UPC MAMME / Discrete Geometry 2018
Sheet 2
Diego Acedo and Xavier Gombau

Solution 1

a) The evenness condition is the following:

$$if\ i < j \notin S \Rightarrow \text{number of k between i and j is EVEN}$$

Now, with that in mind, we have to find all subsets of vertices that make up a facet of $C_4(8)$. We know that $C_4(8)$ is a simplicial polytope, so these facets will have dimension $d - 1 = 3$ and each facet will consist of 4 vertices. Numbering the vertices from 1 to 8, we need to consider all sets S with 4 elements, $S \subset [n] = \{1, 2, 3, 4, 5, 6, 7, 8\}$. A set S will form a facet if and only if the evenness condition is satisfied.

It is easy to come up with the 20 facets by hand, but we wrote a short MATLAB script to check our solution:

5678	4578	4567	3478	3467	3456	2378	2367	2356	2345
1678	1568	1458	1348	1278	1267	1256	1245	1238	1234.

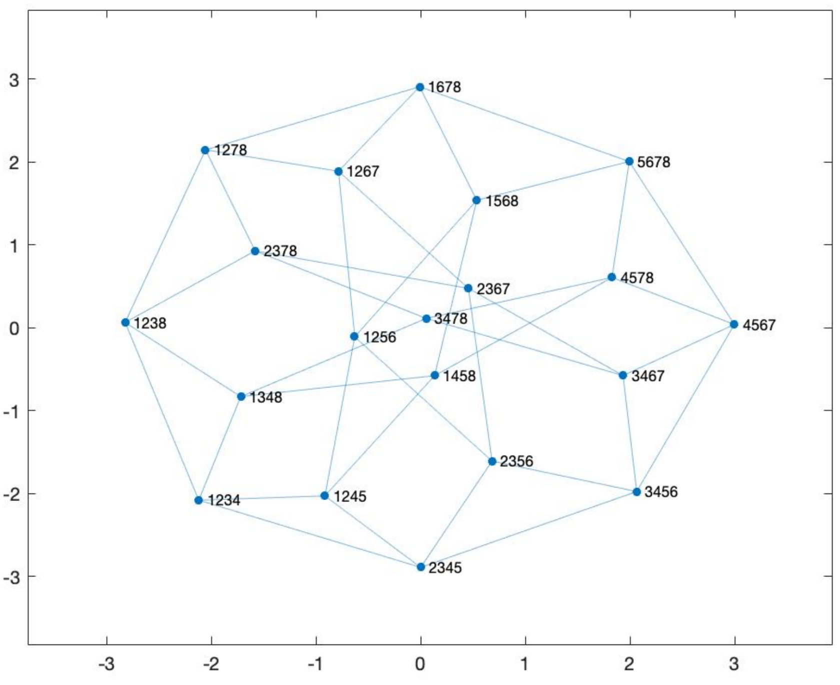
```matlab
% Find all facets of the 4-dimensional cyclic polytope
n = 8;
d = 4;
n_vec = 1:n;

comb = combnk(n_vec, d);
facets = [];
for i=1:numel(comb)/d
    pos_facet = true;
    S = comb(i,:);
    S_n = setxor(n_vec, S);
    for j=1:d-1
        dif = S_n(j)+1:S_n(j+1)-1;
        common = intersect(dif, S);
        if mod(numel(common), 2) == 1
            pos_facet = false;
        end
    end
    if pos_facet
        facets(i,:) = S;
    end
end
facets(~any(facets,2), :) = [];
```

b) To draw the dual graph of $C_4(8)$ it is clearer to program it, labelling the nodes:

```
% Draw the dual graph
pts = facets;
np = numel(pts)/d;

nodelabels = cellstr(arrayfun(@(x)sprintf('%d',x),pts));
edges = [];
for i=1:np
    for j=(i+1):np
        if numel(intersect(pts(i,:),pts(j,:))) == 3
            edges(end+1,:) = [i j];
        end
    end
end
g = graph(edges(:,1),edges(:,2),[],nodelabels);
plot(g)
```



We can see that this is a 4-regular graph with 20 vertices (each one corresponds to a facet of $C_4(8)$) and there exist an edge between two vertices if their corresponding facets share 3 vertices.

Solution 2

a) First of all, all points in the definition of P lie in the sphere centered at 0 with radius $\sqrt{2}$. The hyperplane tangent to the sphere at each of these points only meets P at this point, and so it defines a valid inequality for each of these points. Hence, all points are 0-dimensional faces, and so vertices.

   To count the number of vertices, fix $i \in \{1, 2, 3\}$. For each choice of $j > i$ we have 4 choices of signs, and we have $4 - i$ positions for j. Hence, the total number of vertices is $4 \cdot 3 + 4 \cdot 2 + 4 \cdot 1 = 24$.

