Two-Sided Boundary Labeling with Adjacent Sides

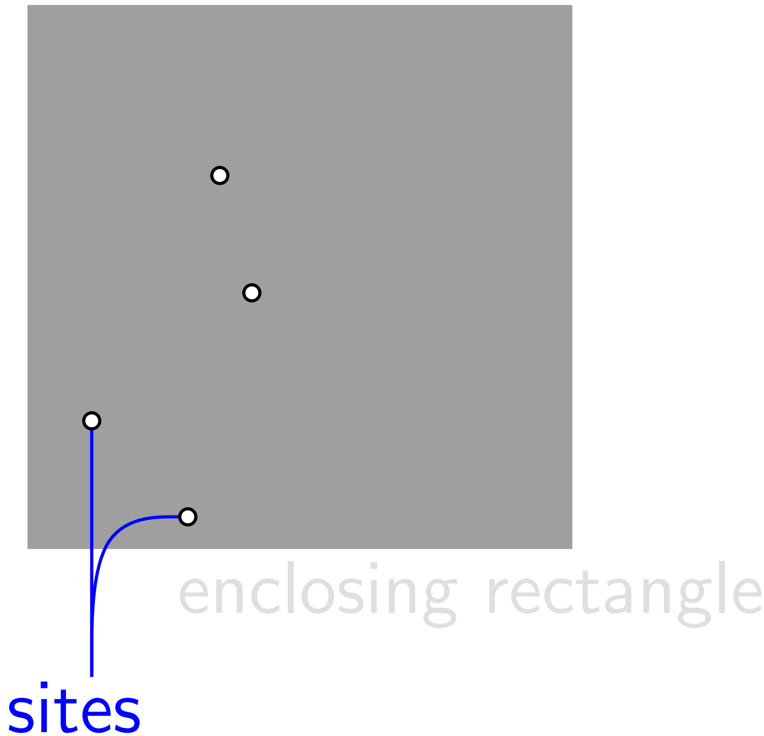
Philipp Kindermann
Lehrstuhl für Informatik I
Universität Würzburg

Joint work with
Benjamin Niedermann, Ignaz Rutter, Marcus Schaefer,
André Schulz & Alexander Wolff

Problem Statement

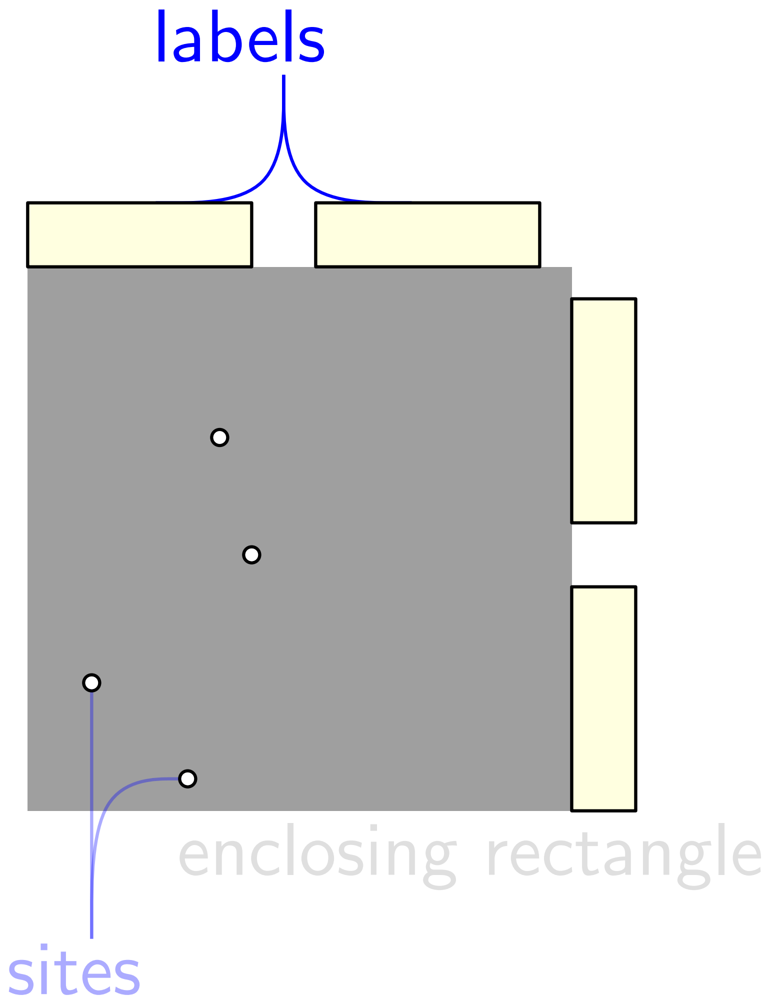
enclosing rectangle

Problem Statement



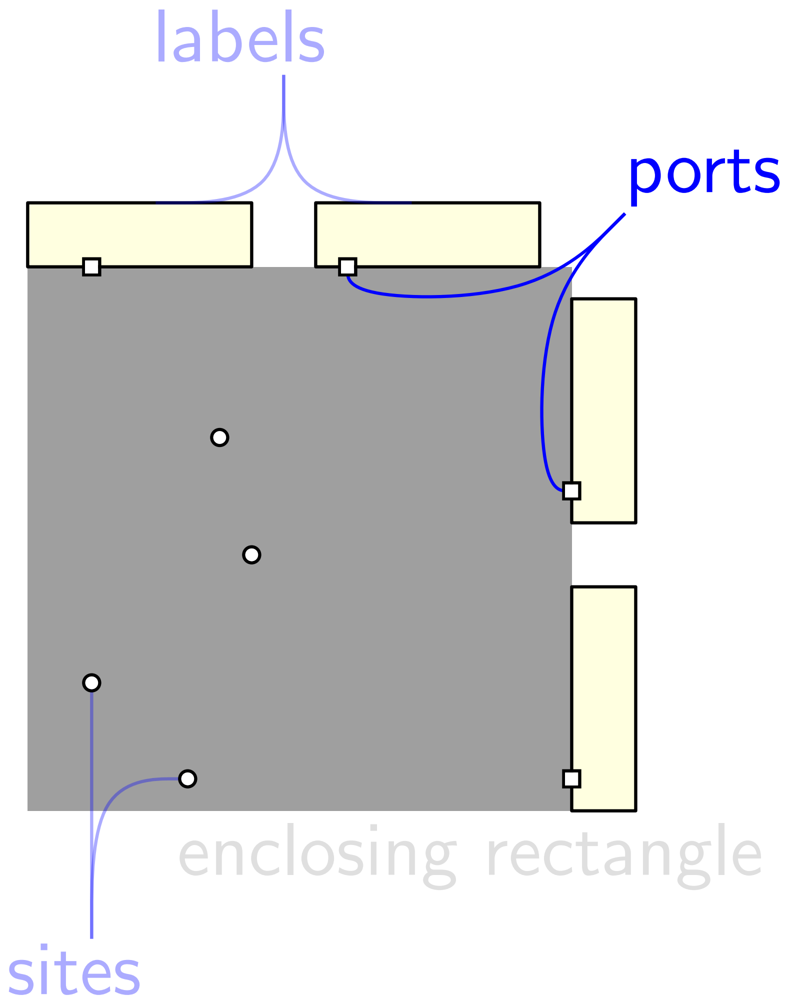
- *n sites* in general position

Problem Statement



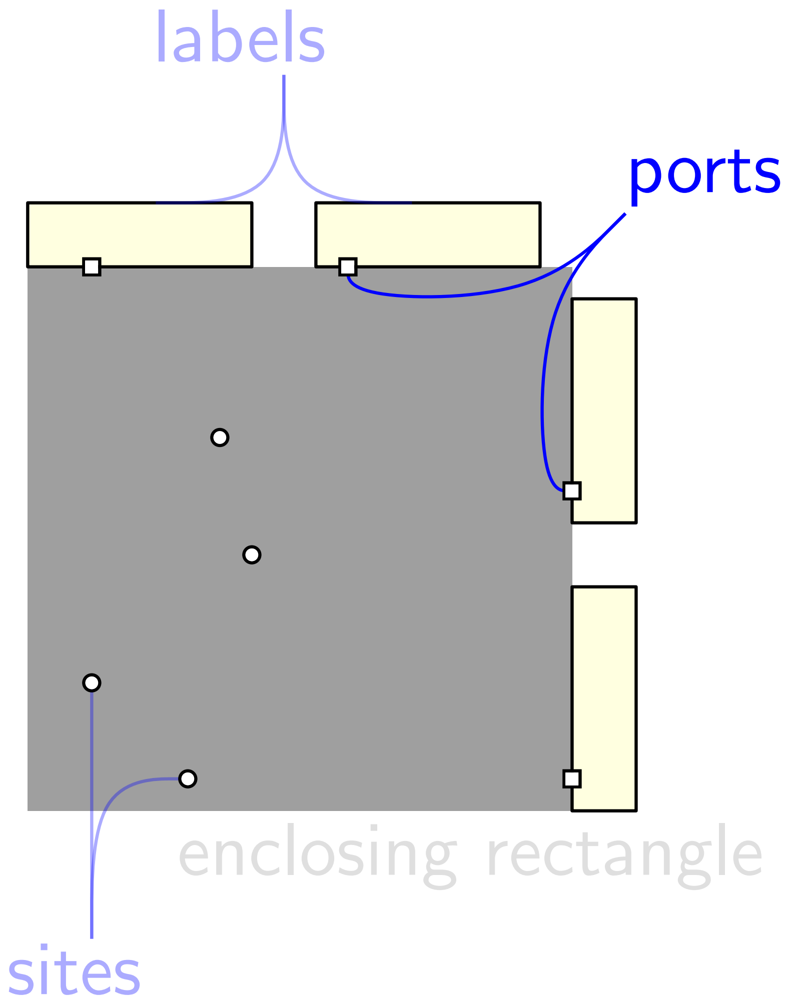
- *n sites* in general position
- *labels* on the boundary

Problem Statement



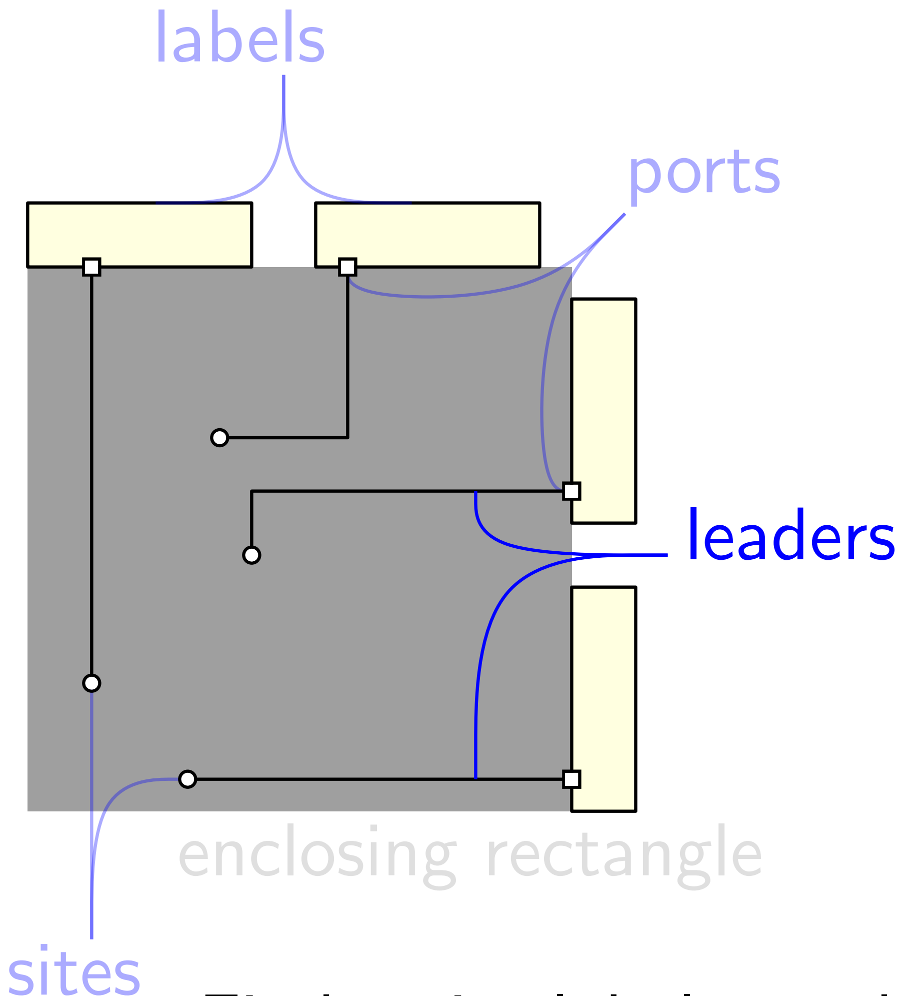
- *n sites* in general position
- *labels* on the boundary
- *ports* are fixed or sliding

Problem Statement



- *n sites* in general position
- *labels* on the boundary
- *ports* are fixed or sliding

Problem Statement

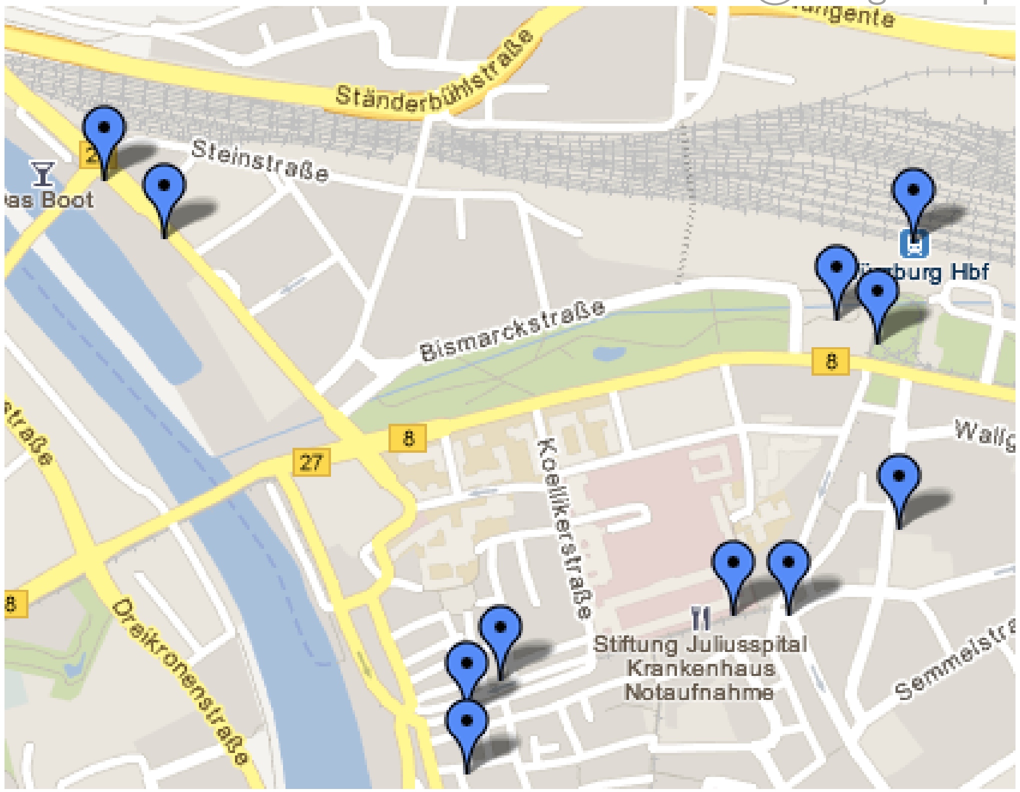


- *n sites* in general position
- *labels* on the boundary
- *ports* are fixed or sliding

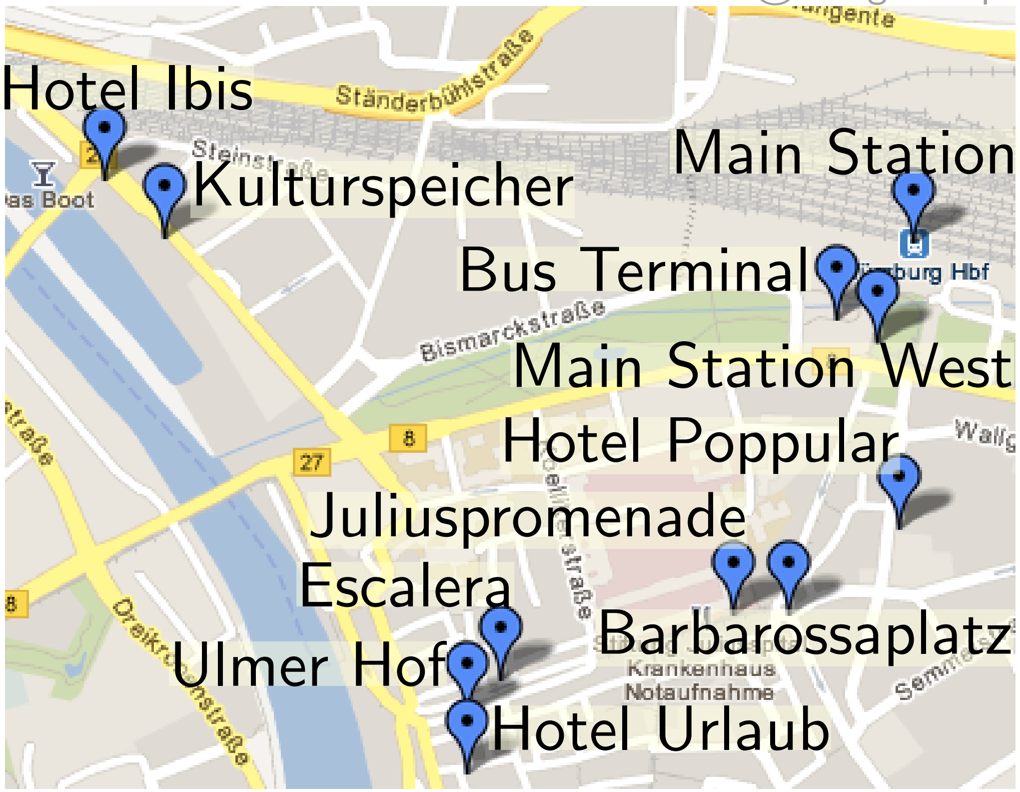
Find a site-label matching with *crossing-free* leaders.

Why Boundary Labeling?

