Algorithms for Graph Visualization

Flow Methods: Compaction and Upward Planarity

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Tamara Mchedlidze

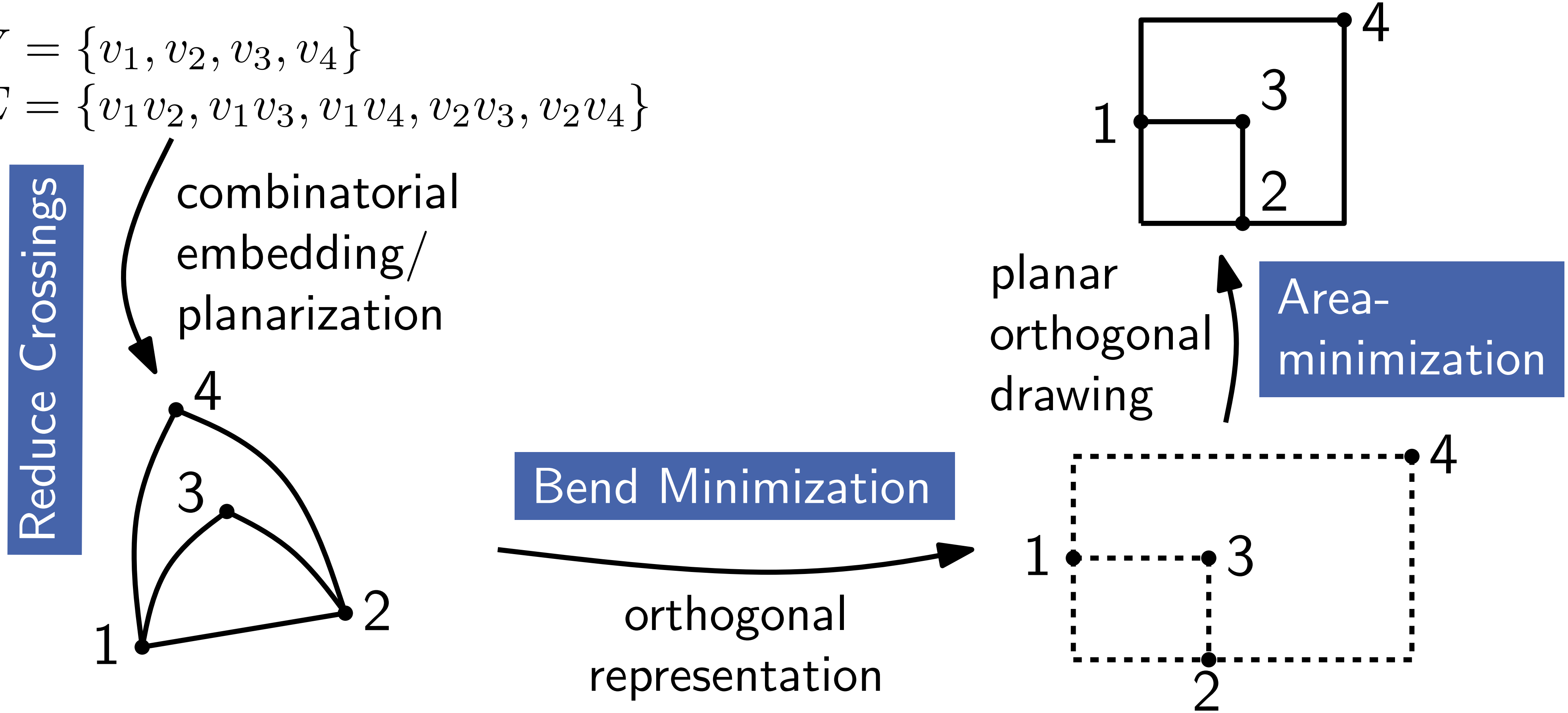
23.01.2017

(Planar) Orthogonal Layout

Three-step approach: *Topology – Shape – Metrics*

[Tamassia SIAM J. Comput. 1987]

$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4\}$

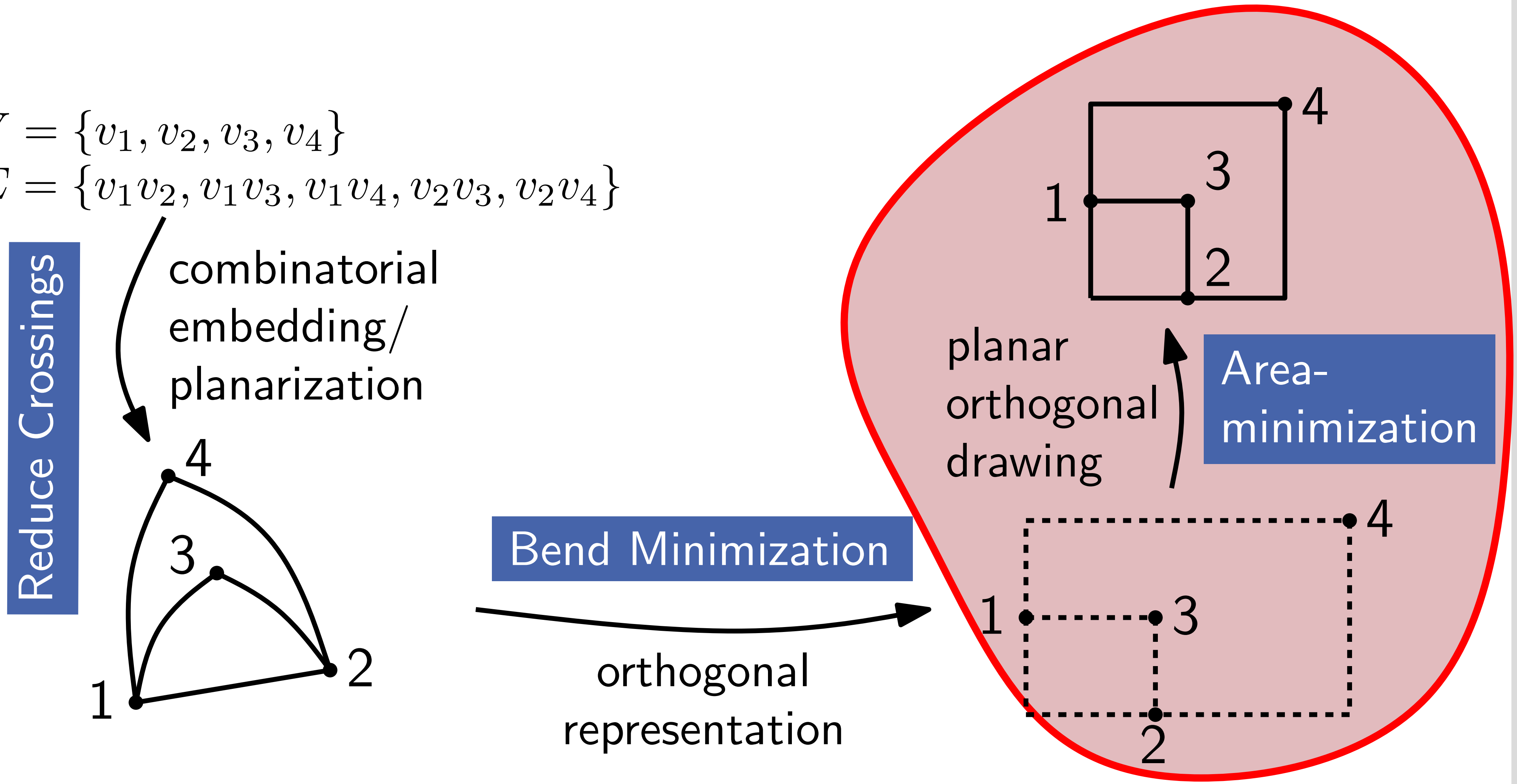


(Planar) Orthogonal Layout

Three-step approach: *Topology – Shape – Metrics*

[Tamassia SIAM J. Comput. 1987]

$V = \{v_1, v_2, v_3, v_4\}$
$E = \{v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4\}$



Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Compaction

Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

Find: compact orthogonal layout of G that realizes $H(G)$

Special Case: all faces are rectangles

$\rightarrow$ Guaratees possible
- minimum total edge length
- minimum area

Compaction

Problem Compaction

Given:
- planar graph $G = (V, E)$ with maximum degree 4
- orthogonal representation $H(G)$