b) To do this, we build a small program in Python which automatices the process explained in the statement of the problem. First of all, we define two functions, for the inner product and the reflection over a hyperplane. Then, we build a function that applies reflections until no new roots are found (that is, until the length of the list of roots does not grow anymore).

```python
def innerproduct(x, y):
    "Computes inner product of two 4-dimensional vectors."

    return x[0]*y[0]+x[1]*y[1]+x[2]*y[2]+x[3]*y[3]


def plane_reflex(a, v):
    "Calculates the reflection of the vector v on the orthogonal plane to vector a."

    scalar = 2 * innerproduct(a, v) / innerproduct(a, a)
    ref = [v[i] - scalar * a[i] for i in range(0, 4)]

    return ref


roots = [[1,-1,0,0],[0,1,-1,0],[0,0,1,0],[-1/2,-1/2,-1/2,-1/2]]

k = 0  # Starting condition
while k < len(roots):  # While new elements are added to the list.
    k = len(roots)
    for i in range(0, k):
        j_list = [h for h in range(0, k)]
        j_list.remove(i)  # reflecting r_i onto H_i returns the opposite vector, which induces the same hyperspace.
        for j in j_list:
            x = plane_reflex(roots[i], roots[j])
            if x not in roots:
                    roots.append(x)
len(roots)
```

Finally, we output the length of the list of roots, which is the number of resulting hyperplanes, and we get the answer 48.

c) It is enough to check that the reflection over each of the hyperplanes induces a bijection between the vertices. In this way, the boundary will stay the same, and the interior points will be mapped to interior points. For this, we build another Python function, which checks this condition: the program outputs “Hyperplane is ok” if the set of images through the reflection is the original set of vertices. We can see that all four hyperplanes behave correctly:

```
e=[] # Define canonical basis
e.append([1,0,0,0])
e.append([0,1,0,0])
e.append([0,0,1,0])
e.append([0,0,0,1])

vertices=[] # Define the set of vertices of P
for i in range(0,4):
    for j in range(i+1,4):
        vertex1 = [e[i][k] + e[j][k] for k in range(0,4)]
        vertex2 = [e[i][k] - e[j][k] for k in range(0,4)]
        vertex3 = [-e[i][k] + e[j][k] for k in range(0,4)]
        vertex4 = [-e[i][k] - e[j][k] for k in range(0,4)]
        vertices.append(vertex1)
        vertices.append(vertex2)
        vertices.append(vertex3)
        vertices.append(vertex4)

# Check that reflecting the plane induces a bijection on the
    vertices of P

roots=[[1,-1,0,0],[0,1,-1,0],[0,0,1,0],[-1/2,-1/2,-1/2,-1/2]]
vertex_set= [tuple(t) for t in vertices] # For the set(.)
    function to work we need to use tuples instead of lists.

for v in roots:
    vertices2=[]
    for w in vertices:
        vertices2.append(plane_reflex(v,w)) # We create the
    list of reflected vertices.
    reflected_set = [tuple(t) for t in vertices2]
    if set(reflected_set)==set(vertex_set):
        print("Hyperplane is ok")
```

The output is the following:

```
Hyperplane is ok
Hyperplane is ok
Hyperplane is ok
Hyperplane is ok
```

d) For this, we simply check which vertices hold the inequality for all roots.

```python
roots = [[1,-1,0,0],[0,1,-1,0],[0,0,1,0],[-1/2,-1/2,-1/2,-1/2]]
cone = []  # list containing all vertices in the fundamental cone.

for x in vertices:
    condicion = 1
    for v in roots:
        if innerproduct(x, v) < 0:
            condicion = 0
    if condicion == 1:
        cone.append(x)
```

There is only one vertex, $(1, 0, 0, -1)$, in the fundamental cone of F_4.

e) We did not finish this part, but we write down the conclusions, and guesses that we got from our work.

It would make sense that the number of facets is 24, given the fact that P is called 24-cell. We searched for all facet inequalities, and we found octahedrons every time. Nevertheless, we could only find 16 octahedrons. Now, we write down the inequalities we found, and show that each of them induces an octahedron.

First of all, all vertices verify that $x_1 + x_2 + x_3 + x_4 \leq 2$, because of the form they are constructed. The equality is reached on those vertices of the form $e_i + e_j$. In particular, this is a hypersimplex, $\Delta(4, 2)$, and so an octahedron. By part c), P is symmetrical with respect to the plane $x_1 + x_2 + x_3 + x_4 = 0$, so we get another octahedron given by the inequality $x_1 + x_2 + x_3 + x_4 \geq -2$. Thus, we find 2 octahedrons.

We can see that, for each $i = 1, \ldots, 4$, we have the inequality $x_i \leq 1$. The inequality is reached on vertices with $x_i = 1$, and there are 6 of them. Furthermore, the resulting facet is affinely isomorphic to the 3-dimensional crosspolytope $\{\pm e_j : j \in \{1, 2, 3, 4\} \setminus \{i\}\}$. Hence, these facets are also octahedrons. Again , we can do the same for the inequality $x_i \geq -1$ to get symmetrical octahedrons. From here, we have 4 choices of i, and two octahedrons, $-1 \leq x_i \leq 1$ for each choice. Thus, we find 8 octahedrons.

Notice also that, if i, j, k, l are all distinct, then all vertices verify the inequality $x_i + x_j \leq x_k + x_l + 2$, and again, equality is reached on 6 vertices, namely

$$e_i + e_j, e_i - e_k, e_i - e_l, e_j - e_k, e_j - e_l, -e_k - e_l.$$

If we assume two vertices are adjacent iff they share a coordinate, then this gives us more octahedrons. The number of inequalities is the number of ways of choosing 2 elements out of 4, so we get another 6 octahedrons. In this case, we do not have the symmetry that we had in the other cases, since that would lead us to $-x_i - x_j \leq x_k + x_l + 2$, which is already one of our cases.

In total, we find 16 octahedrons. To find the (we guess) missing 8, we think it is possible to get to something using part d), since we did not use it yet, and there is only one vertex in the cone.