EuroCG 2024
13-15 March
Ioannina, Greece

A.D. 1308 unipg
UNIVERSITÀ DEGLI STUDI DI PERUGIA

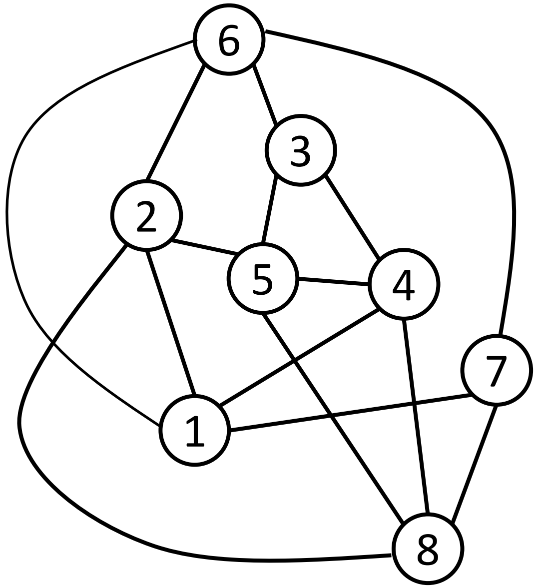
Orthogonal Graph Drawings and the Bend Minimization Problem

Walter Didimo
(University of Perugia)

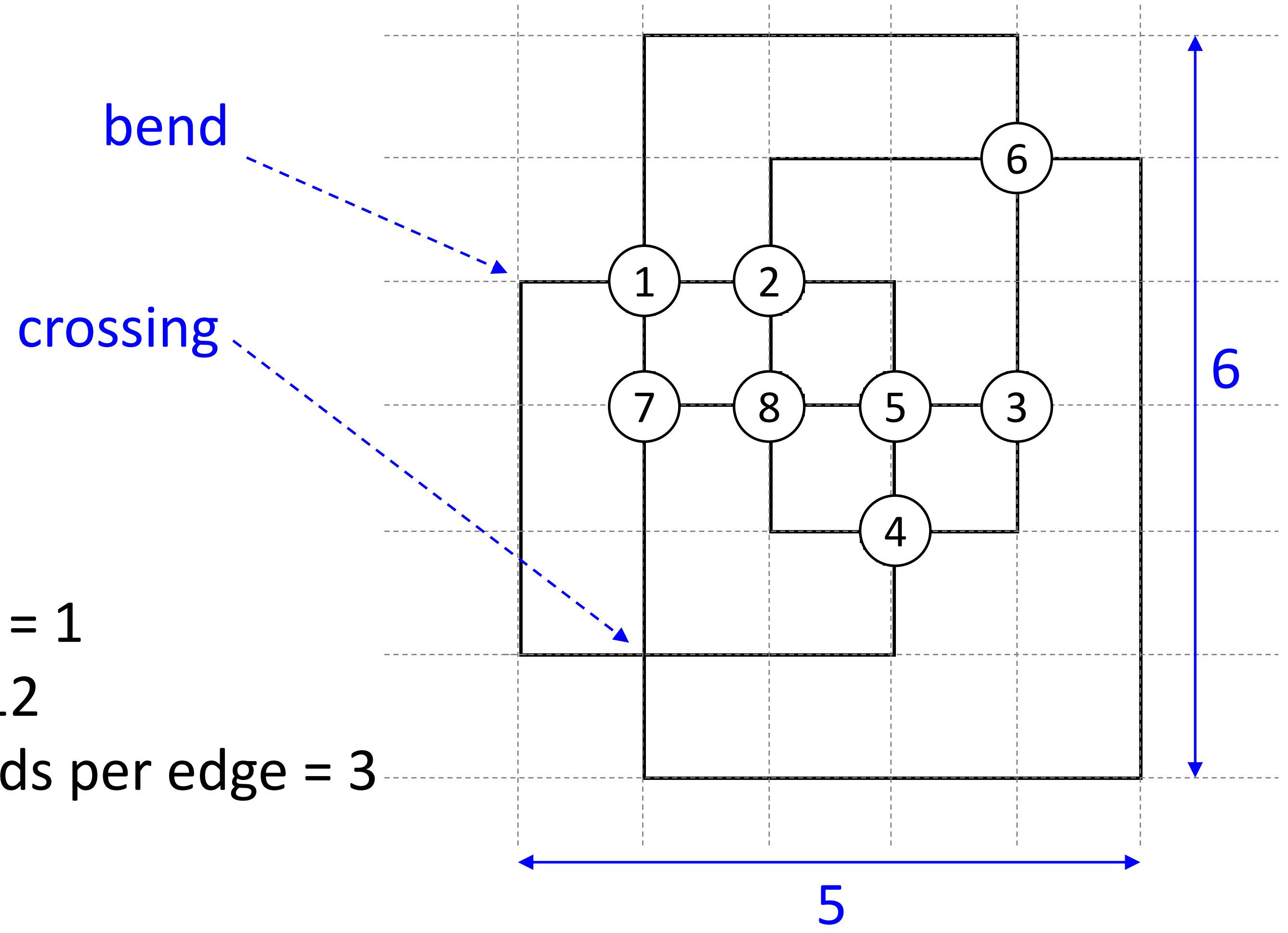
joint work with
Emilio Di Giacomo, Michael Kaufmann,
Giuseppe Liotta, Fabrizio Montecchiani,
Giacomo Ortali, Maurizio Patrignani

Orthogonal drawings: Definition

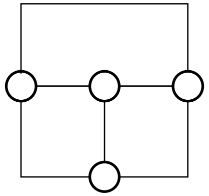
4-graph



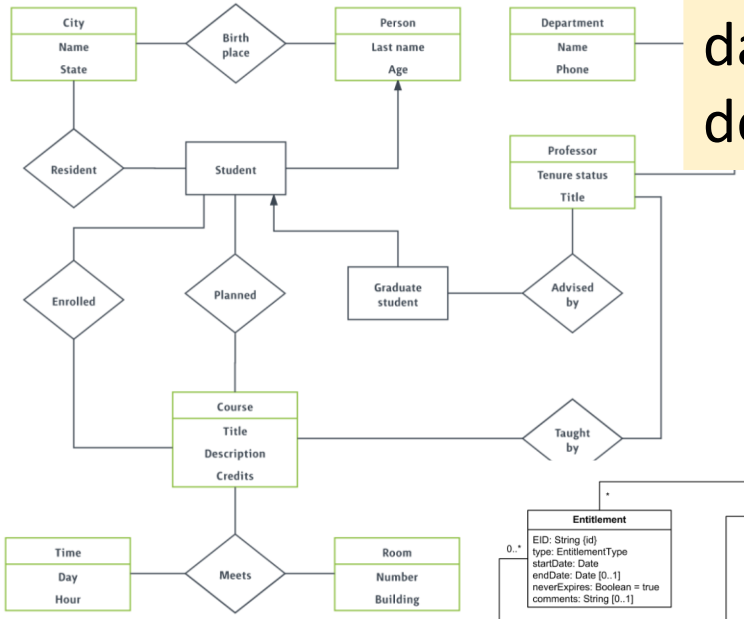
orthogonal drawing



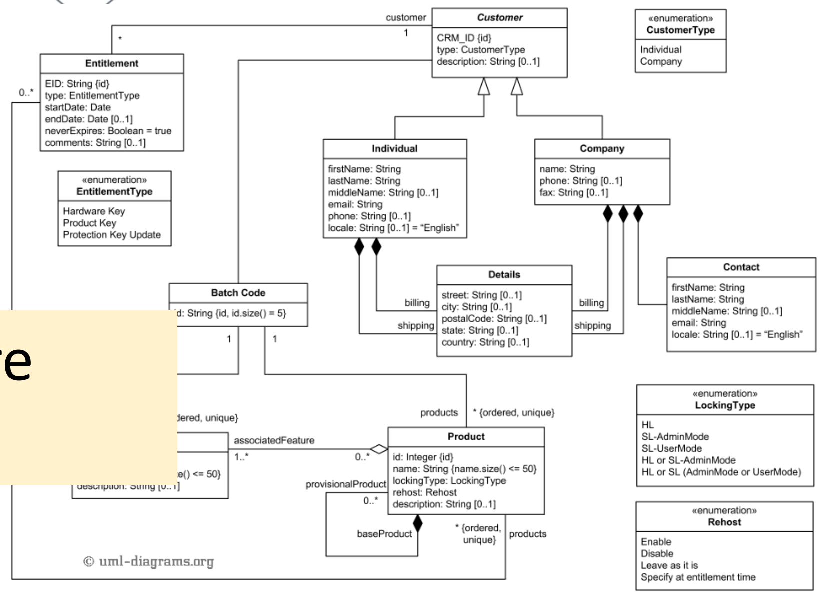
crossings = 1
bends = 12
max. bends per edge = 3
area = 30



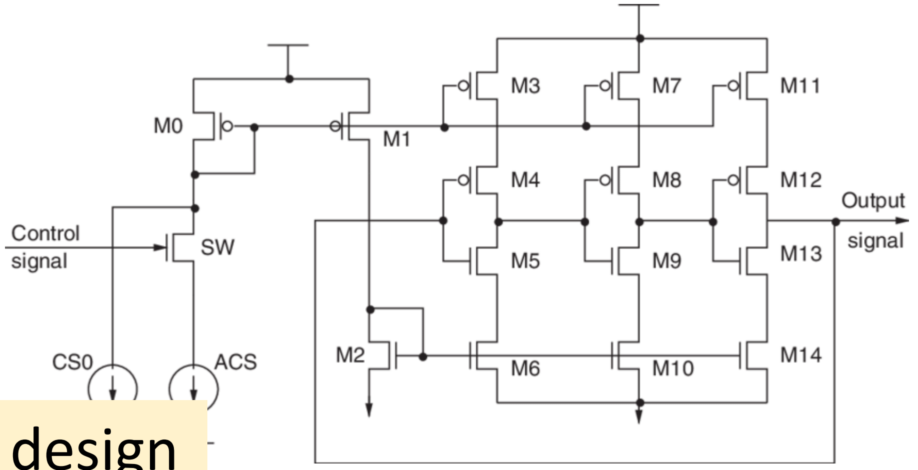
Orthogonal drawings: Applications



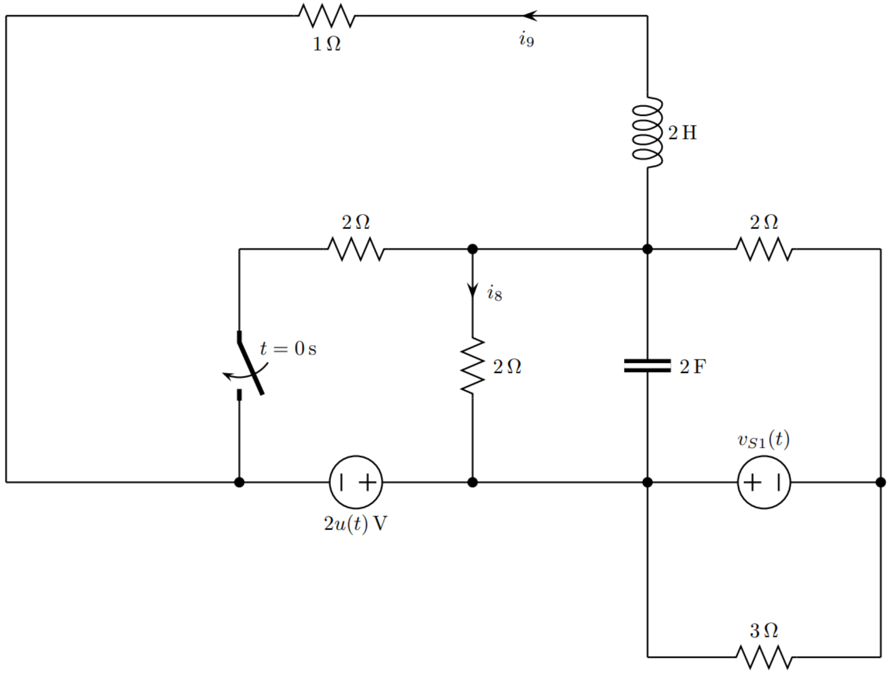
database design

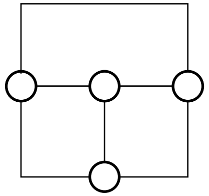


software design

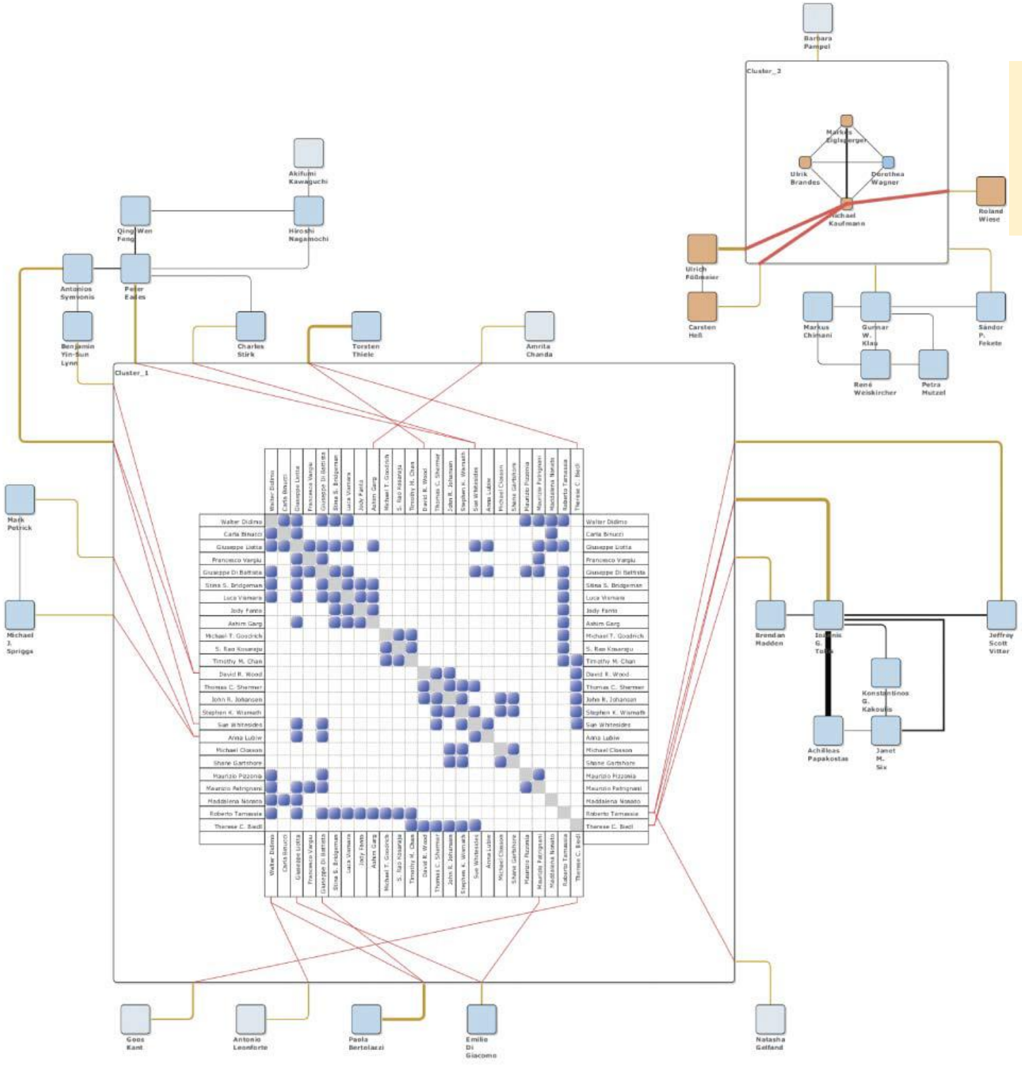


circuit design

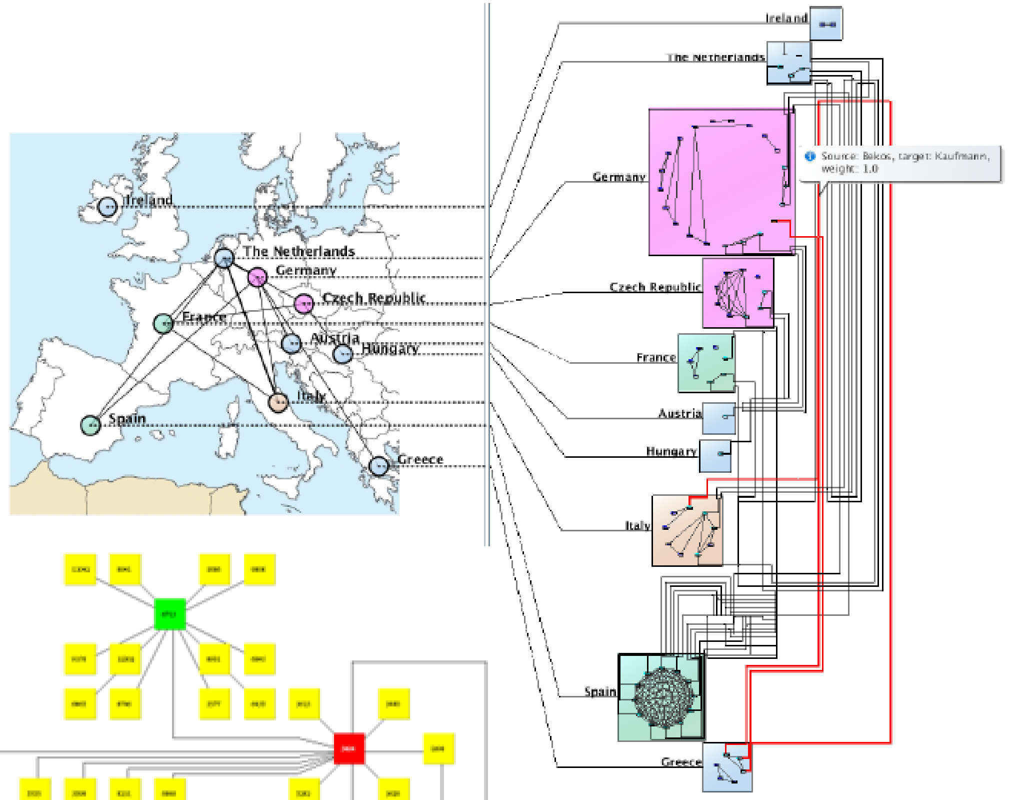


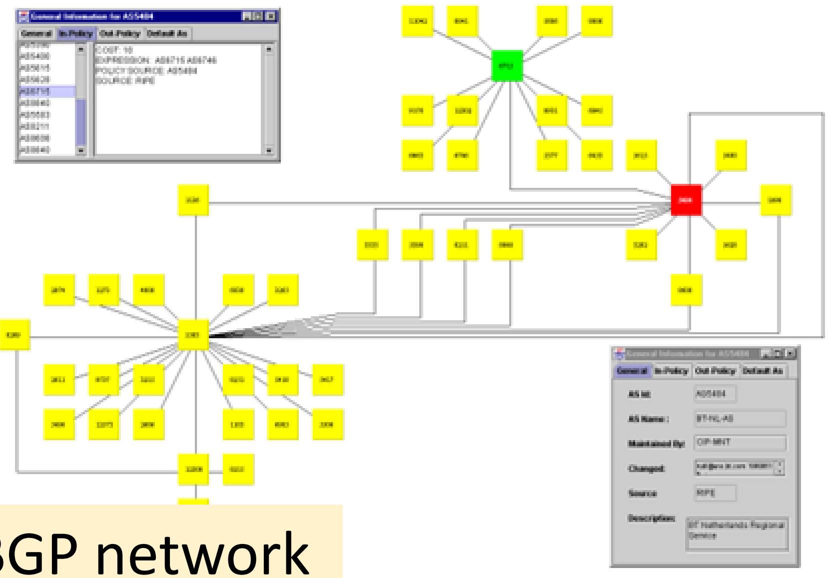


Orthogonal drawings and hybrid visualizations



collaboration networks





BGP network

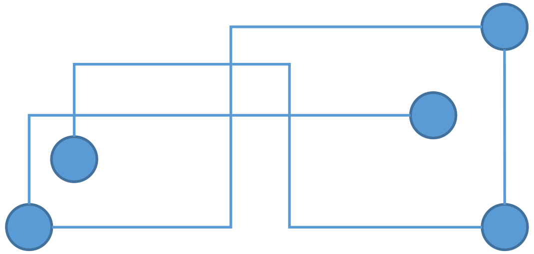
Quality metrics (aesthetics)

- Metrics used to evaluate the "quality" (readability) of a drawing

Quality metrics (aesthetics)

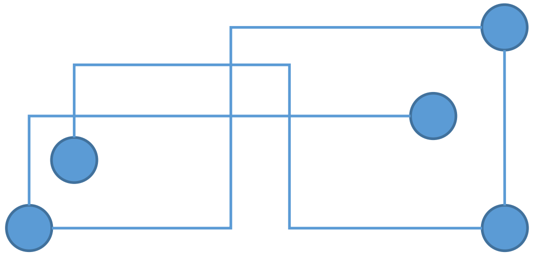
Crossings

☹

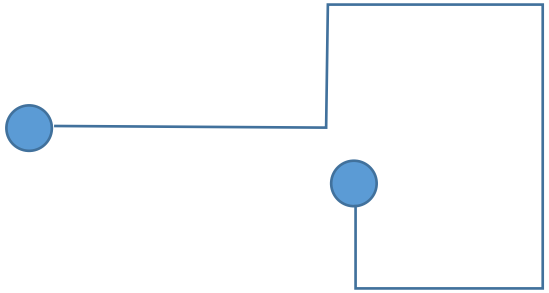


Quality metrics (aesthetics)

Crossings

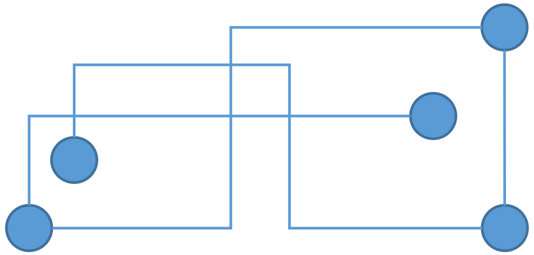


Bends

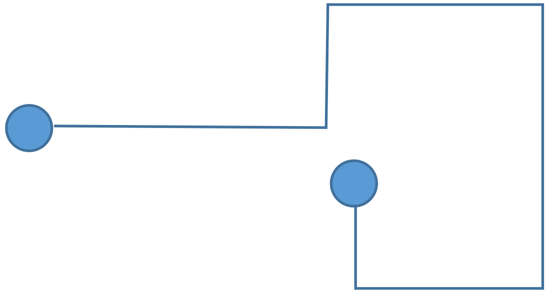


Quality metrics (aesthetics)

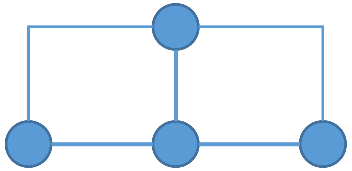
Crossings



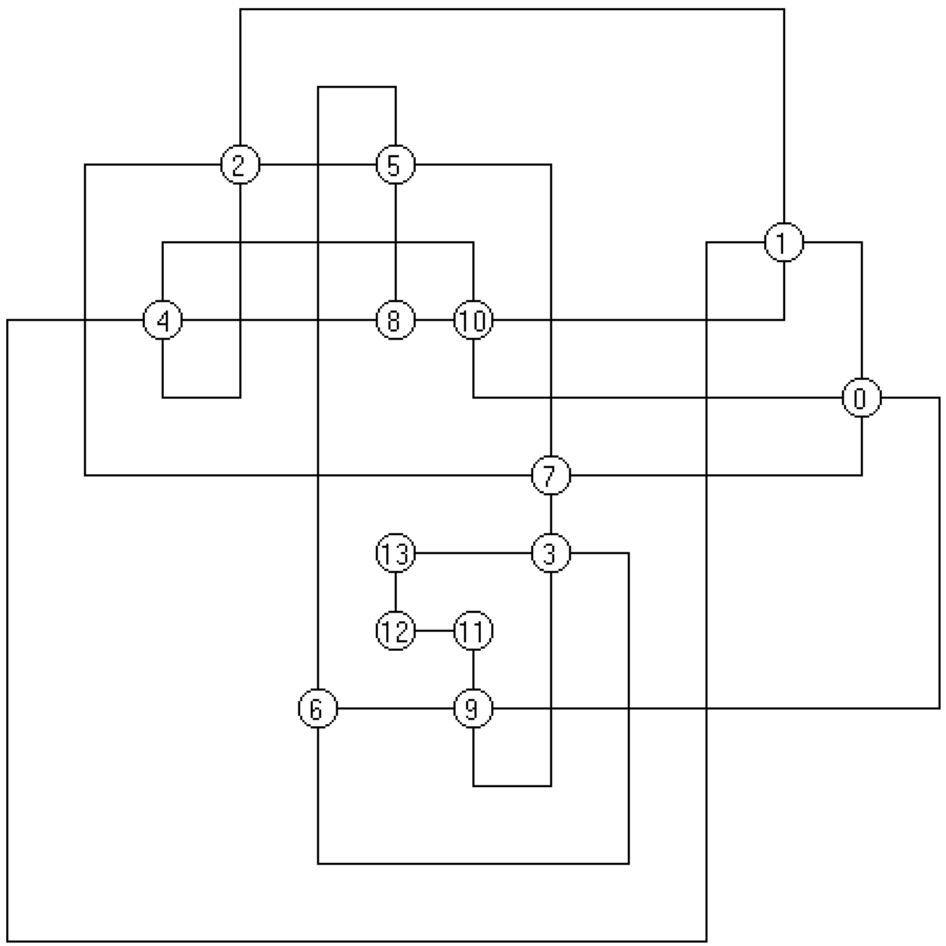
Bends

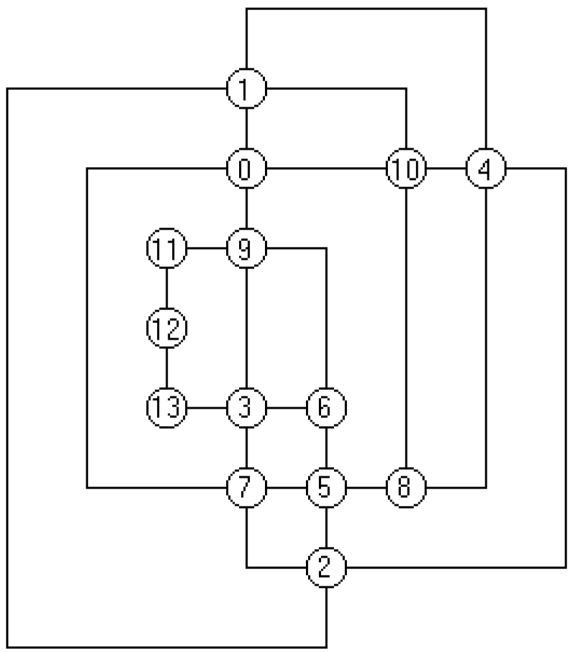


Area



Orthogonal drawings: Comparison

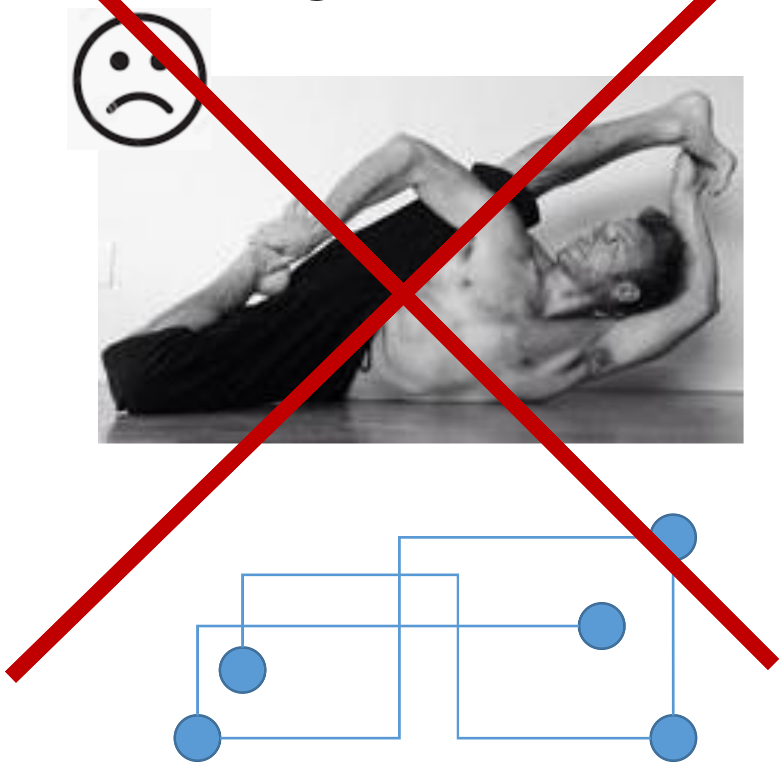




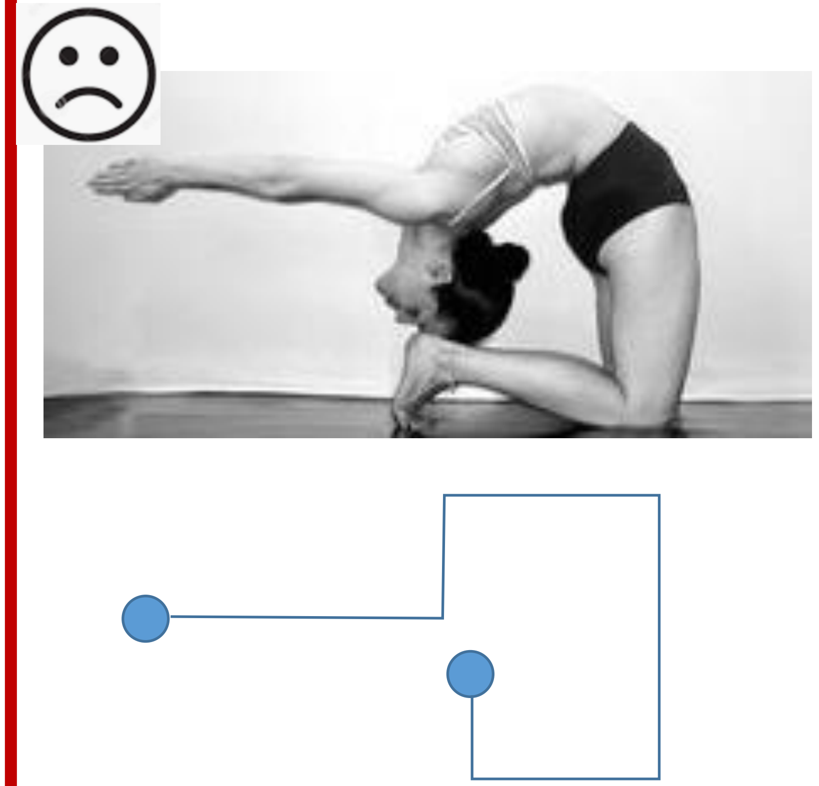
- no crossings
- fewer bends
- smaller area

In this talk: Bend Minimization

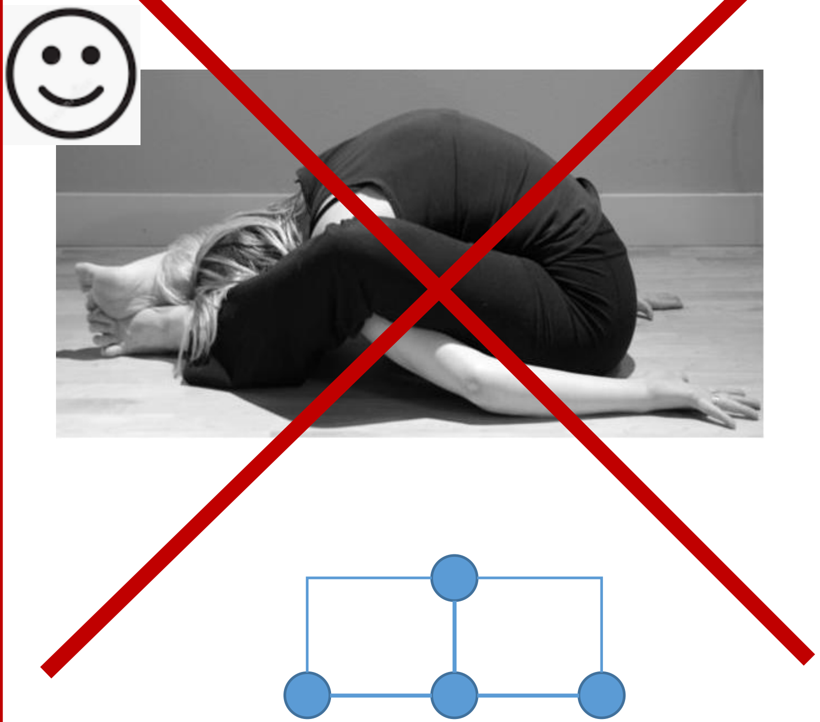
Crossings



Bends

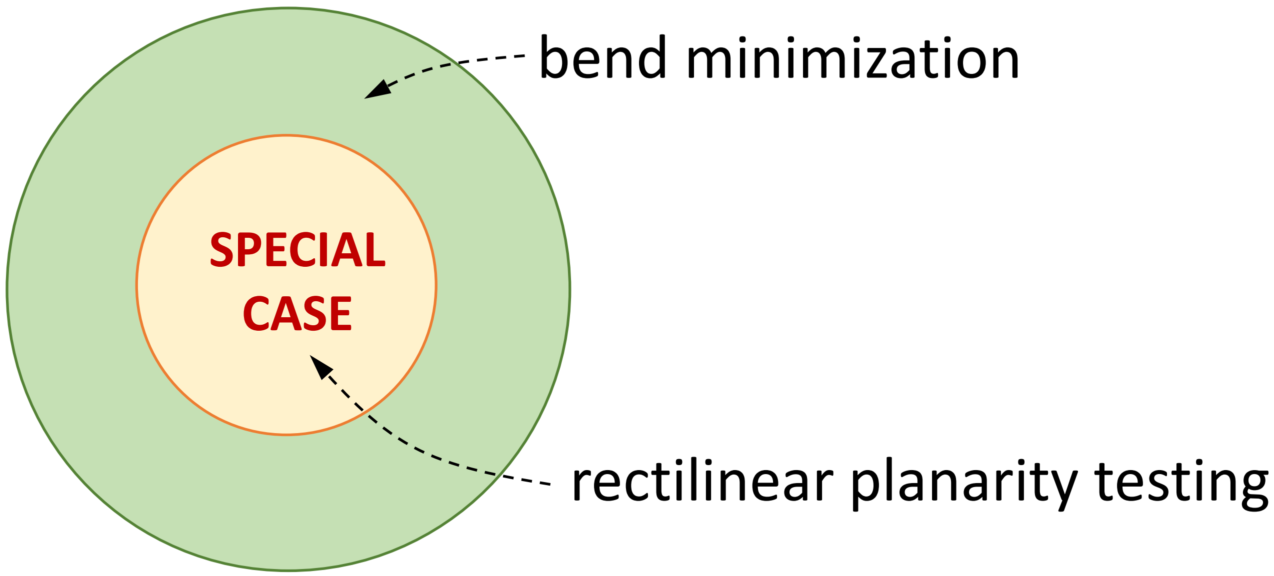


Area



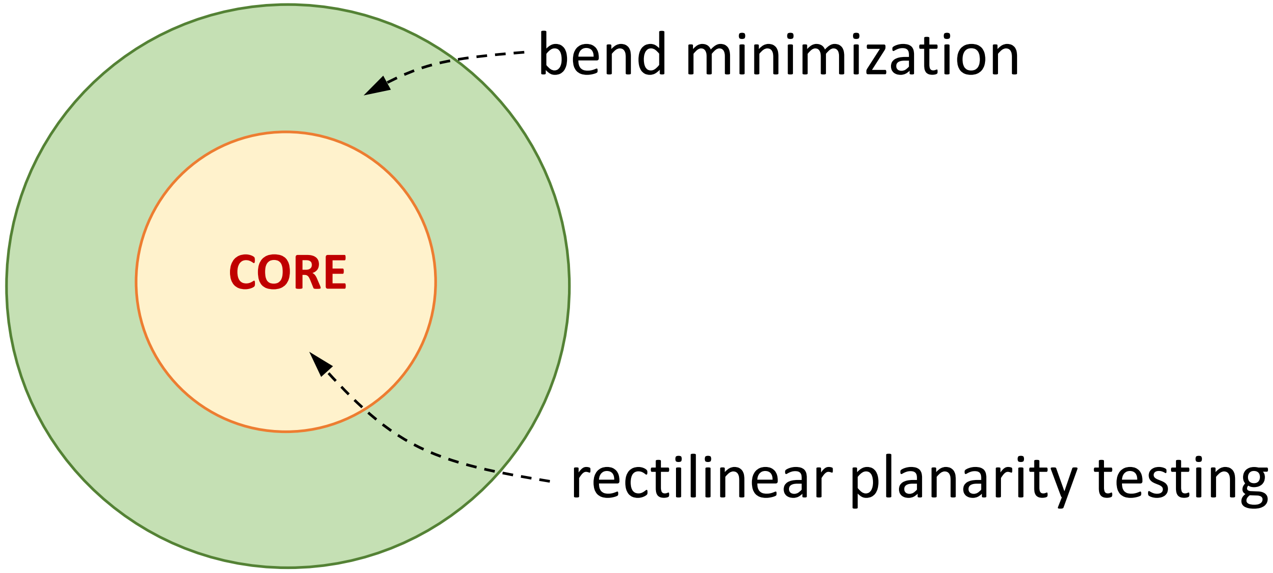
Bend minimization and rectilinear planarity

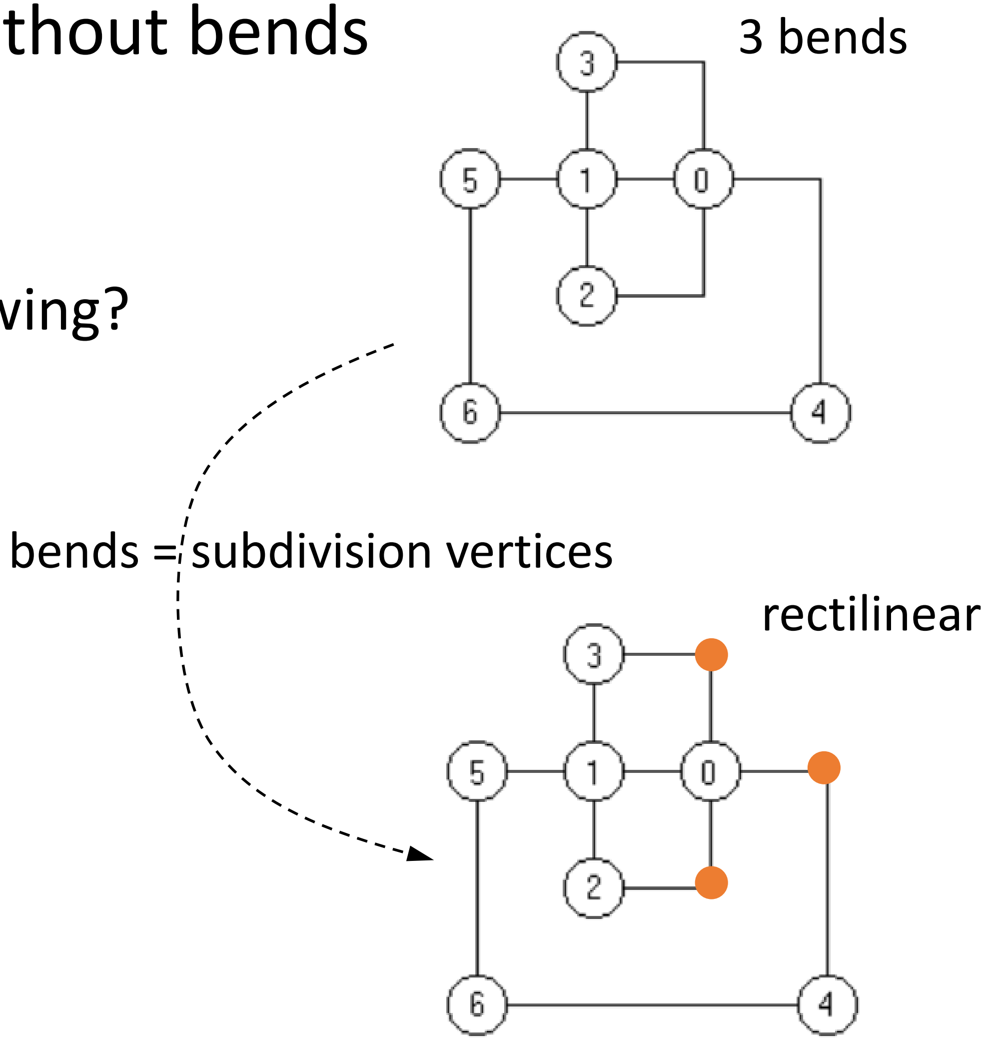
- Rectilinear drawing = orthogonal drawing without bends
- Rectilinear planarity testing:
  - Instance: planar 4-graph G
  - Question: does G admit a rectilinear planar drawing?



Bend minimization and rectilinear planarity

- Rectilinear drawing = orthogonal drawing without bends
- Rectilinear planarity testing:
  - Instance: planar 4-graph G
  - Question: does G admit a rectilinear planar drawing?

