

Generating Subnets for Polyhedra

Kailash Aurangabadkar

September 2004

Department of Computer Science

University of Minnesota Duluth

320, Heller Hall

1114, Kirby Drive

Duluth, MN 55812

Phone: (218) 726- 7607

Email: [cs@d.umn.edu](mailto:cs@d.umn.edu)

Web: [www.d.umn.edu/cs](http://www.d.umn.edu/cs)

# Generating Subnets for Polyhedra

A Thesis  
Submitted to the faculty of the Graduate School  
Of University of Minnesota  
By

Kailash Aurangabadkar

In partial fulfillment of the requirements  
For the degree of  
Master of Science

September 2004

University of Minnesota

This is to certify that I have examined this copy of a Master's thesis  
By

Kailash Aurangabadkar

And have found that it is complete and satisfactory in all respects,  
And that any and all revisions required by the final  
Examining committee have been made

Dr. Douglas Dunham

-----  
Name of Faculty Advisor

-----  
Signature of faculty advisor

-----  
Date

GRADUATE SCHOOL

## **Acknowledgements**

I would like to take this opportunity to thank Dr. Dunham, my faculty advisor, for the invaluable help and support he has provided me throughout this thesis period. He is the one who was always there to help me through challenging situations. I am grateful for his guidance.

I would also like to thank my committee members, Dr. Shute and Dr. Hasan, for their time and suggestions.

Thanks to the staff and faculty of Computer Science Department at UMD for their support.

## **Abstract**

A "Net" of a polyhedron is a single connected polygon formed from the faces of the polyhedron, which can be folded up to form the polyhedron. It is very difficult to find if a polyhedron has a "net". Researchers have given examples of "no-net" convex polyhedra. A "subnet" of a polyhedron is a connected polygon formed from the faces of the polyhedron. Technically, if there is just one subnet that can be folded up to form a polyhedron then it is defined to be a "net" of that polyhedron. In the past, programs have been developed to find subnets of specific polyhedra.

The algorithm designed here takes any arbitrary polyhedron as input and generates subnets of this polyhedron so that these subnets can be folded up and glued together to form the complete polyhedron. The algorithm has been implemented as a computer program. One of Dr. Dunham's research interests lies in visualizing these polyhedra and drawing repetitive patterns on the nets of these polyhedra. The current program can generate subnets of any given non self-intersecting polyhedra, which will be used by Dr. Dunham in designing repetitive patterns on these three dimensional polyhedra.

# Table of Contents

Introduction.....	1
Motivation.....	4
Basic terms and difficulty of Problem.....	6
Polygon.....	6
Polyhedron.....	6
Polyhedron Net.....	8
Polyhedron Subnets.....	9
A Complete Set of Subnets.....	10
Algorithm.....	11
Algorithm Details.....	12
Implementation Details.....	17
Polyhedron Data Structure.....	17
SPHIGS.....	20
PostScript Output.....	20
Results and Conclusion.....	21
Tetrahedron.....	21
Cube.....	22
Octahedron.....	24
Pyramid.....	26
Icosahedron.....	27
Future Work.....	29
References.....	30

## Table of Figures

Fig.1. Nets of Cube and Tetrahedron.....	8
Fig.2. Different Nets of Cube.....	9
Fig.3. Plane diagram resembling net of a cube.....	9
Fig.4. Tetrahedron Subnet.....	21
Fig.5(a). Cube Subnet 1.....	22
Fig.5(b). Cube Subnet 2.....	23
Fig.6(a). Octahedron Subnet 1.....	24
Fig.6(b). Octahedron Subnet 2.....	25
Fig.7. Pyramid Subnet.....	26
Fig.8(a). Icosahedron Subnet 1.....	27
Fig.8(b). Icosahedron Subnet 2.....	28

## Introduction

A "Net" of a polyhedron is a single connected polygon formed from the faces of the polyhedron, which can be folded up to form the polyhedron. An example of a polyhedron "net" is the cross shape made up of six squares that can be folded up to make a cube. In theory polygons can be cut out and folded up to form closed surfaces of genus greater than 1. For any surface of genus  $g$  greater than 1, such a hyperbolic polygon can be found, which is just to say that the hyperbolic plane is the universal covering surface for the original genus  $g$  surface.

It is very difficult to find if a polyhedron has a "net". Even for convex polyhedra there are examples in [1] of a 13 faced no-net polyhedron. Various no-net polyhedra are presented in [2], including one with 24 faces. Although it was conjectured by G.C. Shephard [3] in 1975 that there is a net that can be folded up to form an entire polyhedron, the above examples show that there are examples for which finding a net is not possible.

A "subnet" of a polyhedron is a connected polygon formed from the faces of the polyhedron. A complete set of subnets can be folded up and glued together to form the original polyhedron. The goal of this research is to focus on developing an algorithm for generating these subnets for an arbitrary polyhedron. Subnets come with

labeled tabs that identify “gluing connections” that are to be made to generate the complete polyhedron. Technically, if there is just one subnet that can be folded up to form a polyhedron then it is actually a “net” of that polyhedron. As seen above, the cross shape made up of squares that can be folded up to form a cube is a net. At the other extreme, six individual squares would also form a complete set of subnets for the cube.

According to George Hart, there are no programs to generate subnets given an arbitrary polyhedron [4]. In the past, programs have been developed to find subnets of specific polyhedra. One of the most sophisticated and recent program, Stella [5], uses advanced functions to generate nets and has many other types of navigation within a given polyhedron. This program provides user with a long list of built-in models of polyhedra to choose from. This program uses advanced functions such as stellation, faceting, augmentation, excavation and other possibilities to generate its nets [5].

Some other software designed basically for generating nets of polyhedra and using them for educational purposes are Hypergami, Poly, POLYDRON, Hedron etc. Hypergami is a software application that generates polyhedra nets of various built in polyhedra and their combinations and decorates these nets [6]. Poly is a similar program that can generate nets of 147 different polyhedra.

In his previous experiments Dr. Dunham has used subnets of various typical polyhedron to design repeating patterns on the subnets. He has work related to tessellations in the three classical geometries, the Euclidean plane, the Sphere and the Hyperbolic plane [7]. For this purpose he has used subnets of various regular polyhedra. As is seen in the software products described earlier, the polyhedra for which the subnets are created are pre-selected and can be only one of these built-in polyhedra.

The algorithm to be designed here is to take any arbitrary polyhedron as input and to generate a set of subnets of this polyhedron so that these subnets can be folded up and glued together to form the complete polyhedron. These subnets are built around "seed" faces that are also taken as user input. This algorithm is implemented in C as a graphics program. This program shows the polyhedron in three dimensions using the functions of "SPHIGS", which is a library of functions in C for displaying three dimensional objects. The two dimensional subnets are produced as outputs in the PostScript format.

