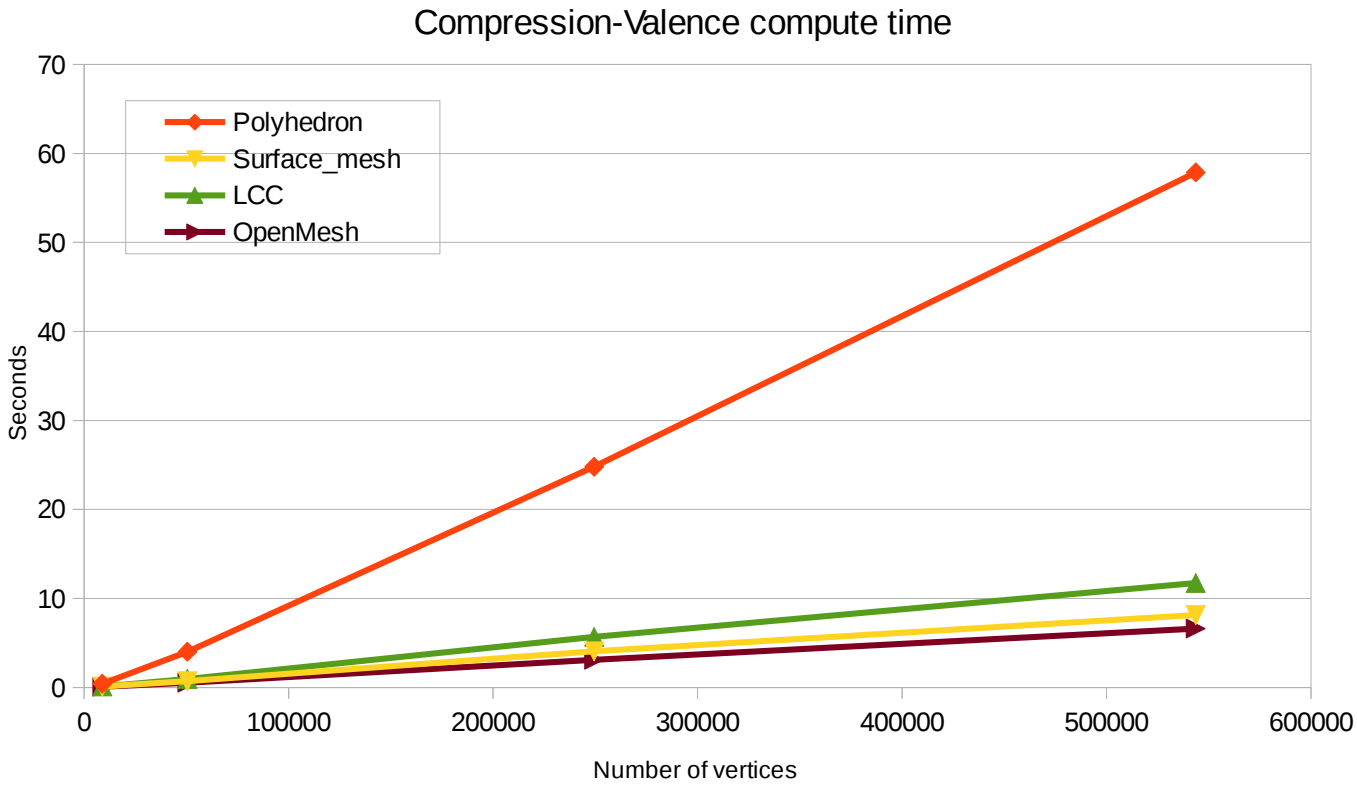
Compression-Decompression Valence component benchmarking

and datastructures comparison

1) Compression compute time

We compare the compression time with four different datastructures (Polyhedron, Surface_mesh, LCC, OpenMesh) for four meshes of growing size, from about 9k vertices to 550k vertices.



