Mining Convex Polygon Patterns

Aimene Belfodil[1,2], Sergei O. Kuznetsov[3],
Céline Robardet[1], Mehdi Kaytoue[1]

[1]Univ Lyon, INSA Lyon, CNRS, LIRIS UMR 5205, F-69621, Lyon, France
[2]Mobile Devices Ingenierie, 100 Avenue Stalingrad, 94800, Villejuif, France
[3]National Research Higher School of Economics, Moscow, Russia

Introduction (1)

Pattern mining is an important task in AI for eliciting hypotheses from the data.

Introduction (1)

Pattern mining is an important task in AI for eliciting hypotheses from the data.

$\mathcal{G}$	i_1	i_2	i_3	*class*
a	X		X	+
b		X	X	+
c		X	X	−
d	X	X	X	+
e	X		X	−
f		X		−

Pattern mining is an important task in AI for eliciting **hypotheses** from the data.

$\mathcal{G}$	i_1	i_2	i_3	*class*
a	X		X	+
b		X	X	+
c		X	X	−
d	X	X	X	+
e	X		X	−
f		X		−

Pattern intent. Having items i_2 **AND** i_3

Pattern mining is an important task in AI for eliciting hypotheses from the data.

$\mathcal{G}$	i_1	i_2	i_3	class
a	X		X	+
b		X	X	+
c		X	X	−
d	X	X	X	+
e	X		X	−
f		X		−

Pattern intent. Having items i_2 **AND** i_3

Pattern extent. $\{b, c, d\}$ (Support $= \mathbf{3}$)

Pattern mining is an important task in AI for eliciting **hypotheses** from the data.

$\mathcal{G}$	i_1	i_2	i_3	*class*
a	X		X	+
b		X	X	+
c		X	X	−
d	X	X	X	+
e	X		X	−
f		X		−

Pattern intent. Having items i_2 **AND** i_3

Pattern extent. $\{b, c, d\}$ (Support = 3)

Pattern quality. Relative Accuracy* $= \frac{2}{3} - \frac{3}{6} = \frac{1}{6}$

Relative Accuracy*: Difference between the proportion of the class under the hypothesis and its proportion in the whole dataset.

Pattern mining is an important task in AI for eliciting hypotheses from the data.

$\mathcal{G}$	i_1	i_2	i_3	class
a	X		X	+
b		X	X	+
c		X	X	−
d	X	X	X	+
e	X		X	−
f		X		−

Pattern intent. Having items i_2 AND i_3

Pattern extent. $\{b, c, d\}$ (Support $= 3$)

Pattern quality. Relative Accuracy* $= \frac{2}{3} - \frac{3}{6} = \frac{1}{6}$

Hypothesis.

When objects have items i_2 and i_3
The + class is fostered

Relative Accuracy*: Difference between the proportion of the class under the hypothesis and its proportion in the whole dataset.

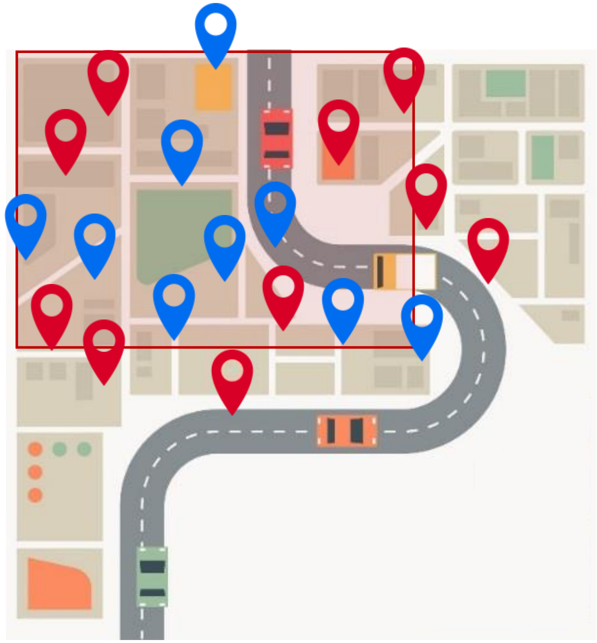
Introduction (2)

In this **paper**, we are interested in mining patterns that consider spatial attribute*.

Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

In this **paper**, we are interested in mining patterns that consider spatial attribute*.

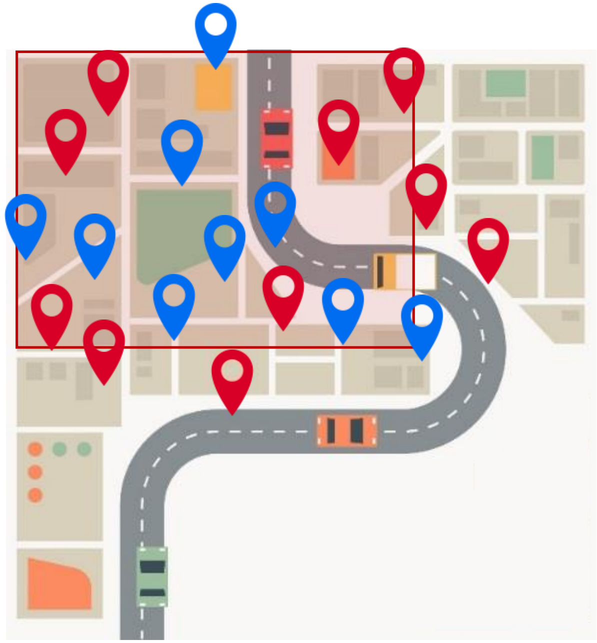
- Numerical attributes are often considered independently (next slide).



Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

In this **paper**, we are interested in mining patterns that consider spatial attribute*.

- Numerical attributes are often considered independently (next slide).
- Consequently, only rectangular shapes can be searched for.

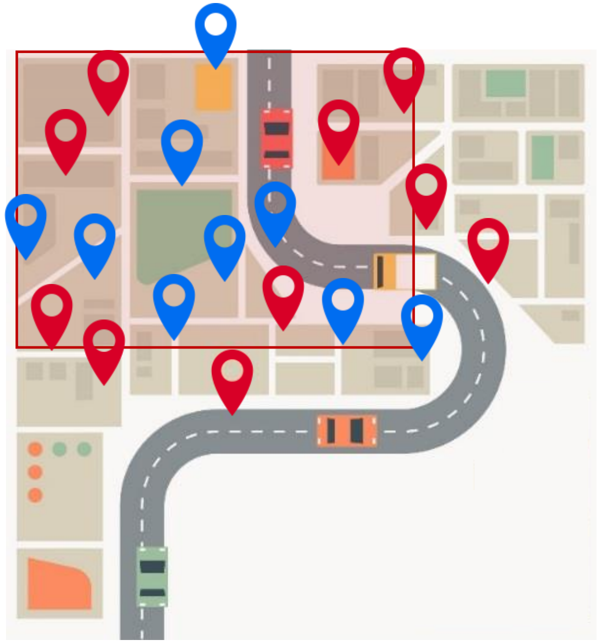


Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

Introduction (2)

In this **paper**, we are interested in mining patterns that consider **spatial attribute***.

- Numerical attributes are often considered independently (next slide).
- Consequently, only rectangular shapes can be searched for.
- Such arbitrary forms are not able to capture interesting regions.

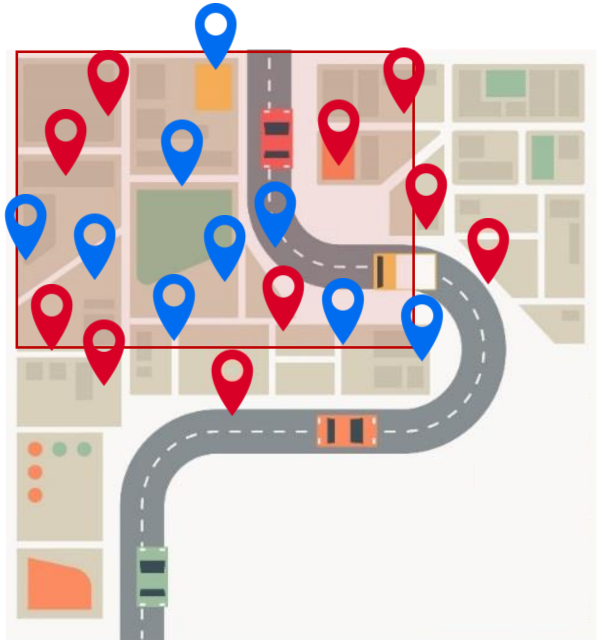


Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

Introduction (2)

In this **paper**, we are interested in mining patterns that consider **spatial attribute***.

- Numerical attributes are often considered independently (next slide).
- Consequently, only rectangular shapes can be searched for.
- Such arbitrary forms are not able to capture interesting regions.
- We need to have more expressive patterns without overfitting the data.

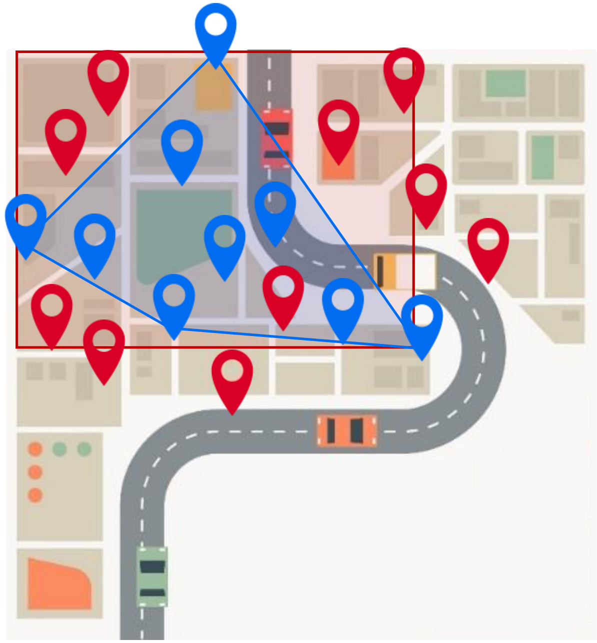


Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

Introduction (2)

In this **paper**, we are interested in mining patterns that consider **spatial attribute***.

- Numerical attributes are often considered independently (next slide).
- Consequently, only rectangular shapes can be searched for.
- Such arbitrary forms are not able to capture interesting regions.
- We need to have more expressive patterns without overfitting the data.
- A good trade-off is the use of convex polygon patterns.

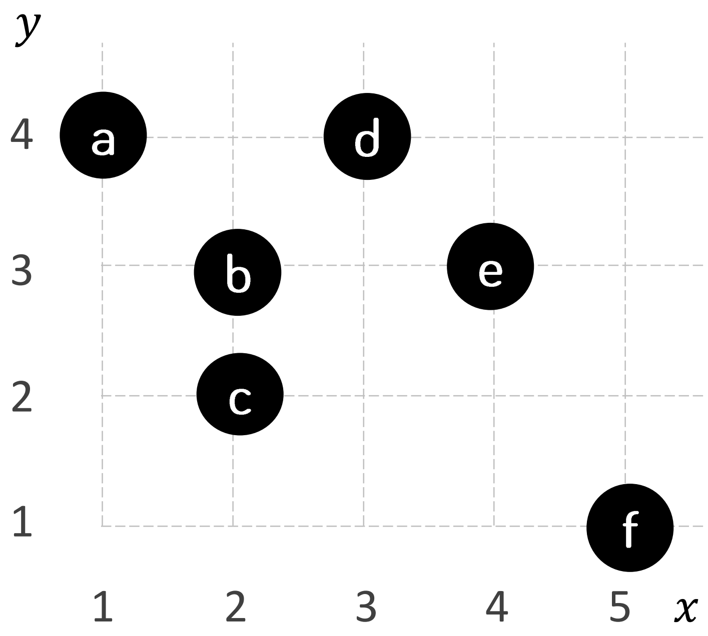


Spatial attribute*: a composite-attribute of two distinct numerical attributes (e.g. geo-coordinates).

Interval Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

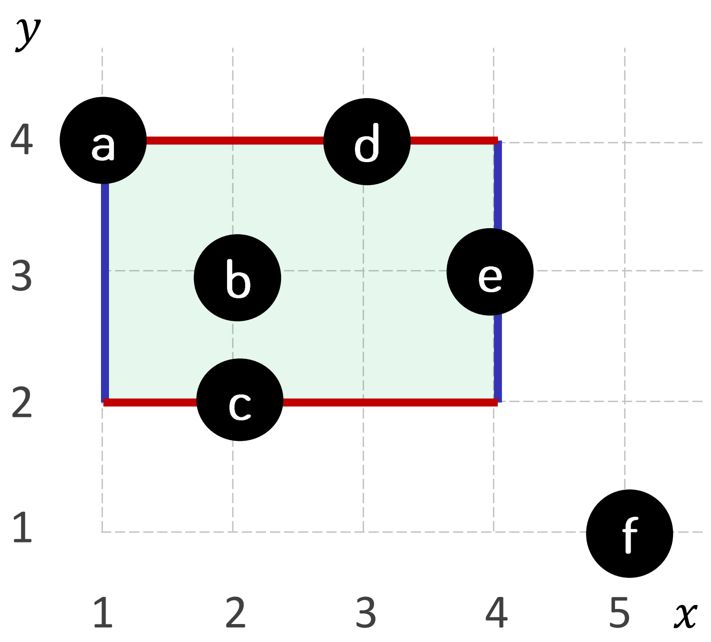
Two numerical attributes x and y



Interval Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Two numerical attributes x and y



Intent. $1 \leq x \leq 4$ ***AND*** $2 \leq y \leq 4$.

Extent. $\{a, b, c, d, e\}$

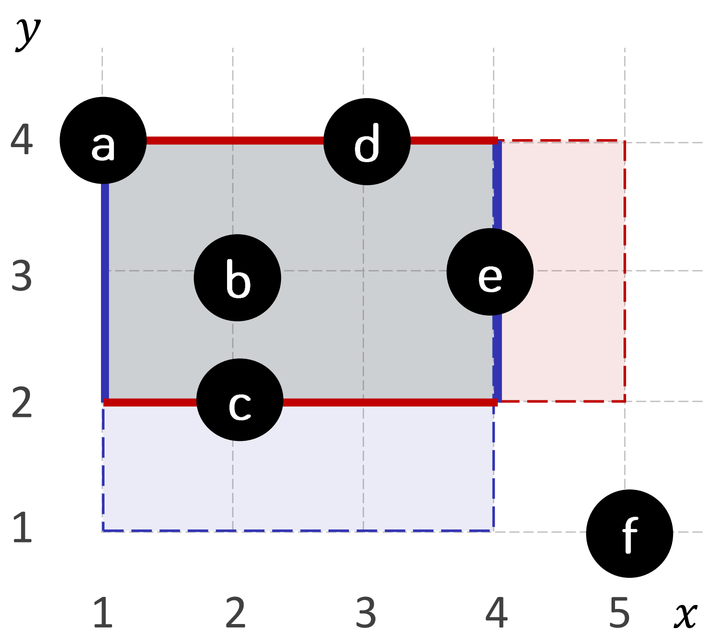
Interval Pattern Language [Kaytoue, M. et al. (IJCAI, 2011)]

- An intent is a rectangle (interval restriction over each attribute).

Interval Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Two numerical attributes x and y



Intent. $1 \leq x \leq 4$ ***AND*** $2 \leq y \leq 4$.

Extent. $\{a, b, c, d, e\}$

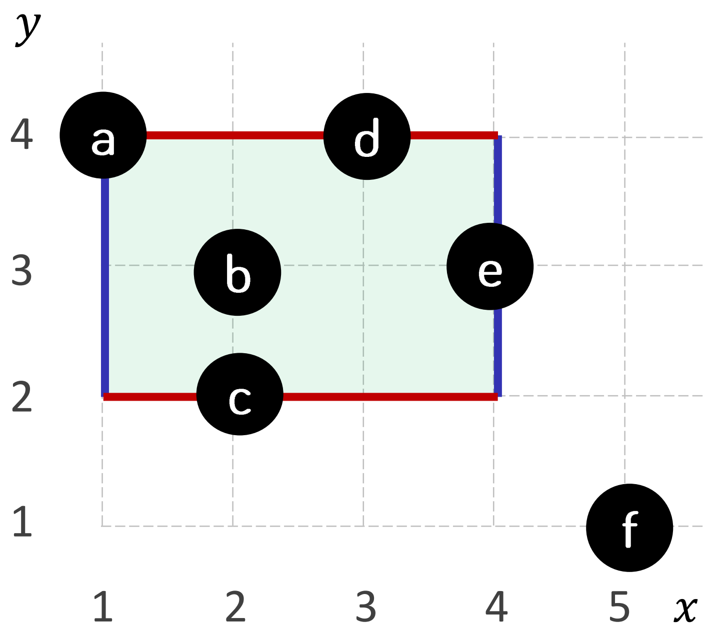
Interval Pattern Language
[Kaytoue, M. et al. (IJCAI, 2011)]

- An intent is a rectangle (interval restriction over each attribute).
- Same set of objects can be described by different intents.

Interval Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Two numerical attributes x and y



Intent. $1 \leq x \leq 4$ ***AND*** $2 \leq y \leq 4$.

Extent. $\{a, b, c, d, e\}$

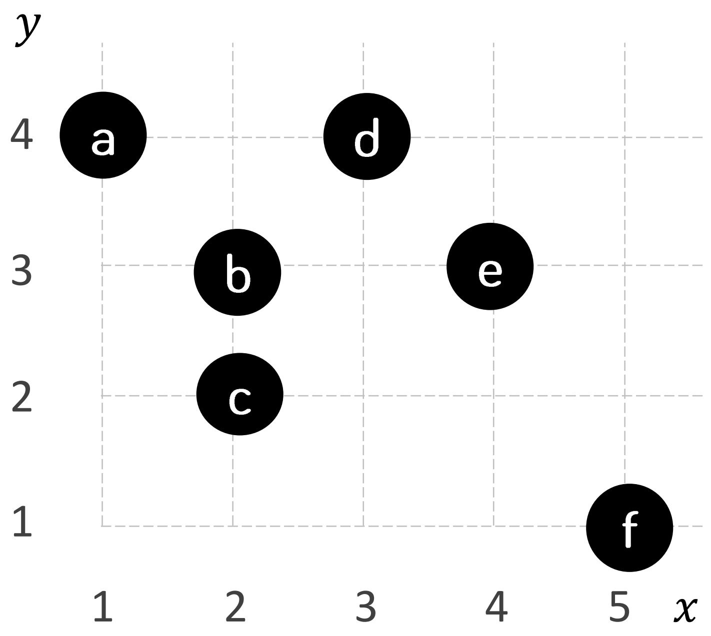
Interval Pattern Language [Kaytoue, M. et al. (IJCAI, 2011)]

- An intent is a rectangle (interval restriction over each attribute).
- Same set of objects can be described by different intents.
- We are interested in the most restrictive ones (i.e. tightest ones, closed ones).

Convex Polygon Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

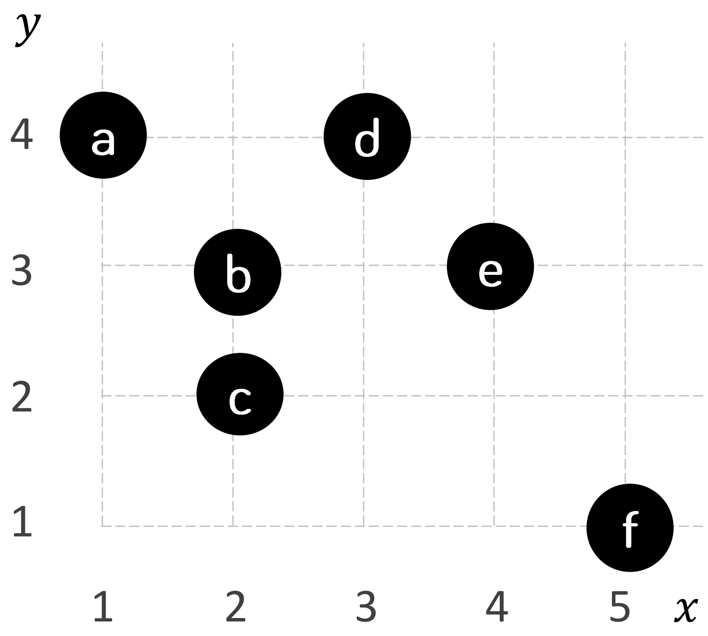
Spatial attribute
(x, y)



Convex Polygon Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Spatial attribute
(x, y)

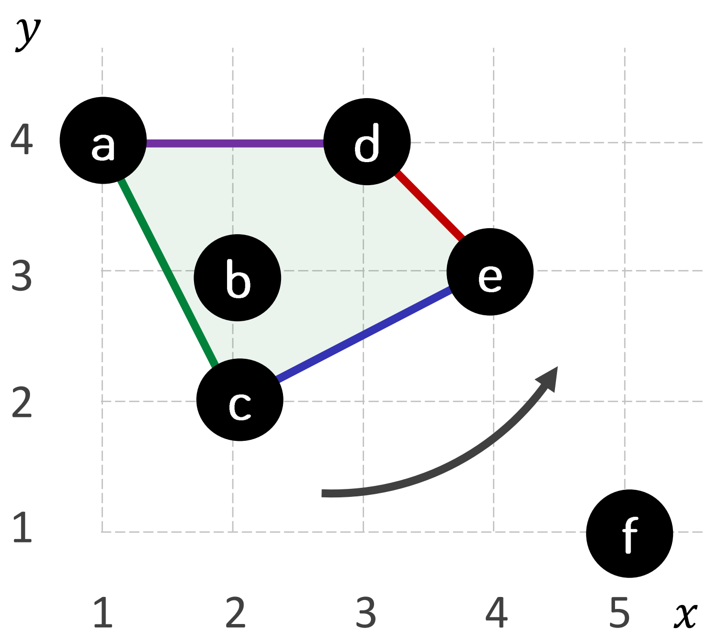


The Proposed Pattern Language.

