A Parallel DCEL Proposal

Andres Calderon

University of California, Riverside

June 12, 2019
Outline

DCEL basic concepts

Literature review

Proposal
What is a DCEL?

- Doubly connected edge list (DCEL) is a data structure suited to represent a connected planar graph embedded in the plane (Muller-Preparata, 2017).
- A planar embedding of $G = (V, E)$ does not allow crossing edges.
- DCEL captures key topological information about vertices, edges and faces.
DCEL basic concepts

DCEL examples

(Freiseisen, 1998)
DCEL examples

Example 1. The following example depicted in Figure 3 shows a simple polygon with a dangling edge represented as a DCEL, where \( A \) is the outer face and \( B \) is the interior one. Note: the dangling edge \( a \) has two equal adjacent faces, i.e. \( F_1 = F_2 = A \).

\[
\begin{array}{c|c|c|c|c}
E & V_1 & V_2 & F_1 & F_2 \\
\hline
a & 1 & 2 & A & A \\
b & 2 & 5 & A & B \\
c & 2 & 3 & B & A \\
d & 3 & 4 & B & A \\
e & 4 & 5 & B & A \\
\end{array}
\]

(Freiseisen, 1998)

Example 2. In Figure 4 is shown a more practical but not too complicated example with three faces and without dangling elements.

Here we give some important type definitions, which will be used in some of the programs. \( I \) is an interface or traits class. It holds a lot of special technical definitions and types necessary for an actual implementation of the given programs (for a detailed information of \( I \) see the appendix).

For storing the elements of a DCEL we use the following \( \cdots \)-containers:

(DCEL proposal, June 12, 2019)
Computing the overlay

Now that we have designed a good representation of a subdivision, we can tackle the general map overlay problem. We define the overlay of two subdivisions $S_1$ and $S_2$ to be the subdivision $O(S_1, S_2)$ such that there is a face $f$ in $O(S_1, S_2)$ if and only if there are faces $f_1$ in $S_1$ and $f_2$ in $S_2$ such that $f$ is a maximal connected subset of $f_1 \cap f_2$. This sounds more complicated than it is: what it means is that the overlay is the subdivision of the plane induced by the edges from $S_1$ and $S_2$. Figure 2.4 illustrates this. The general map overlay problem is to compute a doubly-connected edge list for $O(S_1, S_2)$, given the doubly-connected edge lists of $S_1$ and $S_2$. We require that each face in $O(S_1, S_2)$ be labeled with the labels of the faces in $S_1$ and $S_2$ that contain it. This way we have access to the attribute information stored for these faces. In an overlay of a vegetation map and a precipitation map this would mean that we know for each region in the overlay the type of vegetation and the amount of precipitation. Let’s first see how much information from the doubly-connected edge lists for $S_1$ and $S_2$ we can re-use in the doubly-connected edge list for $O(S_1, S_2)$.
The algorithm

1. Find all intersections over the edges of the polygons.
2. Construct a planar graph by inserting all edges and all intersections (DCEL).
3. Traverse this planar graph in order to perform an intersection, union or difference.
Advantages

▶ DCEL captures topological information.
▶ DCEL allows multiple spatial operations.
▶ DCEL can be constructed from this in $O(n \log(n))$ time using $O(n)$ additional memory.
▶ DCEL allows boolean operations in $O(n)$ time using $O(n)$ additional space.
Outline

DCEL basic concepts

Literature review

Proposal
Literature review


▶ First 5 entries contain solid introductory material about DCELs, specially de Berg et al. (2008) and Freiseisen (1998).

▶ Next entries discuss alternative methods for map overlay, particularly exploring sequential techniques and parallel spatial joins.

▶ *There is no mention to a distributed DCEL implementation.*
Literature review

- Sequential techniques (more relevant):
Literature review

- Parallel spatial joins (more relevant):
Current (sequential) implementations


- **dyn4j**: A 100% Java 2D collision detection and physics engine. [http://www.dyn4j.org/](http://www.dyn4j.org/).

- **anglyan/dcel**: Python implementation of a doubly connected edge list. [https://github.com/anglyan/dcel](https://github.com/anglyan/dcel).

Outline

DCEL basic concepts

Literature review

Proposal
The proposal

A parallel DCEL implementation to support big spatial data overlay operations.
What is the problem?

▶ Currently, most map overlay methods are sequential based.
▶ For layers with thousand of polygons the execution time is not feasible.
▶ Most of techniques are oriented to a specific spatial operation (intersection, union, difference, ...).
▶ Parallel techniques divide the data into partitions and duplicate features if needed in order to solve the problem locally.
▶ Data structures collecting topological properties are not explored.
Why is it important?

- The rise of big (spatial) data makes necessary to count with fast and efficient techniques for spatial analysis.
- In particular, the RIDIR project has to deal with spatial operations between layers collecting thousand of counties nation-wide. The versatility and efficiency of the spatial methods is cardinal for their studies.
- It should be interesting to count with intermediate data structures that allow multiple map overlay queries.
- DCEL allows linear time to compute spatial operations.
What are the limitations of related work?

- Topological data structures are common in computational geometry. However, most implementations are sequential and they do not scale appropriately on large spatial datasets.
- Parallel spatial joins add complexity due to spatial partitioning necessarily introduces replication.
Why is it challenging?

- Initial stages of the DCEL construction expect it fits in main memory.
- Subsequent operations should be able to query the DCEL in a transparent way.
What are our novel contributions?

- At the best of our knowledge, there is not a distributed DCEL implementation.
- It is necessary to design procedures for partitioning and merging during the DCEL construction.
- The spatial methods that run over the DCEL are based on boolean operations. They should be adjusted accordingly to a distributed DCEL.
What is the validation method?

- We can compare the new implementation to sequential versions of the algorithm.
- To test scalability we can perform benchmarks to the current versions of `area_tables` and `area_interpolate`. 