This algorithm could be further optimized by using various other geometric properties of polyhedra to speed up the process of generating the subnets, and also to generate as few subnets as possible. An interactive ability could be added to the program to

make it user friendly, so that user can place subnets at various positions and try different combinations.

The following section explains the motivation behind the project. The next section defines some of the basic terms in polyhedron geometry and discusses the difficulty of finding a net for a polyhedron. Then the algorithm to generate these subnets of polyhedron is presented in detail. This is followed by the implementation details and some runs of this program producing subnets of some sample polyhedra followed by results and analysis. Finally, the conclusion and future work of this work is presented.

## **Motivation**

Polyhedra are inspiring pieces of work and always a center of attraction for the human society. Some ancient polyhedra include the Egyptian pyramids, the five regular polyhedra called Platonic solids discovered by the ancient Greeks, and the Archimedean polyhedra. Polyhedra are aesthetic and are remarkably beautiful as suggested by the previous examples.

It is fascinating to know that these polyhedra can be built by folding paper. Albrecht Durer is credited for the invention of paper "nets" for polyhedra. This process makes it easier to build various polyhedra and study them. It is especially useful in schools where

teachers can build polyhedra to show to their students. This way the students can actually see an abstract three dimensional figure and see how they can be created by folding paper. The positive effect of using polyhedra and their construction process in the understanding of basic geometry by students has been experimentally studied by Piaget and Inhelder in 1948 [6]. It has been studied in more detail in [6].

One of Dr. Dunham's research interests lies in visualizing these polyhedra and drawing repetitive patterns on the nets of these polyhedra. He has previously worked with repetitive patterns on various surfaces including nets of some simple polyhedra. It would be a nice idea to be able to generalize this process. A program which could generate the subnets for any input polyhedron will be useful in further research. Although there are quite a few computer programs available, they are limited by the fact that they only work on certain built-in polyhedra and cannot extend to any polyhedra in general. This is basically due to the fact that most programs strive to find a single net of the polyhedron. It is still an open problem whether there is a single net for any given polyhedron.

The current program can generate subnets for any given non self-intersecting polyhedra. These subnets generated by the program will be very useful for Dr. Dunham in his investigation of the theory of repeating hyperbolic patterns. Dr. Dunham has an active research

area in polygon tessellations of the hyperbolic plane. This program will provide polyhedron models of surfaces corresponding to those polygon tessellations. The subnets of polyhedra produced by this program can then be filled in with repeated patterns to aid Dr. Dunham in his on-going research.

## **Basic Terms and difficulty of the Problem**

The basic terms of geometry required are as follows:

### **Polygon:**

A *polygon* is usually defined as a connected two-dimensional figure bounded by line segments, called *sides*, with exactly two sides meeting at each vertex. There have to be three or more sides to make a closed figure. A polygon is *convex* if it is not self-intersecting and all interior angles are less than 180 degrees. Otherwise it is *non-convex*. A simple example of a non-convex polygon is the pentagram. Some common examples of the convex polygons are triangles, rectangles, and regular polygons. A regular polygon is convex and has congruent sides and interior angles.

### **Polyhedron:**

Most simply, a *polyhedron* is a closed three dimensional figure

having polygons as faces. Some of the popular and more detailed definitions of polyhedron are:

A three dimensional object bounded by polygons, with each edge shared by exactly two polygons. Various authors differ on the fine points of the definition, e.g., whether it is a solid or just the surface, whether it can be infinite, and whether it can have two different vertices that happen to be at the same location [10].

A solid figure bounded by plane polygonal faces. The point at which three or more faces intersect on a polyhedron is called a *vertex*, and a line along which two faces intersect is called an *edge*. In a *regular polyhedron*, all the faces are congruent regular polygons. There are only five regular polyhedra: the tetrahedron, the cube, the octahedron, the icosahedron, and the dodecahedron [11].

Mathworld goes further in defining polyhedron differently in geometry and algebraic geometry. In geometry, a polyhedron is simply a three-dimensional solid which consists of a collection of polygons, usually joined at their edges. In algebraic geometry, a polyhedron is a space that can be built from such "building blocks" as line segments, triangles, tetrahedra, and their higher dimensional analogs by "gluing them together" along their faces [12].

## Polyhedron Net:

A *polyhedron net* is a flat pattern that can be cut out and folded up to make the original polyhedron.

Nets are basically plane diagrams in which the polyhedron edges of a polyhedron are shown, along with the gluing instructions, so that when they are folded they form the three dimensional polyhedron. Examples below show the nets of a cube and a regular tetrahedron.

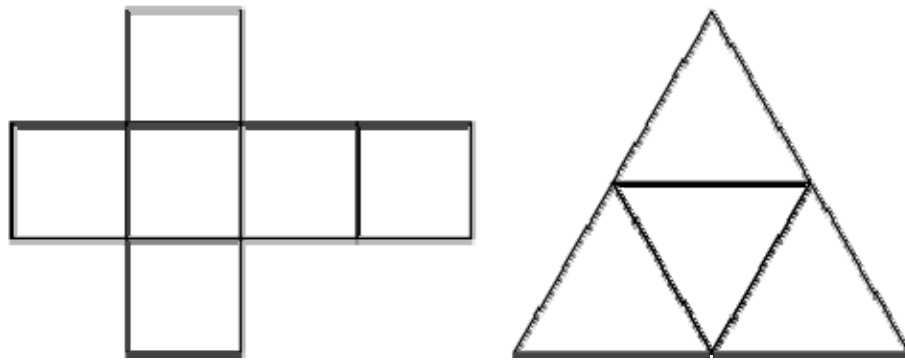


Fig.1. Nets of cube and tetrahedron

It is possible to get different polyhedron shapes from the single net by folding it in different ways. It is important for a net to show which edges are to be joined or glued as one net can be folded up into different polyhedron possibilities. Also nets are not unique to a polyhedron. For example, there are a total of 11 distinct nets for the cube [8], illustrated below. A tetrahedron has 2 nets, the octahedron has 11 nets, while the dodecahedron and icosahedron have 43380 possible nets [8].