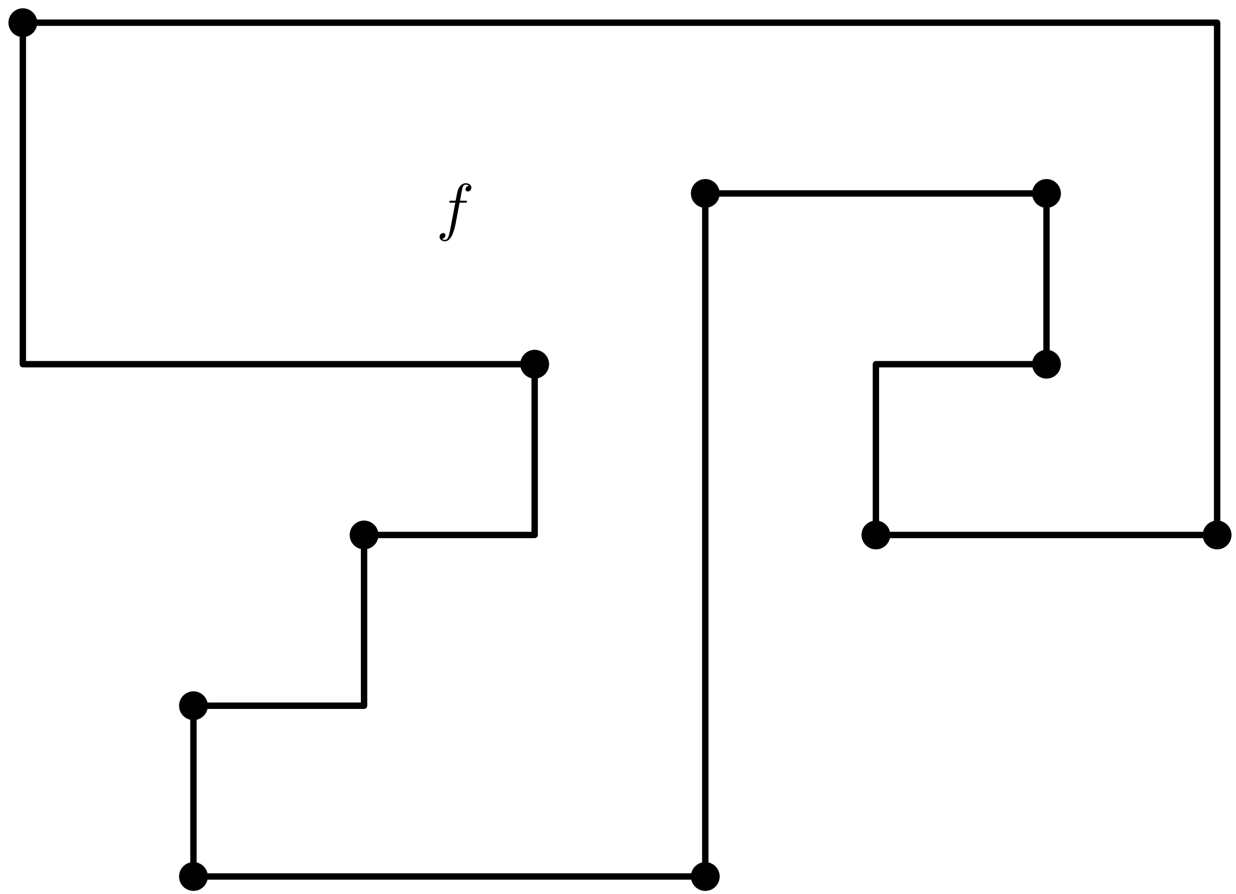
Find: compact orthogonal layout of G that realizes $H(G)$

Special Case: all faces are rectangles

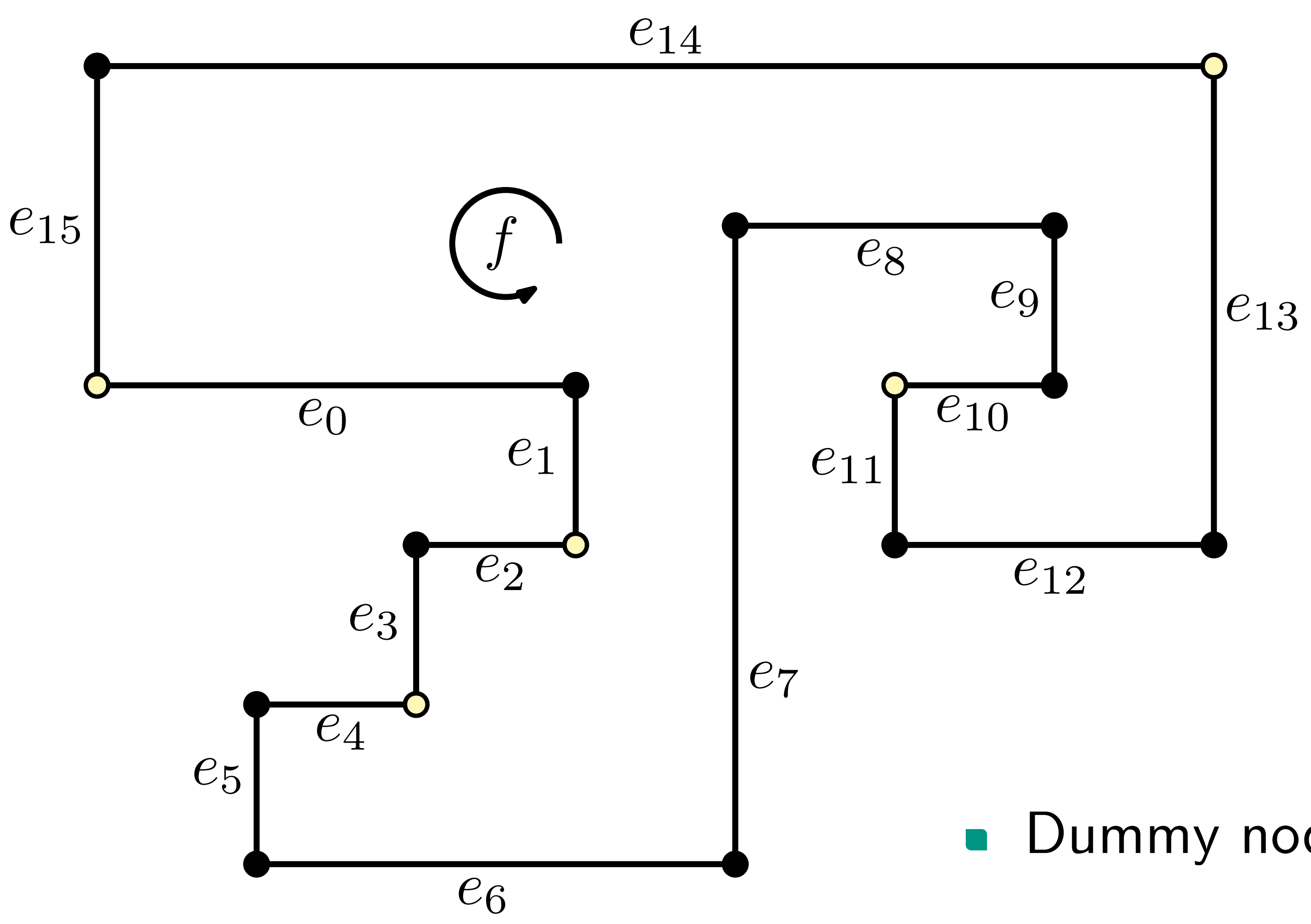
$\rightarrow$ Guaratees possible
- minimum total edge length
- minimum area

What is the faces are not rectangles?

