Optimizing Active Ranges for Consistent Dynamic Map Labeling

Ken Been[1] Martin Nöllenburg[2] Sheung-Hung Poon[3]
Alexander Wolff[4]

[1]Yeshiva University

[2]Universität Karlsruhe

[3]National Tsing Hua University

[4]TU Eindhoven

Outline

- Model
- Complexity
- Approximation
    - Top-to-bottom sweep algorithm
    - Level-based algorithm

Requirements in static map labeling

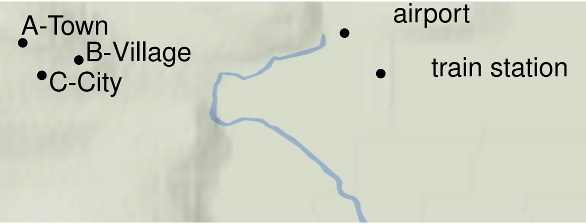
- non-overlapping labels

Requirements in static map labeling

- non-overlapping labels

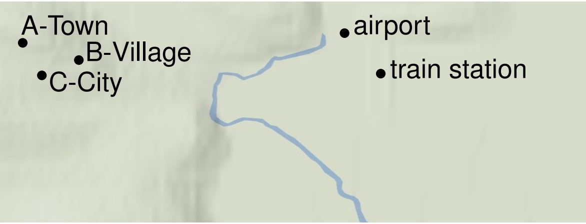
Requirements in static map labeling

- non-overlapping labels
- proximity of feature and label



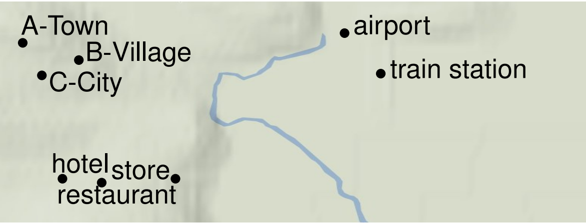
Requirements in static map labeling

- non-overlapping labels
- proximity of feature and label



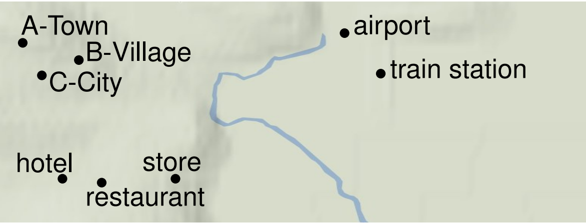
Requirements in static map labeling

- non-overlapping labels
- proximity of feature and label
- unambiguity



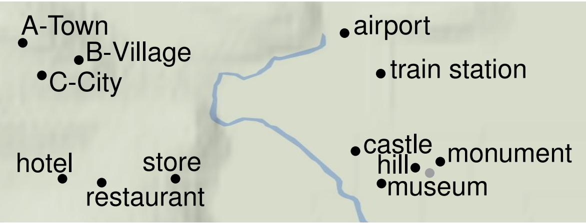
Requirements in static map labeling

- non-overlapping labels
- proximity of feature and label
- unambiguity



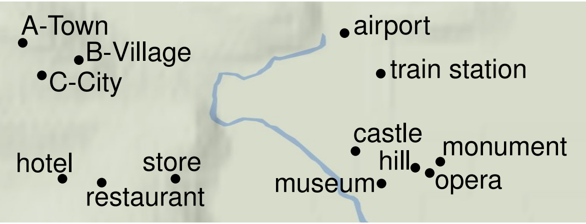
Requirements in static map labeling

- non-overlapping labels
- proximity of feature and label
- unambiguity
- maximize number of labeled features



Requirements in static map labeling

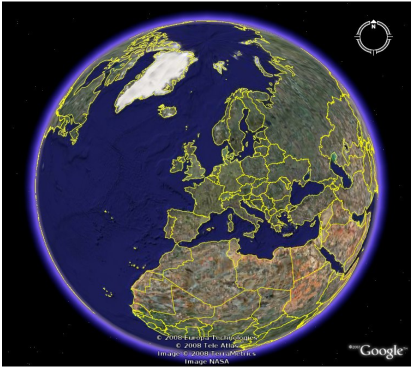
- non-overlapping labels
- proximity of feature and label
- unambiguity
- maximize number of labeled features



Requirements in dynamic map labeling

interactive maps add more requirements

- static map at *each scale*
  - non-overlapping labels
  - feature–label proximity
  - unambiguity
  - maximize label number
- during zooming & panning
  - no popping of labels
  - no jumping of labels
  - map independent of navigation history



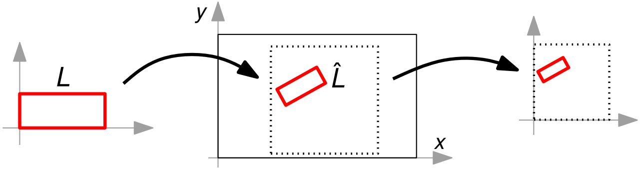
Static model

Static *selection*

Boolean function that selects subset of non-overlapping labels

Static *placement*

1. transform label L to world coordinates (by translation, rotation, dilation)
2. transform world coordinates to screen coordinates with dilation factor $1/s$ (define s as the *scale* of the map)



Our dynamic model ∼[Been, Daiches, Yap 2006]

Dynamic *selection*

Boolean function of scale selects each label L_i in at most one scale interval $[a_i, A_i]$, its active range

Our dynamic model ∼[Been, Daiches, Yap 2006]