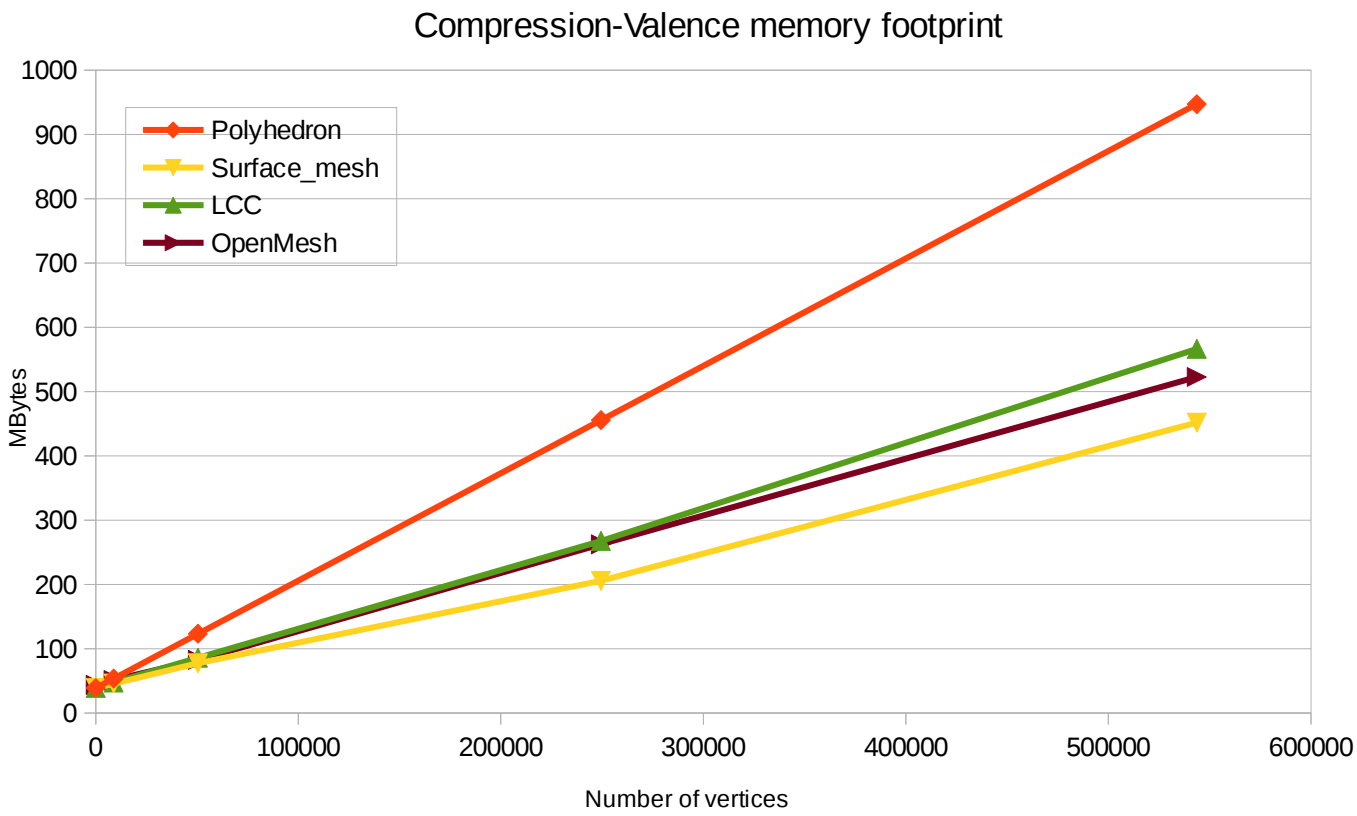
The compute time appears to vary linearly with the number of vertices of the mesh. The slowest result is obtained with Polyhedron and the fastest result is obtained with OpenMesh.

There is a 4 to 6 factor between Polyhedron and the other datastructures. It is important to note that the Polyhedron datastructure uses associative property maps (slower) to store mesh attributes, while the three other datastructures use vector property maps (faster).

Considering only Surface_mesh, LCC and OpenMesh, the fastest compute time is obtained with OpenMesh. Compared to OpenMesh, the compression with Surface_mesh is 23% slower, and the compression with LCC is 77% slower.

2) Compression memory footprint

We compare the memory footprint in the same conditions as before.



Note: in the chart above, the data for zero vertices shows the memory used by the libraries (CGAL and OpenMesh) when no mesh is loaded.

