A variant of backwards analysis applicable to order-dependent sets

Evanthia Papadopoulou [1]   Martin Suderland [2]

[1]Faculty of Informatics, Università della Svizzera italiana (USI), Lugano, Switzerland

[2]Courant Institute, New York University (NYU), New York, NY, USA

14.3.2024 - Ioannina, Greece - EuroCG 2024

Recap: Standard backwards analysis

Randomized incremental algorithm

- n elements
- randomization order $\sigma \in S_n$
- T_i: running time of the i-th step
- O_i: output structure after step i

Recap: Standard backwards analysis

Randomized incremental algorithm

- n elements
- randomization order $\sigma \in S_n$
- T_i: running time of the i-th step
- O_i: output structure after step i

Time analysis idea:
"Analyze an algorithm as if it were running backwards in time, from output to input. This is based on the observation that often the cost of the last step of an algorithm can be expressed as a function of the complexity of the final product output of the algorithm" [Seidel, 1993].

Recap: Standard backwards analysis

Randomized incremental algorithm

- n elements
- randomization order $\sigma \in S_n$
- T_i: running time of the i-th step
- O_i: output structure after step i

Time analysis idea:
"Analyze an algorithm as if it were running backwards in time, from output to input. This is based on the observation that often the cost of the last step of an algorithm can be expressed as a function of the complexity of the final product output of the algorithm" [Seidel, 1993].

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{j=1}^{i} \sum_{\substack{\sigma \in S_i \\ \sigma(i)=j}} T_i(\sigma)}{i!}$$

Recap: Standard backwards analysis

Randomized incremental algorithm

- n elements
- randomization order $\sigma \in S_n$
- T_i: running time of the i-th step
- O_i: output structure after step i

Time analysis idea:
"Analyze an algorithm as if it were running backwards in time, from output to input. This is based on the observation that often the cost of the last step of an algorithm can be expressed as a function of the complexity of the final product output of the algorithm" [Seidel, 1993].

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{j=1}^{i} \sum_{\substack{\sigma \in S_i \\ \sigma(i)=j}} \overbrace{T_i(\sigma)}^{\leqslant f(O_i(\sigma))}}{i!}$$

Recap: Standard backwards analysis

Standard backwards analysis

- First appearance [Chew, 1985]:
  Voronoi diagram of a set of points in convex position in $\mathbb{R}^2$

Recap: Standard backwards analysis

Standard backwards analysis

- First appearance [Chew, 1985]:
  Voronoi diagram of a set of points in convex position in $\mathbb{R}^2$
- Many other applications [Boissonnat and Yvinec, 1998]:
- Popularized by [Seidel, 1993]
  provides a "bad example", where backwards analysis is not applicable

Recap: Standard backwards analysis

Standard backwards analysis

- First appearance [Chew, 1985]:
  Voronoi diagram of a set of points in convex position in $\mathbb{R}^2$
- Many other applications [Boissonnat and Yvinec, 1998]:
- Popularized by [Seidel, 1993]
  provides a "bad example", where backwards analysis is not applicable
- Key requirement:
  algorithm's output is independent of the randomization order

Variant of backwards analysis: groups of permutations

Idea overview:

- look at groups of similar permutations
- bound the time for the entire group in terms of one output structure

Variant of backwards analysis: groups of permutations

Idea overview:

- look at groups of similar permutations
- bound the time for the entire group in terms of one output structure

For a permutation $\sigma = (\sigma_1, \sigma_2 ..., \sigma_n)$ and index $1 \leqslant i \leqslant n$ define the **shifted permutation** $\sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n, \sigma_i)$

Variant of backwards analysis: groups of permutations

Idea overview:

- look at groups of similar permutations
- bound the time for the entire group in terms of one output structure

For a permutation $\sigma = (\sigma_1, \sigma_2 ..., \sigma_n)$ and index $1 \leqslant i \leqslant n$ define the **shifted permutation** $\sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1}, ..., \sigma_n, \sigma_i)$

$$G_\sigma = \left\{ \begin{array}{l} \sigma = (\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n) = \sigma^{(n)} \\ \hline \sigma^{(1)} = (\sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_1) \\ \sigma^{(2)} = (\sigma_1, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_2) \\ ... \\ \sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1} ..., \sigma_n, \sigma_i) \\ ... \\ \sigma^{(n-1)} = (\sigma_1, \sigma_2, ..., \sigma_{n-2}, \sigma_n, \sigma_{n-1}) \end{array} \right\}$$

Partition of the set of permutations

Definition

A set $R_n \subset S_n$, which satisfies $\dot{\bigcup}_{\sigma \in R_n} G_\sigma = S_n$ is called **set of representatives**.

