmetric*). For instance, it should be understood that $N(i, j, k, l) = N(j, k, l, i)$ and $T(i, j, k, l) = T(k, l, i, j)$. Furthermore, due to the fact that the values of N and T depend on the relative order of its labels, but do not depend on the direction these labels are listed, it should also be clear that $N(i, j, k, l) = N(l, k, j, i)$ and $T(i, j, k, l) = T(l, k, j, i)$.

Theorem 6.1

If all of the labels are positive, i.e. $l > 0$, then we have the following results.

1. $i = \max\{i, j, k\}$
   1.1. If $i \geqslant j \geqslant k \geqslant l$ then

$$\begin{aligned} N(i,j,k,l) &= 4^l - 2^{l+3} + 7(2^{j-1} + 2^{k-1}) + 4^{k-l} - 4^{j-l} - 2^{k-l+3} + \frac{4^{j-1}}{2^{l-1}} \\ &\quad + \frac{4^{k-1}}{2^{l-1}} + \frac{4^j}{2^{l+k}} + 2(i-j-k+l) + 7 \\ T(i,j,k,l) &= \frac{4^{l+1}}{3} + 5(2^k + 2^j) + \frac{1}{3}\left(\frac{4^k}{2^{l-1}} + \frac{4^j}{2^{l-1}} + \frac{4^{j+1}}{2^{l+k}}\right) + \frac{4^{k-l+1}}{3} - \frac{4^{j-l+1}}{3} \\ &\quad - \frac{17}{3}(2^{k-l+1} + 2^{l+1}) + 3(i-j-k+l) + \frac{29}{3}. \end{aligned} \tag{1}$$

1.2. If $i \geqslant k > j \geqslant l$ then

$$N(i,j,k,l) = 4^l - 2^{l+3} + 7(2^j - 2^{j-l}) + \frac{4^j}{2^l} + 2(i+k-3j+l) + 7$$

$$T(i,j,k,l) = \frac{4^{l+1}}{3} - \frac{17}{3}2^{l+1} + 5(2^{j+1} - 2^{j-l+1}) + \frac{1}{3}\frac{4^{j+1}}{2^l} + 3(i+k-3j+l) + \frac{29}{3} \qquad (2)$$

2. $j = \max\{i,j,k\}$

2.1. If $j \geqslant i \geqslant k \geqslant l$ then

$$N(i,j,k,l) = 4^l - 2^{l+3} + 7(2^{i-1} + 2^{k-1}) + 4^{k-l} - 4^{i-l} - 2^{k-l+3} + \frac{4^{i-1}}{2^{l-1}} + \frac{4^{k-1}}{2^{l-1}} + \frac{4^i}{2^{l+k}} + 2(j-i-k+l) + 7$$

$$T(i,j,k,l) = \frac{4^{l+1}}{3} + 5(2^k + 2^i) + \frac{2}{3}\left(\frac{4^k}{2^l} + \frac{4^i}{2^l} + \frac{4^{i-l+1}}{2^{k-l+1}}\right) - \frac{17}{3}(2^{l+1} + 2^{k-l+1}) + \frac{4^{k-l+1}}{3} - \frac{4^{i-l+1}}{3} + 3(j-i-k+l) + \frac{29}{3} \qquad (3)$$

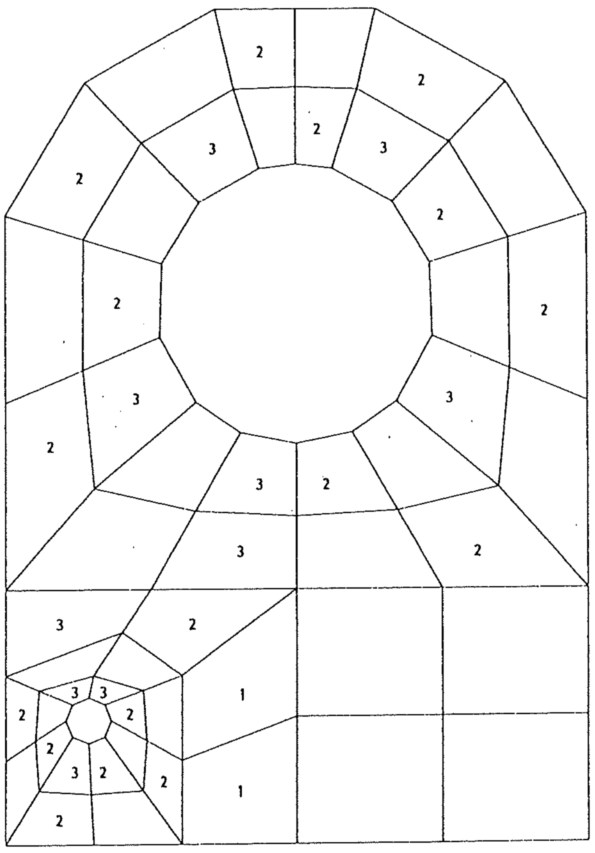

Figure 12(a). Regular quadrilateral network corresponding to a disc and the associated subdivision level assignment; blank faces indicate zero level