Convex Polygon Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Spatial attribute
(x, y)



Intent.

$$y \leq 4 \quad AND$$
$$x + y \leq 7 \quad AND$$
$$x - 2y \leq -2 \quad AND$$
$$2x + y \geq 6 \quad .$$

(H-representation)

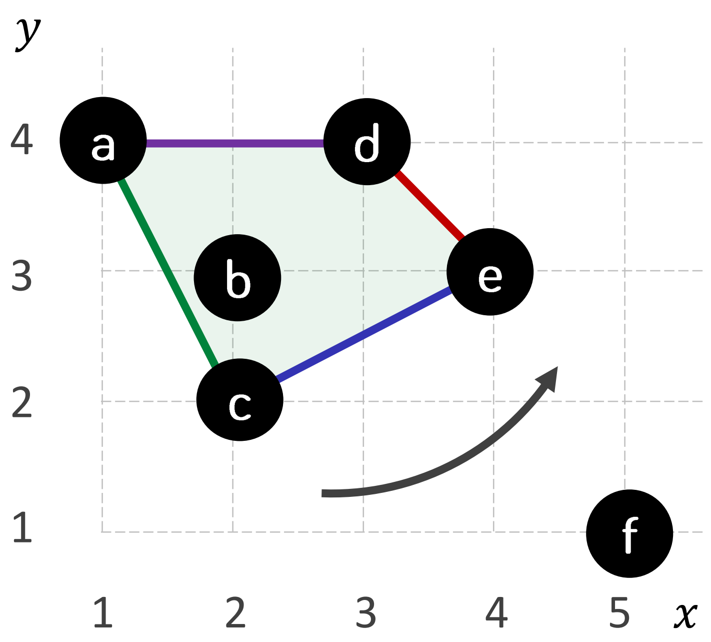
The Proposed Pattern Language.

- An intent is a convex polygon (conjunction of linear inequalities).

Convex Polygon Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Spatial attribute (x, y)



Intent.
$$y \leq 4 \quad \textbf{AND}$$
$$x + y \leq 7 \quad \textbf{AND}$$
$$x - 2y \leq -2 \quad \textbf{AND}$$
$$2x + y \geq 6 \quad .$$

(H-representation)

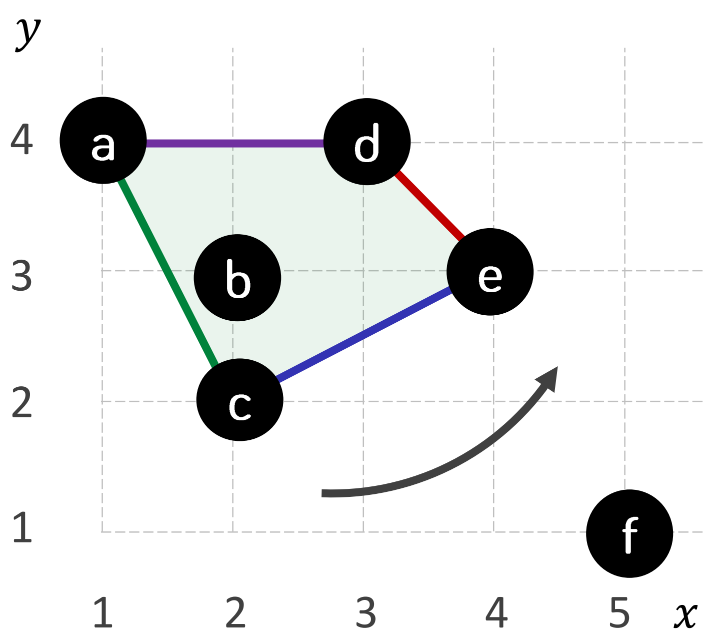
The Proposed Pattern Language.

- An intent is a convex polygon (conjunction of linear inequalities).
- The most restrictive description of a set of objects is its convex hull.

Convex Polygon Pattern

$\mathcal{G}$	x	y
a	1	4
b	2	2
c	2	3
d	3	4
e	4	3
f	5	1

Spatial attribute
(x, y)



Intent.

$$y \leq 4 \quad AND$$
$$x + y \leq 7 \quad AND$$
$$x - 2y \leq -2 \quad AND$$
$$2x + y \geq 6 \quad .$$

(H-representation)

The Proposed Pattern Language.

- An intent is a convex polygon (conjunction of linear inequalities).
- The most restrictive description of a set of objects is its convex hull.
- We will represent the intent as an ordered sequence of extreme points $[a, c, e, d]$ (V-representation).

Problem Definition

Let $\mathcal{G}$ be a **dataset** with one **spatial attribute**. Enumerate all possible convex hulls that can be built using point subsets of $\mathcal{G}$ without redundancy respecting eventually a set of user-specified constraints.

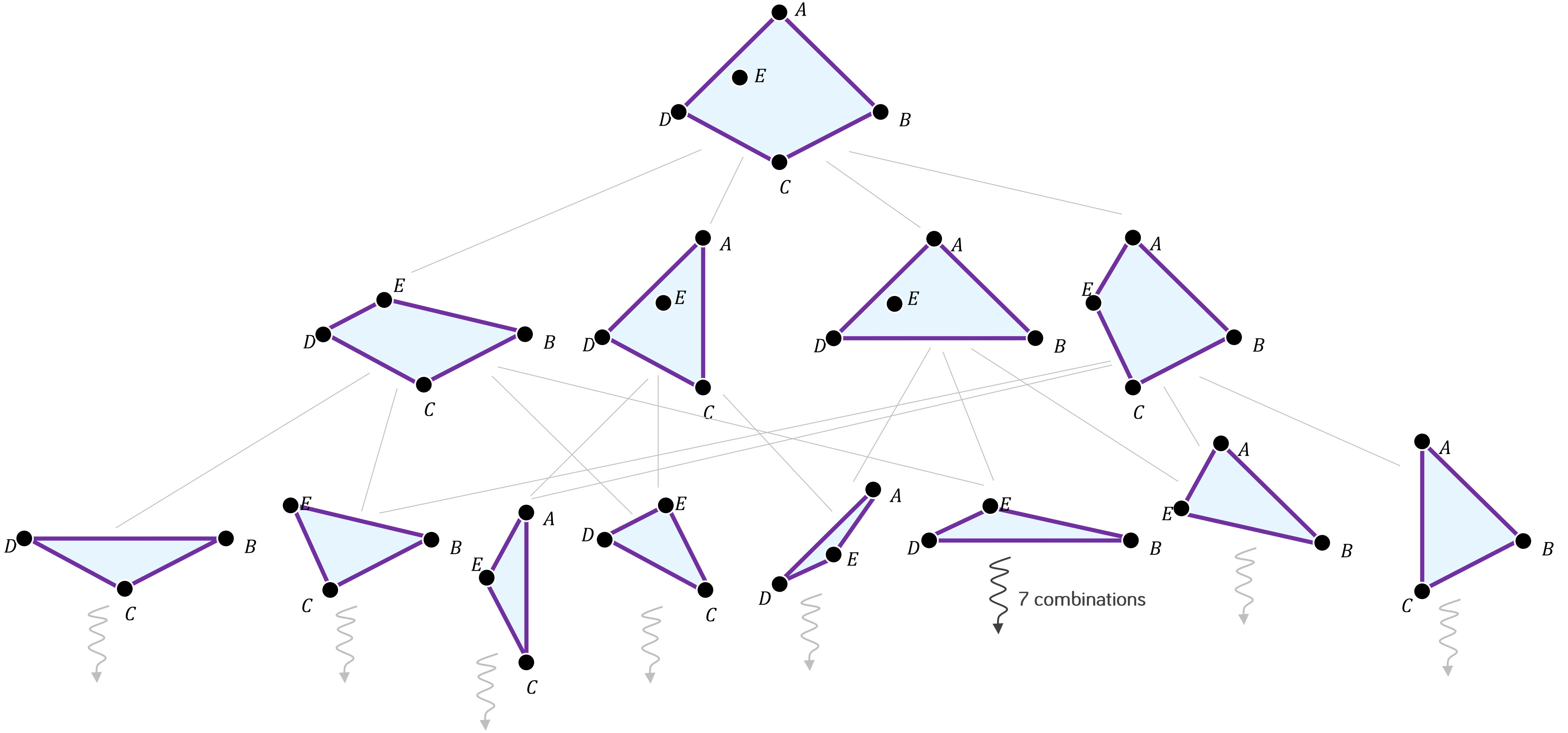
Problem Definition

Let $\mathcal{G}$ be a **dataset** with one **spatial attribute**. Enumerate all possible **convex hulls** that can be built using point subsets of $\mathcal{G}$ without redundancy respecting eventually a set of user-specified constraints

1. Algorithms
2. Experiments
3. Perspective

1. Algorithms
2. Experiments
3. Perspective

Search space – convex polygons lattice

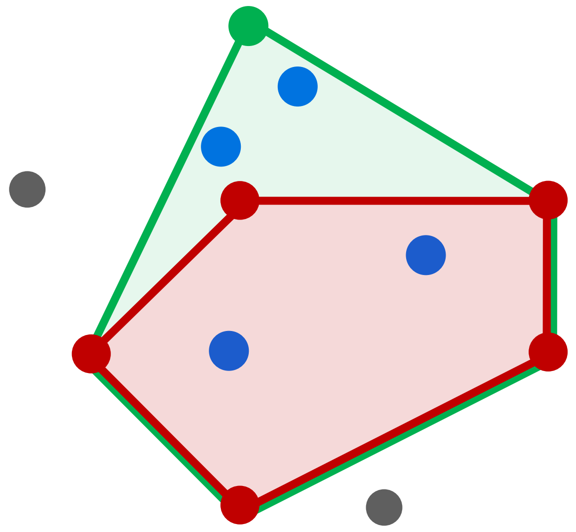


Enumeration Algorithms - Summary

Enumeration Algorithms - Summary

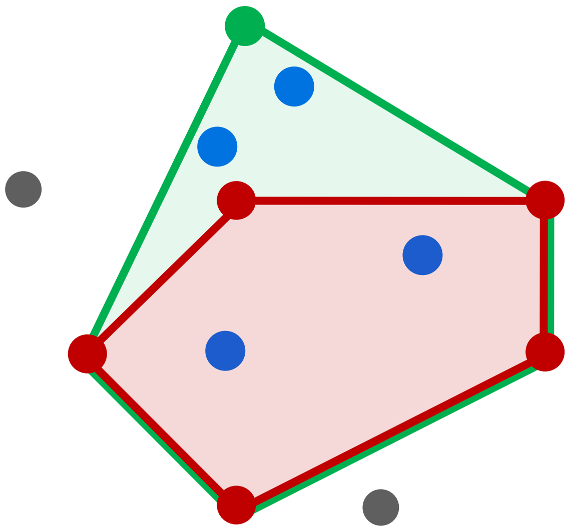
CloseByOne

- Bottom-up enumeration.
- Compute convex hulls.
- Based on canonicity test to ensure non-redundancy.



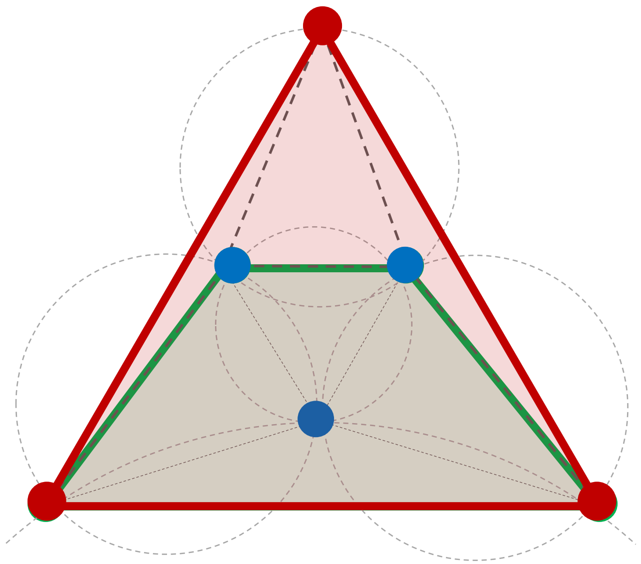
CloseByOne

- Bottom-up enumeration.
- Compute convex hulls.
- Based on canonicity test to ensure non-redundancy.



DT-Based

- Top-down enumeration.
- Based on Delaunay Triangulation.
- One-visit algorithm.



Enumeration Algorithms - Summary

CloseByOne

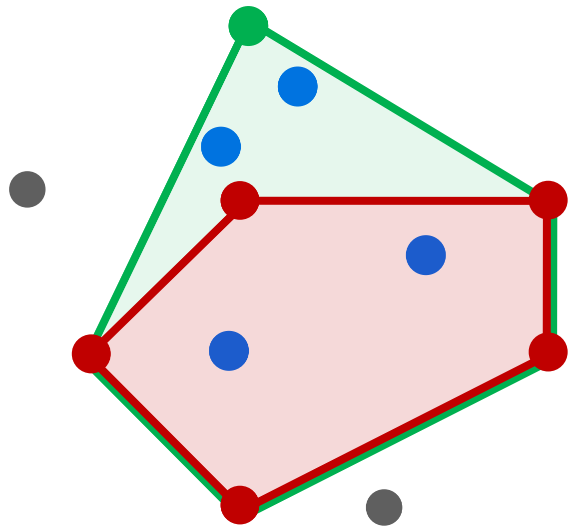
- Bottom-up enumeration.
- Compute convex hulls.
- Based on canonicity test to ensure non-redundancy.

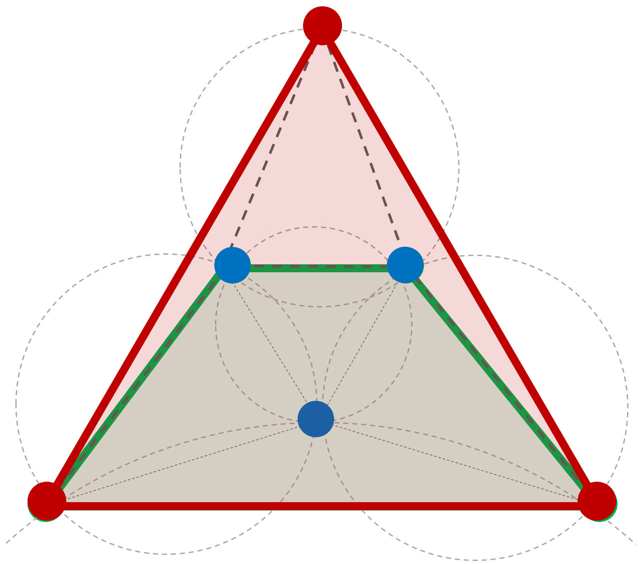
DT-Based

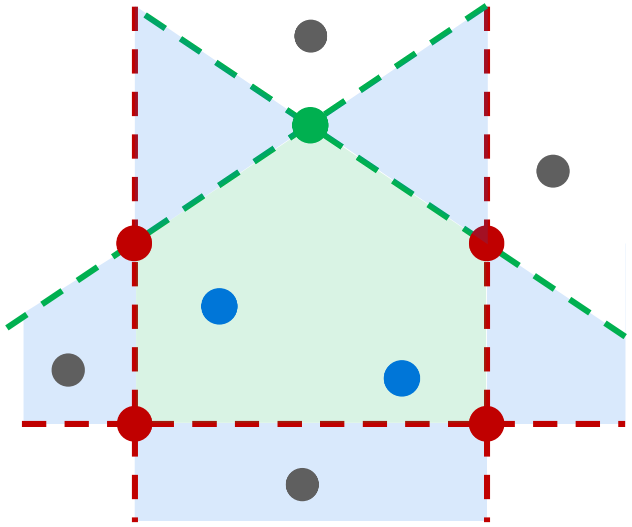
- Top-down enumeration.
- Based on Delaunay Triangulation.
- One-visit algorithm.

Vision-Based

- Bottom-up enumeration.
- Number of vertices increases by one after each extension.
- One-visit algorithm.

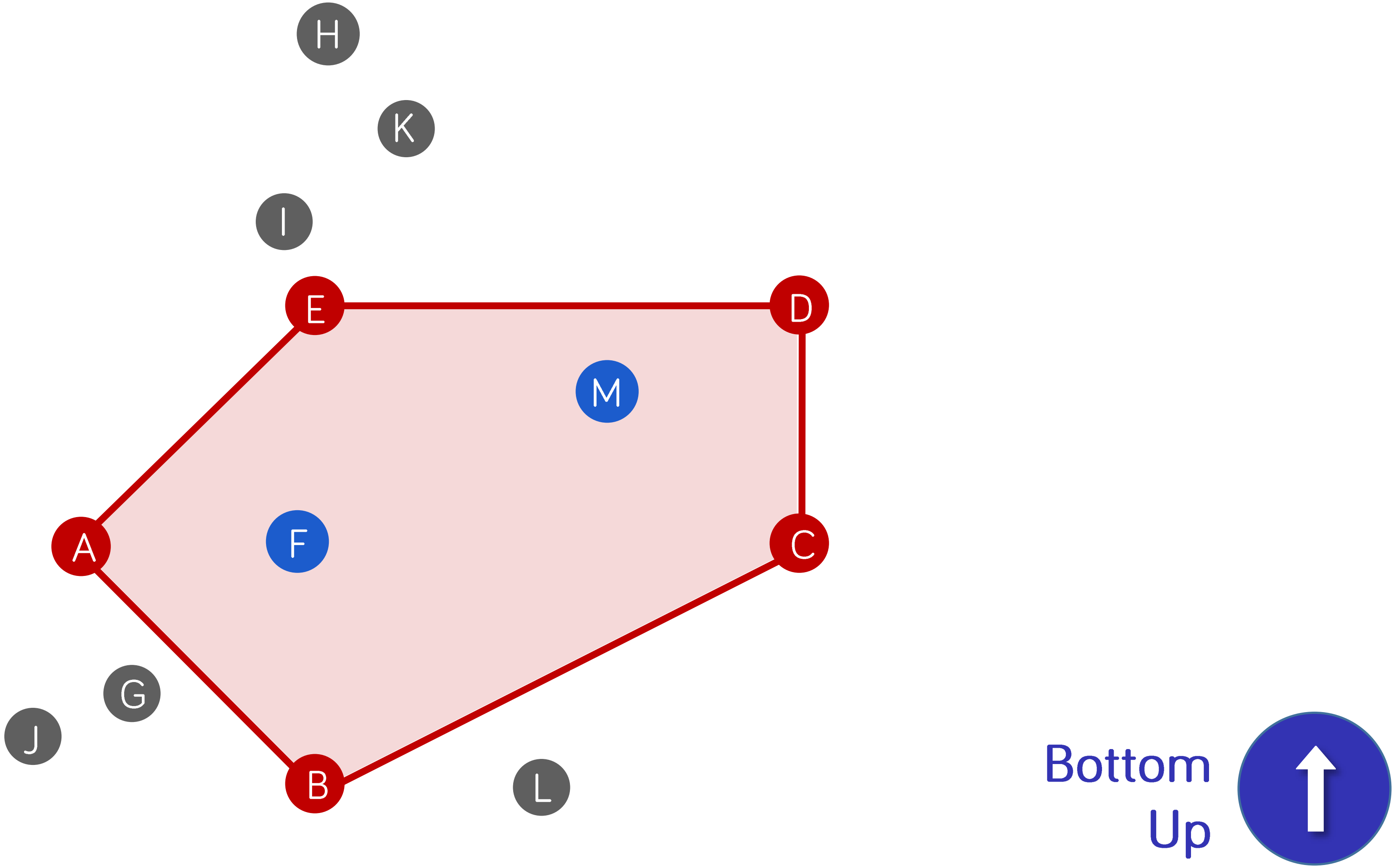






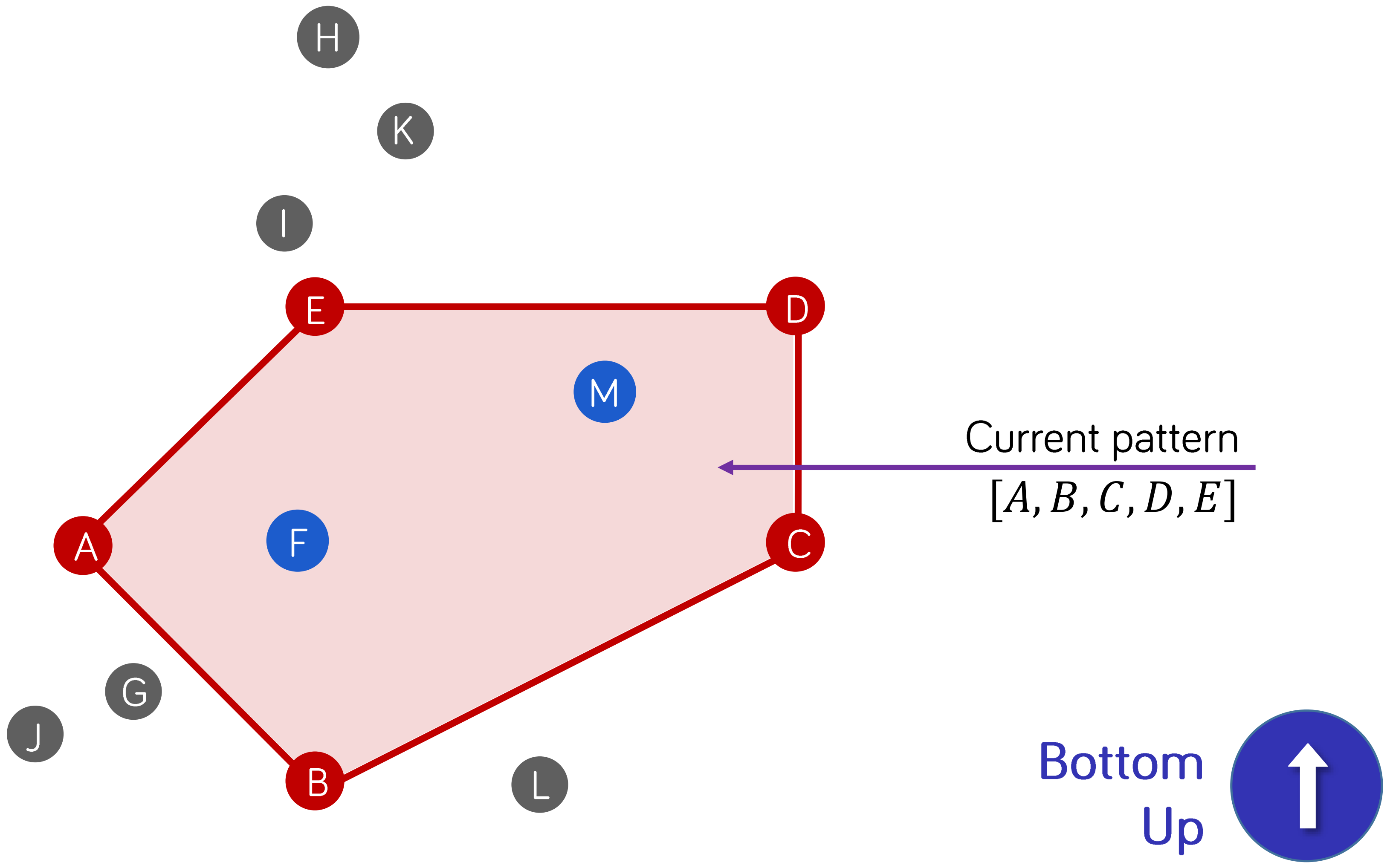
Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($A < B$ ...)