Partition of the set of permutations

Definition

A set $R_n \subset S_n$, which satisfies $\dot{\bigcup}_{\sigma \in R_n} G_\sigma = S_n$ is called **set of representatives**.

For $n = 4$, we can choose

$$R_4 = \{(1,2,3,4),(2,1,4,3),(3,1,4,2),(3,2,4,1),(4,1,3,2),(4,2,3,1)\}$$

Partition of the set of permutations

Definition

A set $R_n \subset S_n$, which satisfies $\dot{\bigcup}_{\sigma \in R_n} G_\sigma = S_n$ is called **set of representatives**.

For $n = 4$, we can choose

$$R_4 = \{(1,2,3,4),(2,1,4,3),(3,1,4,2),(3,2,4,1),(4,1,3,2),(4,2,3,1)\}$$

Lemma ([Levenshtein, 1992])

For all $n \in \mathbb{N}$ there exists a set of representatives R_n.

Partition of the set of permutations

Definition

A set $R_n \subset S_n$, which satisfies $\dot{\bigcup}_{\sigma \in R_n} G_\sigma = S_n$ is called **set of representatives**.

For $n = 4$, we can choose

$$R_4 = \{(1,2,3,4),(2,1,4,3),(3,1,4,2),(3,2,4,1),(4,1,3,2),(4,2,3,1)\}$$

Lemma ([Levenshtein, 1992])

For all $n \in \mathbb{N}$ there exists a set of representatives R_n.

Time analysis idea (variant backwards analysis):

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{\sigma \in R_i} T_i(G_\sigma)}{i!}$$

Partition of the set of permutations

Definition

A set $R_n \subset S_n$, which satisfies $\dot{\bigcup}_{\sigma \in R_n} G_\sigma = S_n$ is called **set of representatives**.

For $n = 4$, we can choose

$$R_4 = \{(1,2,3,4),(2,1,4,3),(3,1,4,2),(3,2,4,1),(4,1,3,2),(4,2,3,1)\}$$

Lemma ([Levenshtein, 1992])

For all $n \in \mathbb{N}$ there exists a set of representatives R_n.

Time analysis idea (variant backwards analysis):

$$E(T_i) = \frac{\sum_{\sigma \in S_i} T_i(\sigma)}{i!} = \frac{\sum_{\sigma \in R_i} \overbrace{T_i(G_\sigma)}^{\leqslant f(O_i(\sigma))}}{i!}$$

Why choose this grouping?

- each element appears exactly once as last element

$$G_\sigma = \left\{ \begin{array}{l} \sigma = (\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n) = \sigma^{(n)} \\ \hline \sigma^{(1)} = (\sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_1) \\ \sigma^{(2)} = (\sigma_1, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_2) \\ ... \\ \sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1} ..., \sigma_n, \sigma_i) \\ ... \\ \sigma^{(n-1)} = (\sigma_1, \sigma_2, ..., \sigma_{n-2}, \sigma_n, \sigma_{n-1}) \end{array} \right\}$$

Why choose this grouping?

- each element appears exactly once as last element
- minimizes the number of inversions between the base permutation σ and the permutations within its group G_σ

$$G_\sigma = \left\{ \begin{array}{l} \sigma = (\sigma_1, \sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n) = \sigma^{(n)} \\ \hline \sigma^{(1)} = (\sigma_2, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_1) \\ \sigma^{(2)} = (\sigma_1, \sigma_3, ..., \sigma_{n-1}, \sigma_n, \sigma_2) \\ ... \\ \sigma^{(i)} = (\sigma_1, ..., \sigma_{i-1}, \sigma_{i+1} ..., \sigma_n, \sigma_i) \\ ... \\ \sigma^{(n-1)} = (\sigma_1, \sigma_2, ..., \sigma_{n-2}, \sigma_n, \sigma_{n-1}) \end{array} \right\}$$

Why choose this grouping?

