Dynamic update of succinct triangulations

CCCG - august 2005

Luca Castelli Aleardi

(joint work with Olivier Devillers and Gilles Schaeffer)

Projet Geometrica
INRIA Sophia-Antipolis

LIX
Ecole Polytechnique
Compact representations

Given a class $C_m$ of objects of size $m$, the goal is to design a space efficient data structure such that:

- queries on objects are answered in constant time;
- the encoding is succinct: the cost of an object $R \in C_m$ matches asymptotically the entropy of the class
  \[
  \text{size}(R) = \log_2 \|C_m\|(1 + o(1))
  \]
- or compact: we content of a cost
  \[
  \text{size}(R) = O(\|C_m\|)
  \]
- for dynamic data structures: updates are supported in
  \[
  O(\lg^c m) \text{ amortized time}
  \]
Compact representations

An example: rooted trees with \( n \) vertices

\[
\|B_n\| = \frac{1}{n + 1} \binom{2n}{n} \approx 2^{2n} n^{-\frac{3}{2}}
\]
Compact representations

An example: rooted trees with $n$ vertices

compact encoding for compression

- size: $\log_2 \| B_n \| = 2n + O(\lg n)$ bits
- no efficient navigation

explicit pointers-based representation

- size: $2n \lg n$ bits
- constant time navigation

succinct representation (Jacobson 89, Munro et Raman 97)

- size: $2n + o(n)$ bits
- adjacency queries in constant time
Motivation

Mesh compression versus compact representation

VRML, 288 or 114 bits/vertex

[Touma Gotsman] 2 bits/vertex (near-optimal)
[Poulalhon Schaeffer] 3.24 bits/vertex (optimal)

Pointer based representation: 208 bits/triangle

2.175 bits/triangle
Succinct dynamic data structures

Succinct dynamic binary trees on \( n \) nodes

Munro Raman Storm (SODA’01)  Raman Rao (ICALP’03)

inserting/deleting a leaf
inserting a node along an edge
\( O(\lg^2 n) \) amortized time

inserting/deleting a leaf
\( O((\lg \lg n)^{1+\varepsilon}) \) amortized time
Previous and related works

- static trees on $n$ nodes (Jacobson FOCS89): space $2n + o(n)$, navigation in $O(\lg n)$ time;
- planar graphs on $n$ vertices and $e$ edges (Munro Raman FOCS97): space $8n + 2e$, $O(1)$ time navigation;
- 3-connected planar graphs on $n$ vertices (Chuang et al. ICALP98): space $2e + n$, $O(1)$ time navigation;
- separable graphs (Blandford et al. SODA03): space $O(n)$, navigation in $O(1)$ time.
- dynamic binary trees (Munro et al. SODA01, Raman Rao ICALP03): space $2n + o(n)$, navigation in $O(1)$ updates in poly-logarithmic amortized time;
Tutte’s entropy (triangulations)
(information theory asymptotic lower bound)

![image]

enumeration of rooted planar triangulations on $n$ vertices:

$$\Psi_n = \frac{2(4n + 1)!}{(3n + 2)!(n + 1)!} \approx \frac{16}{27} \sqrt{\frac{3}{2\pi}} n^{-5/2} \left(\frac{256}{27}\right)^n$$

Tutte’s entropy (1962):

$$e = \frac{1}{n} \log_2 \Psi_n \approx \log_2 \left(\frac{256}{27}\right) \approx 3.2451 \text{ bits/vertex}$$
Planar Triangulations with a boundary

$n + 1$ internal vertices, $m = 2n + k$ faces

$$f(n, k) = \frac{2 \cdot (2k - 3)! (2k + 4n - 1)!}{(k - 1)! (k - 3)! (n + 1)! (2k + 3n)!}$$

$$f'(m, k) = \frac{2 \cdot (2k - 3)! (2m - 1)!}{(k - 1)! (k - 3)! \left(\frac{m-k}{2} + 1\right)!}$$

counting planar triangulations with $m$ faces

$$F(m) = \lg(\sum_{k \geq 3} f'(m, k)) \approx 2.175m$$

3.24 bits/vertex = 1.62 bits/face < 2.17 bits/face
Static succinct triangulations