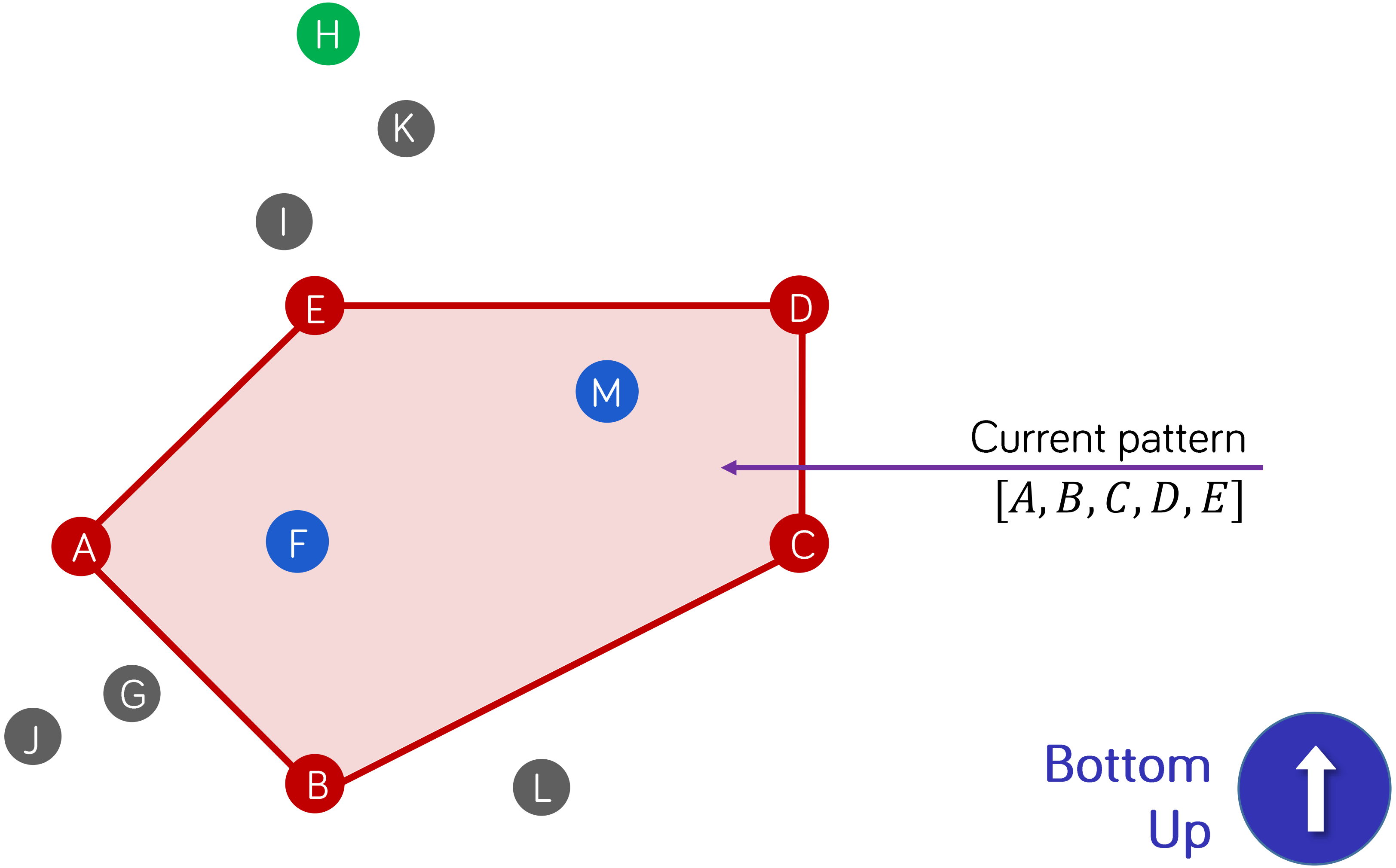
Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($A < B$ ...)



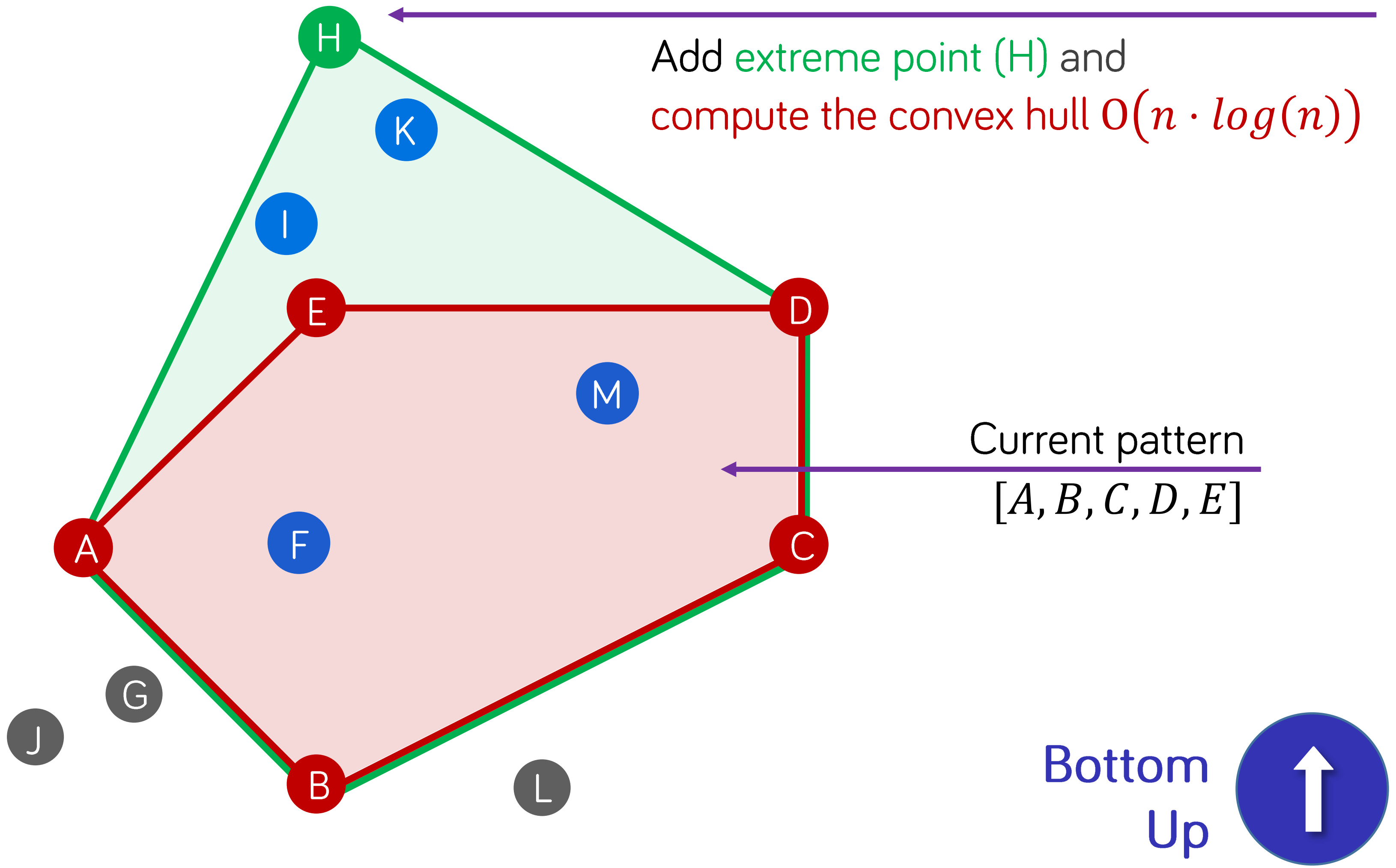
Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)



Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

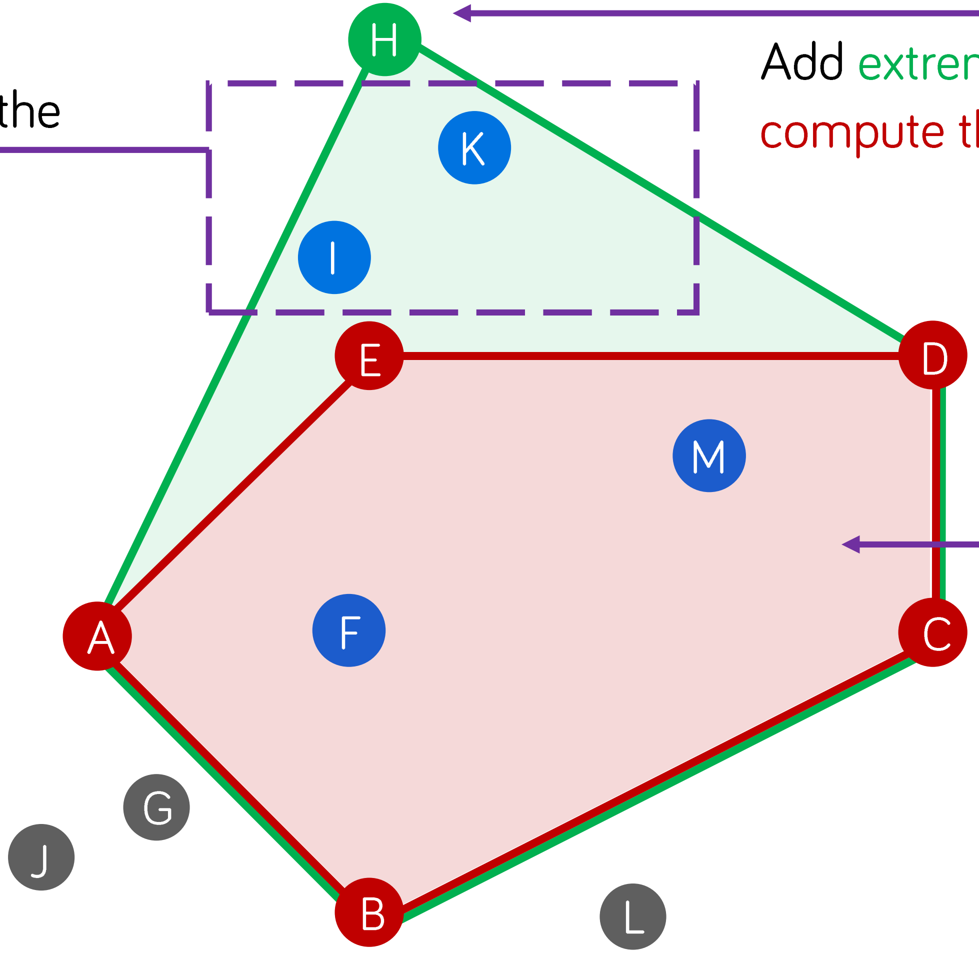


*n designate the number of objects

Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

All the newly added objects to the extent comes after (I, $K > H$)



Add extreme point (H) and compute the convex hull $O(n \cdot log(n))$

Current pattern $[A, B, C, D, E]$

Bottom Up

*n designate the number of objects

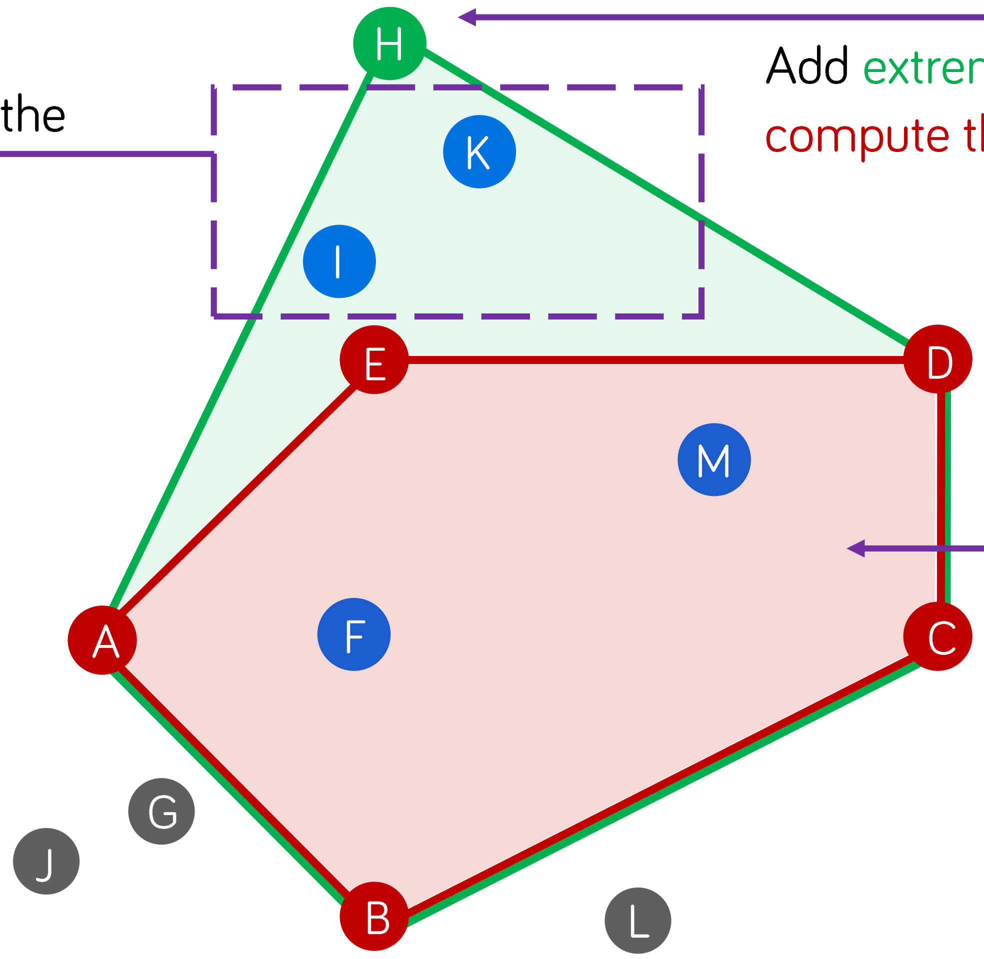
Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

All the newly added objects to the extent comes after (I, $K > H$)

$\Longrightarrow$ pattern not already visited.

Add extreme point (H) and compute the convex hull $O(n \cdot log(n))$



Current pattern $[A, B, C, D, E]$

Bottom Up

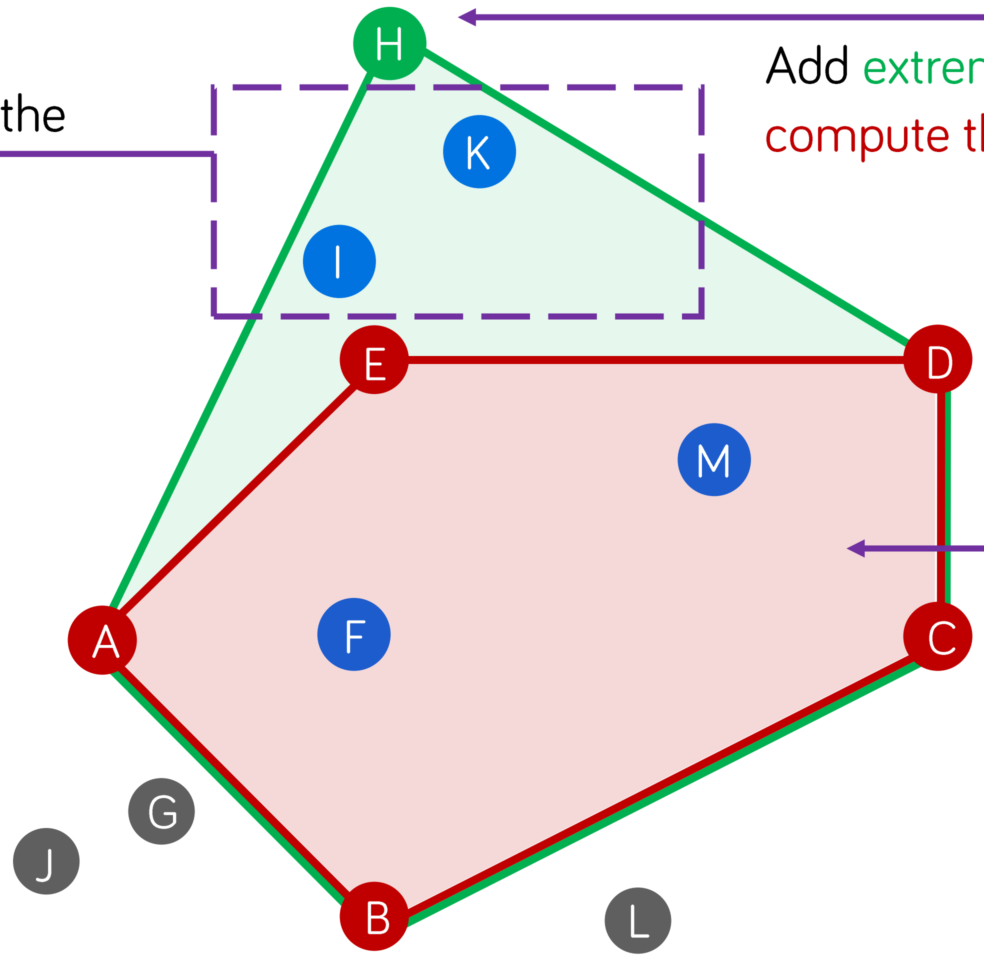
*n designate the number of objects

Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

All the newly added objects to the extent comes after (I, $K > H$)

$\Longrightarrow$ pattern not already visited.



Add extreme point (H) and compute the convex hull $O(n \cdot log(n))$

Current pattern $[A, B, C, D, E]$

If the pattern is already visited the algorithm backtracks.

Bottom Up

*n designate the number of objects

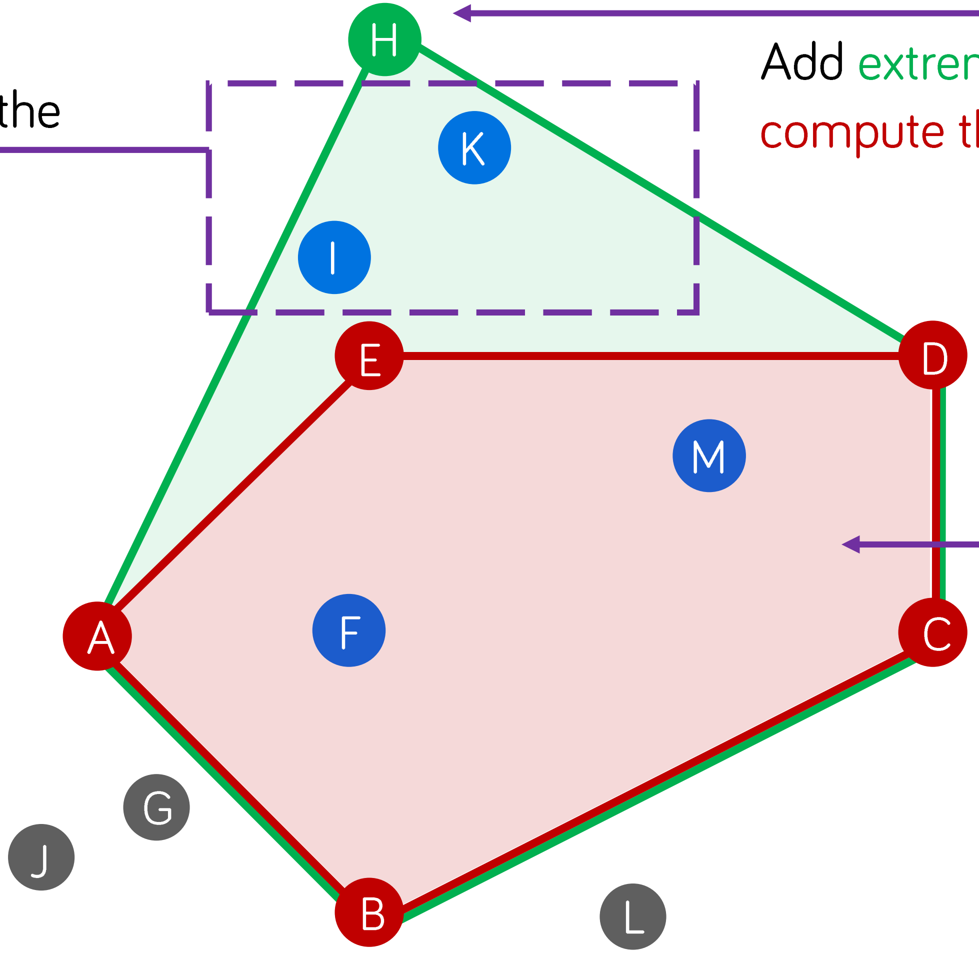
Algorithm 1 - CloseByOne

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

Add extreme point (H) and compute the convex hull $O(n \cdot log(n))$

All the newly added objects to the extent comes after (I, $K > H$)

$\Longrightarrow$ pattern not already visited.