Dynamic *selection*

Boolean function of scale selects each label L_i in at most one scale interval $[a_i, A_i]$, its active range $\rightarrow$ no popping

Our dynamic model ∼[Been, Daiches, Yap 2006]

Dynamic *selection*

Boolean function of scale selects each label L_i in at most one scale interval $[a_i, A_i]$, its active range → no popping

Dynamic *placement*

- static placement $\hat{L}^s$ for each scale s
- continuous with s
- transforms label L to *extended* world coordinates (x, y, s)
- $\hat{L}^s$ is cross section of extended world coordinates at scale s

Our dynamic model ∼[Been, Daiches, Yap 2006]

Dynamic *selection*

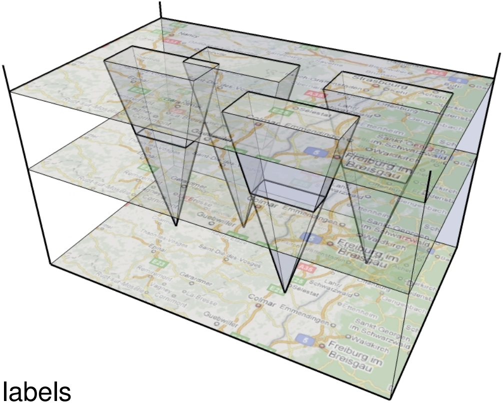
Boolean function of scale selects each label L_i in at most one scale interval $[a_i, A_i]$, its active range $\rightarrow$ no popping

Dynamic *placement*

- static placement $\hat{L}^s$ for each scale s
- continuous with $s \rightarrow$ no jumping
- transforms label L to *extended* world coordinates (x, y, s)
- $\hat{L}^s$ is cross section of extended world coordinates at scale s

Extended world coordinates

- scale as 3^{rd} dimension
- union of label shapes over scale: “extrusion”
- restriction to active range: “truncated extrusion”



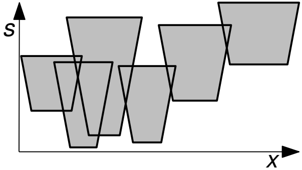
- here:
  - axis-aligned rectangular labels
  - invariant-point placement
  - proportional dilation

Active-range optimization

Problem

IN:
- labels $L_1, \ldots, L_n$ with dynamic placement,
- available ranges $[s_i, S_i]$ for $i = 1, \ldots, n$.

OUT: active ranges $[a_i, A_i] \subseteq [s_i, S_i]$ such that
- *total active range height* $H = \sum_i (A_i - a_i)$ is max,
- truncated extrusions do not overlap.



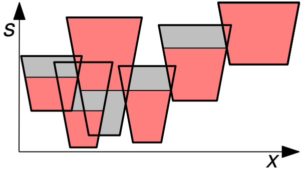
Active-range optimization

Problem

IN:
- labels $L_1, \ldots, L_n$ with dynamic placement,
- available ranges $[s_i, S_i]$ for $i = 1, \ldots, n$.

OUT: active ranges $[a_i, A_i] \subseteq [s_i, S_i]$ such that

- *total active range height* $H = \sum_i (A_i - a_i)$ is max,
- truncated extrusions do not overlap.



Active-range optimization

Problem

IN:
- labels $L_1, \ldots, L_n$ with dynamic placement,
- available ranges $[s_i, S_i]$ for $i = 1, \ldots, n$.

OUT: active ranges $[a_i, A_i] \subseteq [s_i, S_i]$ such that

- *total active range height* $H = \sum_i (A_i - a_i)$ is max,
- truncated extrusions do not overlap.

Simple problem

All available ranges are $[0, S_{\max}]$.

Outline

- Model
- Complexity
- Approximation
  - Top-to-bottom sweep algorithm
  - Level-based algorithm

NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

Sketch of proof

By reduction from PLANAR 3SAT.

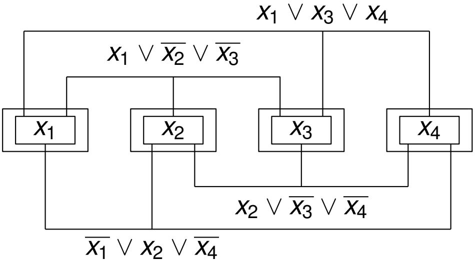
NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

Sketch of proof

By reduction from PLANAR 3SAT.



planar 3SAT formula φ

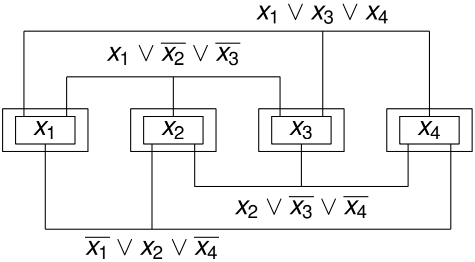
NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

Sketch of proof

By reduction from PLANAR 3SAT.



planar 3SAT formula φ

$\downarrow$

(set of labels, int k) s.t.
$H \geq k \Leftrightarrow \varphi$ satisfiable

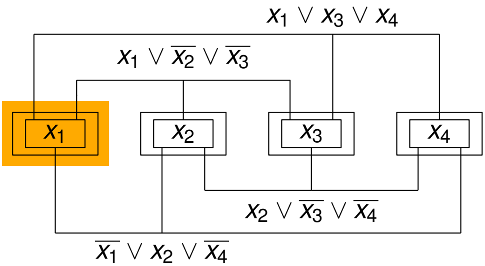
NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

Sketch of proof

By reduction from PLANAR 3SAT.



planar 3SAT formula φ

↓

(set of labels, int k) s.t.
$H \geq k \Leftrightarrow \varphi$ satisfiable

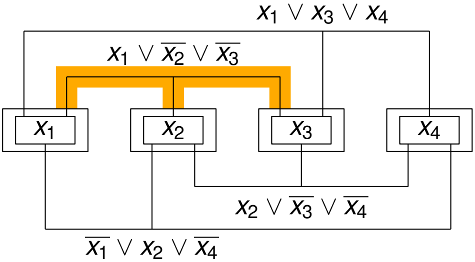
NP-hardness

Theorem

The active-range optimization problem is NP-hard –
even the simple variant.