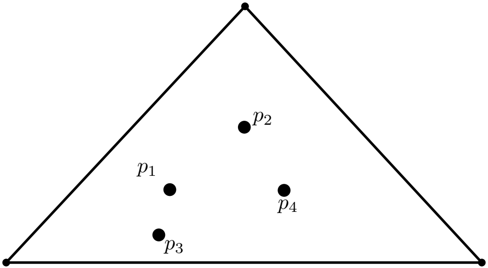
- each element appears exactly once as last element
- minimizes the number of inversions between the base permutation σ and the permutations within its group G_σ
  $\Rightarrow$ similarity between $O_i(\sigma)$ and $O_i(\sigma^{(j)})$ for all $j \leqslant i$

$$G_\sigma = \left\{ \begin{array}{l} \sigma = (\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_{n-1}, \sigma_n) = \sigma^{(n)} \\ \hline \sigma^{(1)} = (\sigma_2, \sigma_3, \ldots, \sigma_{n-1}, \sigma_n, \sigma_1) \\ \sigma^{(2)} = (\sigma_1, \sigma_3, \ldots, \sigma_{n-1}, \sigma_n, \sigma_2) \\ \ldots \\ \sigma^{(i)} = (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1} \ldots, \sigma_n, \sigma_i) \\ \ldots \\ \sigma^{(n-1)} = (\sigma_1, \sigma_2, \ldots, \sigma_{n-2}, \sigma_n, \sigma_{n-1}) \end{array} \right\}$$

Example: Triangulation algorithm [Seidel, 1993]

Input : Set of n points $P = \{p_1, p_2, ..., p_n\}$ within simplex $\Delta \subset \mathbb{R}^d$
Output : Triangulation T of P and Δ

```
T ← Δ;
for i = 1 → n do
    Remove simplex τ, which contains p_i, from T;
    Partition τ into d + 1 simplices, each having p_i as corner;
    Add the new d + 1 simplices to T;
return T;
```

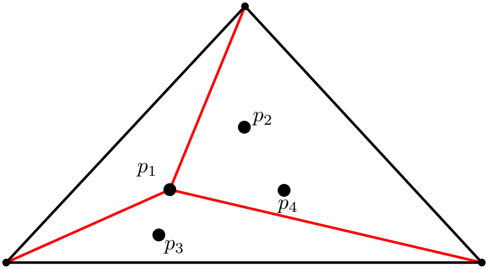


Triangulation for the insertion order $\sigma = (1, 2, 3, 4)$.

Example: Triangulation algorithm [Seidel, 1993]

Input : Set of n points $P = \{p_1, p_2, ..., p_n\}$ within simplex $\Delta \subset \mathbb{R}^d$
Output : Triangulation T of P and Δ

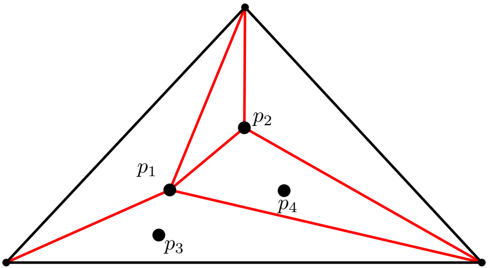
$\mathrm{T} \leftarrow \Delta$;
for $i = 1 \rightarrow n$ **do**
  Remove simplex τ, which contains p_i, from T;
  Partition τ into $d + 1$ simplices, each having p_i as corner;
  Add the new $d + 1$ simplices to T;
return T;



Triangulation for the insertion order $\sigma = (1, 2, 3, 4)$.

Example: Triangulation algorithm [Seidel, 1993]

```
Input  : Set of n points P = {p_1, p_2, ..., p_n} within simplex Δ ⊂ R^d
Output : Triangulation T of P and Δ
T ← Δ;
for i = 1 → n do
    Remove simplex τ, which contains p_i, from T;
    Partition τ into d + 1 simplices, each having p_i as corner;
    Add the new d + 1 simplices to T;
return T;
```

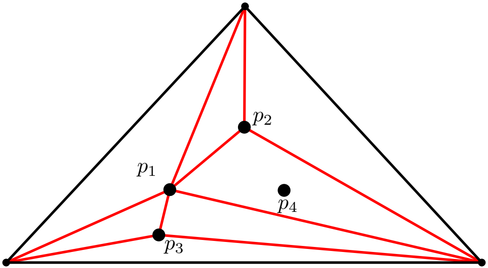


Triangulation for the insertion order $\sigma = (1, 2, 3, 4)$.

Example: Triangulation algorithm [Seidel, 1993]

Input : Set of n points $P = \{p_1, p_2, ..., p_n\}$ within simplex $\Delta \subset \mathbb{R}^d$
Output : Triangulation T of P and Δ

```
T ← Δ;
for i = 1 → n do
    Remove simplex τ, which contains p_i, from T;
    Partition τ into d + 1 simplices, each having p_i as corner;
    Add the new d + 1 simplices to T;
return T;
```

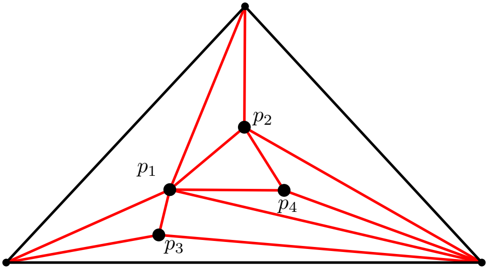


Triangulation for the insertion order $\sigma = (1, 2, 3, 4)$.