Current pattern $[A, B, C, D, E]$

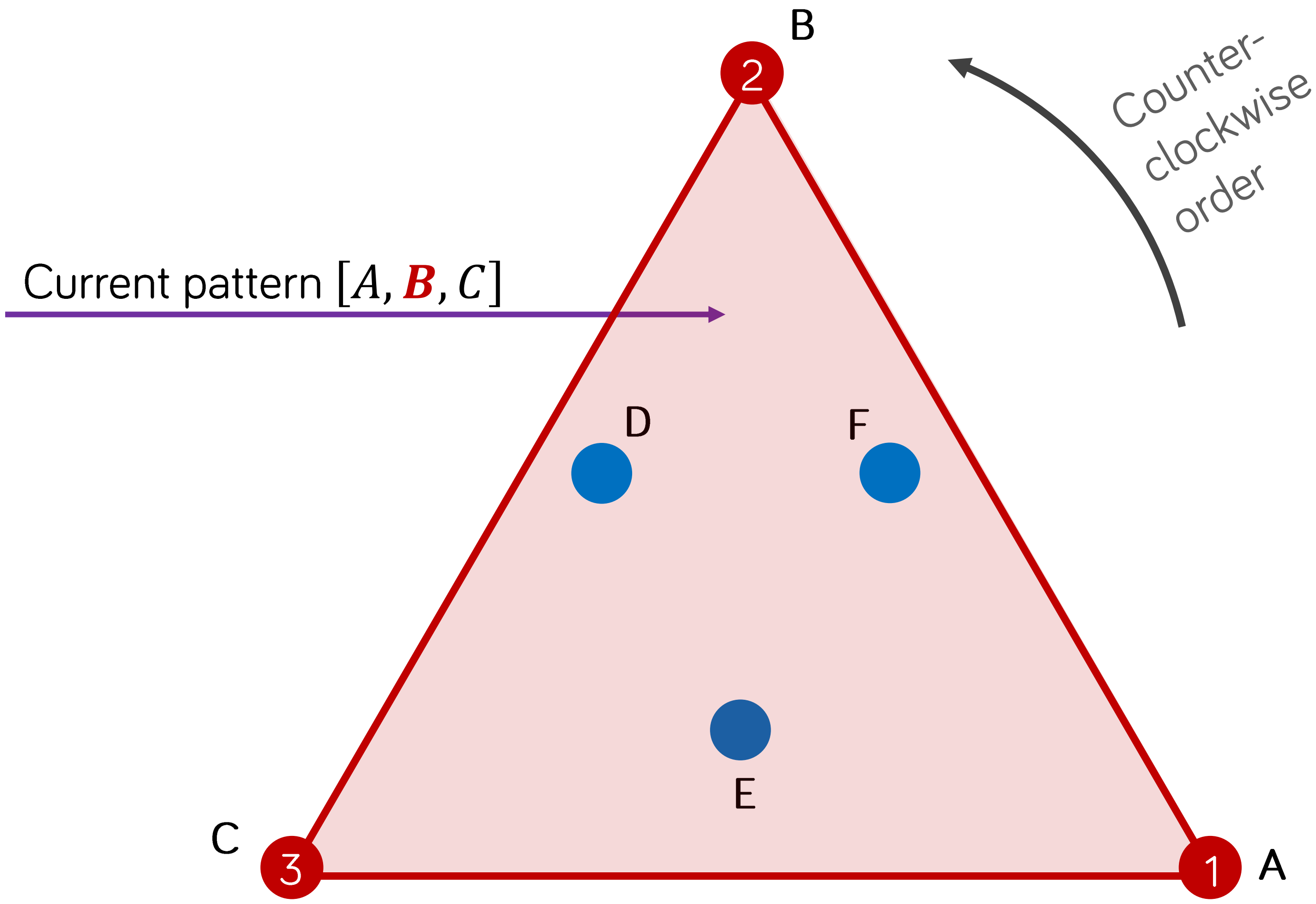
If the pattern is already visited the algorithm backtracks.

Continue enumeration by adding extreme point ($J > H$)

Bottom Up

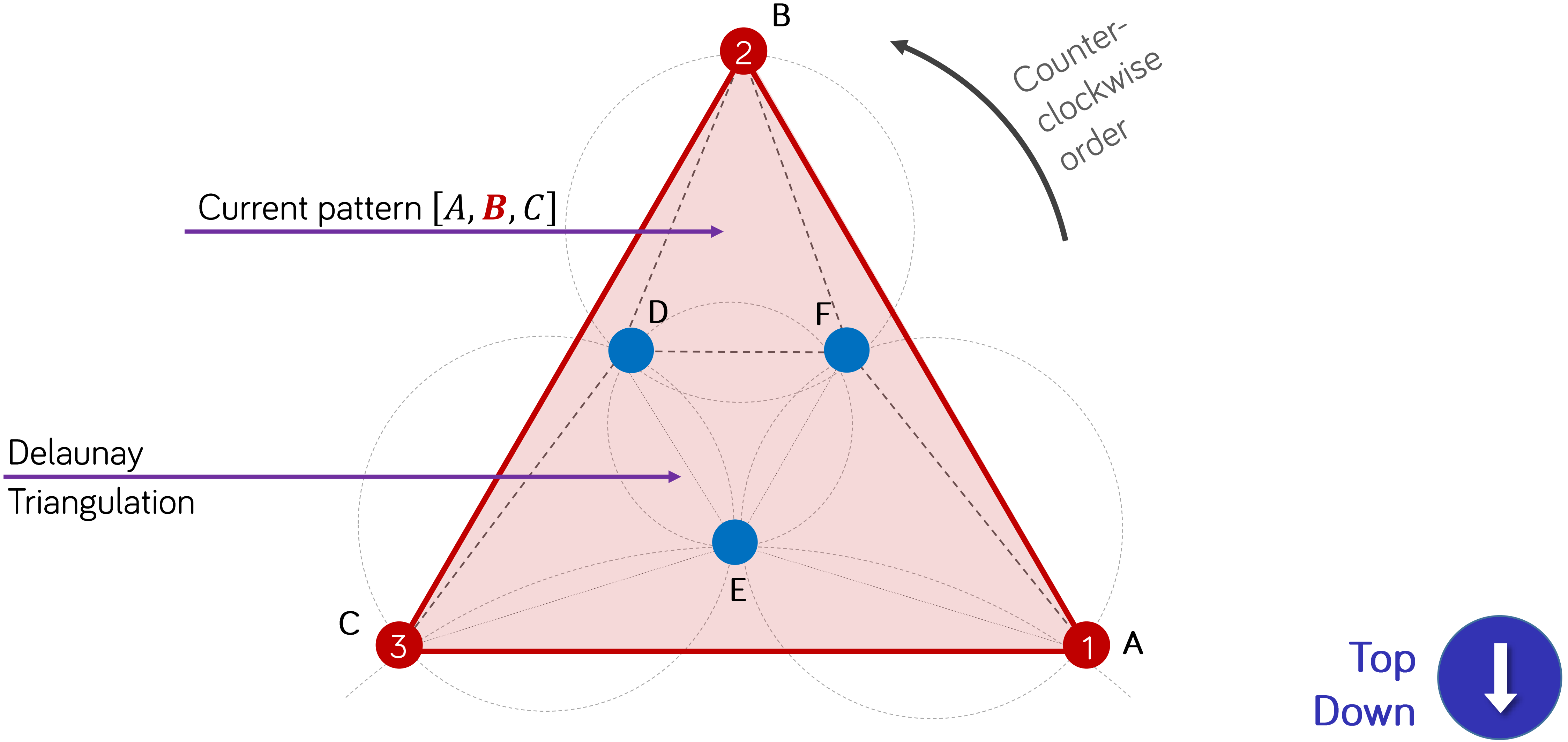
*n designate the number of objects

Algorithm 2 – Delaunay Triangulation Based

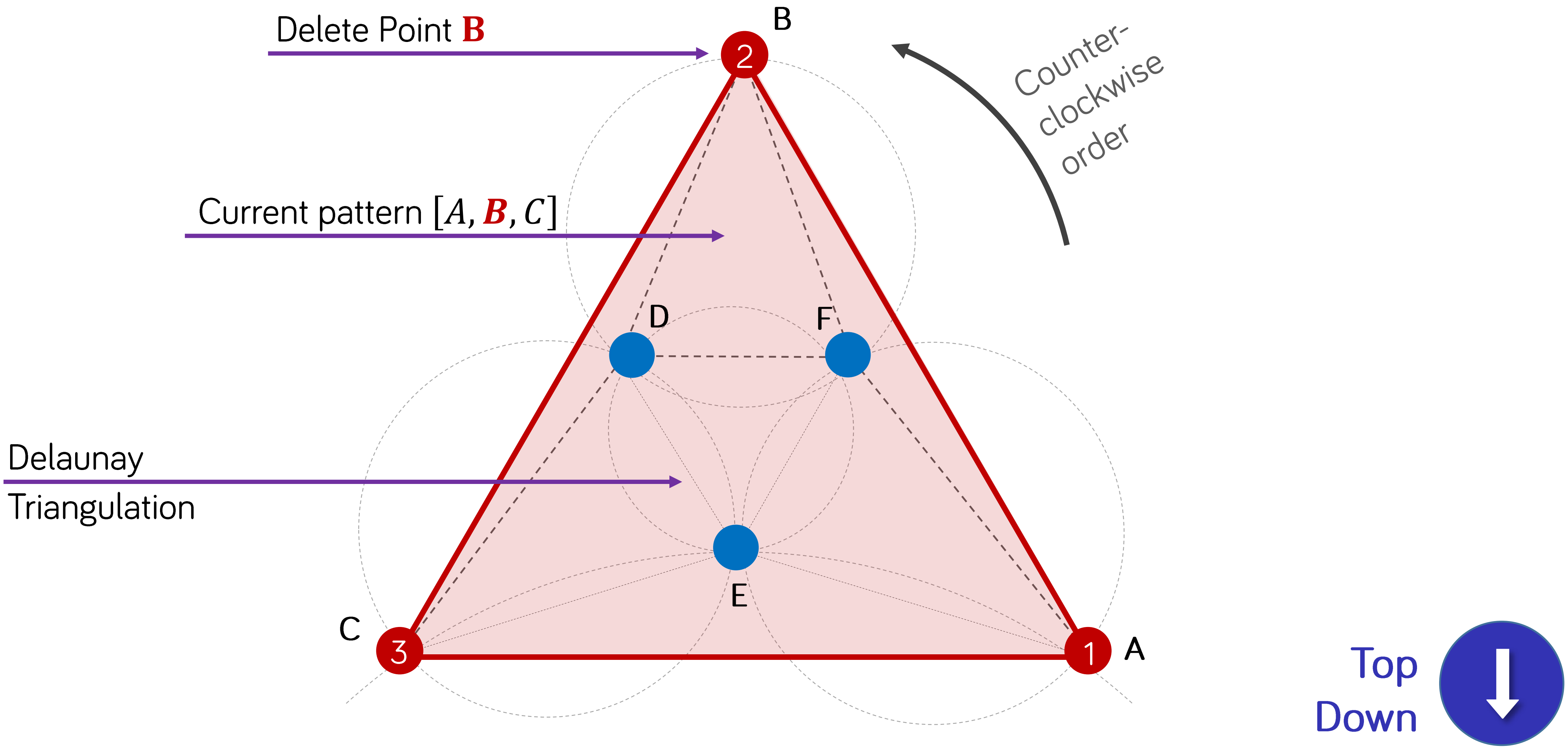


Top Down

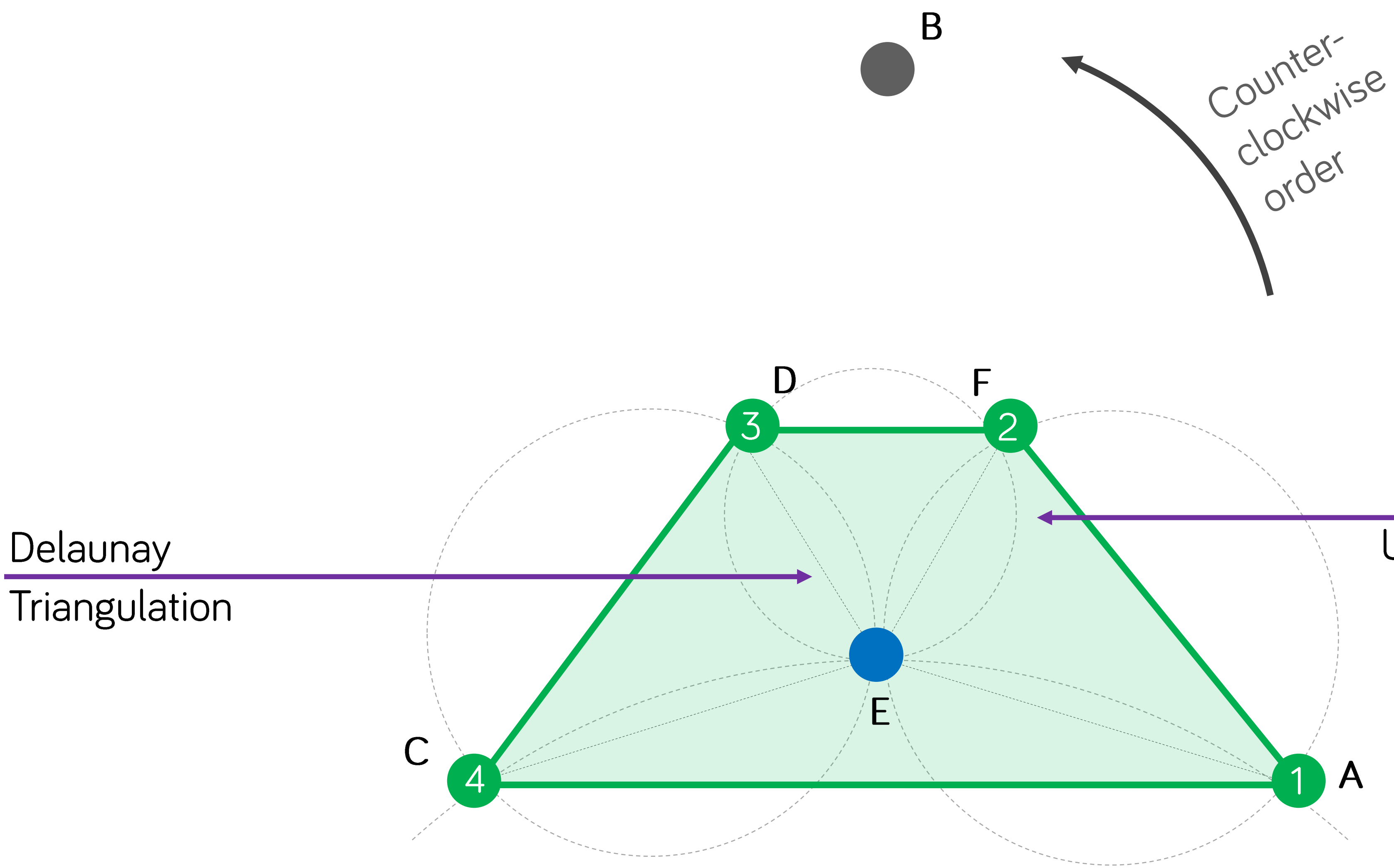
Algorithm 2 – Delaunay Triangulation Based



Algorithm 2 – Delaunay Triangulation Based



Algorithm 2 – Delaunay Triangulation Based



New pattern $[A, \mathbf{F}, \mathbf{D}, C]$

Update Delaunay Triangulation

$O(k \cdot log(k))$

Top Down

*k designates the number of neighbors

Algorithm 2 – Delaunay Triangulation Based

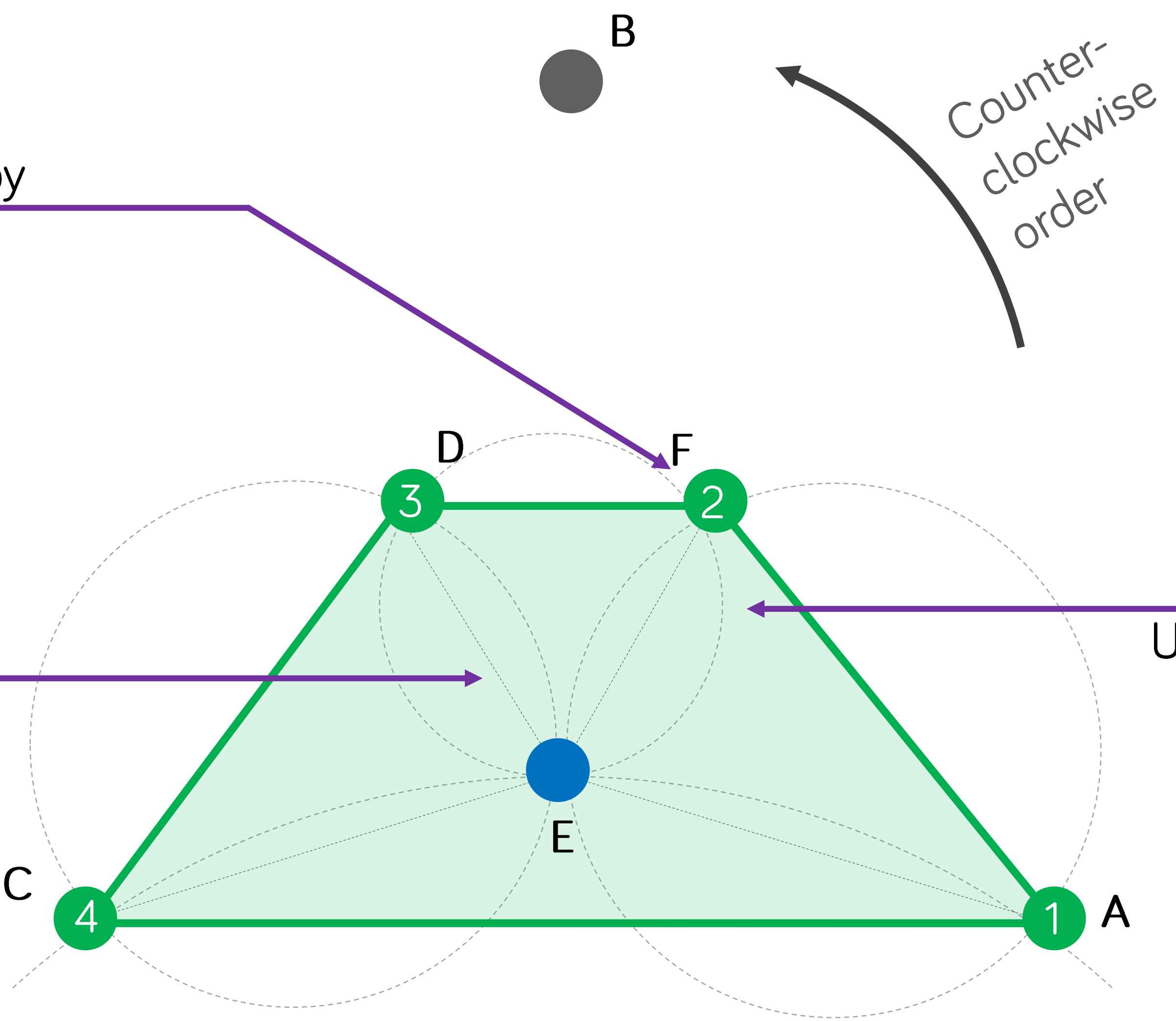
Continue enumeration by deleting F (position 2)