©Google Maps

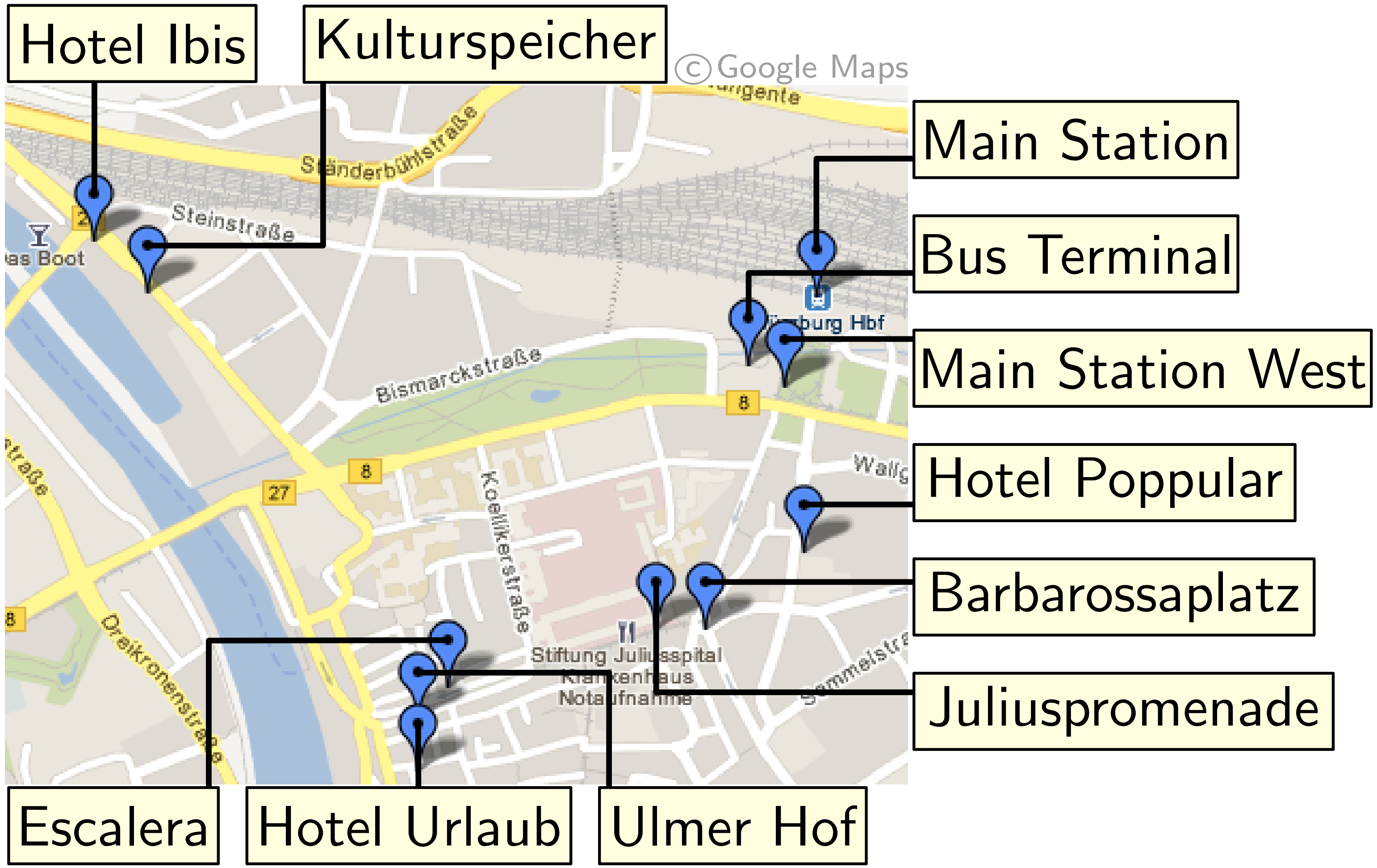


Why Boundary Labeling?



©Google Maps

Why Boundary Labeling?

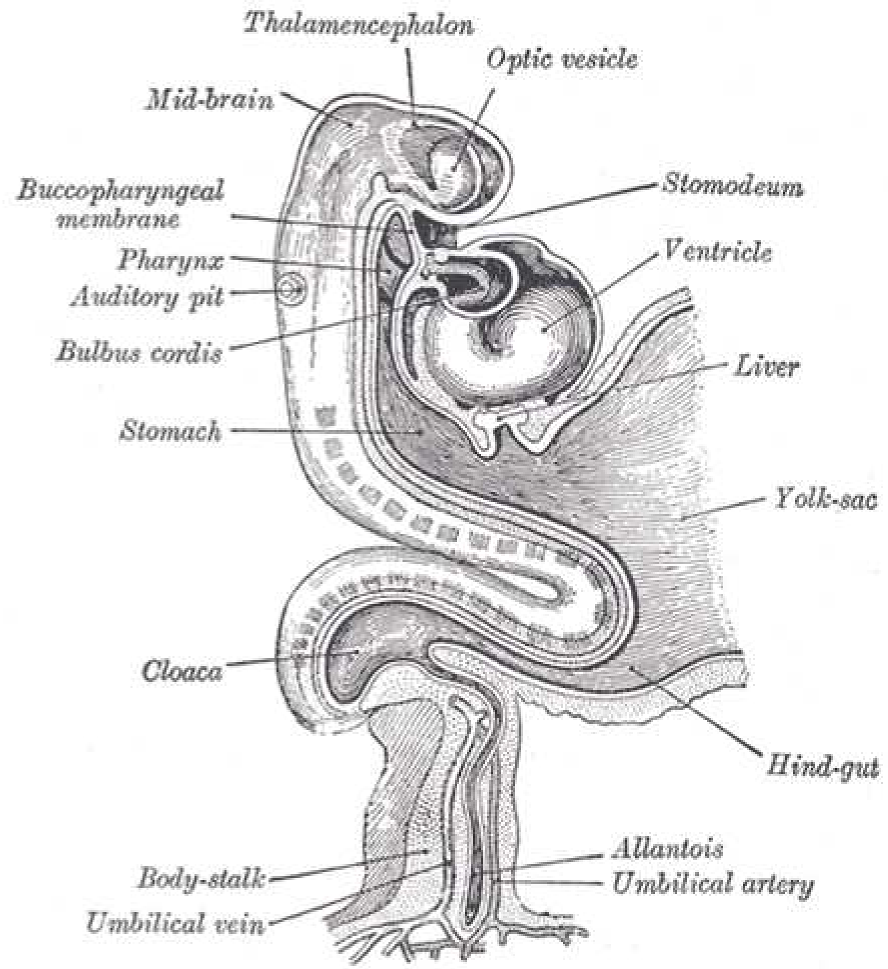


Why Boundary Labeling?

©DW-TV

Why Boundary Labeling?

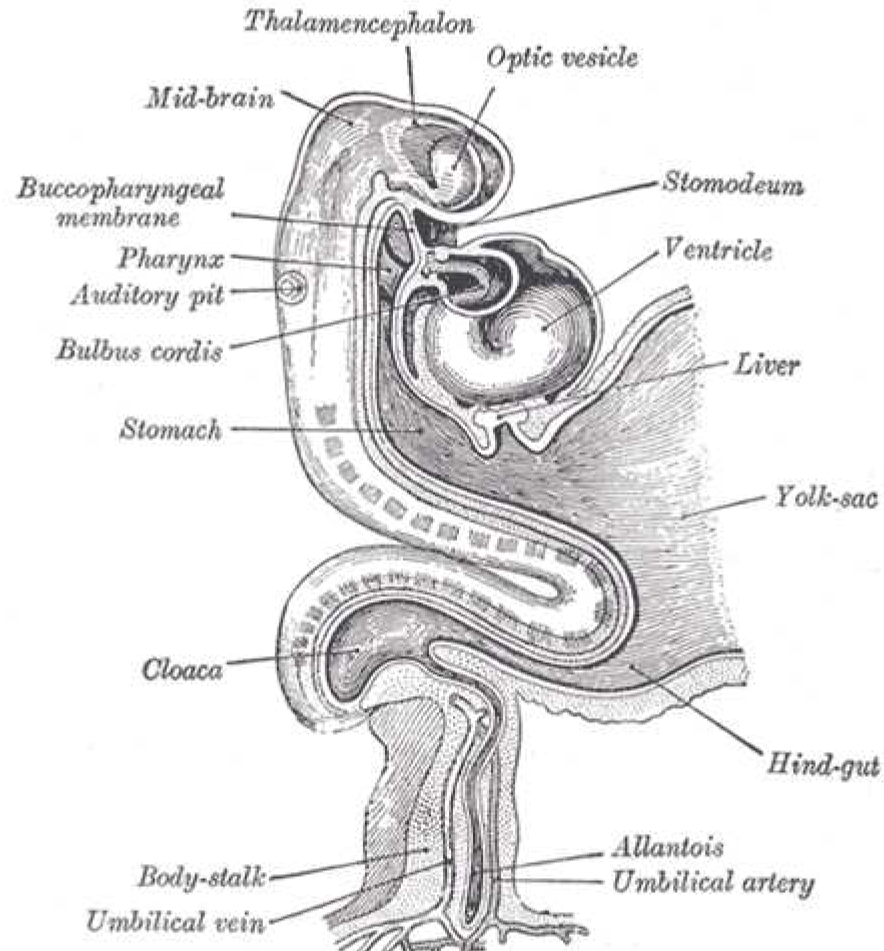
©DW-TV



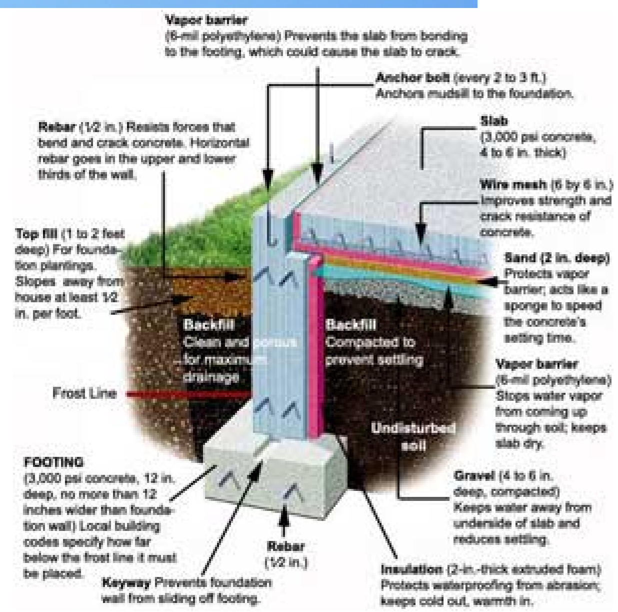
Henry Vandyke Carter, via Wikimedia Commons

Why Boundary Labeling?

©DW-TV

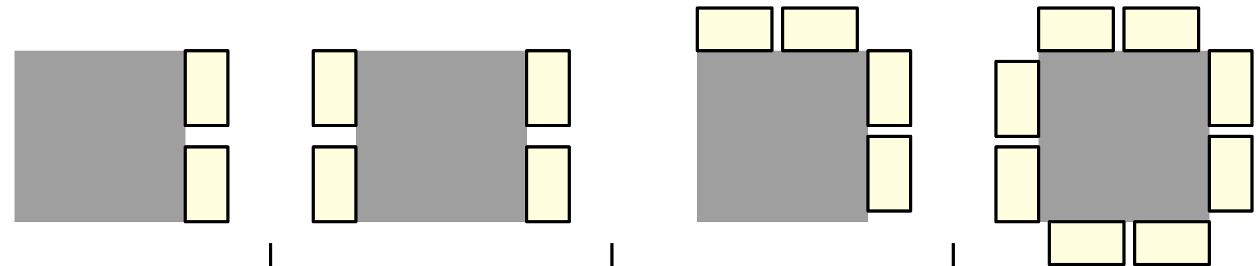


Henry Vandyke Carter, via Wikimedia Commons



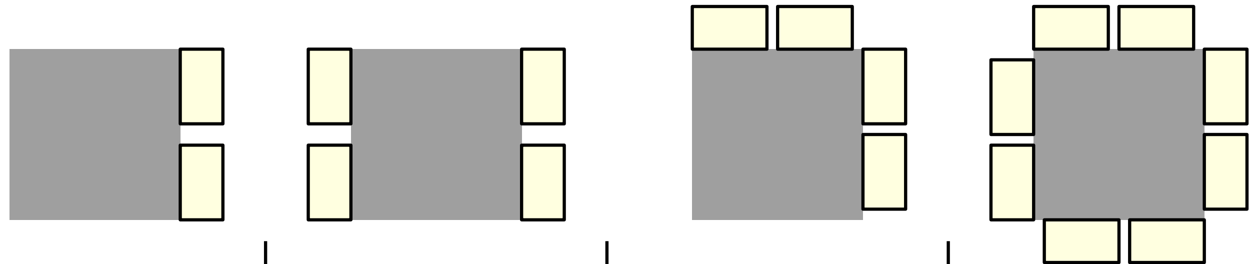
©friv5games.me

Previous Work

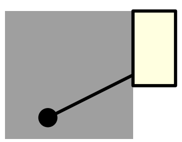


sides / style	1	2 (opp.)	2 (adj.)	4

Previous Work

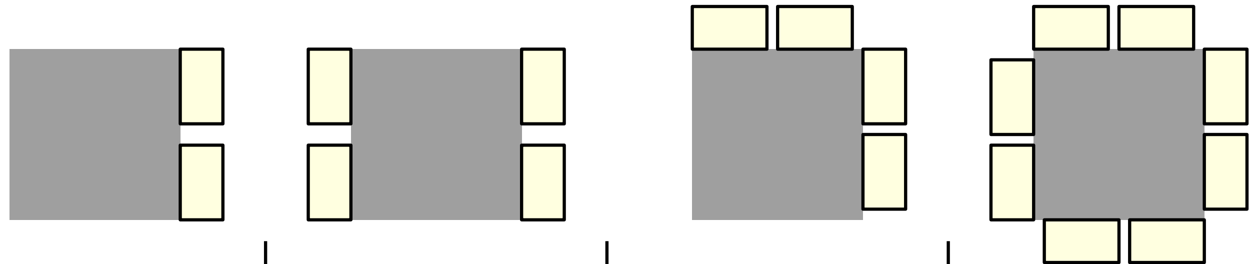


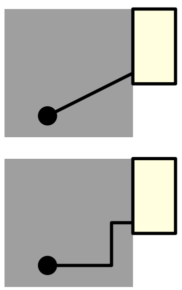
style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$ [1]			



[1] Bekos et al. CGTA'07

Previous Work

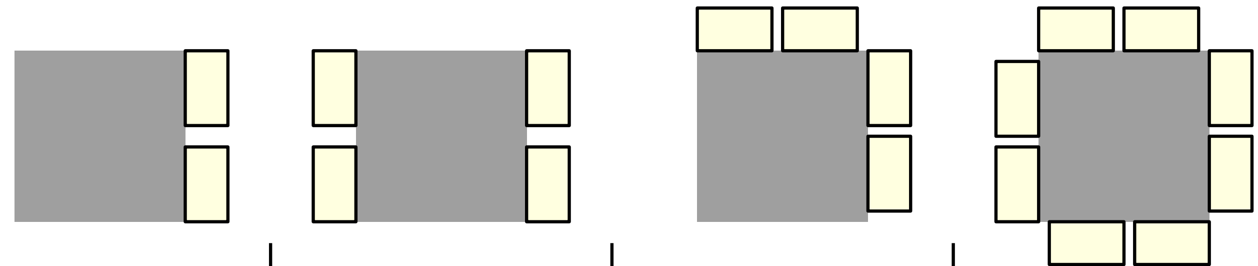


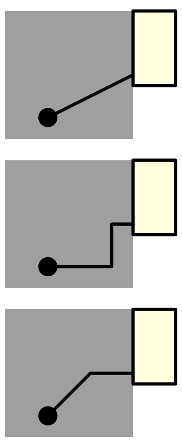


style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$ [1]			
opo	$O(n \log n)$ [1]	$O(n^2)$ [1]	$O(n^2 \log^3 n)$ [1]	

[1] Bekos et al. CGTA'07

Previous Work





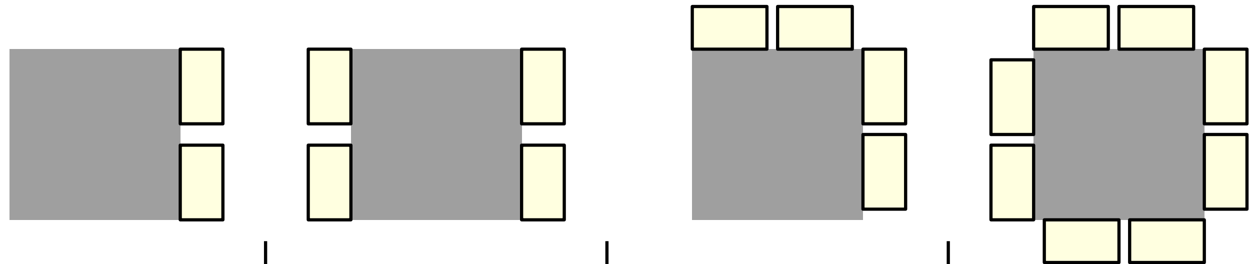
style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$ [1]			
opo	$O(n \log n)$ [1]	$O(n^2)$ [1]	$O(n^2 \log^3 n)$ [1]	
do/pd	$O(n^2)$ [2]	$O(n^3)$ [3]		

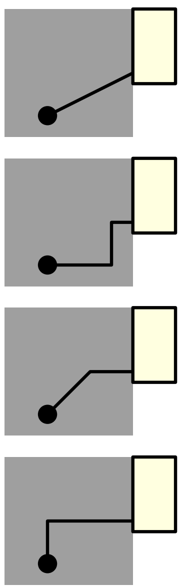
[1] Bekos et al. CGTA'07

[2] Benkert et al. JGAA'09

[3] Bekos et al. Algorithmica'10

Previous Work





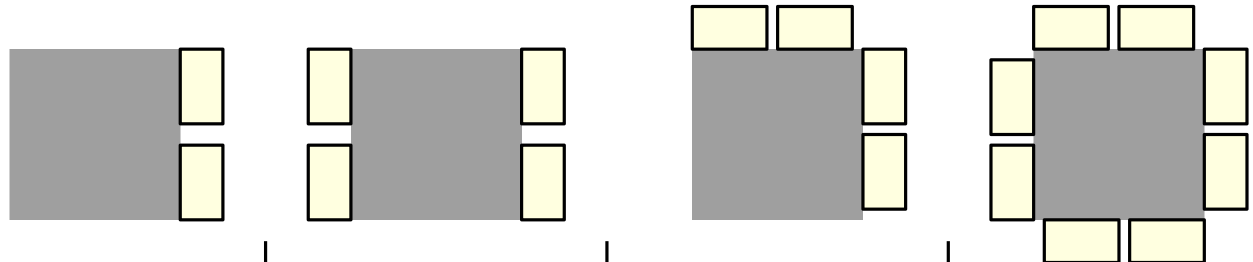
style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$[1]			
opo	$O(n \log n)$[1]	$O(n^2)$[1]	$O(n^2 \log^3 n)$[1]	
do/pd	$O(n^2)$[2]	$O(n^3)$[3]		
po	$O(n \log n)$[2]	$O(n^2)$[1]		

[1] Bekos et al. CGTA'07

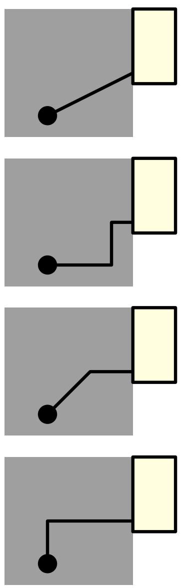
[2] Benkert et al. JGAA'09

[3] Bekos et al. Algorithmica'10

Previous Work



style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$ [1]			
opo	$O(n \log n)$ [1]	$O(n^2)$ [1]	$O(n^2 \log^3 n)$ [1]	
do/pd	$O(n^2)$ [2]	$O(n^3)$ [3]		
po	$O(n \log n)$ [2]	$O(n^2)$ [1]	?	?