Refinement of (G, H) – Innner Face

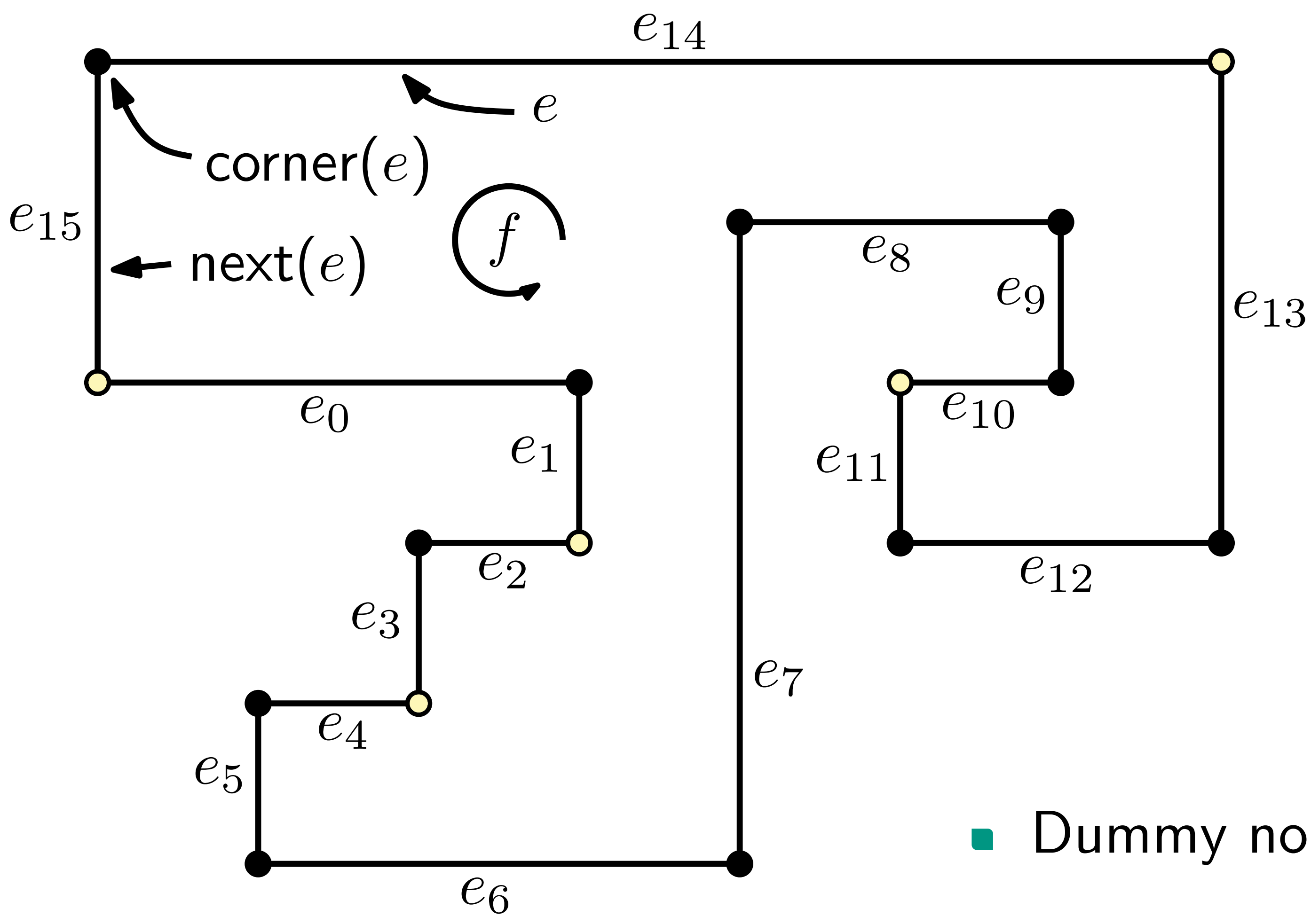


Refinement of (G, H) – Innner Face



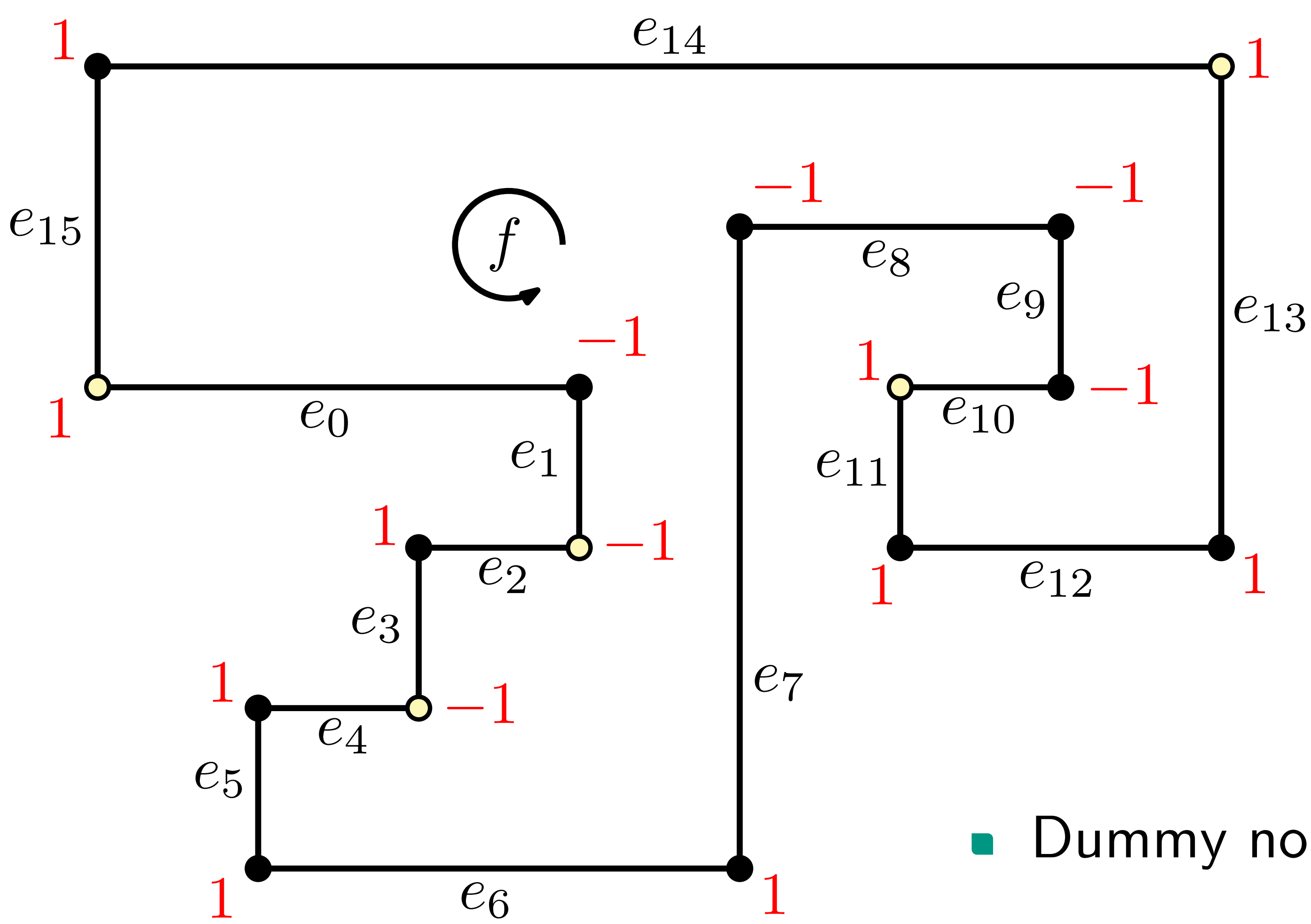
- Dummy nodes for bends

Refinement of (G, H) – Innner Face



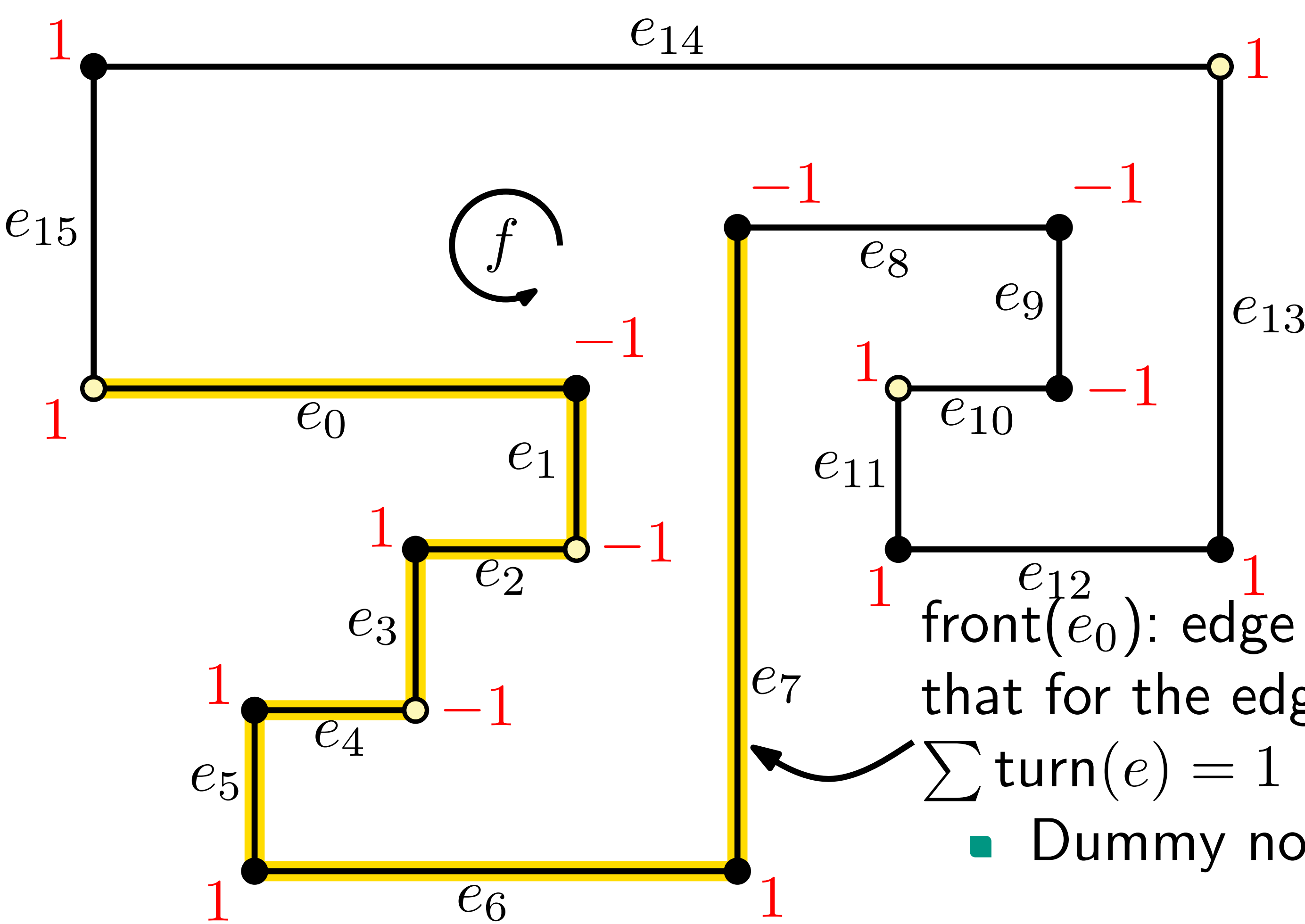
- Dummy nodes for bends

Refinement of (G, H) – Innner Face



- Dummy nodes for bends
- $\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