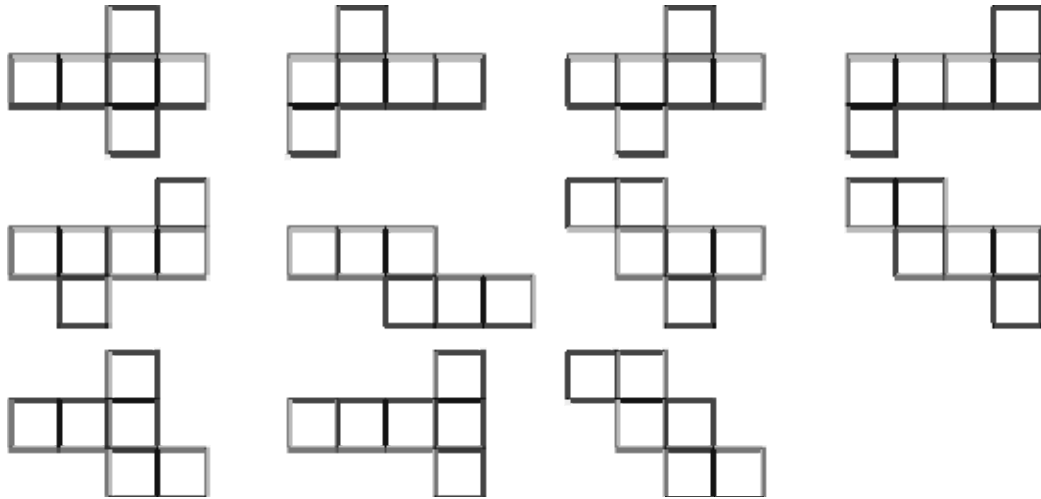


Fig.2. Different nets of cube

Not all plane diagrams are nets of a polyhedron. For example the following diagram cannot be folded up to form a polyhedron although it looks similar to the net of a cube. This is because the polygons when folded up do not form a closed figure.

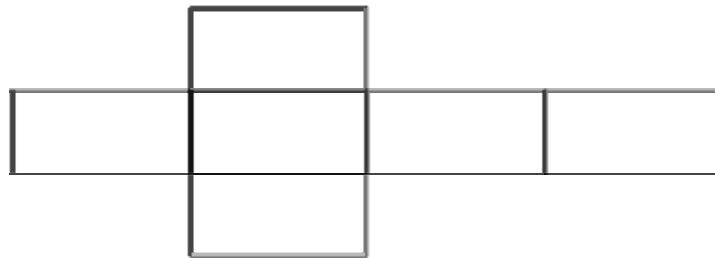


Fig.3. Plane diagram resembling net of a cube

### **Polyhedron Subnets:**

A *subnet* of a polyhedron is a planar polygon made of a set of connected faces of a three dimensional polyhedron. A net is a subnet that includes all the polyhedron's faces.

## **A Complete Set of Subnets:**

A *complete set of subnets* for a polyhedron is a collection of subnets that can be folded up and glued together to form the original polyhedron.

For any arbitrary convex polyhedra the problem of whether a net exists for all is still open. Shephard [3] conjectures that there exist a net for every convex polyhedra. But this is still unsettled, as many examples of convex polyhedra with no net can be found in [1] and [2]. Non-convex polyhedra are more complex than convex polyhedra and hence the problem is much more complicated. This difficulty can be avoided by forming subnets of a polyhedron. These subnets can then be individually folded up and then glued together to form the original polyhedron. In subnets, the edges need to be labeled as in the case of nets. This is for the convenience to determine which edges are to be glued together to get the original polyhedra. Douglas Zongker has sample subnets for the Platonic and Archimedean solids [9].

As mentioned earlier it is still an open problem whether there exists a net for any convex polyhedron. Almost all mathematicians tend to agree that there exist a net for all convex polyhedra. For non convex polyhedra the problem becomes harder, as these polyhedra are much more complicated than the convex polyhedra. The problem of finding nets for polyhedra has been tackled before, but most of

the solution rely on the most frequently used polyhedra and have a built-in list of the polyhedra for which the solution is presented.

The algorithm to be presented here tackles this issue differently. It accepts any polyhedron, including non-convex polyhedra as input and instead of making an attempt to find a net of the polyhedron, the algorithm tries to fit as many polygon faces into a subnet as possible so that no overlapping occurs. This is done by choosing seed faces and then adding adjacent faces to the seed faces into the subnet around the seed polygon. This results in the generation of subnets which can be glued together to form the original polyhedron.

## **Algorithm**

The general algorithm for creating subnets of a polyhedron is as follows:

1. Each face of the polyhedron is made into a subnet. This corresponds to the extreme case where each face is a subnet and they can be glued together to form the complete polygon. For this step every face of the polyhedron is translated to the origin of the two dimensional plane.
2. Put the translated faces together
  - a. Select seed polygons and form subnets around each seed

face

- b. Fill around each vertex until either
  1. No more polygons exist that have that vertex or
  2. An overlap occurs
3. Repeat step 2 for every seed specified.
4. If any unused polygons exist after the above process, then it is passed as a seed and is expected to form a single polygon subnet of the original polyhedron.

### **Algorithm Details**

The algorithm uses various combinations of geometric transformations on the faces of the polyhedron and converts them into two dimension polygons. Each face of the polyhedron is essentially a polygon in two dimensions. If we consider the x-y plane to exactly coincide with the face of polyhedra then the face is nothing but a polygon with the z coordinates equal to zero. The algorithm then stitches as many of these individual polygon-faces together as possible to form one of the subnets that together make up the net of the polyhedron. For the stitching of the polygons, some of the faces of the original polyhedron are considered as seed faces and the corresponding polygons as seed polygons in two dimensions. The polygons corresponding to these seed faces of the polyhedron are

used as seed polygons. All the polygons corresponding to the faces adjacent to the seed face in the polyhedron are then transformed in two dimensions so that they are adjacent to the seed polygon. The seed faces are currently taken as user input. But this could be automated to find the best seed faces.

The algorithm starts off by transforming every face of the polyhedron into a two dimensional polygon.

For every face of the polyhedron

1. Translate first vertex to origin
2. Find the distance  $A$  of second vertex from first vertex in three dimensions. Plot the second vertex at point  $(A,0)$  in the two dimensional plane. This makes the edge between first and second vertex lie along x-axis in the two dimensional plane.
3. Keep Track of the Current Angle with respect to positive x axis.
4. For all other vertices  $i$ 
  - a.  $A$ =distance between vertices  $i-2$  and  $i-1$
  - b.  $B$ =distance between vertices  $i-1$  and  $i$
  - c. Find Theta, the angle between edges between  $(i-2,i-1)$  and  $(i-1,i)$

- d. Update Current Angle by adding Theta to it
- $$\text{vertex}_i = (|B|\cos(\text{current angle}),$$
- $$|B|\sin(\text{current angle})) + \text{vertex}_{i-1}$$

After this step all the faces have been transformed to two dimensional polygons with start vertex corresponding to the origin of the x-y plane, and first edge lies along the x-axis.