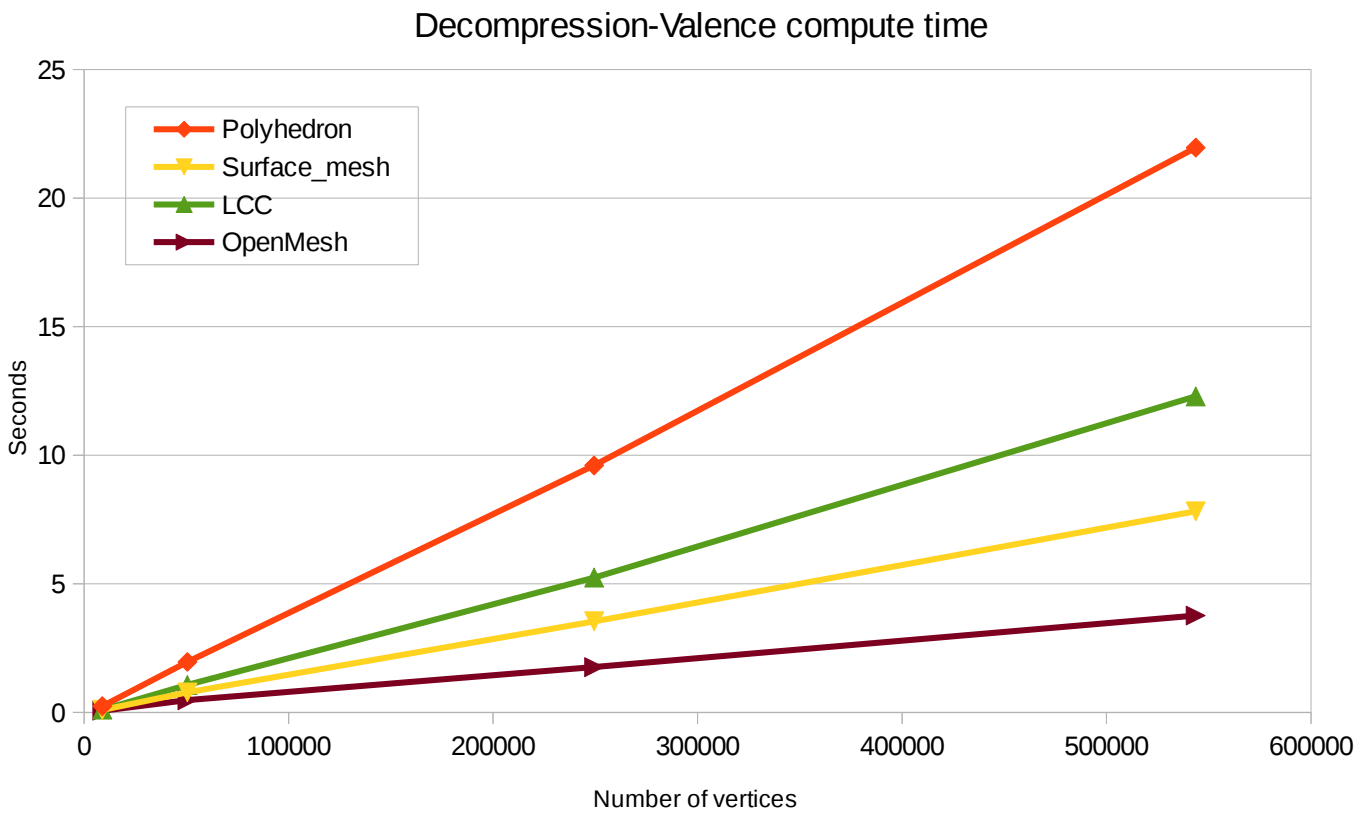
The memory used seems to also vary linearly with the number of vertices of the mesh. The Polyhedron datastructure uses about twice as much memory as the other datastructures.

The difference between Polyhedron and the other datastructure is also due to the side effect of associative property maps. For example, forcing the use of associative property maps with Surface_mesh leads to a memory consumption of 763 MBytes instead of 451 MBytes for the 550k vertices mesh.

Considering only Surface_mesh, LCC and OpenMesh, the lowest memory footprint is obtained with Surface_mesh. Compared to Surface_mesh, the compression with OpenMesh uses 16% more memory, and the compression with LCC uses 25% more memory.

3) Decompression compute time

We compare the decompression time with four different datastructures (Polyhedron, Surface_mesh, LCC, OpenMesh) for the four meshes compressed in section 1.