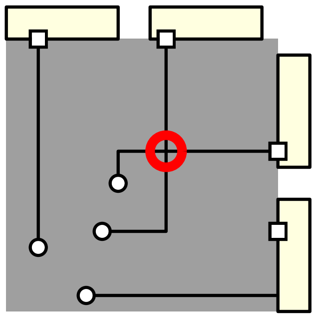


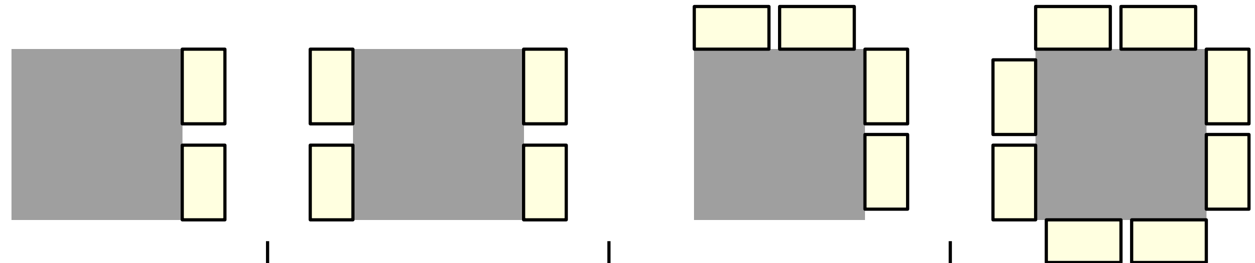
[1] Bekos et al. CGTA'07

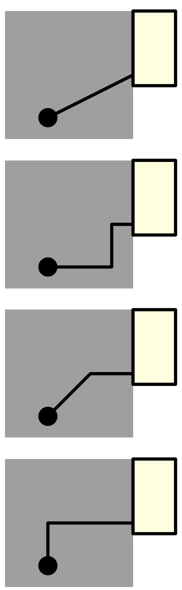
[2] Benkert et al. JGAA'09

[3] Bekos et al. Algorithmica'10

Previous Work



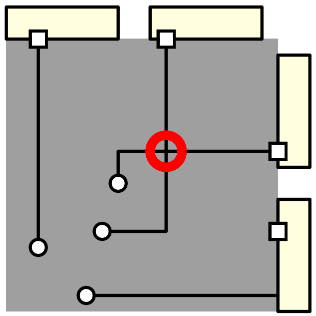


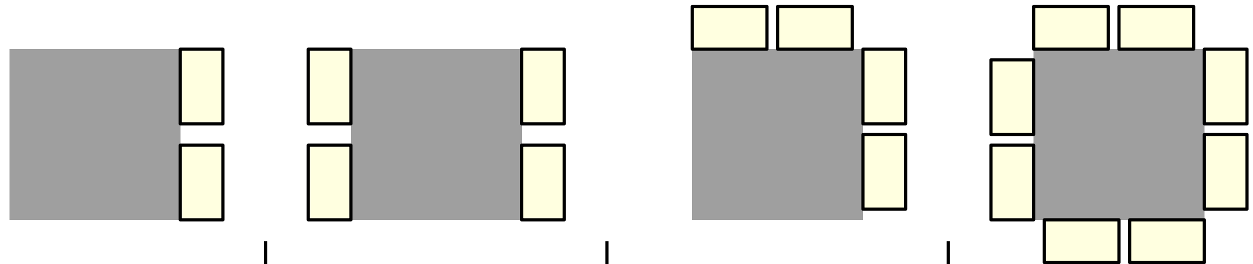


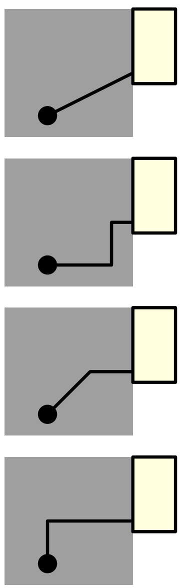
style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$ [1]			
opo	$O(n \log n)$ [1]	$O(n^2)$ [1]	$O(n^2 \log^3 n)$ [1]	
do/pd	$O(n^2)$ [2]	$O(n^3)$ [3]		
po	$O(n \log n)$ [2]	$O(n^2)$ [1]	?	?

[1] Bekos et al. CGTA'07
[2] Benkert et al. JGAA'09
[3] Bekos et al. Algorithmica'10

Previous Work







style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$[1]			
opo	$O(n \log n)$[1]	$O(n^2)$[1]	$O(n^2 \log^3 n)$[1]	
do/pd	$O(n^2)$[2]	$O(n^3)$[3]		
po	$O(n \log n)$[2]	$O(n^2)$[1]	?	?
	2-Approximation on number of bends[4]			

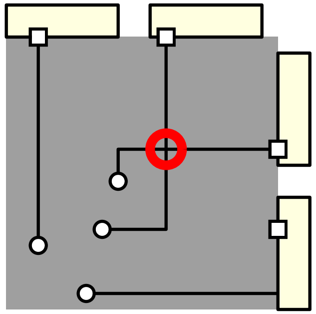
[1] Bekos et al. CGTA'07

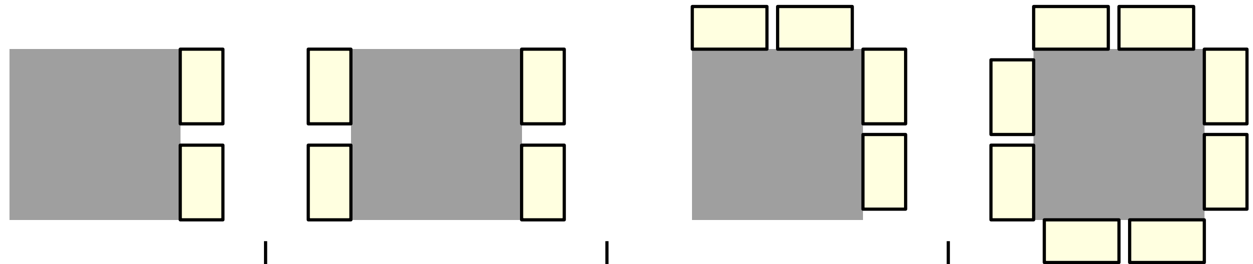
[2] Benkert et al. JGAA'09

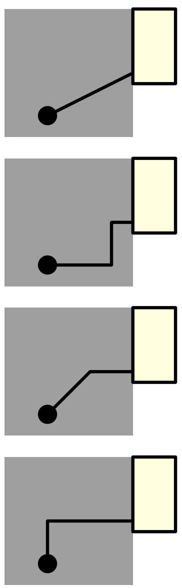
[3] Bekos et al. Algorithmica'10

[4] Speckmann & Verbeek LATIN'10

Previous Work







style \ sides	1	2 (opp.)	2 (adj.)	4
s	$O(n^{2+\epsilon})$[1]			
opo	$O(n \log n)$[1]	$O(n^2)$[1]	$O(n^2 \log^3 n)$[1]	
do/pd	$O(n^2)$[2]	$O(n^3)$[3]		
po	$O(n \log n)$[2]	$O(n^2)$[1]	?	?
	2-Approximation on number of bends[4]			

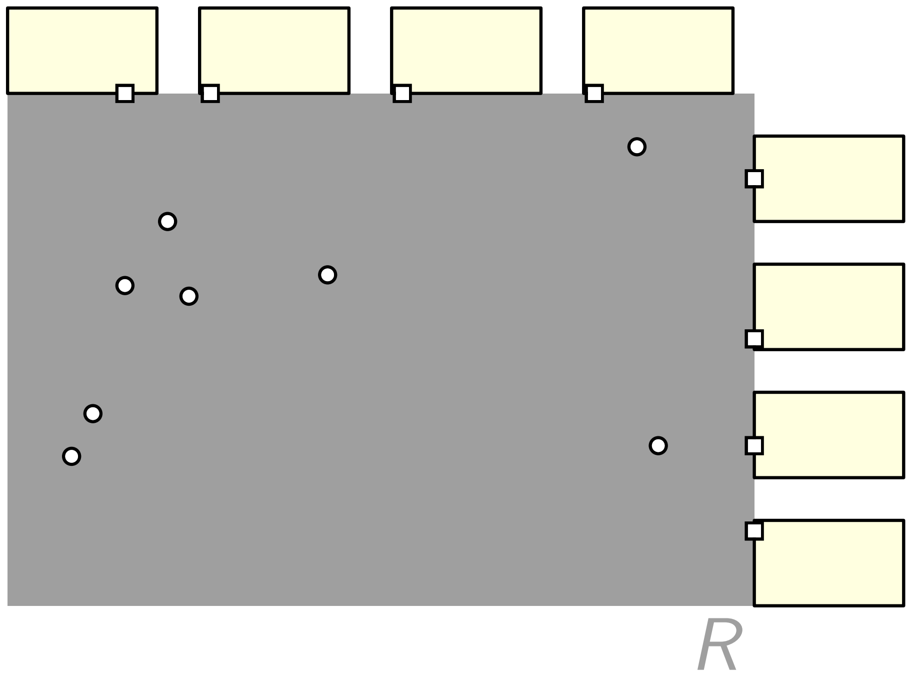
[1] Bekos et al. CGTA'07

[2] Benkert et al. JGAA'09

[3] Bekos et al. Algorithmica'10

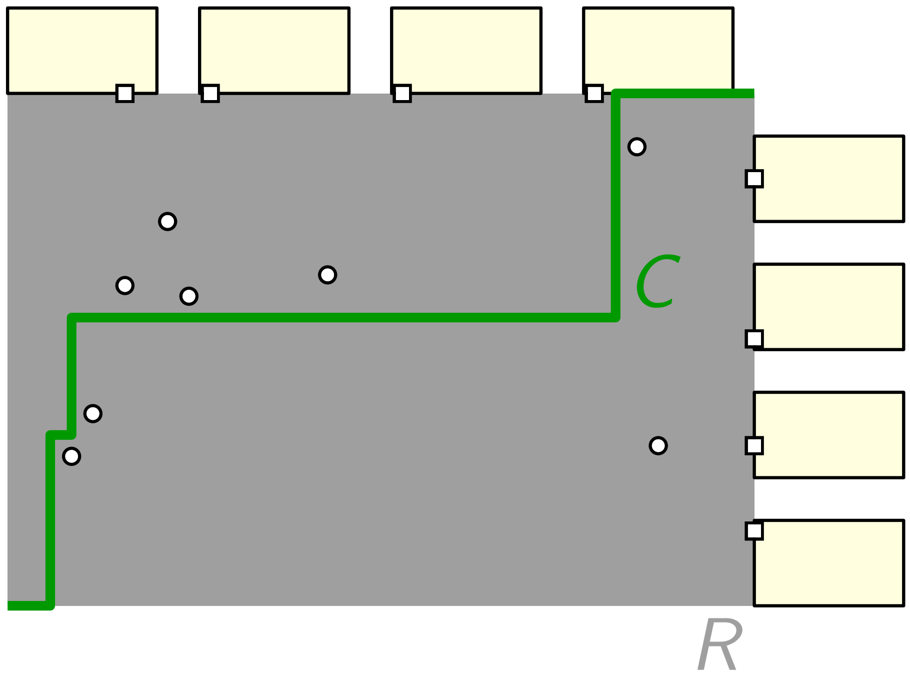
[4] Speckmann & Verbeek LATIN'10

Structure of Planar Solutions



Structure of Planar Solutions

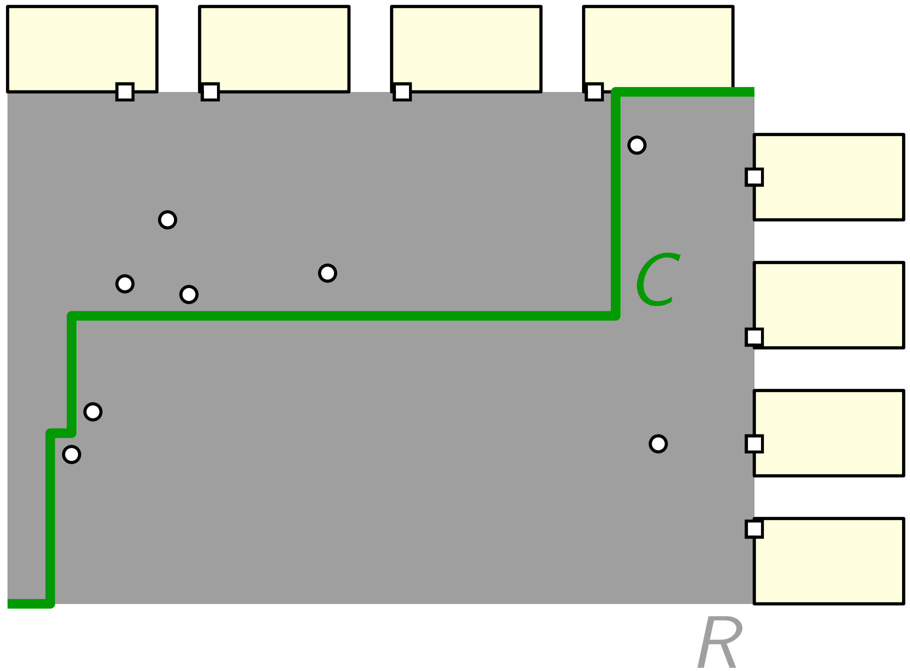
xy-separating curve C



Structure of Planar Solutions

xy-separating curve C

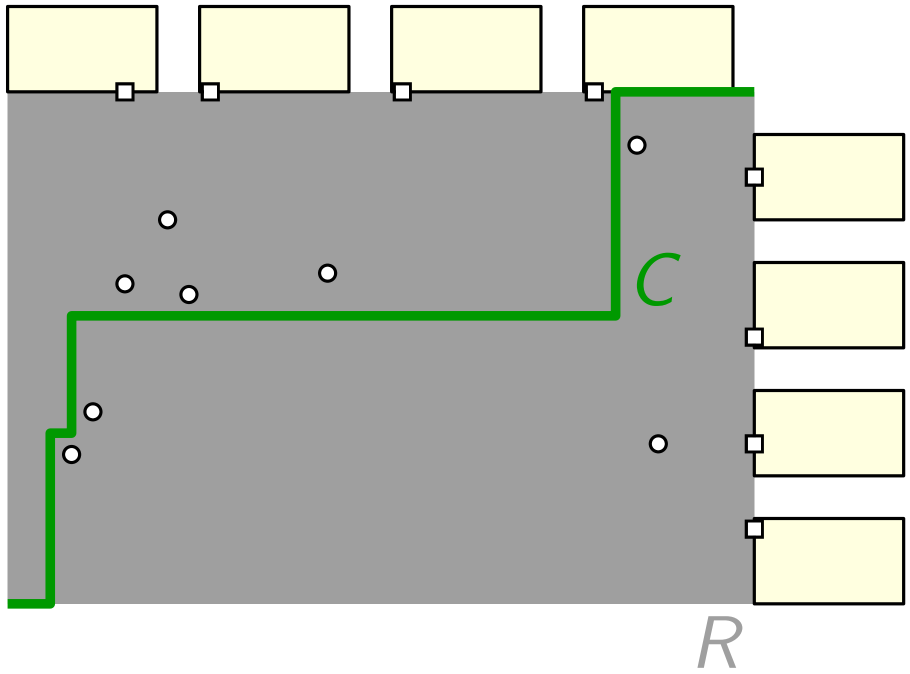
- xy-monotone



Structure of Planar Solutions

xy-separating curve C

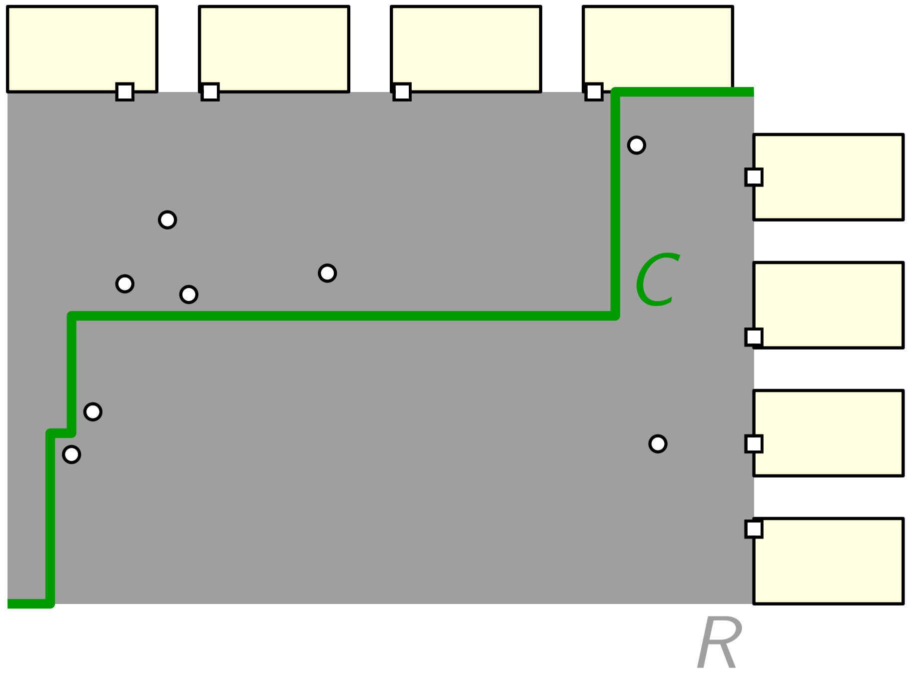
- xy-monotone
- rectilinear



Structure of Planar Solutions

xy-separating curve C

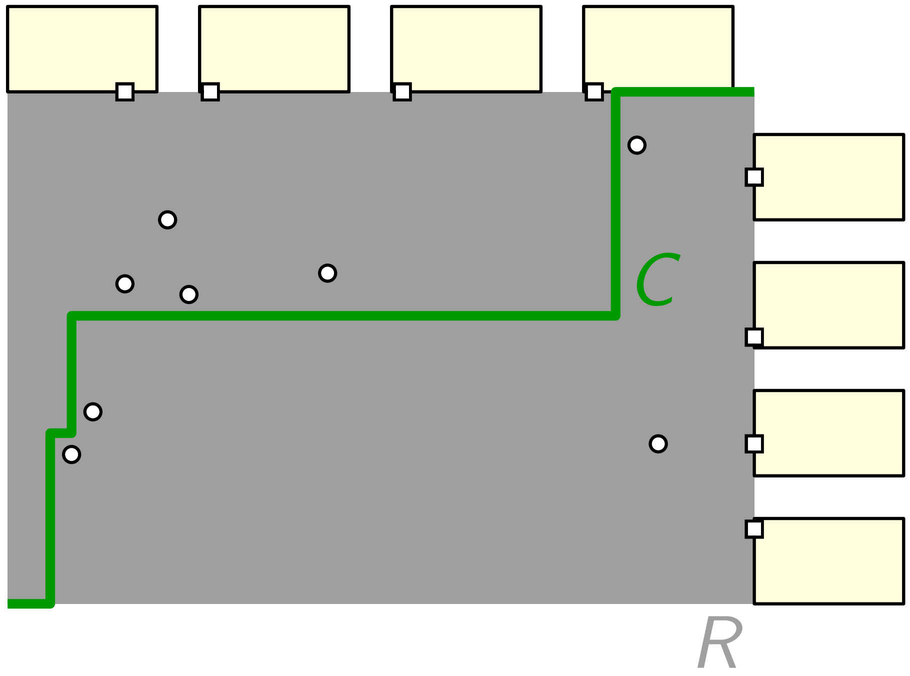
- xy-monotone
- rectilinear
- connects the top-right to the bottom-left corner of R



Structure of Planar Solutions

xy-separating curve C

- xy-monotone
- rectilinear
- connects the top-right to the bottom-left corner of R