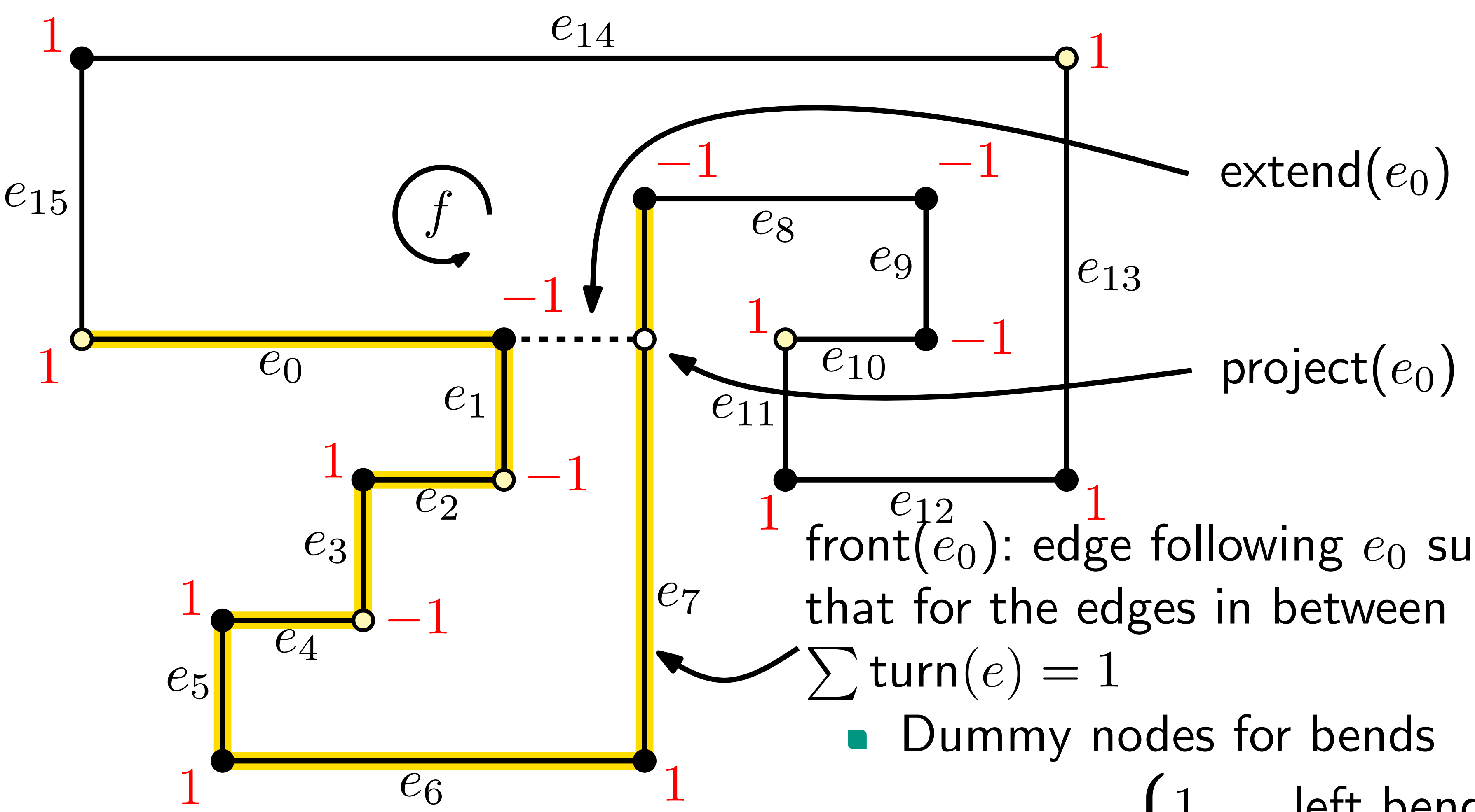
Refinement of (G, H) – Innner Face



front(e_0): edge following e_0 such that for the edges in between $\sum \text{turn}(e) = 1$

- Dummy nodes for bends
- $\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

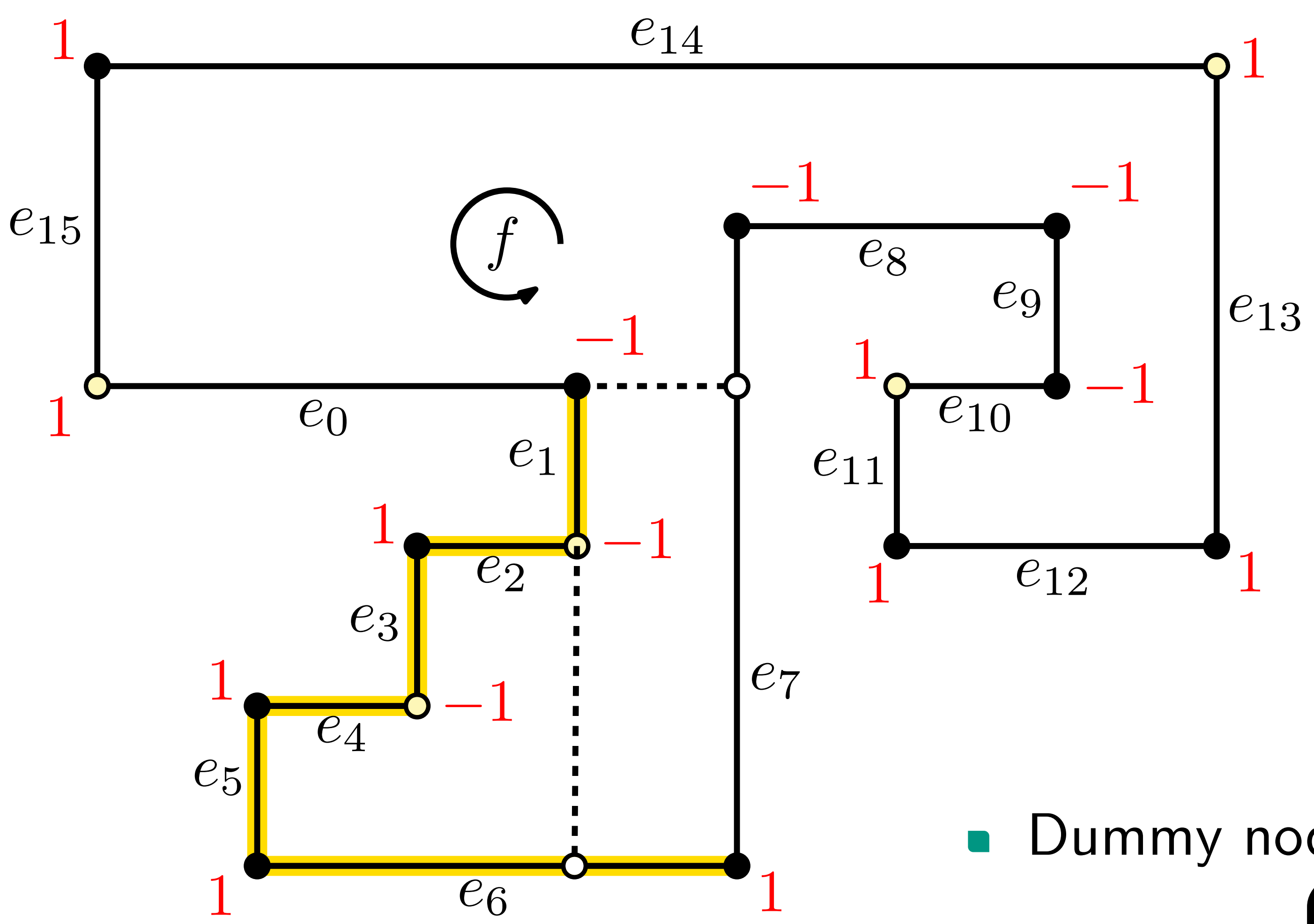
Refinement of (G, H) – Innner Face



front(e_0): edge following e_0 such that for the edges in between $\sum$ turn$(e) = 1$

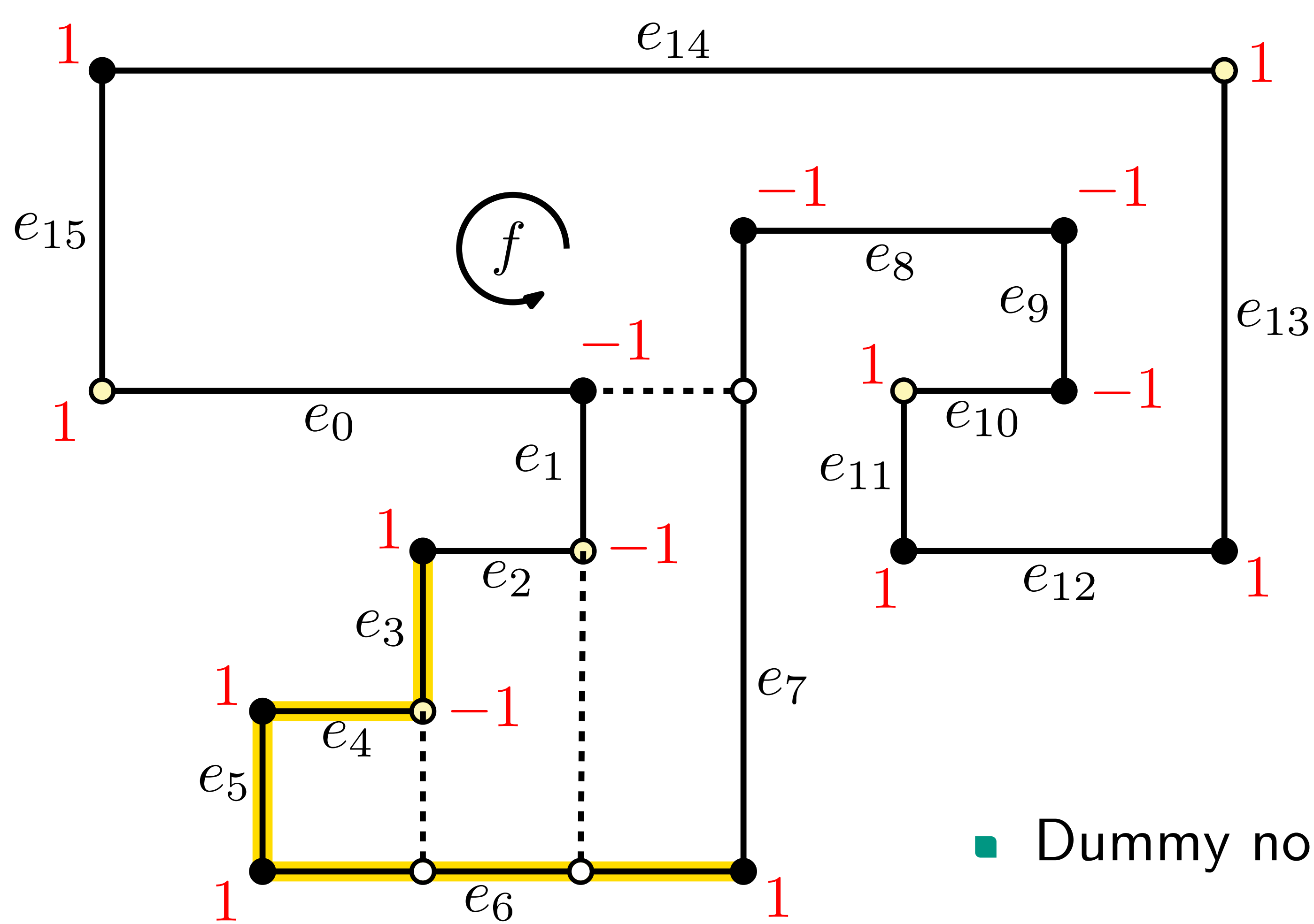
- Dummy nodes for bends
- turn$(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