The adjacent faces of the seed faces are then stitched to corresponding seed faces to form the subnets of the polyhedron. There will be one subnet for each seed face.

for each edge (i , j) of seed

1. Search all unused polygons for the edge (j,i).
2. Let P be the polygon with edge (j,i).
3. Translate the new found polygon so that vertex i of the new polygon coincides with vertex i of the seed polygon.
4. Find the angle theta made by edge (j,i) in the new polygon with the positive x axis.
5. Rotate the new polygon by angle theta so that the edge (j,i) is the topmost horizontal edge of the polygon.
6. Find the angle alpha made by edge (i,j) in the seed polygon with the positive x axis.
7. Rotate the new polygon by angle alpha. The polygon is

already rotated by angle  $\theta$ . So the combined rotation is by angle  $\alpha + \theta$ .

8. This combined rotation makes the edge  $(j,i)$  in the new polygon coincide with the edge  $(i,j)$  in the seed polygon.
9. Mark the polygon as used and continue with other edges of the seed polygon.

This algorithm excludes self-intersecting polyhedra. As the faces do not intersect each other, only two faces of the polyhedra will have an edge in common. After this point, all the faces adjacent to seed faces have been stitched to form the subnets. There still can be some faces not processed yet.

At this stage all the faces that are processed are the faces that may share a vertex with the seed face and an edge with the faces adjacent to the seed face.

For each vertex of the seed

Calculate the angle at that vertex made by the faces so far included in the net

If the angle is less than 360

1. Find the face that has the vertex sharing with seed face and edge  $(j,i)$  sharing with edge  $(i,j)$  of the

adjacent face to the seed and is not included in the subnet.

2. Let the adjacent face be a seed face
3. Add the angle made by this face at the vertex
4. If it is not more than 360 then find the angle  $\theta$  made by edge  $(j,i)$  in the new polygon with the positive x axis.
5. Rotate the new polygon by angle  $\theta$  so that the edge  $(j,i)$  is the topmost horizontal edge of the polygon.
6. Find the angle  $\alpha$  made by edge  $(i,j)$  in the seed polygon with the positive x axis.
7. Rotate the new polygon by angle  $\alpha$ . The polygon is already rotated by angle  $\theta$ .
8. So the combined rotation is by angle  $\alpha + \theta$ .
9. This combined rotation makes the edge  $(j,i)$  in the new polygon coincide with the edge  $(i,j)$  in the seed polygon.
10. Mark the polygon as used and continue with other edges of the seed polygon.

This part of the algorithm tries to put any many polygons

into one net as possible without overlapping. The overlapping is checked by checking for the angle made by all the polygons at a vertex. If the angle is less than 360, then it is not overlapping.

## **Implementation Details**

The above algorithm is implemented in C on UNIX. The program takes polyhedron vertices and faces as input from a text file and implements the above algorithm to generate the subnets for the polyhedron. The original polyhedron is shown in three dimensions using SPHIGS and the two dimensional nets are produced as PostScript files. These can be viewed using GHOSTVIEW PostScript viewing software.

### **Polyhedron data structure**

The polyhedron data structure consists of the *vertex list* which is the three dimensional vertices of the original polyhedron. These are represented in (x, y, z) format by the point data structure. This point data structure is defined in the SPHIGS library.

The faces are represented as a list of indices into the vertex list. The faces of the polyhedron are represented as they are represented in real life geometry. One face is represented by the points it has and

the orientation for specifying the points is anti-clockwise. Each face is separated from others by a boundary value which is -1 in SPHIGS. This format is used because it conforms with SPHIGS as it is used to display the polyhedron in three dimensions. Also this format resembles the real life representation of polyhedra and polygons.

The *face list* is a list of faces similar to the representation of the faces. But it is a two dimensional matrix that is used to store the vertex index of every face separately. Two consecutive entries of vertices for a face in this list represents the edge in the face. This list represents all the edges as the first vertex of the face is again repeated at the end to account for the last edge. This list is generated as it is convenient to deal with this list to find faces that have a common edge or faces that share a vertex and common edges with neighbors.

The vertices in two dimensions are stored in a list called *ver2D*. This list has values for all the vertices in three dimensions of the polygon converted into two dimensions. It is to be noted that one vertex in three dimensions can have different mappings in the two dimensions. That is to say that a vertex of the polyhedron in three dimensions can have more than one vertex corresponding to it in two dimensions. So the two dimensional vertex list corresponds to the three dimensional list of faces stored in the data structure. For

each vertex of every face of the polyhedron there is a two dimensional vertex mapping to that vertex.

A list is also maintained to keep track of the angles made by all the polygons surrounding a seed vertex. This list is a two dimensional list which stores the angle for each seed's vertex made by the polygons surrounding that vertex. This angle is used to determine if new faces can be added to that vertex without causing overlapping of the faces. If the angle is less than 360 and the new face doesn't make it go beyond 360, then the new face is added at that vertex in the subnet. Otherwise it is still marked unused and is considered later on.

All the faces that are added around a vertex of a seed face into the subnet are kept track of in a list for all faces adjacent to each seed's vertex. This list is used while adding new faces at vertices of the seed. Also a corresponding list of the edges these faces share with the seed face is maintained. This list together with the list of adjacent faces helps find new faces from the face list that have the shared vertex and an edge corresponding to an edge in the adjacent face.

A list of all faces that form the subnet around a seed face is maintained. This list of faces along with the seed face is then output to a PostScript page which forms a subnet of the polyhedra.

Apart from these lists in the data structure for polyhedra, a separate list of the faces used so far is maintained. This list helps in

identifying new faces and separating them from faces that are already included in the two dimensional subnet of the polyhedron. This list helps separate the already used faces from that that are still to be processed.

### **SPHIGS:**

SPHIGS is a library of functions in C. These functions can be used to display an object in three dimensions without worrying about implementation of hidden surface removal or any other details. In SPHIGS the polyhedron is input in the form of vertices and faces. SPHIGS provides facilities to change the view plane, view angle and the view direction, so that an object can be viewed in more than one way. This adds functionality to the application in the way that the three dimensional polyhedra can be viewed from different points to get the feel of the polyhedra. Hence SPHIGS is used to display the original polyhedra.

### **PostScript output:**

The two dimensional nets created by the program are output in the form of a PostScript file. This PostScript file can be opened to view with the help of GHOSTVIEW, software used to view PostScript files. The routines to create the output PostScript files have already been

implement by Dr. Dunham. The PostScript files offer a lot of flexibility in changing the position of the diagram in the file by making minimal changes to the file. Dr. Dunham has been using these PostScript files as output in his previous research and these files are compatible to his format and will aid in the future work.