2.2. If $j \geqslant k \geqslant i \geqslant l$
Simply exchange i and k in (3) to get the corresponding expressions for $N(i, j, k, l)$ and $T(i, j, k, l)$.

3. $k = \max\{i, j, k\}$
Similar to case 1. Simply exchange i and k in the corresponding expressions (1) and (2).

The proof of these results uses recursive relations induced by the recursive nature of the algorithm.[6] On the basis of the equations given above it can be shown that if $l > 0$ then $N(i, j, k, l) \geqslant 4^l$. It provides another justification for the satisfiability of the requirement (R1).

7. CONCLUSIONS

A parallel mesh generation algorithm based on the vertex label assignment scheme has been presented. For a given regular quadrilateral network and an associated subdivision level assignment, the algorithm first constructs an admissible label assignment based on the given subdivision level assignment, and then simultaneously subdivides all the faces of the network, according to the labels of their vertices, into subquadrilaterals and repeats the same process

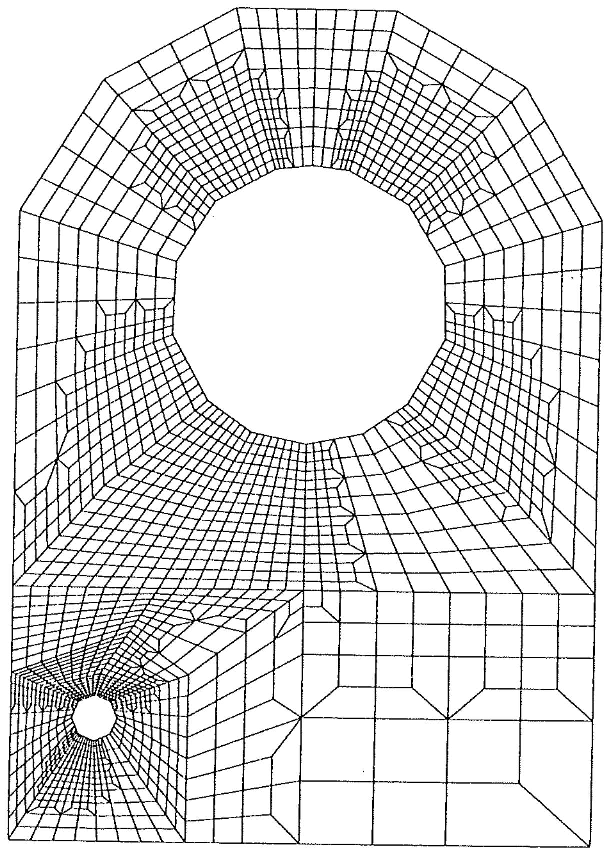

Figure 12(b). Subdivision mesh for the regular quadrilateral network and the associated subdivision level assignment shown in Figure 12(a) recursively to all the subquadrilaterals until the labels of the vertices of the subquadrilaterals are all zero. No checking on the conformity of the resultant mesh is required; the conformity of the resultant mesh is automatically assured. Since changing the subdivision level of a face or the label of a vertex would only affect the mesh layout on a small area of the network, this algorithm allows local and selective mesh refinement.

Note that, since the mesh generation algorithm is based on an abstract geometric model, namely, a regular quadrilateral network, it can easily be modified to generate quadrilateral meshes on piecewise polynomial surfaces such as B-spline surfaces or composite Bézier surfaces as long as appropriate subdivision algorithms are used; simply take each patch of the piecewise surface as a face of the regular quadrilateral network and each joint of the piecewise surface as a vertex of the regular quadrilateral network. Therefore, if an appropriate surface interpolation