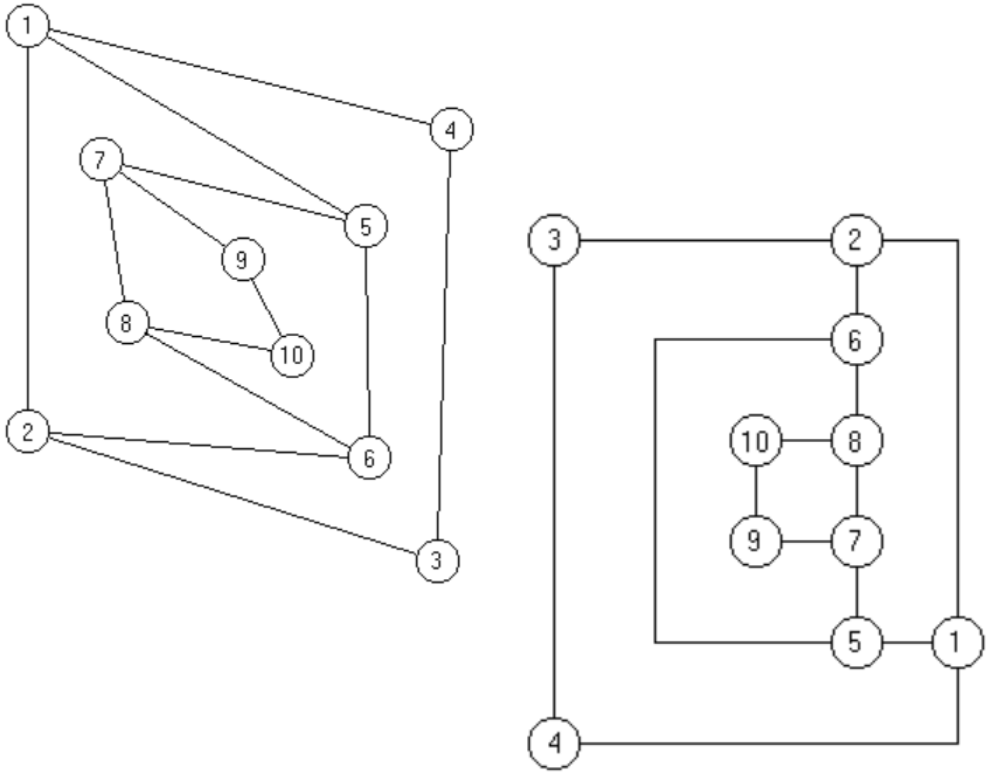




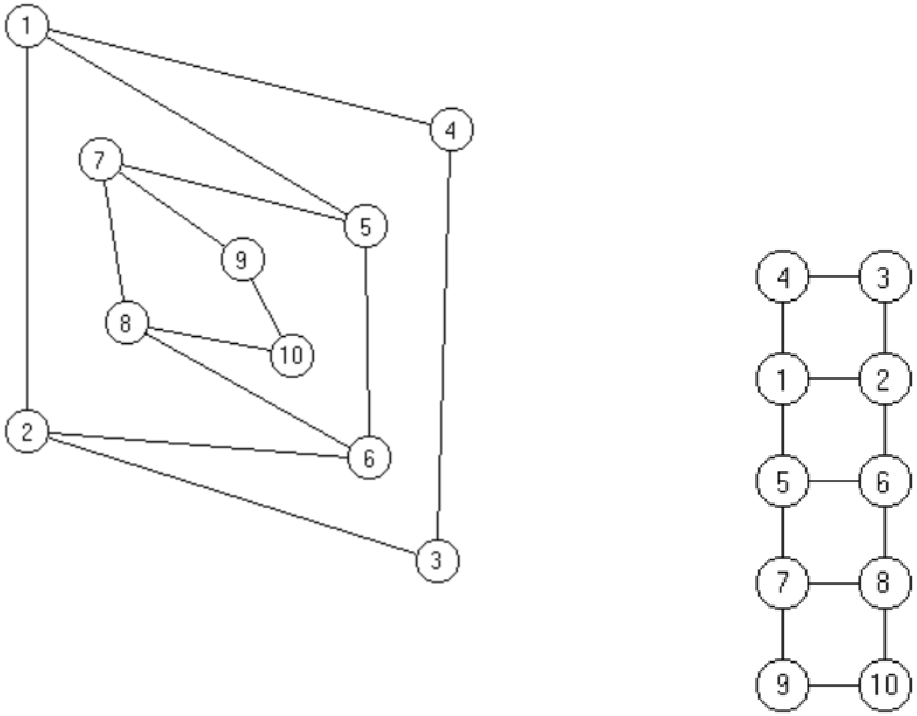
Fixed and Variable Embedding

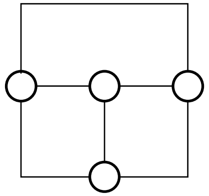
- Two possible scenarios

fixed embedding



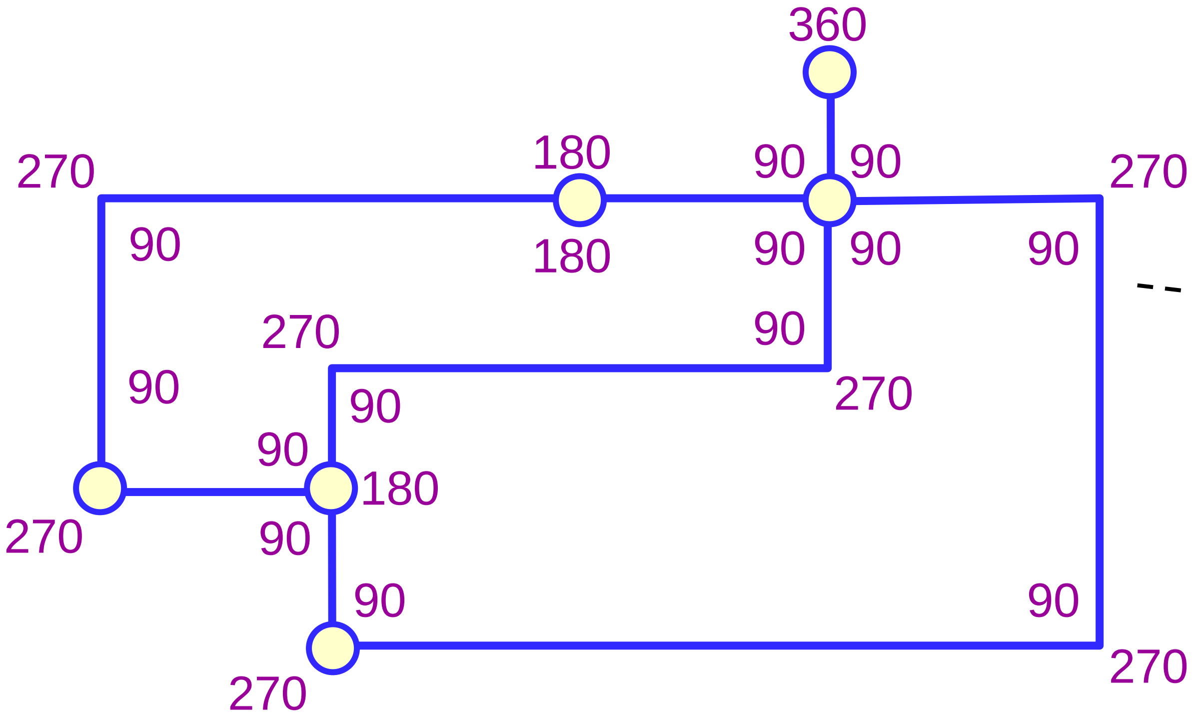
variable embedding





Fixed Embedding: The main result

- O(n^2 log n) time for general 4-graphs [Tamassia, SIAM J. Comp. 1987]
  - based on an elegant reduction to a min-cost flow problem



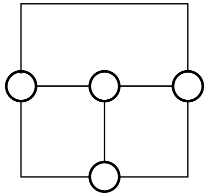
plane 4-graph
(planar embedding)

↓ O(n^2 log n)

orthogonal representation
(angles + bends)

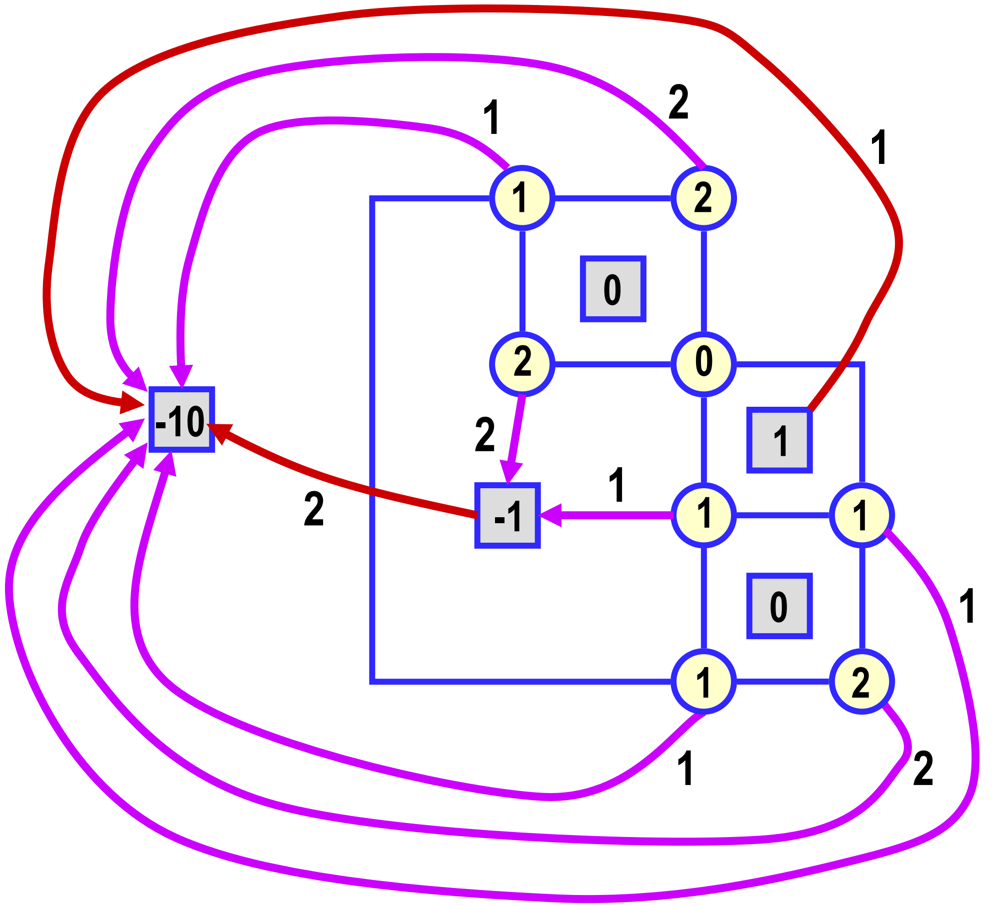
↓ O(n)

orthogonal drawing
(coordinates)



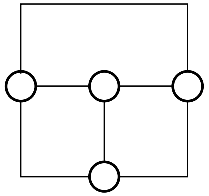
Flow and orthogonal representations

Flow network ⇔ orthogonal representations (angles + bends)



[Tamassia, SIAM J. Comp. 1987]

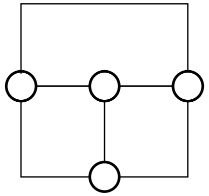
- vertices/faces with deg < 4 supply flow
- faces with deg > 4 demand flow
- adjacent faces can exchange flow
- 1 unit of flow exchanged = 1 bend
- total flow cost = total number of bends



Fixed Embedding: Further improvements

Improvements of Tamassia's result derive from subsequent faster min-cost flow algorithms:

- $O(n^{1.75} \log n)$ time [Garg & Tamassia, GD 1996]
- $O(n^{1.5})$ time [Cornelsen & Karrenbauer, JGAA 2012]

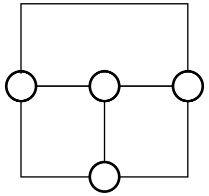


Fixed Embedding: Open problems

Improvements of Tamassia's result derive from subsequent faster min-cost flow algorithms:

- $O(n^{1.75} \log n)$ time [Garg & Tamassia, GD 1996]
- $O(n^{1.5})$ time [Cornelsen & Karrenbauer, JGAA 2012]

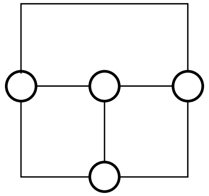
Open Problem 1: Does there exist an O(n)-time bend-minimization algorithm for plane 4-graphs?



Fixed Embedding: Open problems

- Partial answers:
  - O(n)-time algorithm for <u>plane</u> 3-graphs [Rahman & Nishizeki, WG 2002]
    - extends an O(n)-time rectilinear planarity testing for plane 3-graphs [Rahman, Nishizeki, Naznin, GD 2001 & JGAA 2003]
  - O(n)-time algorithm for <u>plane</u> series-parallel graphs (SP-graphs) [D., Kaufmann, Liotta, Ortali, GD 2020 & Algorithmica 2022]

Open Problem 1: Does there exist an O(n)-time bend-minimization algorithm for plane 4-graphs?



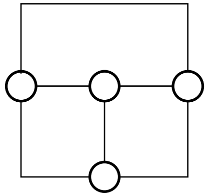
Fixed Embedding: Open problems

not flow-based!

- Partial answers:
  - O(n)-time algorithm for <u>plane</u> 3-graphs [Rahman & Nishizeki, WG 2002]
    - extends an O(n)-time rectilinear planarity testing for plane 3-graphs [Rahman, Nishizeki, Naznin, GD 2001 & JGAA 2003]
  - O(n)-time algorithm for <u>plane</u> series-parallel graphs (SP-graphs) [D., Kaufmann, Liotta, Ortali, GD 2020 & Algorithmica 2022]

Open Problem 1: Does there exist an O(n)-time bend-minimization algorithm for plane 4-graphs?

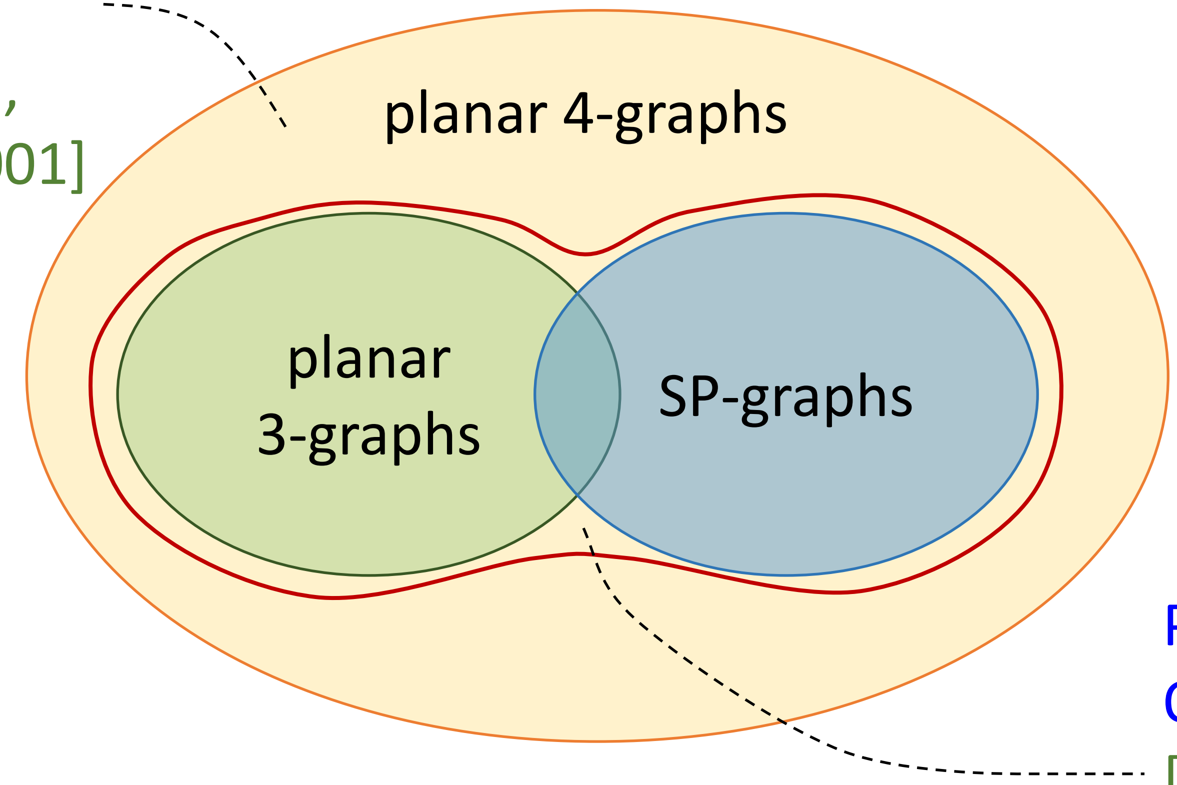
Open Problem 2: Does there exist an O(n)-time bend-minimization algorithm for *triconnected* plane 4-graphs?



Variable Embedding: First overview

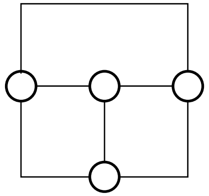
NP-hard (even rectilinear planarity testing)

[Garg & Tamassia, SIAM J. Comp. 2001]



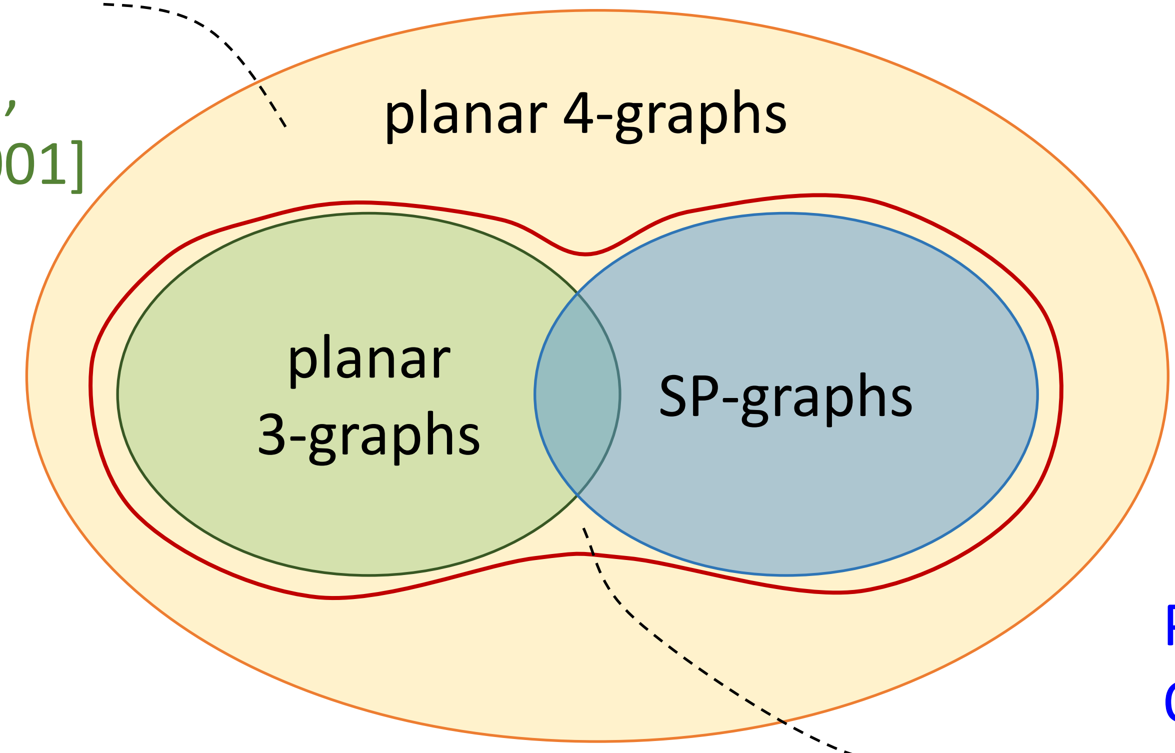
Polynomial-time solvable $O(n^5 \log n)$ and $O(n^4)$ time

[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Variable Embedding: First overview

NP-hard (even rectilinear planarity testing)
[Garg & Tamassia, SIAM J. Comp. 2001]



Polynomial-time solvable
$O(n^5 \log n)$ and $O(n^4)$ time
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]

introduces

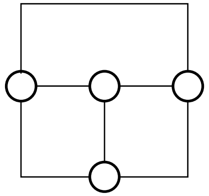
- «spirality»
- dynamic programming on SPQR-trees

Prominent open problems

Problem A. Establishing the exact time complexity of the bend-minimization and the rectilinear planarity testing problems for

- SP-graphs
- Planar 3-graphs

Question: Can we find linear-time algorithms?



Prominent open problems

Problem A. Establishing the exact time complexity of the bend minimization and the rectilinear planarity testing problems for

- SP-graphs
- Planar 3-graphs

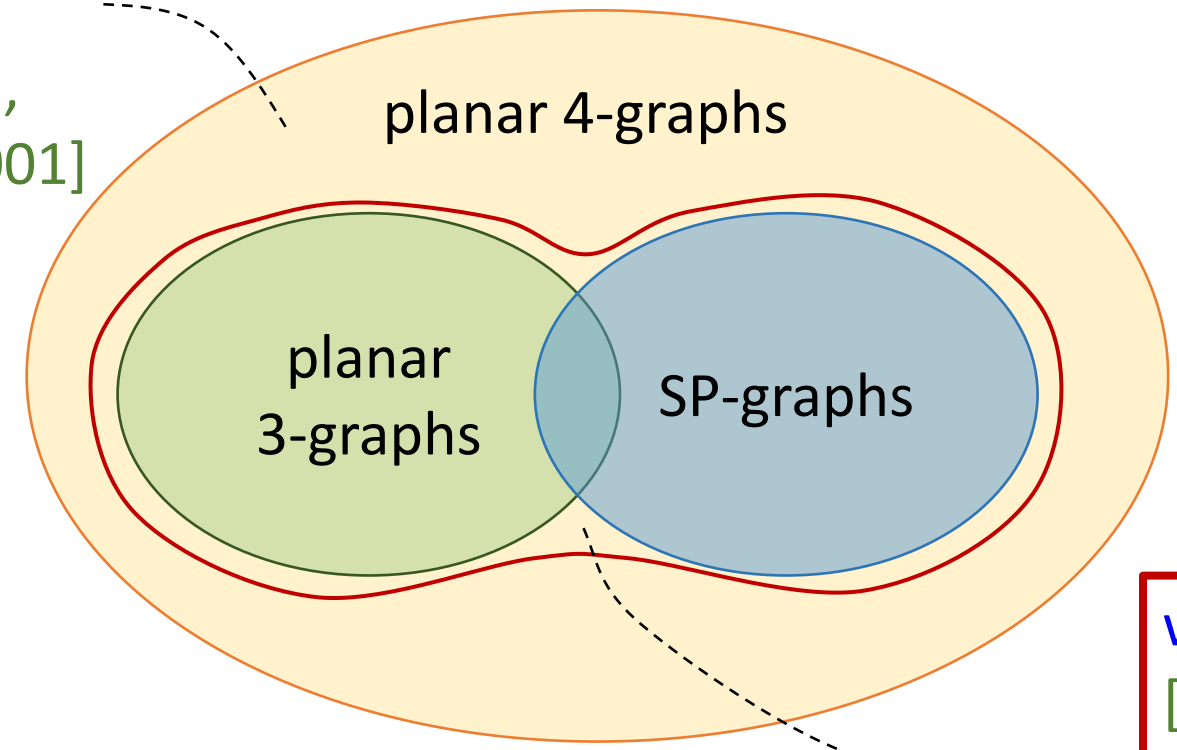
Question: Can we find linear-time algorithms?

Problem B. Exponential-time algorithms for general planar 4-graphs

Question: What about the existence of parameterized algorithms?

In the remainder of the talk: Problem A

NP-hard (even rectilinear planarity testing)
[Garg & Tamassia, SIAM J. Comp. 2001]



1

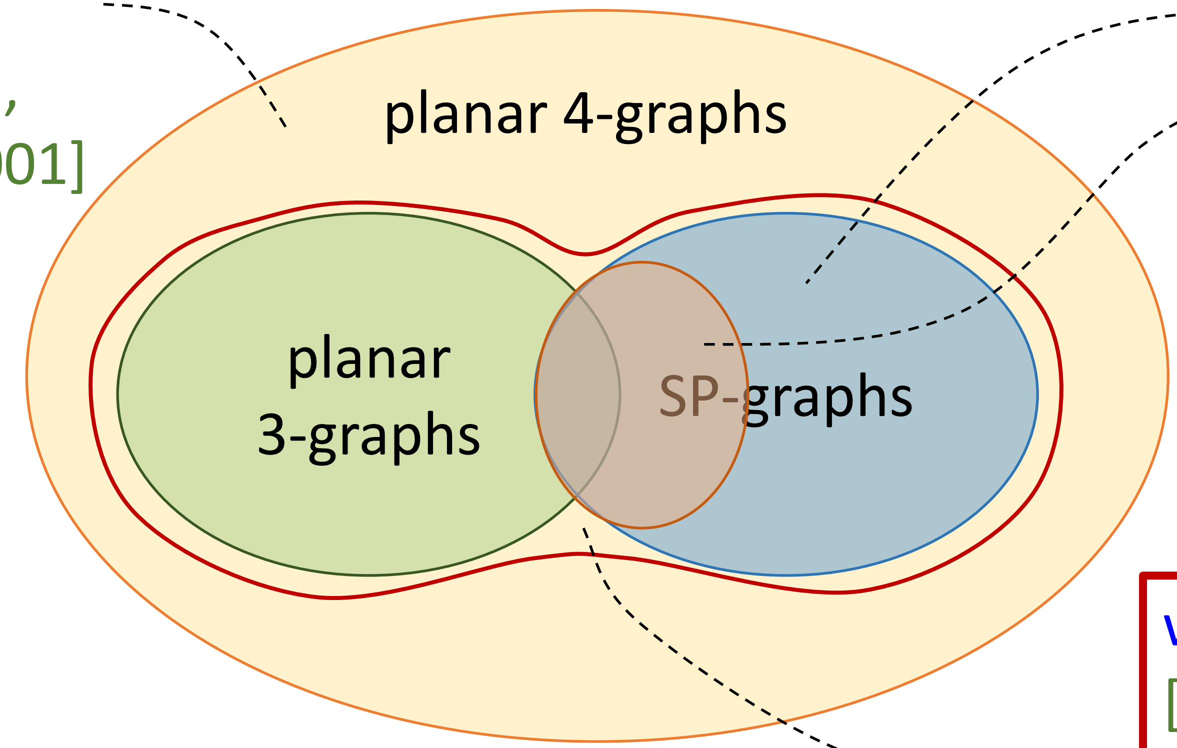
we revisit the strategy of [Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998] showing $O(n^4)$ complexity

introduces

- «spirality»
- dynamic programming on SPQR-trees

In the remainder of the talk: Problem A

NP-hard (even rectilinear planarity testing)
[Garg & Tamassia, SIAM J. Comp. 2001]



2

we discuss recent advances

- $O(n^3)$ bend minimization
- $O(n^2)$ rect. planarity testing
- $O(n)$ rect. planarity testing for *independent-parallel*

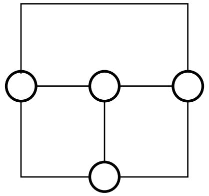
[D., Kaufmann, Liotta, Ortali, JGAA 2023]

1

we revisit the strategy of
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]
showing $O(n^4)$ complexity

introduces

- «spirality»
- dynamic programming on SPQR-trees



In the remainder of the talk: Problem A

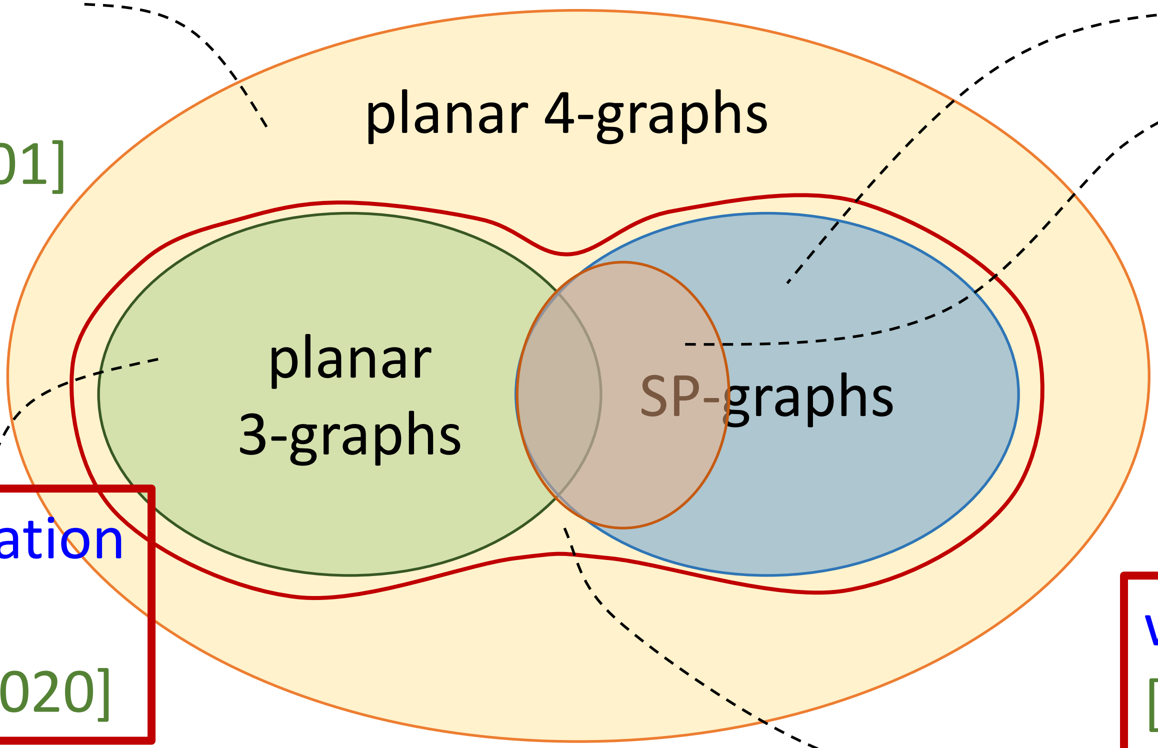
NP-hard (even rectilinear planarity testing)
[Garg & Tamassia, SIAM J. Comp. 2001]

2

we discuss recent advances

- $O(n^3)$ bend minimization
- $O(n^2)$ rect. planarity testing
- $O(n)$ rect. planarity testing for *independent-parallel*

[D., Kaufmann, Liotta, Ortali, JGAA 2023]



3

O(n) bend minimization
[D., Liotta, Ortali, Patrignani, SODA 2020]

1

we revisit the strategy of
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]
showing $O(n^4)$ complexity

introduces

- «spirality»
- dynamic programming on SPQR-trees

In the remainder of the talk: Problem A

NP-hard (even rectilinear planarity testing)
[Garg & Tamassia, SIAM J. Comp. 2001]

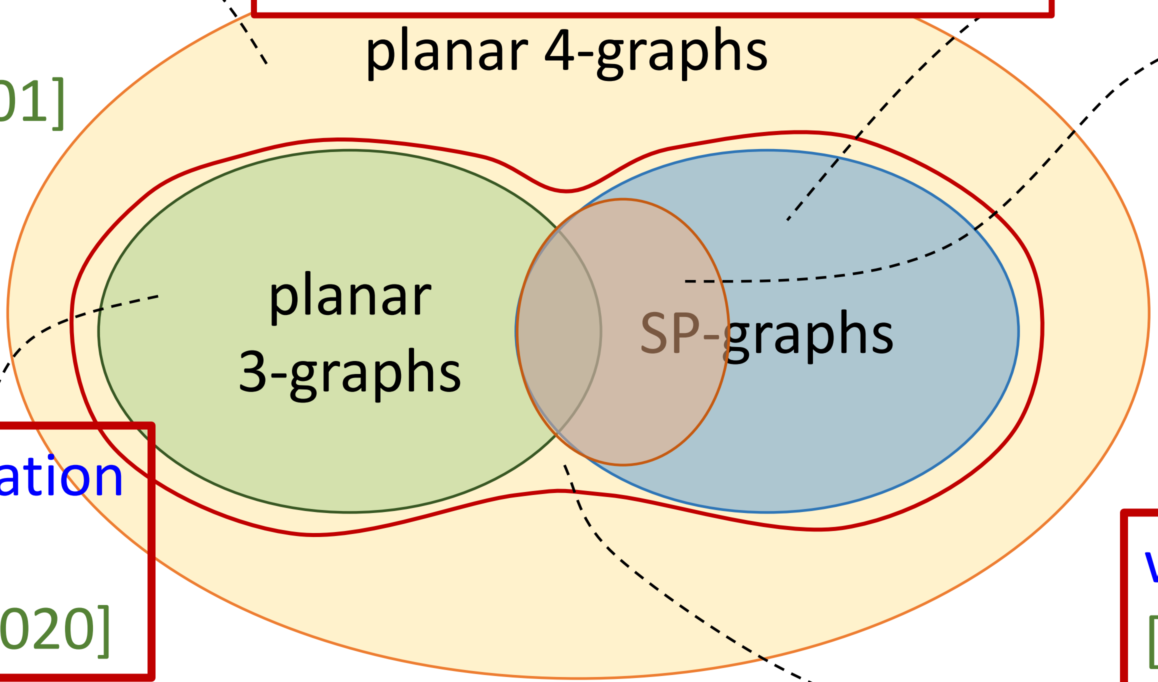
Remark: O(n) rect. plan. test. for *outerplanar graphs*
[Frati, CGTA 2022]

2

we discuss recent advances

- $O(n^3)$ bend minimization
- $O(n^2)$ rect. planarity testing
- O(n) rect. planarity testing for *independent-parallel*

[D., Kaufmann, Liotta, Ortali, JGAA 2023]



3

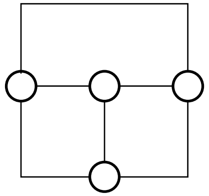
O(n) bend minimization
[D., Liotta, Ortali, Patrignani, SODA 2020]

1

we revisit the strategy of
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]
showing $O(n^4)$ complexity

introduces

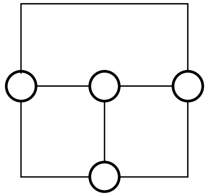
- «spirality»
- dynamic programming on SPQR-trees



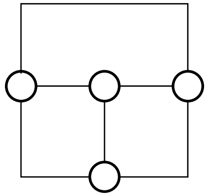
In the remainder of the talk: Problem B

parameterized complexity

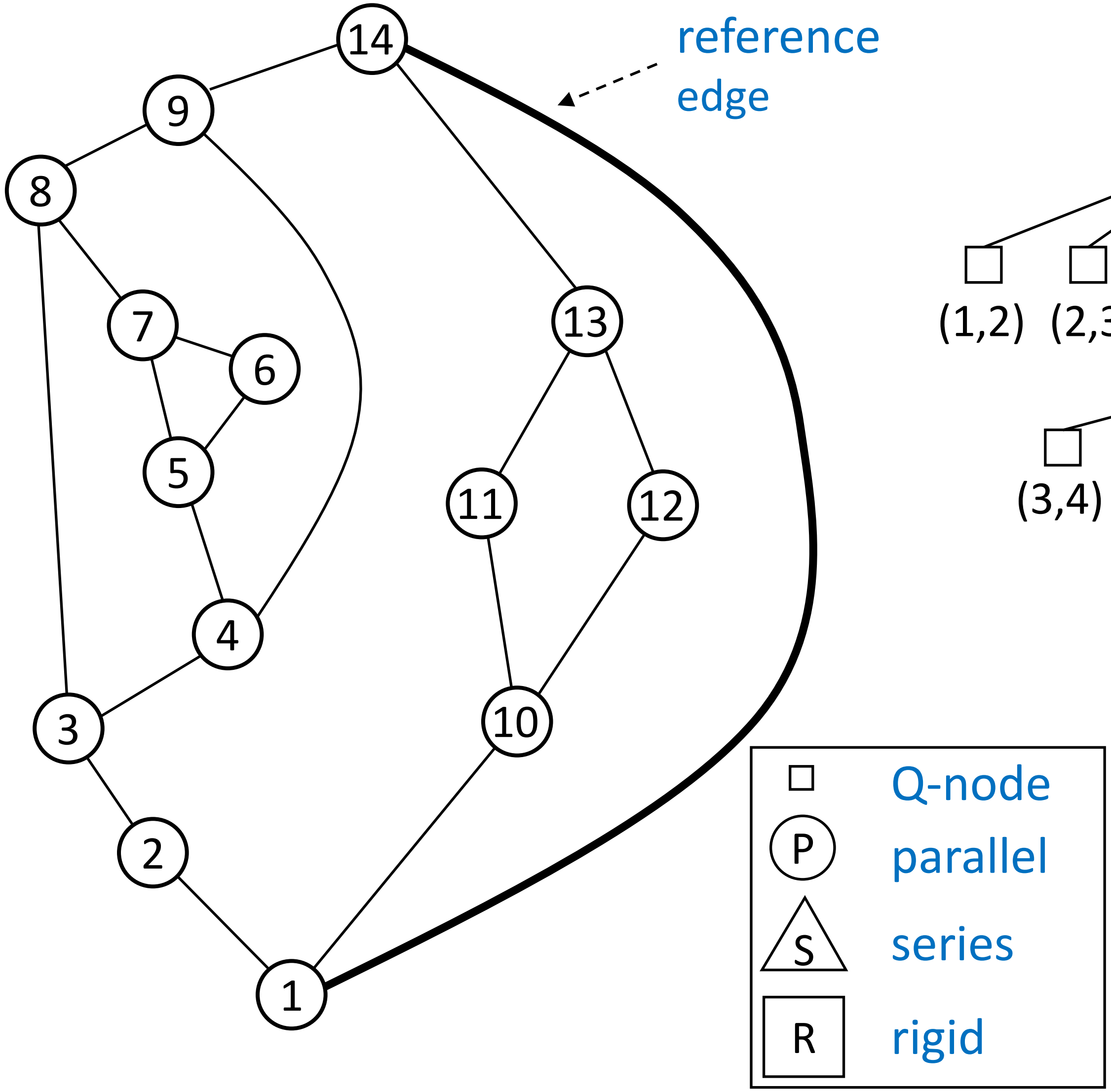
b (bends)	k (deg-2 vert.)	tw (treewidth)	b+k	b+tw	k+tw
Para-NP-hard [Garg & Tamassia, 2001]	Para-NP-hard [Di Giacomo, D., Liotta, Ortali, Montecchiani, 2023]	W[1]-hard [Jansen et al., 2023] XP [Di Giacomo, Liotta, Montecchiani, 2022]	**FPT** [Di Giacomo, D., Liotta, Ortali, Montecchiani, 2023] 4	W[1]-hard [implied]	W[1]-hard [Di Giacomo, D., Liotta, Ortali, Montecchiani, 2023]

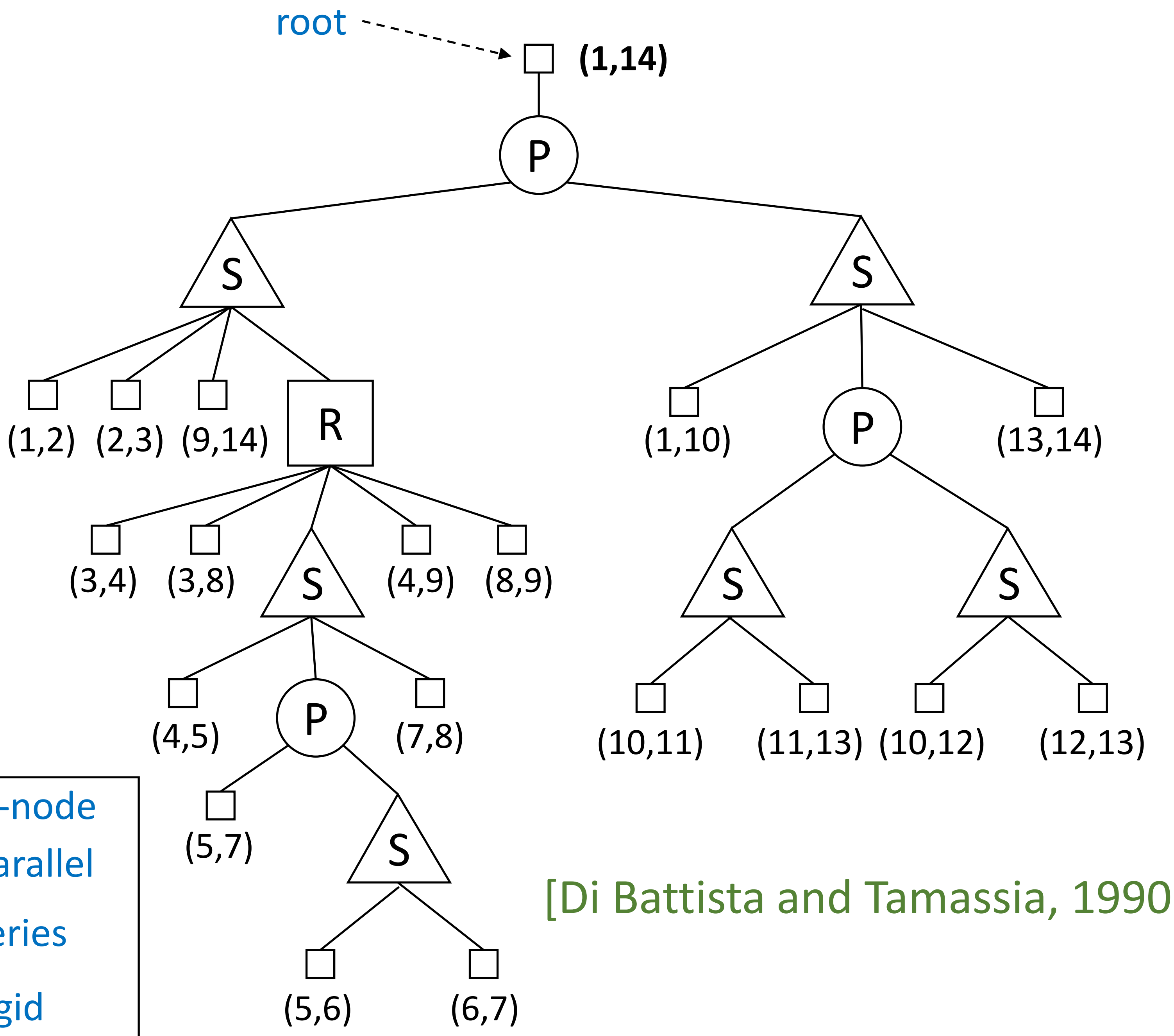


\begin{SPQR-trees}



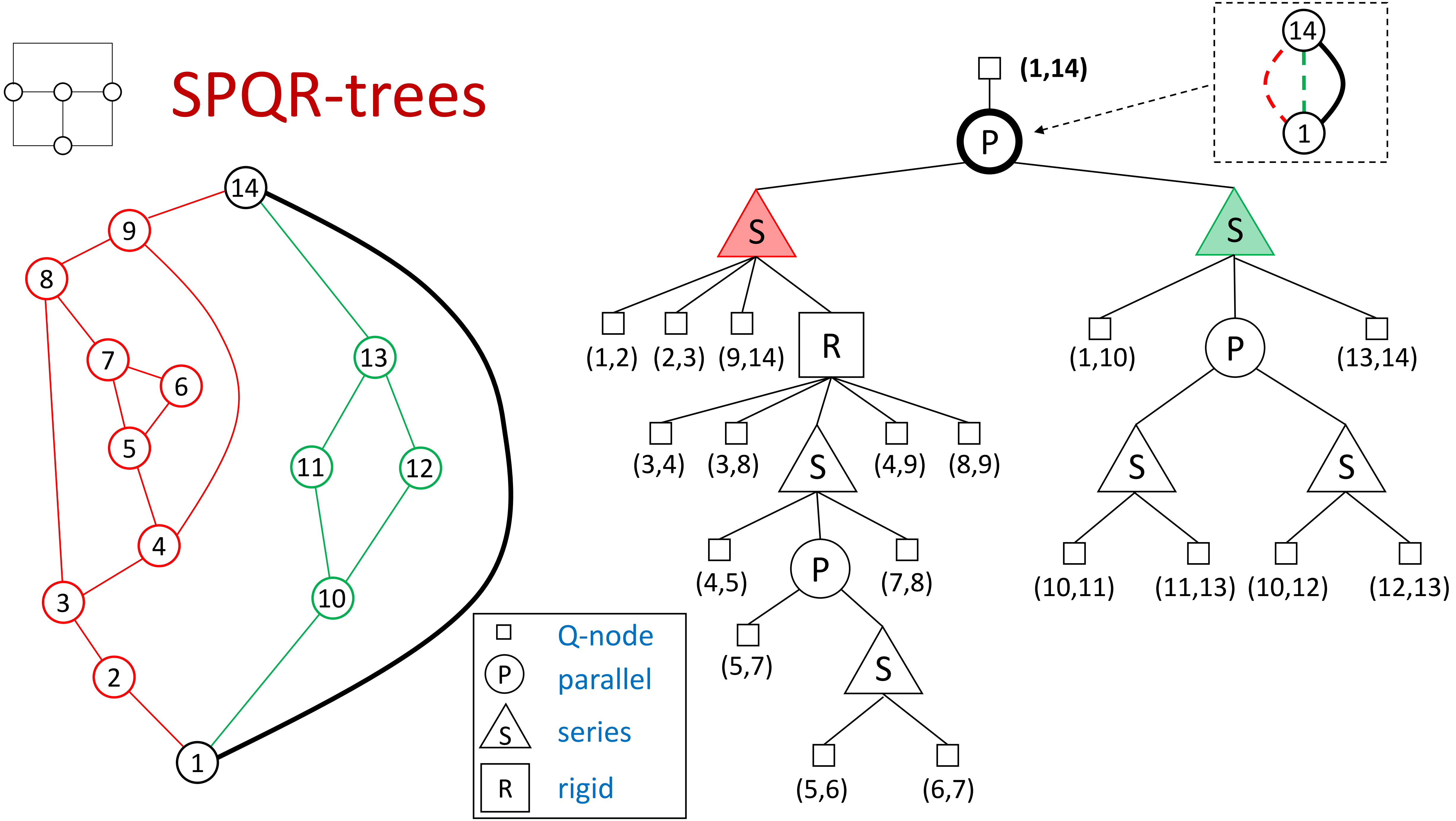
SPQR-trees



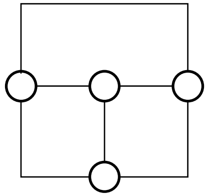


[Di Battista and Tamassia, 1990]