Example: Triangulation algorithm [Seidel, 1993]

Input : Set of n points $P = \{p_1, p_2, ..., p_n\}$ within simplex $\Delta \subset \mathbb{R}^d$
Output : Triangulation T of P and Δ

```
T ← Δ;
for i = 1 → n do
    Remove simplex τ, which contains p_i, from T;
    Partition τ into d + 1 simplices, each having p_i as corner;
    Add the new d + 1 simplices to T;
return T;
```



Triangulation for the insertion order $\sigma = (1, 2, 3, 4)$.

Expected time complexity

Theorem

For any fixed dimension, the triangulation algorithm has $O(n \log n)$ *expected time complexity.*

Expected time complexity

Theorem

For any fixed dimension, the triangulation algorithm has $O(n \log n)$ expected time complexity.

Information maintained during the algorithm:

1. for each triangle we know the points of P, which are contained in it
2. for each point $p \in P$ we know the triangle containing p.

Expected time complexity

Theorem

For any fixed dimension, the triangulation algorithm has $O(n \log n)$ expected time complexity.

Information maintained during the algorithm:

1. for each triangle we know the points of P, which are contained in it
2. for each point $p \in P$ we know the triangle containing p.

$\Rightarrow$ Count how many points fall into the deleted triangle in expectation!

Connection: shifted permutations ↔ group representative

For $0 \leqslant j \leqslant i \leqslant n$, permutation $\sigma \in S_n$ denote by

T_σ^i	partial triangulation after processing i many points specified by σ	
${}^jA_\sigma^i$	union of triangles of T_σ^i, which are incident to point $p_{\sigma(j)}$	
Δ_σ^i	triangle of T_σ^i, which contains point $p_{\sigma(n)}$	if $i < n$

Connection: shifted permutations ↔ group representative

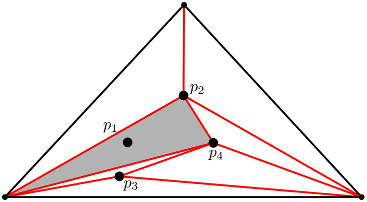
For $0 \leqslant j \leqslant i \leqslant n$, permutation $\sigma \in S_n$ denote by

T_σ^i	partial triangulation after processing i many points specified by σ	
${}^jA_\sigma^i$	union of triangles of T_σ^i, which are incident to point $p_{\sigma(j)}$	
Δ_σ^i	triangle of T_σ^i, which contains point $p_{\sigma(n)}$	if $i < n$

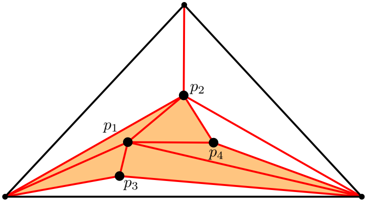
Lemma

For all $1 \leqslant j \leqslant i \leqslant n$ and any permutation $\sigma \in S_n$ it holds: $\Delta_{\sigma(j)}^{i-1} \subset {}^jA_\sigma^i$.

For insertion order $\sigma = (2, 1, 4, 3)$:



$T_{\sigma^{(2)}}^3$ with highlighted $\Delta_{\sigma^{(2)}}^3$



T_σ^4 with highlighted ${}^2A_\sigma^4$

Corollary

In the i-th insertion step, each of the $n - i$ remaining points has to be rebucketed for at most 3 permutations within a group G_σ for any $\sigma \in S_i$.

Corollary

In the i-th insertion step, each of the $n-i$ remaining points has to be rebucketed for at most 3 permutations within a group G_σ for any $\sigma \in S_i$.