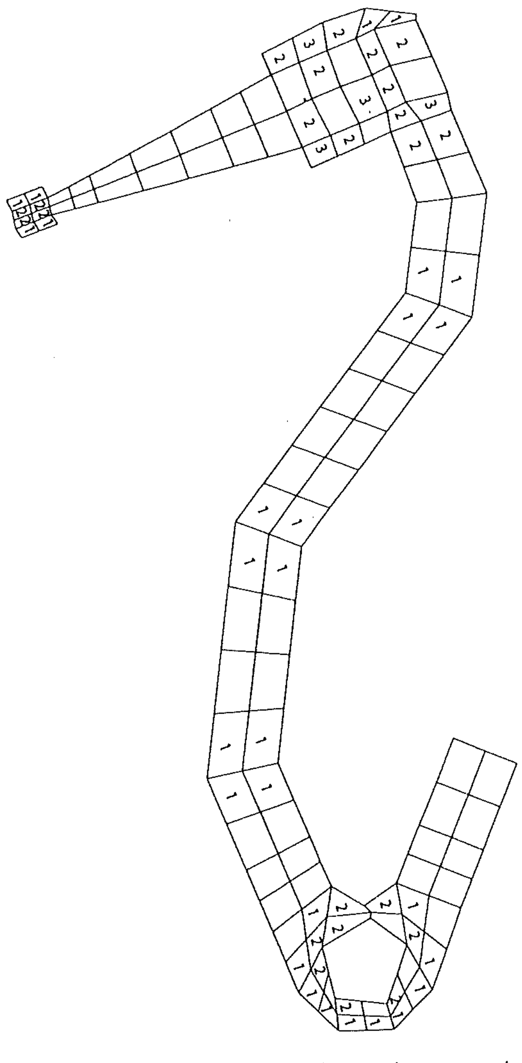
Figure 13(a). Regular quadrilateral network corresponding to a disc arm and the associated subdivision level assignment. Blank faces indicate zero level.

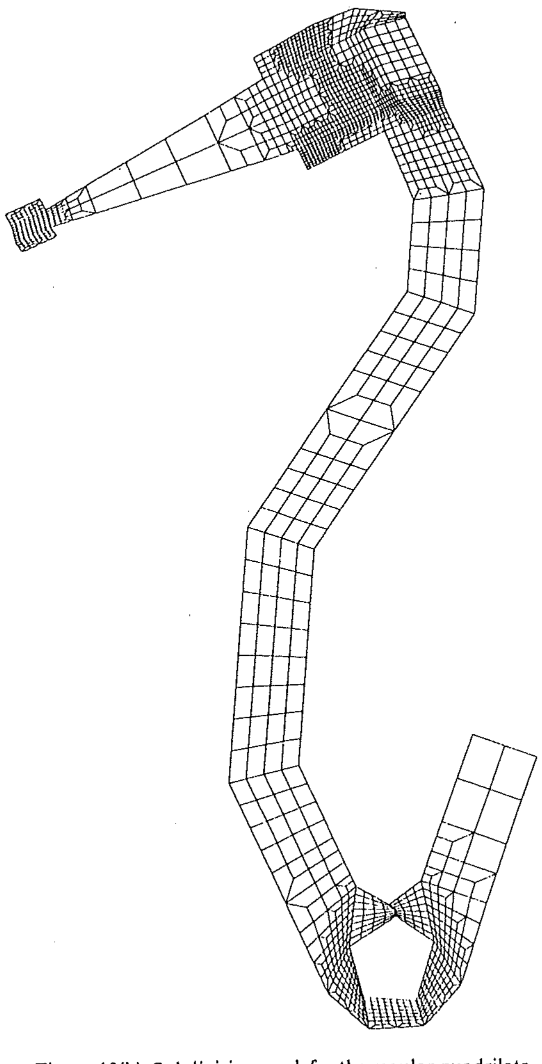
Figure 13(b). Subdivision mesh for the regular quadrilateral network and the associated subdivision level assignment shown in Figure 13(a)

technique is used, our technique can actually be used to generate quadrilateral meshes on any surfaces. Note that the algorithm can easily be modified to generate meshes of triangles also; simply divide each quadrilateral by adding a diagonal.