Sketch of proof

By reduction from PLANAR 3SAT.

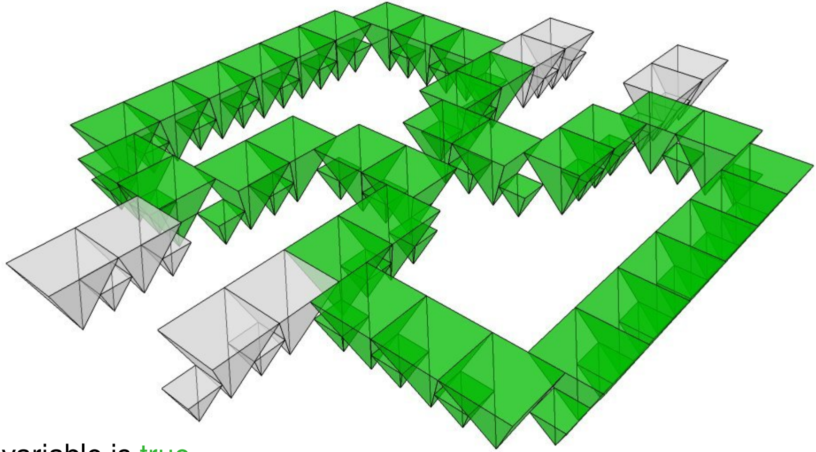


planar 3SAT formula φ

$\downarrow$

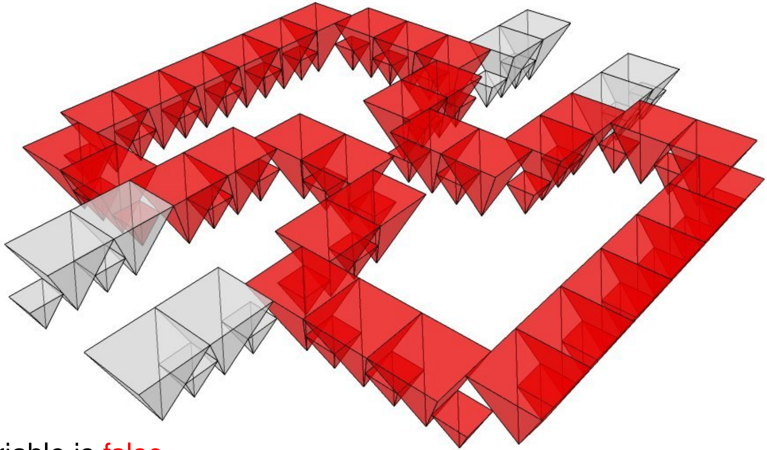
(set of labels, int k) s.t.
$H \geq k \;\Leftrightarrow\; \varphi$ satisfiable