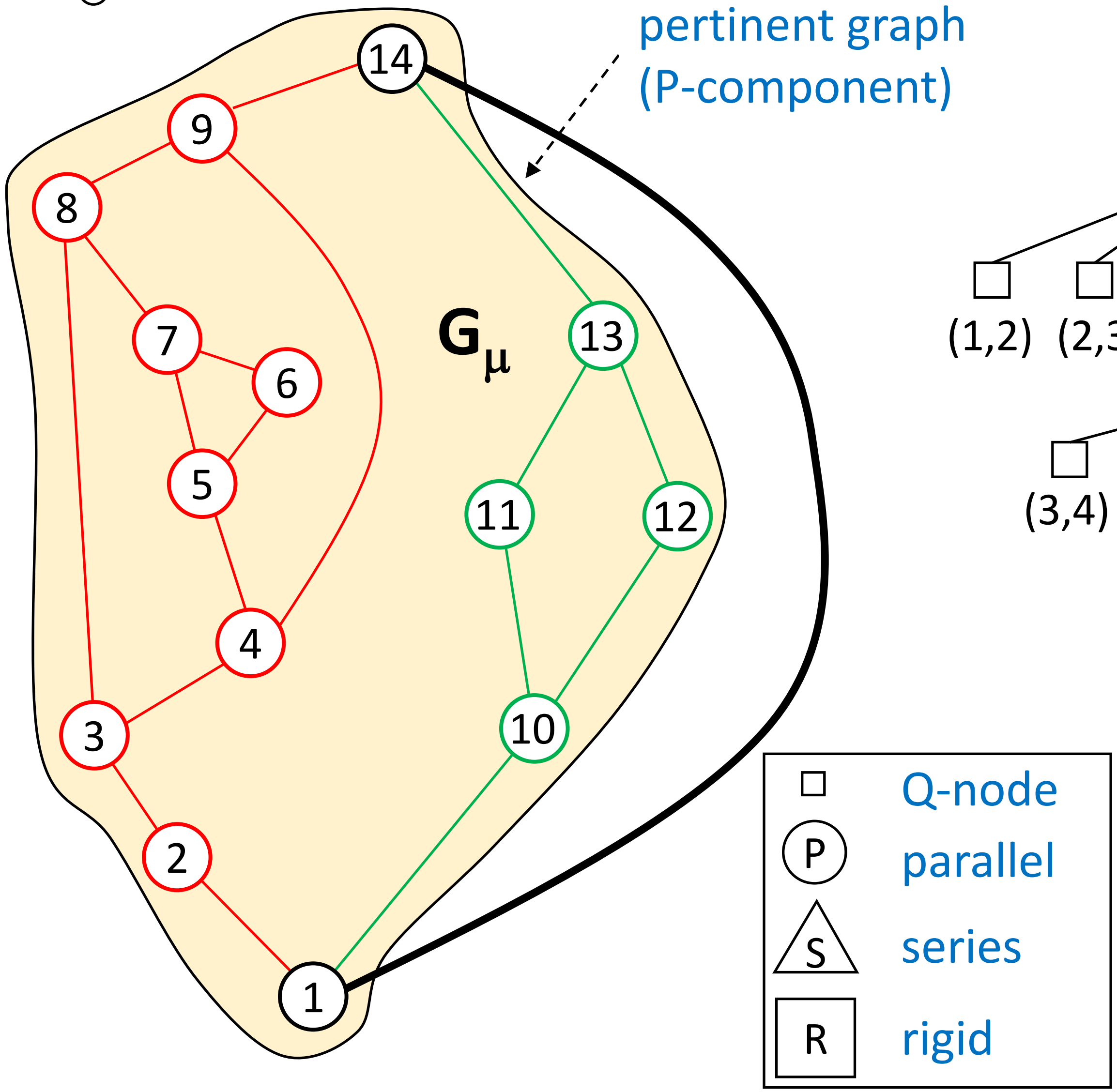
SPQR-trees



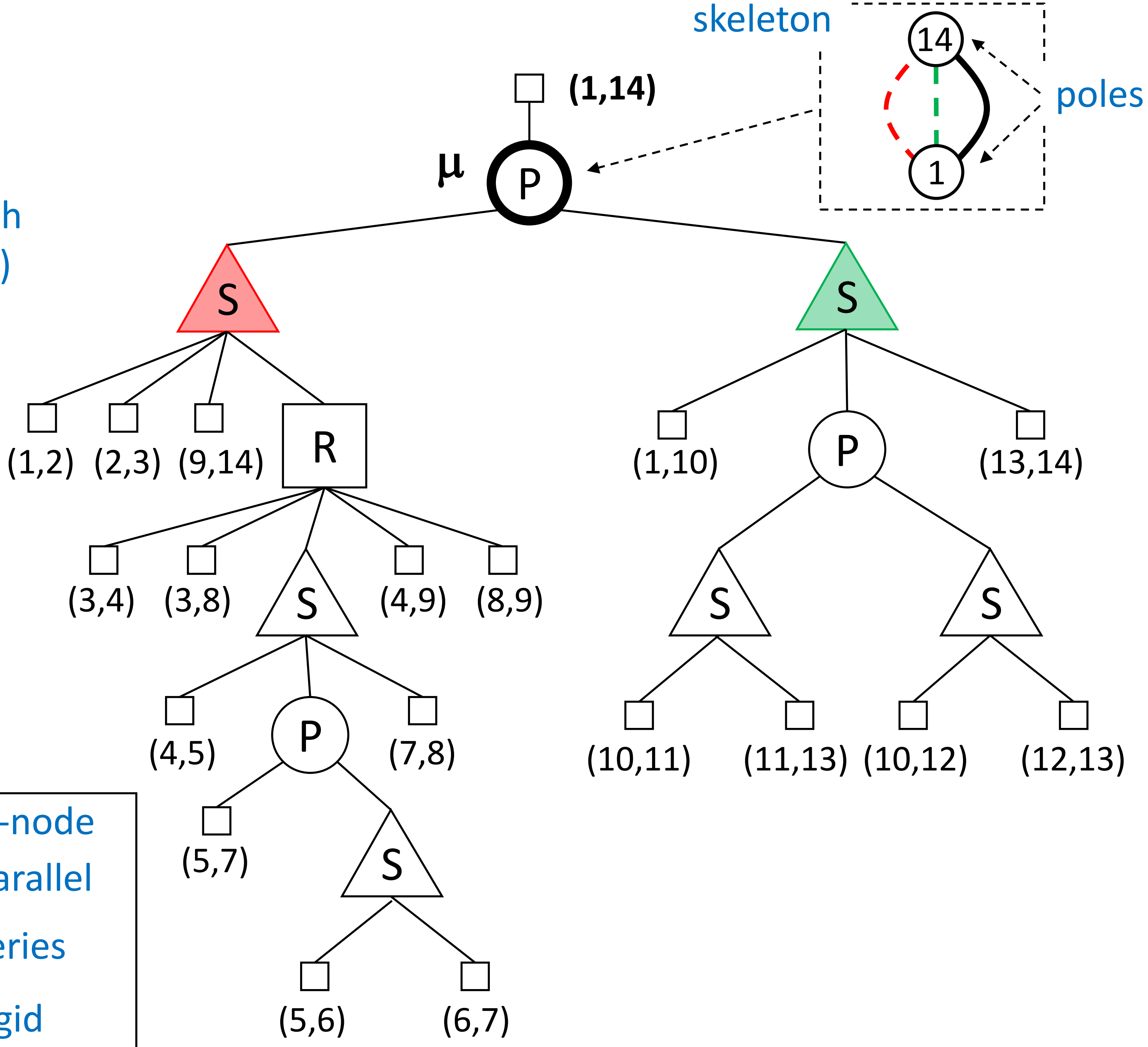
(1,14)
P
14
1
S
S
(1,2)
(2,3)
(9,14)
R
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
S
S
(4,5)
P
(7,8)
(10,11)
(11,13)
(10,12)
(12,13)
(5,7)
S
(5,6)
(6,7)
Q-node
parallel
series
rigid
14
9
8
7
6
5
4
3
2
1
13
11
12
10



SPQR-trees

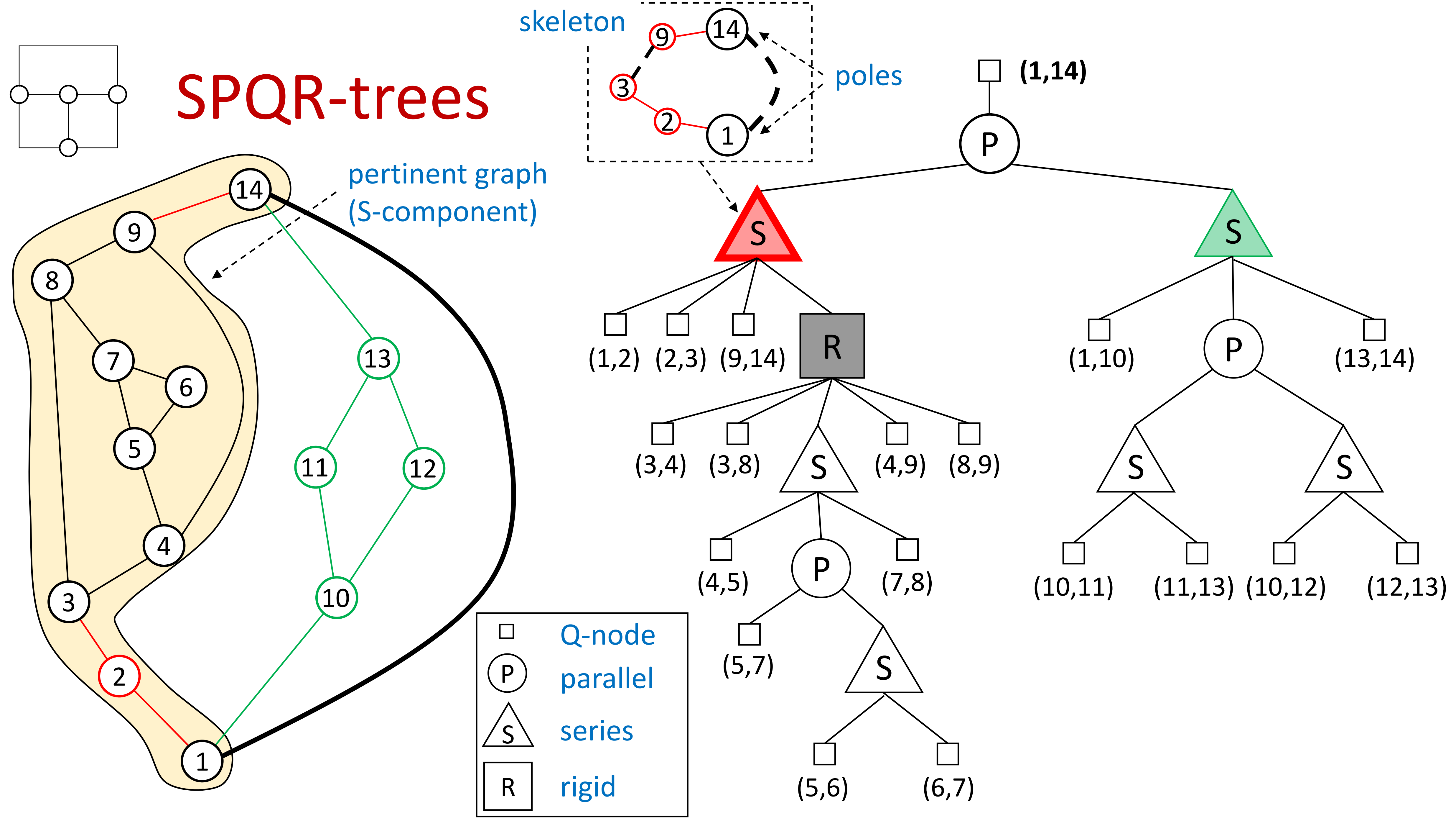


pertinent graph
(P-component)
G_μ
14
9
8
7
6
5
4
3
2
1
13
11
12
10
Q-node
P
parallel
S
series
R
rigid



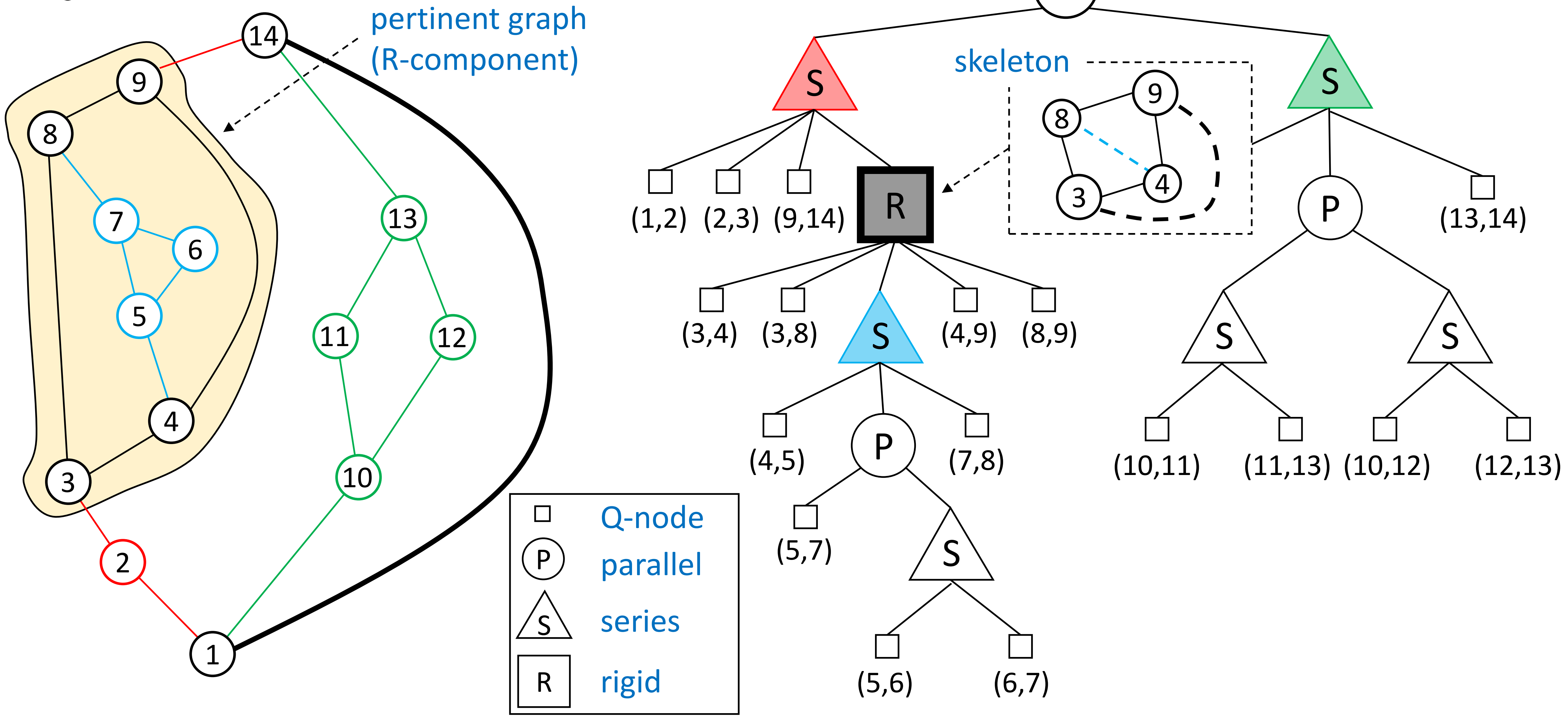
skeleton
poles
14
1
(1,14)
μ
P
S
S
(1,2)
(2,3)
(9,14)
R
(3,4)
(3,8)
S
(4,9)
(8,9)
(4,5)
P
(7,8)
(5,7)
S
(5,6)
(6,7)
(1,10)
P
(13,14)
S
S
(10,11)
(11,13)
(10,12)
(12,13)

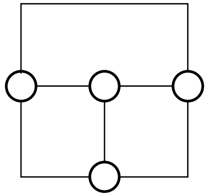
SPQR-trees



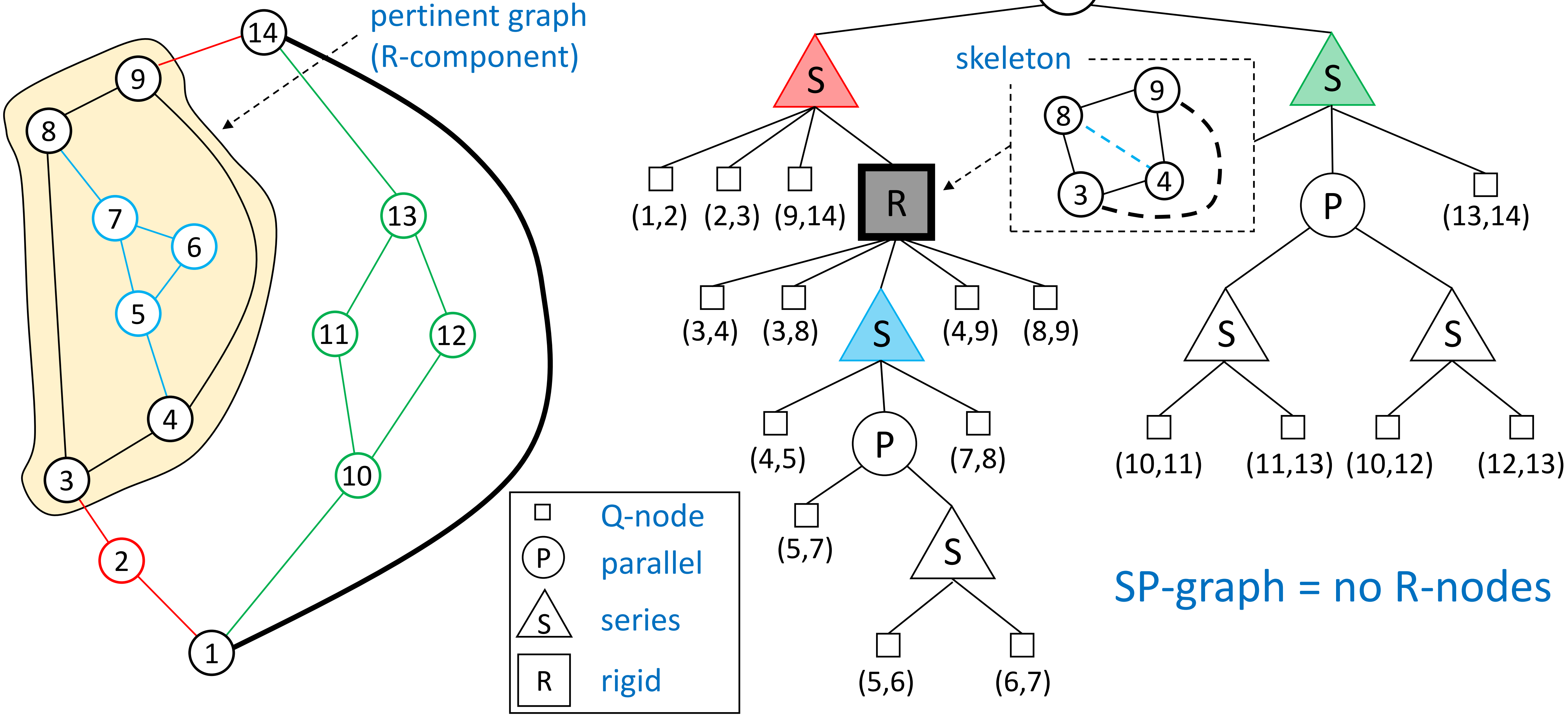
skeleton
poles
pertinent graph
(S-component)
(1,14)
P
S
S
(1,2) (2,3) (9,14)
R
(1,10)
P
(13,14)
(3,4) (3,8)
S
(4,9) (8,9)
S
S
(4,5)
P
(7,8)
(10,11) (11,13) (10,12) (12,13)
(5,7)
S
(5,6) (6,7)
Q-node
parallel
series
rigid

SPQR-trees

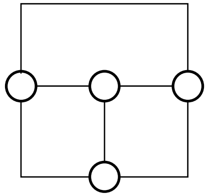

pertinent graph
(R-component)
skeleton
(1,14)
(1,2) (2,3) (9,14)
(3,4) (3,8) (4,9) (8,9)
(4,5) (7,8)
(5,7)
(5,6) (6,7)
(13,14)
(10,11) (11,13) (10,12) (12,13)
Q-node
parallel
series
rigid