This algorithm has been implemented in Pascal on a Sequent Balance 21000 computer with 26 processors. Quadtree is used as a main data structure to support the parallel mesh generation algorithm. The given regular quadrilateral network is represented in an array structure easy to be accessed for the information related to vertices, labels and faces. For each face of the quadrilateral network, a quadtree, called a *mesh tree*, is created to store the information of the subdivision mesh for this face: the given face represents the root of the tree, each other node of the tree represents a subquadrilateral generated after a balanced or unbalanced subdivision. Several tests have been performed; some of them are shown in Figures 12 and 13. The comparison between the parallel version and the sequential version is shown in Figure 14. The execution time collected for the

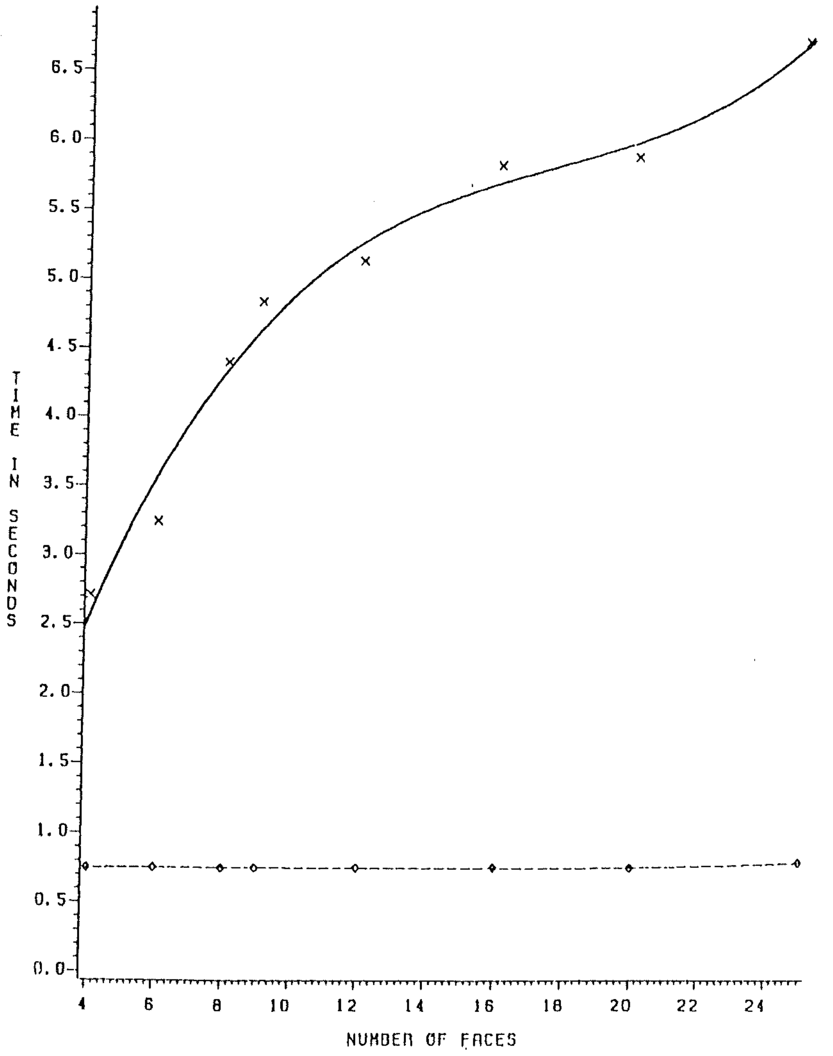

Figure 14. Performance comparison between the parallel version and the sequential version of algorithm PMG

parallel version and the sequential version is shown in squares and crosses, respectively. The curves are generated using the least squares approximation. For a regular quadrilateral network of $m \times n$ faces with randomly assigned subdivision levels (maximum subdivision level $= 3$), the sequential version is implemented using one processor only, the parallel version is implemented using $m \times n$ processors; one processor per face. According to the data we have collected, the parallel version appears well suited to the mesh generation process.

As we have pointed out in Section 4, the algorithm AEC was designed to approximate the solution to the optimal direct admissible extension only. It is still an open question whether an exact solution to the problem can be found efficiently. A more interesting problem is to construct an algorithm that minimizes the number of quadrilaterals generated in the subdivision mesh for a given subdivision level assignment. Several other interesting problems remain open. In particular, it is interesting whether the proposed algorithm can be extended to 3-D problems.

ACKNOWLEDGEMENTS

We thank Jinlong Chen and Robert Feng for implementing a current version of our algorithm. We appreciate the financial support of the Center for Robotics and Manufacturing Systems, University of Kentucky and the Industrial Technology & Research Institute, Taiwan, R.O.C., during the preparation of this work.