## Results and Conclusion

### Tetrahedron:

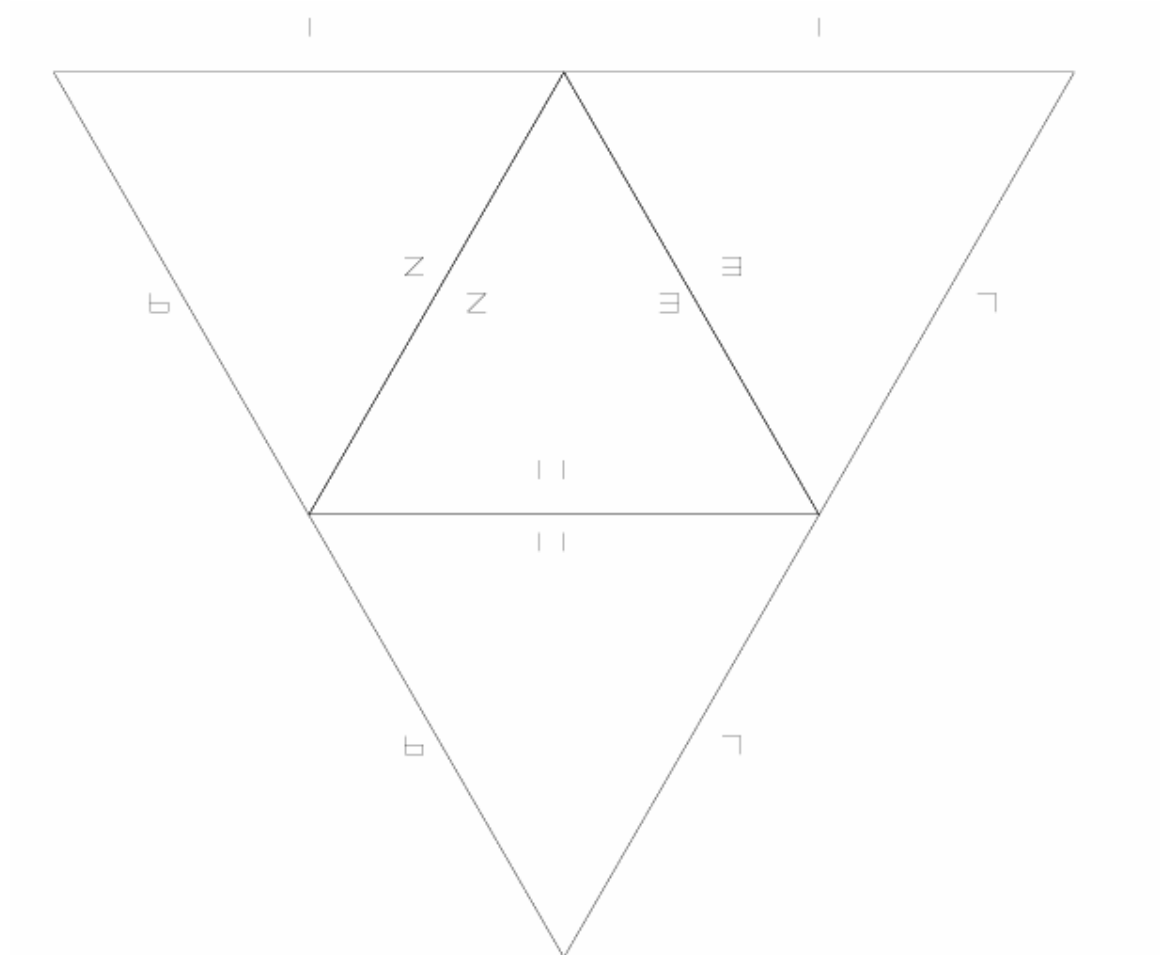


Fig.4. Tetrahedron Subnet.

**Cube:**

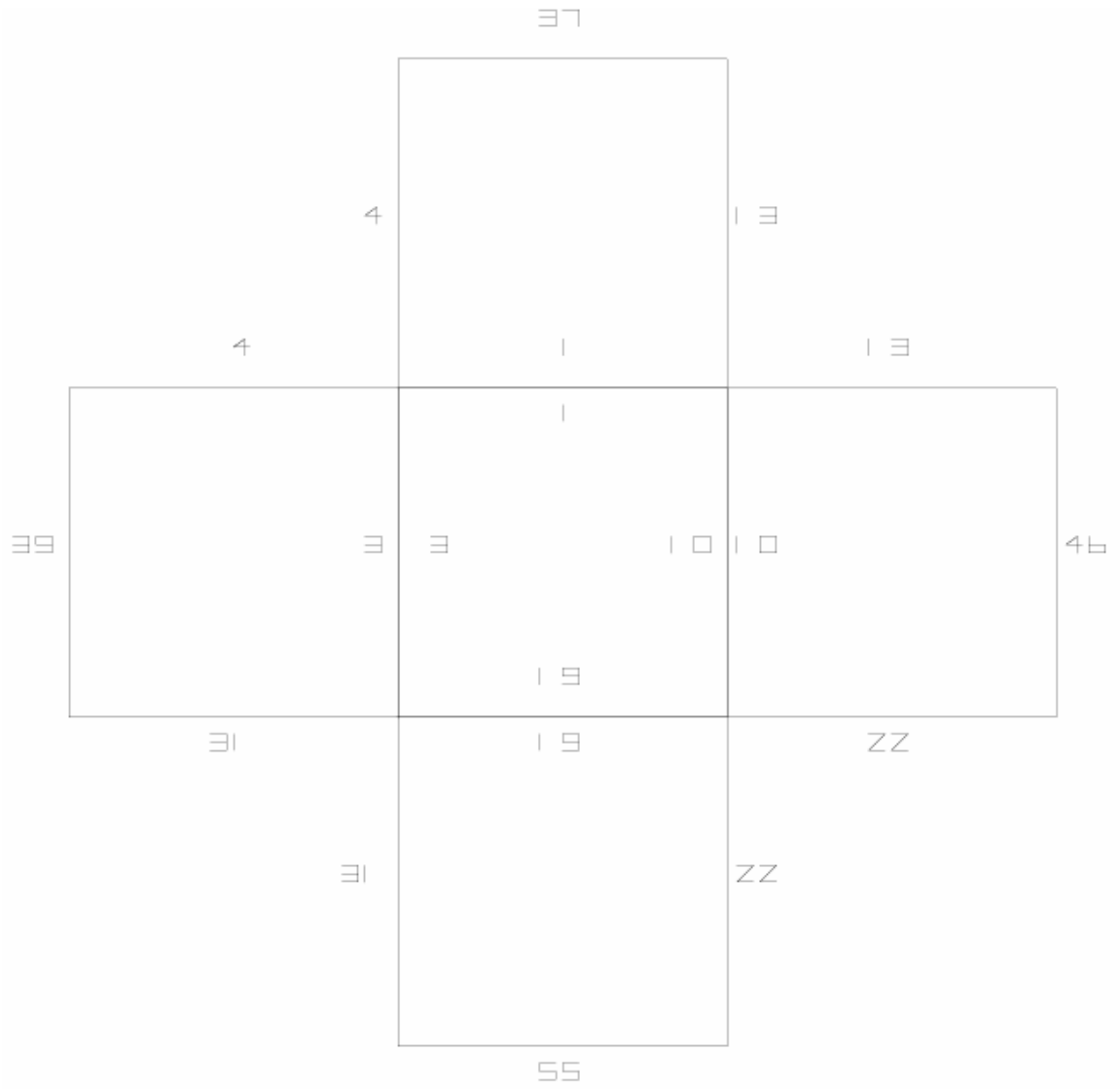


Fig.5(a). Cube subnet 1.