(to be presented at WADS 2005)

Theorem (Castelli Aleardi, Devillers and Schaeffer). For planar triangulations with a boundary having \( m \) faces, there exists an optimal succinct representation supporting efficient navigation in \( O(1) \) time, requiring

\[
2.175m + O(m \frac{\lg \lg m}{\lg m}) = 2.175m + o(m) \text{ bits}
\]

For triangulations of genus \( g \) surfaces \( (g = o(\frac{m}{\lg m})) \) the same representation requires

\[
2.175m + 36(g - 1) \lg m + O(m \frac{\lg \lg m}{\lg m} + g \lg \lg m) \text{ bits}
\]
# Comparison: space efficiency

Compact representations of triangulations with \( n \) vertices, \( e \) edges, \( m \) faces (lower order term are omitted)

<table>
<thead>
<tr>
<th>Encoding</th>
<th>queries</th>
<th>planar</th>
<th>higher genus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jacobson (FOCS 89)</td>
<td>( O(\lg n) )</td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>Munro Raman (FOCS 97)</td>
<td>( O(1) )</td>
<td>8( n + 2e ) or 7( m )</td>
<td>no</td>
</tr>
<tr>
<td>Chuang et al. (ICALP 98)</td>
<td>( O(1) )</td>
<td>2( e + n ) or 3.5( m )</td>
<td>no</td>
</tr>
<tr>
<td>Chiang et al. (SODA 01)</td>
<td>( O(1) )</td>
<td>2( e + n ) or 3.5( m )</td>
<td>no</td>
</tr>
<tr>
<td><strong>Castelli Aleardi et al. WADS 2005</strong></td>
<td>( O(1) )</td>
<td>2.175( m )</td>
<td>2.175( m )</td>
</tr>
</tbody>
</table>
Basic ideas

• Multi-level hierarchical structure
• Exhaustive enumeration
• Optimal encoding
• Information sharing

Additional tools for the dynamic case

• Memory organization based on \textit{space efficient dynamic arrays}
• New strategy for local redecomposition
Decomposing $\mathcal{T}$ into sub-triangulations

- we compute tiny triangulations having between $\frac{1}{12} \lg m$ and $\frac{1}{4} \lg m$ triangles;
- we regroup tiny triangulations to form small triangulations containing $\Theta(\lg m)$ tiny triangulations.
Decomposition phase

We start with a triangulation having $m$ triangles
Decomposition phase

Computing tiny triangulations having $\Theta(\lg m)$ triangles
Decomposition phase

There are $\Theta\left(\frac{m}{\lg m}\right)$ tiny triangulations
Decomposition phase

Only boundary edges are shared by tiny triangulations
Decomposition phase

Graph $G$ linking adjacent tiny triangulations
Decomposition phase

A small triangulation contains $\Theta(\lg^2 m)$ triangles
Decomposition phase

There are $\Theta\left(\frac{m}{\lg^2 m}\right)$ small triangulations
Decomposition phase

Graph $F$ linking adjacent small triangulations
Decomposition phase

Partitioning graph $G$: graphs $G_i$ link tiny triangulations lying in a same small triangulation.
Overview: representation of a small triangulation

- adjacency relations are described by map $G_i$;
- internal connectivity is implicitly represented (variable size pointers);
- boundary neighboring relations are represented by boundary coloring (variable length bit-vector).