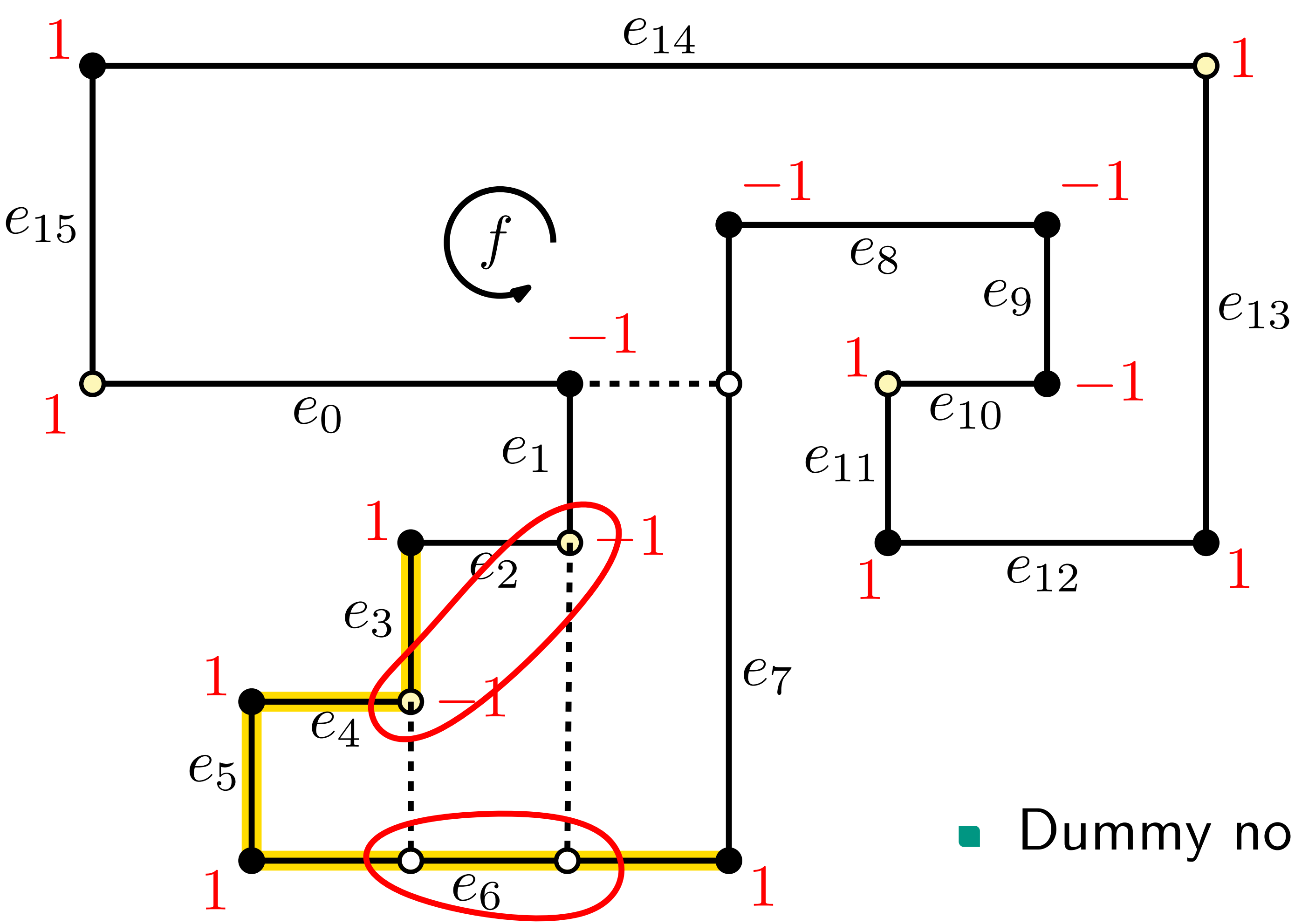
- Dummy nodes for bends
- turn$(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



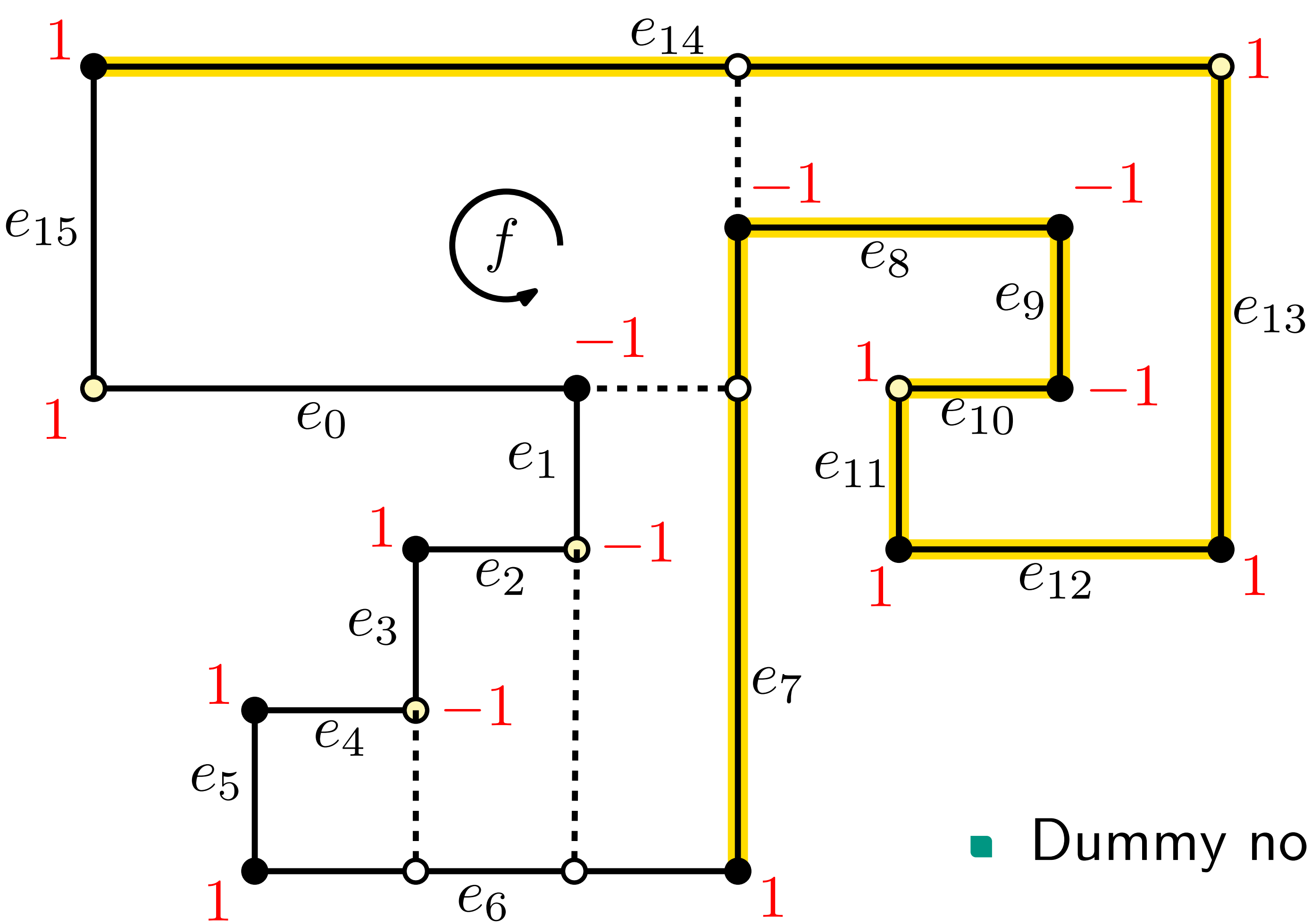
- Dummy nodes for bends
- turn$(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