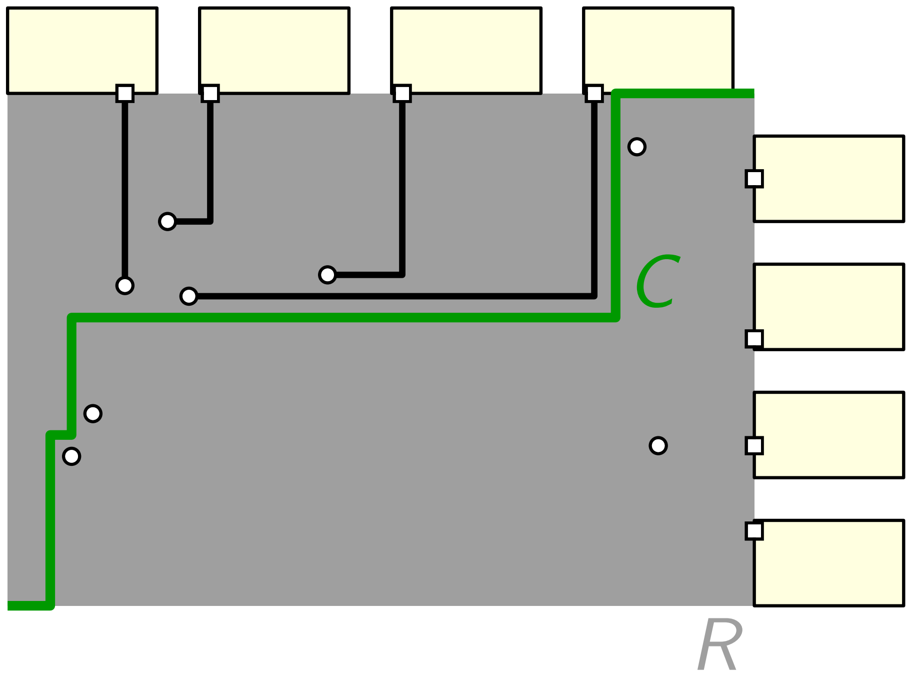


xy-separated solution

Structure of Planar Solutions

xy-separating curve C

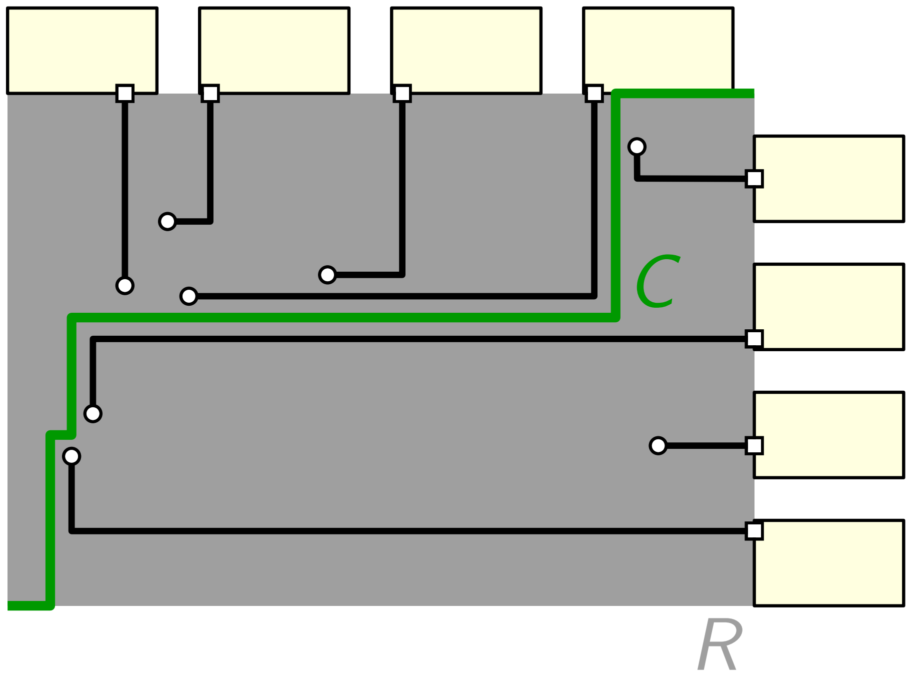
- xy-monotone
- rectilinear
- connects the top-right to the bottom-left corner of R



xy-separated solution

- top sites and leaders lie above C

Structure of Planar Solutions



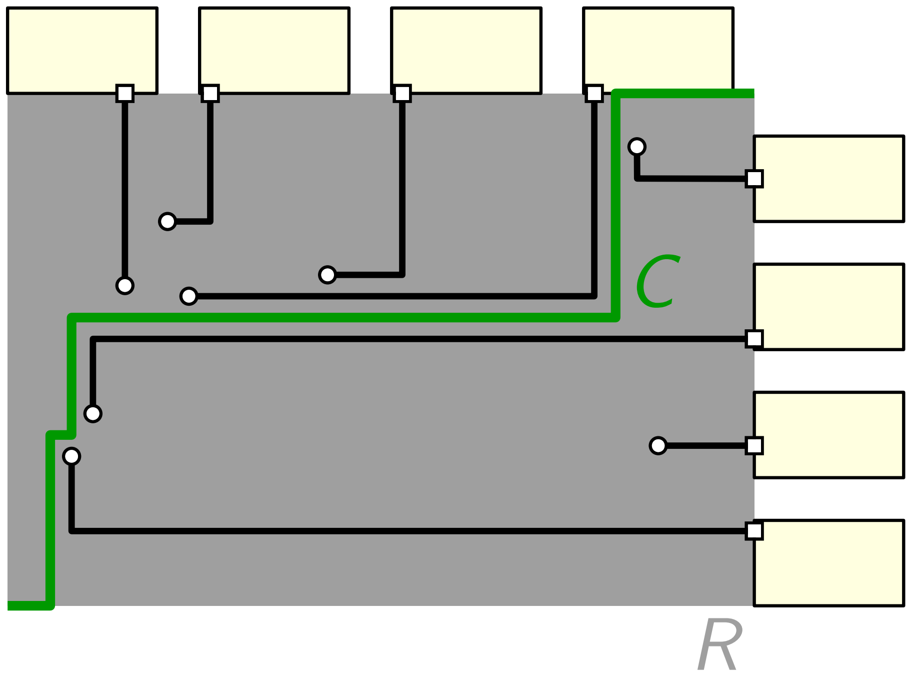
xy-separating curve C

- xy-monotone
- rectilinear
- connects the top-right to the bottom-left corner of R

xy-separated solution

- top sites and leaders lie above C
- right sites and leaders lie below C

Structure of Planar Solutions

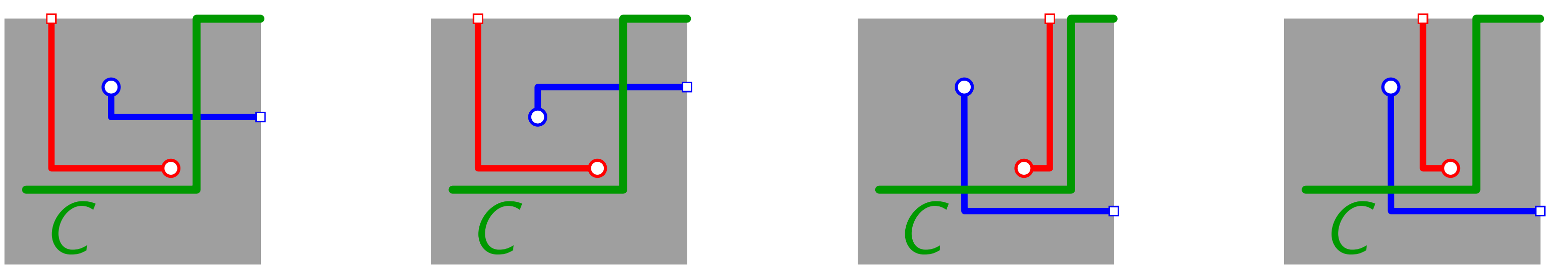


xy-separating curve C

- *xy*-monotone
- rectilinear
- connects the top-right to the bottom-left corner of R

xy-separated solution

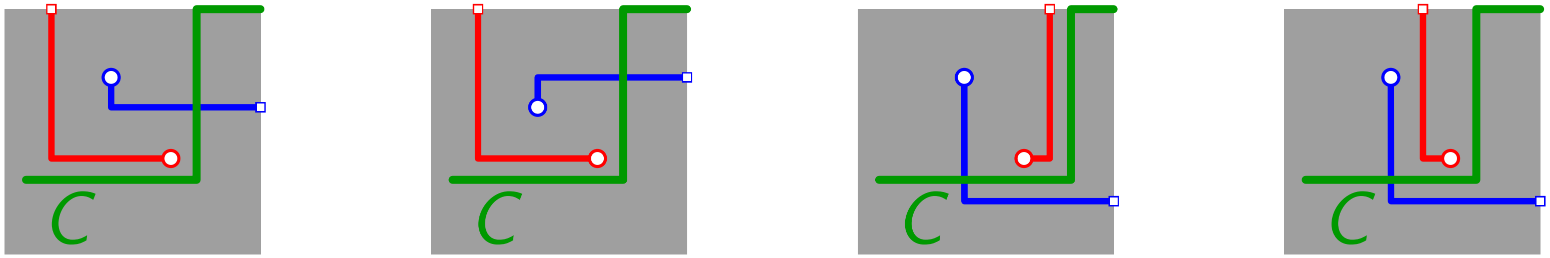
- top sites and leaders lie above C
- right sites and leaders lie below C
- does not contain any of the following patterns



Structure of Planar Solutions

xy-separated solution

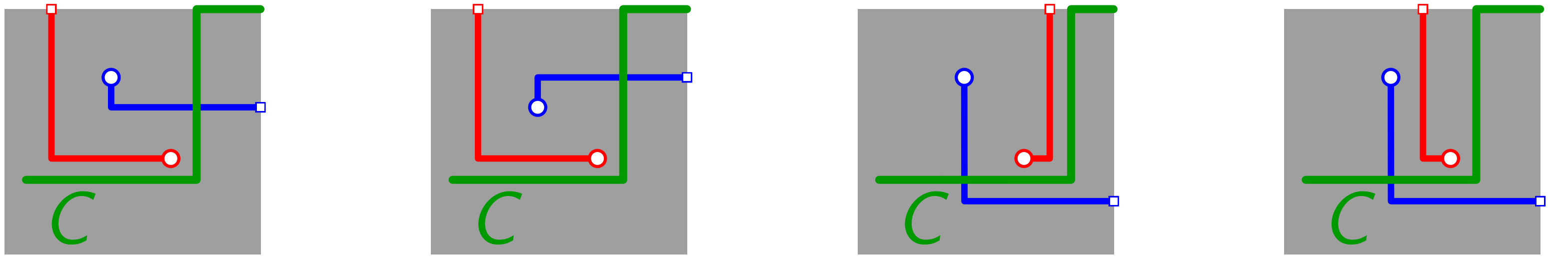
- does not contain any of the following patterns



Structure of Planar Solutions

xy-separated solution

- does not contain any of the following patterns

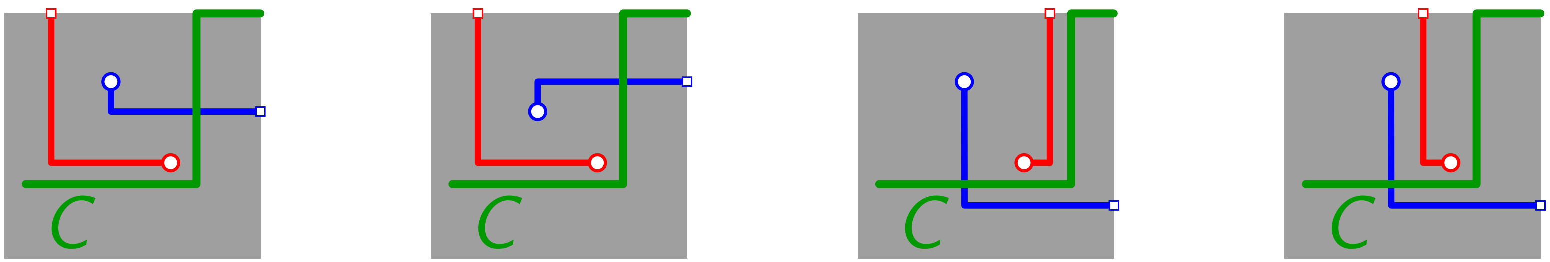


Planar solution $\Rightarrow$ *xy*-separated planar solution

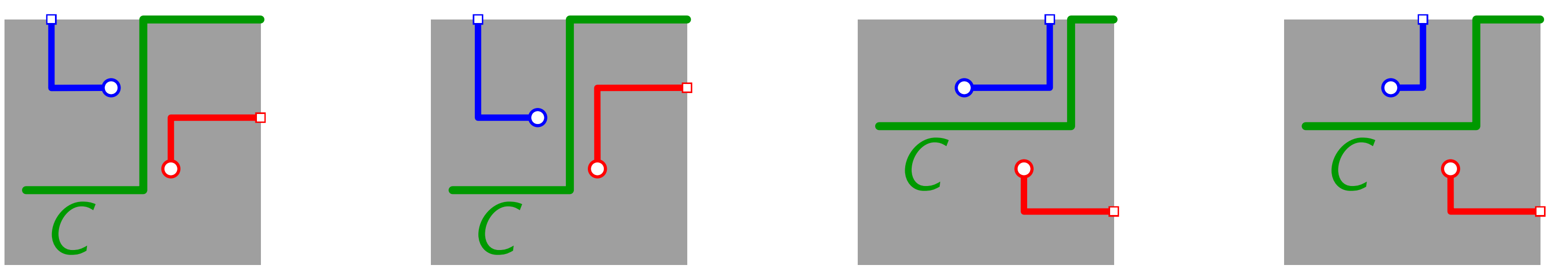
Structure of Planar Solutions

xy-separated solution

- does not contain any of the following patterns



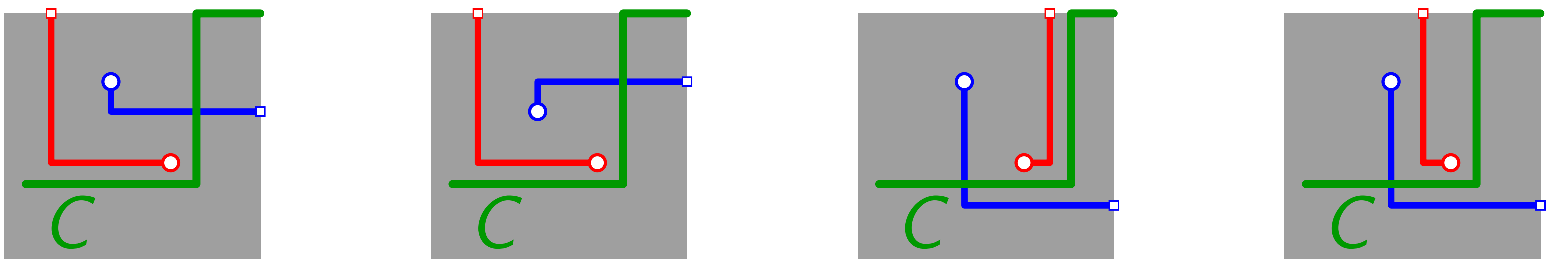
Planar solution $\Rightarrow$ *xy*-separated planar solution



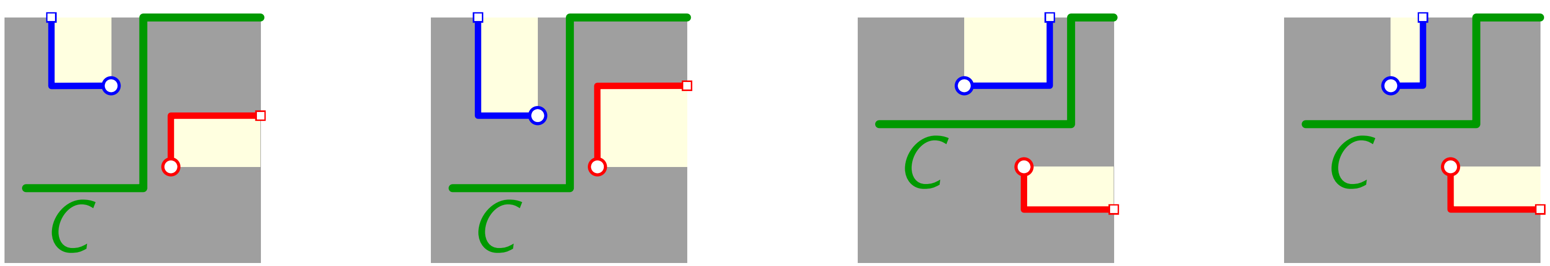
Structure of Planar Solutions

xy-separated solution

- does not contain any of the following patterns



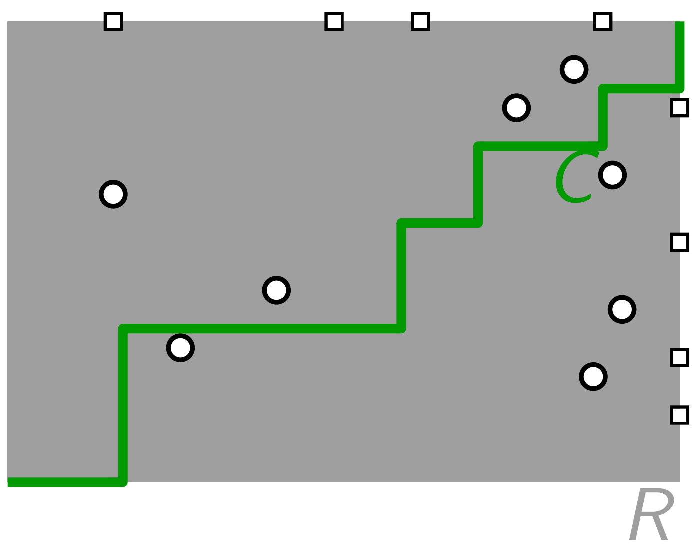
Planar solution $\Rightarrow$ xy-separated planar solution



Eliminate local crossings

The Strip Condition

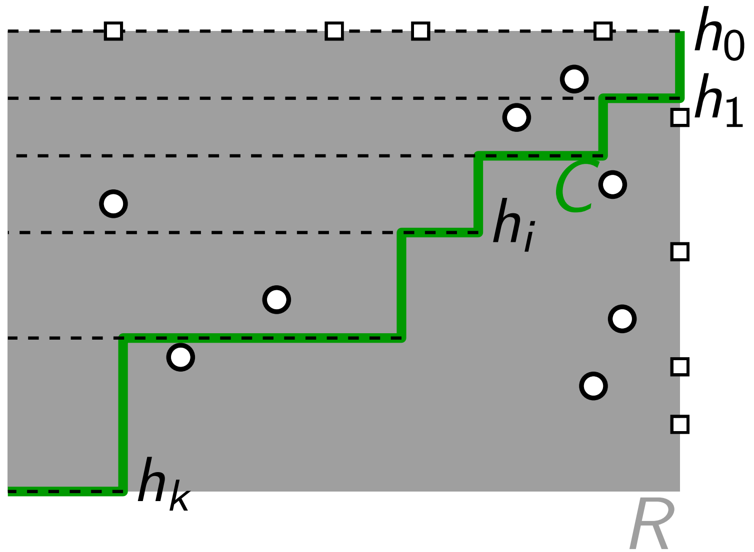
Given: xy-monotone curve C



The Strip Condition

Given: xy-monotone curve C

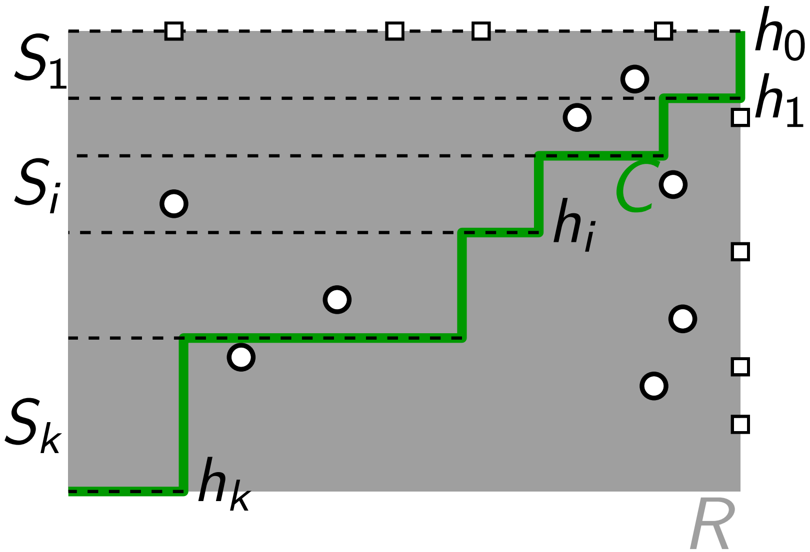
- $h_0, \ldots, h_k$: horizontal segments of C extended to the left