Dynamic update of succinct triangulations – p.23/51
Graph $G_i$ linking adjacent tiny triangulations

- $G_i$ has a node for each tiny triangulation and an \textit{arc} for each pair of adjacent tiny triangulations;
- $G_i$ is a planar map, having faces of degree at least 3, multiple edges and loops are allowed;
• Because of Euler’s formula, the overall number of arcs in maps $G_i$ is:

$$\sum_i \|E(G_i)\| = O\left(\frac{m}{\lg m}\right)$$
Memory organization overview

Graph of tiny triangulations

node of $G$

node of $F$

degree  # triangles  # boundary
neighbor neighbor neighbor

index  backward index

node of $F$

degree
voisin voisin voisin

local pointer

global pointer
Graph of tiny triangulations

node of $G$

<table>
<thead>
<tr>
<th>degree</th>
<th># triangles</th>
<th># boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Boundary coloring

Dynamic update of succinct triangulations – p.27/51
Extendible arrays (Raman Rao ICALP’03)

**Proposition.** It is possible to maintain $n$ records of $r$ bits each under insertion of new records, while supporting access in $O(1)$ worst-case time. The updates (grow and shrink) are performed in $O(1)$ amortized time and the wasted space is $O(w + \sqrt{nrw})$ ($w$ being the size of a word machine).

![Diagram](image_url)

- $B_1$, $B_2$, $B_3$, $B_i$
- $O(\sqrt{s/w})$ blocks
- $\|B_i\| = i$ records
- Block names
Memory organization

Collection of extendible arrays storing implicitly the tiny triangulations: $2.17r$ bits pointers

- $\sqrt{\lg m}$ bits
- $2\sqrt{\lg m}$ bits
- $3\sqrt{\lg m}$ bits
- $4\sqrt{\lg m}$ bits
- $k\sqrt{\lg m}$ bits: $2.17r$, $\lg \lg m$, $O(\sqrt{\lg m})$ backward pointer, wasted space
- $\lg m$ bits

$\parallel t_{ij} \parallel = r$
Memory organization

Collection of extendible arrays storing implicitly the boundary colorings: $w_{ij} \lg b_{ij}$ bits pointers

\[ PB_i \]

- $\sqrt{\lg m}$ bits
- $2\sqrt{\lg m}$ bits
- $3\sqrt{\lg m}$ bits
- $4\sqrt{\lg m}$ bits
- $k\sqrt{\lg m}$ bits

\[ w_{ij} \lg b_{ij} \quad \lg \lg m \quad O(\sqrt{\lg m}) \quad \ldots \quad \ldots \quad \ldots \]

\[ b_{ij} = O(\lg m) \quad w_{ij} = O(\lg m) \]
Memory organization

Representation of a node $n_{ij}$ in map $G_i$
Overall cost of graphs $G_i$

- list of neighbors, nodes degrees, size of nodes, ... (local pointers of size $O(\lg \lg m) - O\left(\frac{m}{\lg m}\right)$ nodes and arcs)

$$O\left(m\frac{\lg \lg m}{\lg m}\right)$$

- pointers to table $A_r$ (combinatorial information)

$$2.17m + O(\lg m)$$

- pointers to "Rank/Select" tables (boundary coloring)

$$\sum_t \|RS(t)\| \leq \sum_t \lg \left(\frac{\lg m}{w(t)}\right) \leq O\left(m\frac{\lg \lg m}{\lg m}\right)$$
Total space used

• Catalog of all different tiny triangulations

\[ O\left(m^{\frac{1}{4}2.17} \lg^2 m \lg \lg m\right) = o(m) \]

• catalog of bit-vectors (with Rank/Select)

\[ O\left(m^{\frac{1}{4}2.17} \lg m \lg \lg m\right) = o(m) \]

• representation of graph \( F \): \( O\left(\frac{m}{\lg^2 m} \lg m\right) = o(m) \)

• graphs \( G_i \)

\[ 2.17m + O(m\frac{\lg \lg m}{\lg m}) \]
Local updates

Problems arising from degree 3 vertex insertion

- increasing of the size of tiny (small) triangulations, after vertex insertion

\[ \|t_r\| \leq \frac{1}{4} \lg m \quad \|t_r\| > \frac{1}{4} \lg m \quad \|t_1\| \leq \frac{1}{4} \lg m \quad \|t_2\| \leq \frac{1}{4} \lg m \]
Local updates