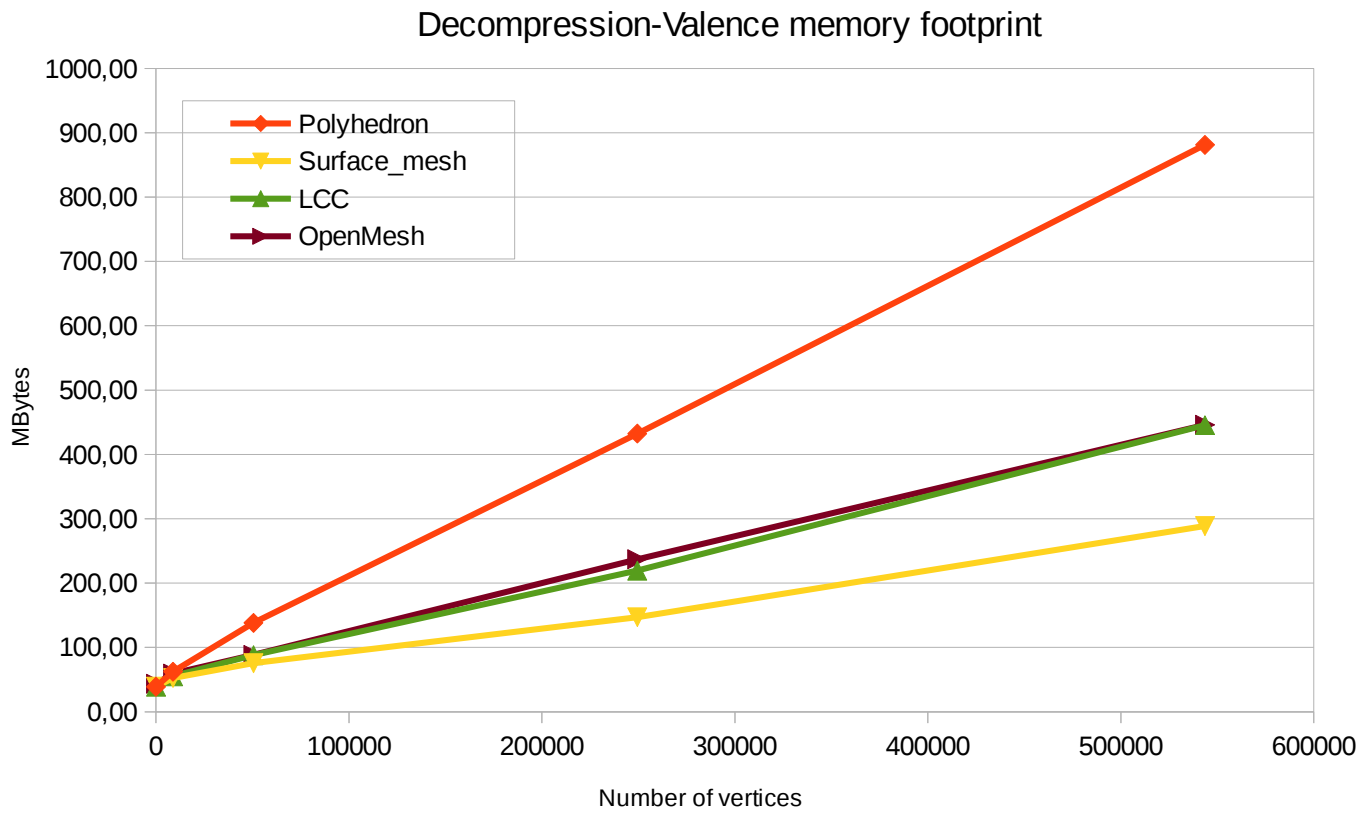
The compute time seems to vary linearly with the number of vertices of the mesh. Like for the compression, the slowest result is obtained with Polyhedron and the fastest result is obtained with OpenMesh.

The slow compute time with Polyhedron is due to the use of associative property maps with this datastructure.

Considering the three other datastructures, the results are more spread out than for the compression. The fastest compute time is still obtained with OpenMesh. Compared to OpenMesh, the decompression with Surface_mesh is 2 times slower, and the decompression with LCC is more than 3 times slower.

4) Decompression memory footprint

We compare the memory footprint for the decompression in the same conditions as section 3.



The memory footprint varies rather linearly with the number of vertices of the mesh. The decompression with Polyhedron uses 2 times more memory than any other datastructure. As usual, this is due to the associative property maps.

The decompression with Surface_Mesh uses the lowest amount of memory. The decompression with OpenMesh and LCC use the same amount of memory, which is 55% higher than with Surface_Mesh.

5) Synthesis

The table below shows the best results obtained according to each criterion, and the penalty factor encountered when using a non-optimal datastructure.

Best result according to criterion and related penalty factor for other datastructures

Criterion	Polyhedron	Surface_mesh	LCC	OpenMesh
Compression compute time	x 8,75	x 1,23	x 1,77	best
Compression memory footprint	x 2,10	best	x 1,25	x 1,16
Decompression compute time	x 5,79	x 2,05	x 3,23	best
Decompression memory footprint	x 3,05	best	x 1,54	x 1,54

From the results above we can conclude that the optimal datastructure for Compression-Decompression Valence component is OpenMesh according to the compute time criterion, and Surface_mesh according to the memory footprint criterion.