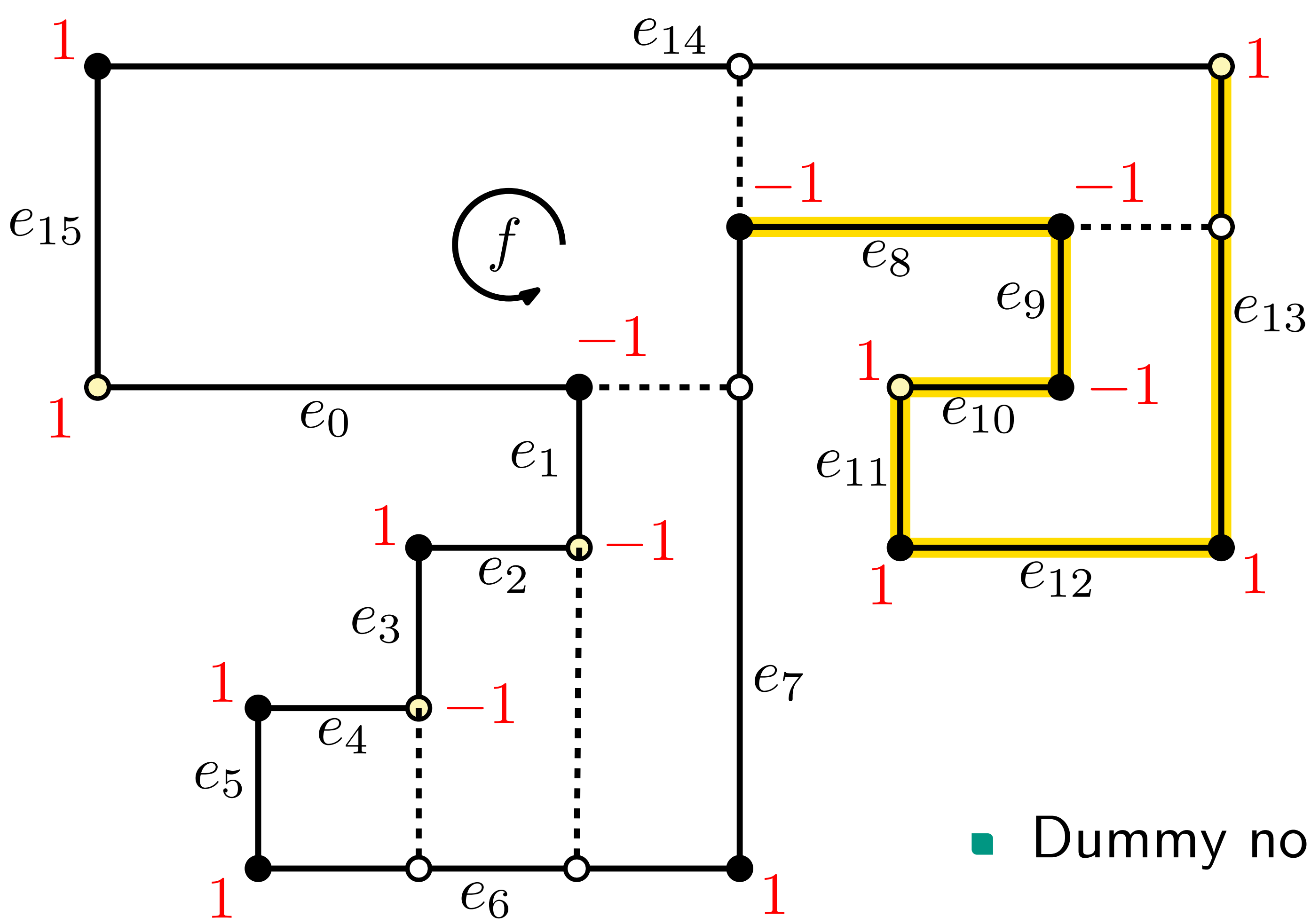
- Dummy nodes for bends
- $\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



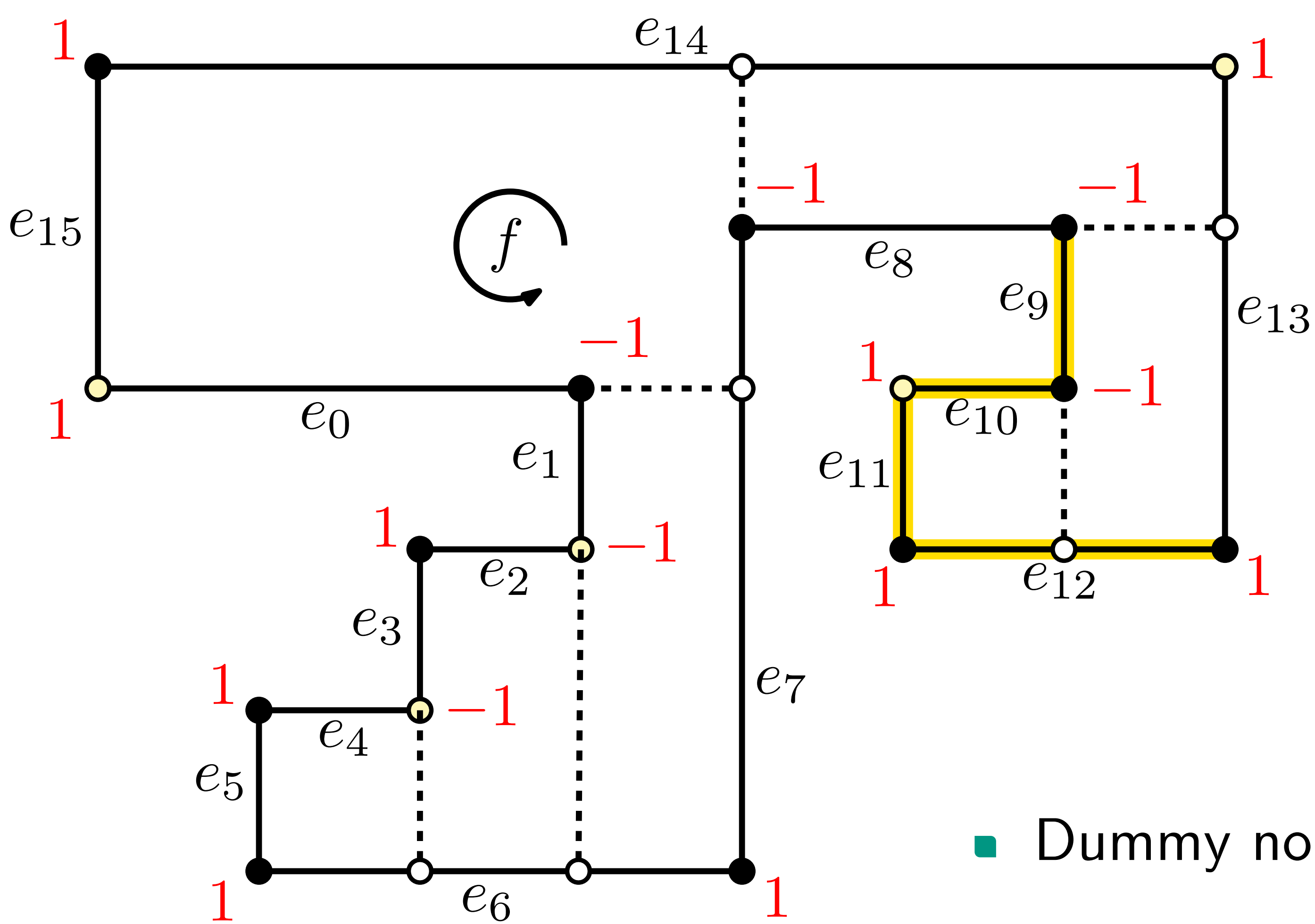
- Dummy nodes for bends
- turn$(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



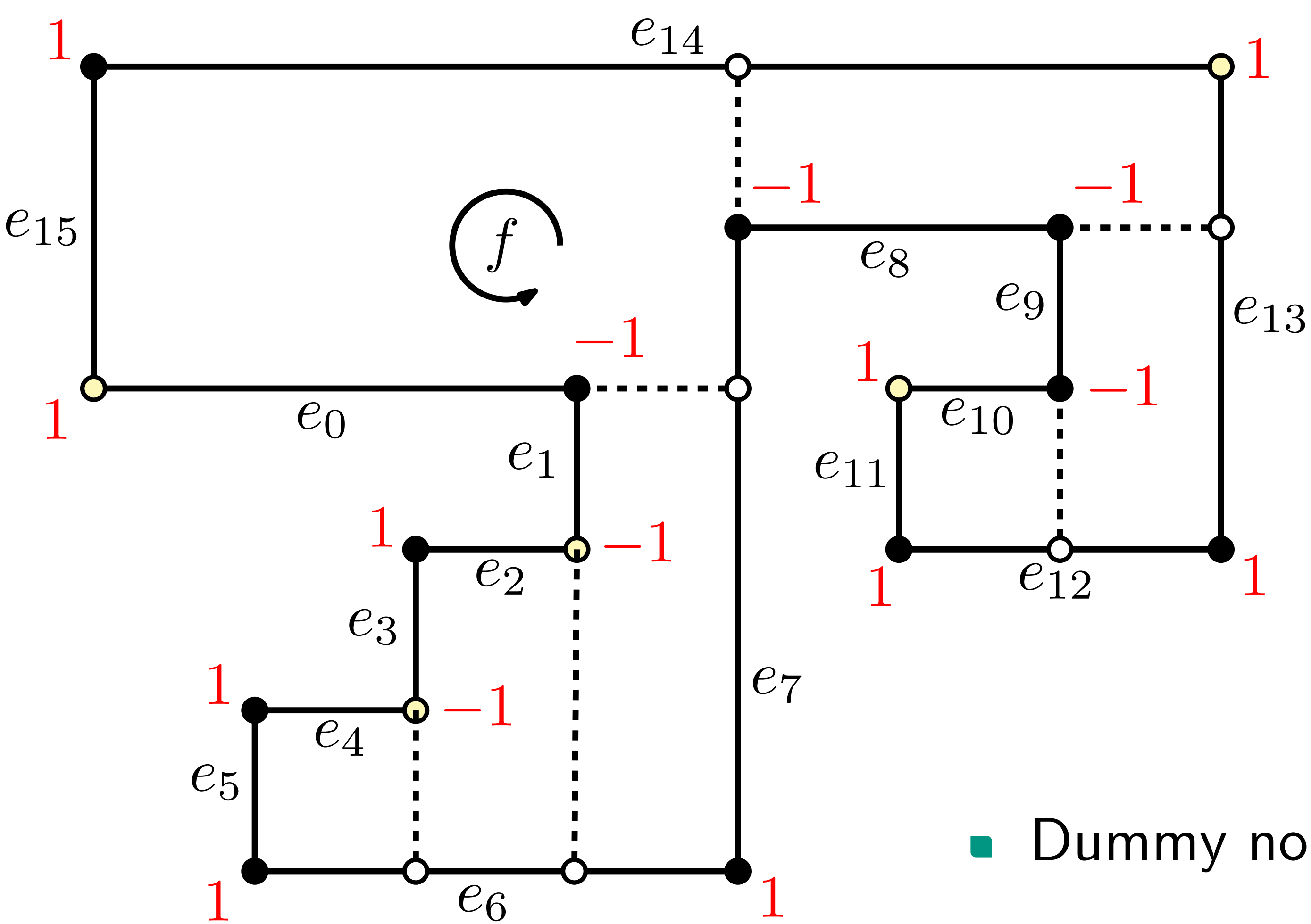
- Dummy nodes for bends
- $\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face