Delaunay Triangulation

New pattern $[A, \boldsymbol{F}, \boldsymbol{D}, C]$

Update Delaunay Triangulation

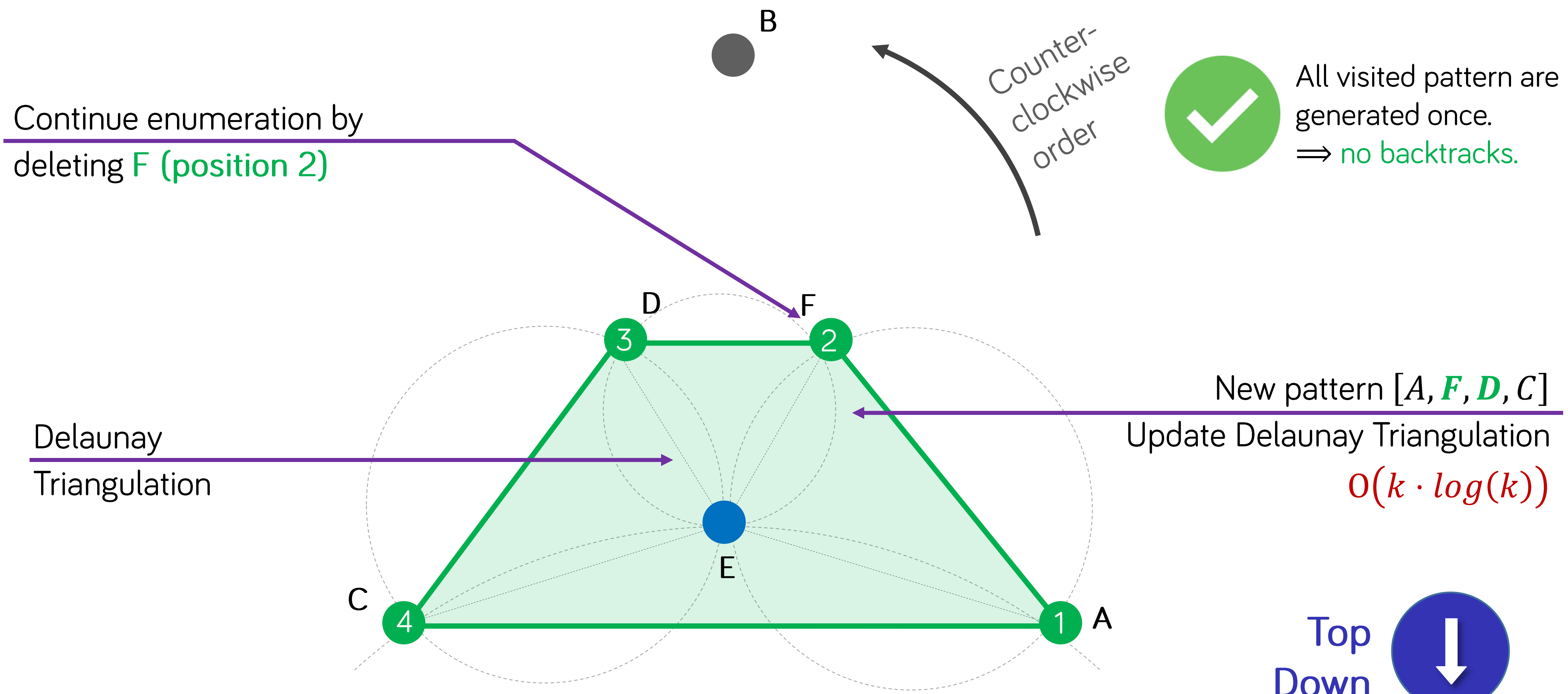
$O(k \cdot log(k))$



Top Down

*k designates the number of neighbors

Algorithm 2 – Delaunay Triangulation Based



Top Down

*k designates the number of neighbors

Remark

Let be $E \subset \mathbb{R}^2$ and $e \in \mathbb{R}^2$. Let be $\boldsymbol{ch(E)}$ the set of vertices of the convex hulls of E. We have:

$$|\boldsymbol{ch(E \cup \{e\})}| \leq |\boldsymbol{ch(E)}| + 1$$

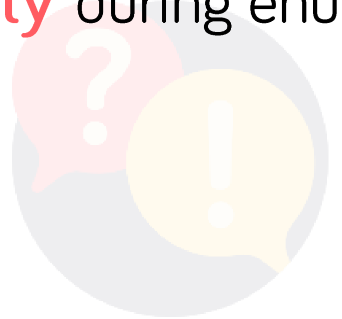
Algorithm 3 – Vision Based (1)

Remark

Let be $E \subset \mathbb{R}^2$ and $e \in \mathbb{R}^2$. Let be $\boldsymbol{ch(E)}$ the set of vertices of the convex hulls of E. We have:

$$|\boldsymbol{ch}(\boldsymbol{E} \cup \{\boldsymbol{e}\})| \leq |\boldsymbol{ch}(\boldsymbol{E})| + 1$$

We want the following property during enumeration:



Remark

Let be $E \subset \mathbb{R}^2$ and $e \in \mathbb{R}^2$. Let be $\boldsymbol{ch(E)}$ the set of vertices of the convex hulls of E. We have:

$$|\boldsymbol{ch(E \cup \{e\})}| \leq |\boldsymbol{ch(E)}| + 1$$

We want the following property during enumeration:

Any direct refinement of a description d produce a new one d' such that:

$$|d'| = |d| + 1$$

Algorithm 3 – Vision Based (1)

Remark

Let be $E \subset \mathbb{R}^2$ and $e \in \mathbb{R}^2$. Let be $\boldsymbol{ch(E)}$ the set of vertices of the convex hulls of E. We have:

$$|\boldsymbol{ch(E \cup \{e\})}| \leq |\boldsymbol{ch(E)}| + 1$$

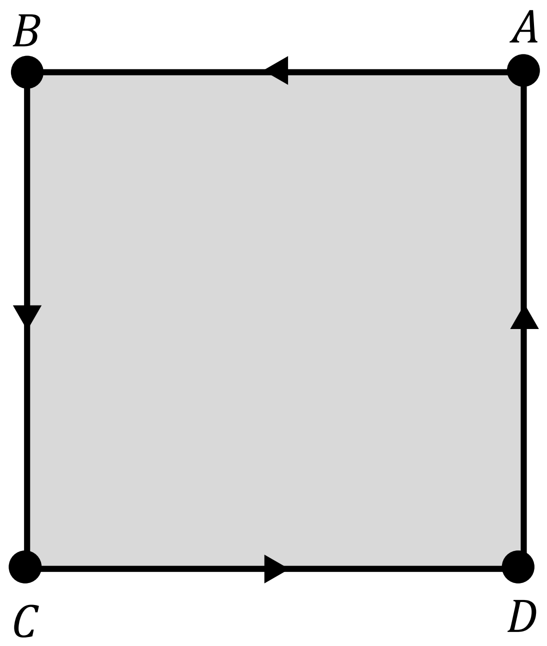
We want the following property during enumeration:

Any direct refinement of a description d produce a new one d' such that:

$$|d'| = |d| + 1$$

The constraint "maximum number of vertices $\leq \tau$ " becomes monotonic.

Algorithm 3 – Vision Based (2)



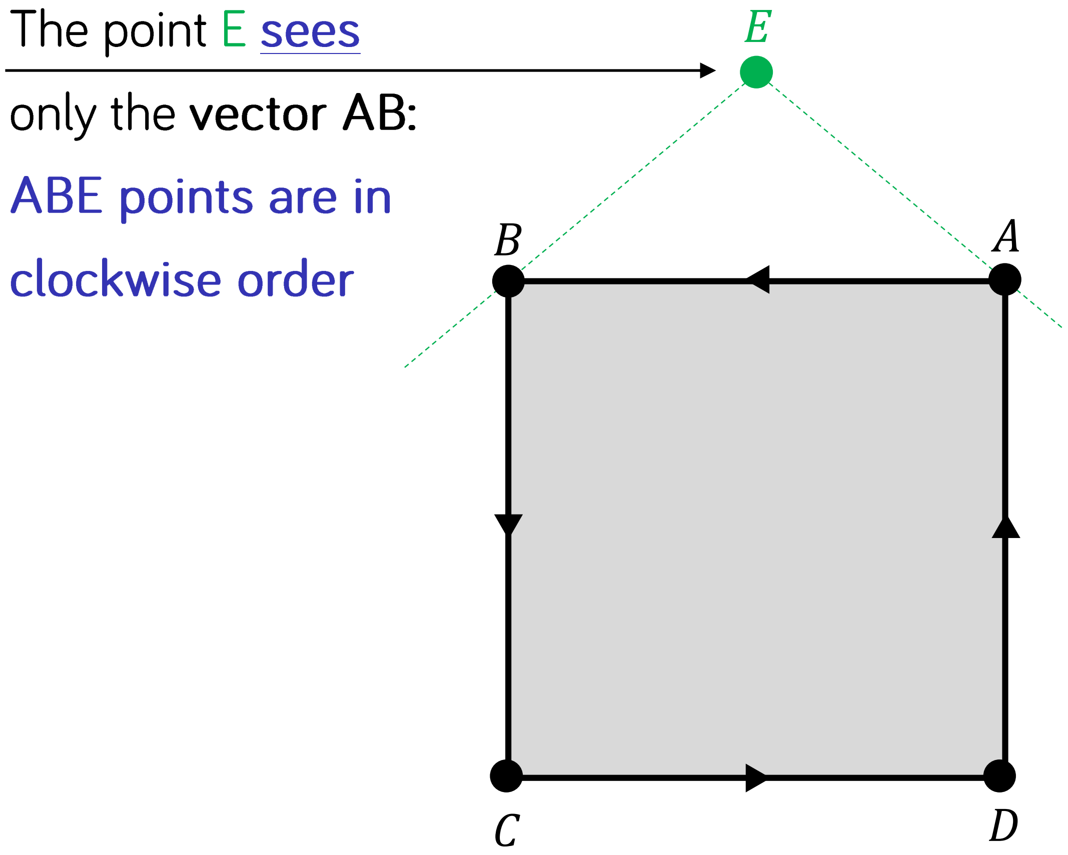
Algorithm 3 – Vision Based (2)

The point E sees

only the **vector** AB:

ABE points are in

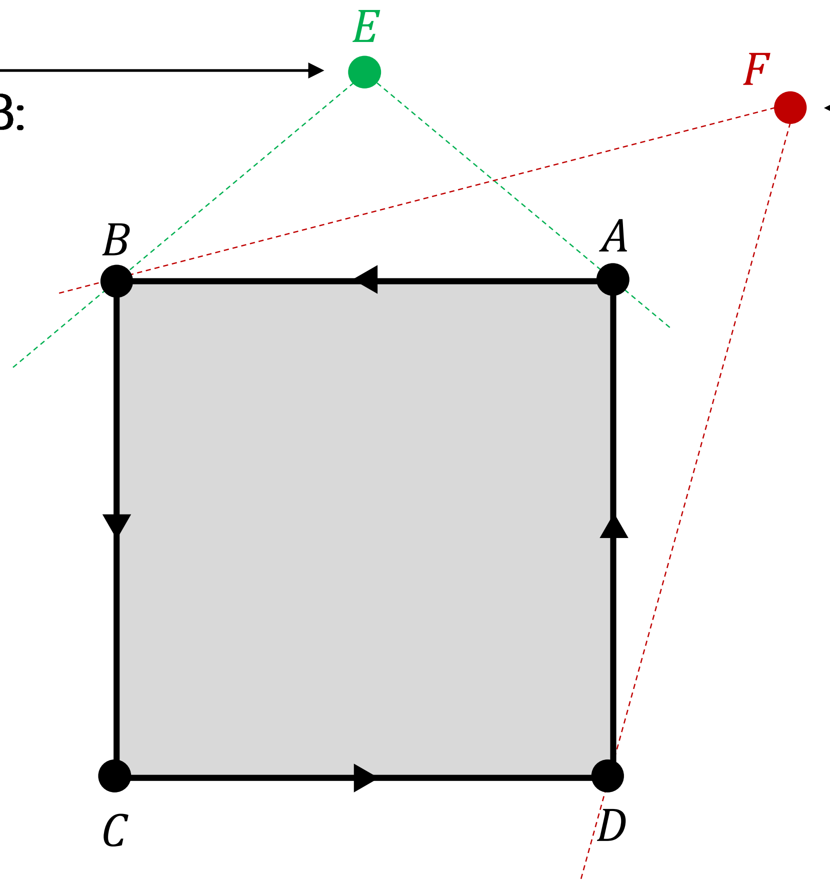
clockwise order



Algorithm 3 – Vision Based (2)

The point E sees only the **vector AB**:

ABE points are in clockwise order



The point F sees the **vectors DA and AB**

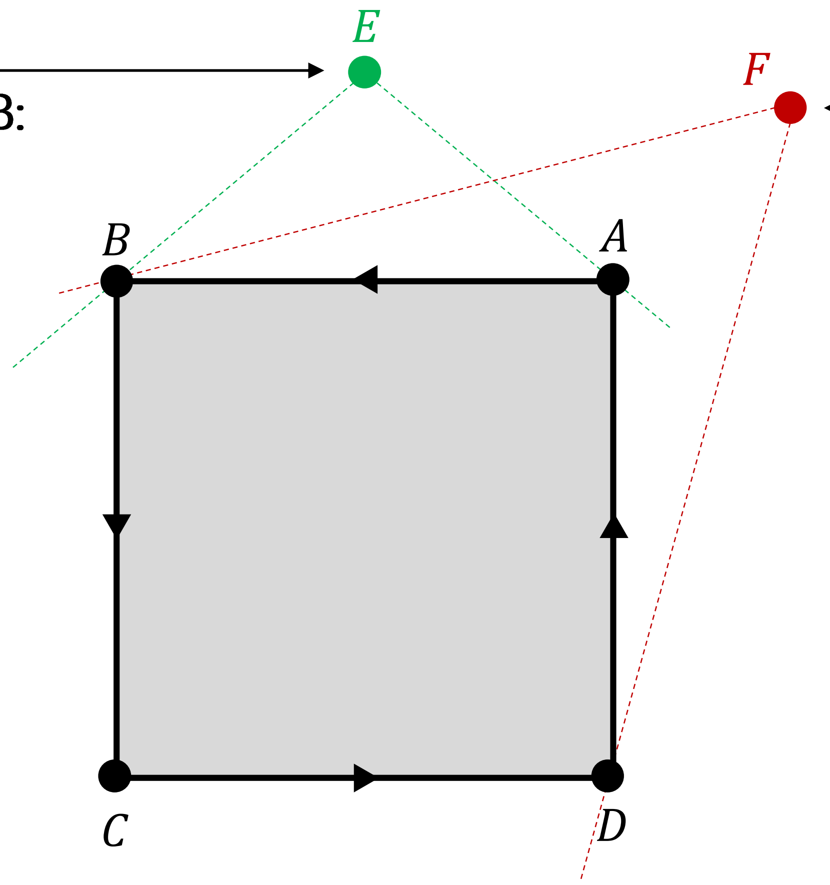
ABF points are in clockwise order

DAF points are in clockwise order

Algorithm 3 – Vision Based (2)

The point E sees only the **vector AB**:

ABE points are in clockwise order



The point F sees the **vectors DA and AB**

ABF points are in clockwise order

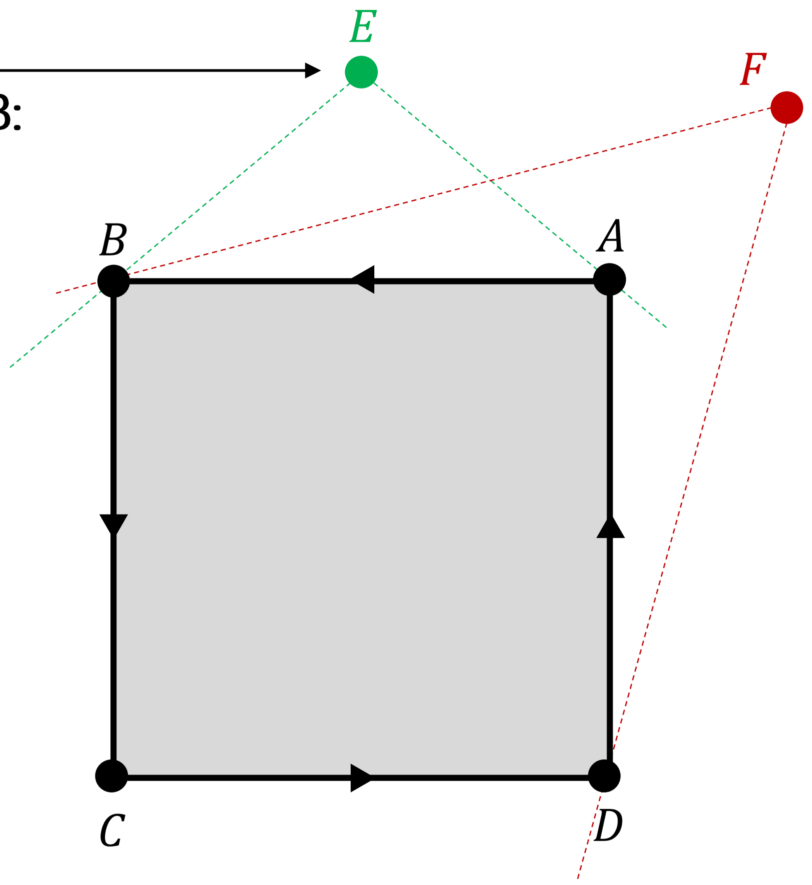
DAF points are in clockwise order

Adding F as an **extreme point** to the description $d = [A, B, C, D]$ will destroy the extreme point A creating $d' = [F, B, C, D]$

Algorithm 3 – Vision Based (2)

The point E sees only the **vector AB**:

ABE points are in clockwise order



The point F sees the **vectors DA and AB**

ABF points are in clockwise order

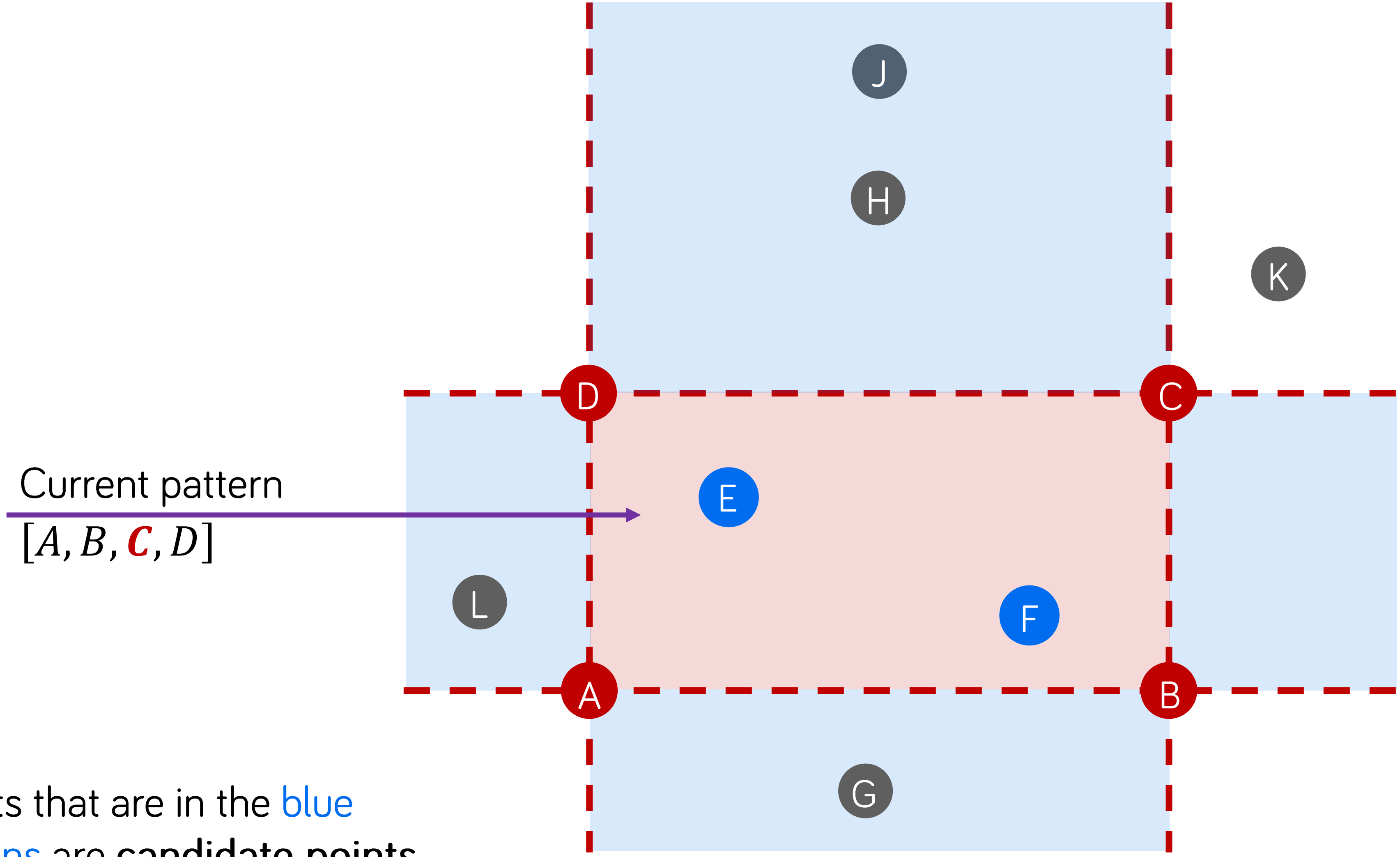
DAF points are in clockwise order

Adding F as an **extreme point** to the description $d = [A, B, C, D]$ will destroy the extreme point A creating $d' = [F, B, C, D]$

To ensure that $|\boldsymbol{d}'| = |\boldsymbol{d}| + \mathbf{1}$. One should add only points which sees only one segment. An addable points for a segment is called a **candidate point.**

Algorithm 3 – Vision Based (3)