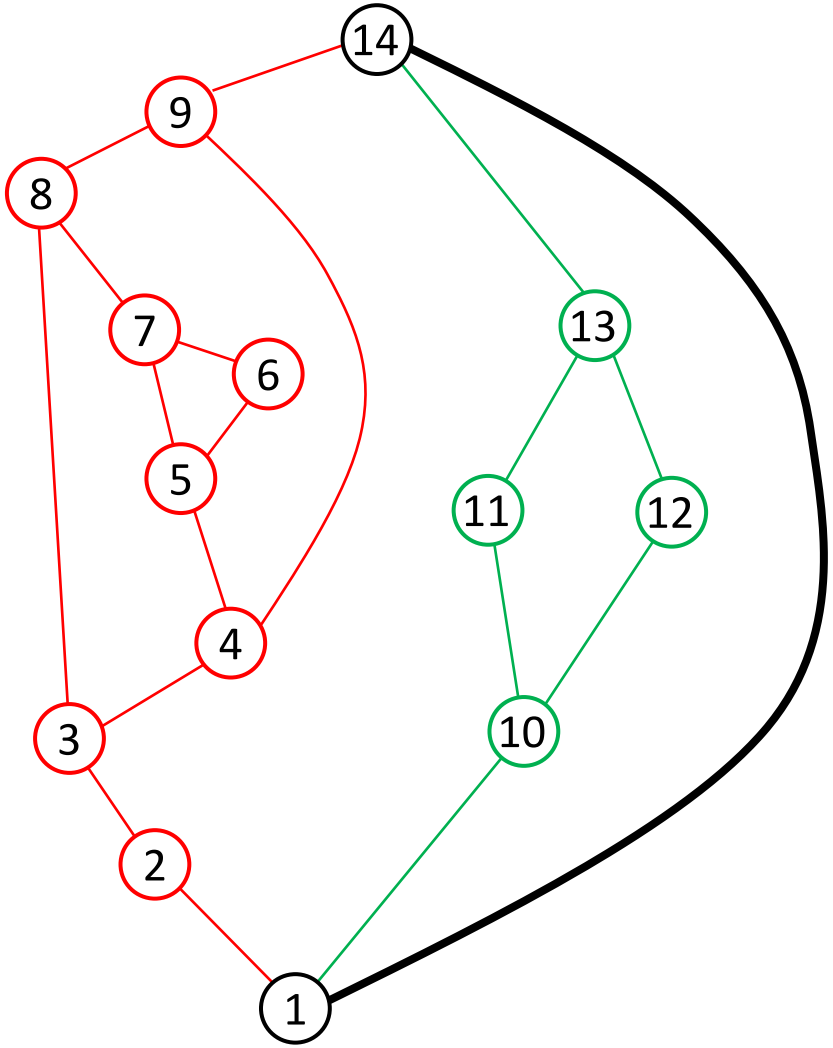
SPQR-trees

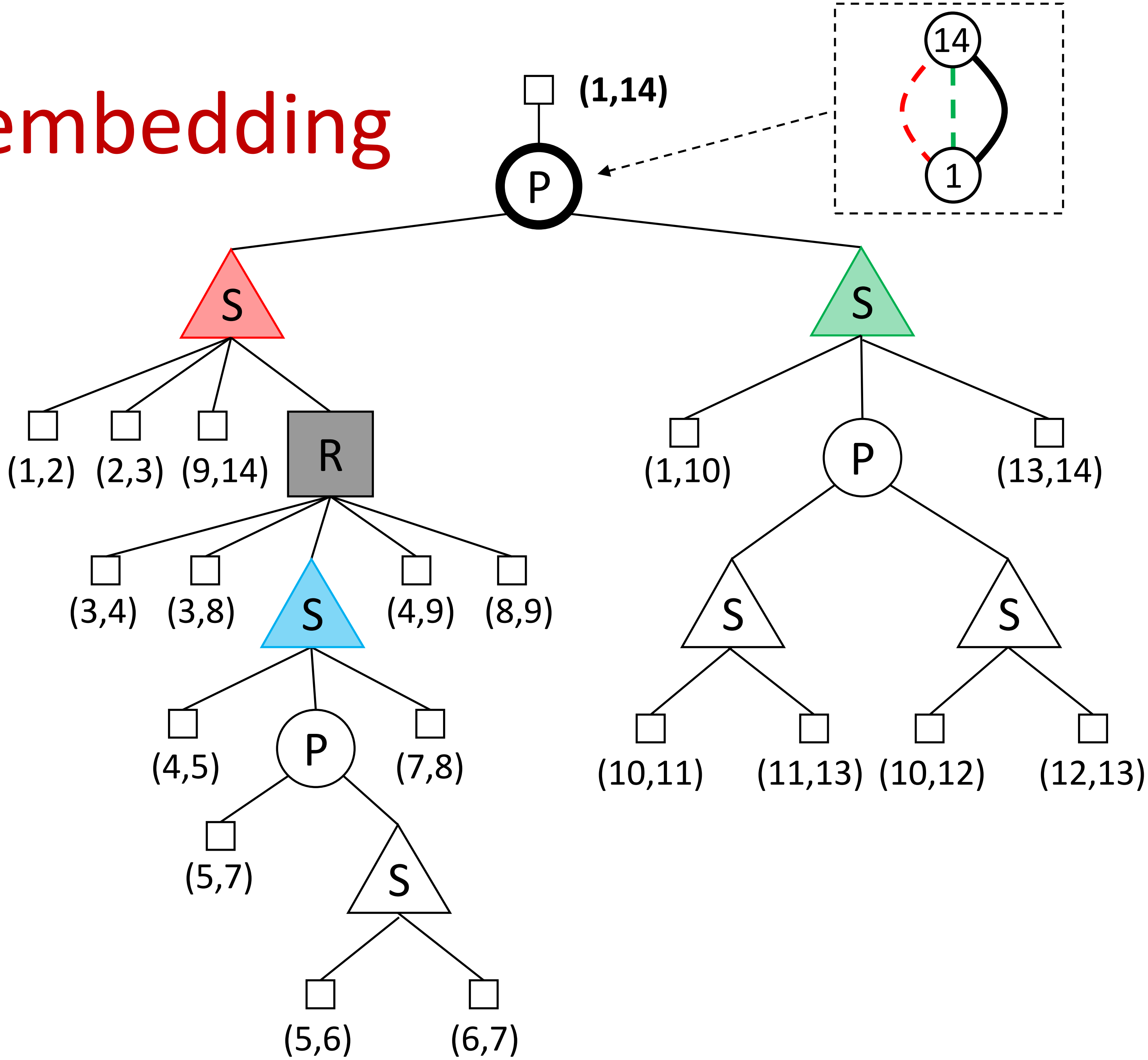


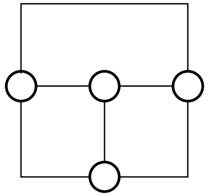
SP-graph = no R-nodes



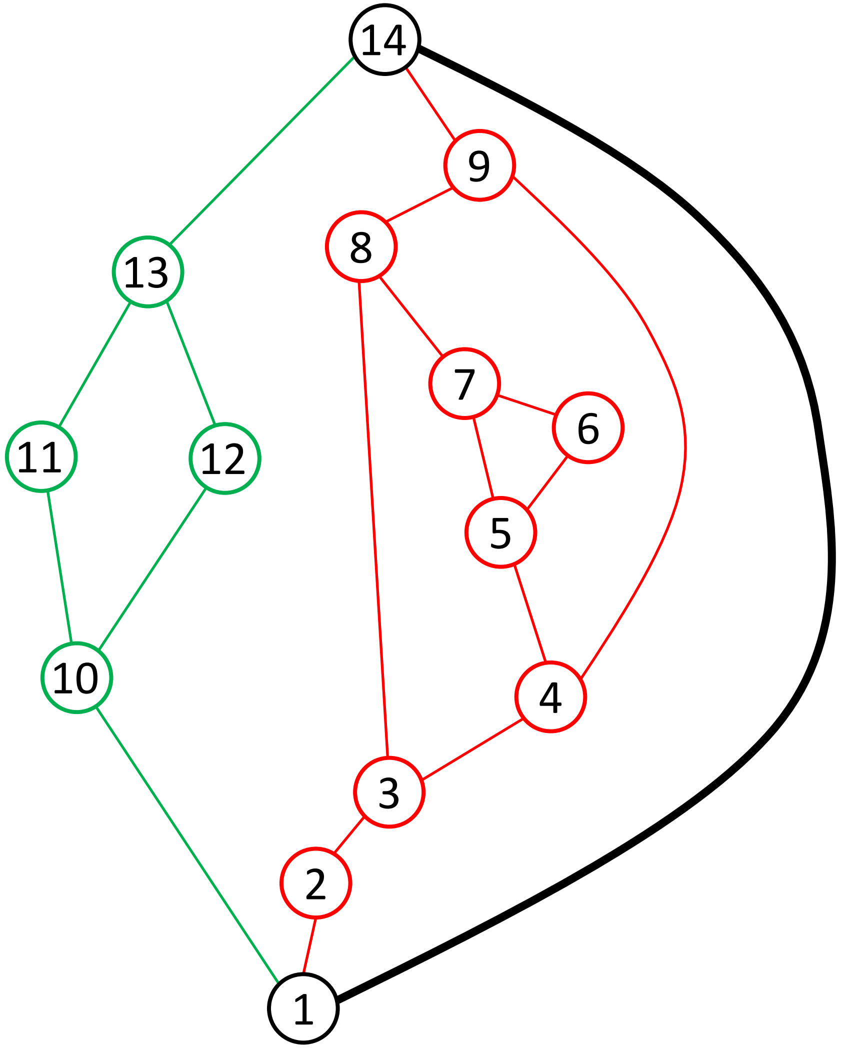
Changing the embedding

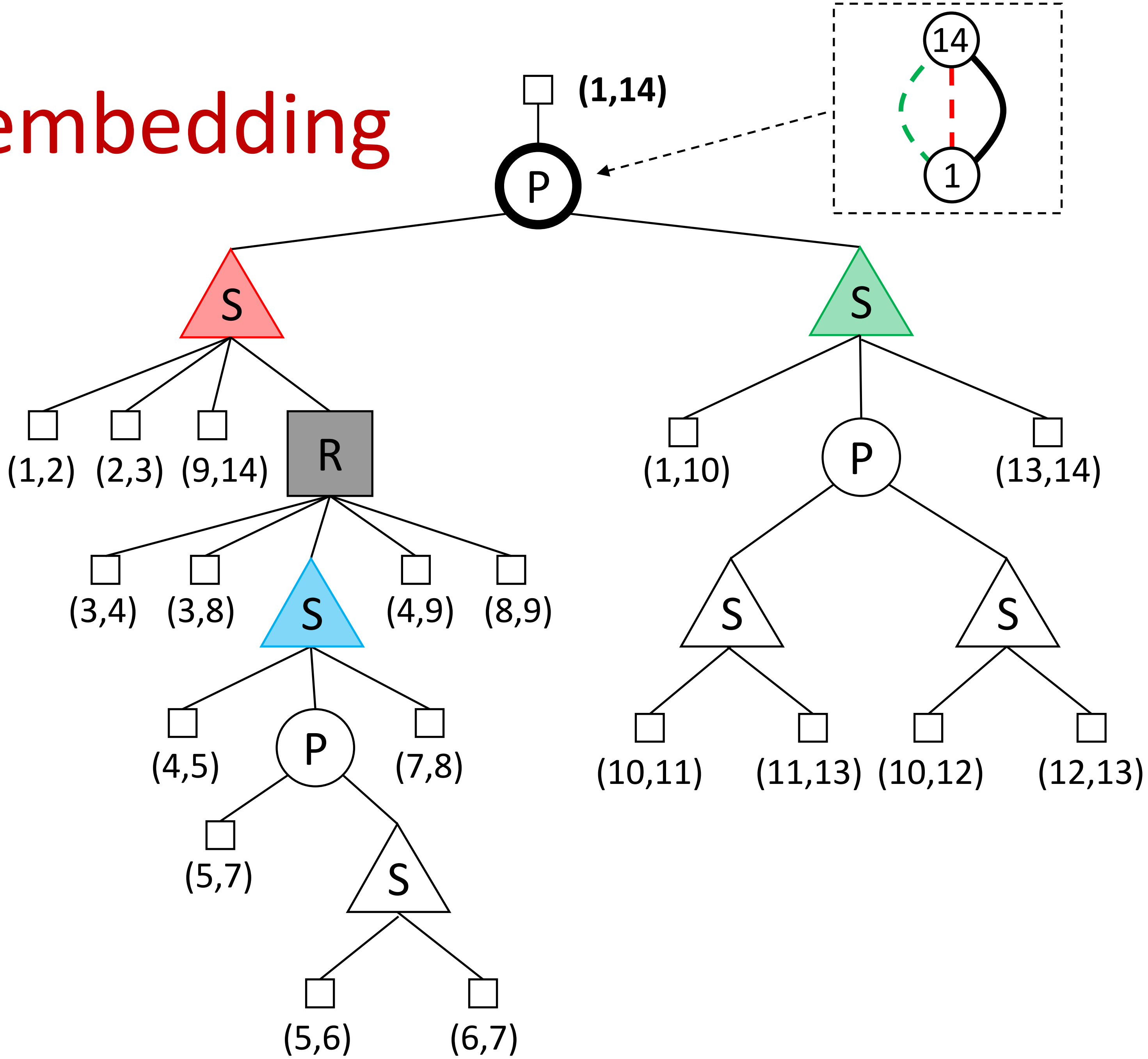


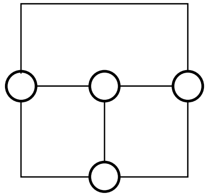




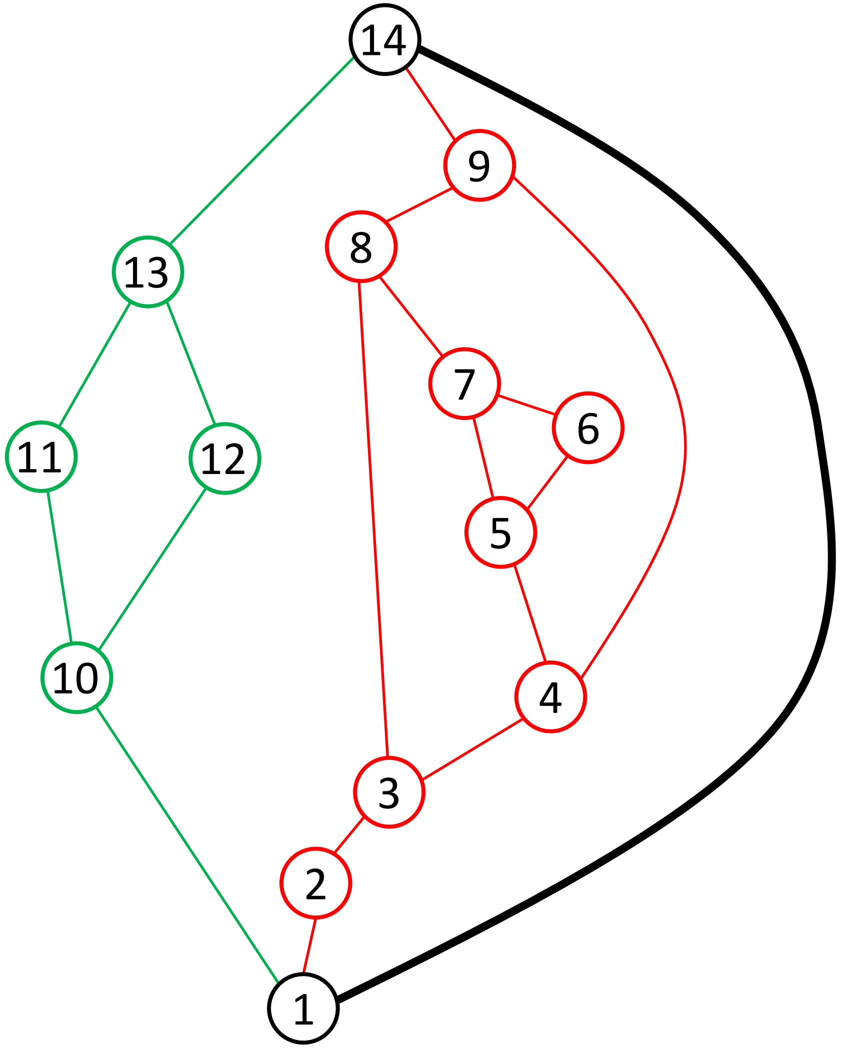
Changing the embedding

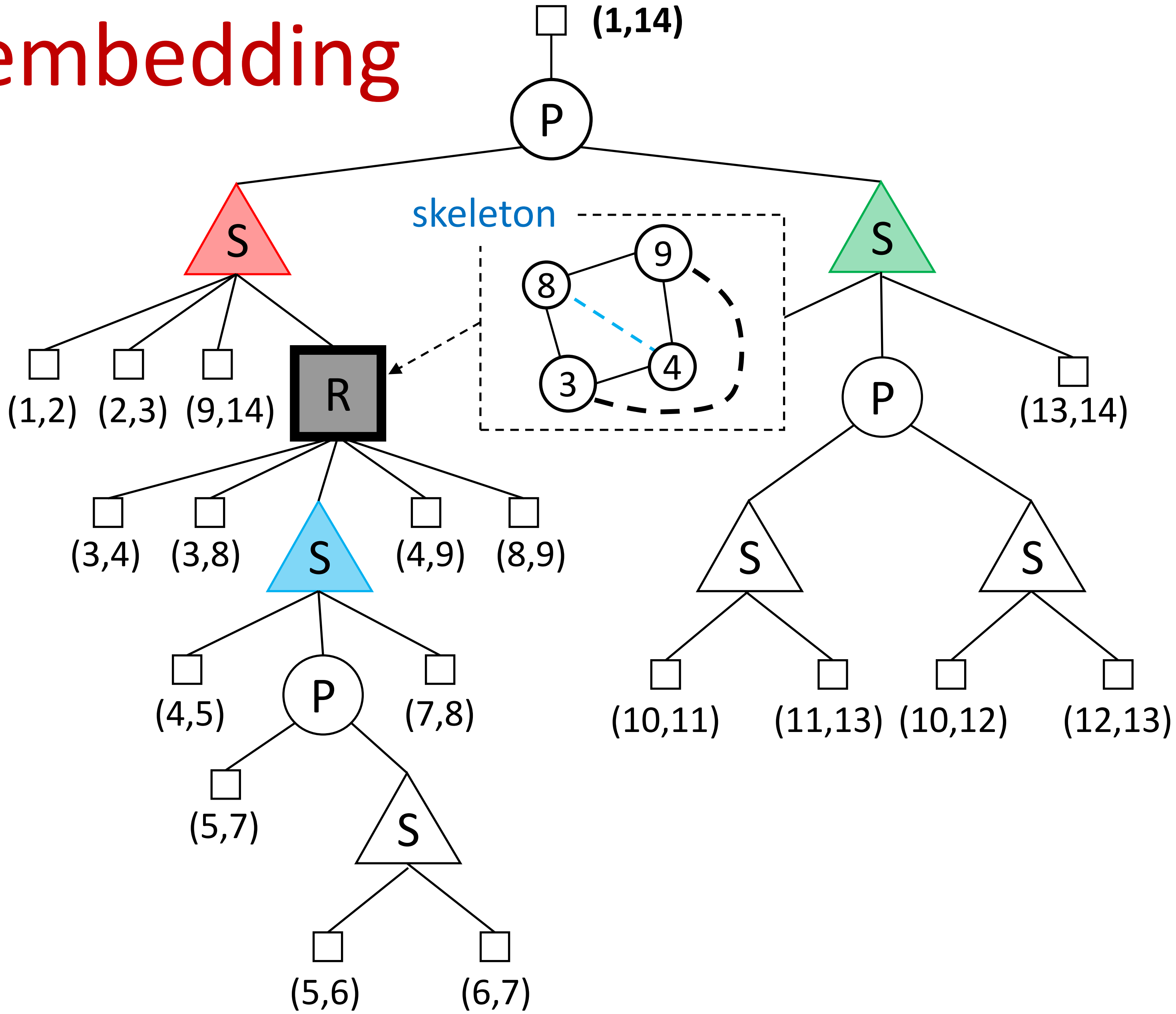




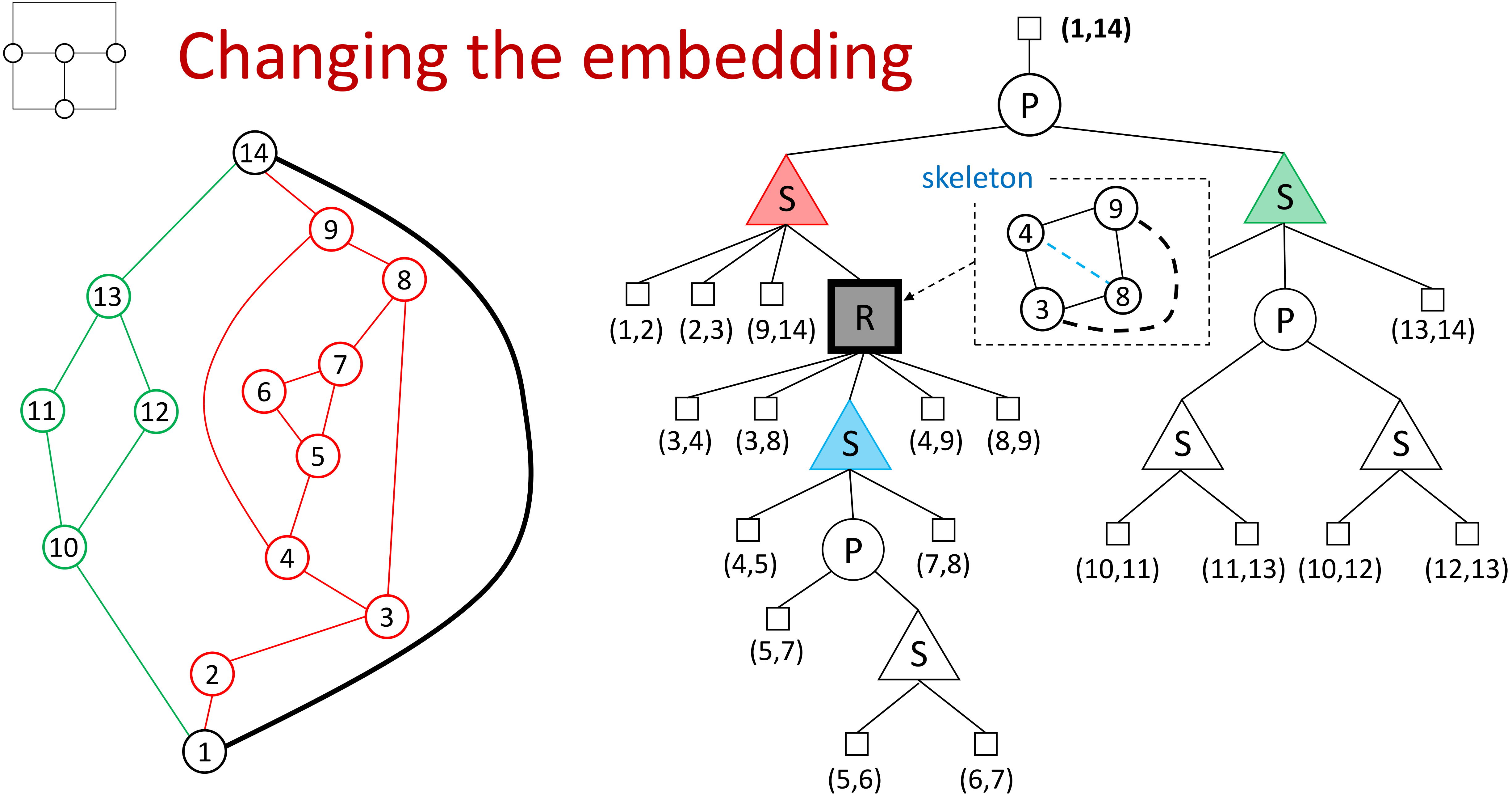


Changing the embedding

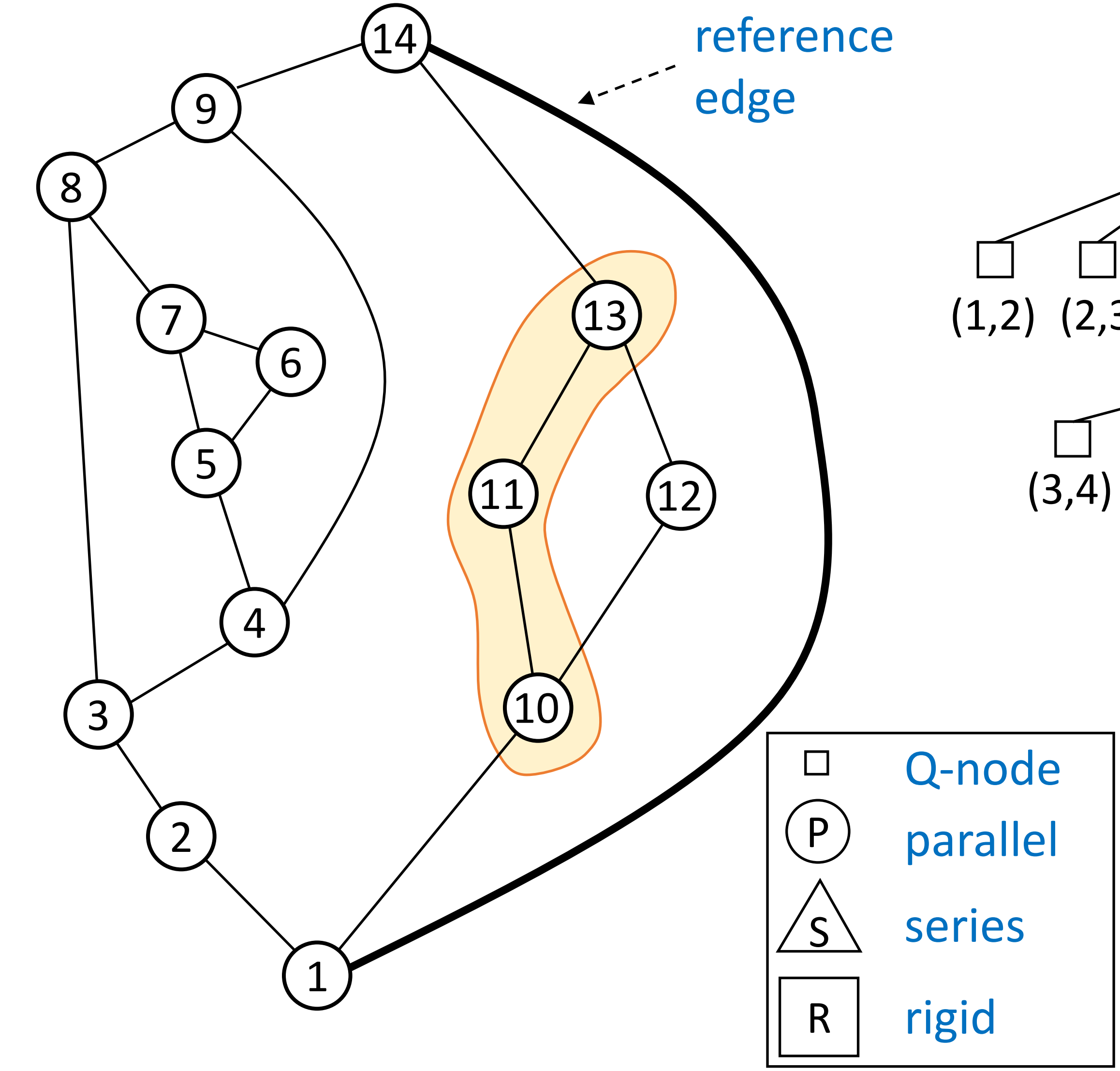




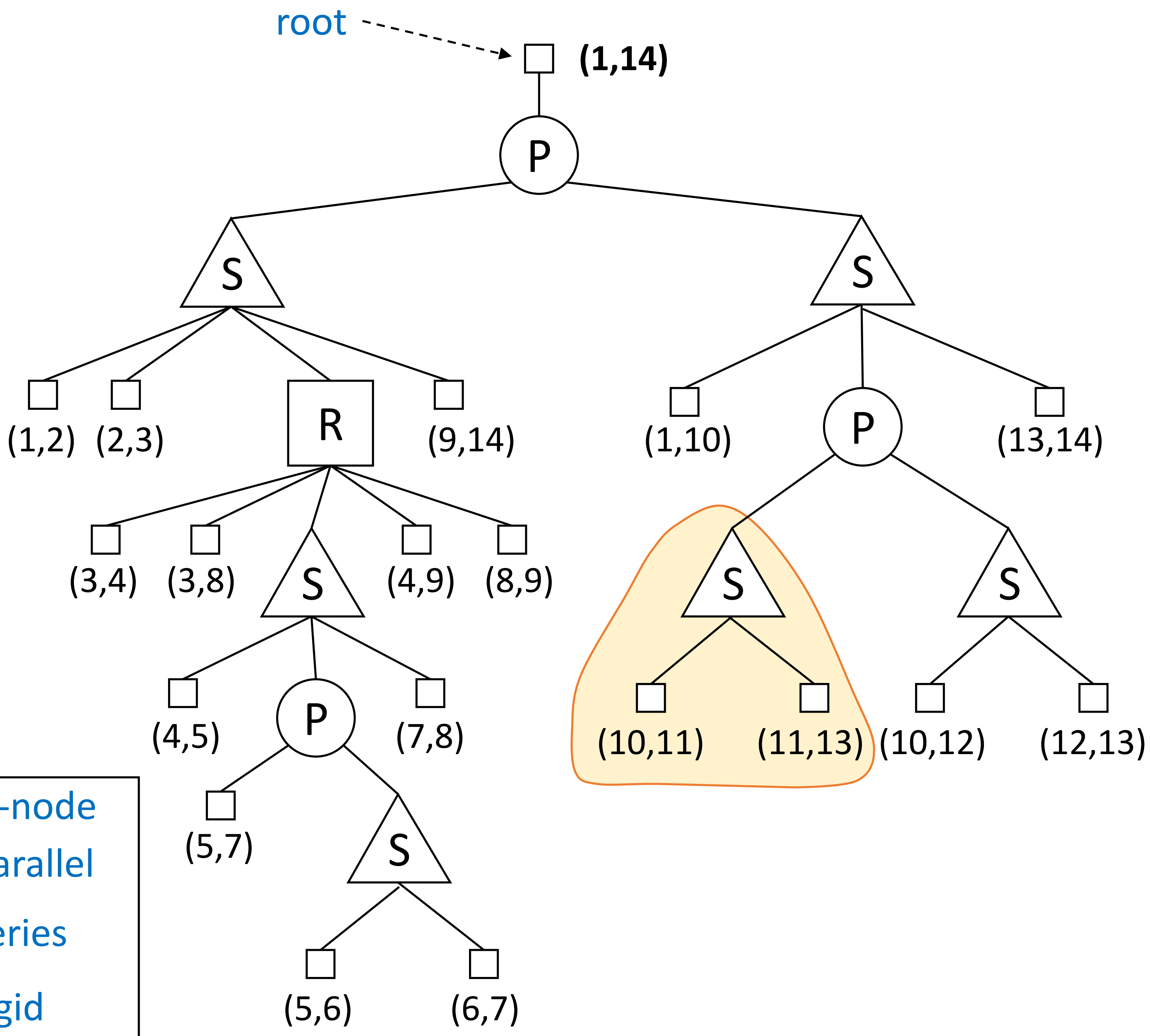
Changing the embedding



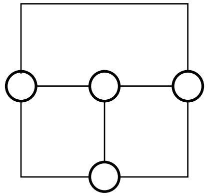
SPQ*R-trees



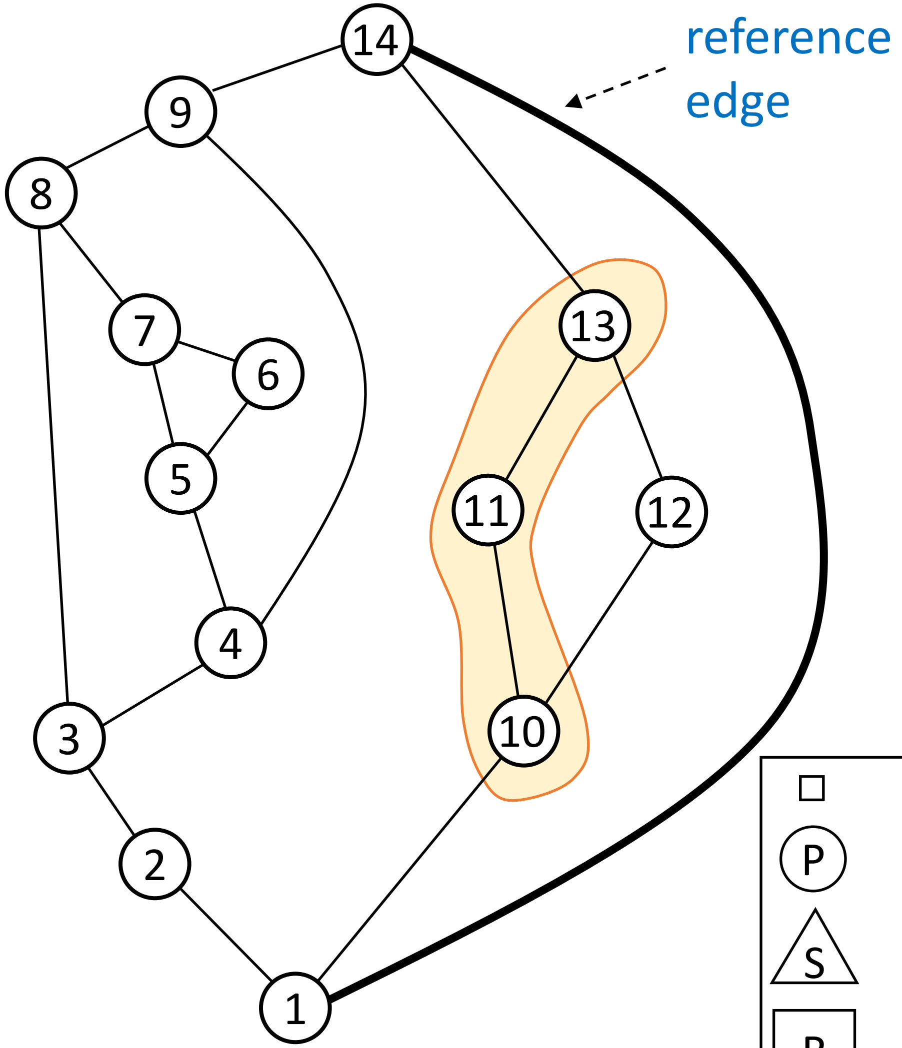
reference
edge
14
9
8
7
6
5
4
3
2
1
13
11
12
10
Q-node
P
parallel
S
series
R
rigid



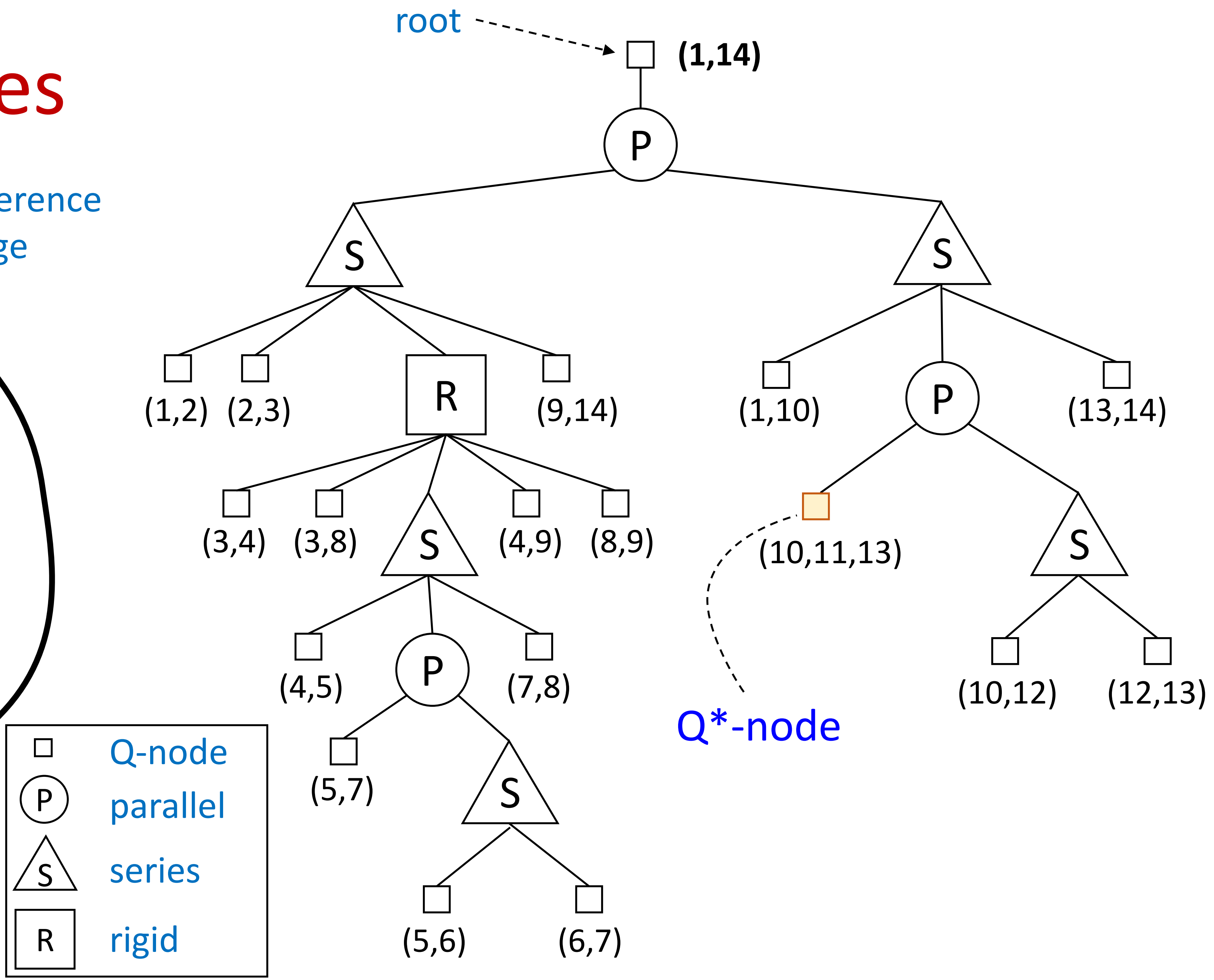
root
(1,14)
P
S
S
(1,2)
(2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
S
S
(4,5)
P
(7,8)
(10,11)
(11,13)
(10,12)
(12,13)
(5,7)
S
(5,6)
(6,7)



SPQ*R-trees

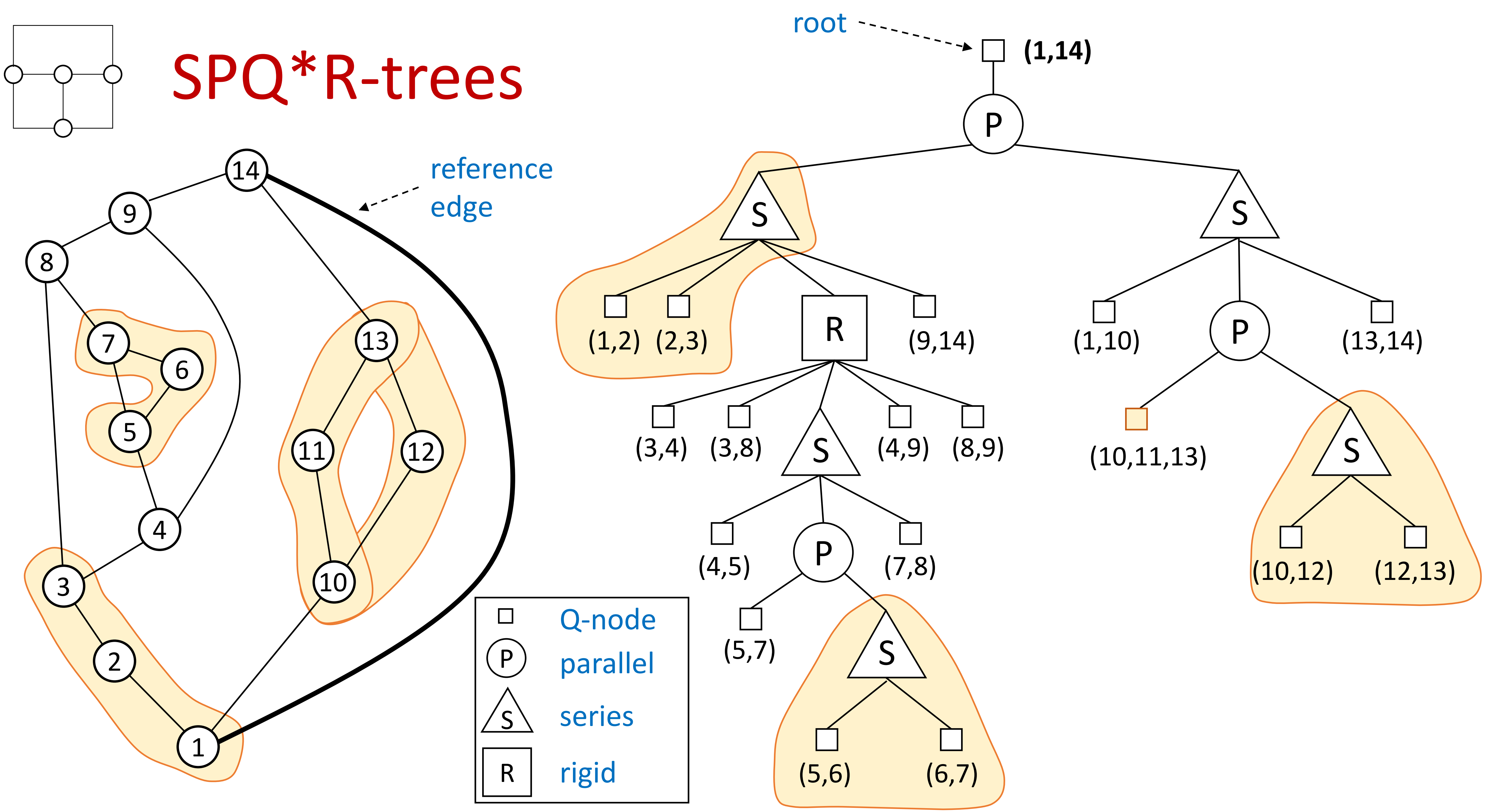


reference
edge
14
9
8
7
6
5
4
3
2
1
13
11
12
10



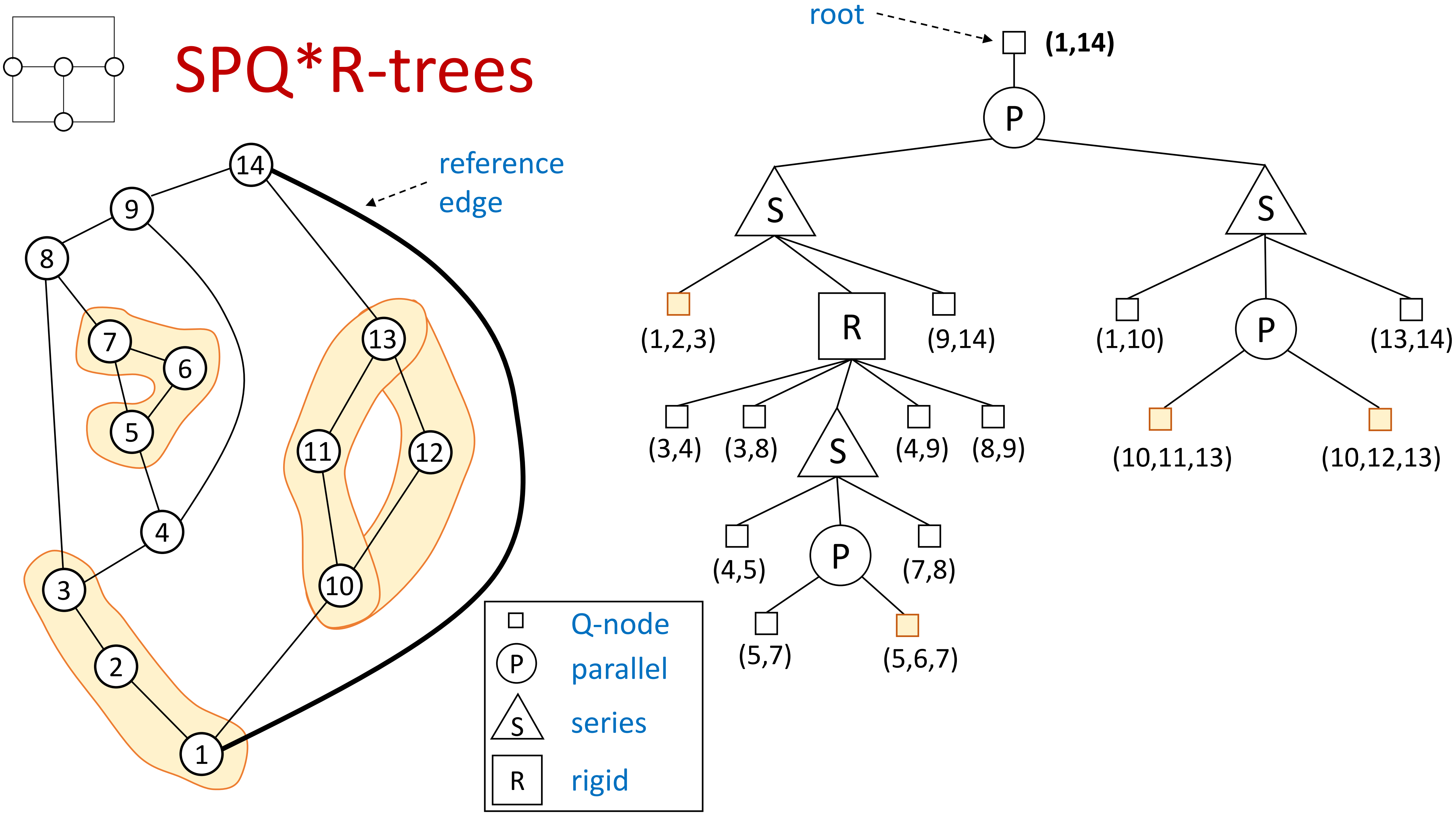
root
(1,14)
P
S
S
(1,2)
(2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
(10,11,13)
S
(4,5)
P
(7,8)
(10,12)
(12,13)
Q*-node
(5,7)
S
(5,6)
(6,7)
Q-node
P
parallel
S
series
R
rigid

SPQ*R-trees



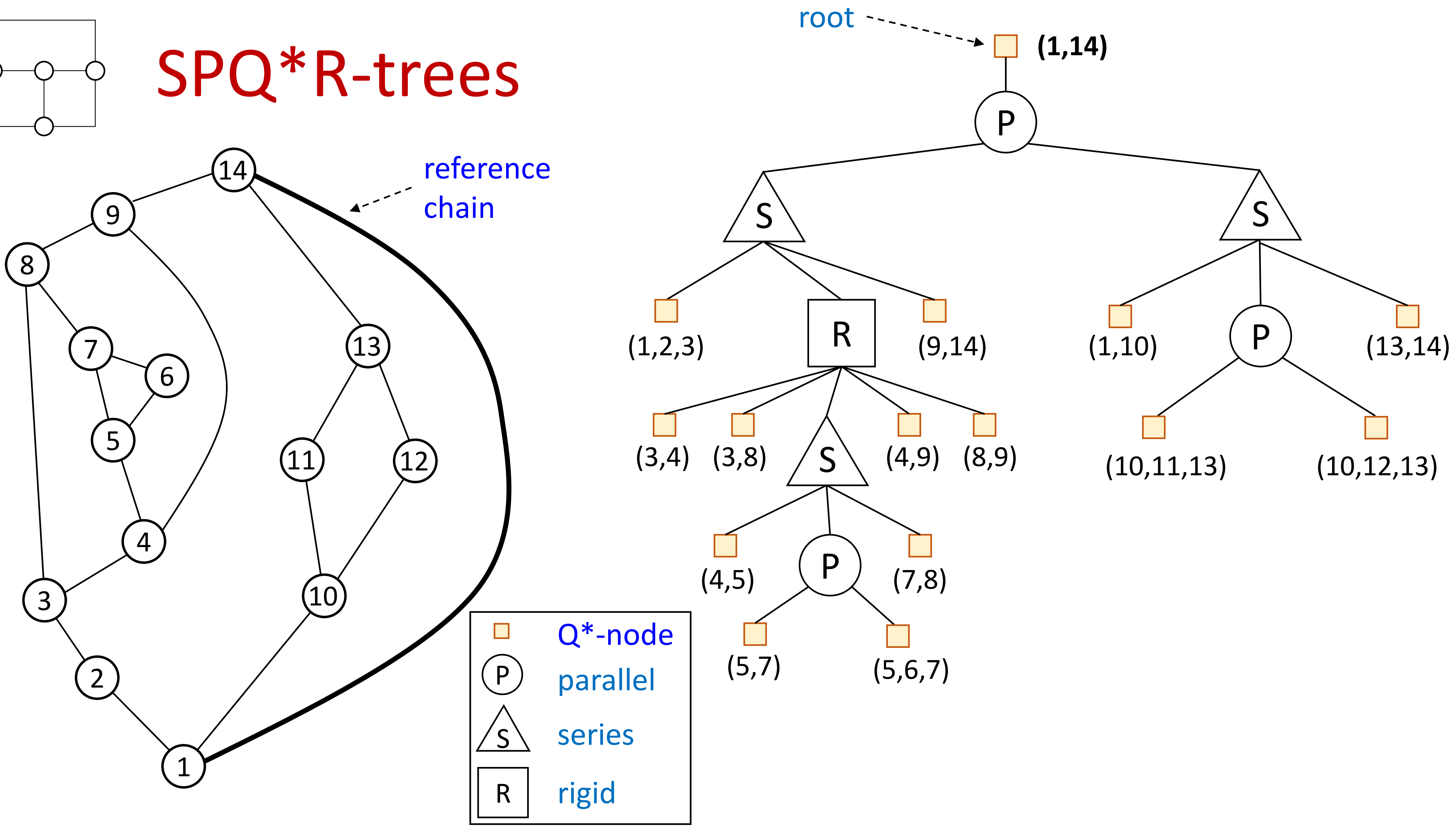
reference edge
root
(1,14)
P
S
S
(1,2)
(2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
(10,11,13)
S
(4,5)
P
(7,8)
(10,12)
(12,13)
(5,7)
S
(5,6)
(6,7)
Q-node
parallel
series
rigid

SPQ*R-trees



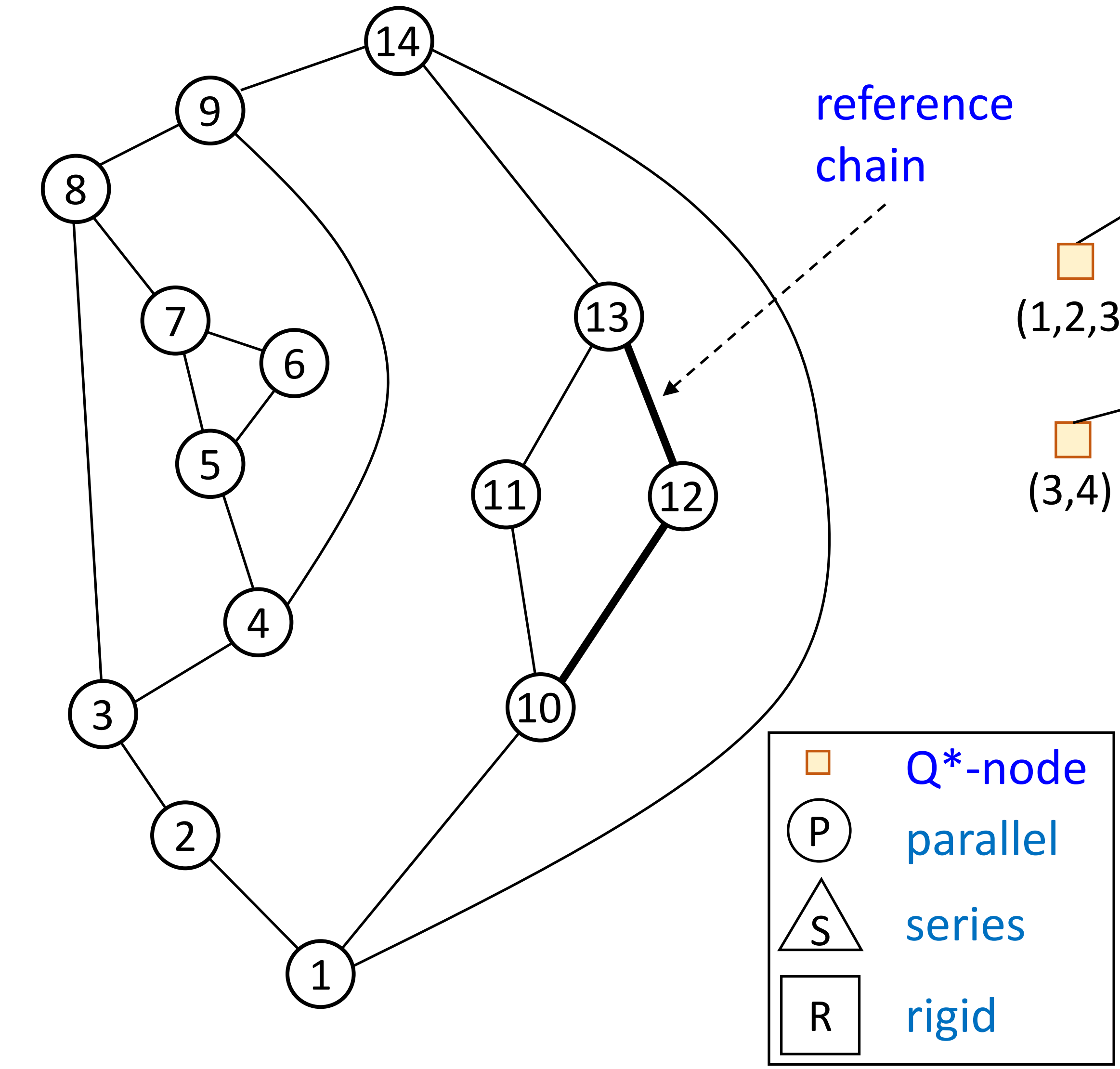
reference edge
14
9
8
7
6
5
4
3
2
1
13
11
12
10
root
(1,14)
P
S
S
(1,2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
(10,11,13)
(10,12,13)
(4,5)
P
(7,8)
(5,7)
(5,6,7)
Q-node
P parallel
S series
R rigid

SPQ*R-trees

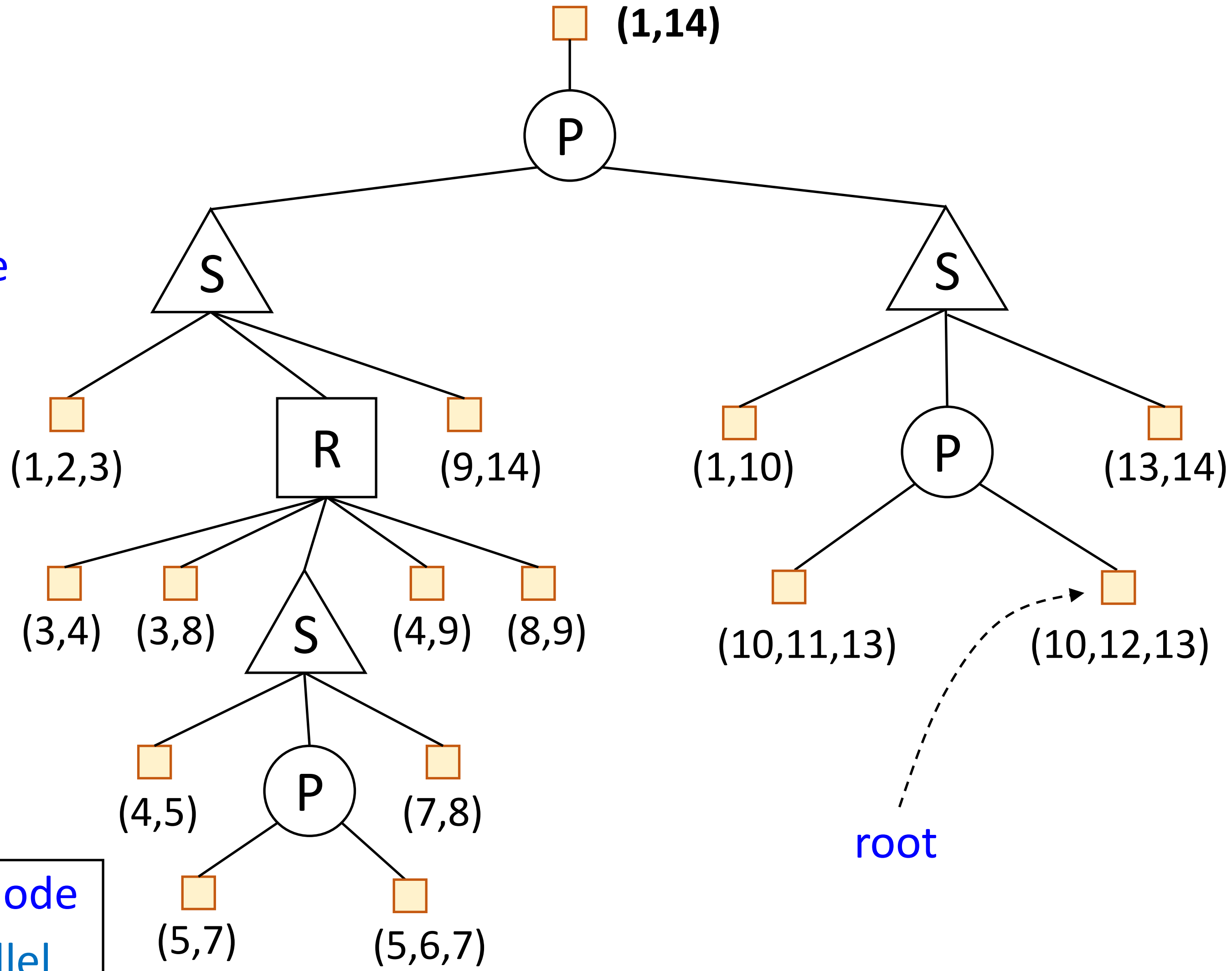


reference chain
14
9
8
7
6
5
4
3
2
1
13
11
12
10
root
(1,14)
P
S
S
(1,2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
(10,11,13)
(10,12,13)
(4,5)
P
(7,8)
(5,7)
(5,6,7)
Q*-node
P parallel
S series
R rigid

SPQ*R-trees

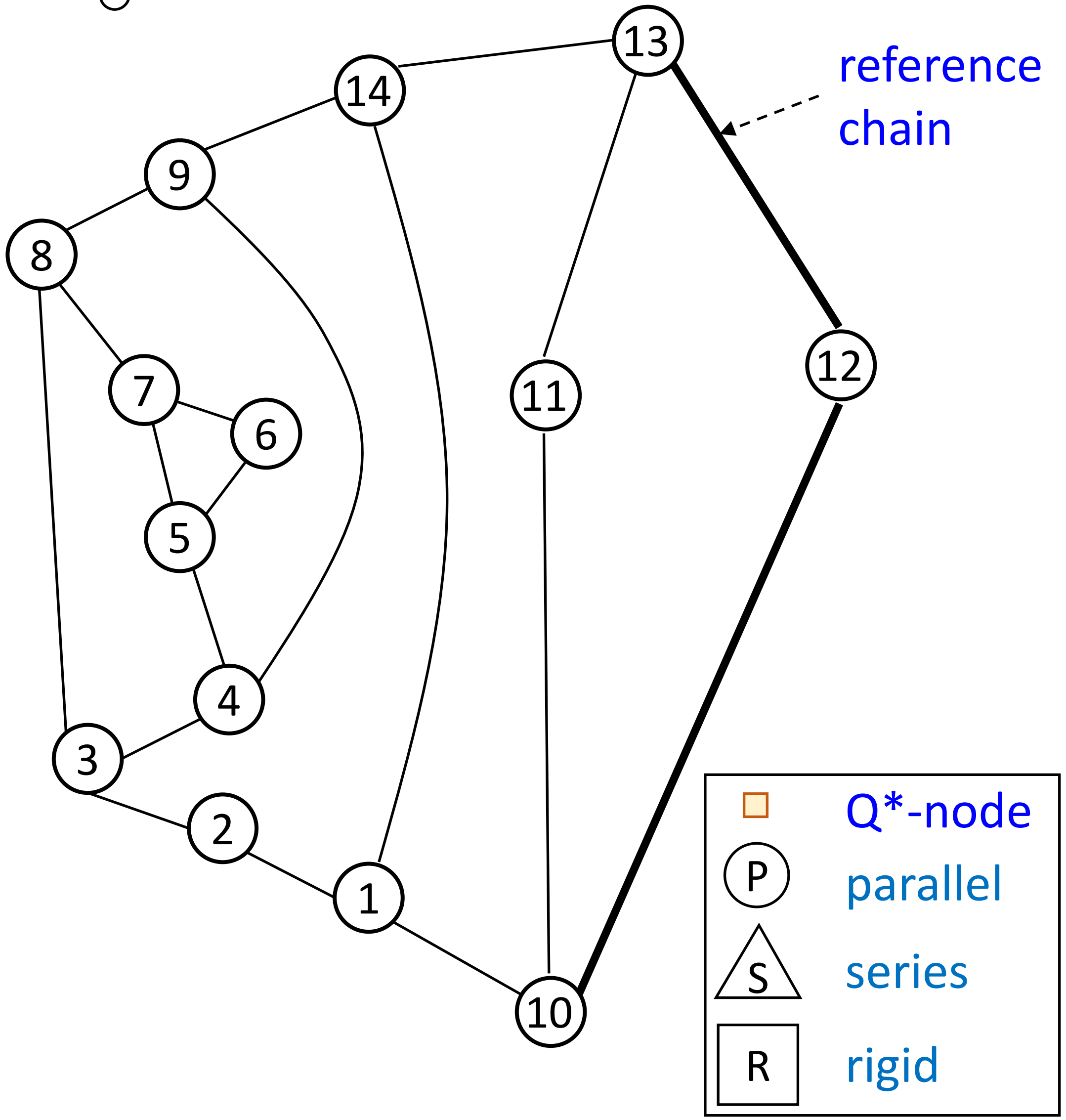


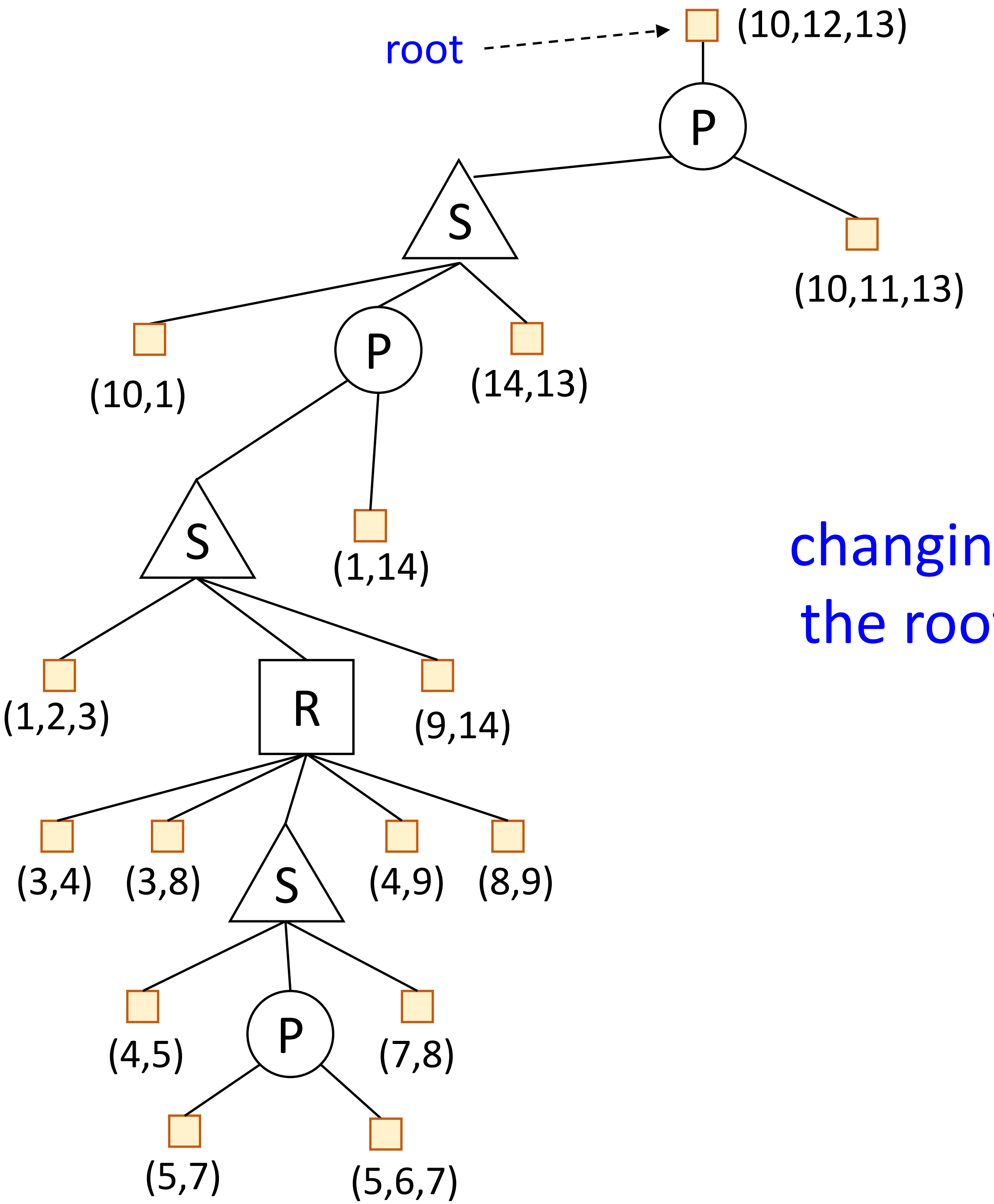
14
9
8
7
6
5
4
3
2
1
13
11
12
10
reference chain
Q*-node
P parallel
S series
R rigid



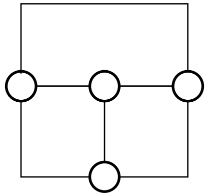
(1,14)
P
S
S
(1,2,3)
R
(9,14)
(1,10)
P
(13,14)
(3,4)
(3,8)
S
(4,9)
(8,9)
(10,11,13)
(10,12,13)
(4,5)
P
(7,8)
(5,7)
(5,6,7)
root