Variable gadget



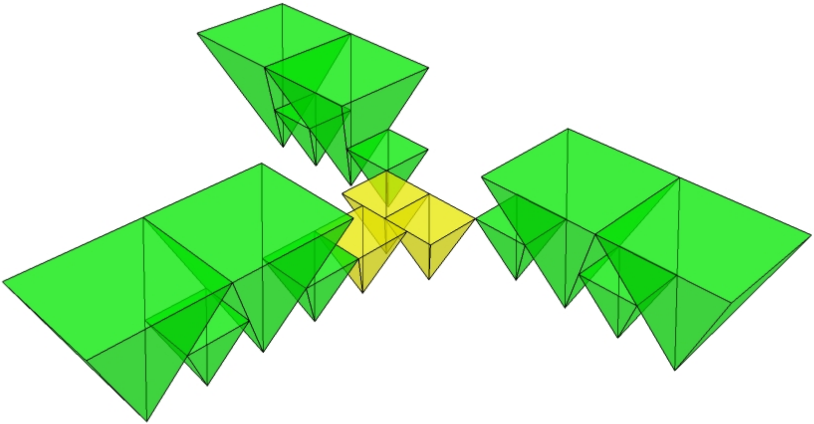
variable is true

Variable gadget



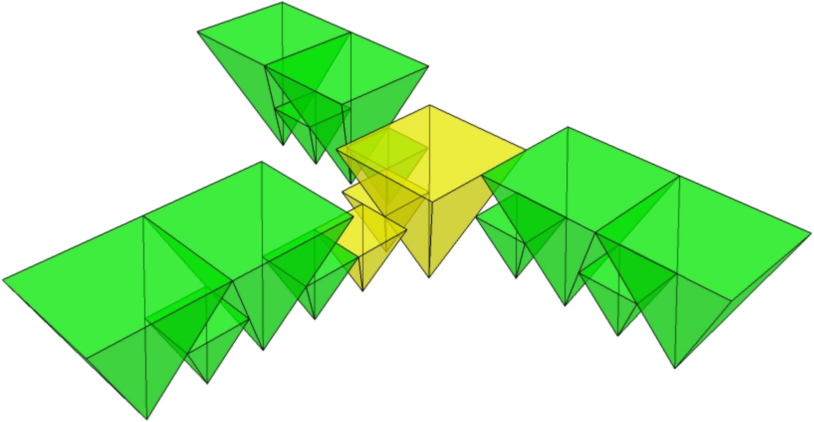
variable is false

Clause gadget



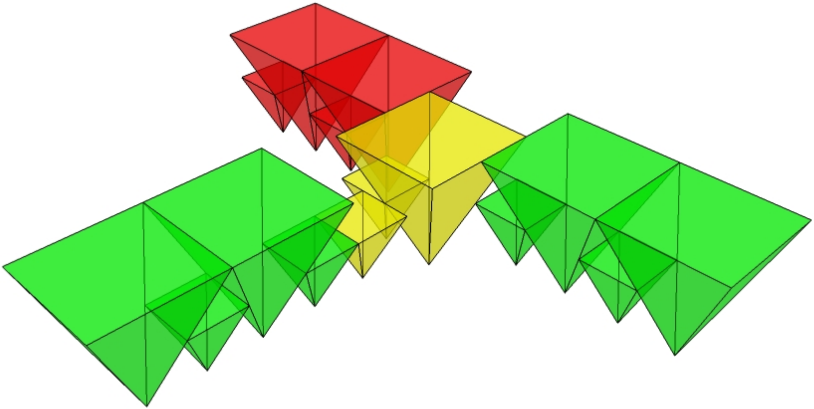
3 literals are true

Clause gadget



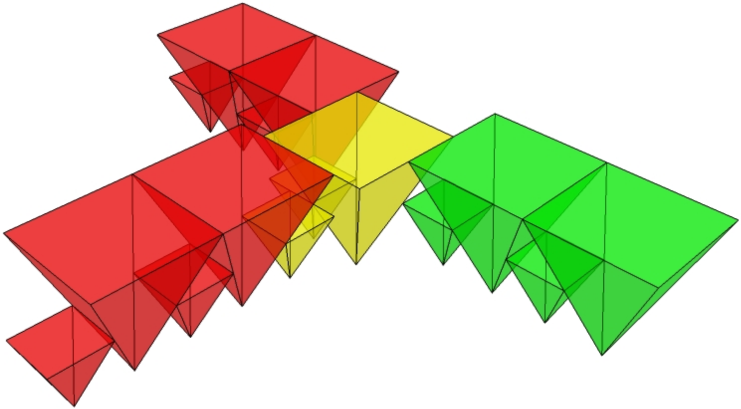
3 literals are true → contribution to H: $2 \cdot S_{\max}$

Clause gadget



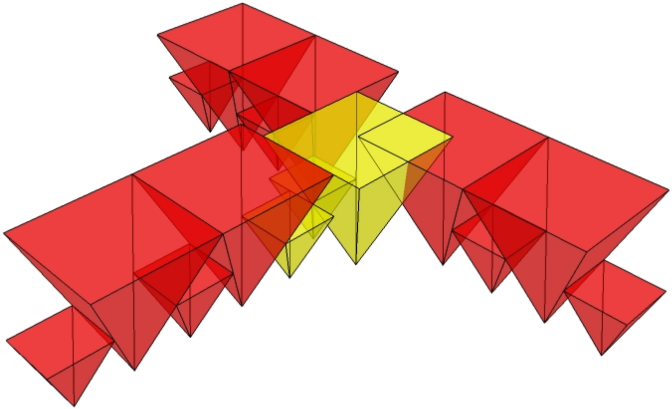
2 literals are true → contribution to H: $2 \cdot S_{\max}$

Clause gadget



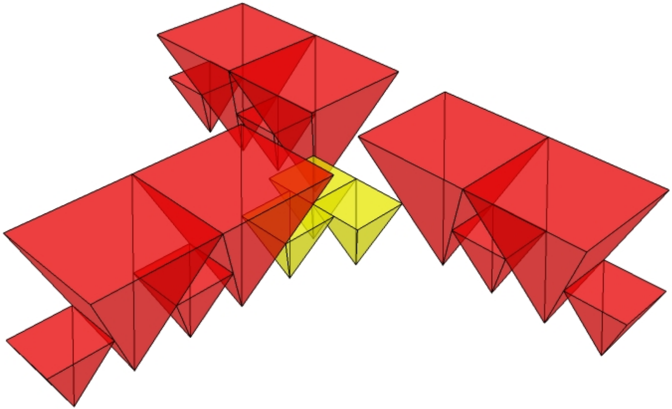
1 literal is true → contribution to H: $2 \cdot S_{\max}$

Clause gadget



0 literals are true → contribution to H: ?