The Strip Condition

Given: xy-monotone curve C

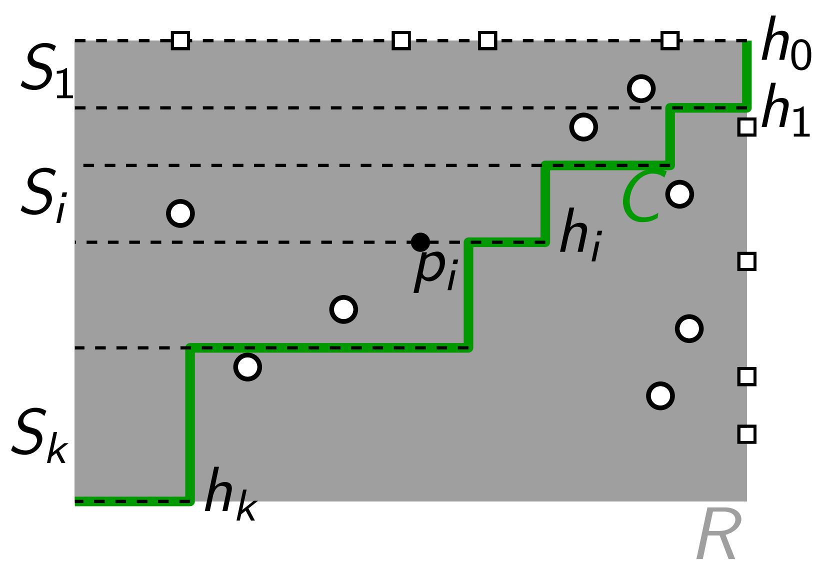
- $h_0, \ldots, h_k$: horizontal segments of C extended to the left
- $S_1, \ldots, S_k$: strips of R partitioned by $h_0, \ldots, h_k$



The Strip Condition

Given: xy-monotone curve C

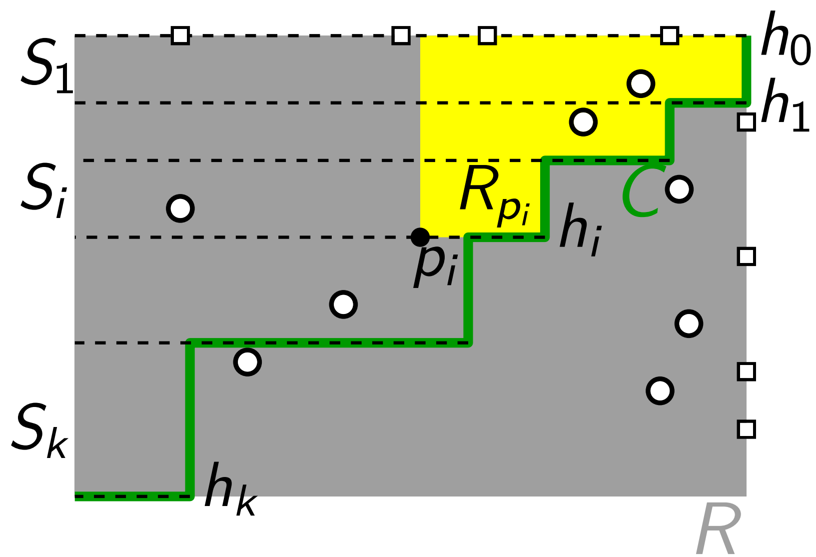
- $h_0, \ldots, h_k$: horizontal segments of C extended to the left
- $S_1, \ldots, S_k$: strips of R partitioned by $h_0, \ldots, h_k$
- p_i: any point on $h_i \setminus C$



The Strip Condition

Given: xy-monotone curve C

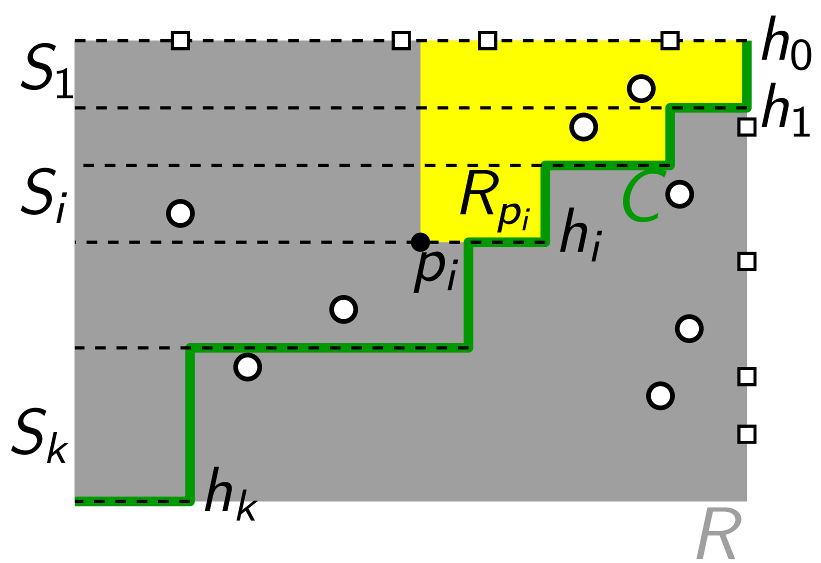
- $h_0, \ldots, h_k$: horizontal segments of C extended to the left
- $S_1, \ldots, S_k$: strips of R partitioned by $h_0, \ldots, h_k$
- p_i: any point on $h_i \setminus C$
- R_{p_i}: polygon spanned by p_i, h_i and C



The Strip Condition

Given: xy-monotone curve C

- $h_0, \ldots, h_k$: horizontal segments of C extended to the left
- $S_1, \ldots, S_k$: strips of R partitioned by $h_0, \ldots, h_k$
- p_i: any point on $h_i \setminus C$
- R_{p_i}: polygon spanned by p_i, h_i and C
- R_{p_i} is *valid*: number of sites $\geq$ number of ports



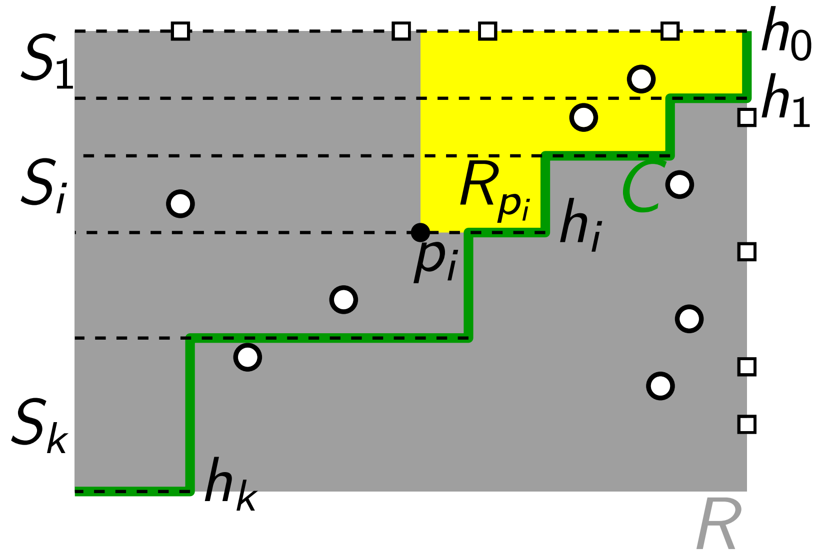
The Strip Condition

Given: xy-monotone curve C

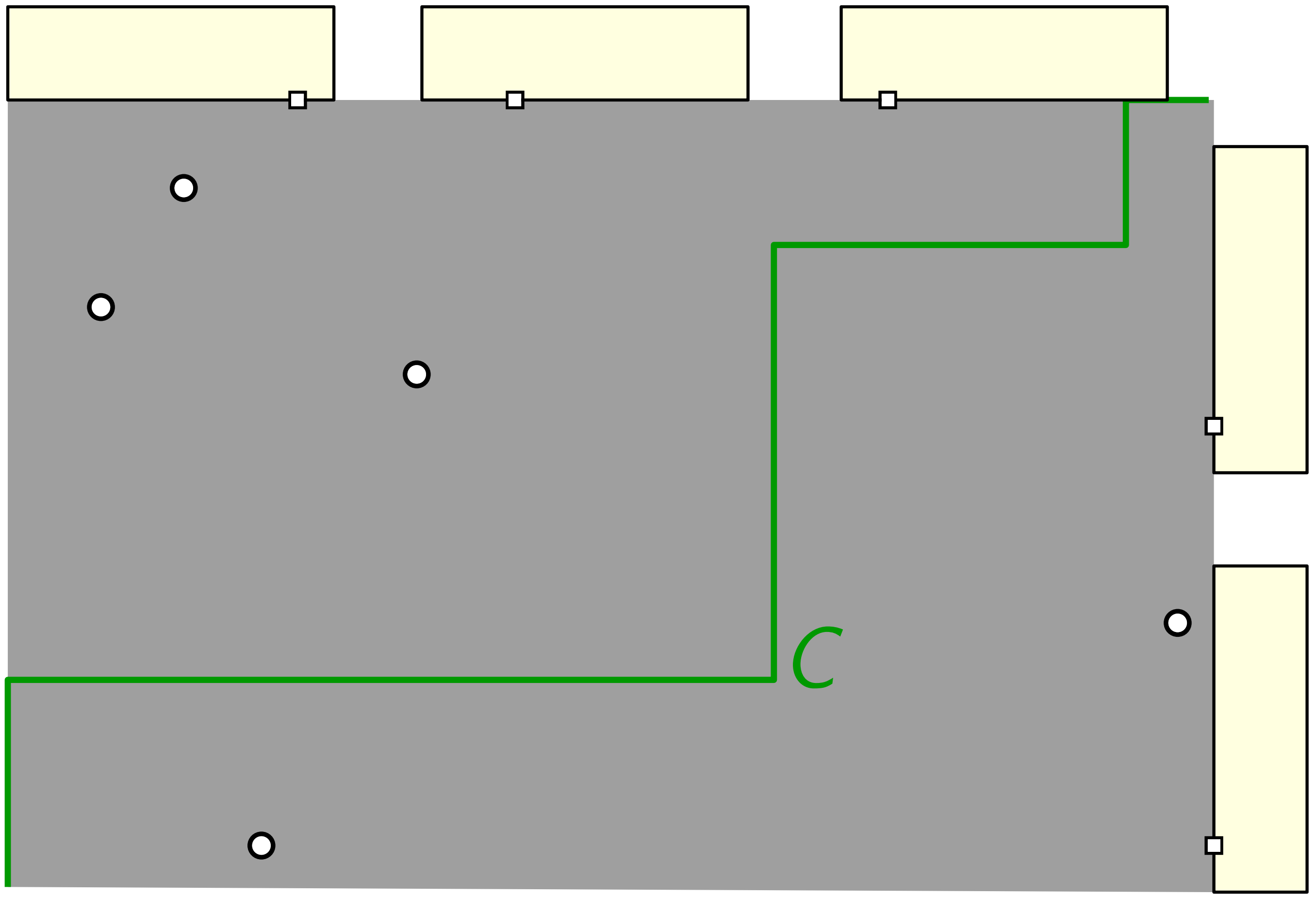
- $h_0, \ldots, h_k$: horizontal segments of C extended to the left
- $S_1, \ldots, S_k$: strips of R partitioned by $h_0, \ldots, h_k$
- p_i: any point on $h_i \setminus C$
- R_{p_i}: polygon spanned by p_i, h_i and C
- R_{p_i} is *valid*: number of sites $\geq$ number of ports

Condition. *strip condition* of S_i is satisfied

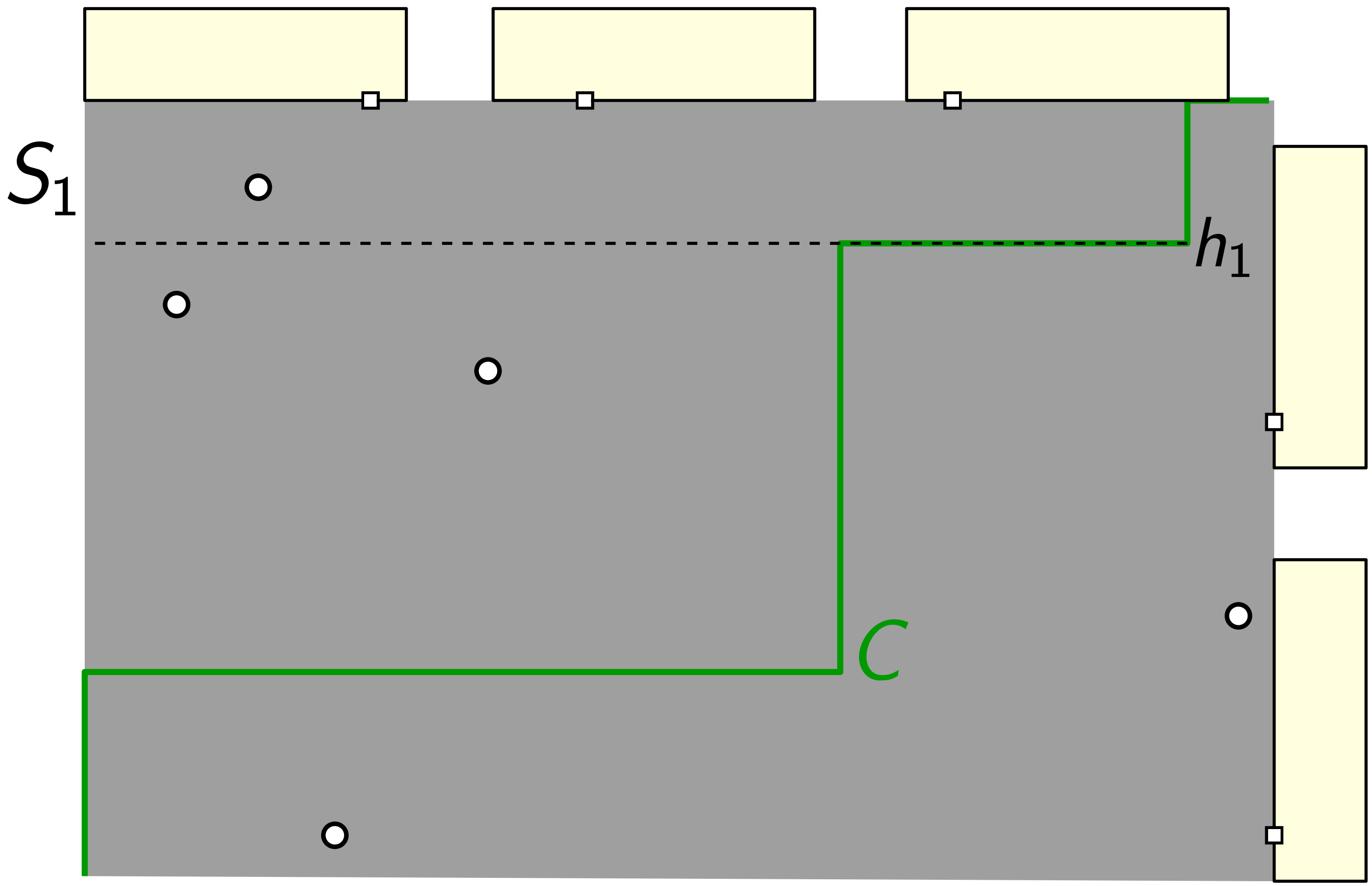
$$\Leftrightarrow \exists p_i \in h_i \setminus C : R_{p_i} \text{ is valid.}$$