SPQ*R-trees

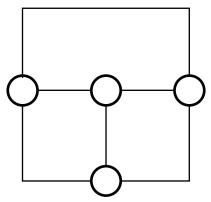




changing
the root

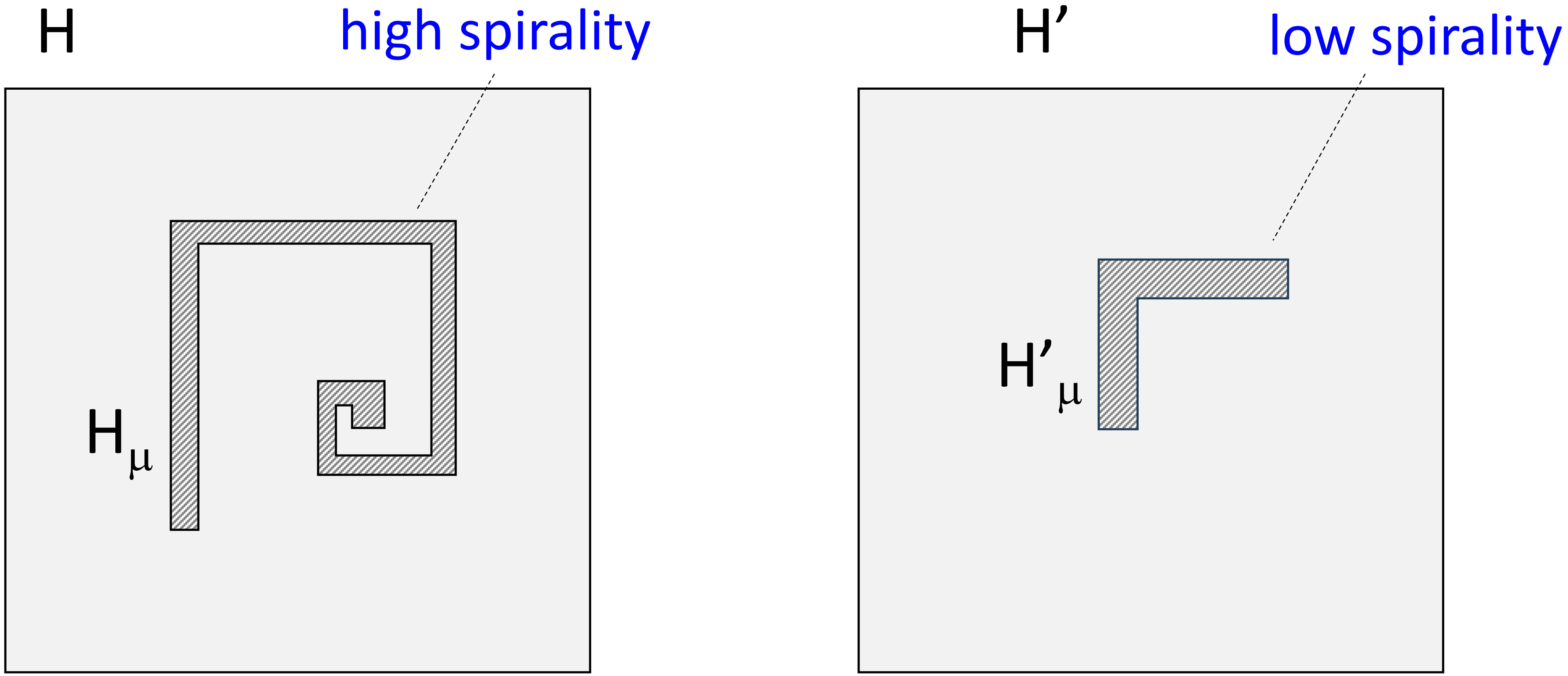


\end{SPQR-trees}

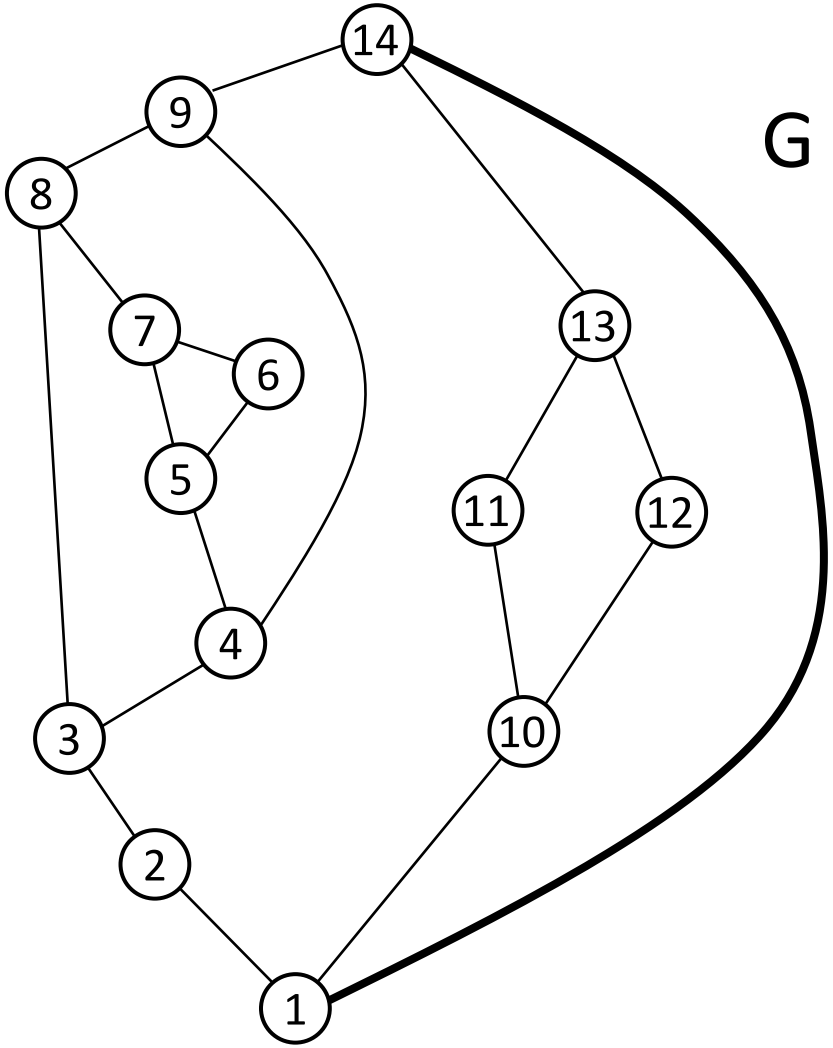


\begin{Spirality}

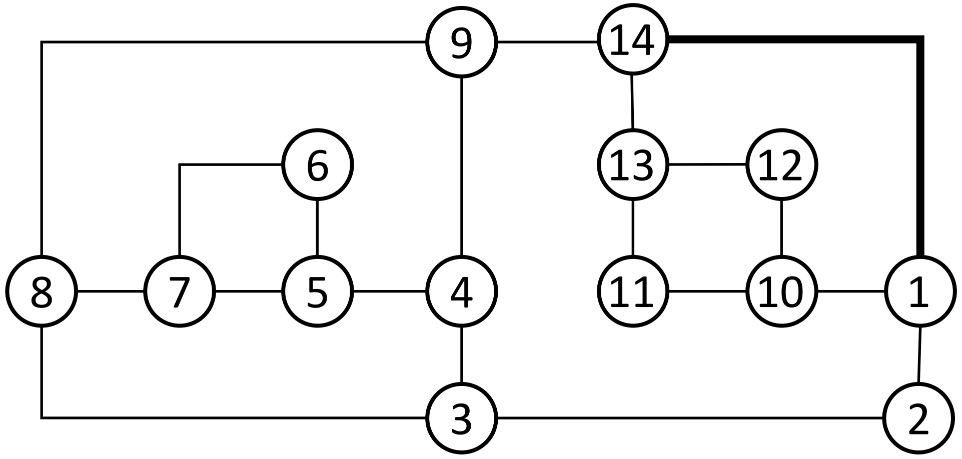
Spirality of orthogonal components: Intuition



Spirality of orthogonal components

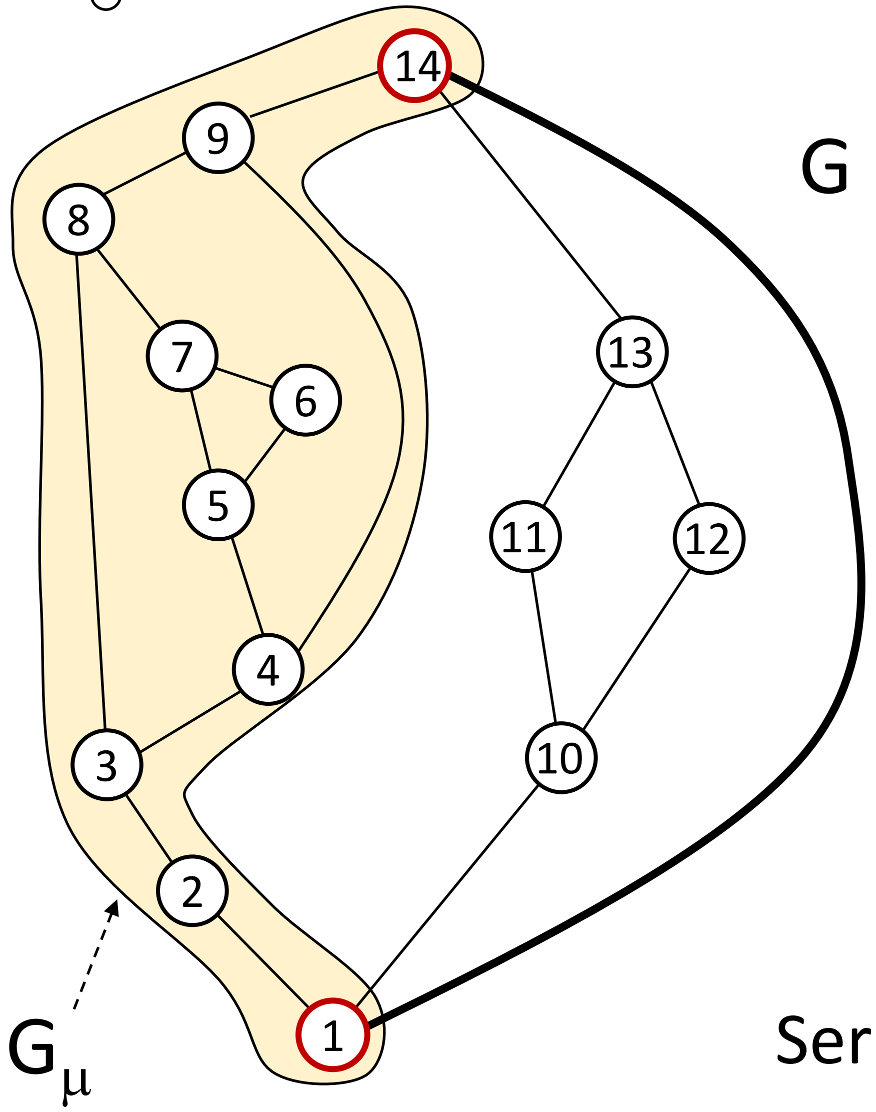


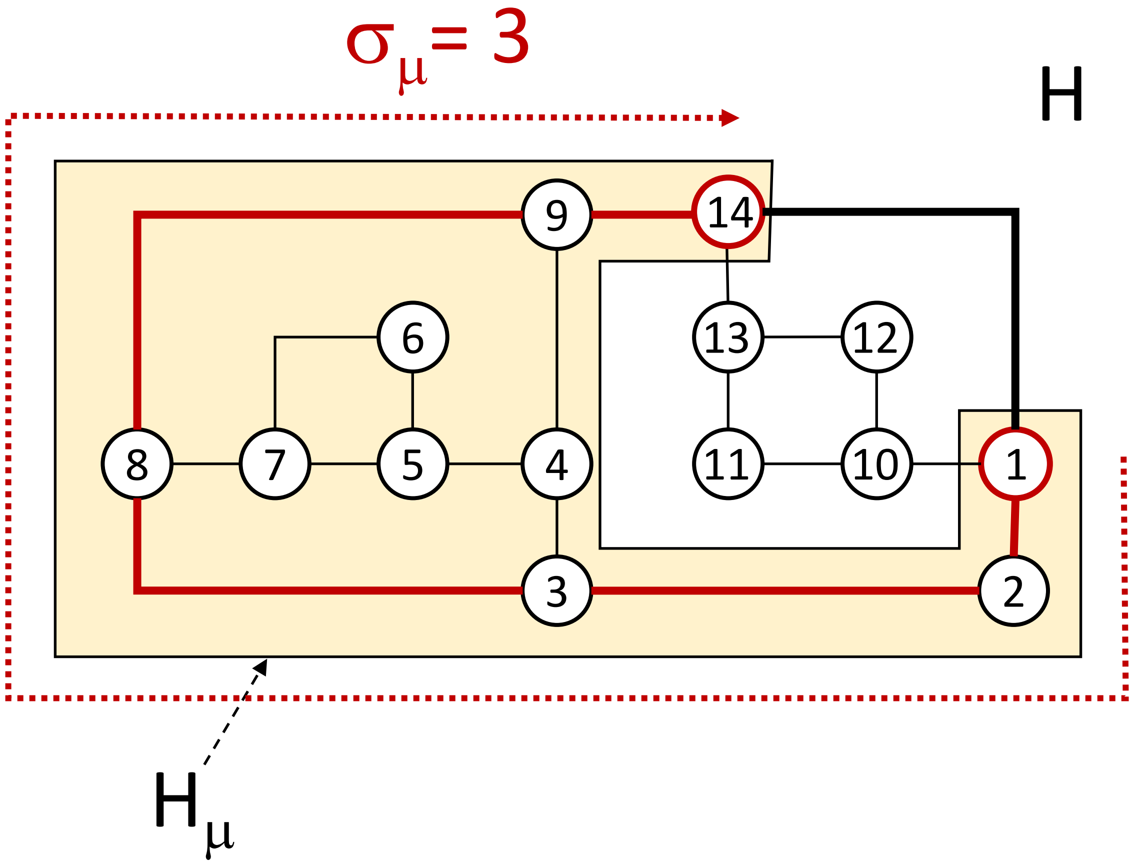
G



H

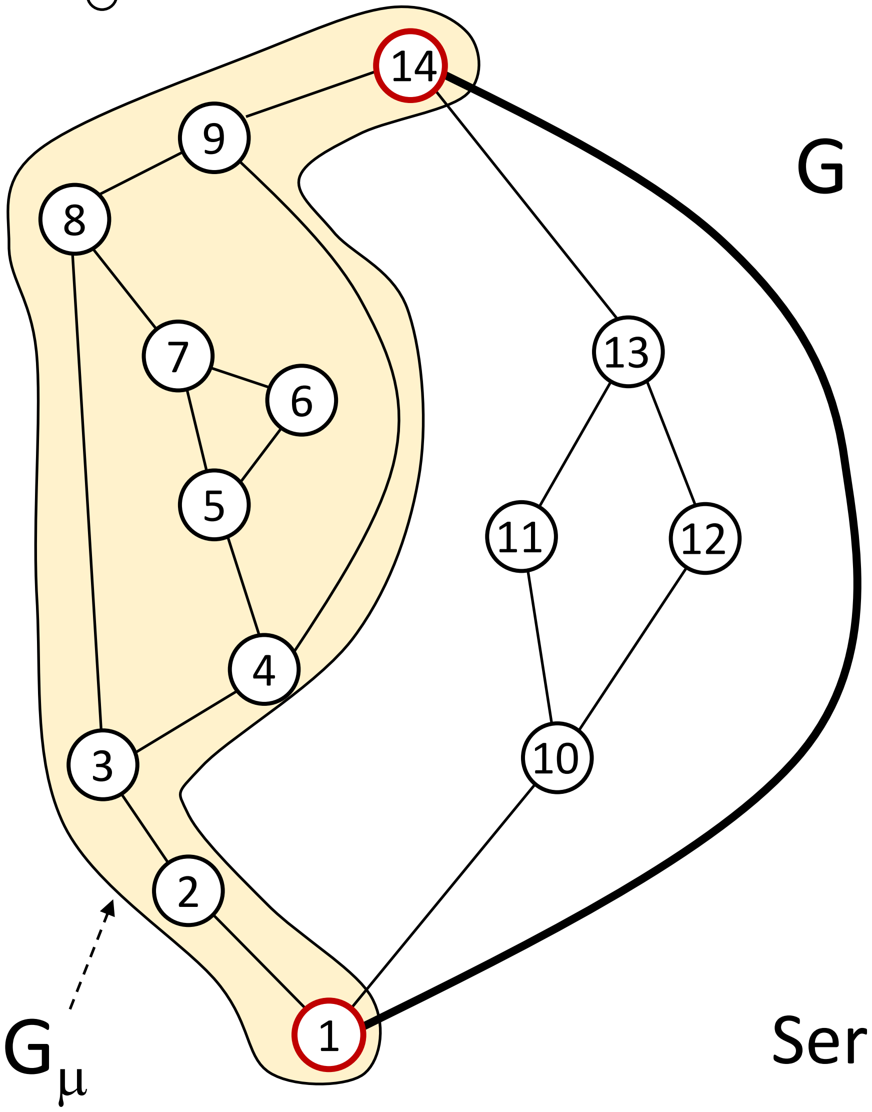
Spirality of orthogonal components

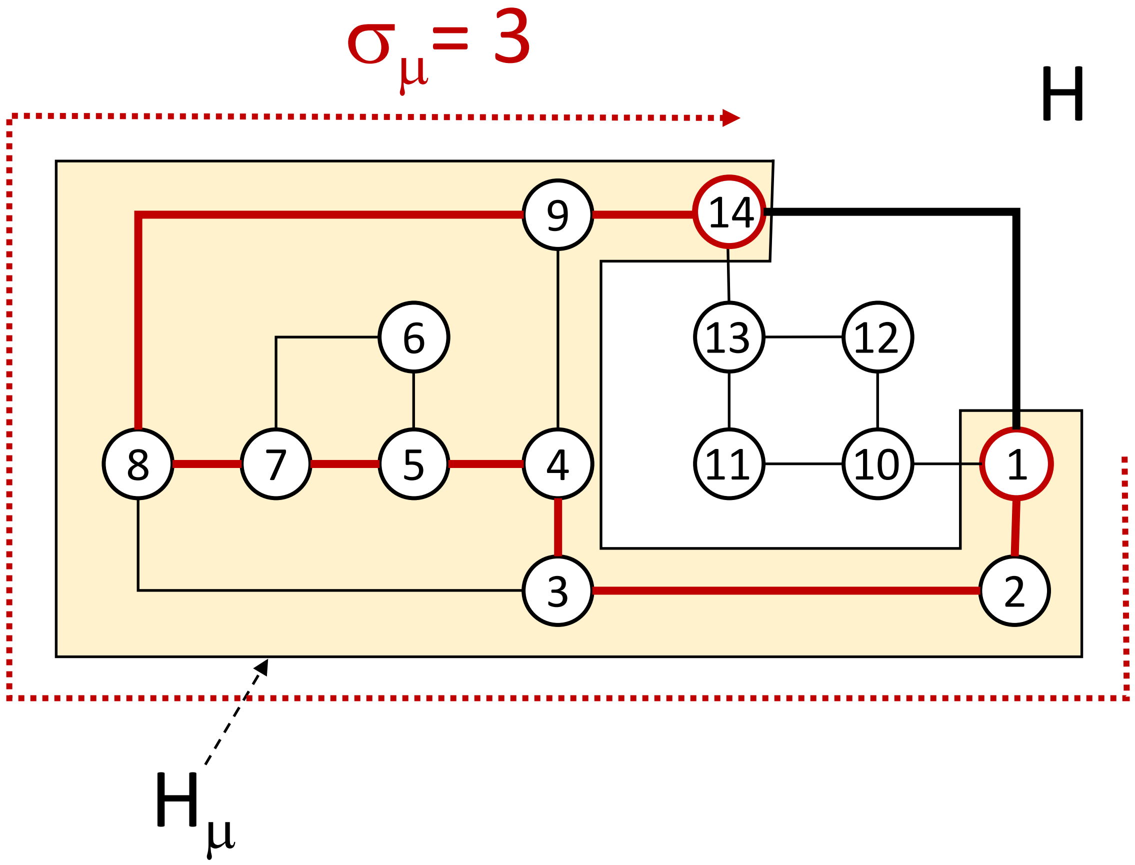




Series (orthogonal) component

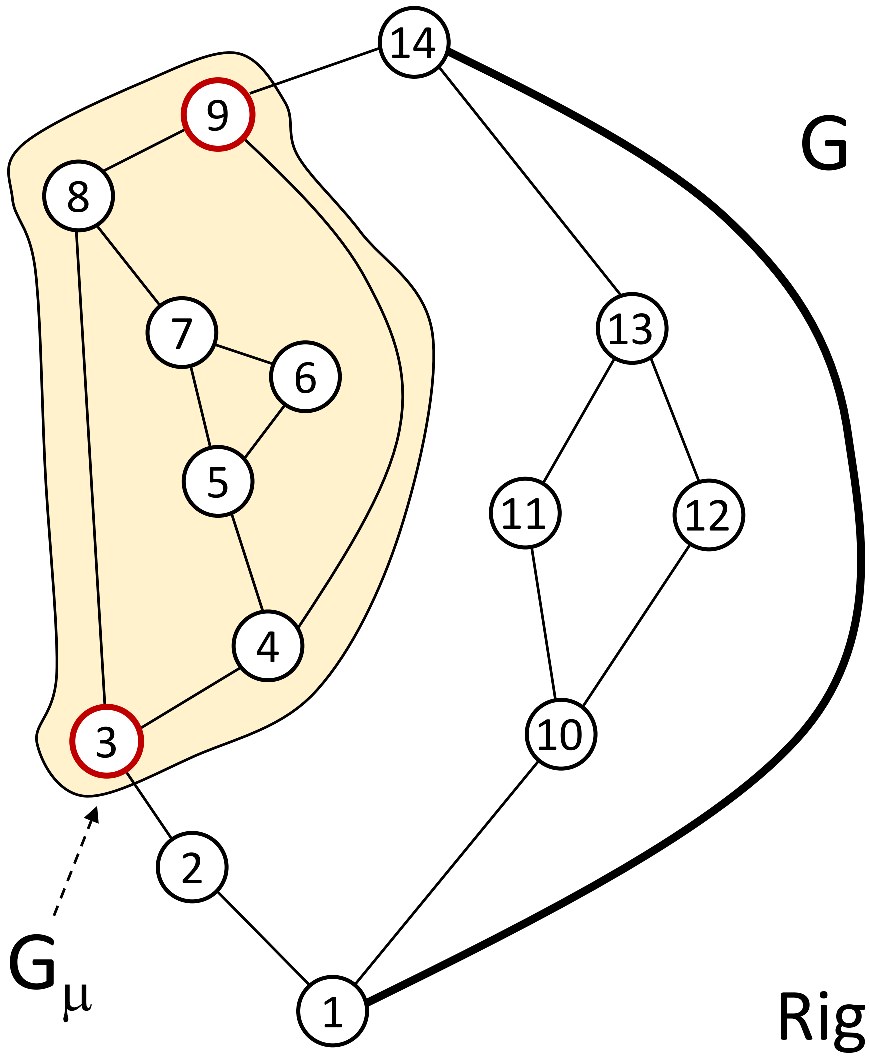
Spirality of orthogonal components

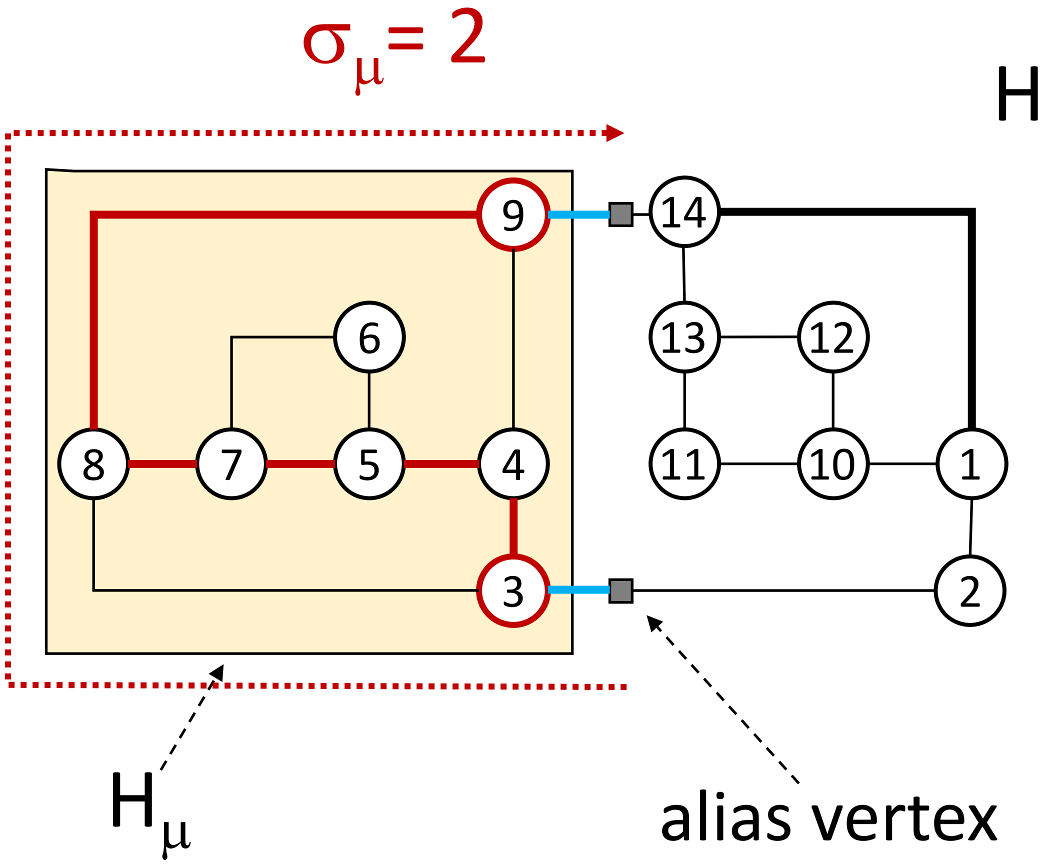




Series (orthogonal) component

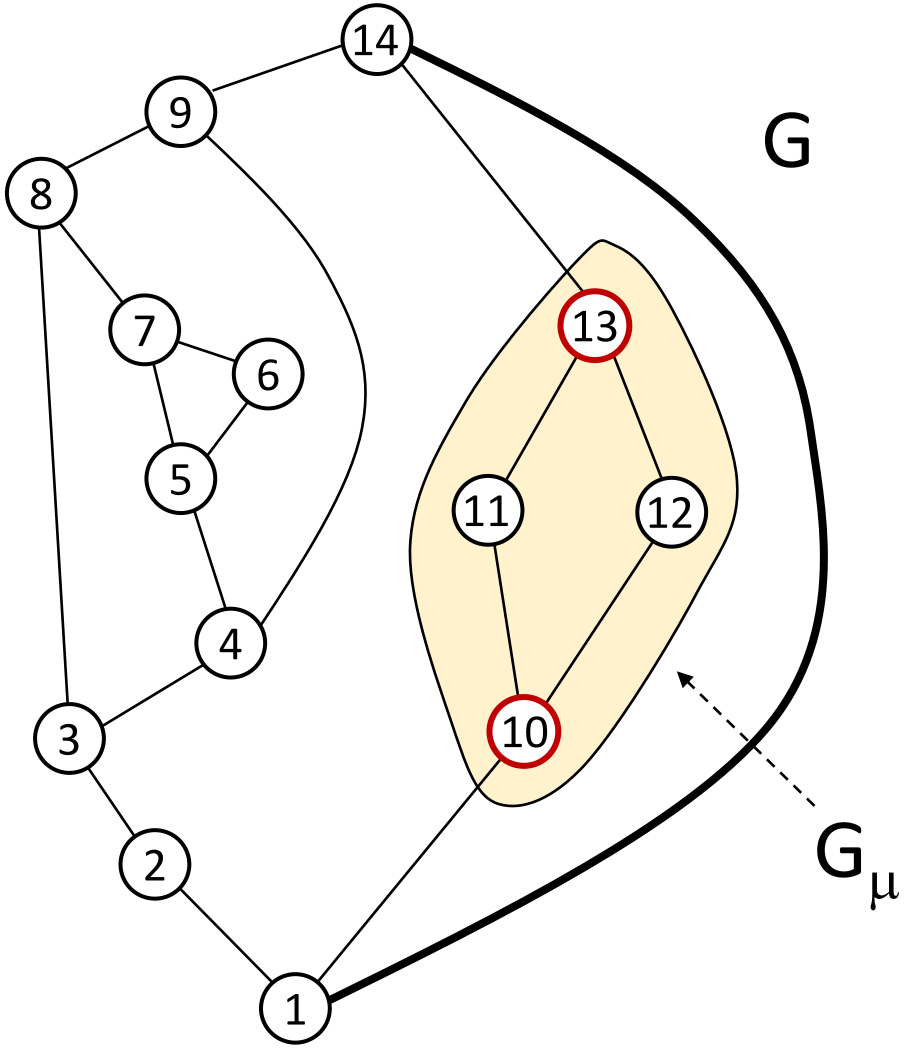
Spirality of orthogonal components

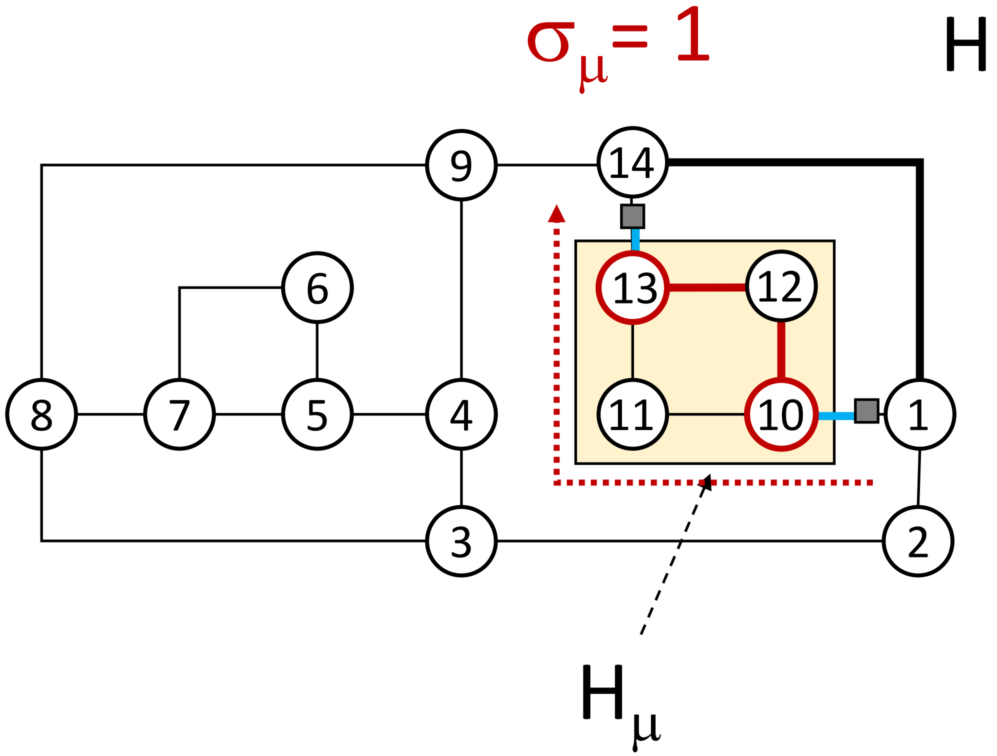




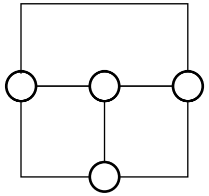
Rigid (orthogonal) component

Spirality of orthogonal components



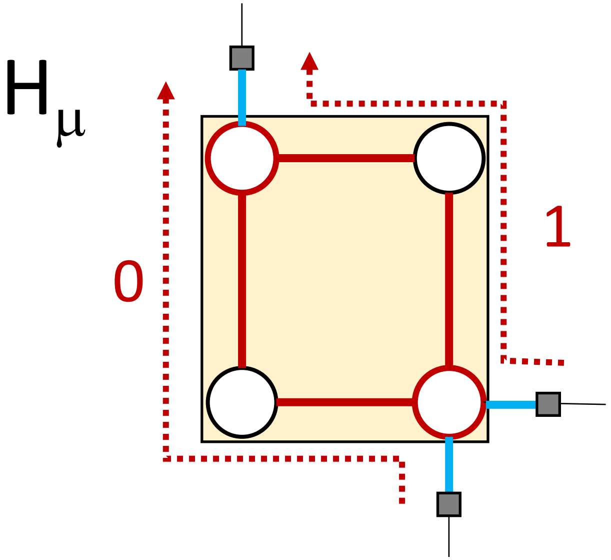


Parallel (orthogonal) component

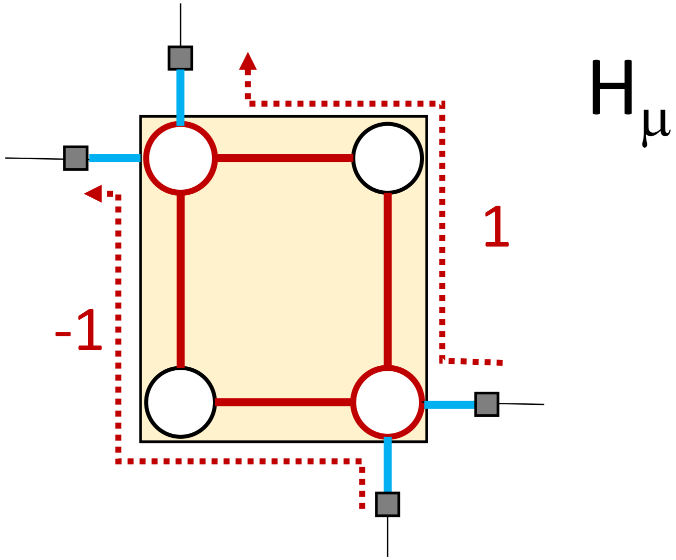


Spirality: More cases

$\sigma_\mu = 1/2$



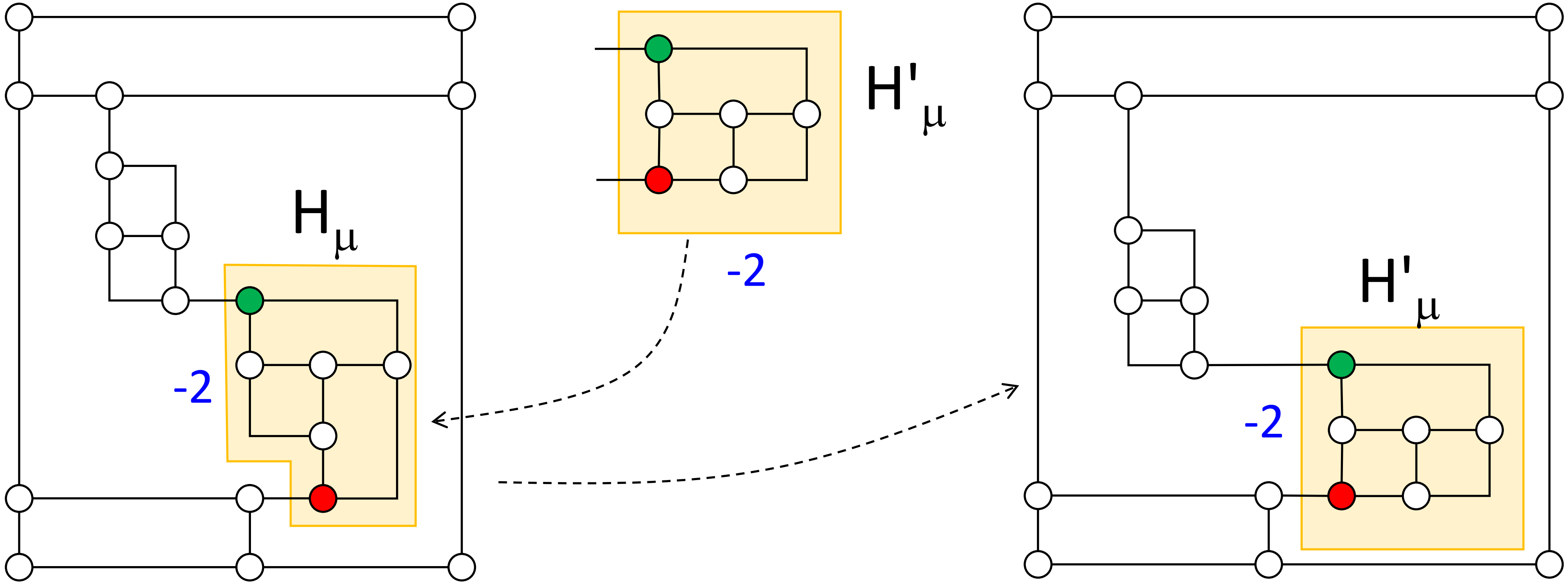
$\sigma_\mu = 0$



The spirality is either an integer or a semi-integer number

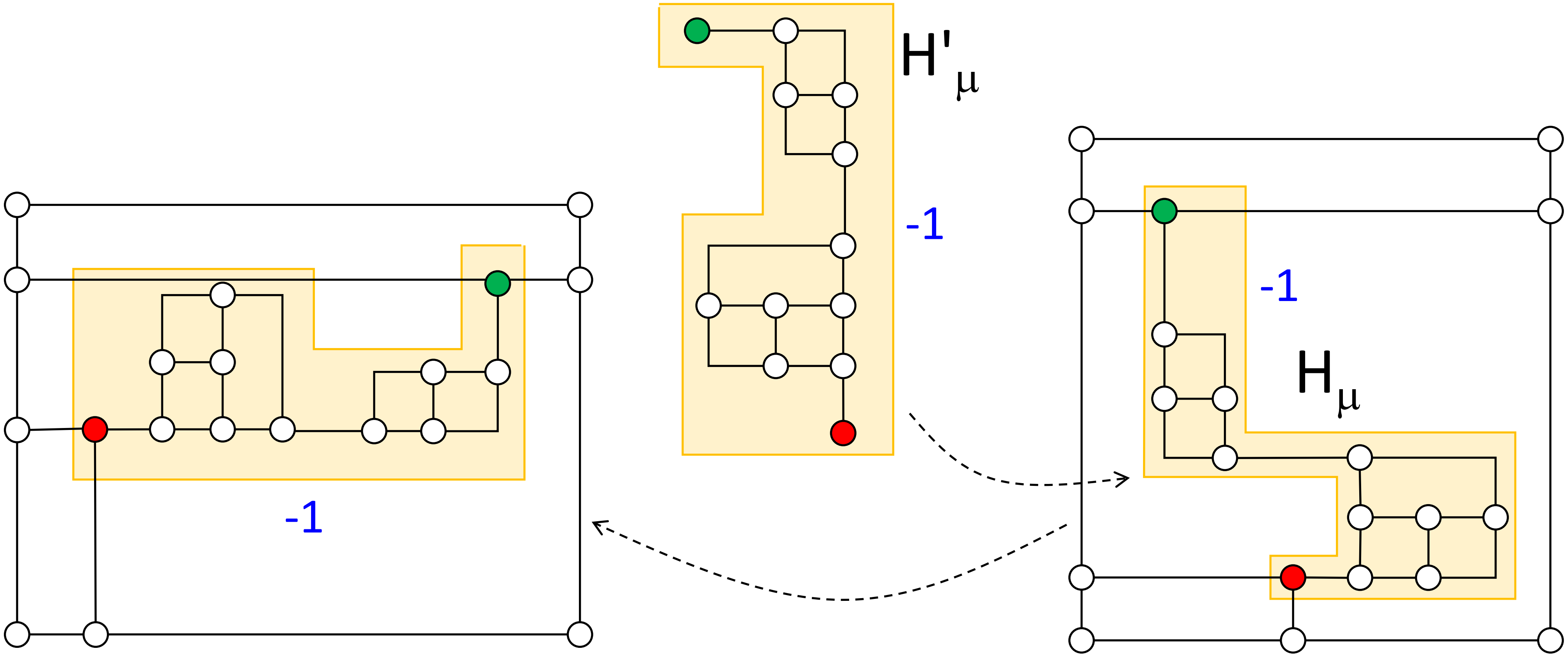
Substitution of components

Theorem (substitution). Two orthogonal components with the same spirality are "interchangeable" (even if they have different embeddings)

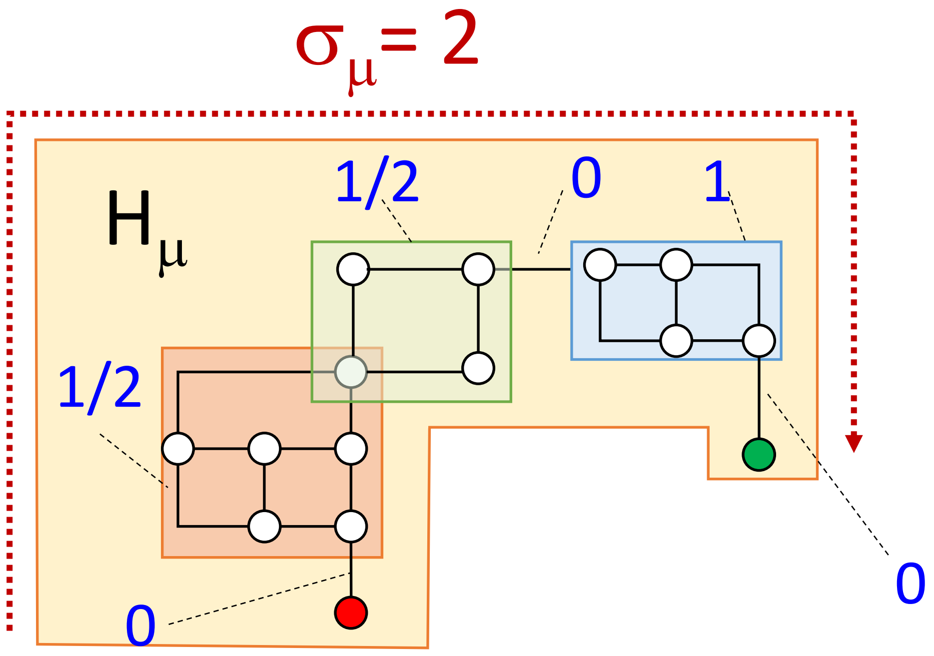


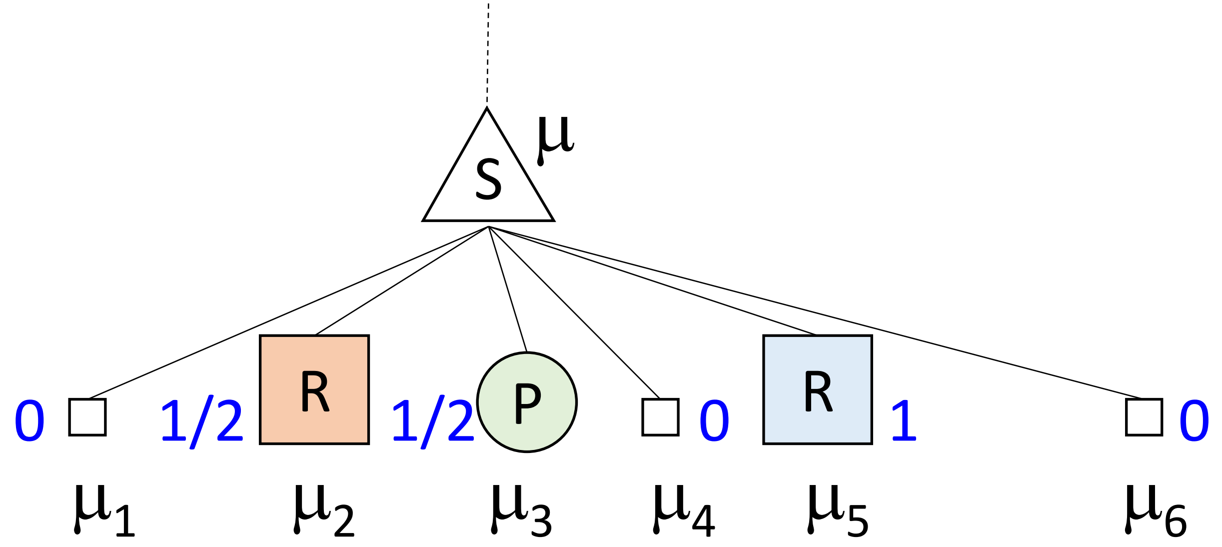
Substitution of components

Theorem (substitution). Two orthogonal components with the same spirality are “interchangeable” (even if they have different embeddings)