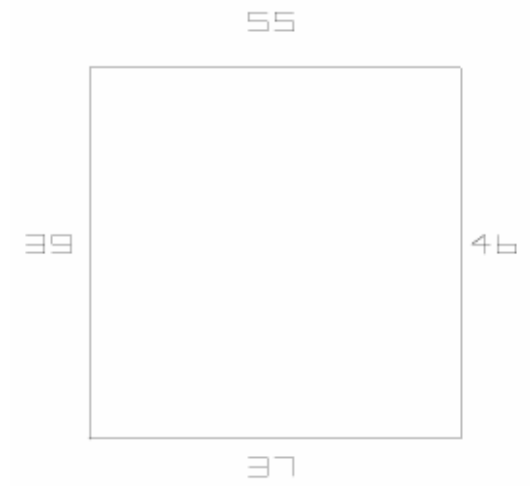


Fig.5(b). Cube Subnet 2.

**Octahedron:**

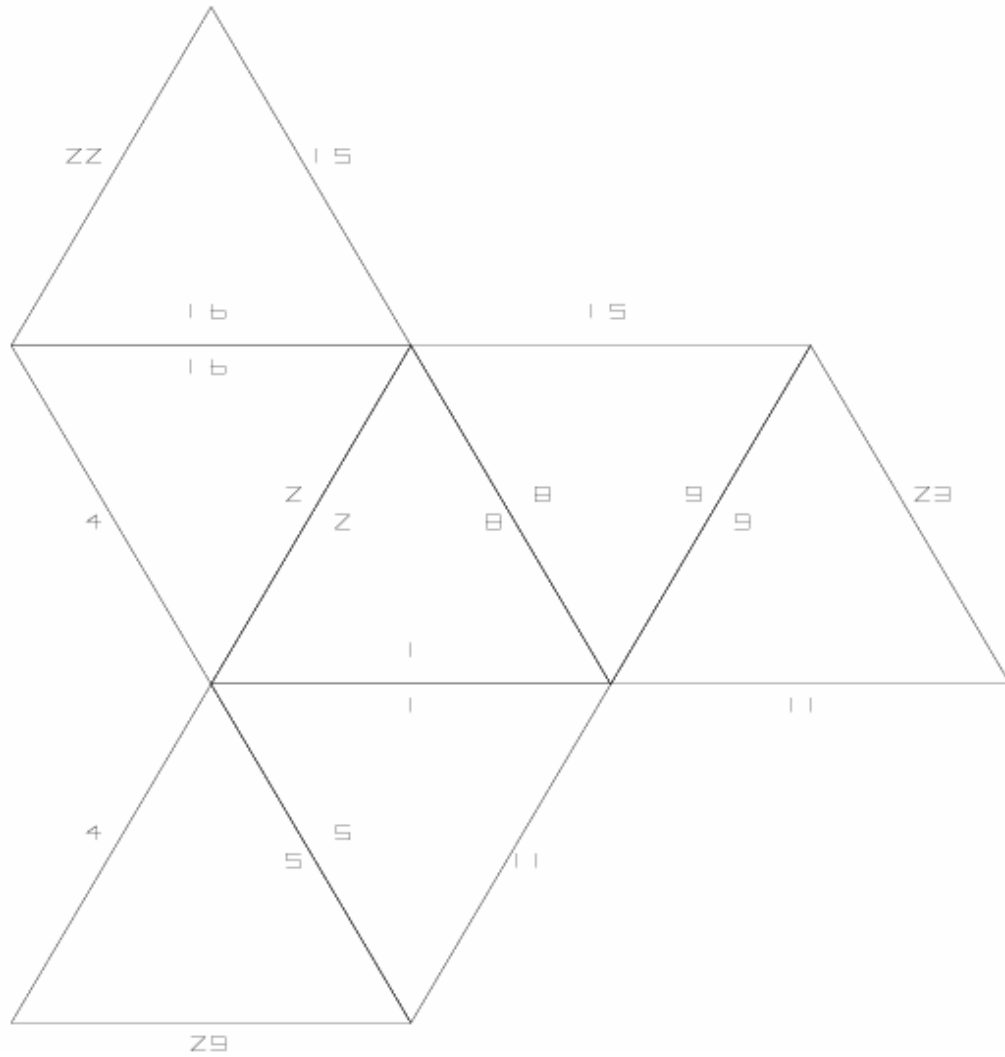


Fig.6(a). Octahedron subnet 1.

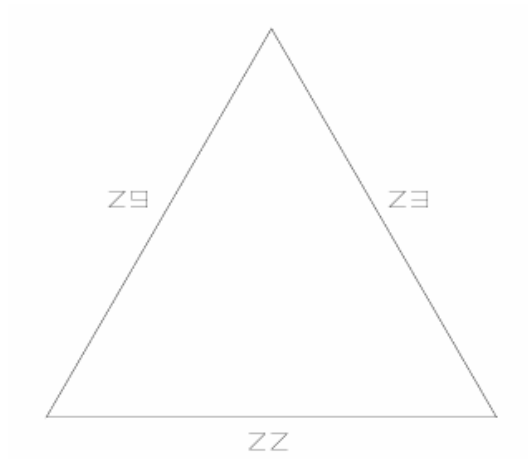


Fig.6(b). Octahedron subnet 2.