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

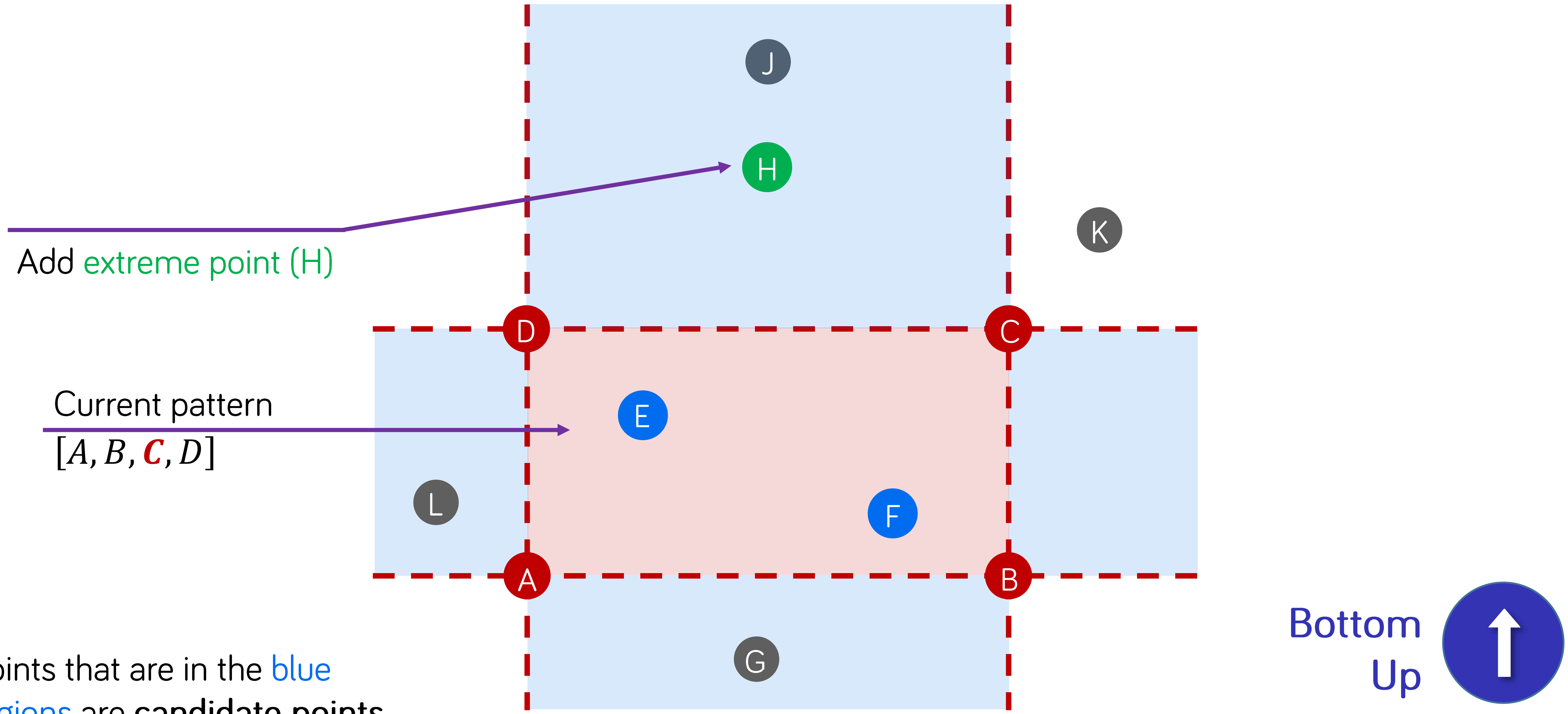


- Points that are in the blue regions are candidate points.

Bottom Up

Algorithm 3 – Vision Based (3)

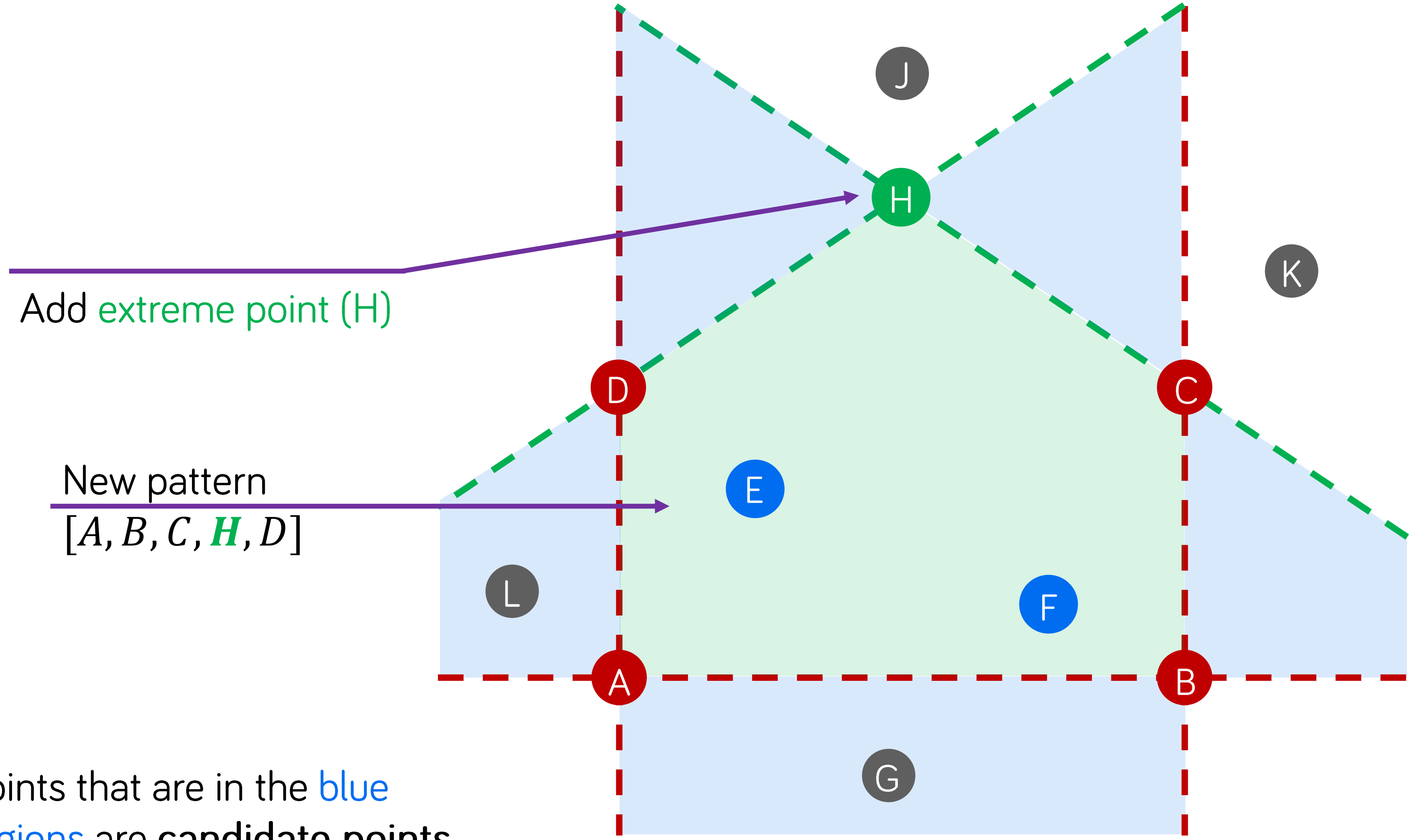
Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)



- Points that are in the blue regions are <u>candidate points</u>.

Algorithm 3 – Vision Based (3)

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

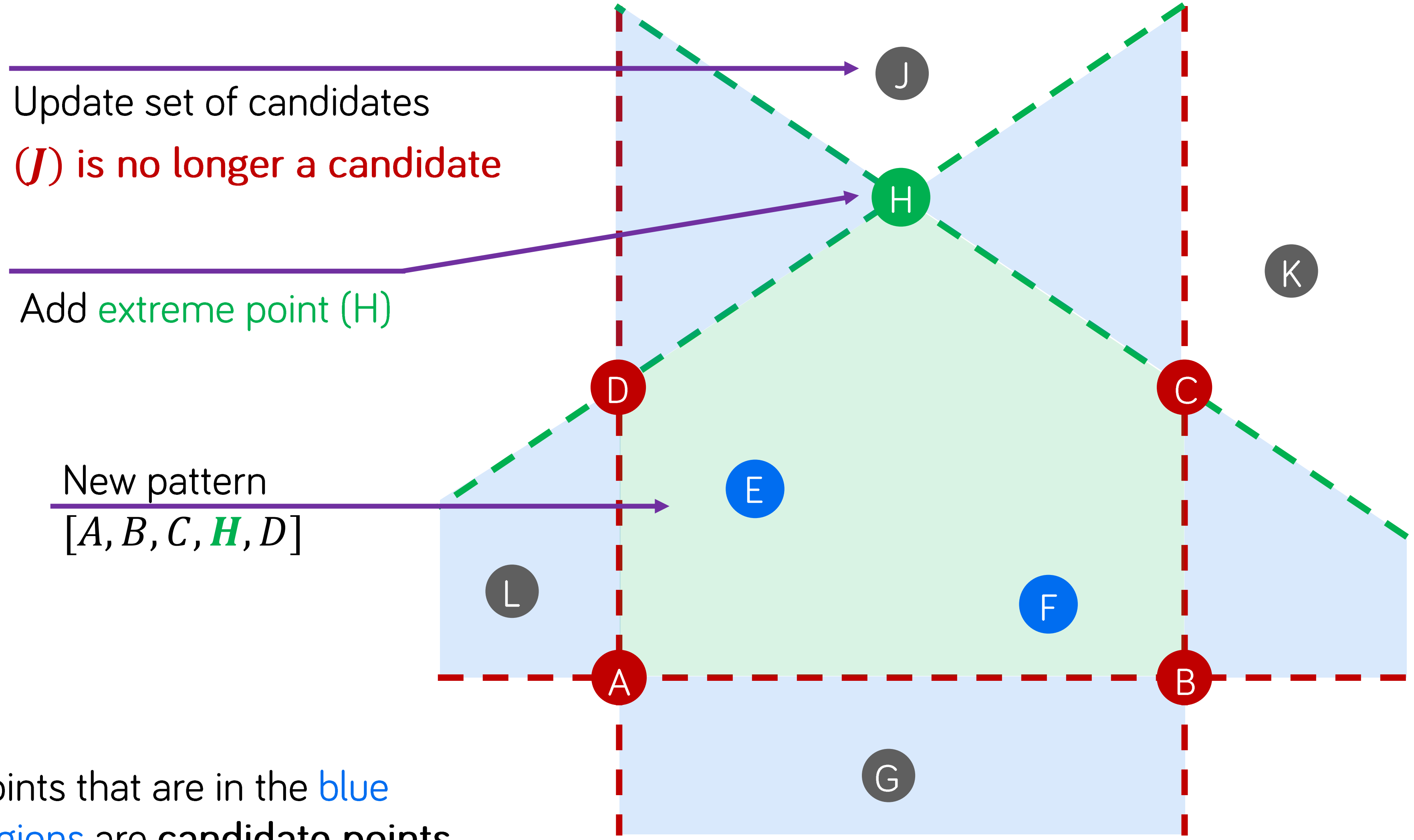


- Points that are in the blue regions are candidate points.

Bottom Up

Algorithm 3 – Vision Based (3)

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ …)



- Points that are in the blue regions are candidate points.

Bottom Up

Algorithm 3 – Vision Based (3)

Objects in the dataset are canonically ordered ($\boldsymbol{A} < \boldsymbol{B}$ ...)

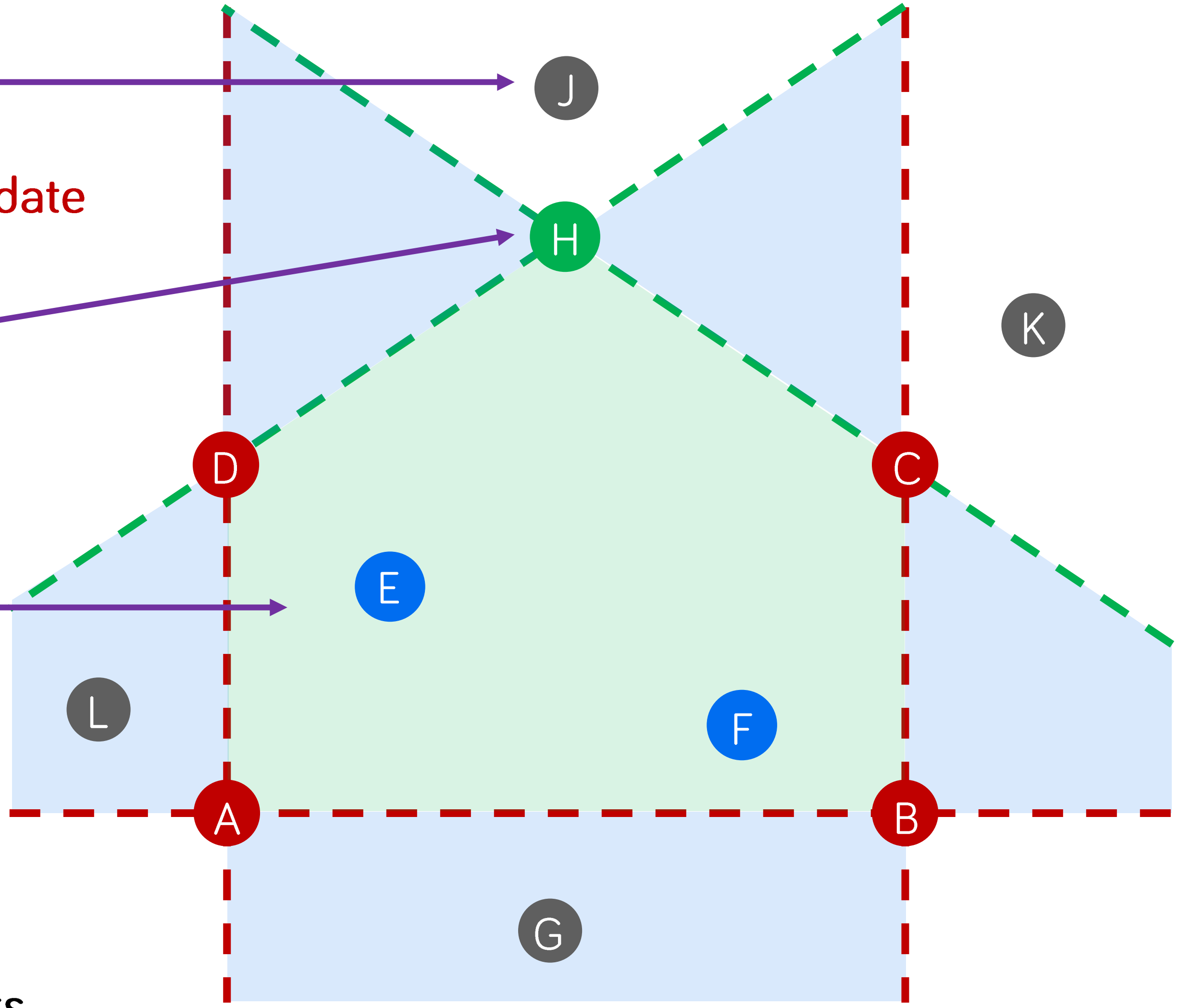
Update set of candidates

($\boldsymbol{J}$) is no longer a candidate

Add extreme point (H)

New pattern

$[A, B, C, \boldsymbol{H}, D]$



Continue enumeration from a candidate point that comes after (H).
$\Longrightarrow$ $not\ G, not\ J,$ $not\ K, \boldsymbol{but\ L}$

Bottom Up

- Points that are in the blue regions are candidate points.

Algorithm 3 – Vision Based (3)

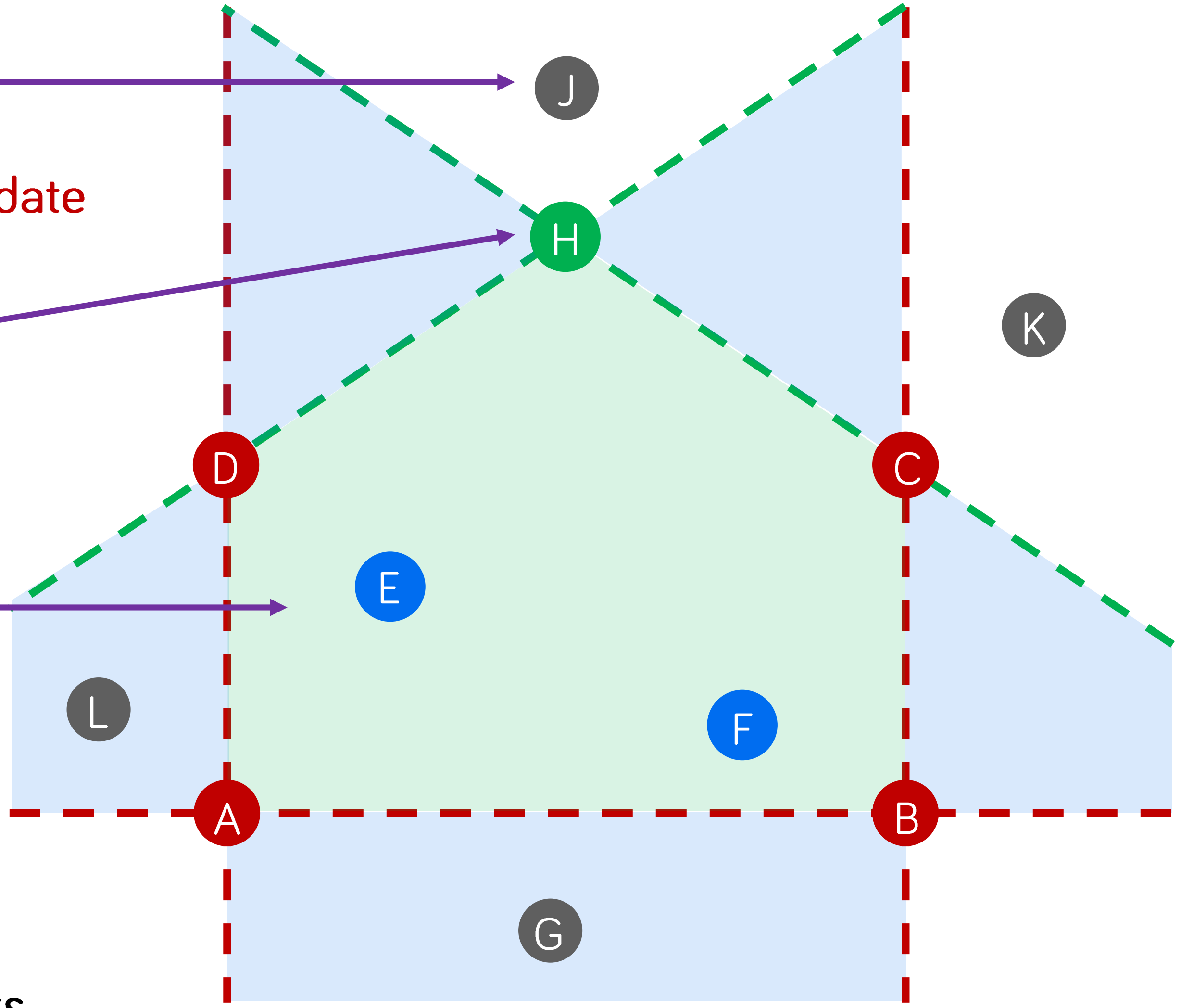
Objects in the dataset are canonically ordered ($A < B$ ...)

Update set of candidates

(J) is no longer a candidate

Add extreme point (H)

New pattern
$[A, B, C, \boldsymbol{H}, D]$



All visited pattern are generated once.
$\Longrightarrow$ no backtracks.

Continue enumeration from a candidate point that comes after (H).
$\Longrightarrow$ $not\ G, not\ J,$
$not\ K,$ $\boldsymbol{but\ L}$

Bottom Up

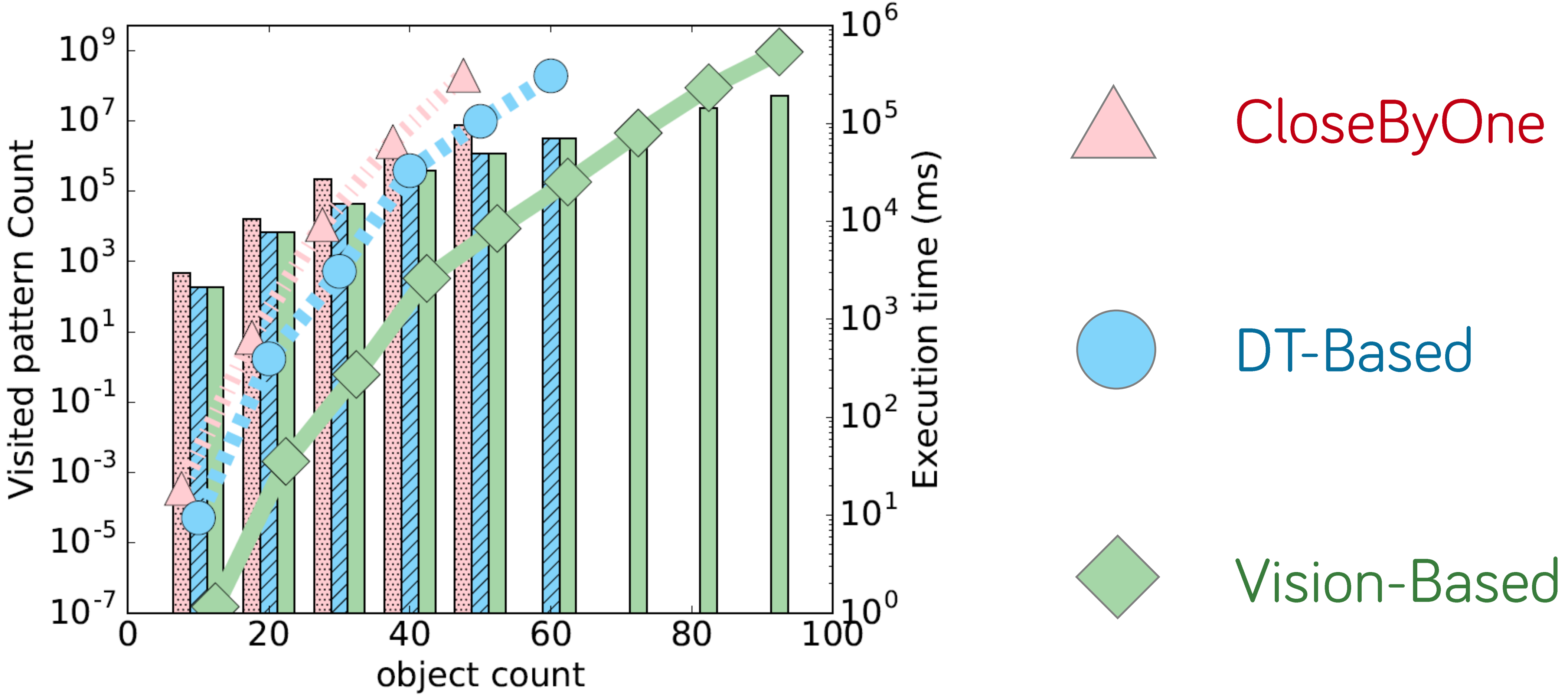
- Points that are in the blue regions are candidate points.