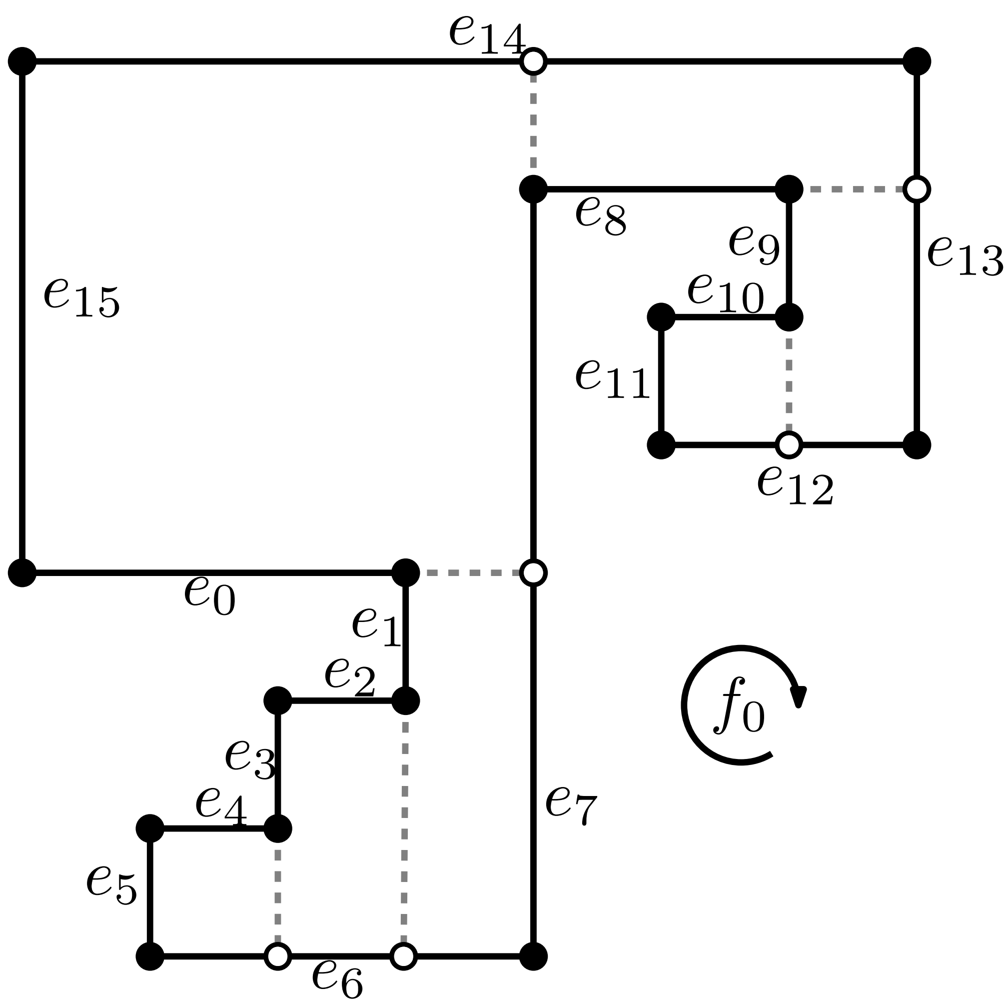
- Dummy nodes for bends
- turn$(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Innner Face

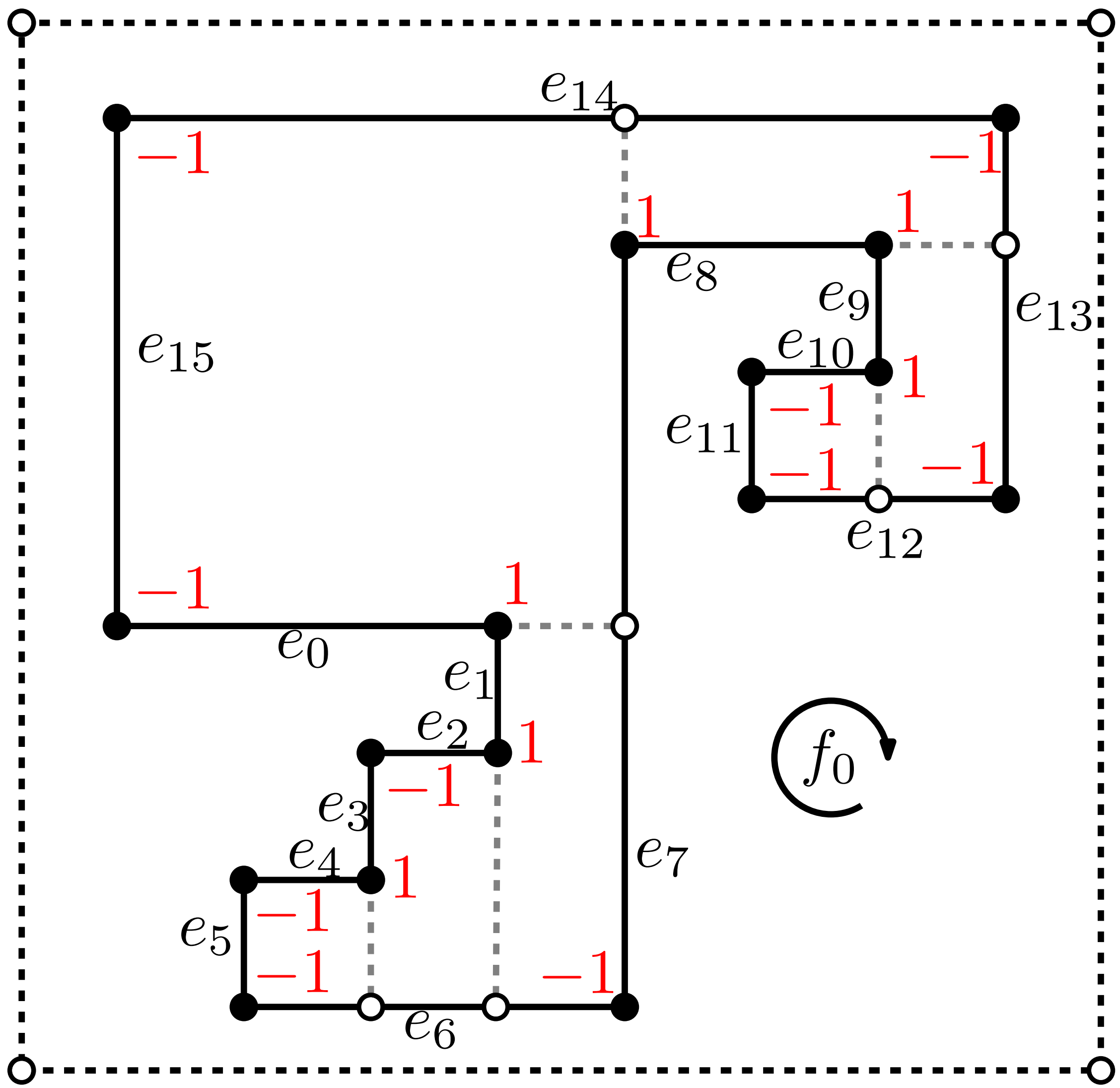


- Dummy nodes for bends
- $\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$