**Pyramid:**

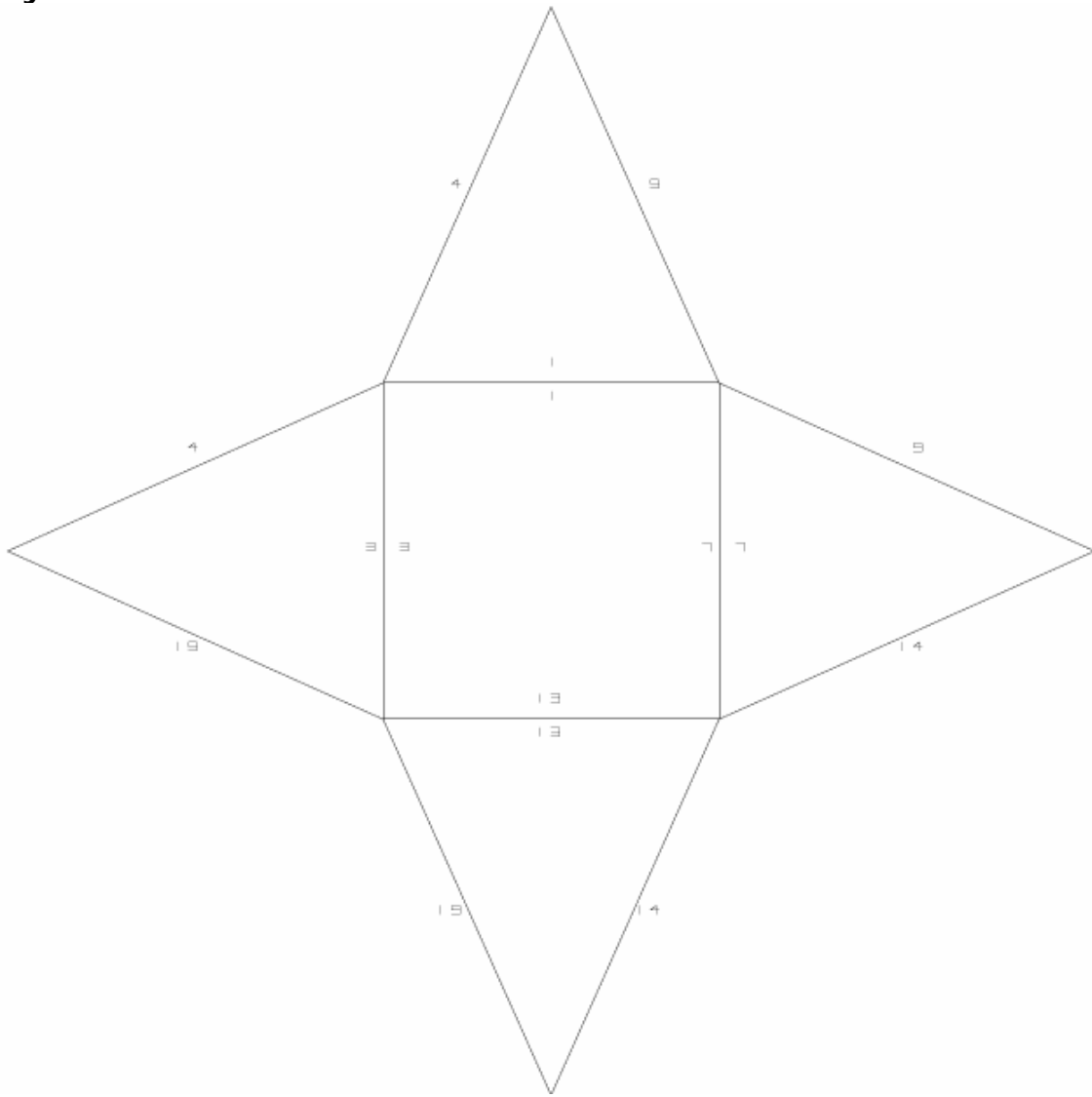


Fig.7. Pyramid subnet.

**Icosahedron:**

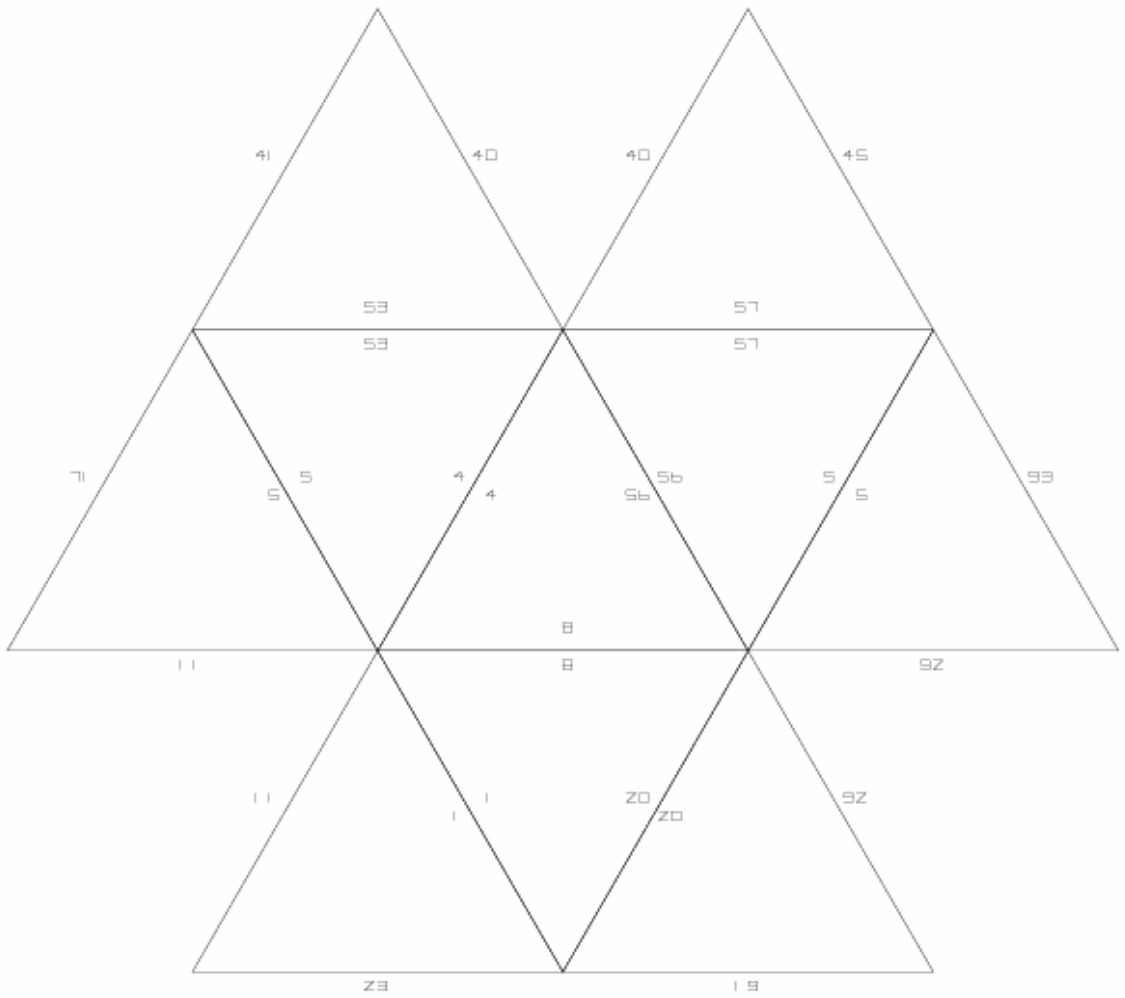


Fig.8(a). Icosahedron subnet 1.