Triangulation expected time complexity:

$$E(T(n)) = \sum_{i=1}^{n} \frac{\sum_{\sigma \in R_i} T_i(G_\sigma)}{i!} = \sum_{i=1}^{n} O\left(\frac{\sum_{\sigma \in R_i} 3(n-i)}{i!}\right)$$

$$= O\left(\sum_{i=1}^{n} \frac{n}{i}\right) = O(n \log n)$$

Another example: Voronoi-like diagrams

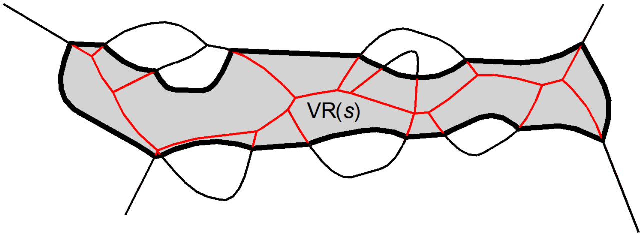
Voronoi-like diagrams [Junginger and Papadopoulou, 2023]

- intermediate structure in the incremental construction of Voronoi diagrams

Another example: Voronoi-like diagrams

Voronoi-like diagrams [Junginger and Papadopoulou, 2023]

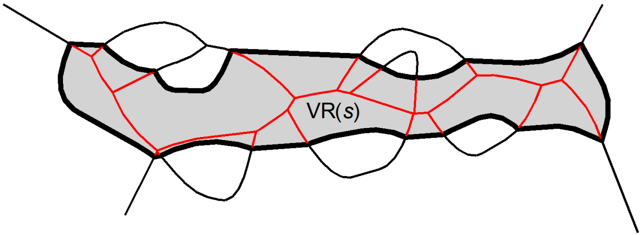
- intermediate structure in the incremental construction of Voronoi diagrams
- perform site-deletion in an abstract Voronoi diagram in linear time



Another example: Voronoi-like diagrams

Voronoi-like diagrams [Junginger and Papadopoulou, 2023]

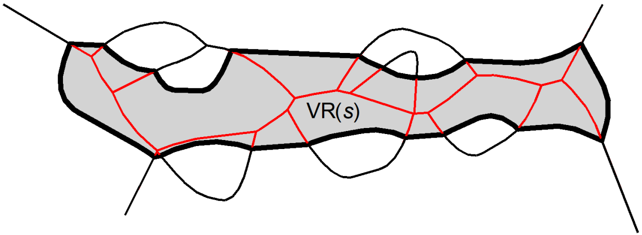
- intermediate structure in the incremental construction of Voronoi diagrams
- perform site-deletion in an abstract Voronoi diagram in linear time
- intermediate Voronoi-like diagrams are order dependent



Another example: Voronoi-like diagrams

Voronoi-like diagrams [Junginger and Papadopoulou, 2023]

- intermediate structure in the incremental construction of Voronoi diagrams
- perform site-deletion in an abstract Voronoi diagram in linear time
- intermediate Voronoi-like diagrams are order dependent
- final Voronoi-like diagram is oder independent!
  (it corresponds to a real Voronoi diagram)



Summary: A variant of backwards analysis

New variant suitable for:

- randomized incremental algorithms
- output may be **dependent** on the randomization order

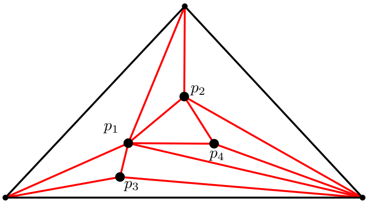
Summary: A variant of backwards analysis

New variant suitable for:

- randomized incremental algorithms
- output may be **dependent** on the randomization order

Example: Triangulation algorithm

[Seidel 1993]



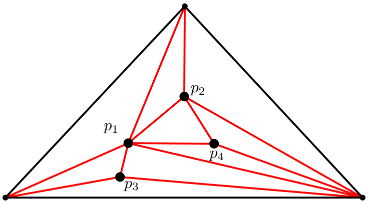
Our new method leads to:
$\rightarrow$ expected $O(n \log n)$ time complexity

Summary: A variant of backwards analysis

New variant suitable for:

- randomized incremental algorithms
- output may be **dependent** on the randomization order

Example: Triangulation algorithm
[Seidel 1993]



Looking for other applications!

Our new method leads to:
→ expected $O(n \log n)$ time complexity

Literature —— Thank you for your attention!

J. Boissonnat and M. Yvinec
Algorithmic Geometry.
Cambridge University Press, New York, NY, USA, 1998.

L. Chew
Building Voronoi diagrams for convex polygons in linear expected time.
Tech. Report, Dartmouth College, Hanover, 1985.

K. Junginger and E. Papadopoulou
Deletion in abstract Voronoi diagrams in expected linear time and related problems.
Discrete & Computational Geometry, 2023.

V. Levenshtein
On perfect codes in deletion and insertion metric.
Discrete Mathematics and Applications, 1992.

R. Seidel
Backwards analysis of randomized geometric algorithms.
New trends in discrete and computational geometry, 1993.