Problems arising from degree 3 vertex deletion

- topology of graphs $G_i$ can change drastically after vertex deletion
Local updates

Problems arising from edge flip

- topology of graphs $G_i$ can change drastically after edge flip
Our contribution

An updatable succinct representation for triangulations

Theorem. For triangulations with a boundary having \( m \) faces, it is possible to maintain a succinct representation under vertex insertion/deletion and edge flip, while supporting navigation in \( O(1) \) time. The storage is

\[
2.175m + O\left(m \frac{\lg \lg m}{\lg m}\right) = 2.175m + o(m) \text{ bits}
\]

The cost for an update is:

- \( O(1) \) amortized time for degree 3 vertex insertion;
- \( O(\lg^2 m) \) amortized time for vertex deletion and edge flip;
Updating the data structures

Updates of the implicit representation

\[ \|t_{ij}\| = r \]

\[ PE_i \]

\[ PA_i \]

<table>
<thead>
<tr>
<th>Bits</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \sqrt{\lg m} )</td>
<td>[ \text{structure} ]</td>
</tr>
<tr>
<td>( 2\sqrt{\lg m} )</td>
<td>[ \text{structure} ]</td>
</tr>
<tr>
<td>( 3\sqrt{\lg m} )</td>
<td>[ \text{structure} ]</td>
</tr>
<tr>
<td>( 4\sqrt{\lg m} )</td>
<td>[ \text{structure} ]</td>
</tr>
<tr>
<td>( k\sqrt{\lg m} )</td>
<td>[ \text{structure} ]</td>
</tr>
</tbody>
</table>
Updating the data structures

Pointers in collection $PA_i$ have to be updated after a vertex insertion

$\|t_{ij}\| = r + 4$
Updating the data structures

The updated tiny triangulation is still valid ($||t_{ij}|| \leq \frac{1}{4} \log m$): no decomposition procedure is needed.

$\|
\begin{align*}
\sqrt{\log m} \text{ bits} & \quad \n_i \\
2\sqrt{\log m} \text{ bits} & \quad P_{E_i} \\
3\sqrt{\log m} \text{ bits} & \quad PA_i \\
4\sqrt{\log m} \text{ bits} & \\
k\sqrt{\log m} \text{ bits} & \\
\end{align*}
\|
=r+10
Local decomposition

Splitting a small triangulation

\[ ST'_{ij} \]

\[ ST''_{ij} \]
Local decomposition

Splitting a small triangulation

$TT_{ij}$
Local decomposition

Splitting a small triangulation

\[ T T_{ij} \]
Local decomposition

Splitting a small triangulation
Local decomposition

Splitting a small triangulation

\[ \mathcal{B}_2 \]
Local decomposition

Splitting a small triangulation

\[ B' \]

\[ B'' \]

Dynamic update of succinct triangulations – p.46/51
Local decomposition

Splitting a small triangulation

- 

\[ \mathcal{T}_i \mathcal{T}_j \]

\[ \mathcal{T}_i' \mathcal{T}_j' \]

\[ \mathcal{T}_i'' \mathcal{T}_j'' \]
Local decomposition

Splitting a small triangulation

• ;

$ST_i$, $TT_{ij}$, $TT'_{ij}$, $TT''_{ij}$, $ST'_i$, $ST''_i$, $ST_{ij}$
A practical solution

(Abdelkrim Mebarki and Olivier Devillers)

C++ implementation based on CGAL library

Idea: gathering triangles in small groups
Open problem

Optimal succinct encoding for planar triangulations, achieving Tutte’s entropy: 1.62 bits/face.

Idea: canonical decomposition strategy based on optimal encoding (Poulalhon et Schaeffer ICALP03)
Future work

Triangulations 3D

Any idea?