Clause gadget



0 literals are true → contribution to H: $1.5 \cdot S_{\max}$

Outline

- Model
- Complexity
- Approximation
  - Top-to-bottom sweep algorithm
  - Level-based algorithm

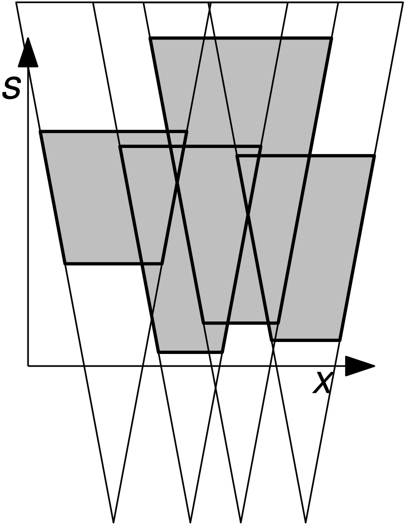
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



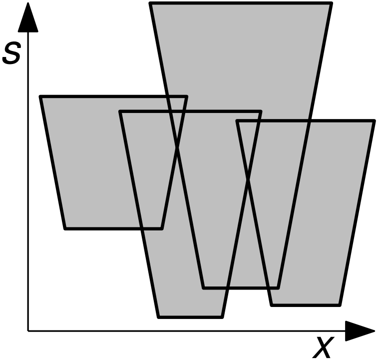
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



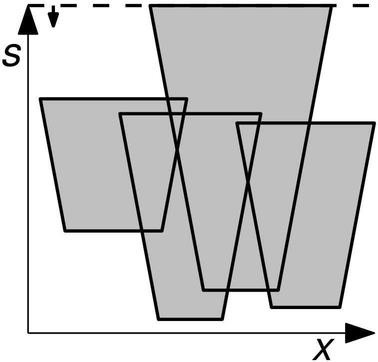
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



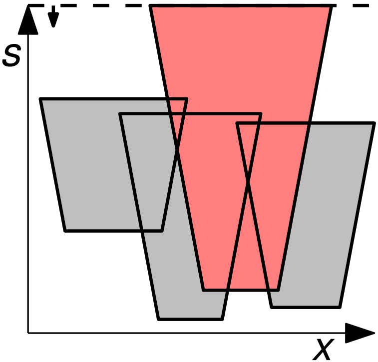
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



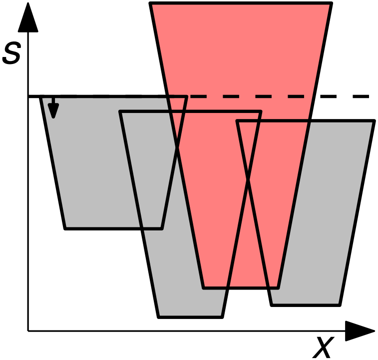
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



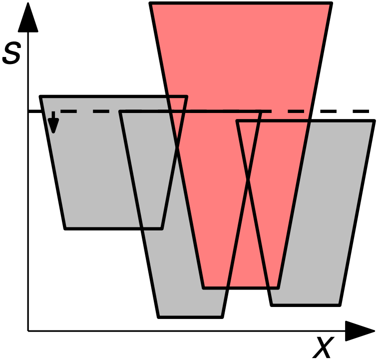
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



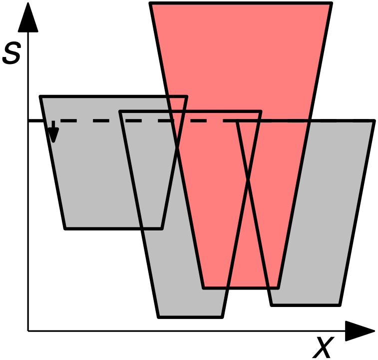
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



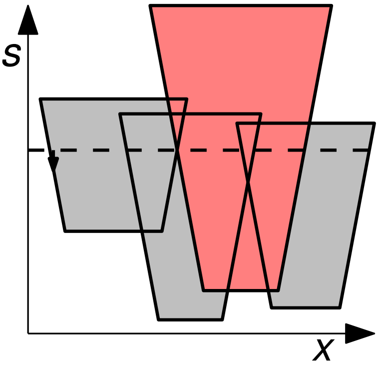
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s, **then** set $[a_i, A_i] = [s_i, s]$.



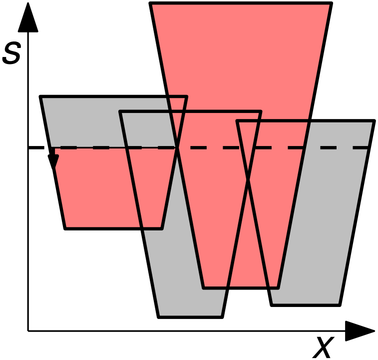
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



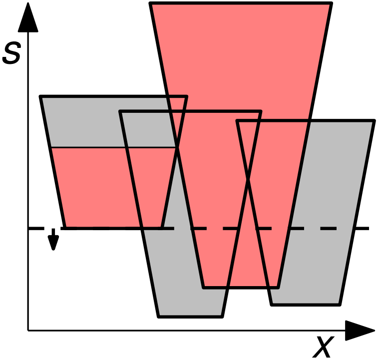
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



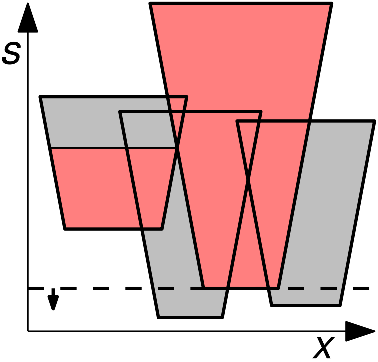
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



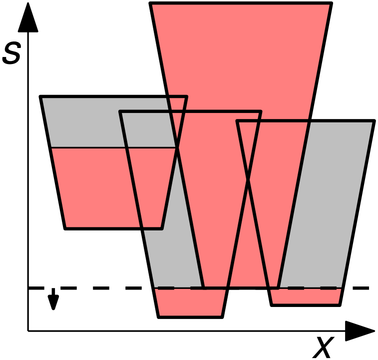
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