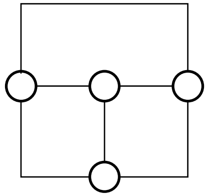


Spirality: Series relationship

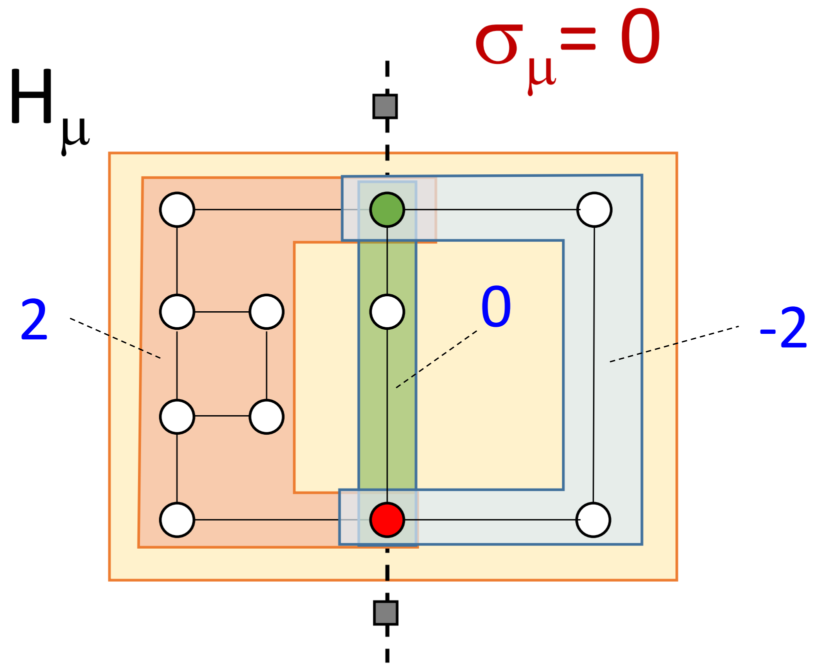


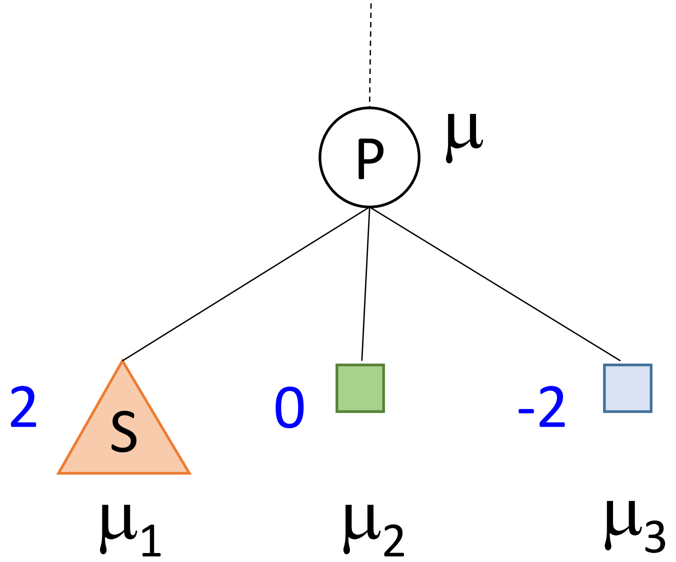


$$\sigma_\mu = \sum_{i=1}^{6} \sigma_{\mu i} = 2$$

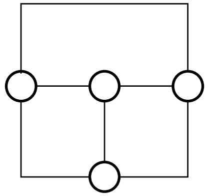


Spirality: Parallel relationship

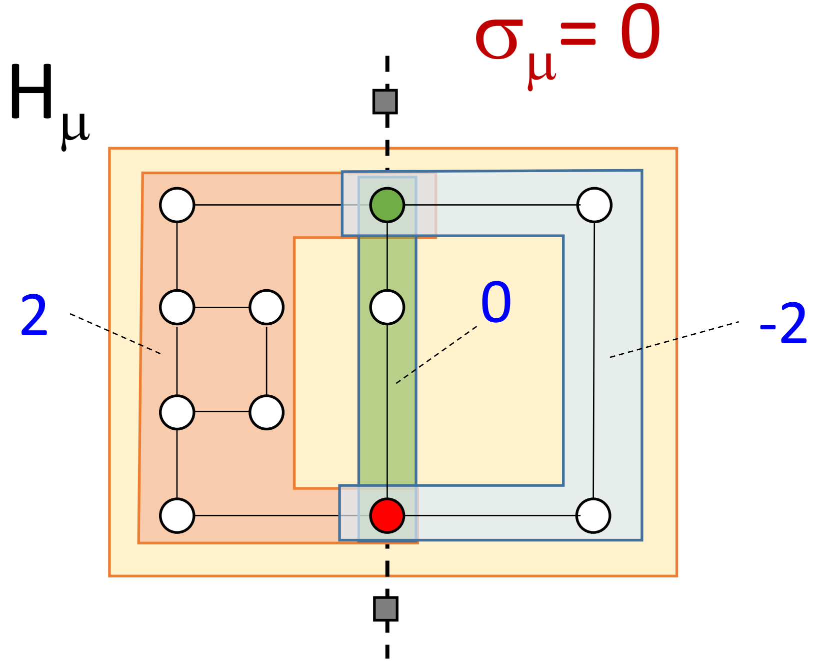


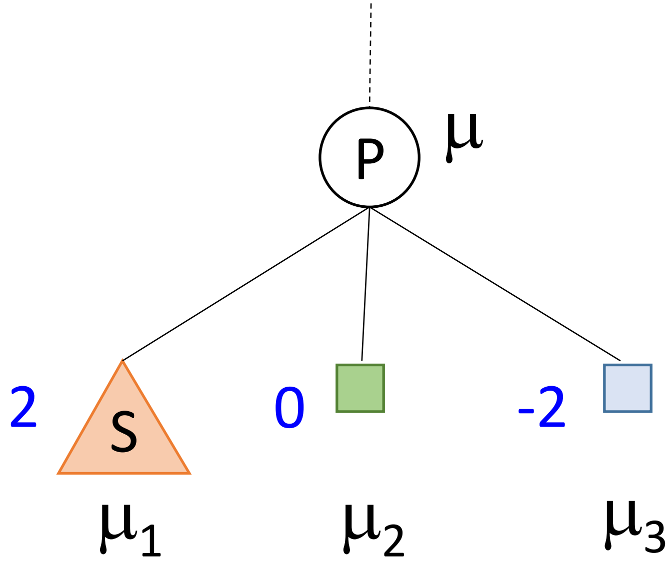


$$\sigma_\mu = \sigma_{\mu 1} - 2 = \sigma_{\mu 2} = \sigma_{\mu 3} + 2$$



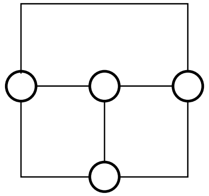
Spirality: Parallel relationship





- For P-nodes with two children the relationship is a bit more involved, but similar

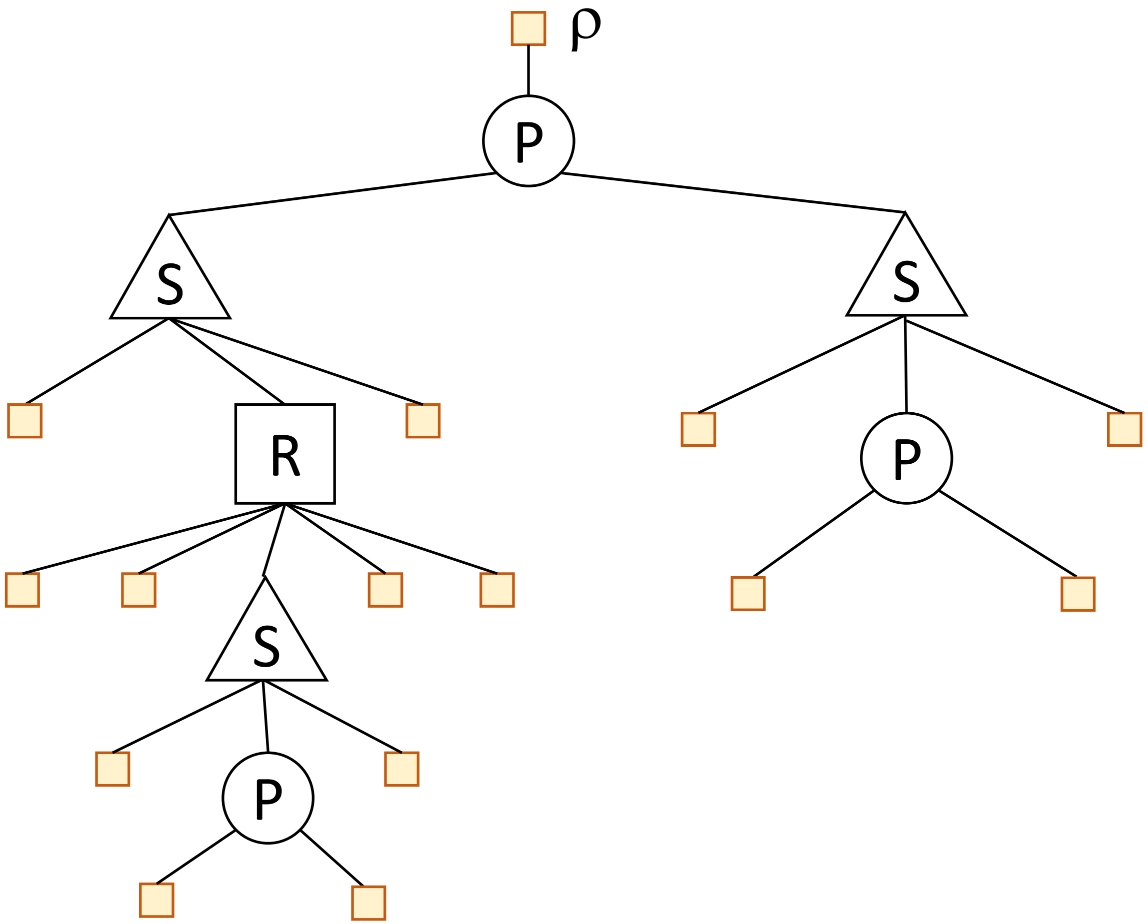
$$\sigma_\mu = \sigma_{\mu 1} - 2 = \sigma_{\mu 2} = \sigma_{\mu 3} + 2$$



\end{Spirality}

Bend-min algorithm: General strategy

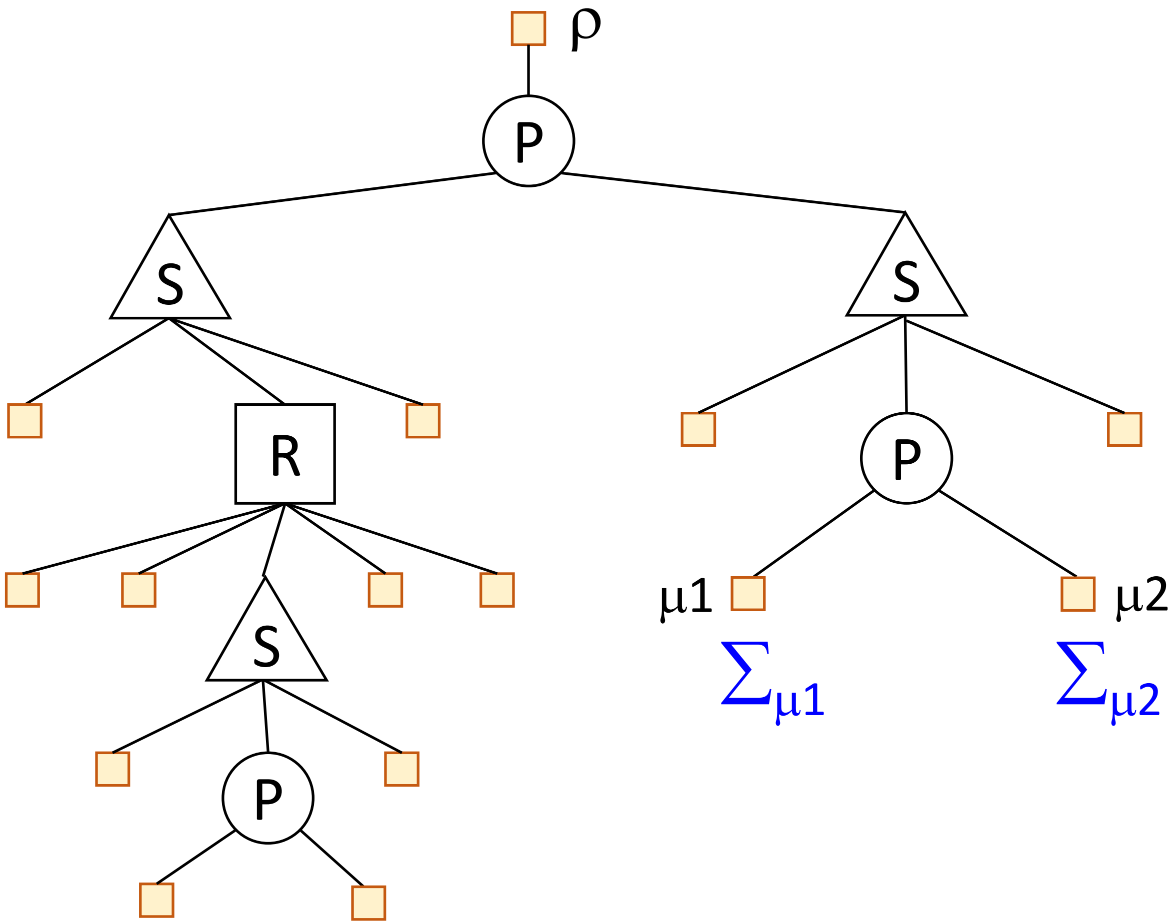
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{\langle\sigma_\mu, b(\sigma_\mu)\rangle \mid$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

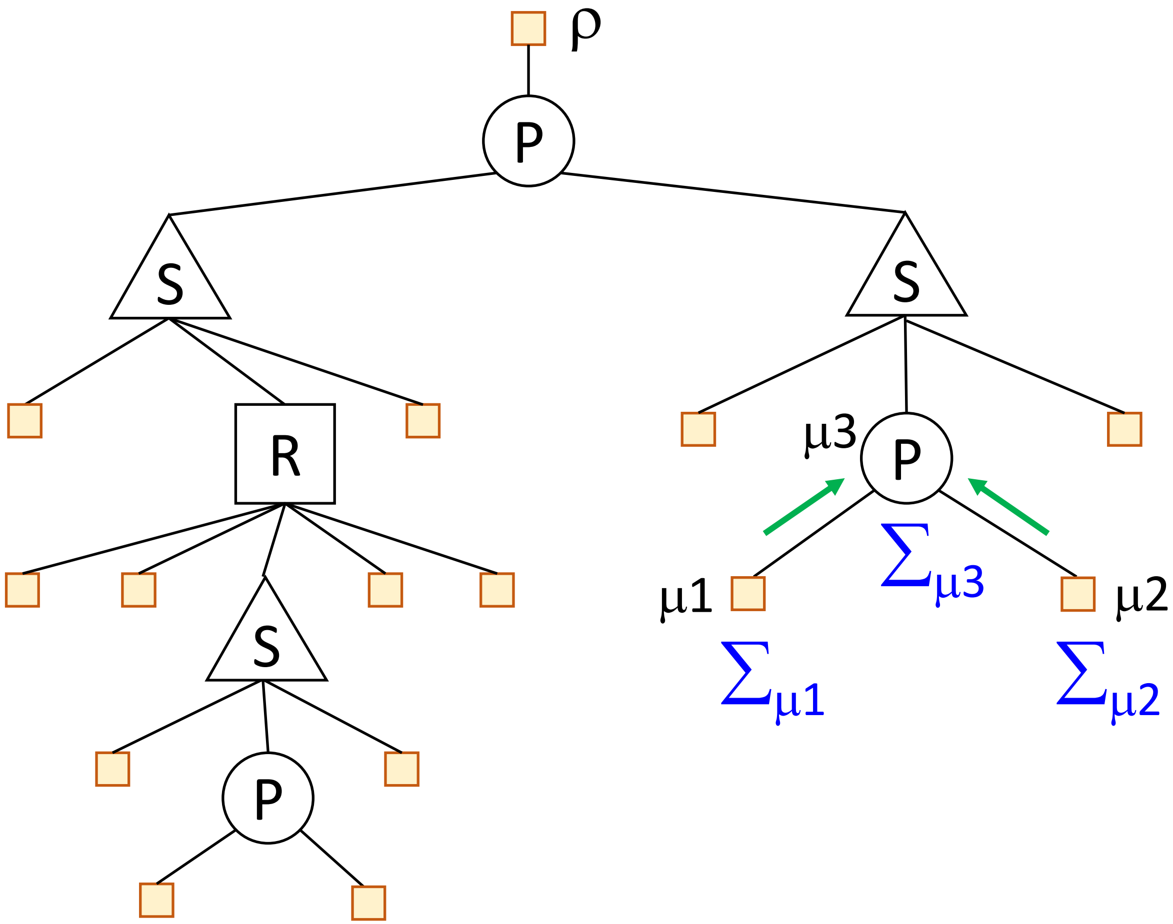
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{<\sigma_\mu, b(\sigma_\mu)> |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

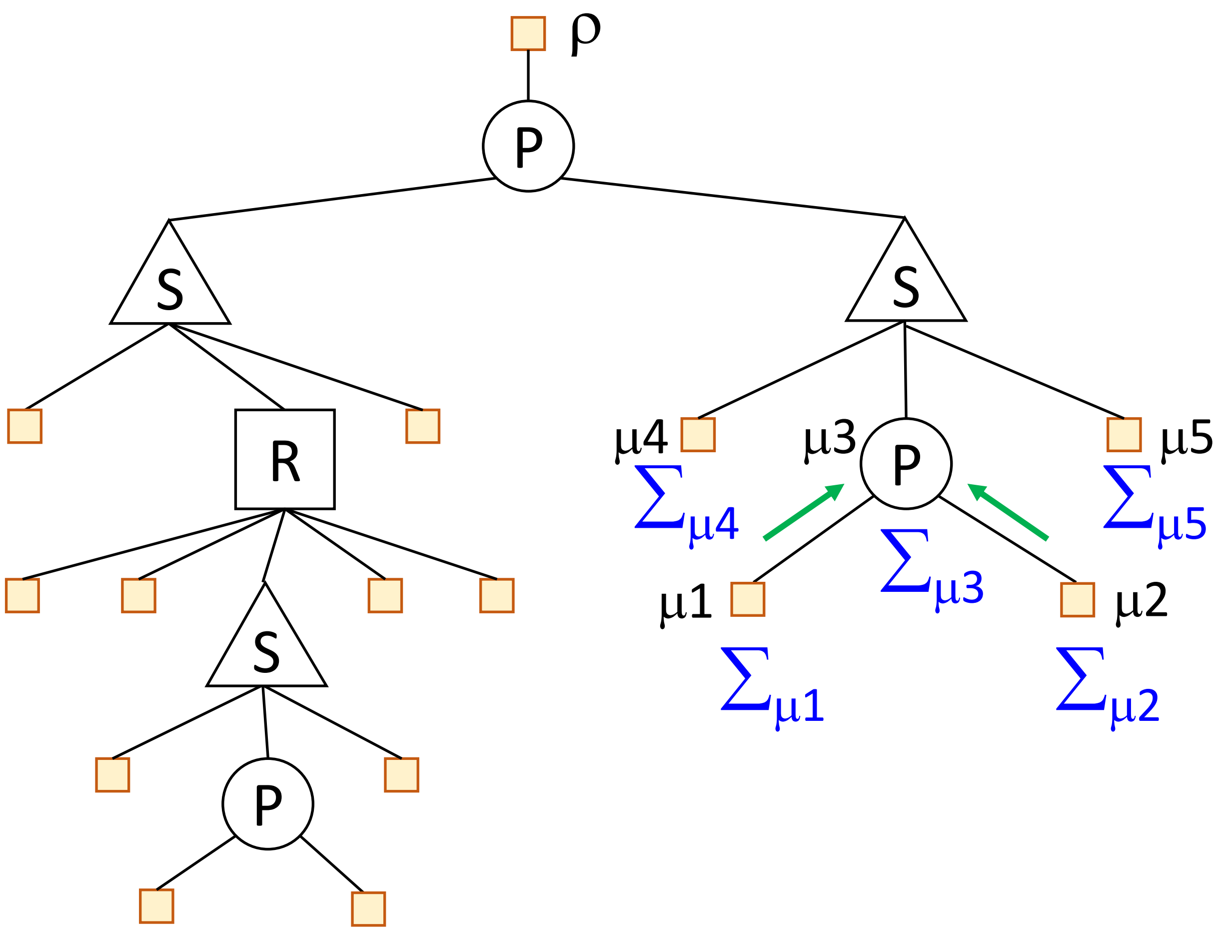
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{<\sigma_\mu , b(\sigma_\mu) > |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

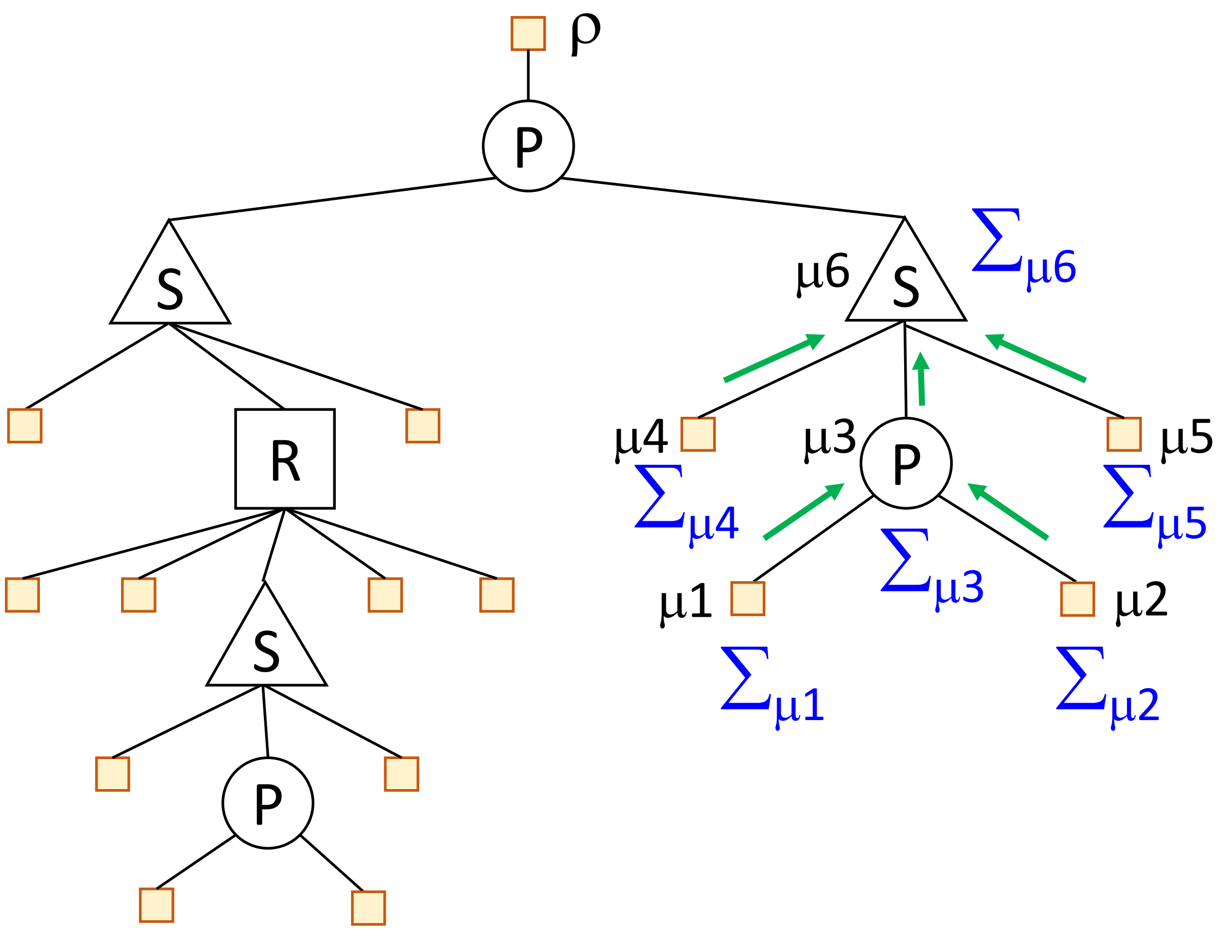
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{<\sigma_\mu, b(\sigma_\mu)> |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

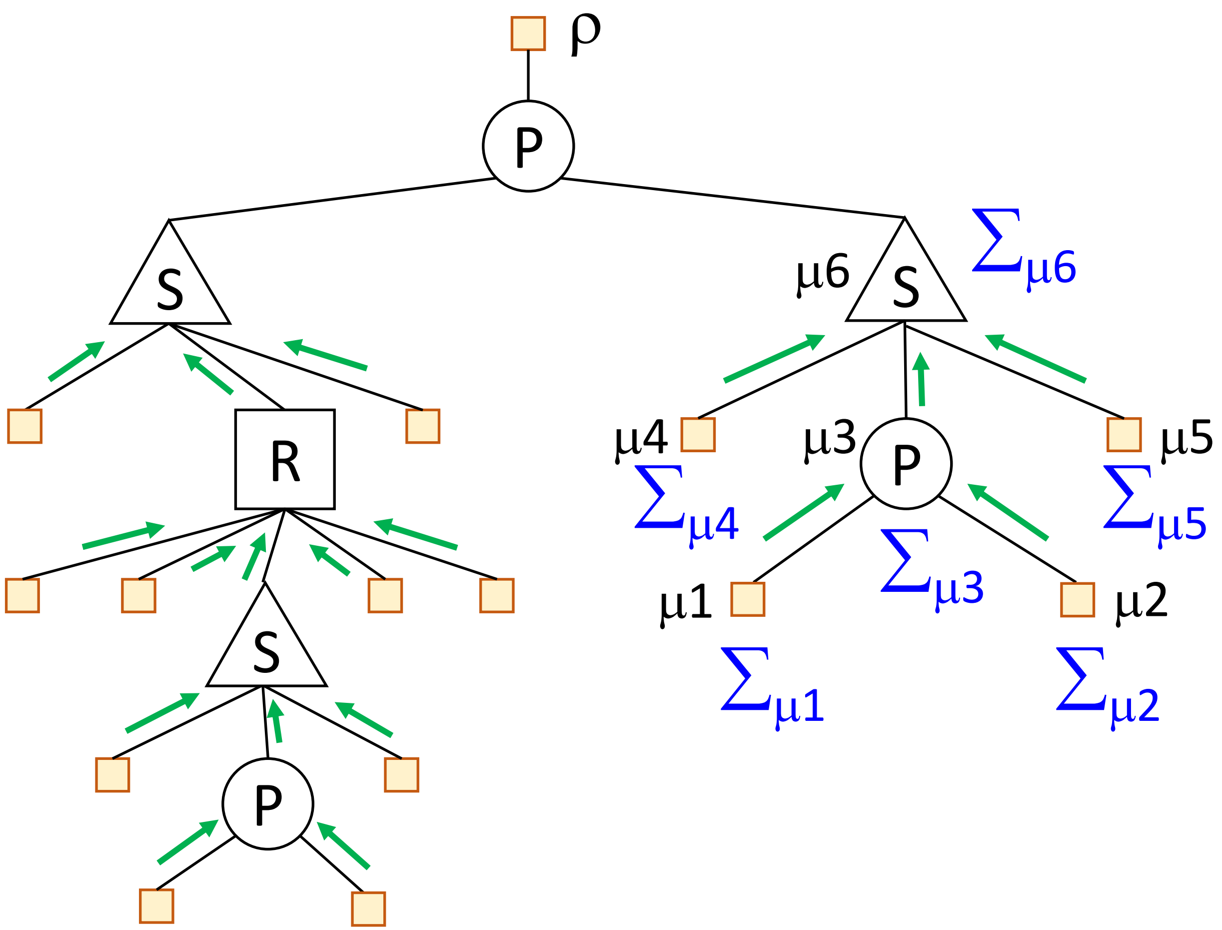
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ = $\{<\sigma_\mu, b(\sigma_\mu)> |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

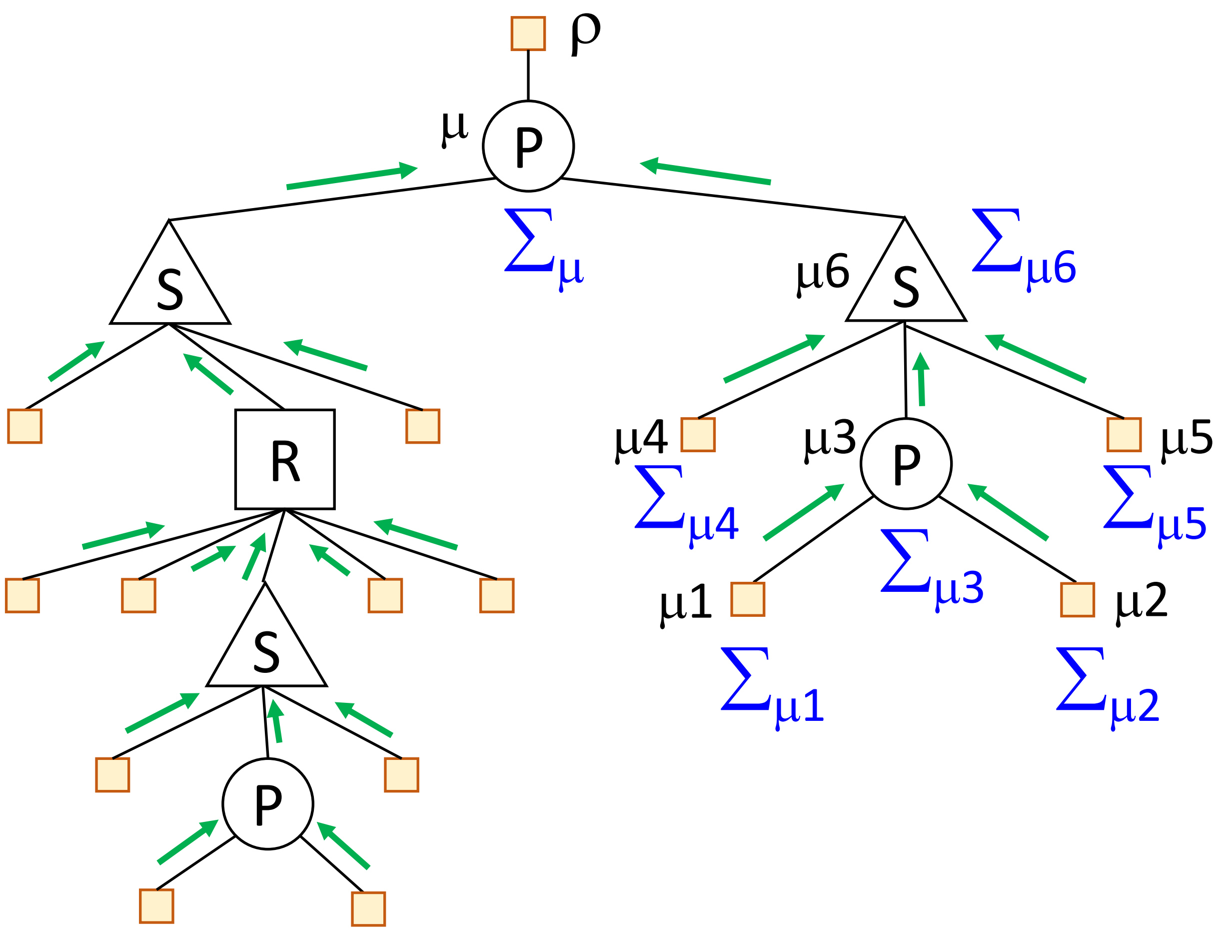
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{<\sigma_\mu, b(\sigma_\mu)> |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

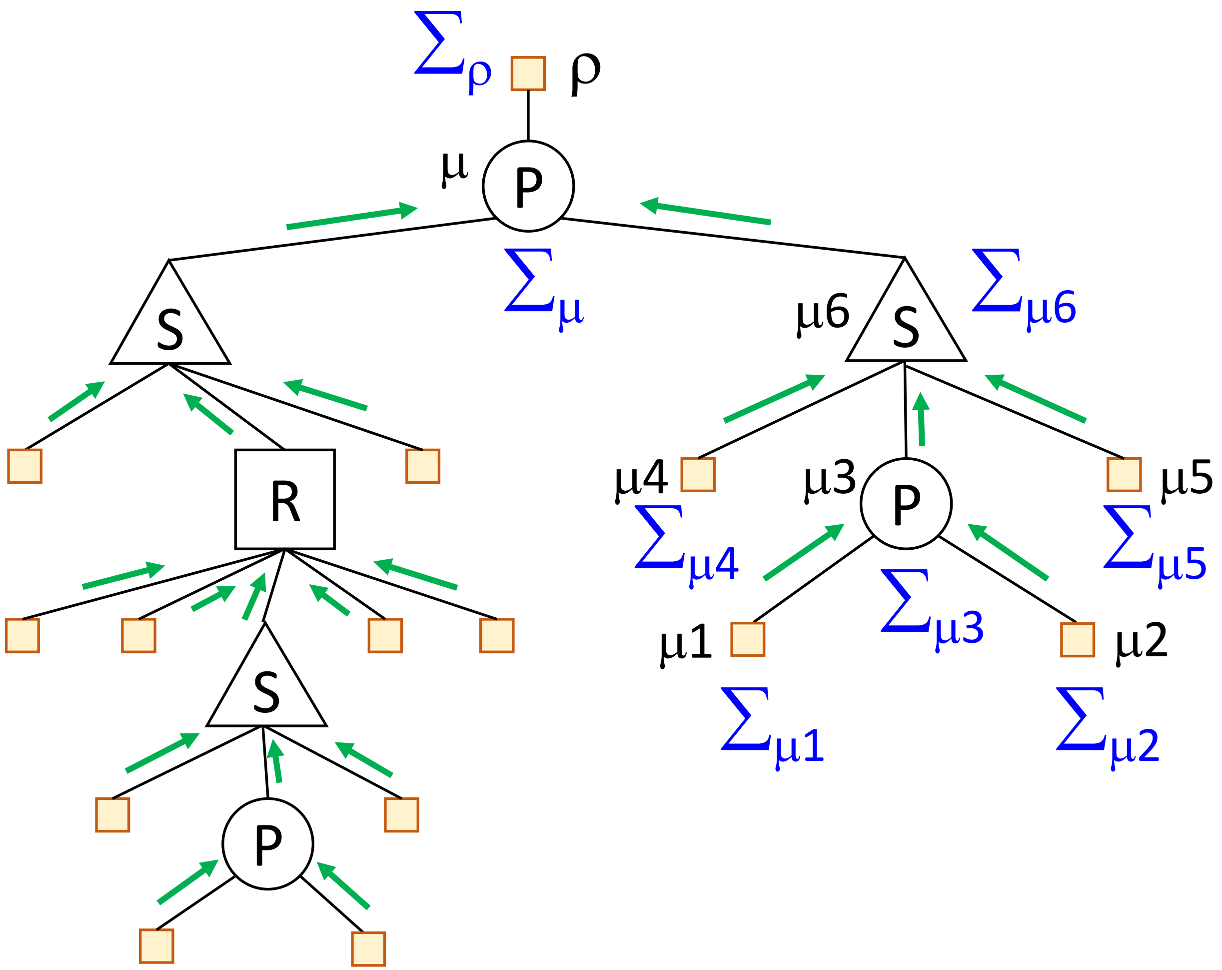
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ =
$\{<\sigma_\mu, b(\sigma_\mu)> |$
σ_μ = spirality of a component H_μ;
$b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

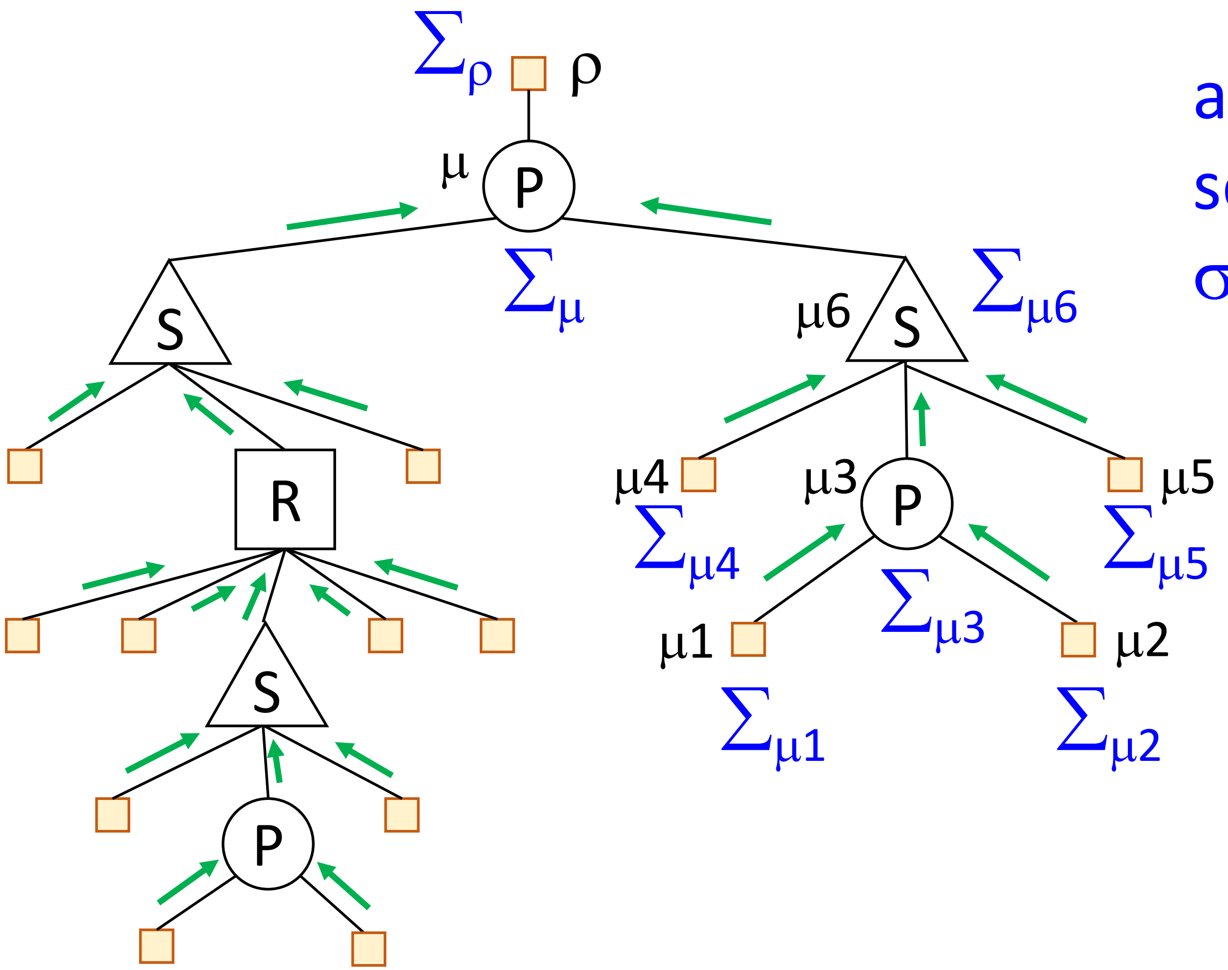
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



Σ_μ = optimal set of μ = $\{<\sigma_\mu , b(\sigma_\mu) > |$ σ_μ = spirality of a component H_μ; $b(\sigma_\mu)$ = min. bend for $\sigma_\mu\}$

Bend-min algorithm: General strategy

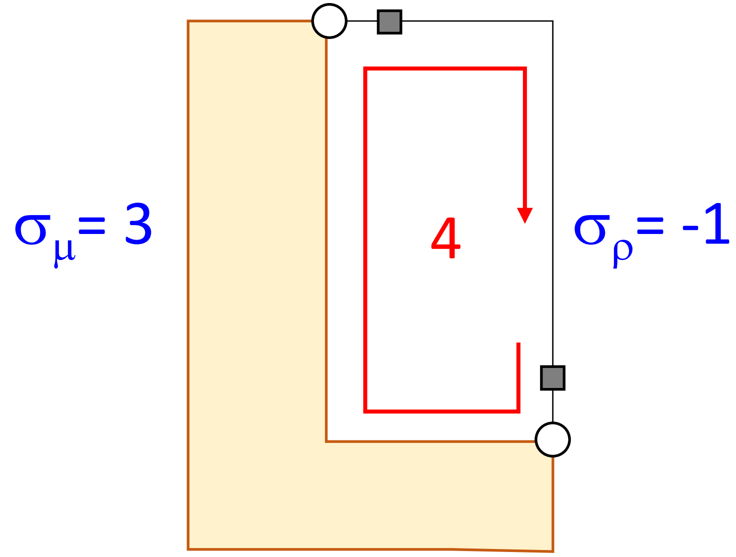
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



at the root level,

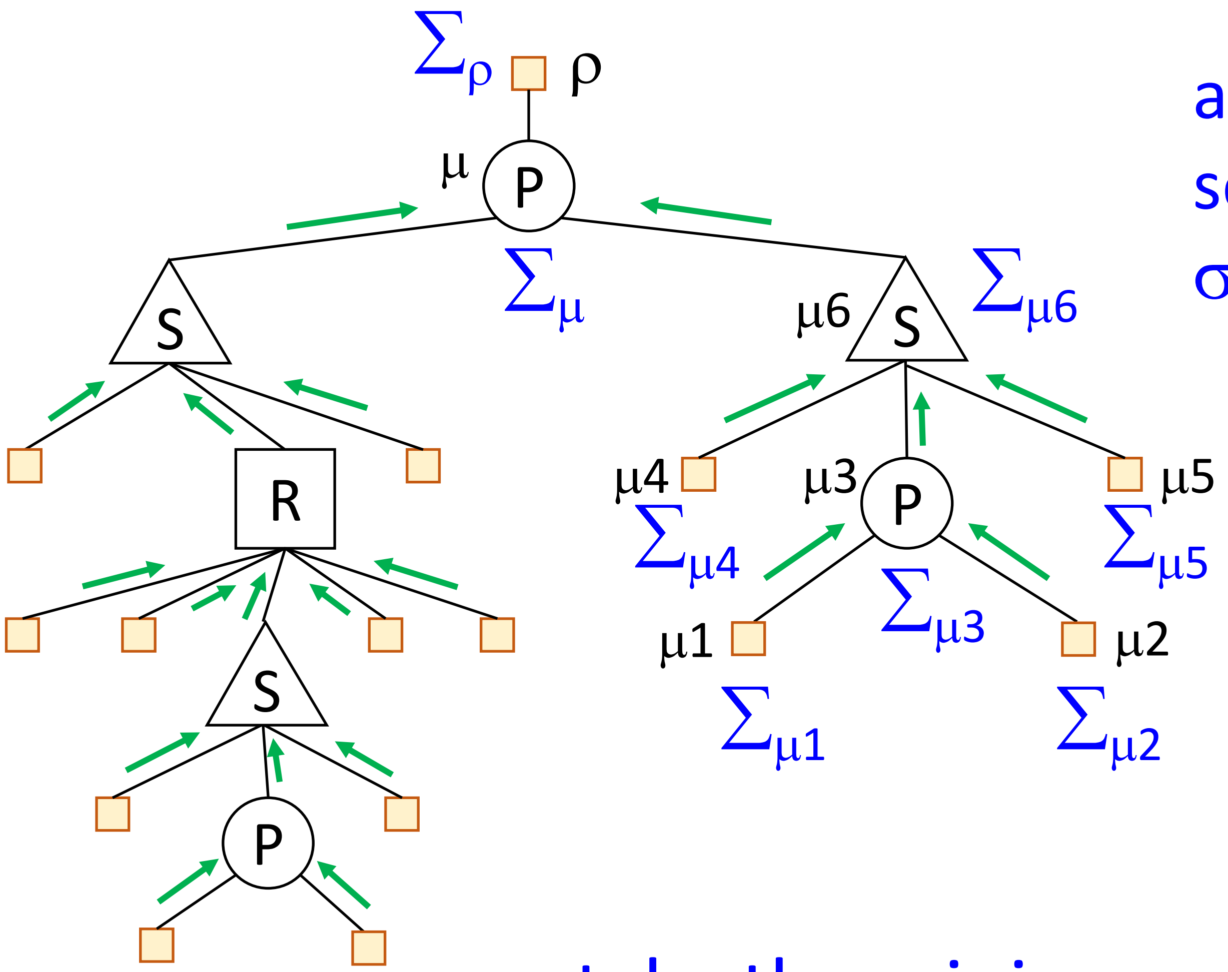
select the pair $\{\sigma_\mu , \sigma_\rho\}$ such that:

$\sigma_\mu - \sigma_\rho = 4$ and $b(\sigma_\mu) + b(\sigma_\rho)$ is minimum

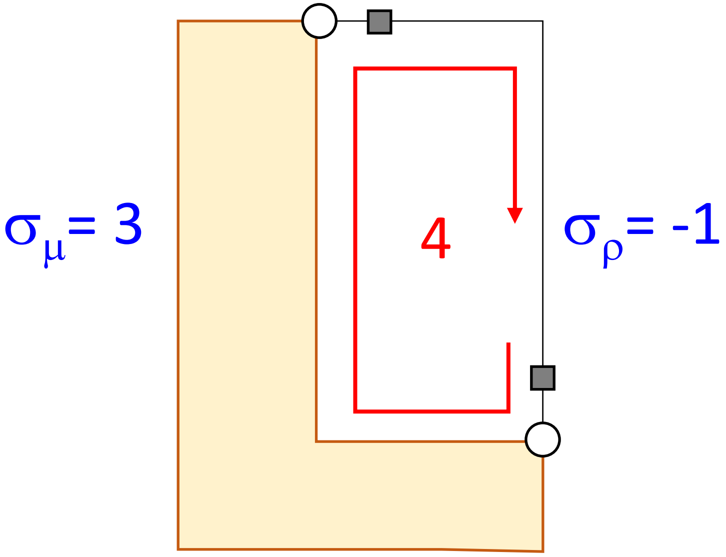


Bend-min algorithm: General strategy

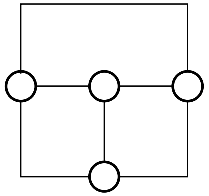
[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



at the root level,
select the pair $\{\sigma_\mu , \sigma_\rho\}$ such that:
$\sigma_\mu - \sigma_\rho = 4$ and $b(\sigma_\mu) + b(\sigma_\rho)$ is minimum

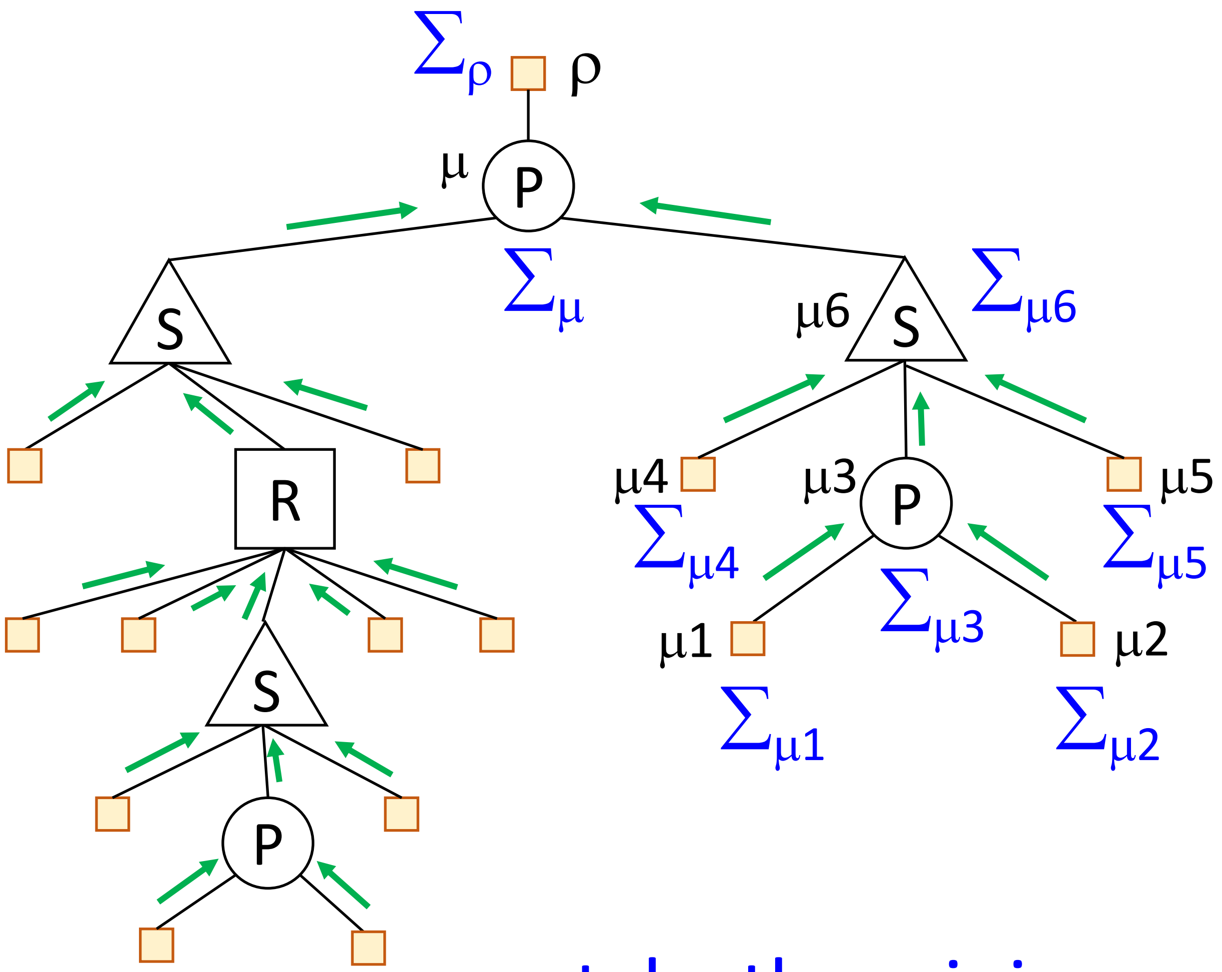


take the minimum over all roots!