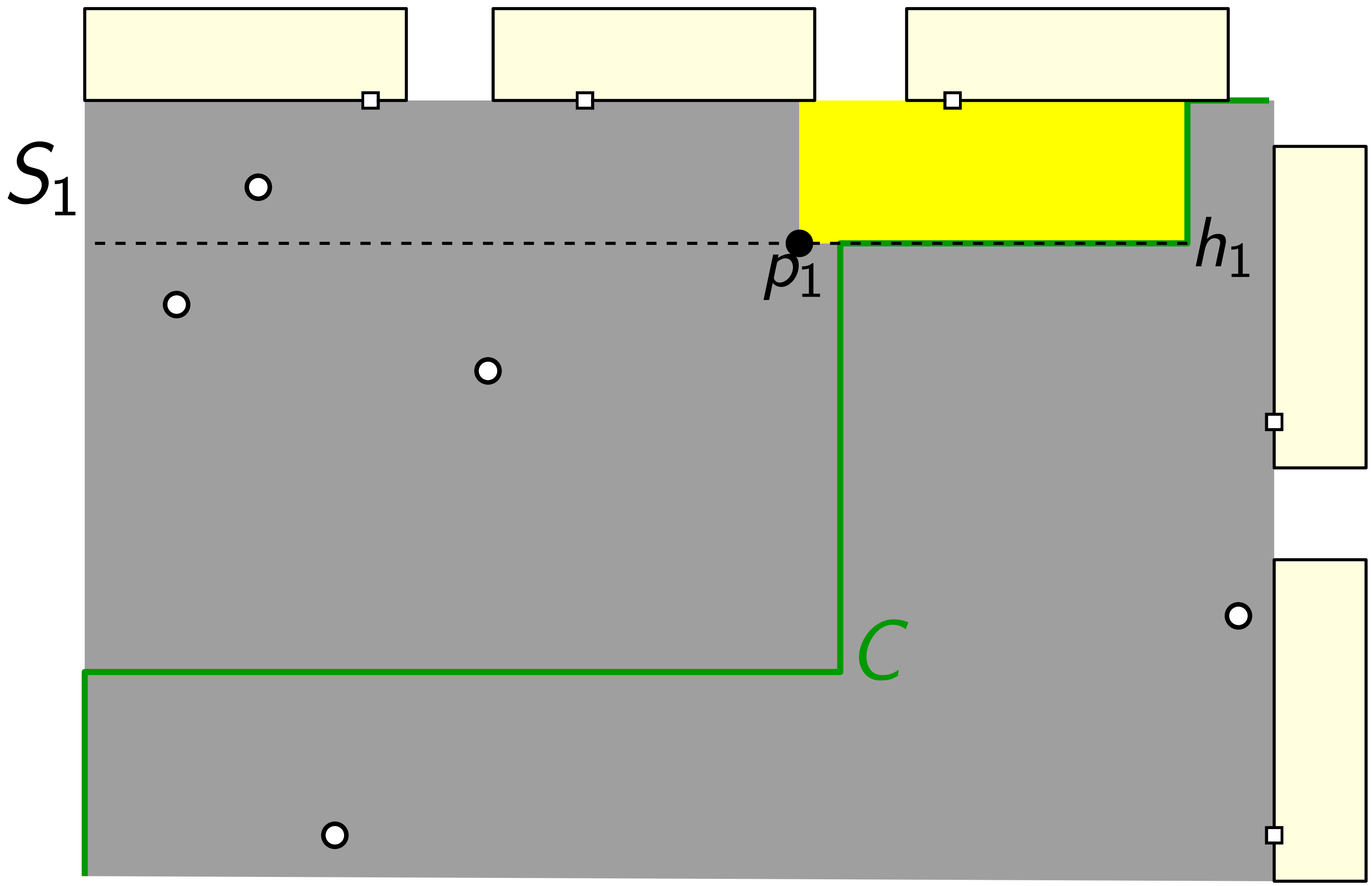
The Strip Condition - An Example



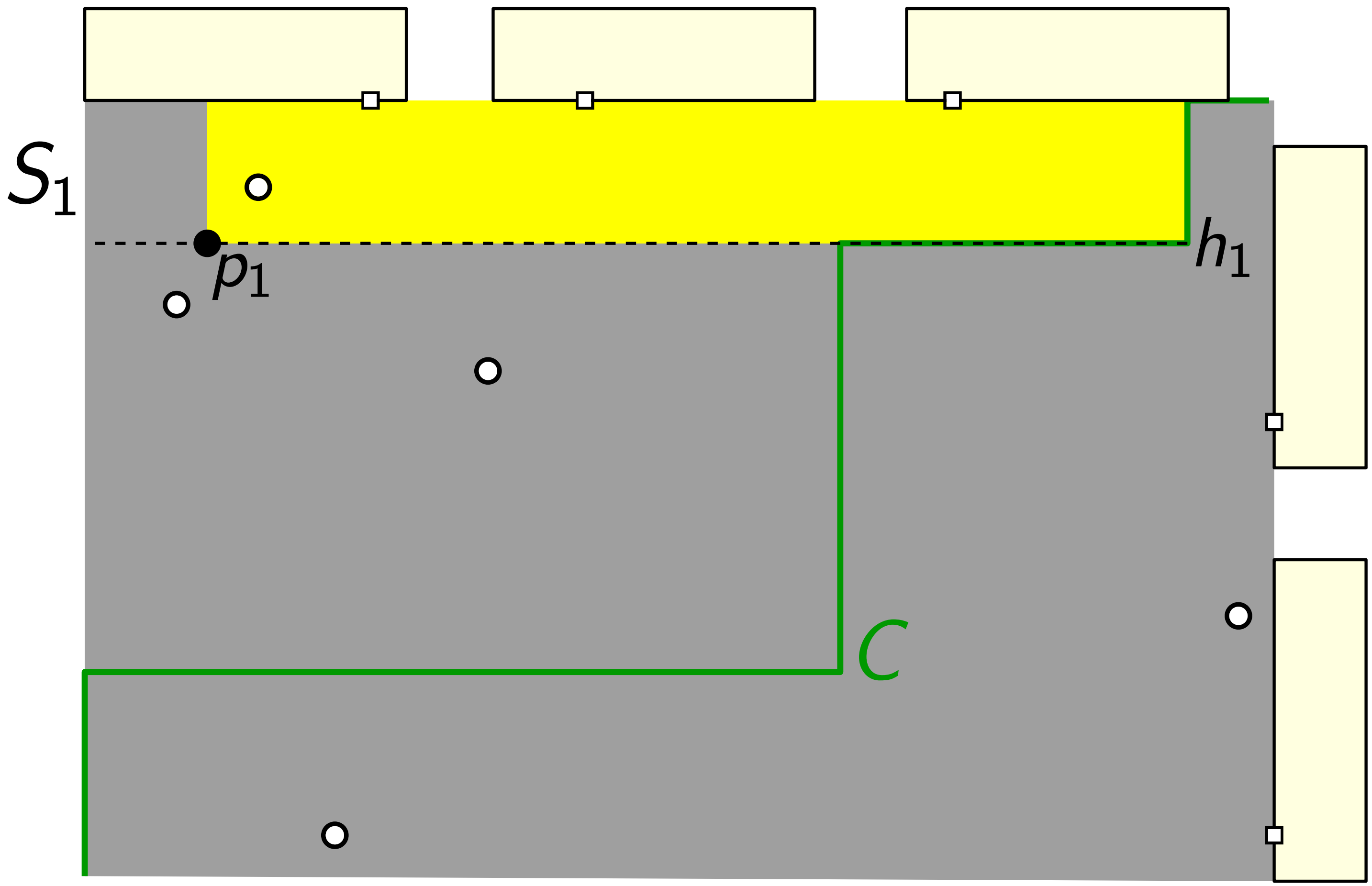
The Strip Condition - An Example



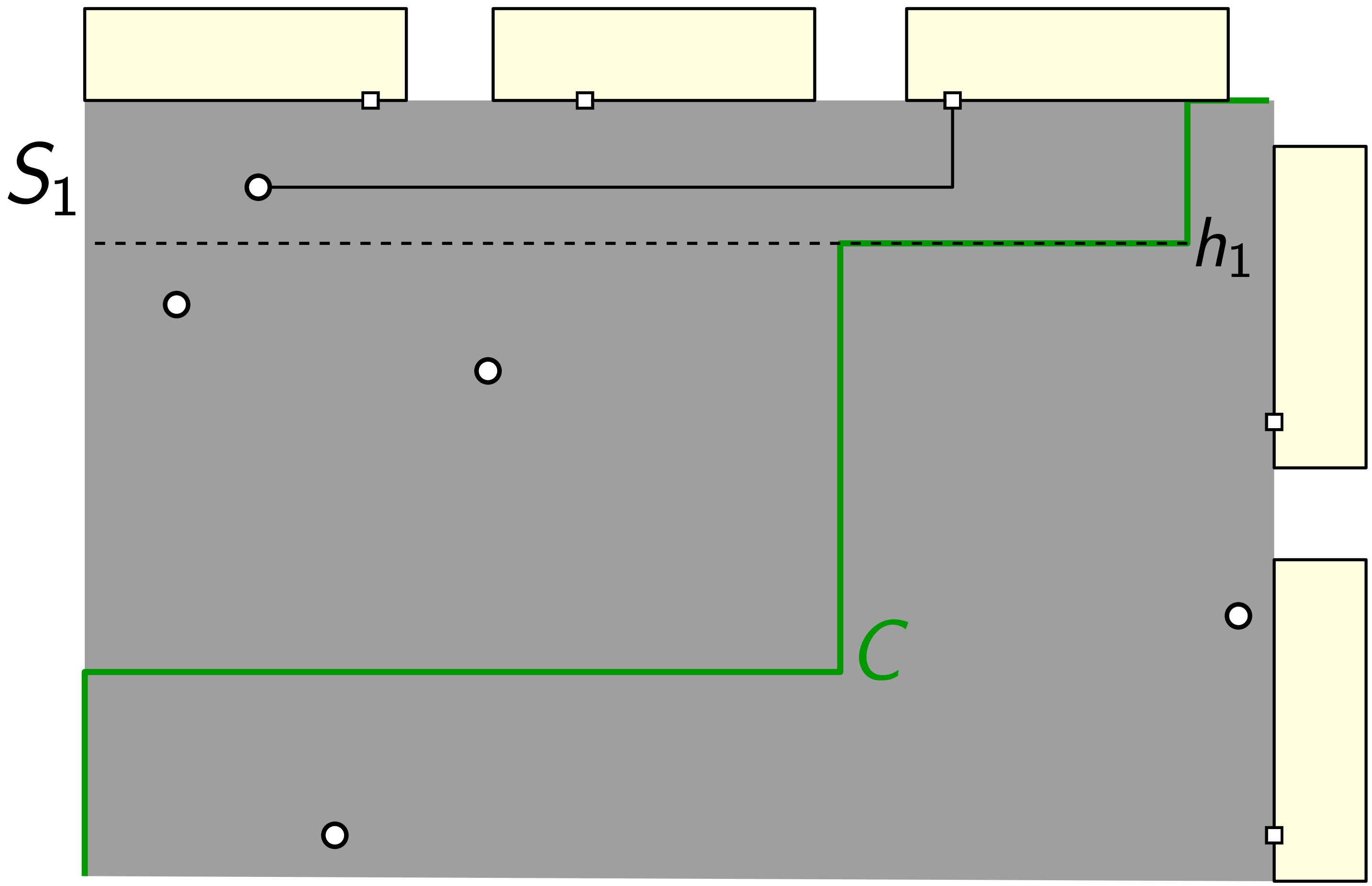
The Strip Condition - An Example



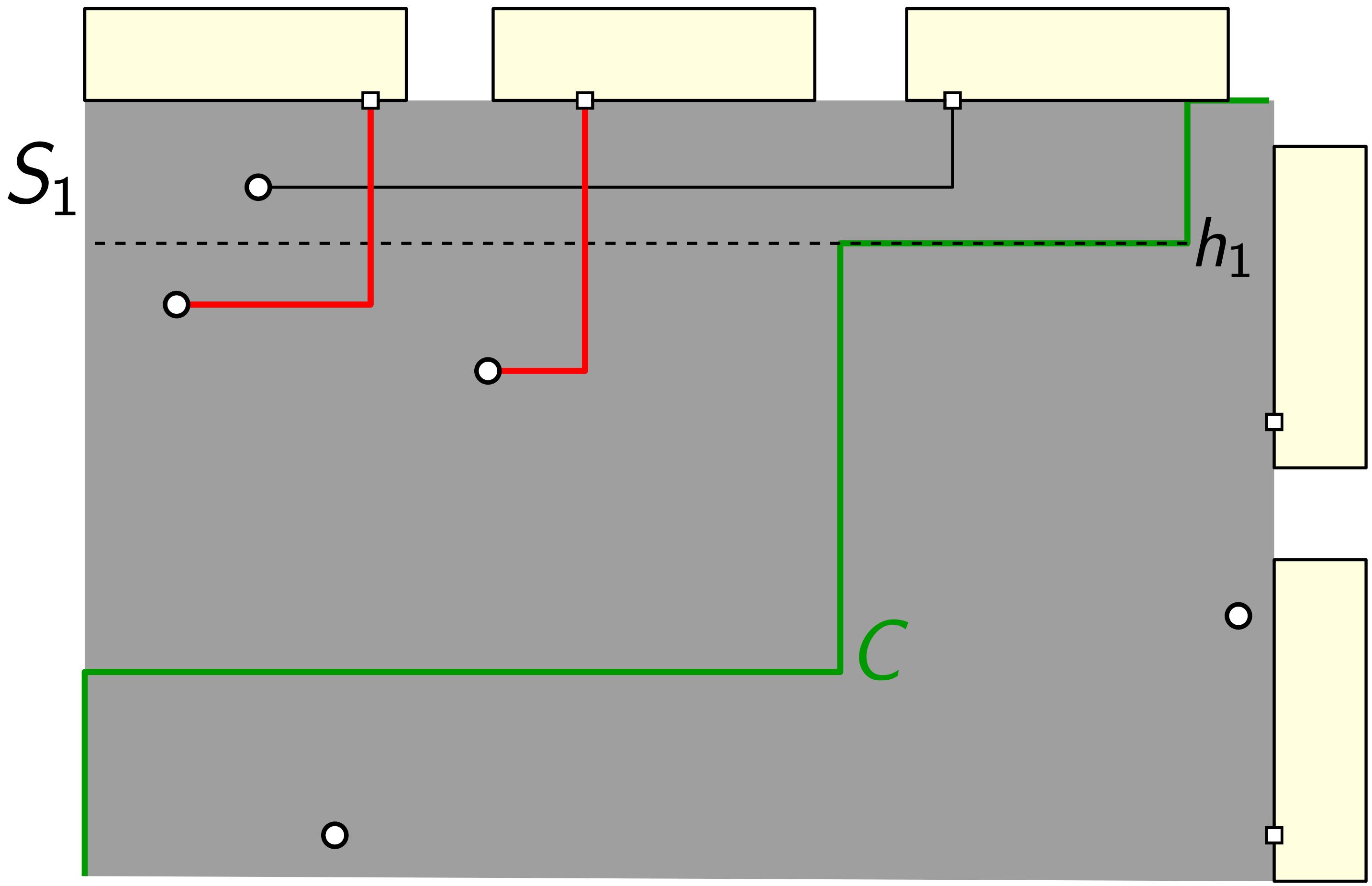
The Strip Condition - An Example



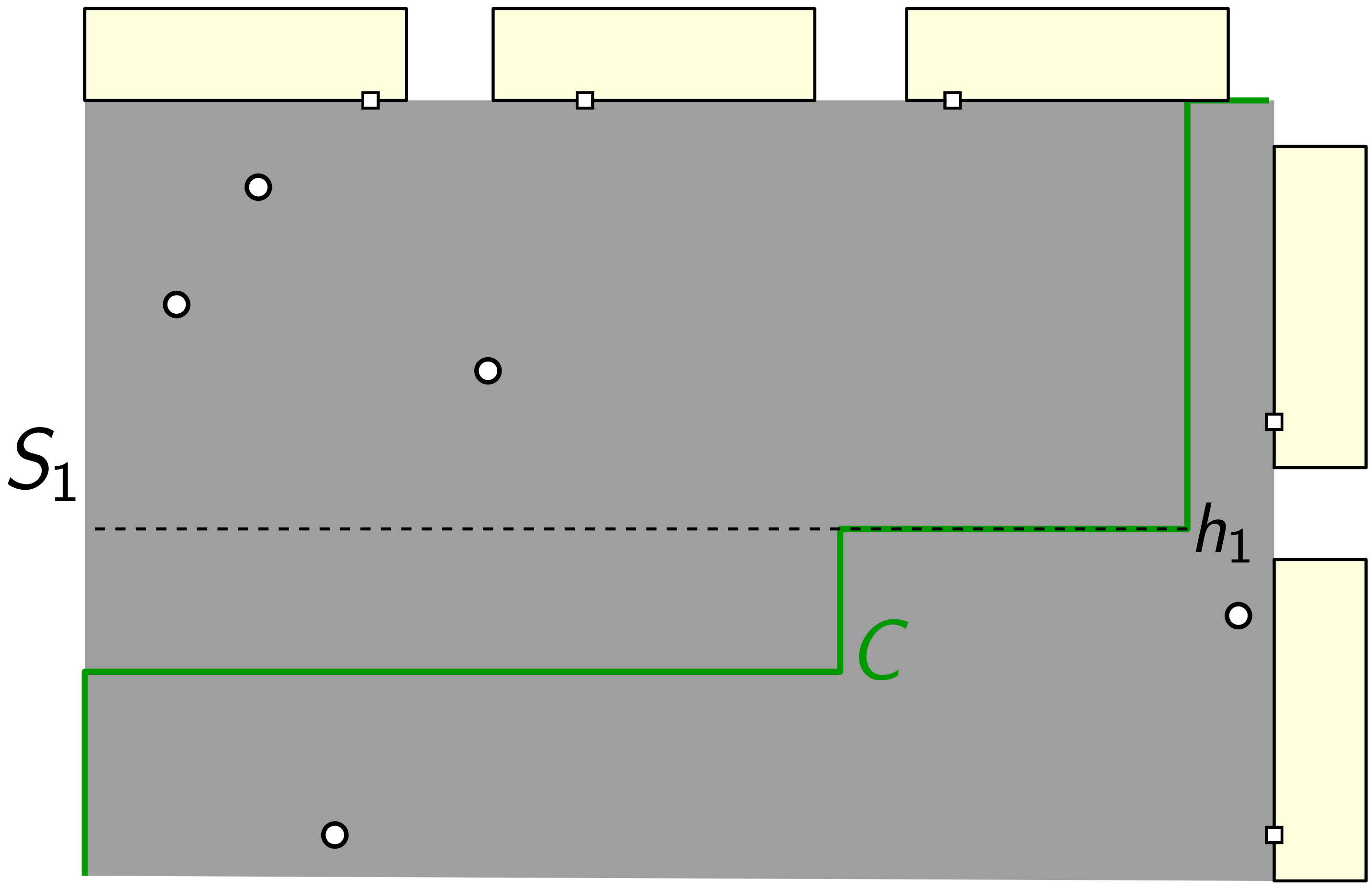
The Strip Condition - An Example



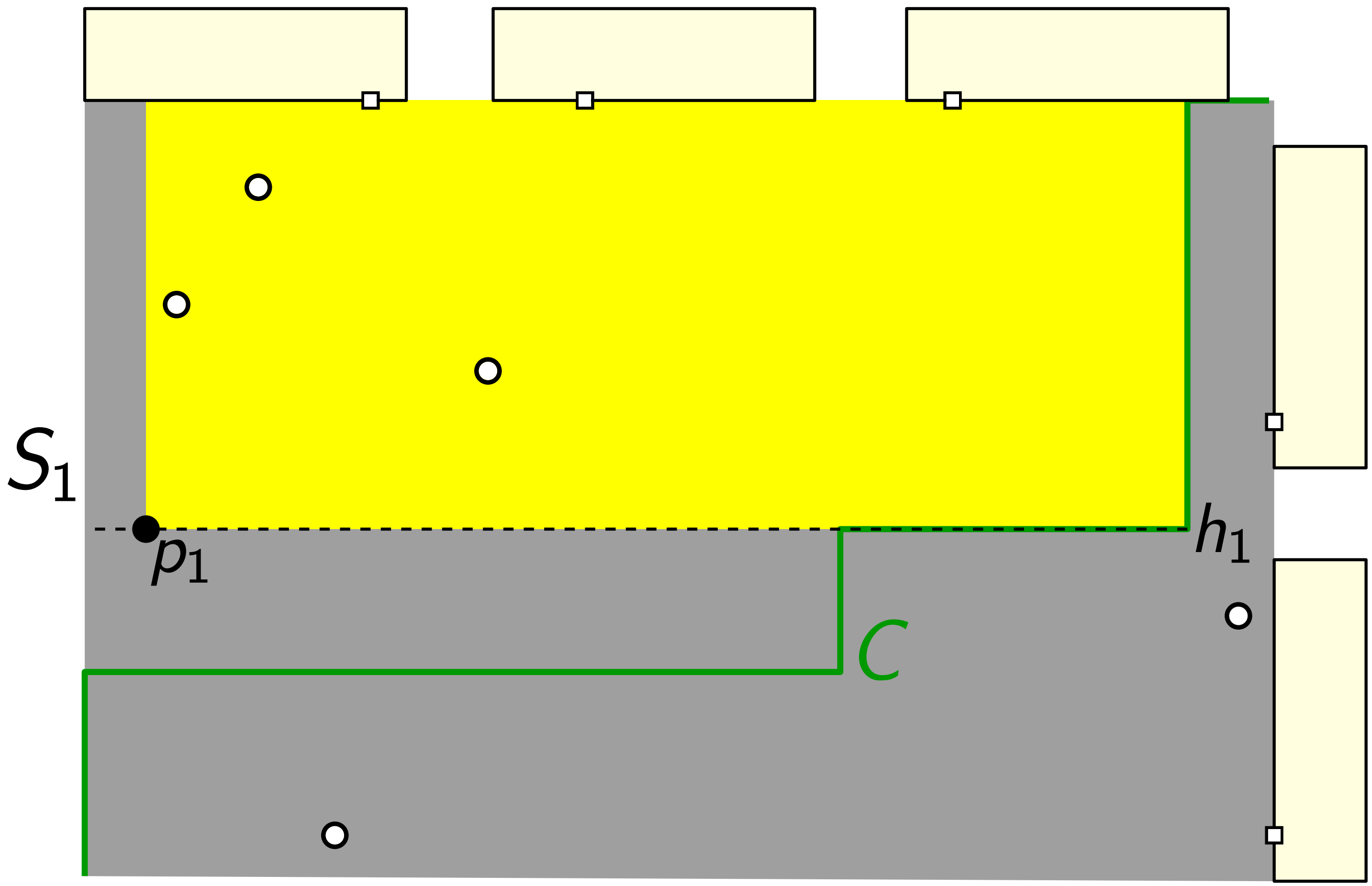
The Strip Condition - An Example



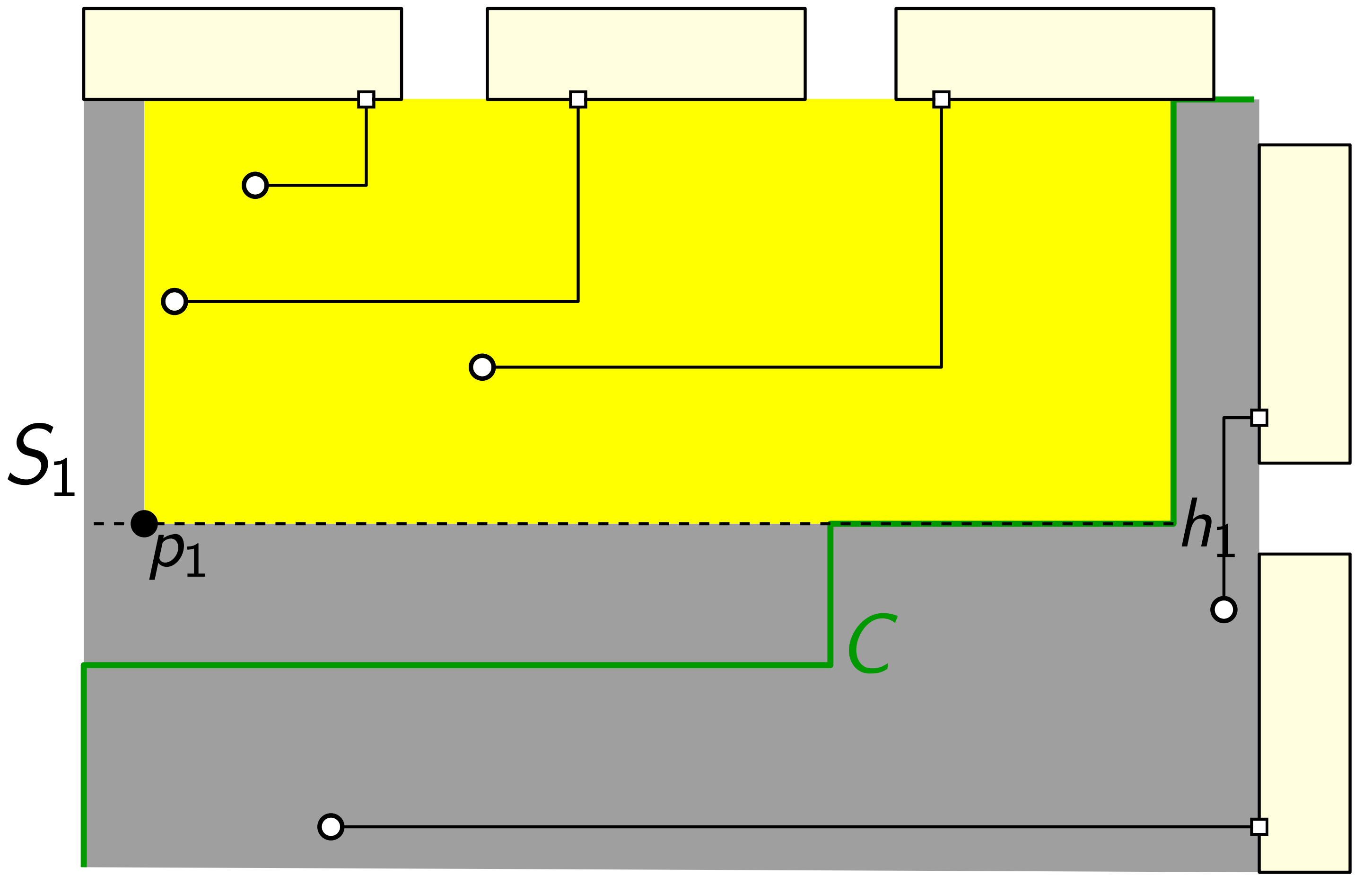
The Strip Condition - An Example



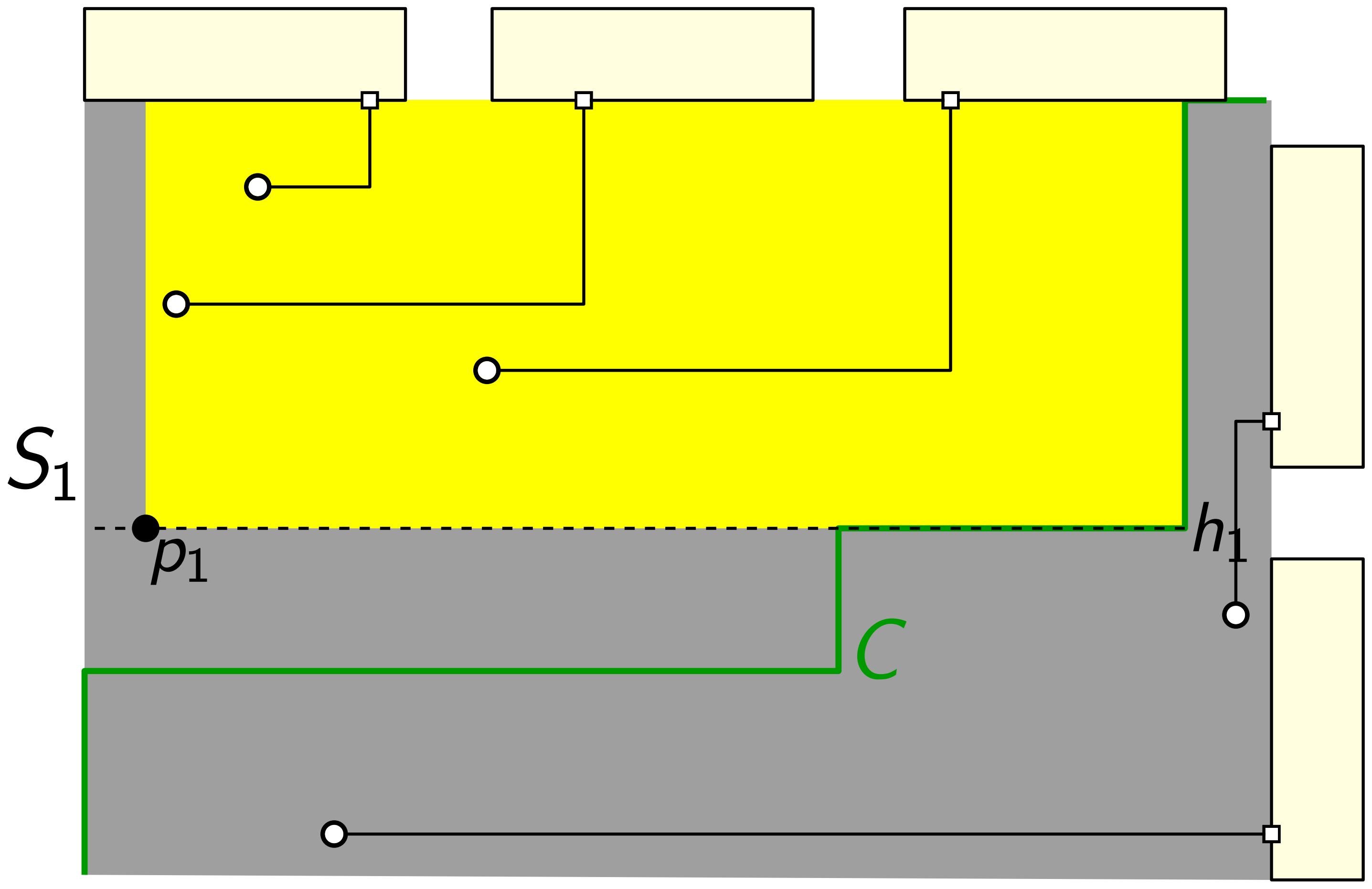
The Strip Condition - An Example



The Strip Condition - An Example



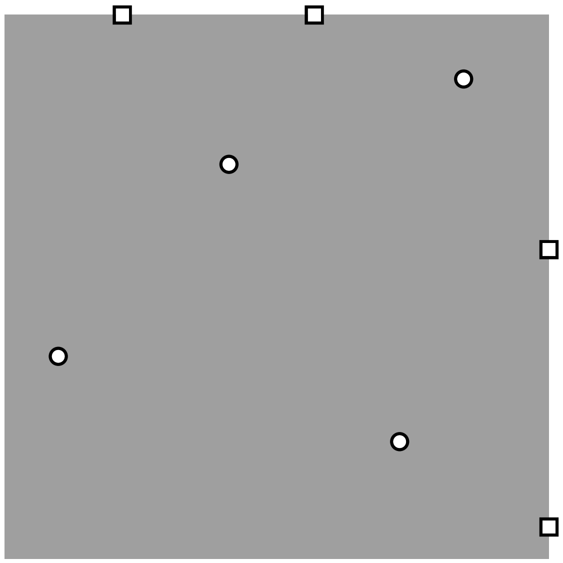
The Strip Condition - An Example



The strip conditions are sufficient and necessary

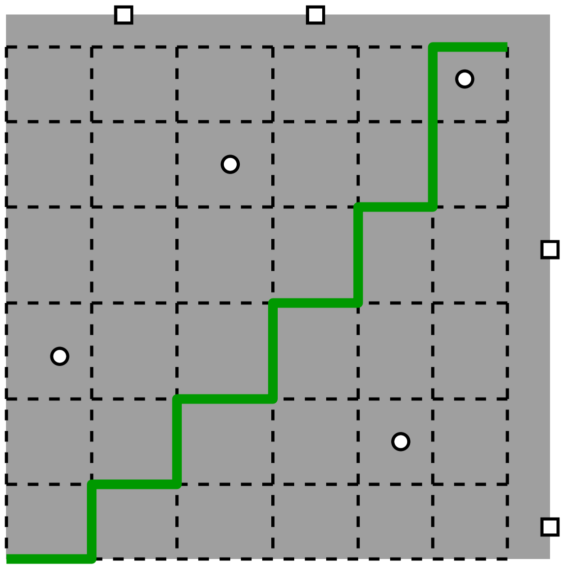
The Algorithm

- Find an xy-separating curve C that satisfies the strip conditions.



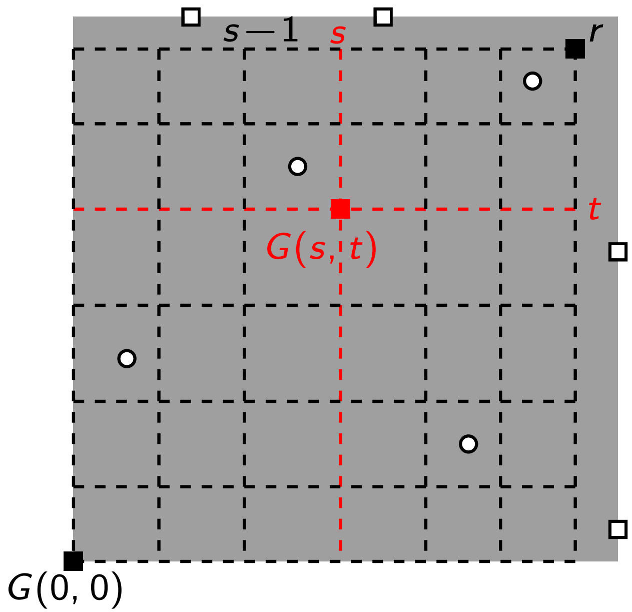
The Algorithm

- Find an xy-separating curve C that satisfies the strip conditions.
- Consider the dual of the grid induced by sites and ports



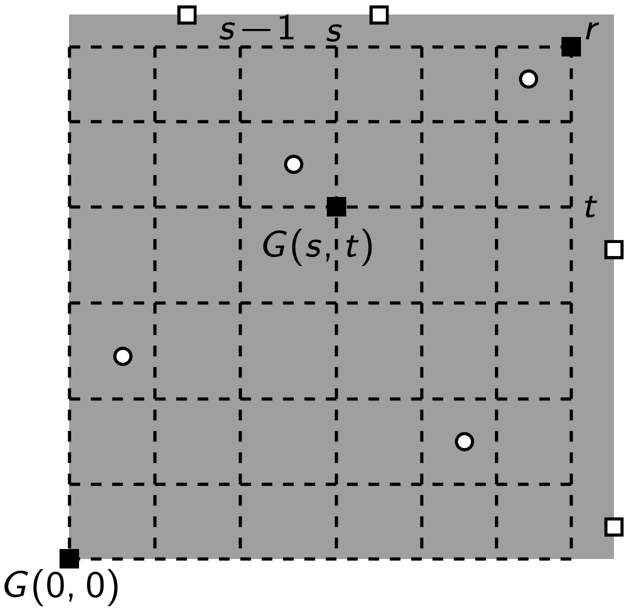
The Algorithm

- Find an xy-separating curve C that satisfies the strip conditions.
- Consider the dual of the grid induced by sites and ports
- Define grid points $G(s, t)$ and top-right corner r