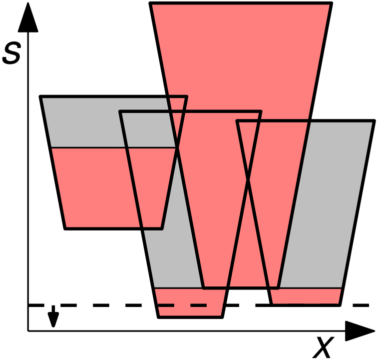
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



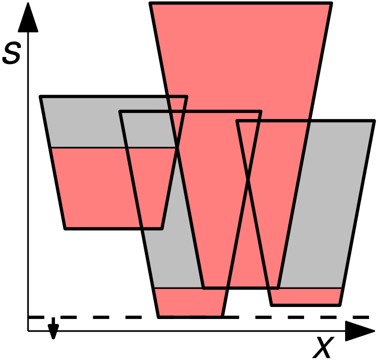
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s, **then** set $[a_i, A_i] = [s_i, s]$.



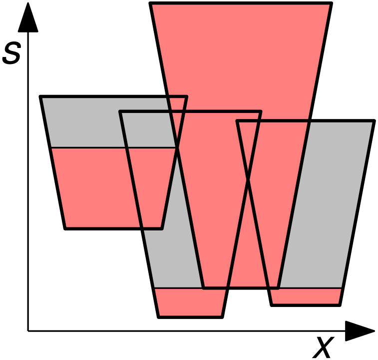
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



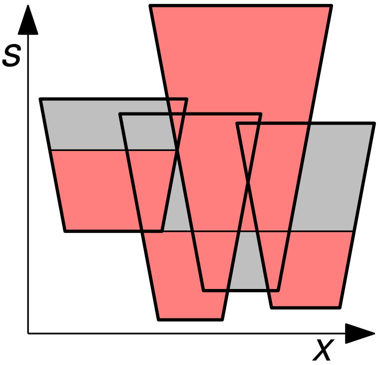
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s,
then set $[a_i, A_i] = [s_i, s]$.



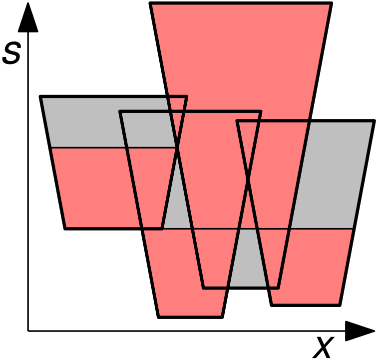
Top-to-bottom fill-down sweep

Algorithm

1. sweep from top to bottom
2. at each event try to fill available but inactive extru.

Subroutine try to fill extrusion E_i

If E_i doesn't intersect any active extrusion at current scale s, **then** set $[a_i, A_i] = [s_i, s]$.



Theorem

For segments of congruent triangles, this is a $\frac{1}{2}$-approximation.

Approximation

Blocking Lemma

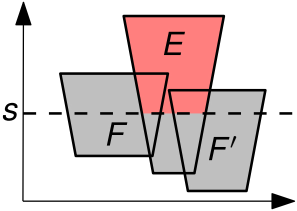
If an extrusion E never blocks more than c pairwise independent extrusions,
then our algorithm computes a $1/c$-approximation.

Approximation

Blocking Lemma

If an extrusion E never blocks more than c pairwise independent extrusions,
then our algorithm computes a $1/c$-approximation.

- E *blocks* F at scale s if E is active and overlaps F at s.

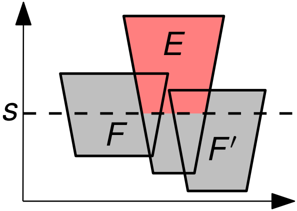


Approximation

Blocking Lemma

If an extrusion E never blocks more than c pairwise independent extrusions,
then our algorithm computes a $1/c$-approximation.

- E *blocks* F at scale s if E is active and overlaps F at s.
- F and F' are *independent* at s if they do not overlap at s.



Approximation

Blocking Lemma

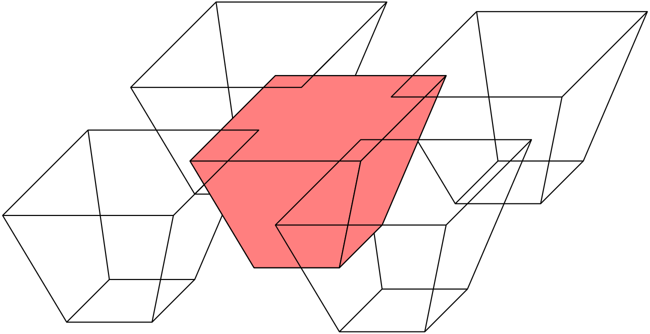
If an extrusion E never blocks
more than c pairwise independent extrusions,
then our algorithm computes a $1/c$-approximation.

- E *blocks* F at scale s if E is active and overlaps F at s.
- F and F' are *independent* at s if they do not overlap at s.

Proof

Integrate **if**-condition over all scales $\Rightarrow$ **then**-statement.