Bend-min algorithm: General strategy

[Di Battista, Liotta, Vargiu, SIAM J. Comp. 1998]



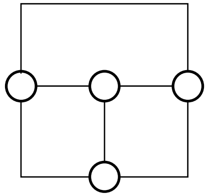
Question: How can we efficiently compute the optimal set of each node?

take the minimum over all roots!

Optimal sets: Preliminary observations

- A bend-min orthogonal representation has at most 2n-2 bends [Tamassia, Tollis, Vitter 1991]
  $\Rightarrow$ the spirality of a component is in the interval $[-3n+2, 3n-2]$

- Each bend is along a chain of a Q*-node

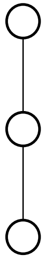


Optimal sets: Q*-nodes

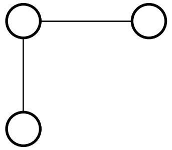
μ= Q*-node whose chain has length k

- consider all possible spirality values $\sigma_\mu \in [-3n+2, 3n-2]$
- for each $\sigma_\mu \Rightarrow b(\sigma_\mu) = \max\{0, |\sigma_\mu| - k + 1\}$

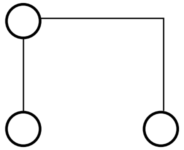
b(0) = 0



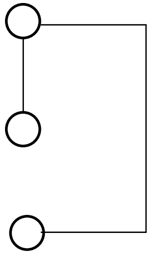
b(1) = 0



b(2) = 1



b(3) = 2



k=2

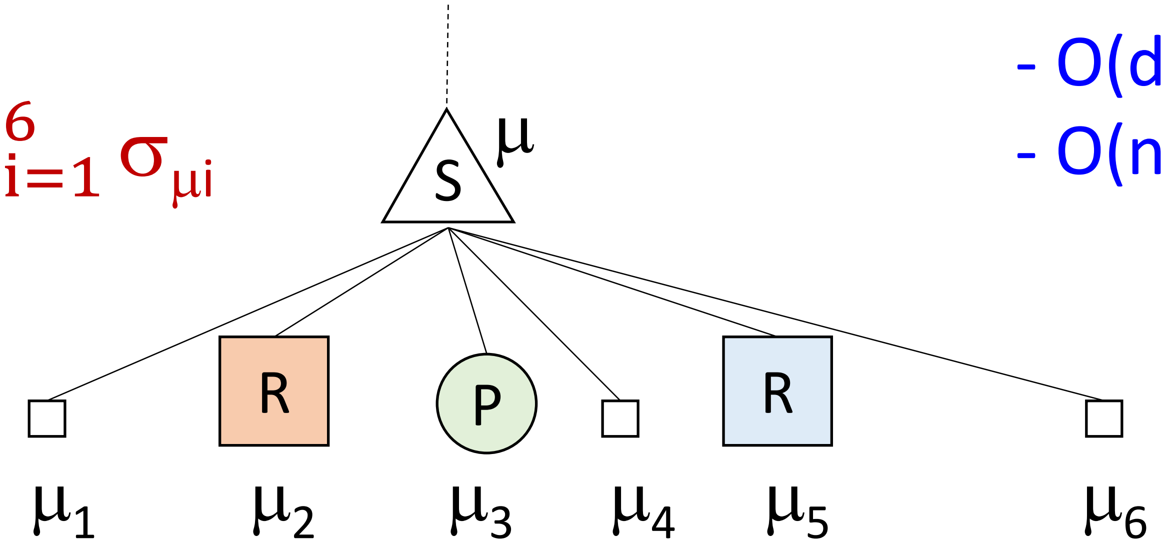
- O(n) time x node
- O(n^2) time all nodes

Optimal sets: S-nodes

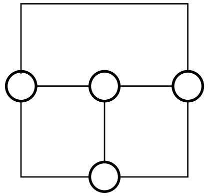
μ = S-node

- the spirality values σ_μ are all possible summations of the values of the children of μ
- for each summation $\sigma_\mu \Rightarrow$ $b(\sigma_\mu)$ = sum of the bends of the children (keep the minimum for σ_μ)

$$\sigma_\mu = \sum_{i=1}^{6} \sigma_{\mu i}$$



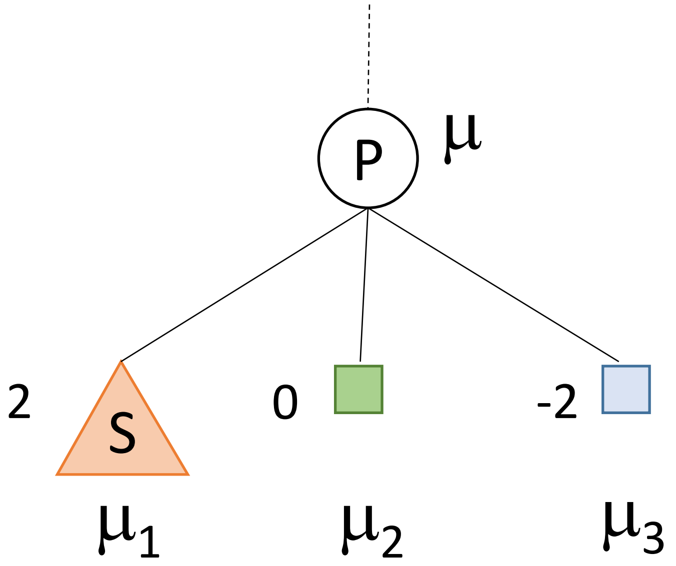
- $O(\deg(\mu)\, n^2)$ time x node
- $O(n^3)$ time all nodes



Optimal sets: P-nodes

μ = P-node

- the spirality values σ_μ must satisfy the parallel relationship
- for each $\sigma_\mu \Rightarrow$ b(σ_μ) = sum of the bends of the children (keep the minimum on σ_μ)



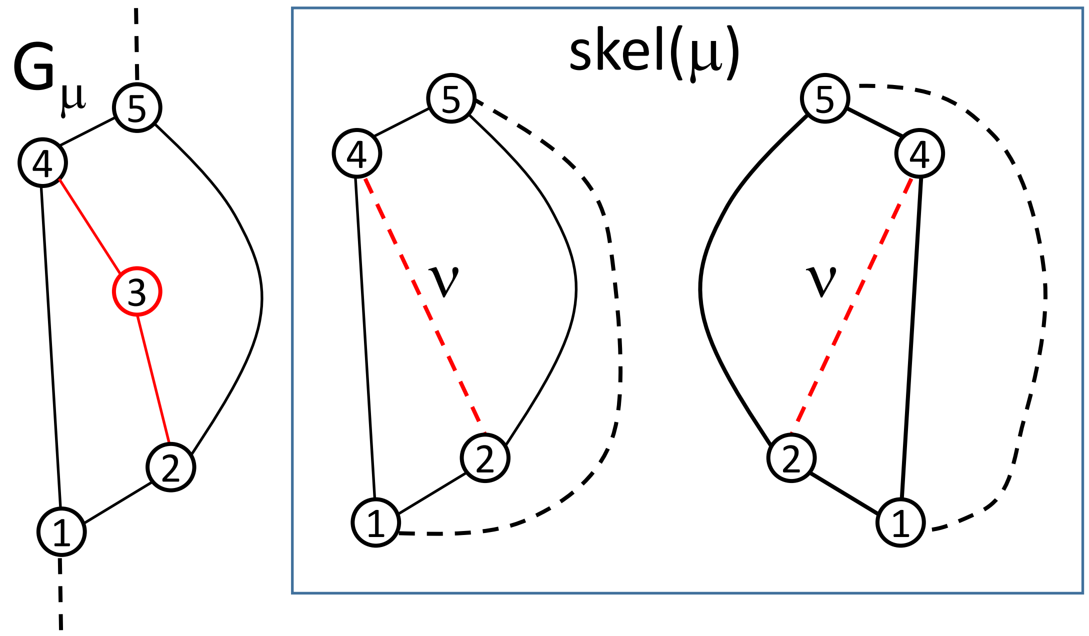
- O(n) time x node
- O(n^2) time all nodes

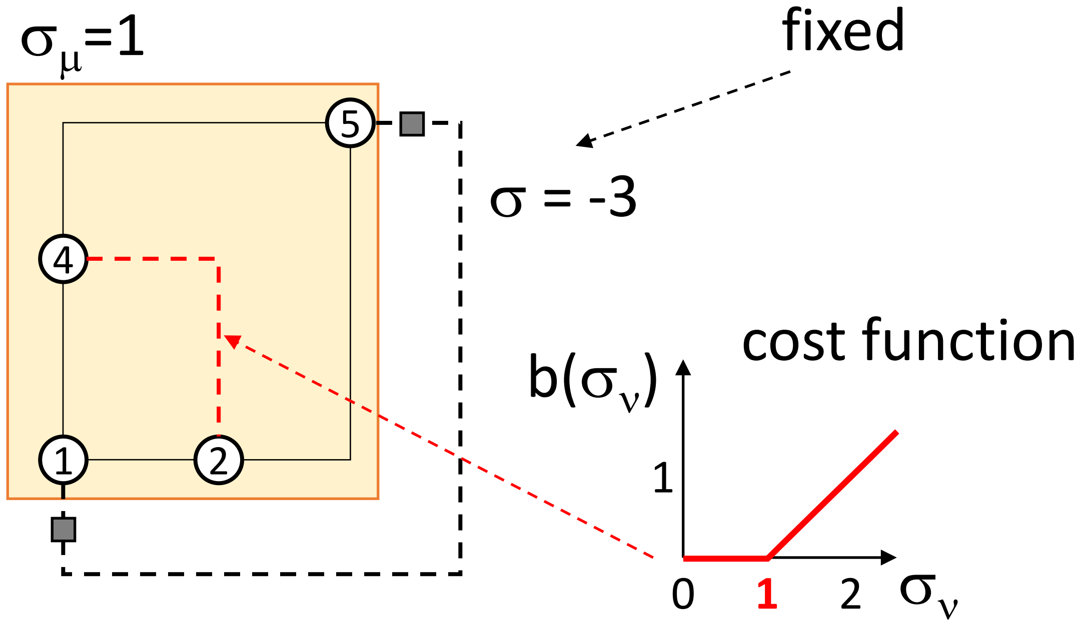
$$\sigma_\mu = \sigma_{\mu 1} - 2 = \sigma_{\mu 2} = \sigma_{\mu 3} + 2$$

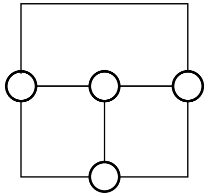
Optimal sets: R-nodes

μ = R-node

- consider all possible values $\sigma_\mu \in [-3n+2, 3n-2]$
- for each $\sigma_\mu \Rightarrow b(\sigma_\mu)$ = *constrained* min-cost flow with virtual edges having *convex-cost functions* (the cost of the corresponding series)







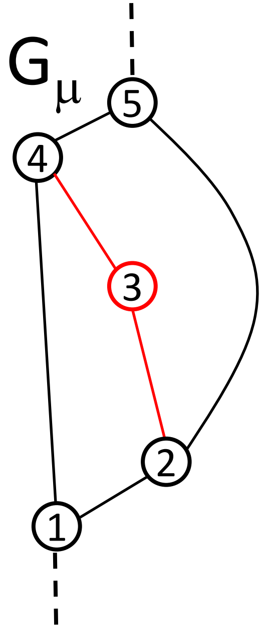
Optimal sets: R-nodes

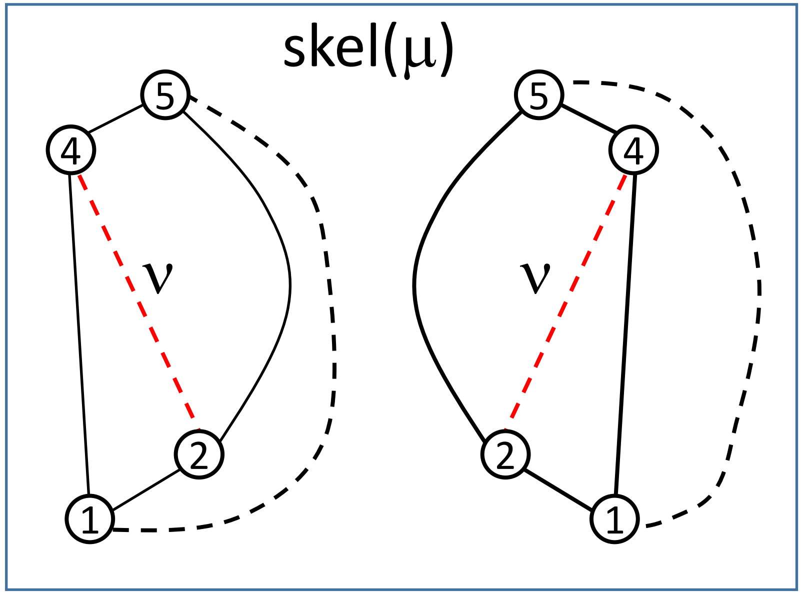
- $O(nT(deg(\mu))$ time x node
- $O(nT(n))$ time all nodes

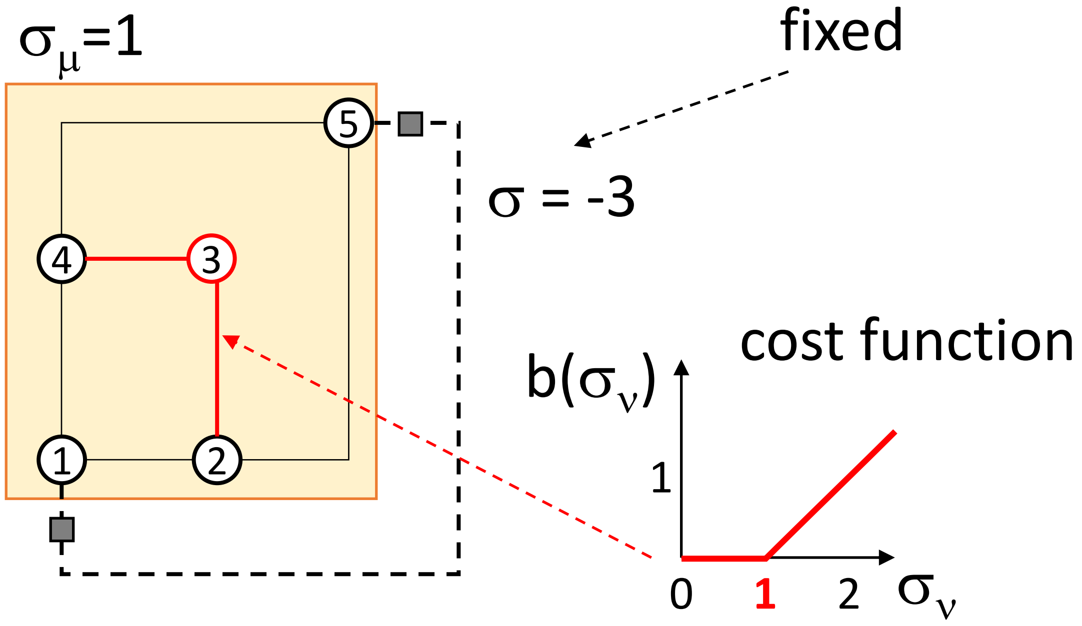
T(.) = min-cost flow time

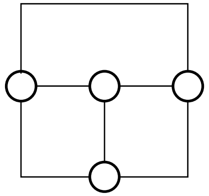
μ = R-node

- consider all possible values $\sigma_\mu \in [-3n+2, 3n-2]$
- for each $\sigma_\mu \Rightarrow b(\sigma_\mu)$ = *constrained* min-cost flow with virtual edges having *convex-cost functions* (the cost of the corresponding series)





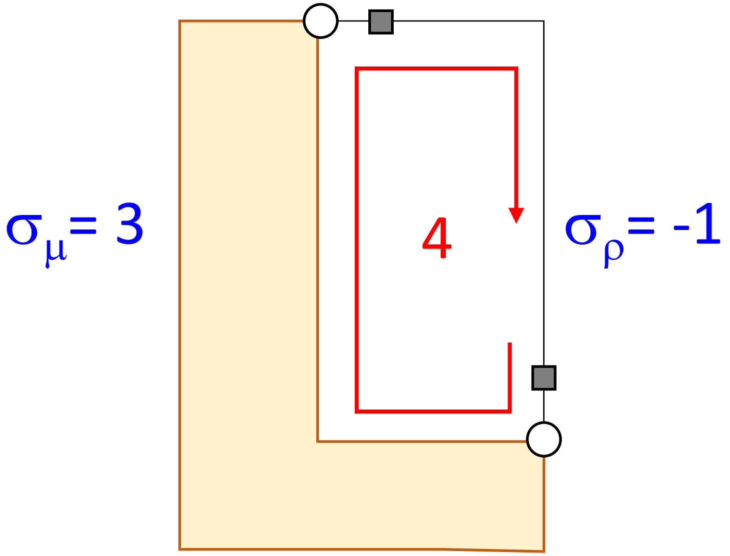




Optimal sets: Root level

μ = child of the root

- consider all possible values σ_μ and σ_ρ such that $\sigma_\mu - \sigma_\rho = 4$
- $b(\rho) = \min \{b(\sigma_\mu) + b(\sigma_\rho) \mid \sigma_\mu - \sigma_\rho = 4 \}$



- O(n) time

Time complexity: Summary

- For a single rooted tree T_ρ

assuming $T(n) = o(n^2)$

$$O(n^2) + O(n^3) + O(n^2) + O(nT(n)) + O(n) \quad = O(n^3)$$

Q*-nodes S-nodes P-nodes R-nodes root

- For all rooted trees T_ρ $O(n^4)$

Time complexity: Summary

- For a single rooted tree T_ρ

assuming $T(n) = o(n^2)$

$$\underbrace{O(n^2)}_{\text{Q*-nodes}} + \underbrace{O(n^3)}_{\text{S-nodes}} + \underbrace{O(n^2)}_{\text{P-nodes}} + \underbrace{O(nT(n))}_{\text{R-nodes}} + \underbrace{O(n)}_{\text{root}} = O(n^3)$$

- For all rooted trees T_ρ $\boxed{O(n^4)}$

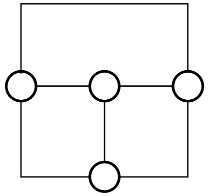
Question: Can we do better?

Time complexity: Bottlenecks

1. Processing all S-nodes for a single tree takes $O(n^3)$ time – can we reduce the time needed to process S-nodes?

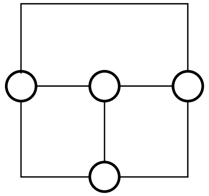
Time complexity: Bottlenecks

1. Processing all S-nodes for a single tree takes $O(n^3)$ time – can we reduce the time needed to process S-nodes?
2. Can we avoid the O(n) factor caused by exploring all roots?



Time complexity: Bottlenecks

1. Processing all S-nodes for a single tree takes $O(n^3)$ time – can we reduce the time needed to process S-nodes?
2. Can we avoid the $O(n)$ factor caused by exploring all roots?
3. Can we avoid to solve a min-cost-flow problem in the presence of R-nodes?



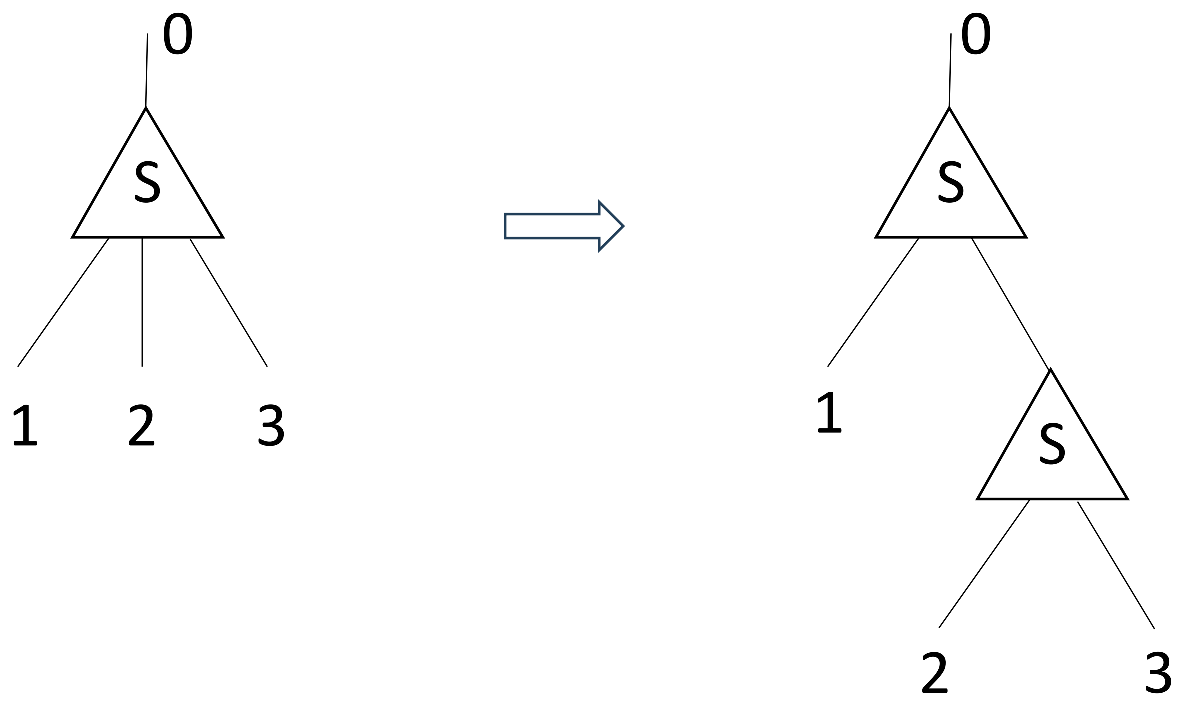
Time complexity: Bottlenecks

1. Processing all S-nodes for a single tree takes $O(n^3)$ time – can we reduce the time needed to process S-nodes?
2. Can we avoid the $O(n)$ factor caused by exploring all roots?
3. Can we avoid to solve a min-cost-flow problem in the presence of R-nodes?

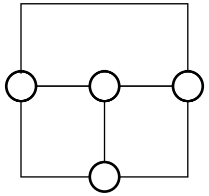
SP-graphs

S-nodes – a smarter approach

- Consider normalized SPQ*-trees
  - each S-node has two children



- The number of S-nodes is still O(n) and the structure of the tree does not change when we change the root



S-nodes – a smarter approach

- Consider any ordered sequence of all normalized rooted SPQ*-trees $T_{\rho 1}$, $T_{\rho 2}$, … , $T_{\rho h}$

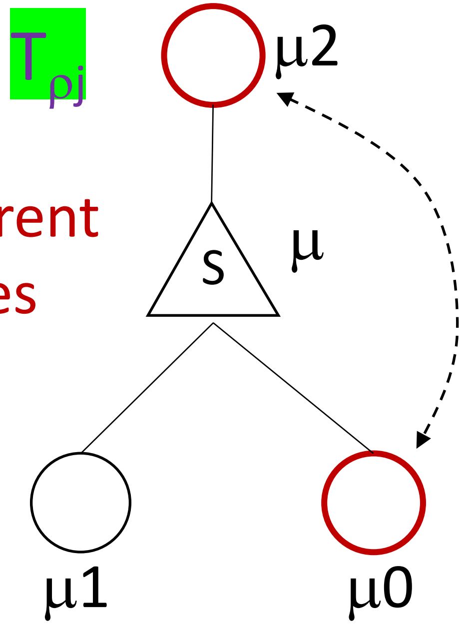
S-nodes – a smarter approach

$T_{\rho 1}, \ldots T_{\rho i}, \ldots T_{\rho j}, \ldots, T_{\rho h}$

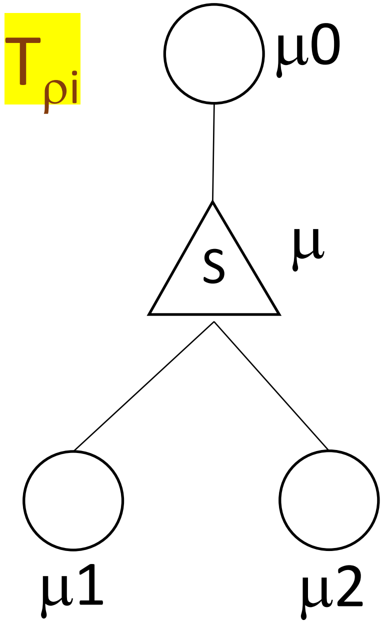
processed

case 2

the parent changes

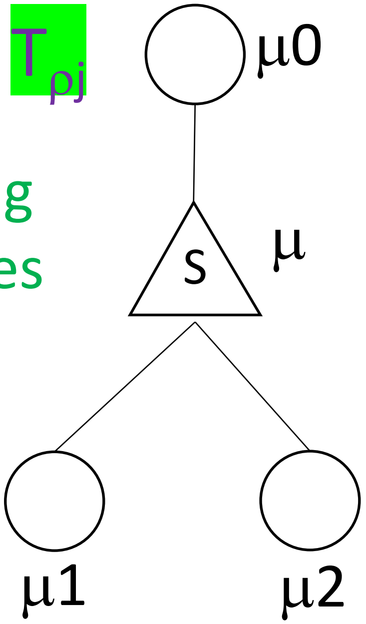


O(n^2) time



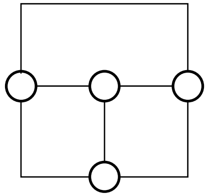
case 1

nothing changes



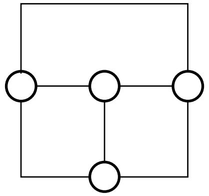
O(1) time (just reuse)

- 3 distinct parents per S-node
- O(n^3) over all O(n) S-nodes



SP-graphs: improved time complexity

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time



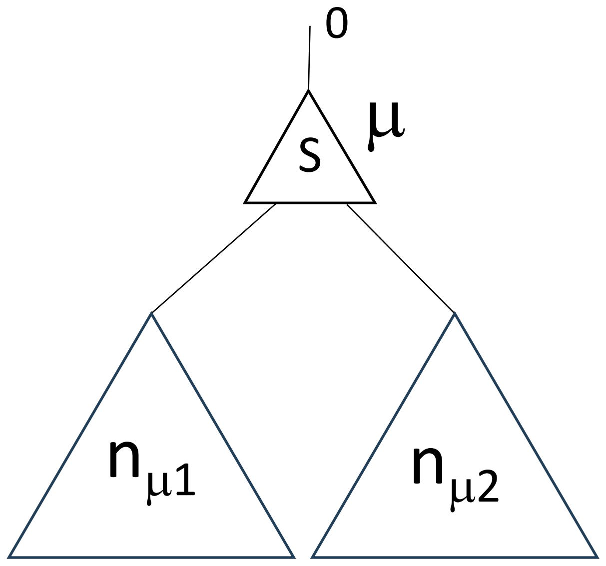
SP-graphs: improved time complexity

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

Question. Can we further improve the time complexity for the rectilinear planarity testing problem (0 bends)?

S-nodes: rectilinear planarity testing

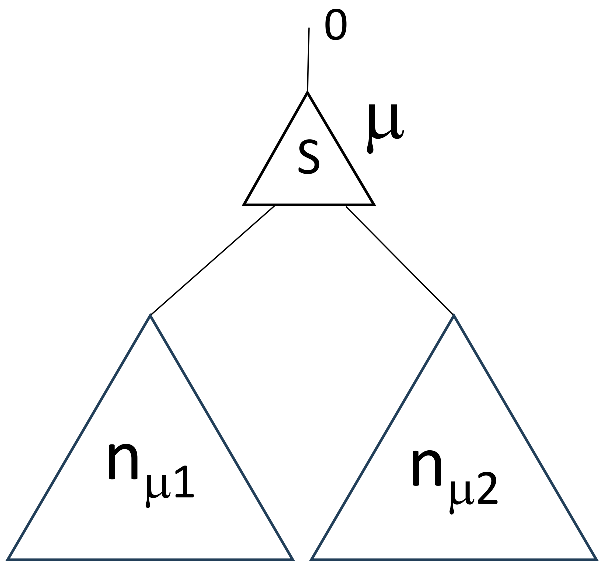
Observation. Computing the set of spirality values for an S-node μ with two child components of size $n_{\mu 1}$ and $n_{\mu 2}$ take $O(n_{\mu 1}\, n_{\mu 2})$ time



.. because in a rectilinear drawing the absolute spirality of a component with k vertices is at most k-2

S-nodes: rectilinear planarity testing

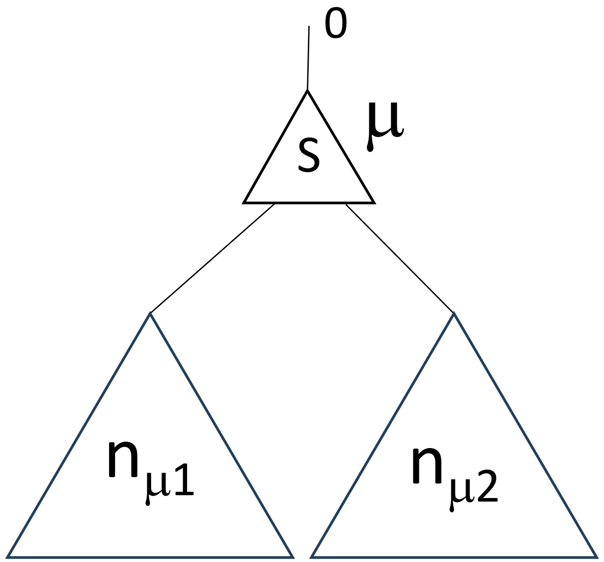
Lemma. The sum of the products of the sizes of the pertinent graphs of all S-node children in a normalized rooted SPQ*-tree is $O(n^2)$



$$\sum_{\mu} O(n_{\mu 1}\, n_{\mu 2}) = O(n^2)$$

S-nodes: rectilinear planarity testing

Lemma. The sum of the products of the sizes of the pertinent graphs of all S-node children in a normalized rooted SPQ*-tree is $O(n^2)$



$$\sum_{\mu} O(n_{\mu 1}\, n_{\mu 2}) = O(n^2)$$

proved by induction on the depth of the subtree $T(\nu)$ rooted at ν

$$\boxed{s(\nu) = \sum_{\mu \in \{S\text{−nodes in } T(\nu)\}} n_{\mu 1}\, n_{\mu 2} \leq 4 m_\nu^{\,2}}$$

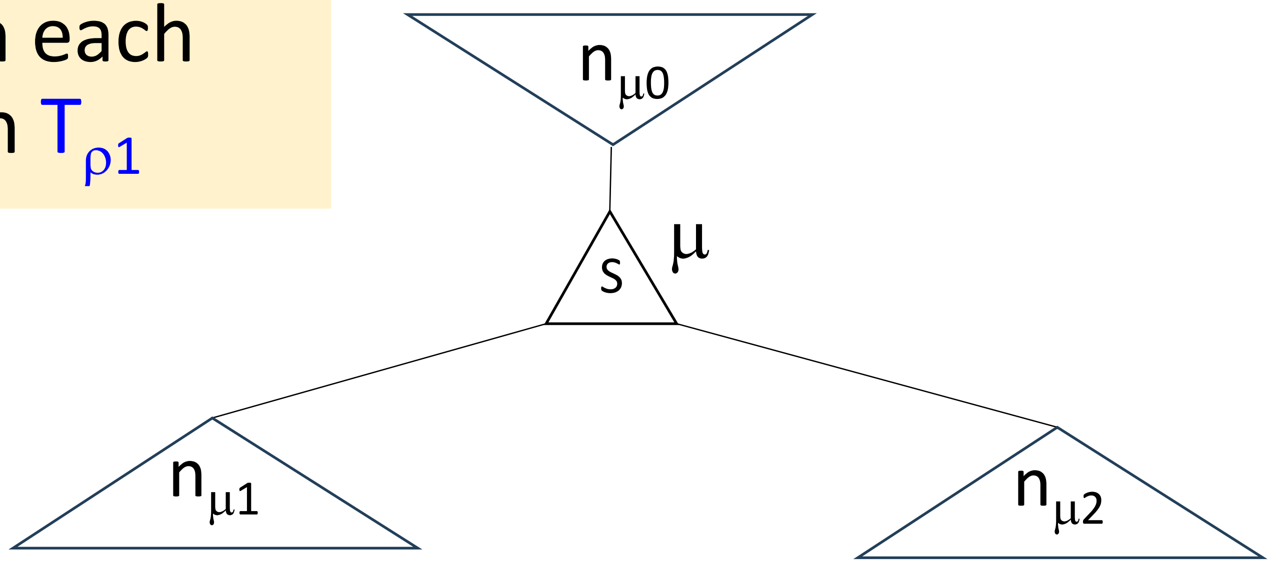
SP-graphs

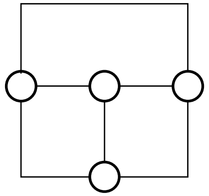
Lemma. The sum of the products of the sizes of the pertinent graphs of all S-node children in a normalized rooted SPQ*-tree is $O(n^2)$

+

Lemma. Processing an S-node μ in each tree $T_{\rho j}$ does not cost more than in $T_{\rho 1}$

$\min\{n_{\mu 1}n_{\mu 2}, n_{\mu 2}n_{\mu 0}, n_{\mu 0}n_{\mu 1}\}$

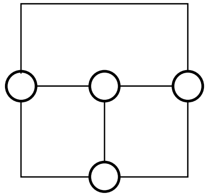




SP-graphs

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

Theorem 2. Let G be an n-vertex SP-graph**.** There exists an algorithm that tests whether G is rectilinear planar in $O(n^2)$ time



SP-graphs

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

Theorem 2. Let G be an n-vertex SP-graph**.** There exists an algorithm that tests whether G is rectilinear planar in $O(n^2)$ time

Question. Can we achieve linear time?

Strive for linear time

- The given approach stores O(n) spirality values per node
- This does not allow us to achieve O(n) time complexity in total!

Question: is there any constant upper bound on the maximum spirality of a component?

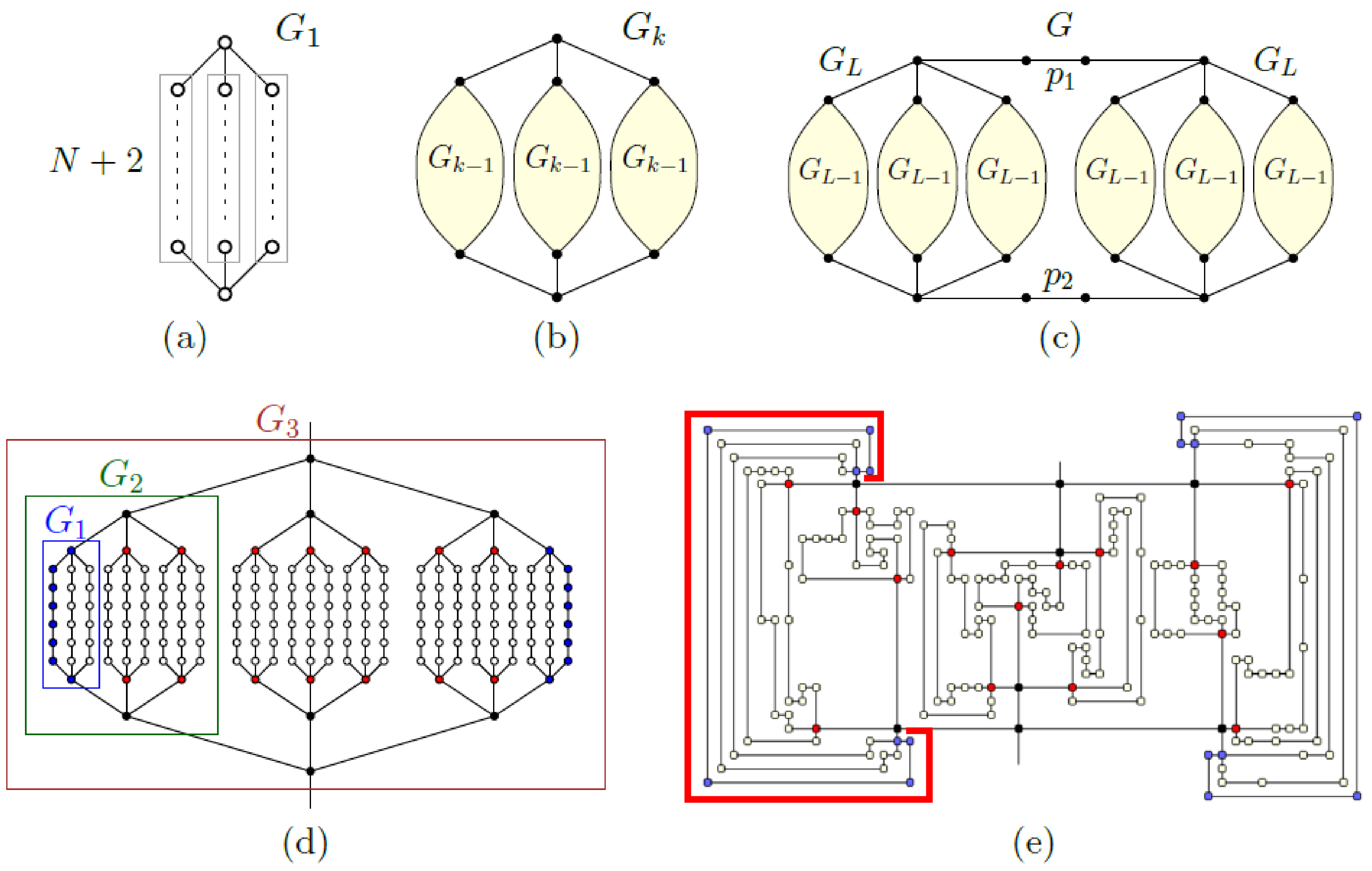
Strive for linear time

- The given approach stores O(n) spirality values per node
- This does not allow us to achieve O(n) time complexity in total!

Question: is there any constant upper bound on the maximum spirality of a component?

Answer: this is not always the case!

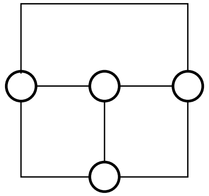
Spirality – Logarithmic lower bound



- $N \geq 2$ (even)
- $L = N/2+1$
- $n = \theta(3^N)$

one of the three series of G_1 will have spirality $\sigma = N+2$

$N = 4$, $L=3$, $\sigma = 6$



SP-graphs

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

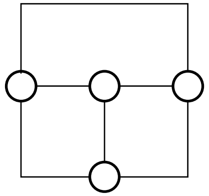
Theorem 2. Let G be an n-vertex SP-graph**.** There exists an algorithm that tests whether G is rectilinear planar in $O(n^2)$ time

Theorem 3. There exist infinitely many SP-graphs that require components with spirality $\Omega(\log n)$ in any given rectilinear planar representation

Let’s not lose hope

Even if we must handle sets of spirality of non-constant size …

Question: can we represent them in O(1) space?



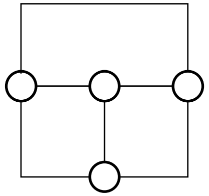
Let's not lose hope

Even if we must handle sets of spirality of non-constant size …

Question: can we represent them in O(1) space?