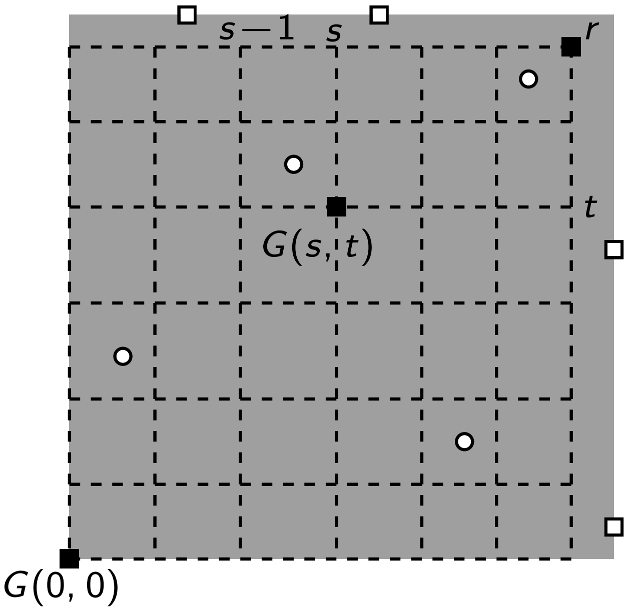
The Algorithm

Dynamic Program:
Compute table $T[(s,t),u]$.



The Algorithm

Dynamic Program:
Compute table $T[(s,t),u]$.
$T[(s,t),u] = \texttt{true}$
$\Leftrightarrow \exists xy$-monotone chain C:



The Algorithm

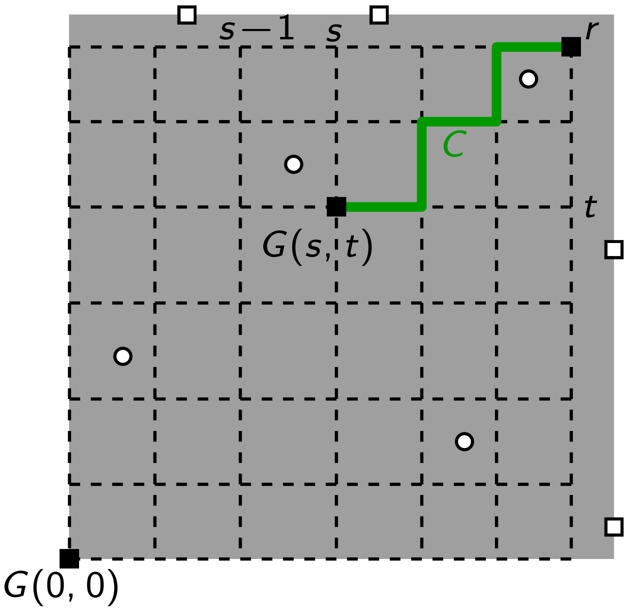
Dynamic Program:

Compute table $T[(s,t),u]$.

$T[(s,t),u] = \texttt{true}$

$\Leftrightarrow \exists xy$-monotone chain C:

- C starts at r and ends at $G(s,t)$



The Algorithm

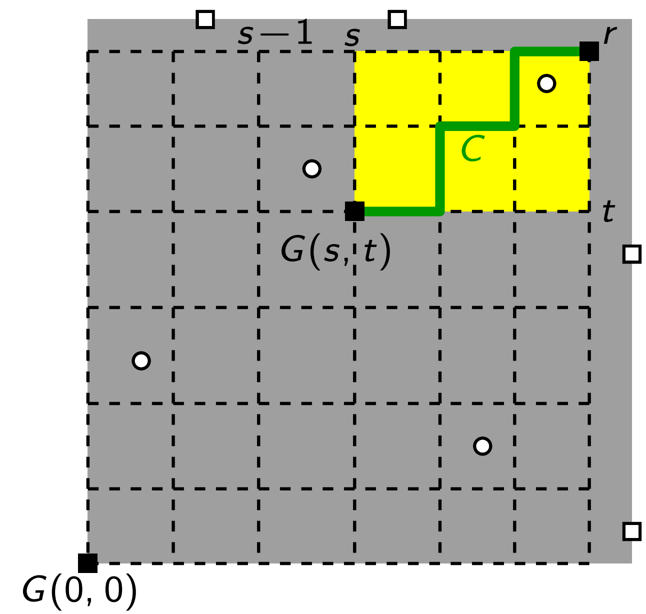
Dynamic Program:

Compute table $T[(s,t),u]$.

$T[(s,t),u] = \texttt{true}$

$\Leftrightarrow \exists xy$-monotone chain C:

- C starts at r and ends at $G(s,t)$
- u sites above C



The Algorithm

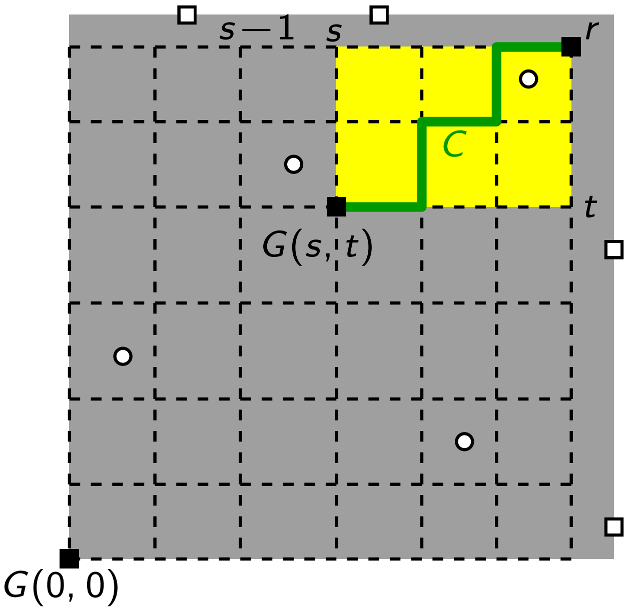
Dynamic Program:

Compute table $T[(s,t),u]$.

$T[(s,t),u] = \texttt{true}$

$\Leftrightarrow \exists xy$-monotone chain C:

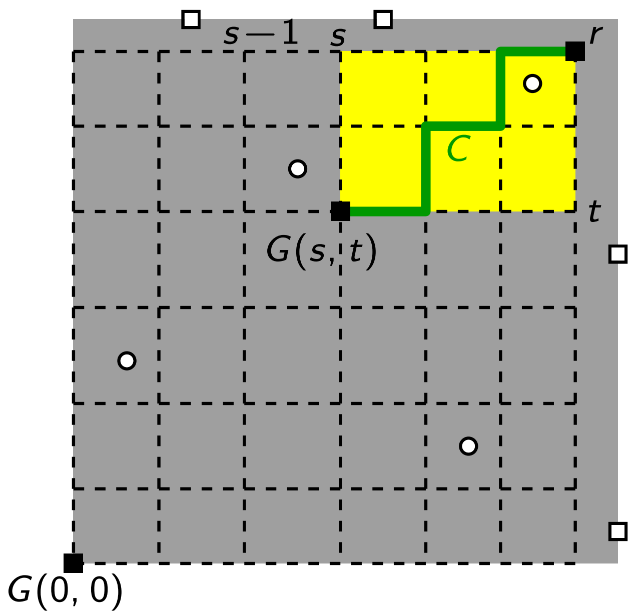
- C starts at r and ends at $G(s,t)$
- u sites above C
- strip conditions are satisfied



The Algorithm

Dynamic Program:
Compute table $T[(s,t),u]$.
$T[(s,t),u] = \texttt{true}$
$\Leftrightarrow \exists xy$-monotone chain C:

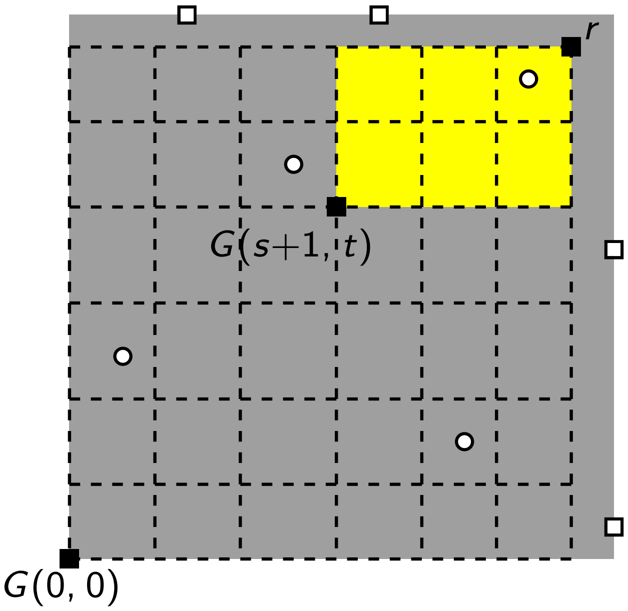
- C starts at r and ends at $G(s,t)$
- u sites above C
- strip conditions are satisfied



Instance solvable $\Leftrightarrow T[(0,0),u] = \texttt{true}$ for some u.