Example: frustal segments of congruent cones

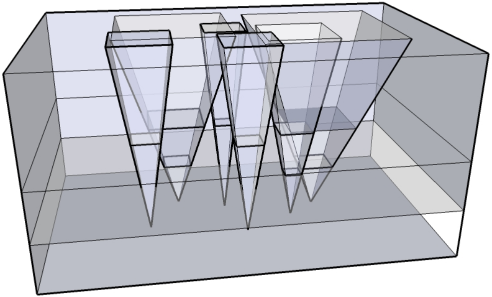


- each label at each scale has the same shape
- blocking lemma $\Rightarrow$ sweep yields $1/4$-approximation

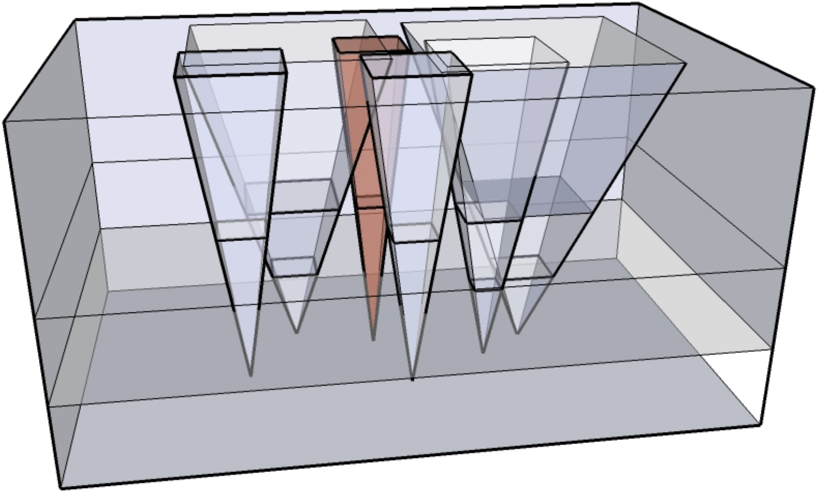
Level-based algorithm (sketch)

Setting

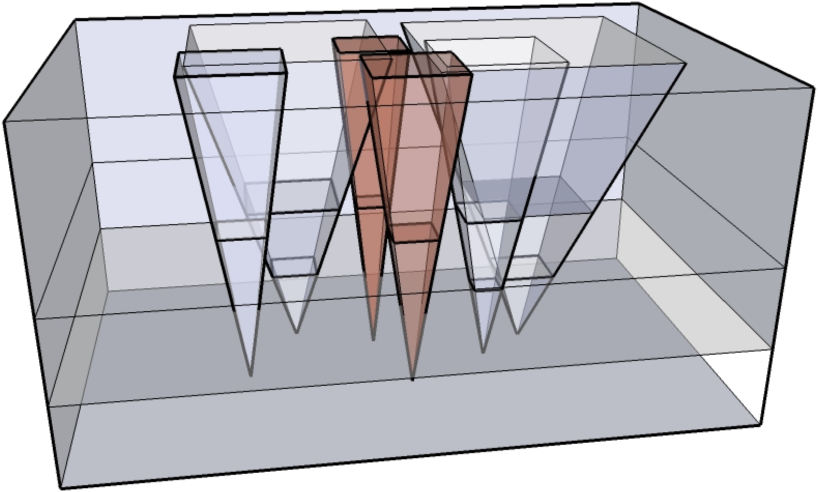
- n arbitrary square cones
- available ranges $[0, S_{\max}]$
- use horizontal planes at scales $S_{\max}/2^i$ for $i = 0, \dots, \log n$