Constraint	CloseByOne (⬆)	DT-Based (⬇)	Vision-Based (⬆)
Min Support	✗	✓	✗
Min Area	✗	✓	✗
Max Area	✓	✗	✓
Min Perimeter	✗	✓	✗
Max Perimeter	✓	✗	✓
Max **complexity***	✗	✗	✓

***Complexity:** number of polygon vertices (or edges).

1. Algorithms
2. Experiments
3. Perspective

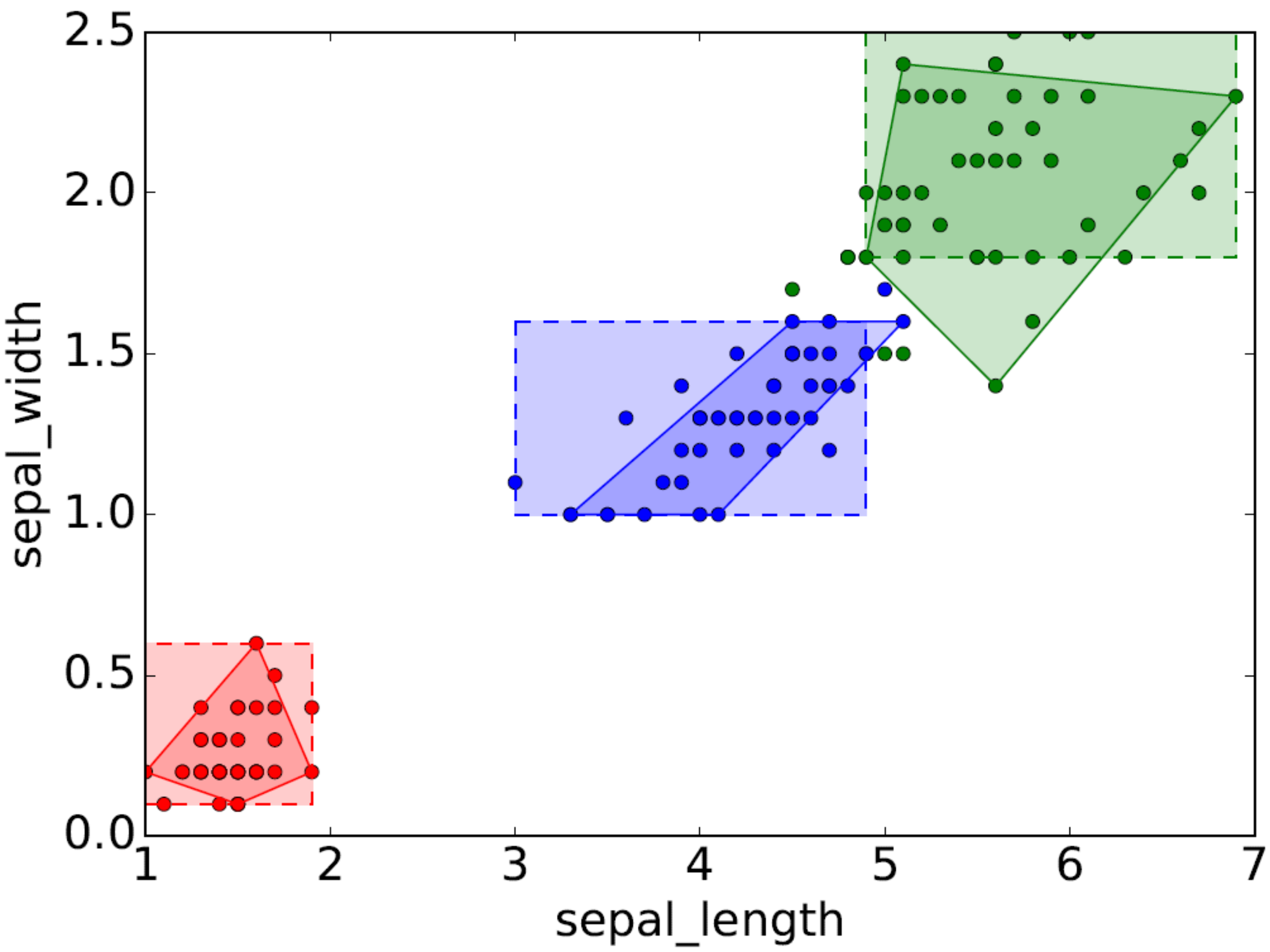
Algorithms Performance



*The used dataset is IRIS dataset projected on sepal width and petal width dimensions.

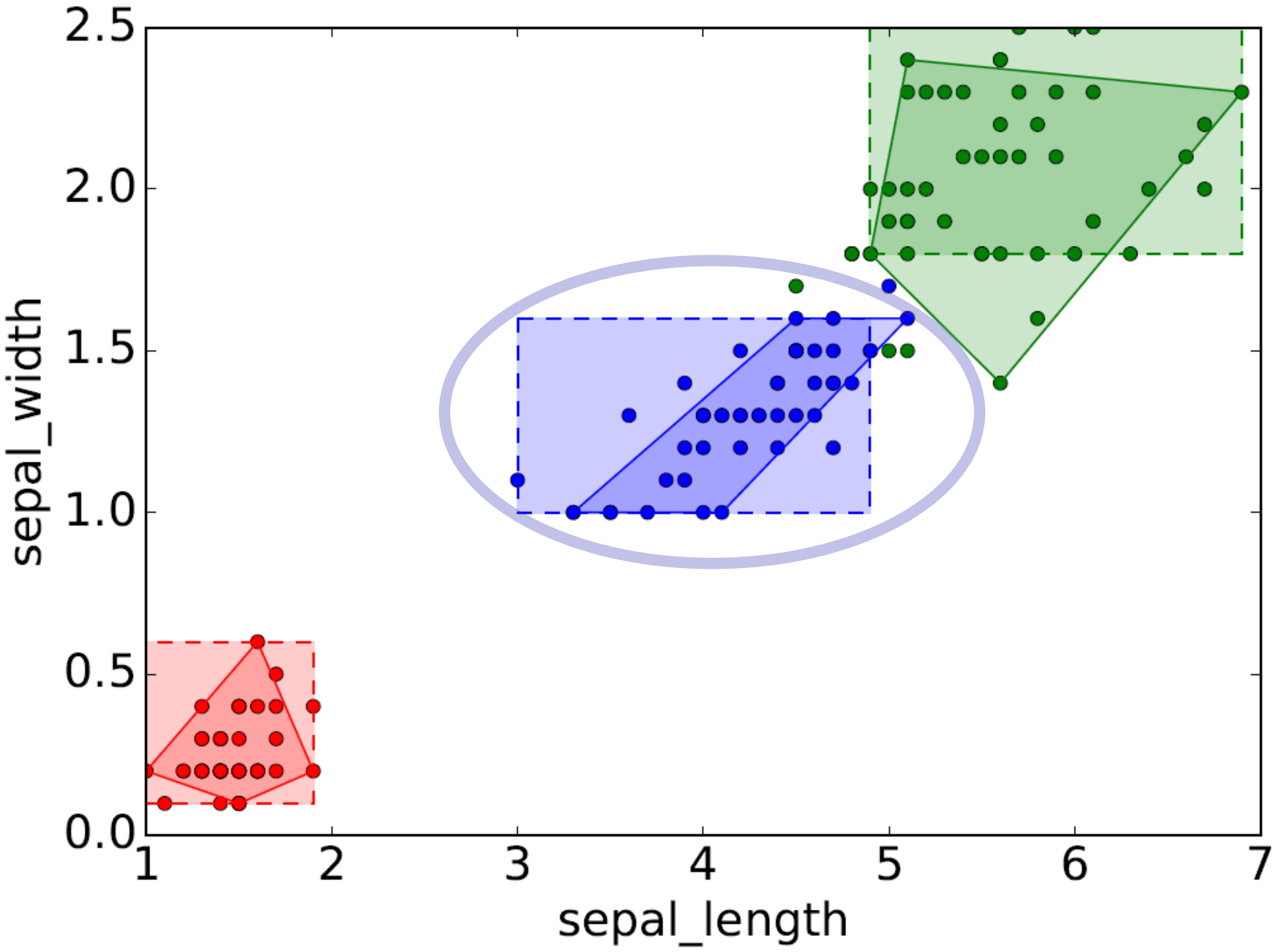
Interval Pattern VS Convex Polygon Pattern

Top-three homogenous interval patterns and convex polygon patterns with largest support and at most 4 vertices.

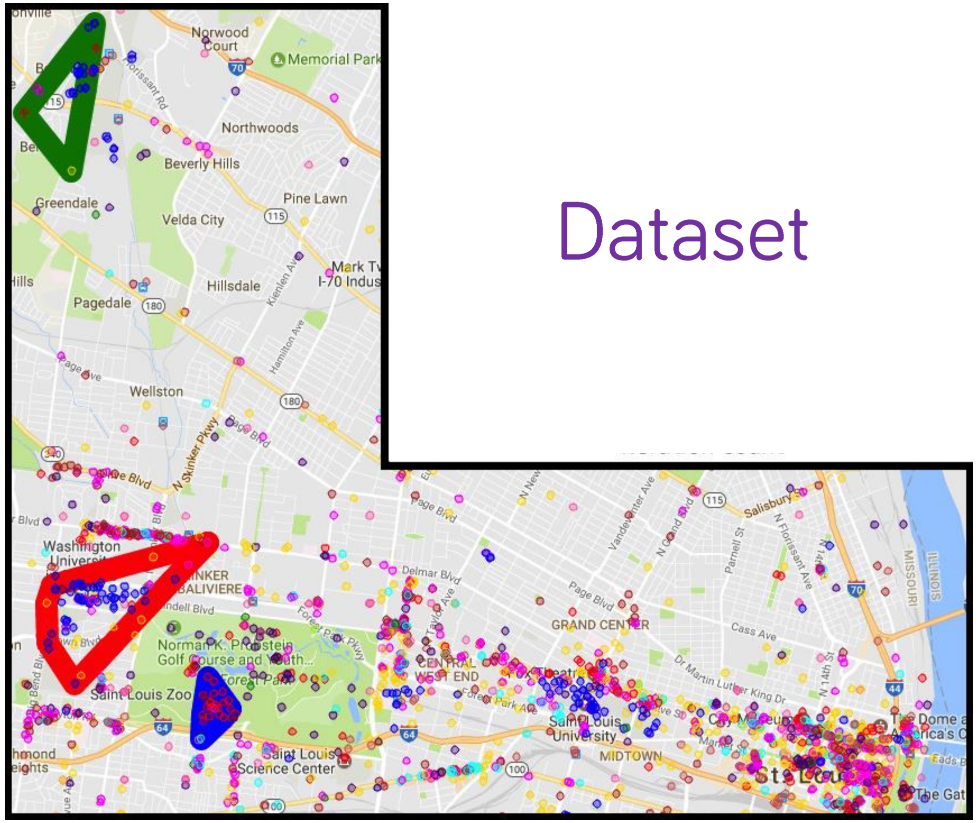


Interval Pattern VS Convex Polygon Pattern

Top-three homogenous interval patterns and convex polygon patterns with largest support and at most 4 vertices.



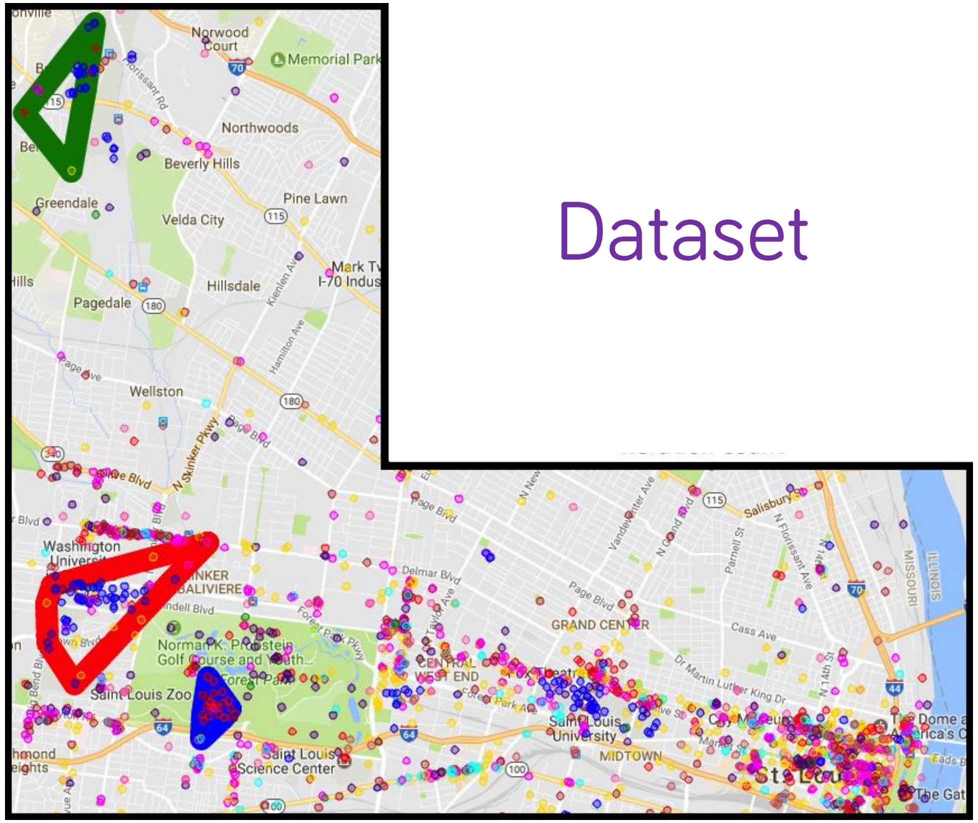
Convex Polygon Pattern in Action



Dataset

Convex Polygon Pattern in Action

Dataset. **St. Louis** town described by **3464** Points of interest (University, shop, ...) collected from *Foursquare*.

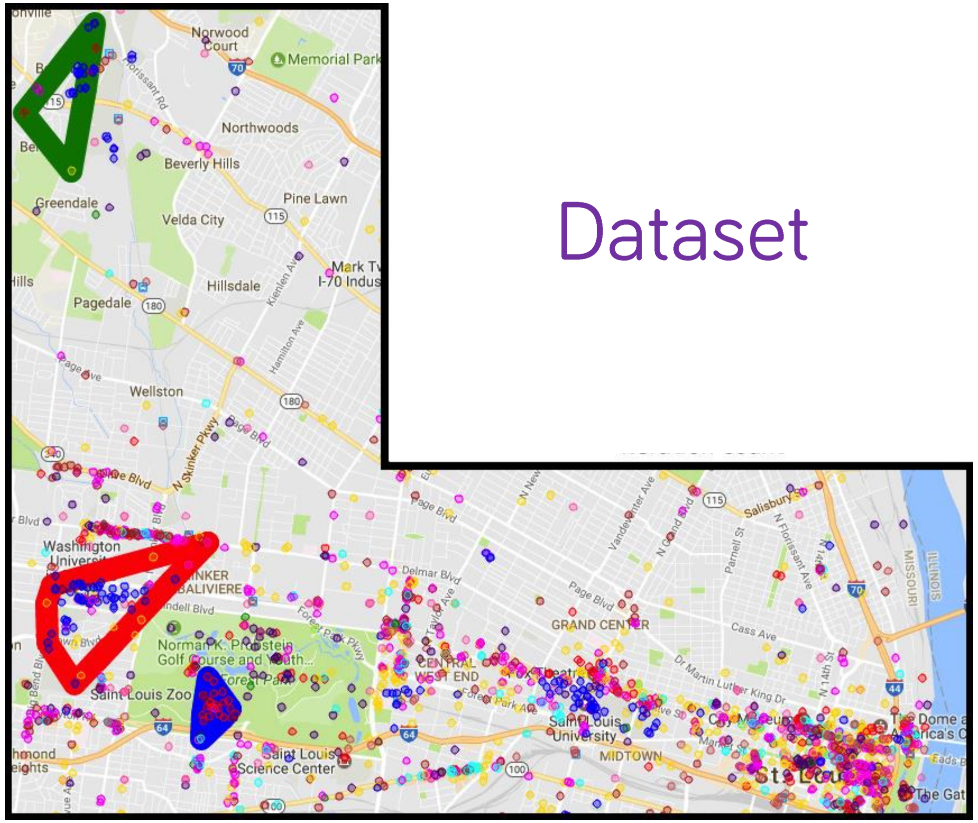


Dataset

Convex Polygon Pattern in Action

Dataset. **St. Louis** town described by **3464** Points of interest (University, shop, ...) collected from *Foursquare*.

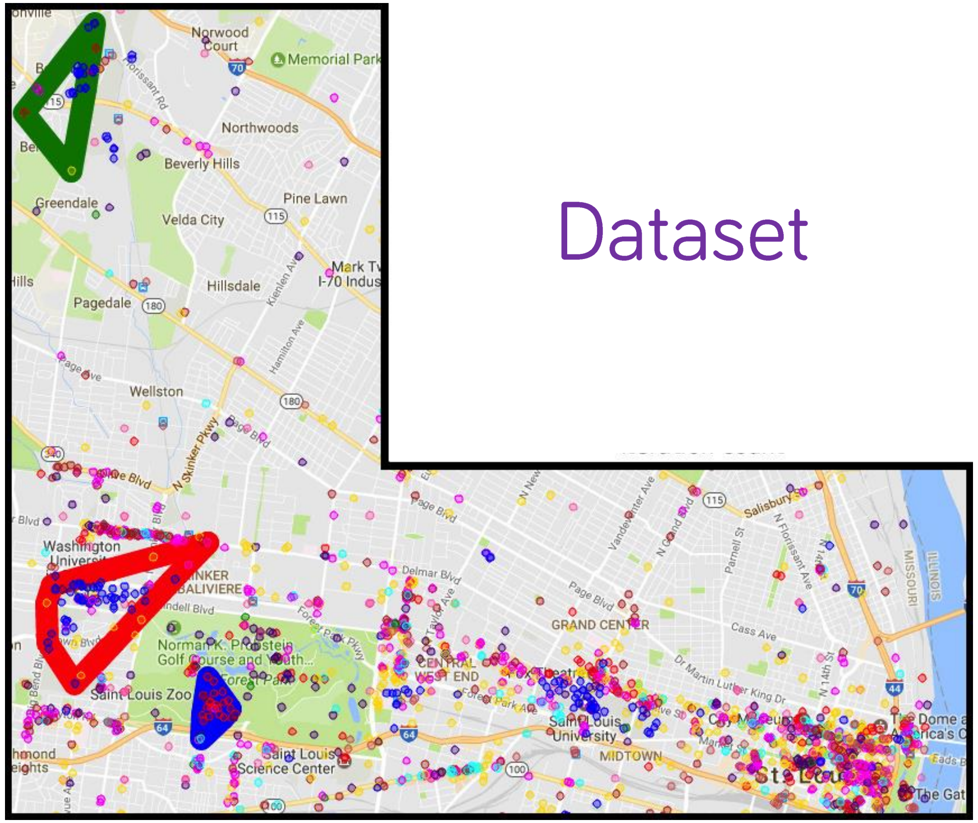
Objective. Homogenous convex regions with large support.



Dataset. St. Louis town described by $\mathbf{3464}$ Points of interest (University, shop, ...) collected from *Foursquare*.

Objective. Homogenous convex regions with large support.

Fact. Running an exhaustive search is almost impossible.