One Step of the Algorithm

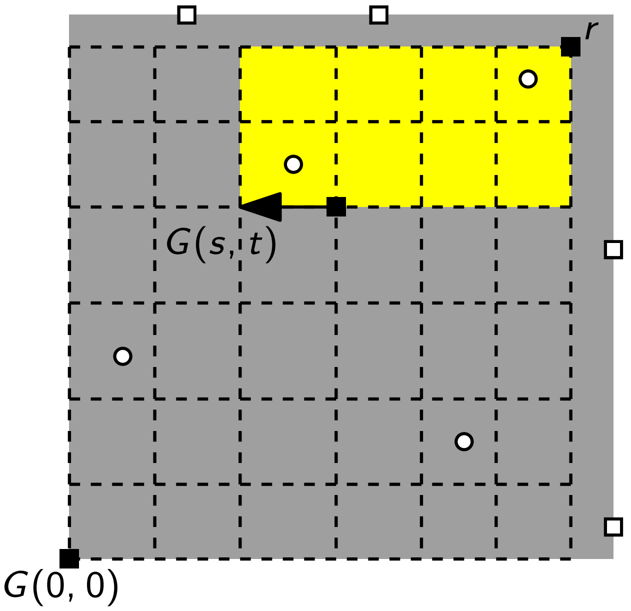
Assume $T[(s+1,t),u] = \texttt{true}$.



One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

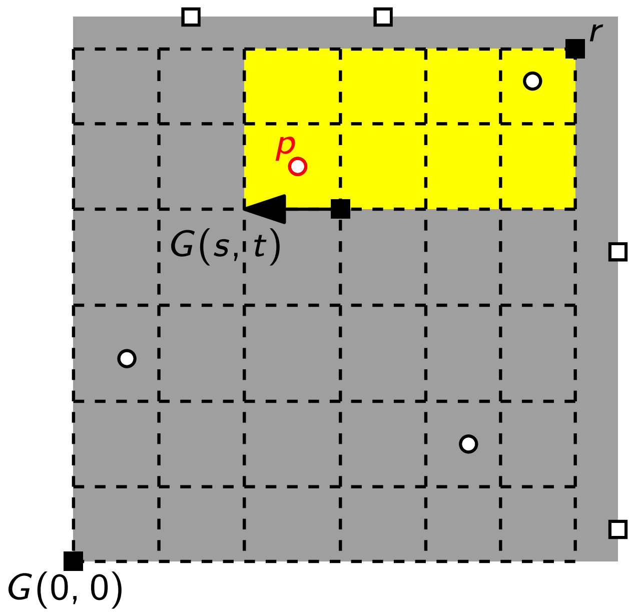


One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*



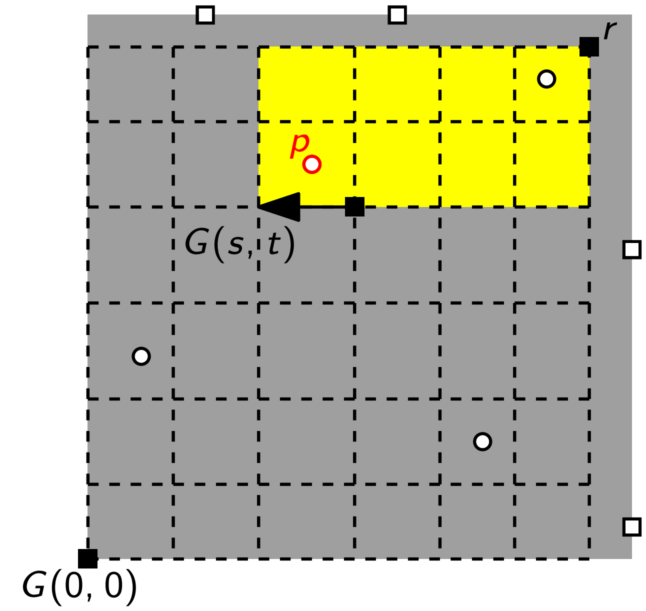
One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

$y(p) > G_y(t)$



One Step of the Algorithm

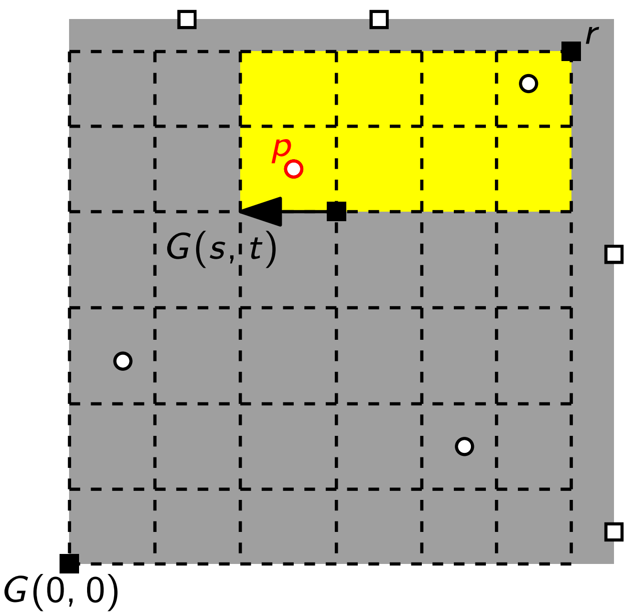
Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

$$y(p) > G_y(t)$$

$$\Rightarrow T[(s,t),u+1] = \texttt{true}.$$



One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

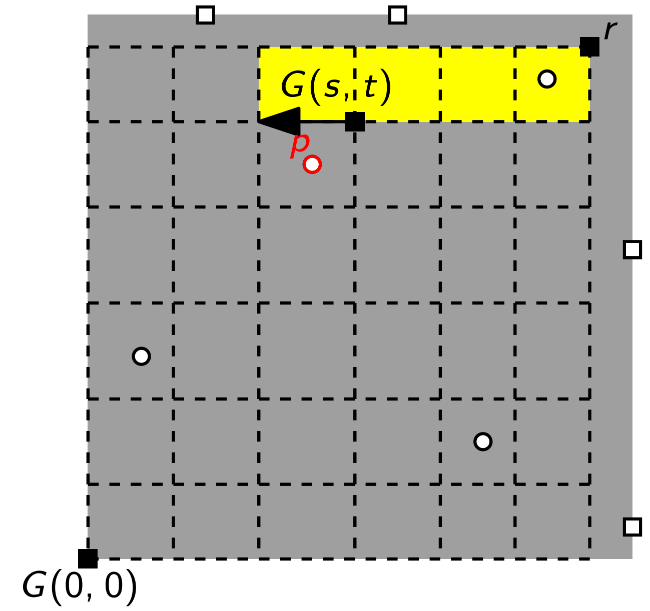
Go from $s+1$ to s.

Case 1: *site event*

$$y(p) > G_y(t)$$

$$\Rightarrow T[(s,t),u+1] = \texttt{true}.$$

$$y(p) < G_y(t)$$



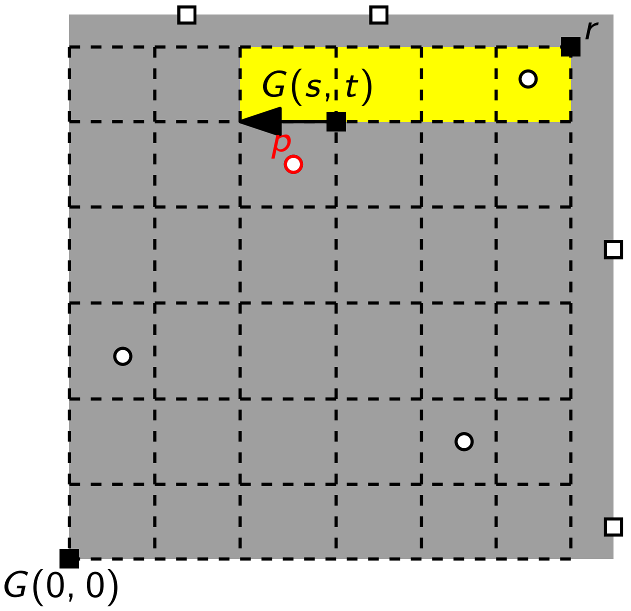
One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

$$y(p) > G_y(t)$$
$$\Rightarrow T[(s,t),u+1] = \texttt{true}.$$
$$y(p) < G_y(t)$$
$$\Rightarrow T[(s,t),u] = \texttt{true}.$$



One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

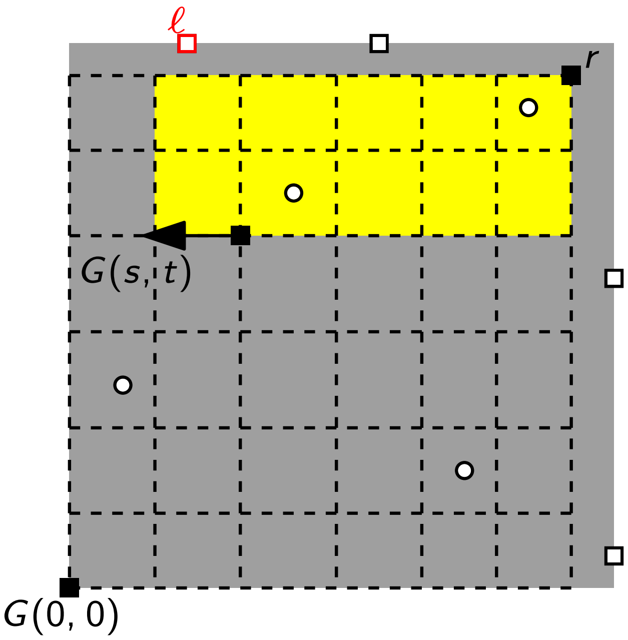
$$y(p) > G_y(t)$$

$$\Rightarrow T[(s,t),u+1] = \texttt{true}.$$

$$y(p) < G_y(t)$$

$$\Rightarrow T[(s,t),u] = \texttt{true}.$$

Case 2: *port event*



One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

$y(p) > G_y(t)$

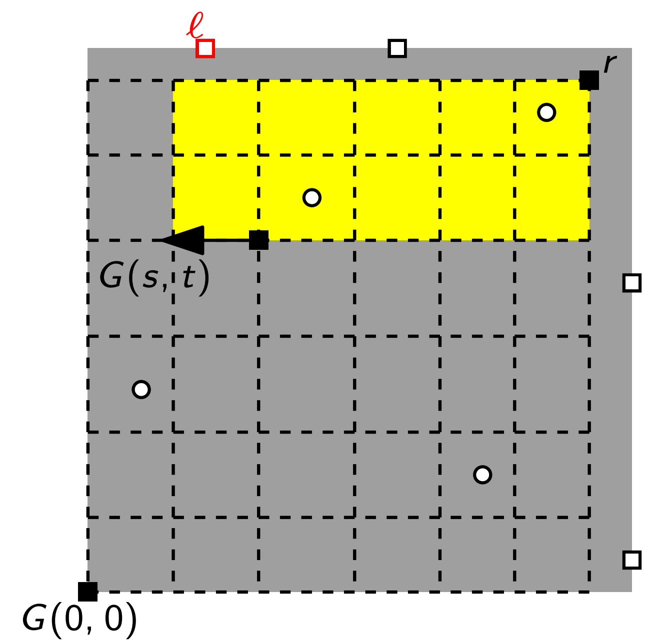
$\Rightarrow T[(s,t),u+1] = \texttt{true}$.

$y(p) < G_y(t)$

$\Rightarrow T[(s,t),u] = \texttt{true}$.

Case 2: *port event*

Check strip condition



One Step of the Algorithm

Assume $T[(s+1,t),u] = \texttt{true}$.

Go from $s+1$ to s.

Case 1: *site event*

$y(p) > G_y(t)$

$\Rightarrow T[(s,t),u+1] = \texttt{true}$.

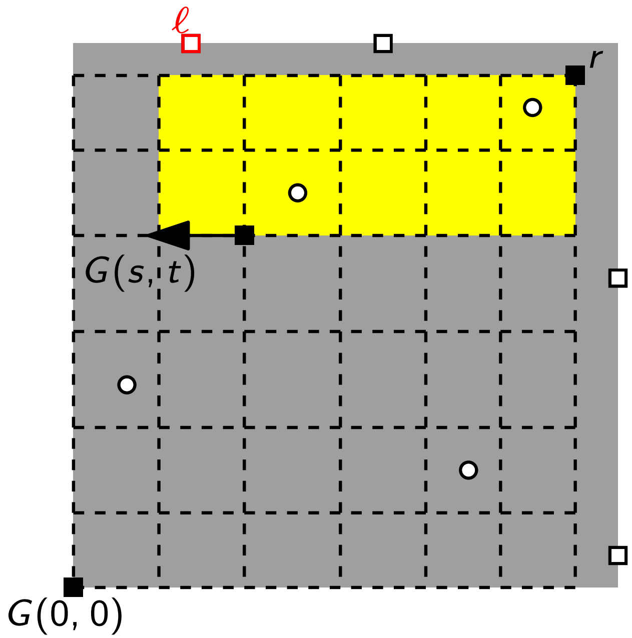
$y(p) < G_y(t)$

$\Rightarrow T[(s,t),u] = \texttt{true}$.

Case 2: *port event*

Check strip condition

satisfied $\Rightarrow T[(s,t),u] = \texttt{true}$



Running Time

- Three table entries $\Rightarrow O(n^3)$ steps

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

$$\rightarrow O(n^3) \text{ time, } O(n^3) \text{ space}$$

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

  $\rightarrow O(n^3)$ time, $O(n^3)$ space

- Entries of T depend only on previous row and column

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

  $\rightarrow O(n^3)$ time, $O(n^3)$ space

- Entries of T depend only on previous row and column
- Use algorithm of [Hirschberg, 1975] to backtrack a solution

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

$$\rightarrow O(n^3) \text{ time, } O(n^3) \text{ space}$$

- Entries of T depend only on previous row and column
- Use algorithm of [Hirschberg, 1975] to backtrack a solution

$$\rightarrow O(n^3) \text{ time, } O(n^2) \text{ space}$$

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

$$\rightarrow O(n^3) \text{ time, } O(n^3) \text{ space}$$

- Entries of T depend only on previous row and column
- Use algorithm of [Hirschberg, 1975] to backtrack a solution

$$\rightarrow O(n^3) \text{ time, } O(n^2) \text{ space}$$

- Valid values of u form an interval
  $\Rightarrow$ We only need to save the boundaries of u

Running Time

- Three table entries $\Rightarrow O(n^3)$ steps
- $O(n^2)$ preprocessing $\Rightarrow$ check strip condition in $O(1)$ time

$$\rightarrow O(n^3) \text{ time}, \ O(n^3) \text{ space}$$

- Entries of T depend only on previous row and column
- Use algorithm of [Hirschberg, 1975] to backtrack a solution

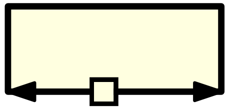
$$\rightarrow O(n^3) \text{ time}, \ O(n^2) \text{ space}$$

- Valid values of u form an interval
  $\Rightarrow$ We only need to save the boundaries of u

$$\boxed{\rightarrow O(n^2) \textbf{ time}, \ O(n) \textbf{ space}}$$

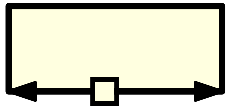
Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



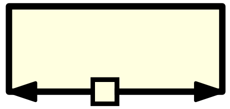
Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space

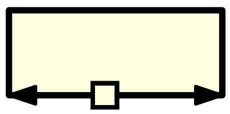
Extensions



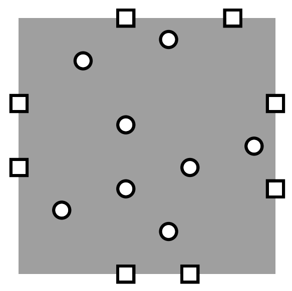
- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space
- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space
- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space

Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space

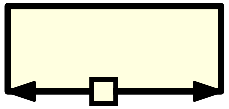


- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space
- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space
- Four-Sided Boundary Labeling:



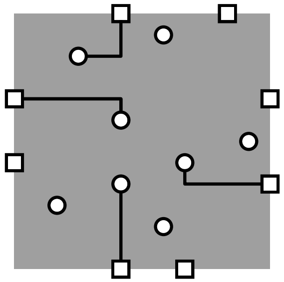
Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



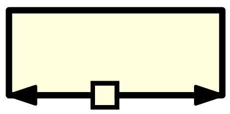
- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space
- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space
- Four-Sided Boundary Labeling:

Guess „extremal“ leader for all sides



Extensions

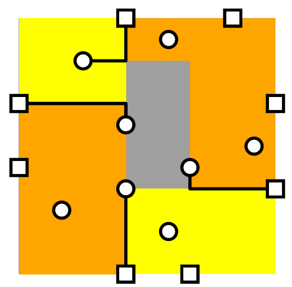
- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space

- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space

- Four-Sided Boundary Labeling:

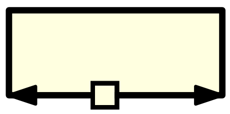


Guess „extremal“ leader for all sides

Partition rectangle into four areas

Extensions

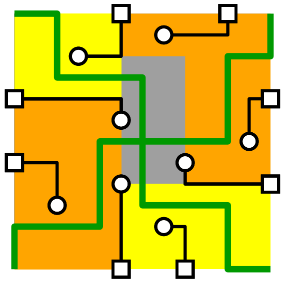
- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space

- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space

- Four-Sided Boundary Labeling:



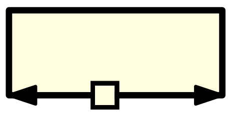
Guess „extremal“ leader for all sides

Partition rectangle into four areas

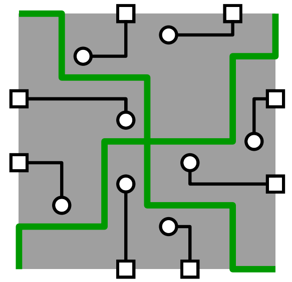
Solve each area independantly

Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space

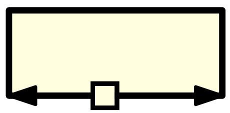


- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space
- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space
- Four-Sided Boundary Labeling:
  $O(n^{10})$ time, $O(n)$ space

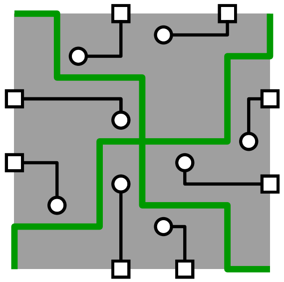


Extensions

- Sliding Ports:
  $O(n^2)$ time, $O(n)$ space



- Maximize number of labeled sites:
  $O(n^3 \log n)$ time, $O(n)$ space
- Leader-length minimization:
  $O(n^8 \log n)$ time, $O(n^6)$ space
- Four-Sided Boundary Labeling:
  $O(n^{10})$ time, $O(n)$ space



- Three-Sided Boundary Labeling:
  $O(n^4)$ time, $O(n)$ space