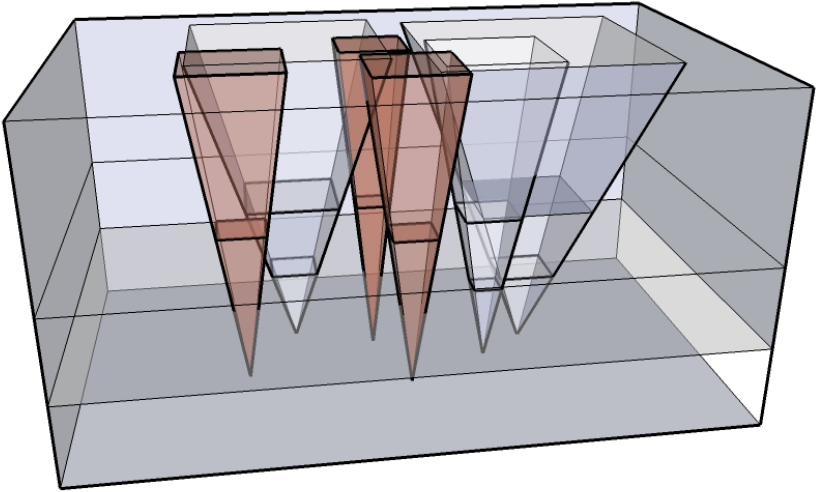
Level-based algorithm



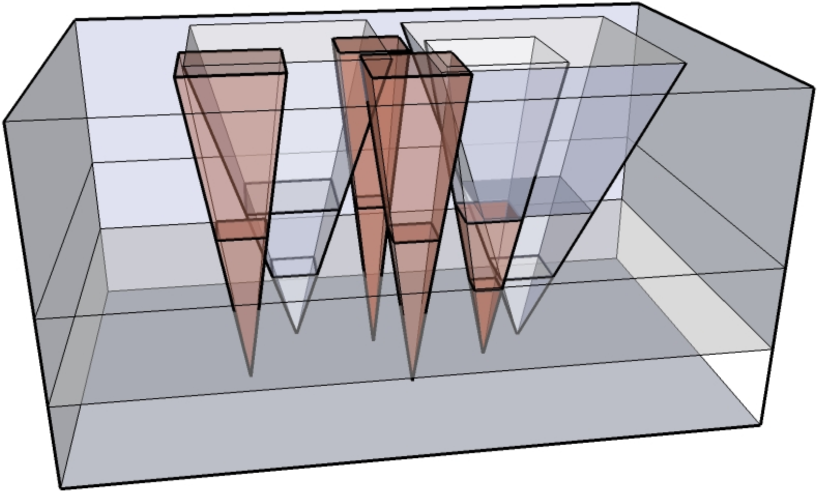
Level-based algorithm



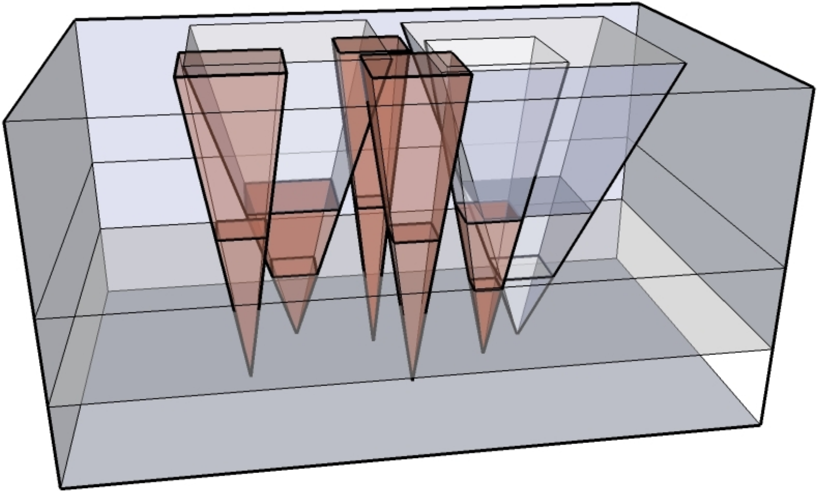
Level-based algorithm



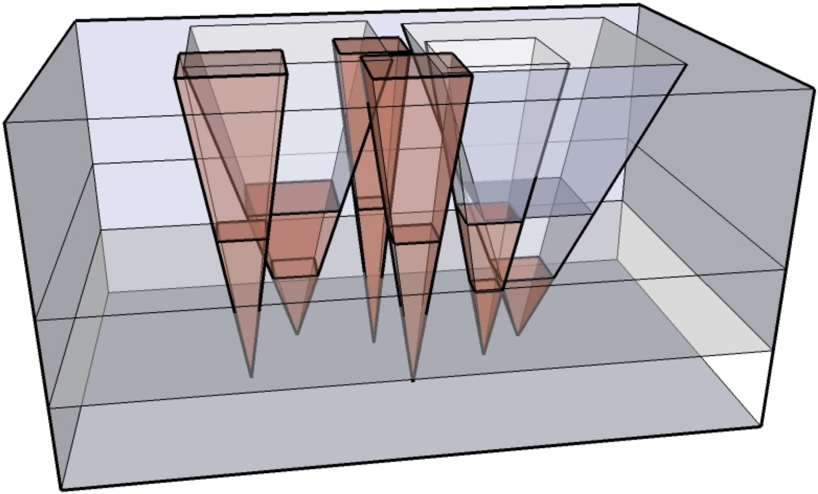
Level-based algorithm



Level-based algorithm



Level-based algorithm



Summary

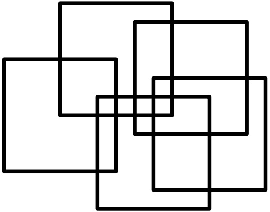
extrusions	approx.	running time
frustal segm. of congr. cones	$1/4$	$O((n+k)\log^2 n)$
congruent frusta	$1/(4W)$	$O(n^4)$
arbitrary square cones (simple)	$1/24$	$O(n\log^3 n)$
congruent square cones (simple)	$1/8$	$O(n\log^3 n)$

Summary

extrusions	approx.	running time
frustal segm. of congr. cones	$1/4$	$O((n+k)\log^2 n)$
congruent frusta	$1/(4W)$	$O(n^4)$
arbitrary square cones (simple)	$1/24$	$O(n \log^3 n)$
congruent square cones (simple)	$1/8$	$O(n \log^3 n)$

Open Problems

- better approximations, also in 1d
- (non-) existence of a PTAS
- more realistic extrusion shapes

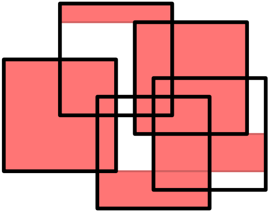


Summary

extrusions	approx.	running time
frustal segm. of congr. cones	$1/4$	$O((n+k)\log^2 n)$
congruent frusta	$1/(4W)$	$O(n^4)$
arbitrary square cones (simple)	$1/24$	$O(n \log^3 n)$
congruent square cones (simple)	$1/8$	$O(n \log^3 n)$

Open Problems

- better approximations, also in 1d
- (non-) existence of a PTAS
- more realistic extrusion shapes



Summary

extrusions	approx.	running time
frustal segm. of congr. cones	$1/4$	$O((n+k)\log^2 n)$
congruent frusta	$1/(4W)$	$O(n^4)$
arbitrary square cones (simple)	$1/24$	$O(n\log^3 n)$
congruent square cones (simple)	$1/8$	$O(n\log^3 n)$

Open Problems

- better approximations, also in 1d
- (non-) existence of a PTAS
- more realistic extrusion shapes