Dataset

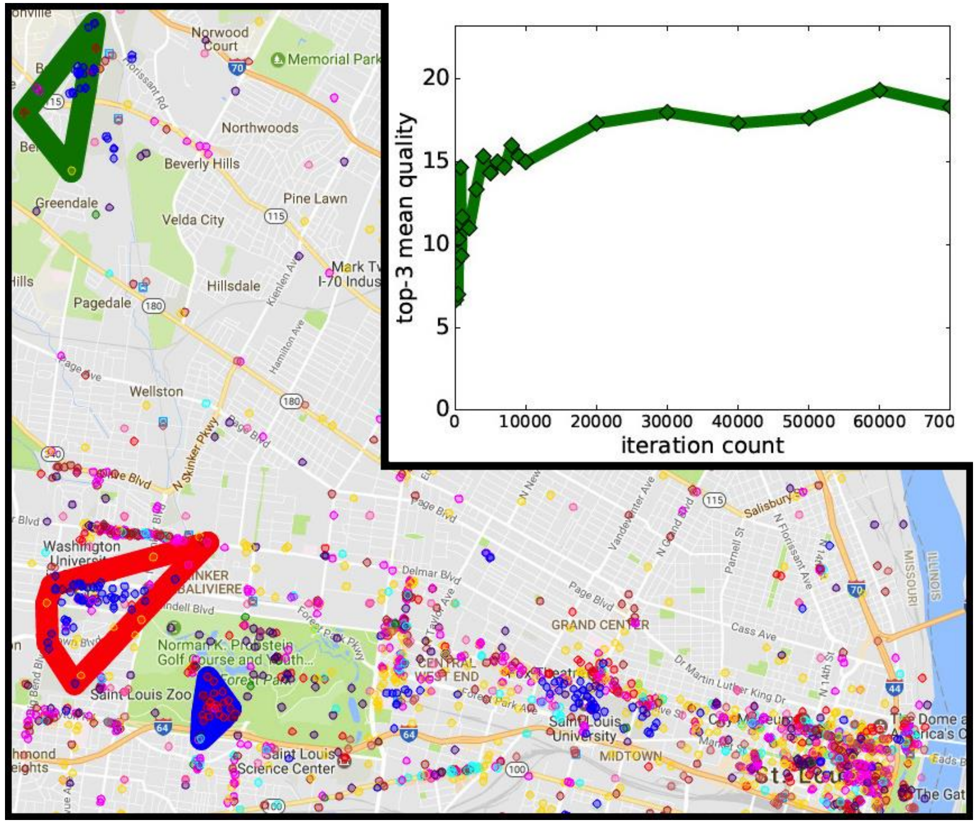
Convex Polygon Pattern in Action

Dataset. **St. Louis** town described by $\mathbf{3464}$ Points of interest (University, shop, ...) collected from *Foursquare*.

Objective. Homogenous convex regions with large support.

Fact. Running an exhaustive search is almost impossible.

Solution. Make use of heuristics or pattern sampling techniques to find good patterns fast. We made use of the newly proposed Monte-Carlo Tree Search pattern sampling technique [Bosc, G. et al. (**DMKD - minor revision**)] .



Guillaume Bosc, Chedy Raïssi, Jean-François Boulicault, Mehdi Kaytoue
Any-time Diverse Subgroup Discovery with Monte-Carlo Tree Search.
CoRR, abs/1609.08827, 2016.

1. Algorithms
2. Experiments
3. Perspective

Perspectives

1. Enumerate these patterns in higher dimensions.

Perspectives

1. Enumerate these patterns in higher dimensions.
2. Explore and enhance pattern sampling techniques for this new pattern domain.

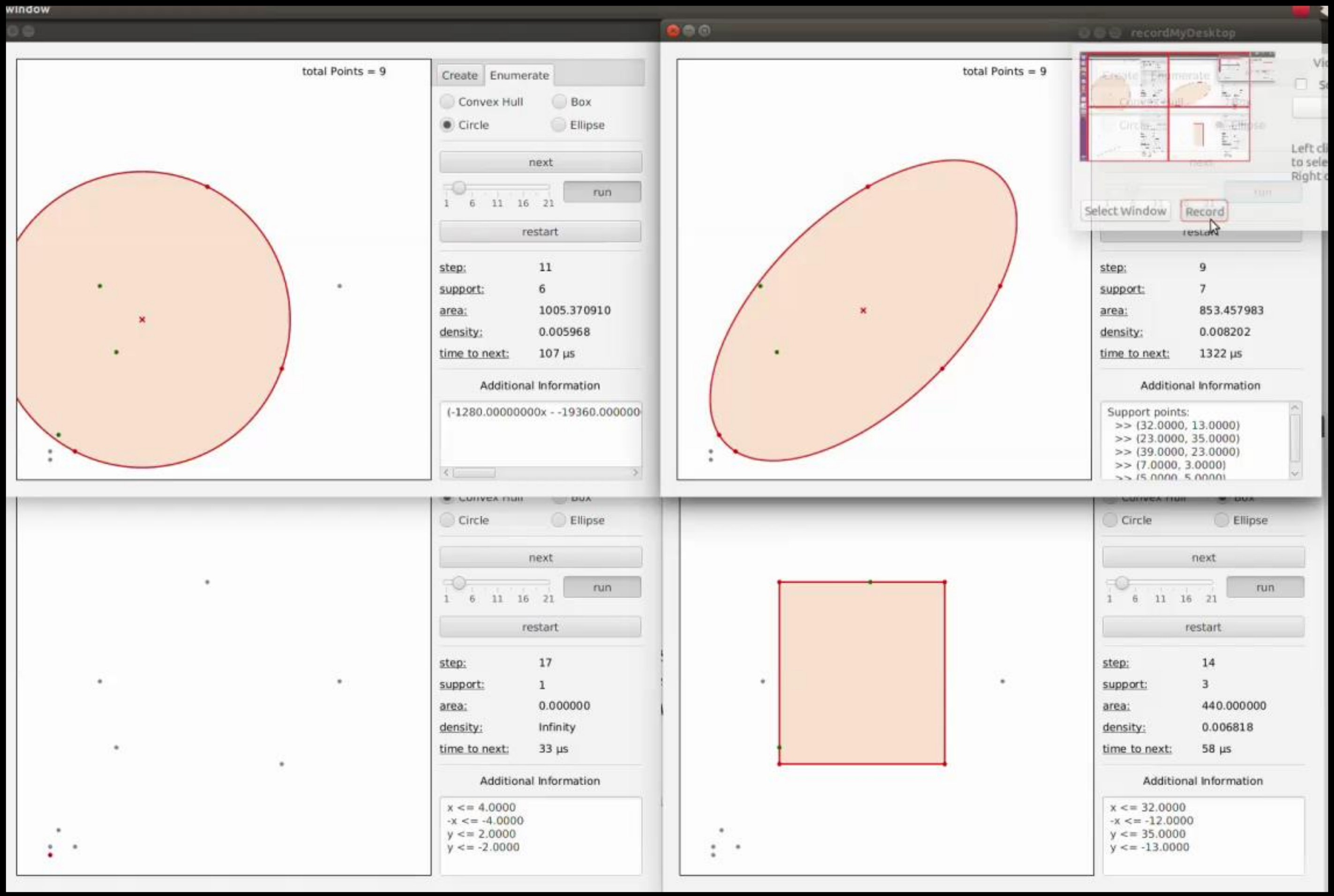
Perspectives

1. Enumerate these patterns in higher dimensions.
2. Explore and enhance pattern sampling techniques for this new pattern domain.
3. Evaluate predictive performance of this new kind of patterns.

Perspectives

1. Enumerate these patterns in higher dimensions.
2. Explore and enhance pattern sampling techniques for this new pattern domain.
3. Evaluate predictive performance of this new kind of patterns.
4. Explore other geometric shapes, yet more expressive than interval patterns, but less expensive than convex polygon patterns.

A glimpse on current works ...



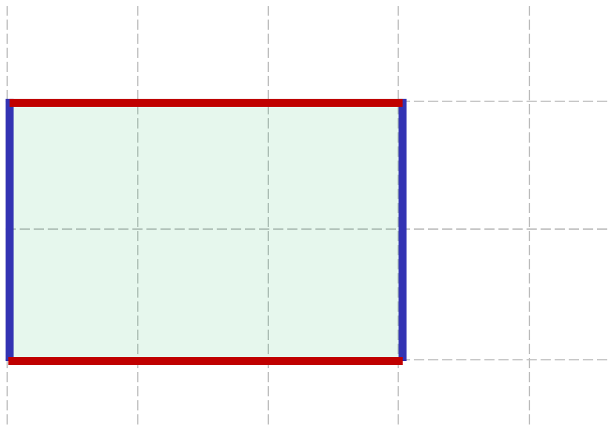
Small discussion (1)

More
Interpretability

More
Expressivity

Small discussion (1)

More Interpretability — More Expressivity

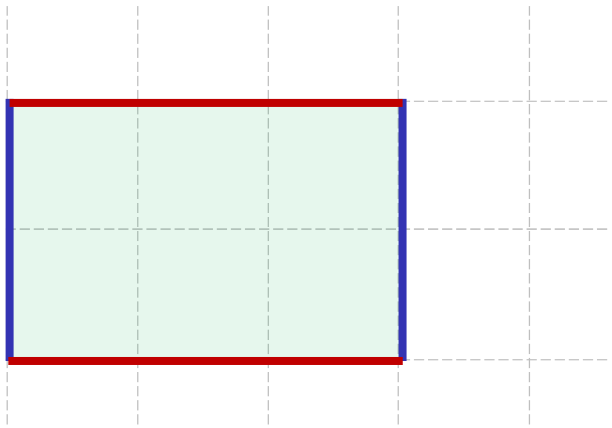


$1 \leq x \leq 4$ ***AND***
$2 \leq y \leq 4$.

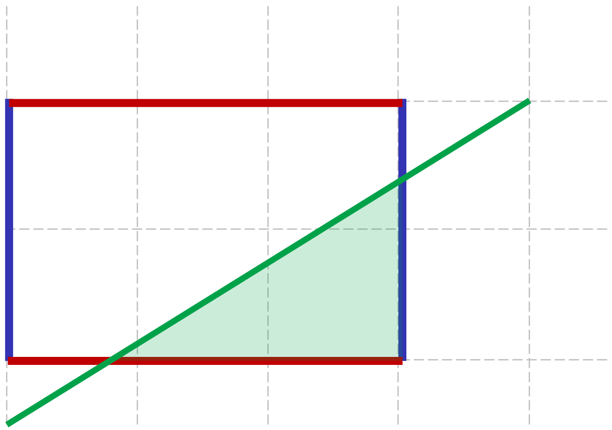
Small discussion (1)

More Interpretability

More Expressivity



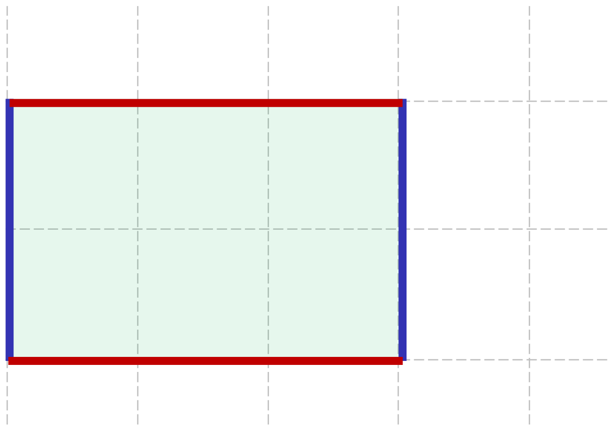
$1 \leq x \leq 4$ ***AND***
$2 \leq y \leq 4$.



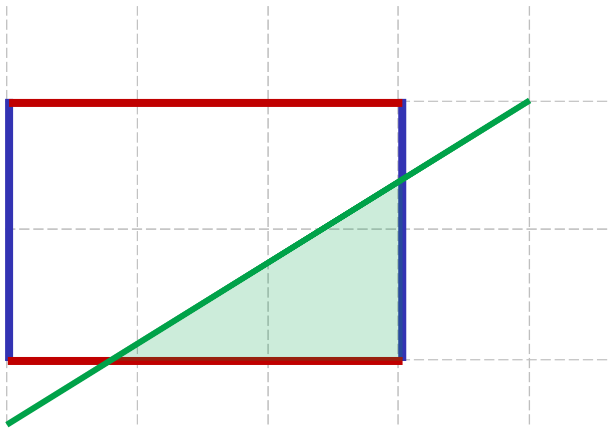
$1 \leq x \leq 4$ ***AND***
$2 \leq y \leq 4$ ***AND***
$\boldsymbol{x \leq y}$.

Small discussion (1)

More Interpretability — More Expressivity



$1 \leq x \leq 4$ $\boldsymbol{AND}$
$2 \leq y \leq 4$.

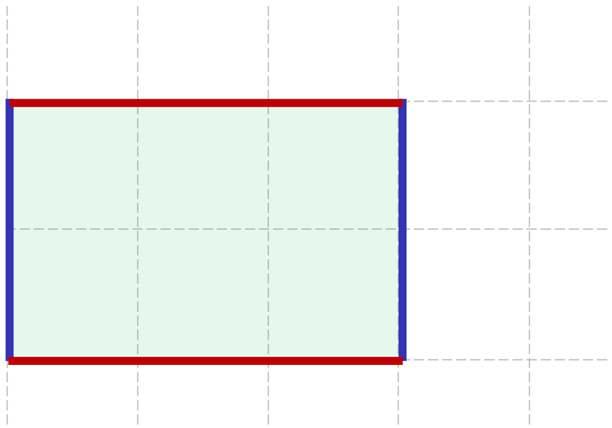


$1 \leq x \leq 4$ $\boldsymbol{AND}$
$2 \leq y \leq 4$ $\boldsymbol{AND}$
$\boldsymbol{x \leq y}$ → Boss salary ≤ Employee salary

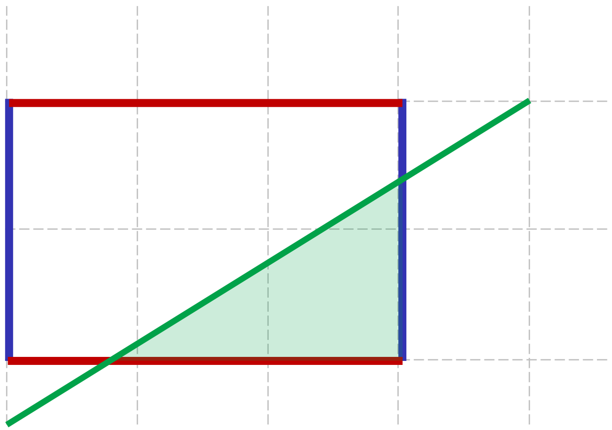
Small discussion (1)

More Interpretability

More Expressivity



$$1 \leq x \leq 4 \ \boldsymbol{AND} \ 2 \leq y \leq 4 .$$



$$1 \leq x \leq 4 \quad \boldsymbol{AND}$$
$$2 \leq y \leq 4 \quad \boldsymbol{AND}$$
$$\boldsymbol{x} \leq \boldsymbol{y} .$$

...

Boss salary ≤ Employee salary

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

Small discussion (2)

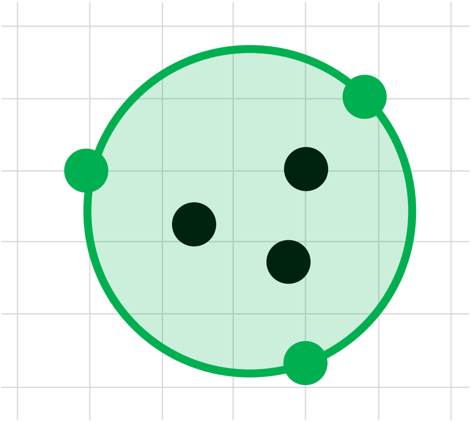
For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^3)$

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

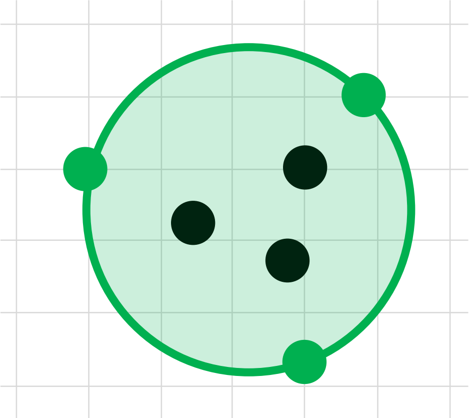
- Circle pattern language is of size $O(n^3)$



3 point suffice to build a circle

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^{\mathbf{3}})$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$

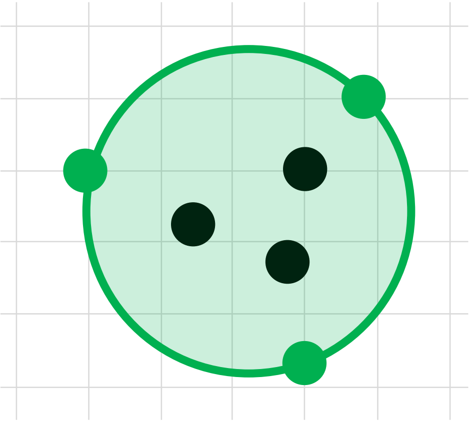


3 point suffice to build a circle

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

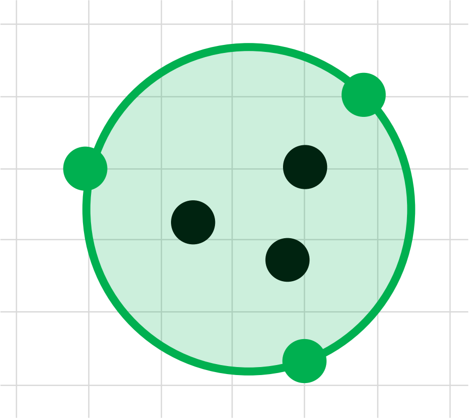
- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$



3 point suffice to build a circle

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

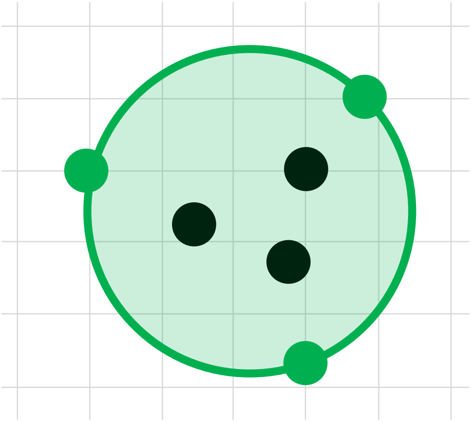
- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$
- Convex polygon pattern language is of size $O(2^n)$



3 point suffice to build a circle

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$
- Convex polygon pattern language is of size $O(2^n)$
- Oriented rectangles pattern language, Enclosing k-gons pattern language (what about uniqueness?)

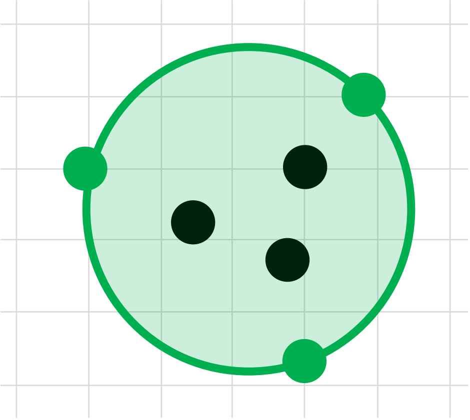


3 point suffice to build a circle

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$
- Convex polygon pattern language is of size $O(2^n)$
- Oriented rectangles pattern language, Enclosing k-gons pattern language (what about uniqueness?)



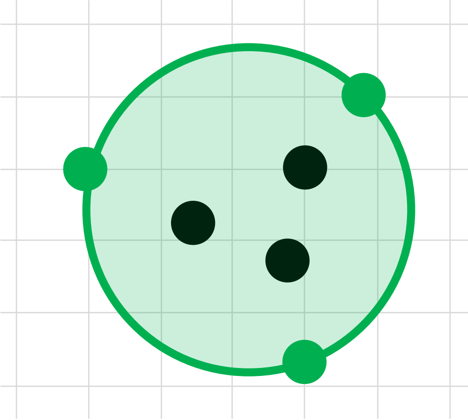
3 point suffice to build a circle

Not all language are comparable.

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$
- Convex polygon pattern language is of size $O(2^n)$
- Oriented rectangles pattern language, Enclosing k-gons pattern language (what about uniqueness?)



3 point suffice to build a circle

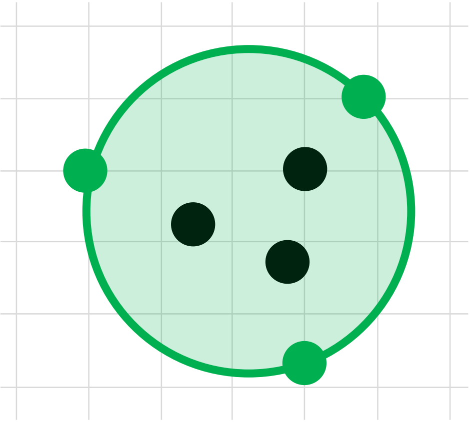
Not all language are comparable.

- **Convex polygon pattern language** subsume **all convex forms based language.**

Small discussion (2)

For finite subsets of $\mathbb{R}^2$ of size n we have **(minimum area)**:

- Circle pattern language is of size $O(n^3)$
- Rectangle (axis-parallel) pattern language is of size $O(n^4)$
- Ellipsoids pattern language is of size $O(n^5)$
- Convex polygon pattern language is of size $O(2^n)$
- Oriented rectangles pattern language, Enclosing k-gons pattern language (what about uniqueness?)



3 point suffice to build a circle

Not all language are comparable.

- **Convex polygon pattern language** subsume **all convex forms based language.**
- **Rectangle (axis-parallel) pattern language** is not comparable with **ellipsoids pattern language**

Thank you for your attention.
Questions ?

Contact: aimene.Belfodil@insa-lyon.fr
Materials: https://github.com/BelfodilAimene/MiningConvexPolygonPatterns
Paper: https://www.ijcai.org/proceedings/2017/0197.pdf