REFERENCES

1. P. L. Baehmann, S. L. Wittchen, M. A. Shephard, K. R. Grice and M. A. Yerry, 'Robust, geometrically based, automatic two-dimensional mesh generation', *Int. j. numer. methods eng.*, **24**, 1043–1078 (1987).
2. E. E. Barnhill, T. Birkhoff and W. J. Gordon, 'Smooth interpolation in triangles', *J. Approx. Theory*, **8**, 114–128 (1973).
3. A. Bykat, 'Automatic generation of triangular grid: I—Subdivision of general polygon into convex subregions. II—Triangulation of convex polygons', *Int. j. numer. methods eng.*, **10**, 1329–1342 (1976).
4. J. C. Cavendish, 'Automatic triangulation of arbitrary planar domains for the finite element method', *Int. j. numer. methods eng.*, **8**, 679–696 (1974).
5. J. C. Cavendish, D. A. Field and W. H. Frey, 'An approach to automatic three-dimensional finite element mesh generation', *Int. j. numer. methods eng.*, **2**, 329–347 (1985).
6. F. Cheng, J. Jaromczyk, J.-R. Lin, S.-S. Chang and J.-Y. Lu, 'Automatic mesh generation based on vertex label assignment', *TR #105-88*, Department of Computer Science, University of Kentucky, 1988.
7. W. A. Cook, 'Body oriented (natural) co-ordinates for generating three-dimensional meshes', *Int. j. numer. methods eng.*, **8**, 27–43 (1974).
8. C. O. Frederick, Y. C. Wong and F. W. Edge, 'Two-dimensional automatic mesh generation for structural analysis', *Int. j. numer. methods eng.*, **2**, 133–144 (1970).
9. W. J. Gordon and C. A. Hall, 'Transfinite element methods: blending interpolation over arbitrary curved domains', *Numer. Math.* **21**, 109–129 (1973).
10. W. J. Gordon and C. A. Hall, 'Construction of curvilinear coordinate systems and applications to mesh generation', *Int. j. numer. methods eng.*, **7**, 461–477 (1973).
11. R. H. Haber, M. S. Shephard, J. F. Abel, R. H. Gallagher and D. P. Greenberg, 'A general two-dimensional finite element preprocessor utilizing discrete transfinite mappings', *Int. j. numer. methods eng.*, **17**, 1015–1044 (1981).
12. H. A. Kamel and H. K. Eisenstein, 'Automatic mesh generation in two and three-dimensional inter-connected domains', in *Symp. on High Speed Computing of Elastic Structure (IUTAM)*, Liege, Belgium, 1970.
13. Y. T. Lee, Automatic finite element mesh generation based on constructive solid geometry', *Ph.D. Dissertation*, Department of Mechanical Engineering, University of Leeds, Leeds, England, 1983.
14. Y. T. Lee, A. De Pennington and N. K. Shaw, 'Automatic finite-element mesh generation from geometric models—A point-based approach', *ACM Trans. Graphics*, 3, 287–311 (1984).
15. M.-C. Rivara, 'A grid generator bsed on 4-triangles conforming mesh-refinement algorithms', *Int. j. numer. methods eng.*, **24**, 1343–1354 (1987).
16. E. A. Sadek, 'A scheme for the automatic generation of triangular finite elements', *Int. j. numer. methods eng.*, **15**, 1813–1822 (1980).
17. M. S. Shephard and M. A. Yerry, 'Finite element mesh generation for use with solid modelling and adaptive analysis', *Proc. General Motors Symposium on Solid Modelling by Computers: From Theory to Applications*, General Motors Research Laboratories, Warren, Mich., 1983.

18. J. Suhara and J. Fukuda, *Automatic Mesh Generation for Finite Element Analysis*, UAH Press, Huntsville, Ala., 1972, pp. 607–624.
19. R. B. Tilove, 'Line/polygon classification: A study of the complexity of geometric', *IEEE Comp. Graph. Appl.* **1**, 75–84 (1981).
20. F. T. Tracy, 'Graphics pre- and post-processor for two-dimensional finite element programs', *Computer Graphics* (*Proc. Siggraph '77*), **11**, 8–12, (1977).
21. B. Wordenweber, 'Automatic mesh generation of 2 and 3 dimensional curvilinear manifolds', *Ph.D. Dissertation*, Computer Laboratory, University of Cambridge, Cambridge, England, 1981.
22. M. A. Yerry and M. S. Shephard, 'A modified quadtree approach to finite element mesh generation', *IEEE Comp. Graph. Appl.*, **3**, 39–46 (1983).
23. O. C. Zienkiewicz and D. V. Phillips, 'An automatic mesh generation scheme for plane and curved surfaces by isoparametric co-ordinates', *Int. j. numer. methods eng.*, **3**, 519–528 (1971).