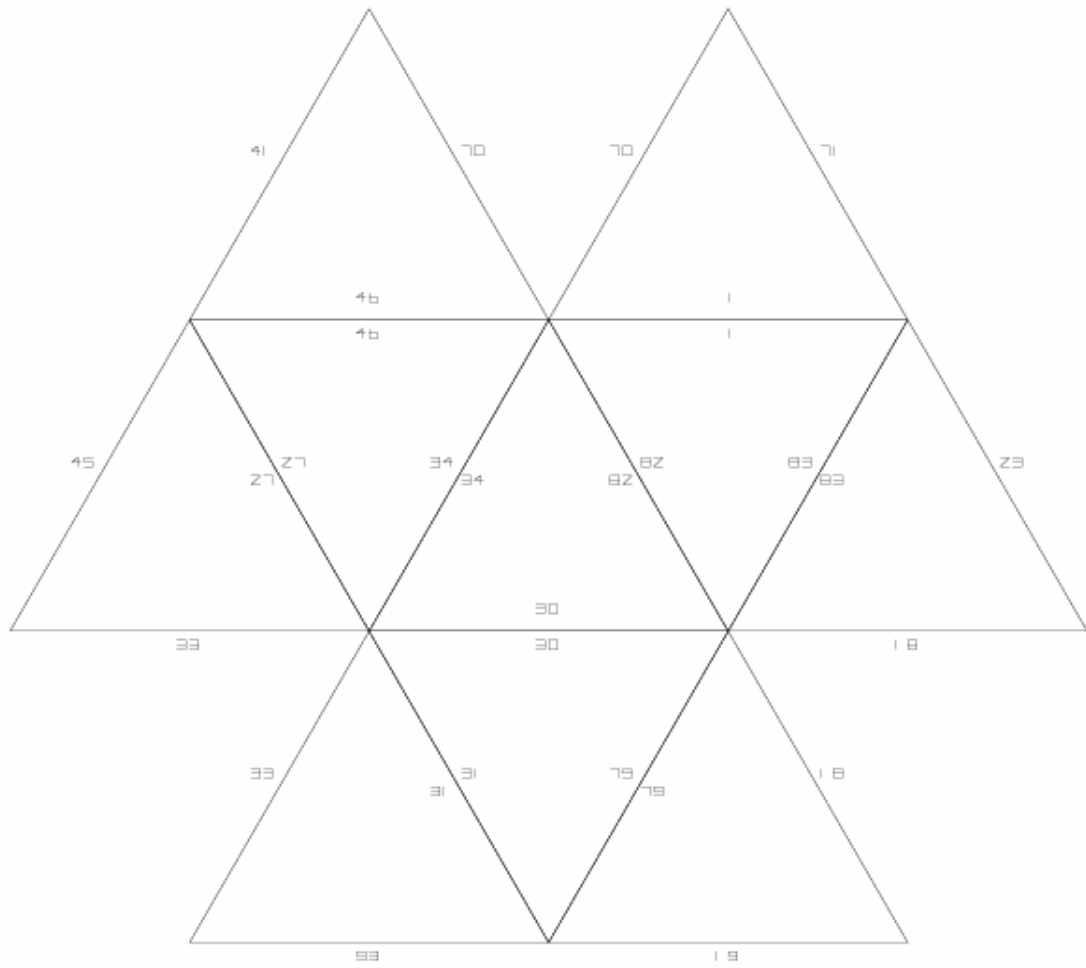


Fig.8(b). Icosahedron subnet 2.

The algorithm has been successfully implemented and tested on various polyhedra as shown above. All edges of the polygon are shown labeled here. But this is for checking purpose. The program actually displays edge labels only to those edges that are the outer edges of a subnet.

## Future Work

The software can be made more user friendly by adding interactive capabilities. Adding interactivity to the output of the program will help in moving subnets from one position to another to fit it exactly where it is wanted on the paper. An algorithm can be developed to find the best match for faces that can be used as seeds, which will give more encompassing and symmetrical subnets for the original polyhedron.

It would seem possible that the symmetry group of a polyhedron could be exploited to more efficiently generate subnets or to generate more symmetric subnets. However none of the currently available programs seem to make use of the symmetry group for generating subnets, so perhaps this is a more difficult problem than it seems.

The subnets of the polyhedra created will be worked on in future to design repeating patterns on them. This will aid Dr. Dunham in his research related to repeating patterns.

## References

- [1] Grünbaum, B. "NO-NET POLYHEDRA." *Geombinatorics* 11, 111 – 114, 2002
- [2] Bern, M., Demaine, E.D., Epstein, D., Kuo, E., Mantler, A. and Snoeyink, J. "Ununfoldable polyhedra with convex faces." *Computational Geometry: Theory and Applications* Vol. 24, Issue. 2, 51 – 62, 2003
- [3] Shephard, G.C. "Convex polytopes with convex nets." *Math. Proc. Camb. Phil. Soc.*, 78,389 – 403, 1975
- [4] Hart, G. Private e-mail message to Dr. Dunham, Nov. 18, 2002, (For more on George Hart see: <http://www.georgehart.com/>)
- [5] Webb, R. "Stella: Polyhedron Navigator", preprint, to appear in journal "Symmetry: Culture and Science". URL: <http://home.connexus.net.au/~robandfi/PolyNav/PolyNavigator.html>
- [6] Eisenberg, M., Nishioka, A. "Creating Polyhedral Models by Computer." *Journal of Computers of Mathematics and Science*

Teaching, 1997.

[7] Parsekar, A., "A Unified Data Representation and Visualization of Patterns Based on Regular Tessellations in the Three Classic Geometries" Computer Science Thesis, Univ. of MN. Duluth, 2002

[8] Buekenhout, F. and Parker, M. "The Number of Nets of the Regular Convex Polytopes in Dimension  $\leq 4$ ." *Disc. Math.* 186, 69 – 94, 1998.

[9] Zongker, D. "Polyhedra models", web page, URL:  
<http://www.cs.washington.edu/homes/dougz/polyhedra/>

[10] Hart, G. "Polyhedra Glossary", web page, URL:  
<http://www.georgehart.com/virtual-polyhedra/vp.html>

[11] UniStates, "RMT Glossary", web page, URL:  
<http://unistates.com/rmt/explained/glossary/rmtglossaryopq.html>

[12] Wolfram Research, Mathworld web page, URL:  
<http://mathworld.wolfram.com/Polyhedron.html>