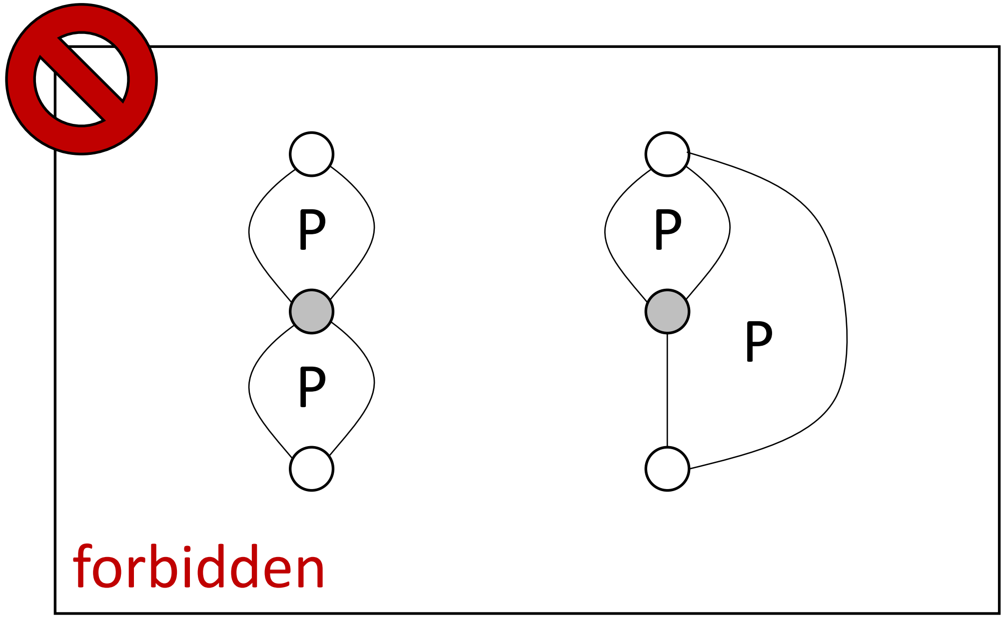
Answers:

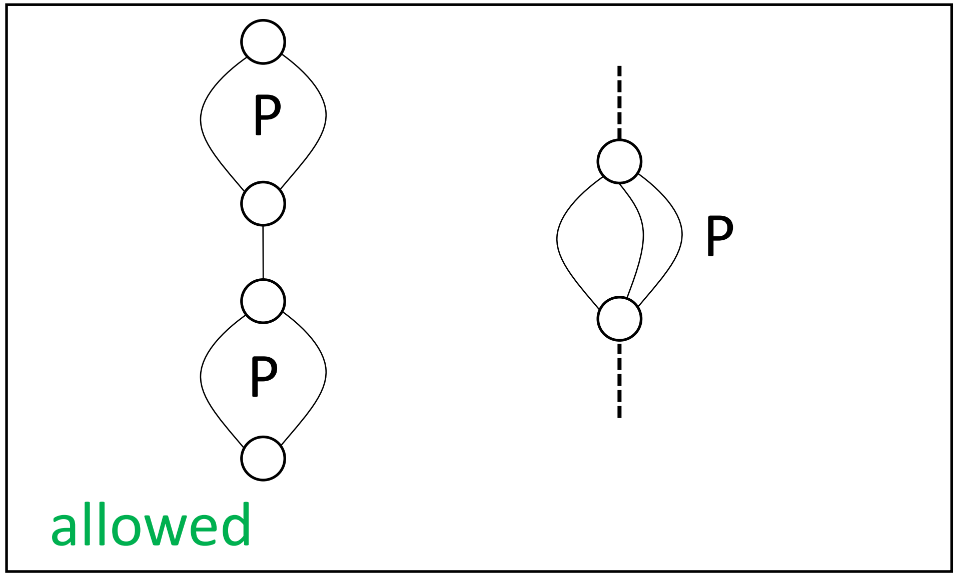
1) possible for a meaningful subclass of SP-graphs, called independent-parallel SP-graphs
2) unclear for general SP-graphs (probably not)



Independent-parallel SP-graphs

Independent-parallel means “no two P-nodes share a pole”

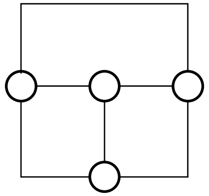




Independent-parallel SP-graphs

- Rectilinear planarity testing of independent-parallel SP-graphs can be executed in O(n) time

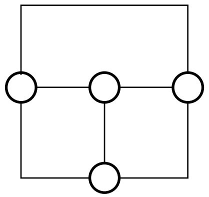
- **Main ingredients**:
  - each component has one of the following sets of spirality values:
    - {0};
    - {-1, 1};
    - {-2, -1, 1, 2};
    - {-M, -M+1,…, 0,…, M-1, M};
    - {-M, -M+2, -M+4,…., M-4, M-2, M}



Independent-parallel SP-graphs

- Rectilinear planarity testing of independent-parallel SP-graphs can be executed in O(n) time

- **Main ingredients**:
  - dynamic programming on (not normalized) SPQ*-trees
  - the set of each Q*-node and of each P-node is computed in O(1) time
  - the set of each S-node μ is computed in O(deg(μ)) time in the first tree and in O(1) time in the remaining trees
  - rectilinear planarity at the root level is tested in O(1) time



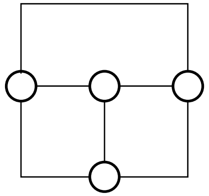
SP-graphs

Theorem 1. Let G be an n-vertex SP-graph**.** There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

Theorem 2. Let G be an n-vertex SP-graph**.** There exists an algorithm that tests whether G is rectilinear planar in $O(n^2)$ time

Theorem 3. There exist infinitely many SP-graphs that require components with spirality $\Omega(\log n)$ in any given rectilinear planar representation

Theorem 4. Let G be an n-vertex independent-parallel SP-graph. There exists an algorithm that tests whether G is rectilinear planar in $O(n)$ time



SP-graphs

[D., Kaufmann, Liotta, Ortali, JGAA 2023]

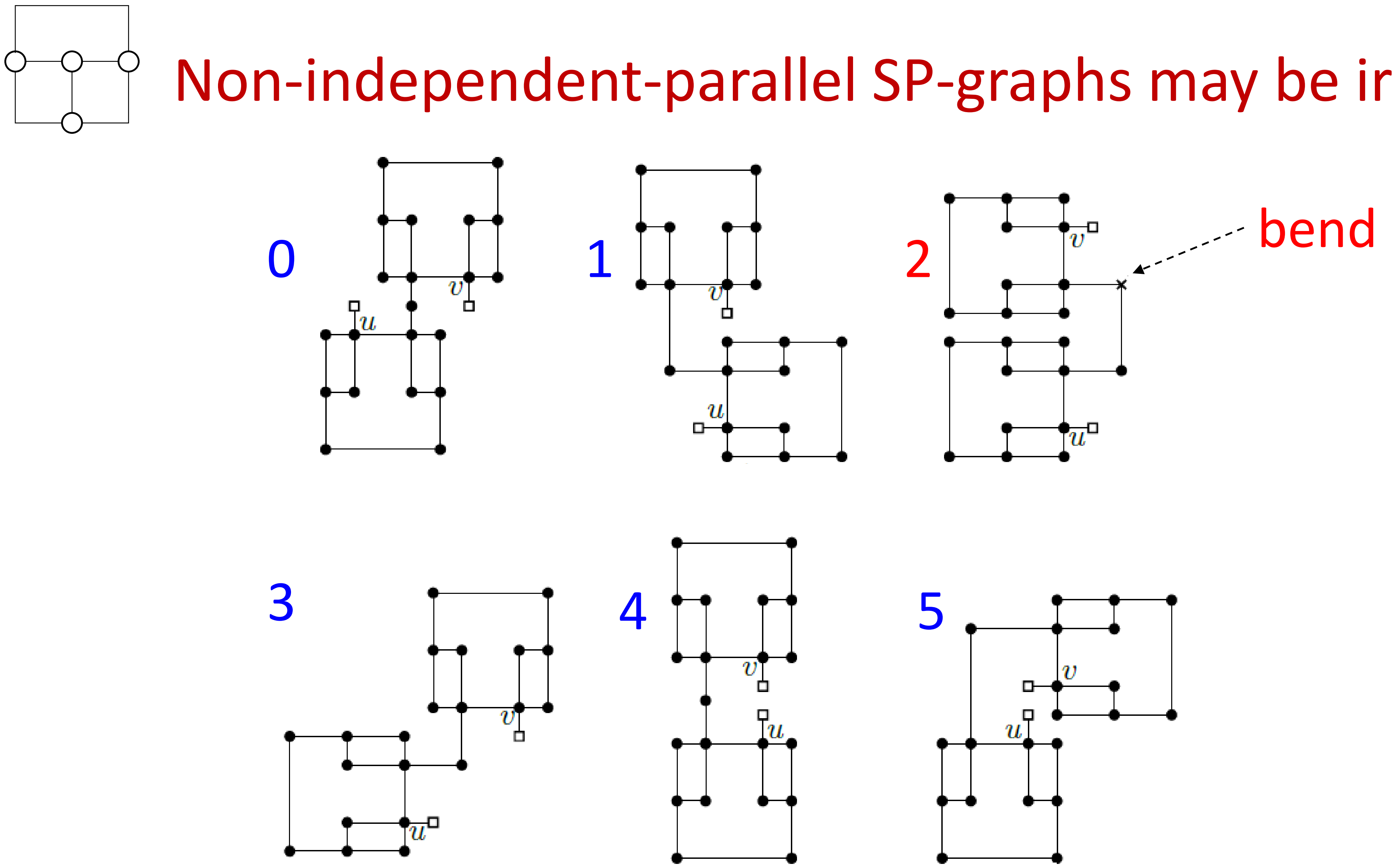
Theorem 1. Let G be an n-vertex SP-graph. There exists an algorithm that computes a bend-minimum orthogonal drawing of G in $O(n^3)$ time

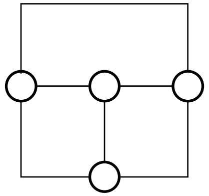
Theorem 2. Let G be an n-vertex SP-graph. There exists an algorithm that tests whether G is rectilinear planar in $O(n^2)$ time

Theorem 3. There exist infinitely many SP-graphs that require components with spirality $\Omega(\log n)$ in any given rectilinear planar representation

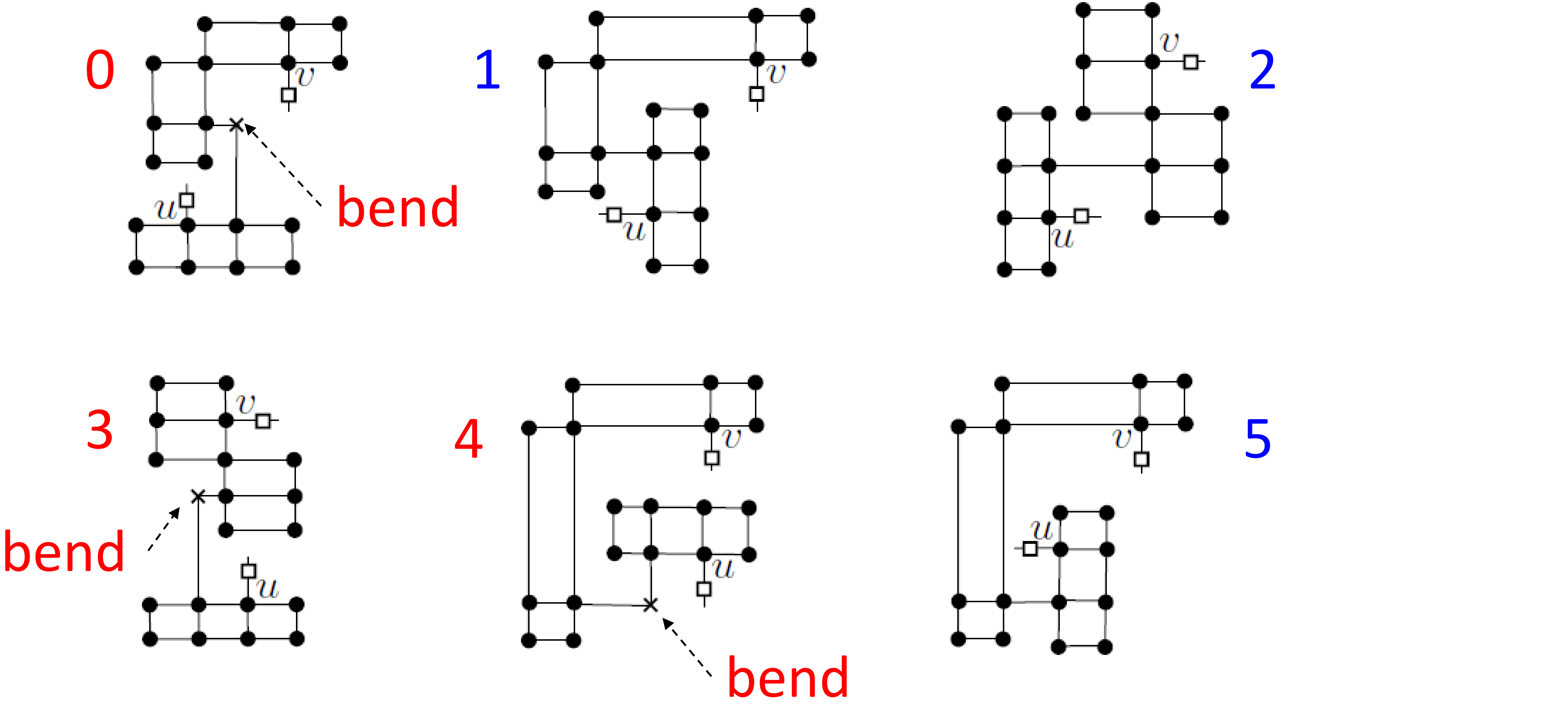
Theorem 4. Let G be an n-vertex independent-parallel SP-graph. There exists an algorithm that tests whether G is rectilinear planar in $O(n)$ time

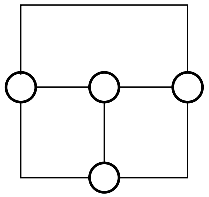
Non-independent-parallel SP-graphs may be irregular



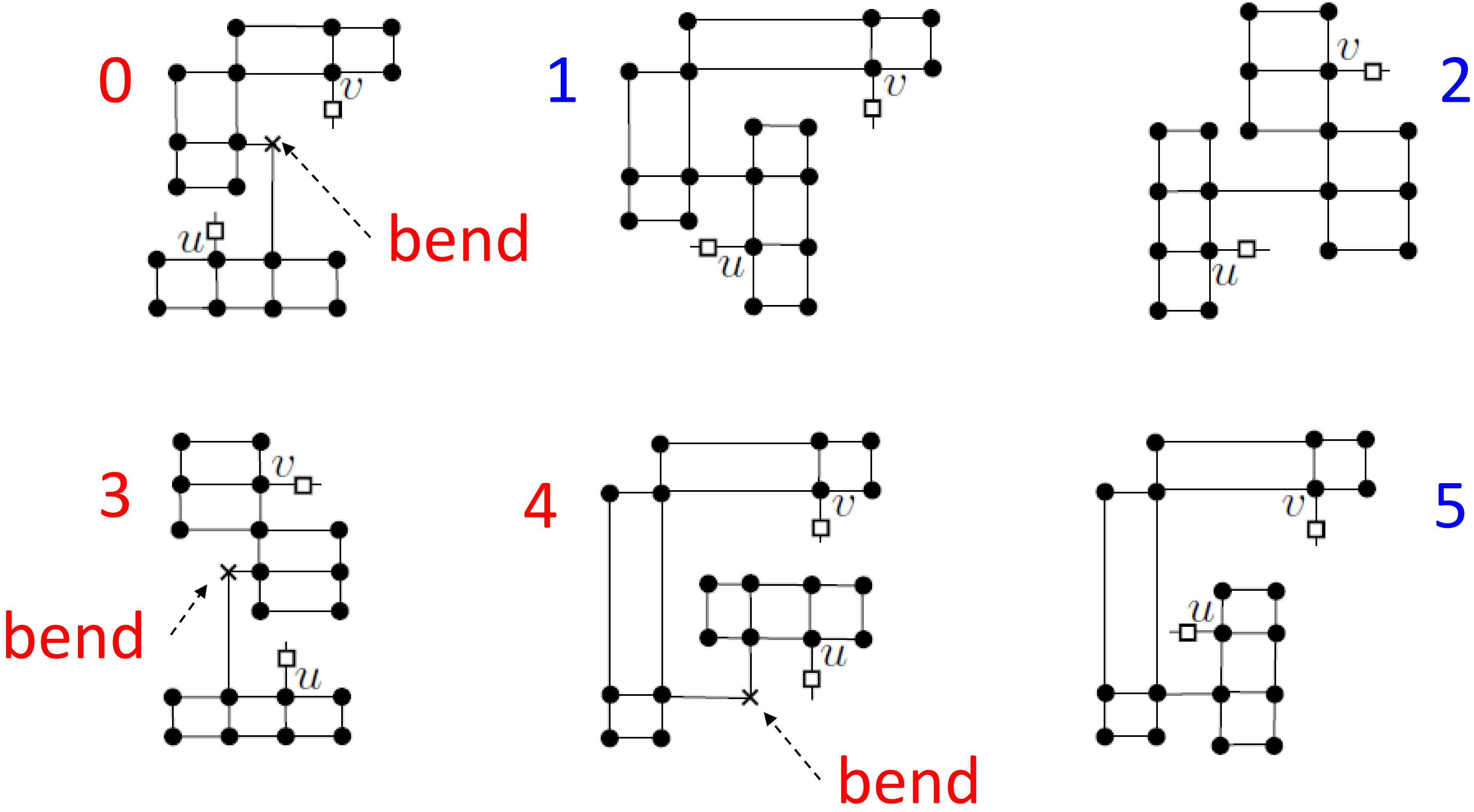


Non-independent-parallel SP-graphs may be irregular

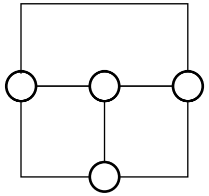




Non-independent-parallel SP-graphs may be irregular



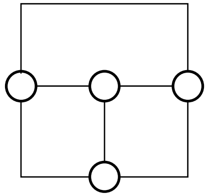
we *cannot* generalize the approach used for independent-parallel SP-graphs



SP-graphs: Open Problems

Open Problem 3: Can we improve the complexity of the bend-minimization problem for SP-graphs?

Recall: The best bound is $O(n^3)$



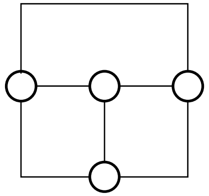
SP-graphs: Open Problems

Open Problem 3: Can we improve the complexity of the bend-minimization problem for SP-graphs?

Recall: The best bound is $O(n^3)$

Open Problem 4: Can we improve the complexity of rectilinear planarity testing for (general) SP-graphs?

Recall: The best bound is $O(n^2)$



Planar 3-graphs: A long history

1998

$O(n^5 \log n)$
Di Battista, Liotta, Vargiu

1998

$O(n^4)$
with a finer analysis

2017

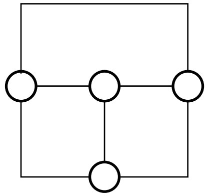
$O(n^{2.43} \log^k n)$
Chang and Yen

2018

$O(n^2)$
D., Liotta, Patrignani

2020

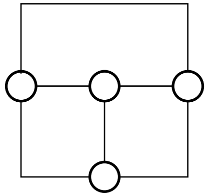
$O(n)$ time
D., Liotta, Ortali, Patrignani



Planar 3-graphs in linear time

[D., Liotta, Ortali, Patrignani, SODA 2020]

Theorem 5. Let G be an n-vertex planar 3-graph. There exists an algorithm that computes a bend-minimum orthogonal drawing of G in O(n) time. Also, if G is not K_4, the drawing has at most one bend per edge



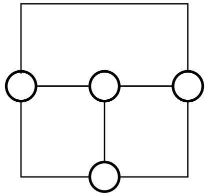
Planar 3-graphs in linear time

[D., Liotta, Ortali, Patrignani, SODA 2020]

Theorem 5. Let G be an n-vertex planar 3-graph. There exists an algorithm that computes a bend-minimum orthogonal drawing of G in O(n) time. Also, if G is not K_4, the drawing has at most one bend per edge

Optimal in terms of:

- time complexity
- total number of bends
- number of bends per edge



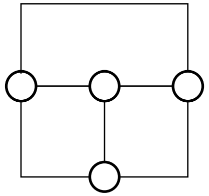
Planar 3-graphs in linear time

[D., Liotta, Ortali, Patrignani, SODA 2020]

Theorem 5. Let G be an n-vertex planar 3-graph. There exists an algorithm that computes a bend-minimum orthogonal drawing of G in O(n) time. Also, if G is not K_4, the drawing has at most one bend per edge

Main ingredients:

- constant number of “shapes” (related to spirality) for each type of node
- efficient processing of R-nodes without flow-based techniques
  - in particular, we can find in O(n) time the optimal solution for triconnected planar 3 graphs (over all choices of the external face)
- update of the optimal set of each type of node in O(1) time

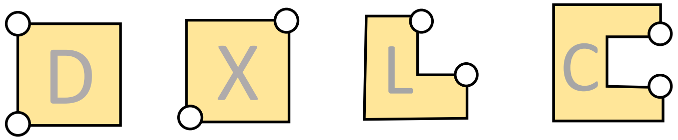


Constant number of “shapes”

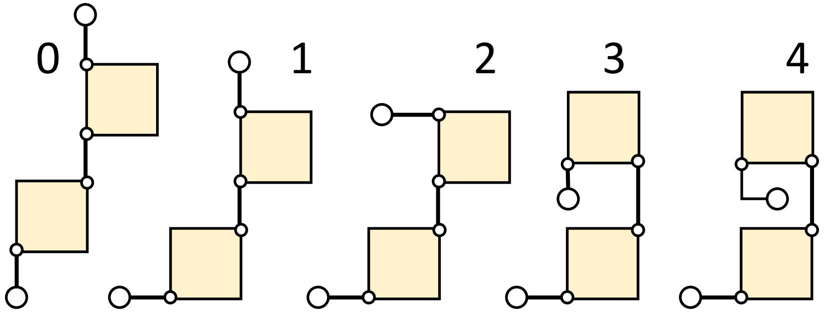
Shape-Lemma. Every biconnected planar 3-graph distinct from K_4 admits a <u>bend-minimum orthogonal representation</u> such that:

O1. every edge has at most one bend

O2. every P- and R-component is D-, X-, L-, or C-shaped



O3. every S-component has absolute spirality $k \leq 4$



O4. every component is optimal within its shape
(thanks to substitution)

Triconnected Graphs: Open Problems

Open Problem 5: Does there exist an o(nT(n)) algorithm for planar *triconnected* 4-graphs?

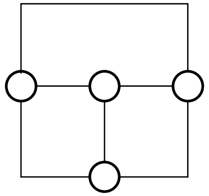
T(n) = time needed to solve a min-cost-flow problem

In particular: Can we find an $O(n^2)$-time algorithm (or better)?

Recall: For triconnected planar 3-graphs we can solve the problem in O(n) time [D., Liotta, Patrignani, SODA 2020]

FPT Algorithm for planar 4-graphs

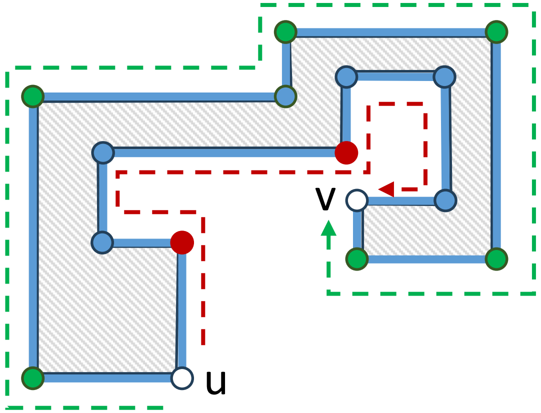
Theorem 6. Let G be an n-vertex planar 4-graph with k vertices of degree at most two, let b be a non-negative integer, and let $p = b+k$. There exists an $O(2^{p \log p})n^{O(1)}$–time algorithm that tests whether G admits an orthogonal representation with at most b bends, and that computes one if it exists

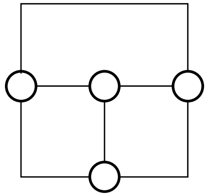


Bounding the spirality

Lemma. For any node μ of a rooted SPQ*R-tree T of G, the absolute value of the spirality of any orthogonal representation of G_μ is at most p+2

Intuition: right (left) turns on the external left (right) path must be either bends or degree-2 vertices



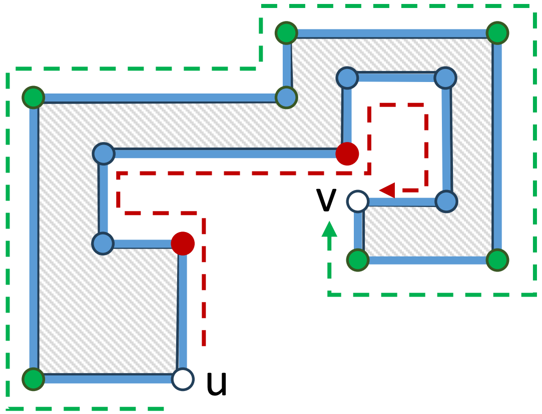


Bounding the spirality: Q*-, S-, P-nodes

Lemma. For any node μ of a rooted SPQ*R-tree T of G, the absolute value of the spirality of any orthogonal representation of G_μ is at most p+2

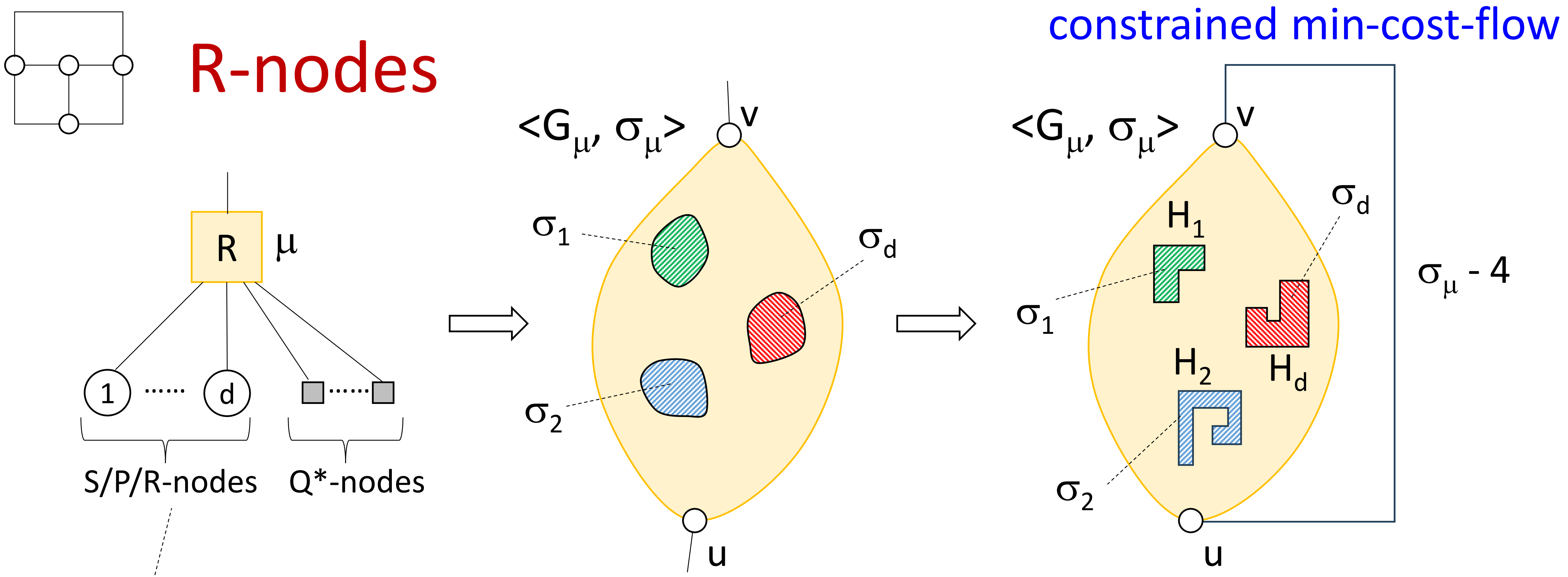
Intuition: right (left) turns on the external left (right) path must be either bends or degree-2 vertices

⇓

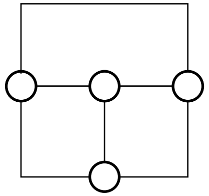


- The optimal set of a Q*-node is computed in O(p) time
- The optimal set of an S-node is computed in O(p^2) time from those of its children
- The optimal set of a P-node is computed in O(p) time from those of its children

R-nodes



$d \leq p$ $\Rightarrow$ $O(p^p)=O(2^{p \log p})$ combinations of target spirality values

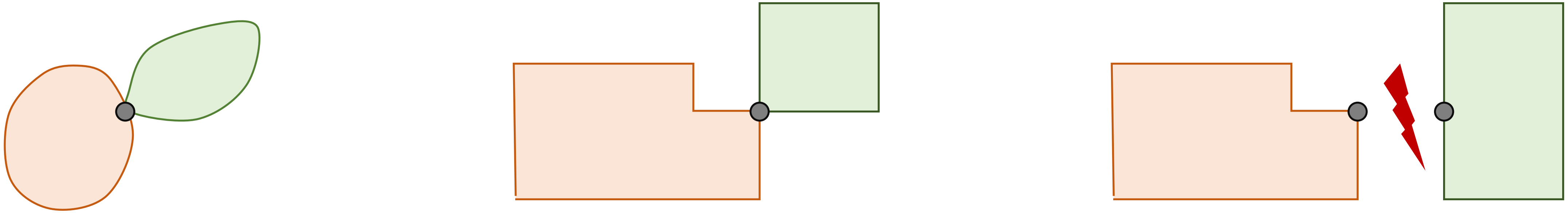


Extension to non-biconnected graphs

The presented results can be extended to 1-connected graphs

Main ingredients:

- use the block-cutvertex tree of the graph
- angle constraints at the cutvertices (representations that share a cutvertex must be glued together)

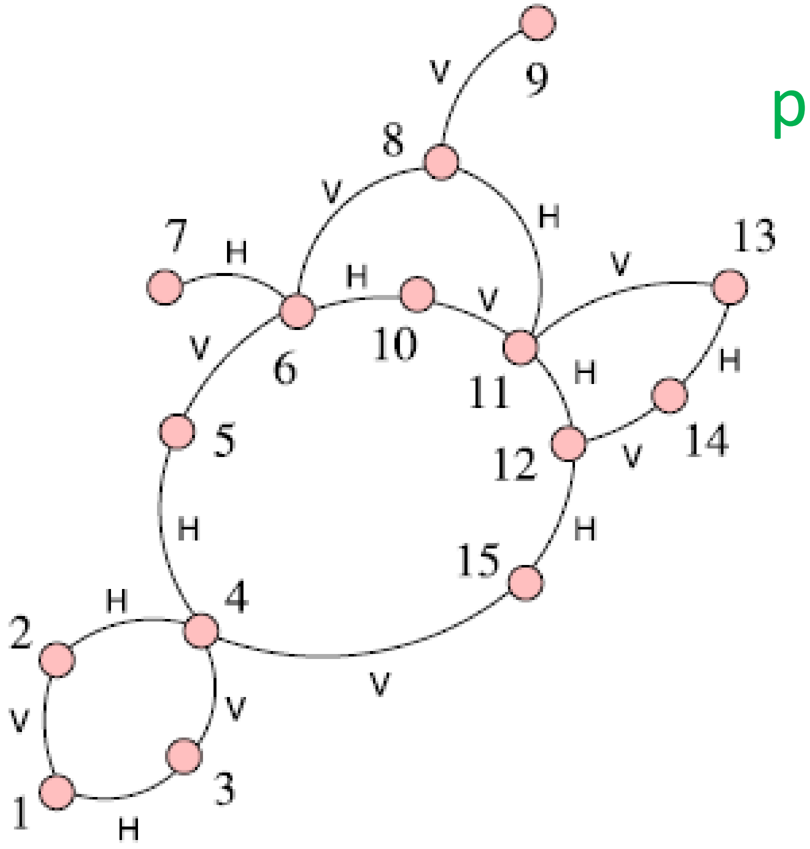


Concluding remarks

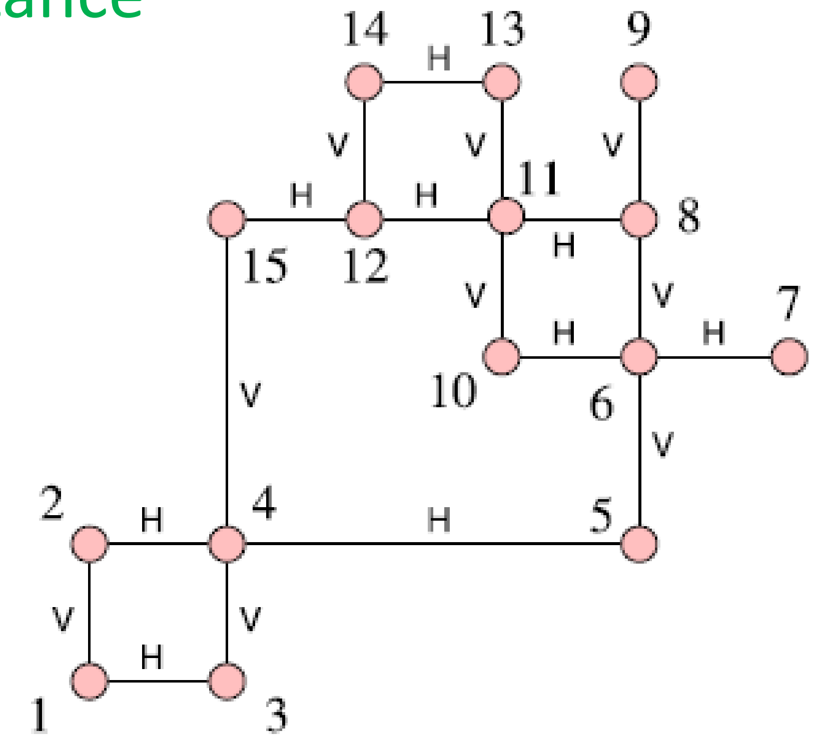
constrained scenarios and some more problems

Constrained scenarios: HV-drawings

- **Problem** HV-PLANARITYTESTING. Rectilinear planarity testing where each edge is assigned a “direction”, i. e., horizontal (H) or vertical (V)



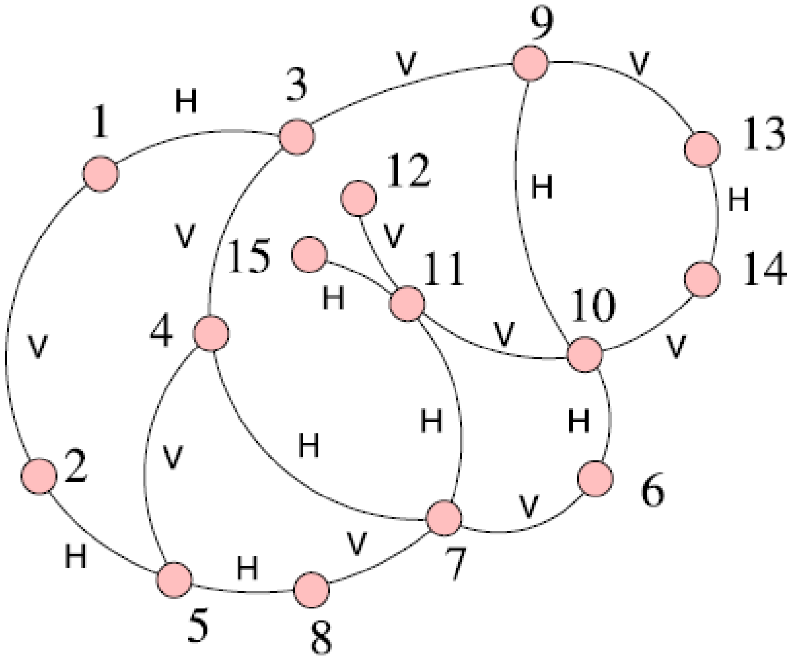
positive instance

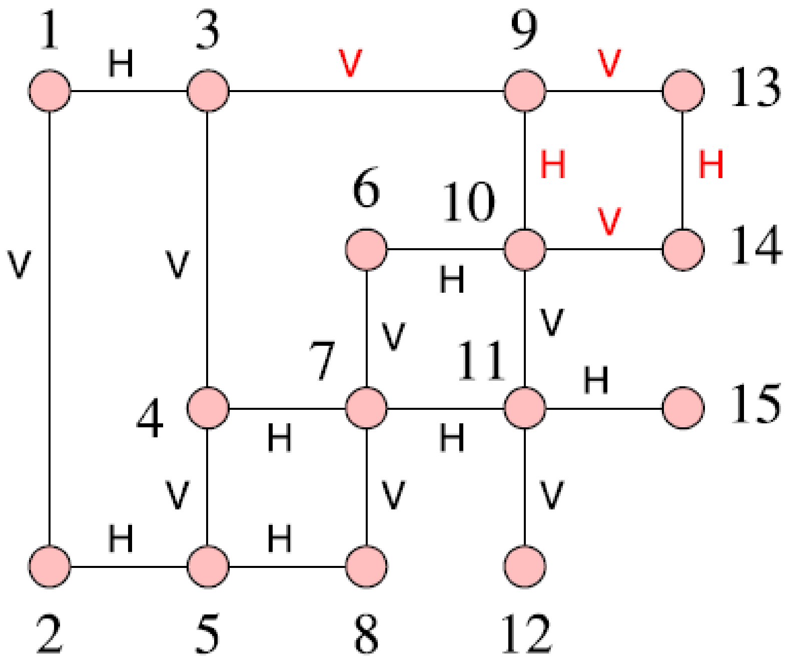


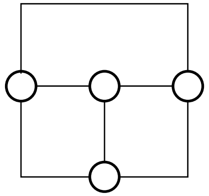
Constrained scenarios: HV-drawings

- **Problem** HV-PLANARITYTESTING. Rectilinear planarity testing where each edge is assigned a "direction", i. e., horizontal (H) or vertical (V)

negative instance

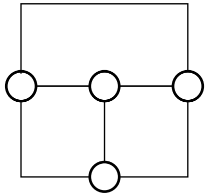






Constrained scenarios: HV-drawings

- HV-PLANARITYTESTING is NP-complete even for planar 3-graphs [D., Liotta, Patrignani – JCSS 2019]
- HV-PLANARITYTESTING is polynomial-time solvable for SP-graphs
  - $O(n^4)$-time for SP-graphs [D., Liotta, Patrignani – JCSS 2019]
    - exploiting SPQ*-trees and spirality
  - $O(n^2)$-time for SP-graphs
    - improving the time required by S-nodes as in [D., Kaufmann, Liotta, Ortali – JGAA 2023]



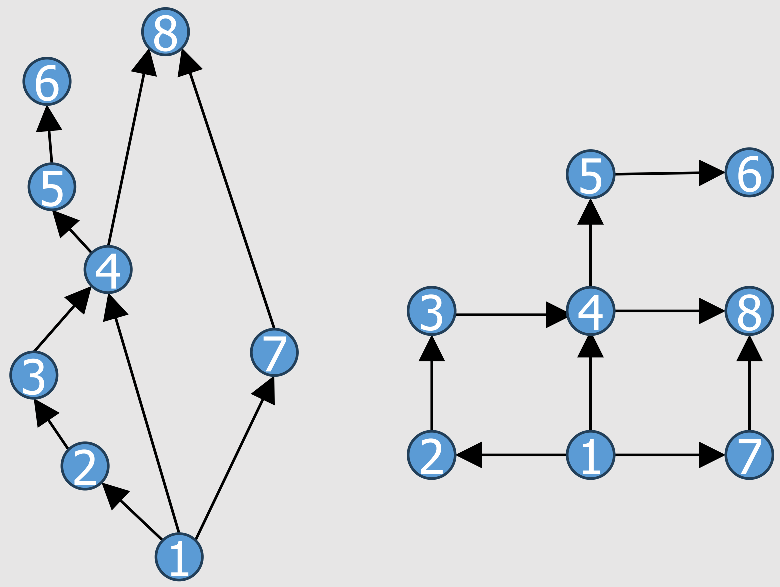
Constrained scenarios: HV-drawings

- HV-PLANARITYTESTING is NP-complete even for planar 3-graphs
  [D., Liotta, Patrignani – JCSS 2019]
- HV-PLANARITYTESTING is polynomial-time solvable for SP-graphs
  - $O(n^4)$-time for SP-graphs [D., Liotta, Patrignani – JCSS 2019]
    - exploiting SPQ*-trees and spirality
  - $O(n^2)$-time for SP-graphs
    - improving the time required by S-nodes as in
      [D., Kaufmann, Liotta, Ortali – JGAA 2023]

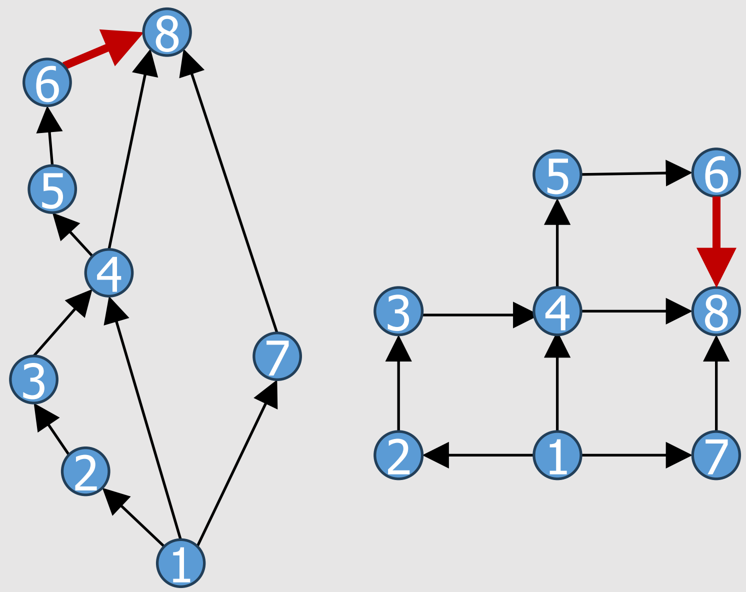
Open Problem 6: Study the parametrized complexity of HV-PLANARITYTESTING for planar 4-graphs (with rigid components)

Constrained scenarios: Rectilinear-Upward

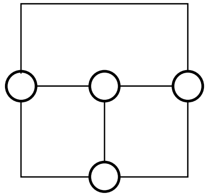
- **Problem** RU-PLANARITYTESTING. Rectilinear planarity testing where each edge cannot point downward



positive instance



negative instance

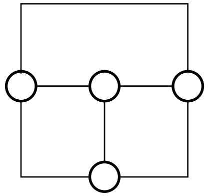


Constrained scenarios: Rectilinear-Upward

[D., Kaufmann, Liotta, Ortali, Patrignani – ISAAC 2023 + advances]

- RU-PlanarityTesting is NP-complete
- RU-PlanarityTesting is O(n)-time solvable for <u>*upward-plane*</u> digraphs
  – based on a 2-SAT formulation
- RU-PlanarityTesting is $O(n^2)$-time solvable for <u>*biconnected*</u> SP-digraphs
  – based on spirality + SPQ-trees
- RU-PlanarityTesting is FPT, parameterized by k= # of switches (sources/sinks)
  – $O(2^{k \log k + 2k} n)$-time algorithm, based on spirality + SPQR-trees + 2-SAT

Open Problem 7: Can we solve RU-PlanarityTesting in polynomial time for any *plane* digraph

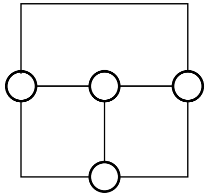


Constrained scenarios: Rectilinear-Upward

[D., Kaufmann, Liotta, Ortali, Patrignani – ISAAC 2023 + advances]

- RU-PLANARITYTESTING is NP-complete
- RU-PLANARITYTESTING is O(n)-time solvable for ***upward-plane*** digraphs
  - based on a 2-SAT formulation
- RU-PLANARITYTESTING is O(n^2)-time solvable for ***biconnected*** SP-digraphs
  - based on spirality + SPQ-trees
- RU-PLANARITYTESTING is FPT, parameterized by k= # of switches (sources/sinks)
  - O($2^{k \log k+2k}$ n)-time algorithm, based on spirality + SPQR-trees + 2-SAT

Open Problem 8: Can we solve RU-PLANARITYTESTING in O(n^2) time for any directed *partial 2-tree ("1-connected" SP-digraphs)*



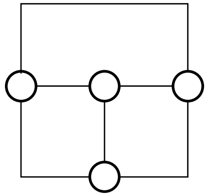
Constrained scenarios: Rectilinear-Upward

[D., Kaufmann, Liotta, Ortali, Patrignani – ISAAC 2023 + advances]

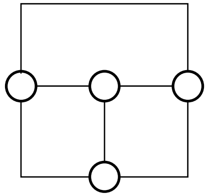
- RU-PlanarityTesting is NP-complete
- RU-PlanarityTesting is $O(n)$-time solvable for ***upward-plane*** digraphs
  - based on a 2-SAT formulation
- RU-PlanarityTesting is $O(n^2)$-time solvable for ***biconnected*** SP-digraphs
  - based on spirality + SPQ-trees
- RU-PlanarityTesting is FPT, parameterized by k= # of switches (sources/sinks)
  - $O(2^{k \log k+2k}\, n)$-time algorithm, based on spirality + SPQR-trees + 2-SAT

Open Problem 9: Can we find FPT/XP algorithms with respect to other parameters (i.e., other than the number of switches)?

Thank you for your attention!



Additional details

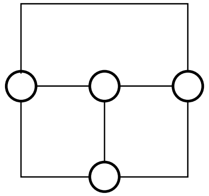


Fixed Embedding: State-of-the-art

Improvements of Tamassia's result derive from subsequent faster min-cost flow algorithms:

- $O(n^{1.75} \log n)$ time [Garg & Tamassia, GD 1996]
- $O(n^{1.5})$ time [Cornelsen & Karrenbauer, JGAA 2012]

Remark: for rectilinear planarity testing the min-cost flow problem is reduced to a max-flow problem (faces cannot exchange flow); $O(n \log^3 n)$ time [Borradaile, Klein, Mozes, Nussbaum, Wulff- Nilsen, SIAM J. Comp. 2017]



Fixed Embedding: State-of-the-art

Improvements of Tamassia's result derive from subsequent faster min-cost flow algorithms:

- $O(n^{1.75} \log n)$ time [Garg & Tamassia, GD 1996]
- $O(n^{1.5})$ time [Cornelsen & Karrenbauer, JGAA 2012]

Remark: for bend minimization we could use in principle a recent "almost-linear time" min-cost-flow algorithm - $O(n^{1+o(1)} \log n)$ time
[Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva, FOCS 2022, Comm. ACM 2023]

"For now, the new algorithms introduced by Prof. Kyng, Dr. Probst Gutenberg, and their co-authors remain impractical, as they rely on a theoretical analysis of algorithm performance on networks larger than anything even giant corporations like Google would ever consider. But, the race is now on to simplify and improve the algorithm to make it work well in practice."