Refinement of (G, H) – Outer Face



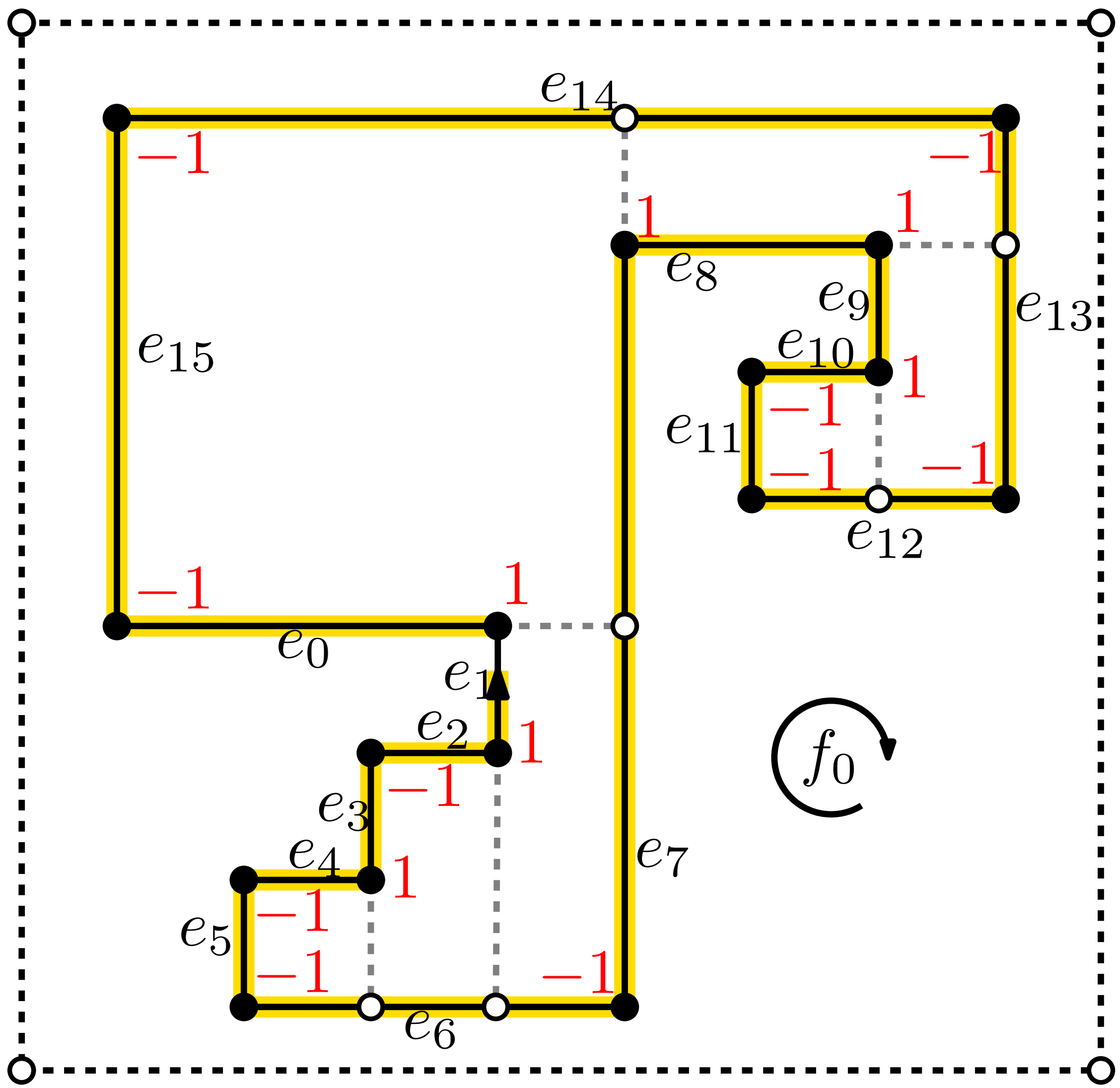
Refinement of (G, H) – Outer Face



- front(e) may be undefined

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

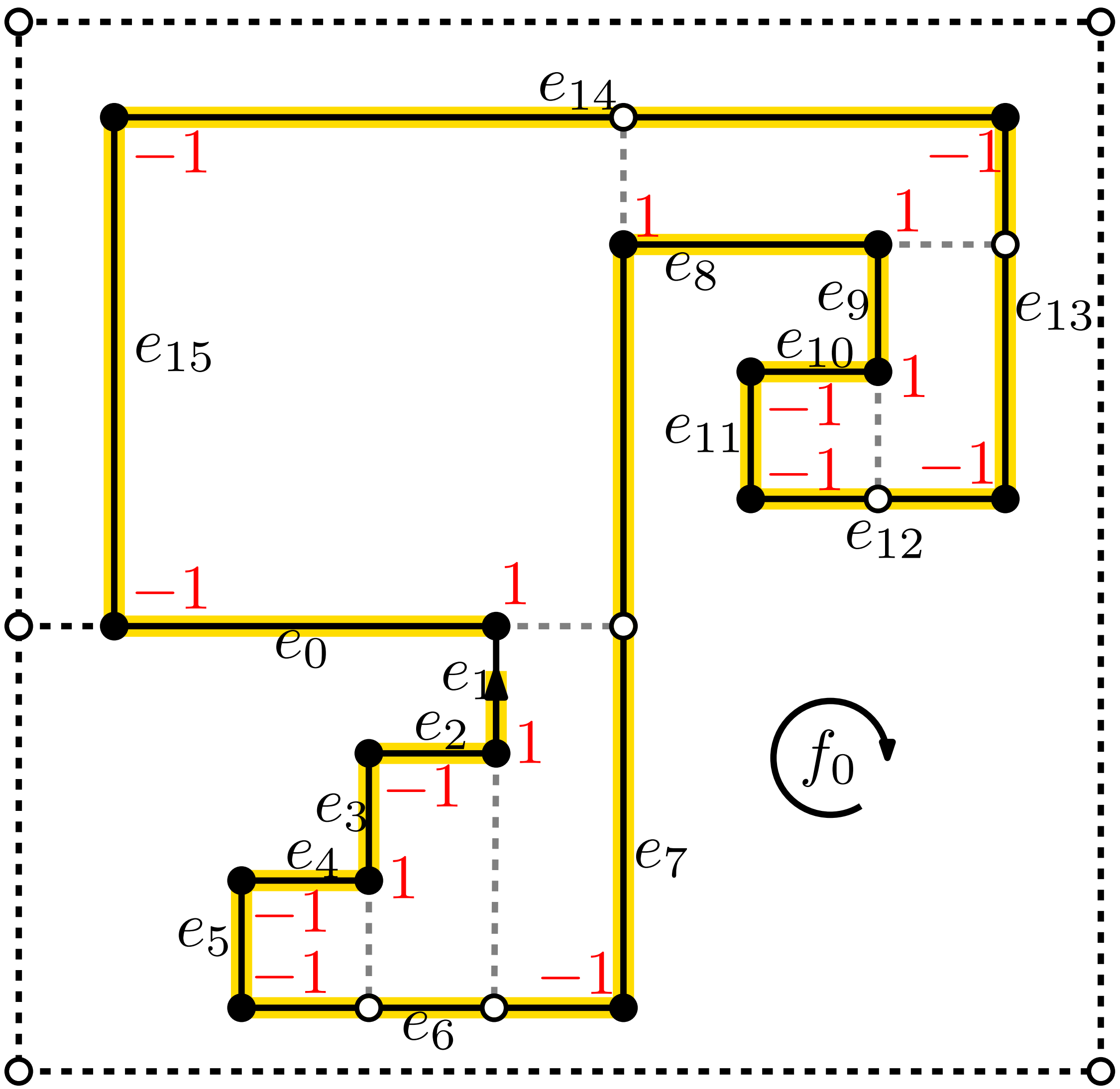
Refinement of (G, H) – Outer Face



- front(e) may be undefined

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

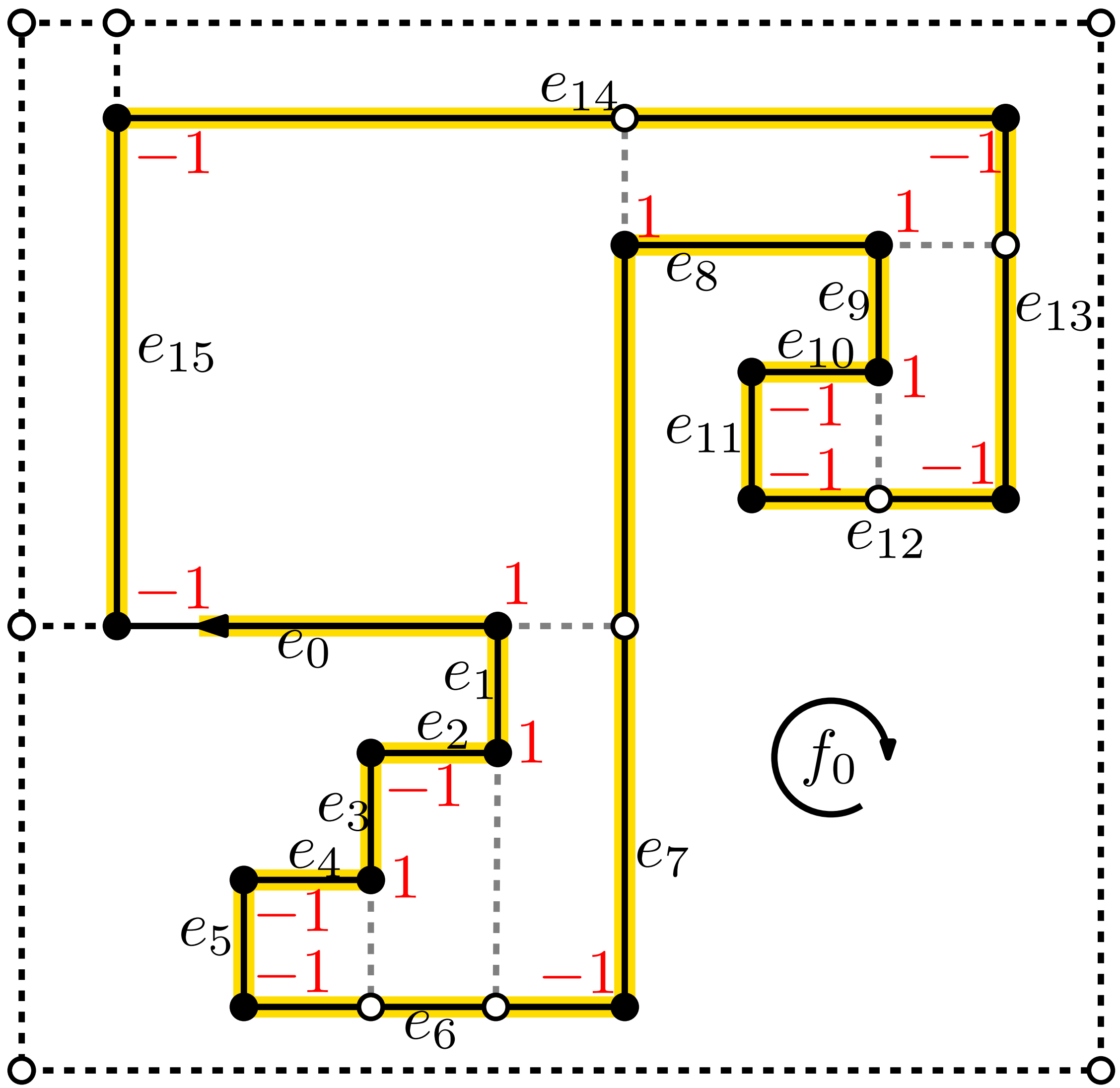
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn$(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

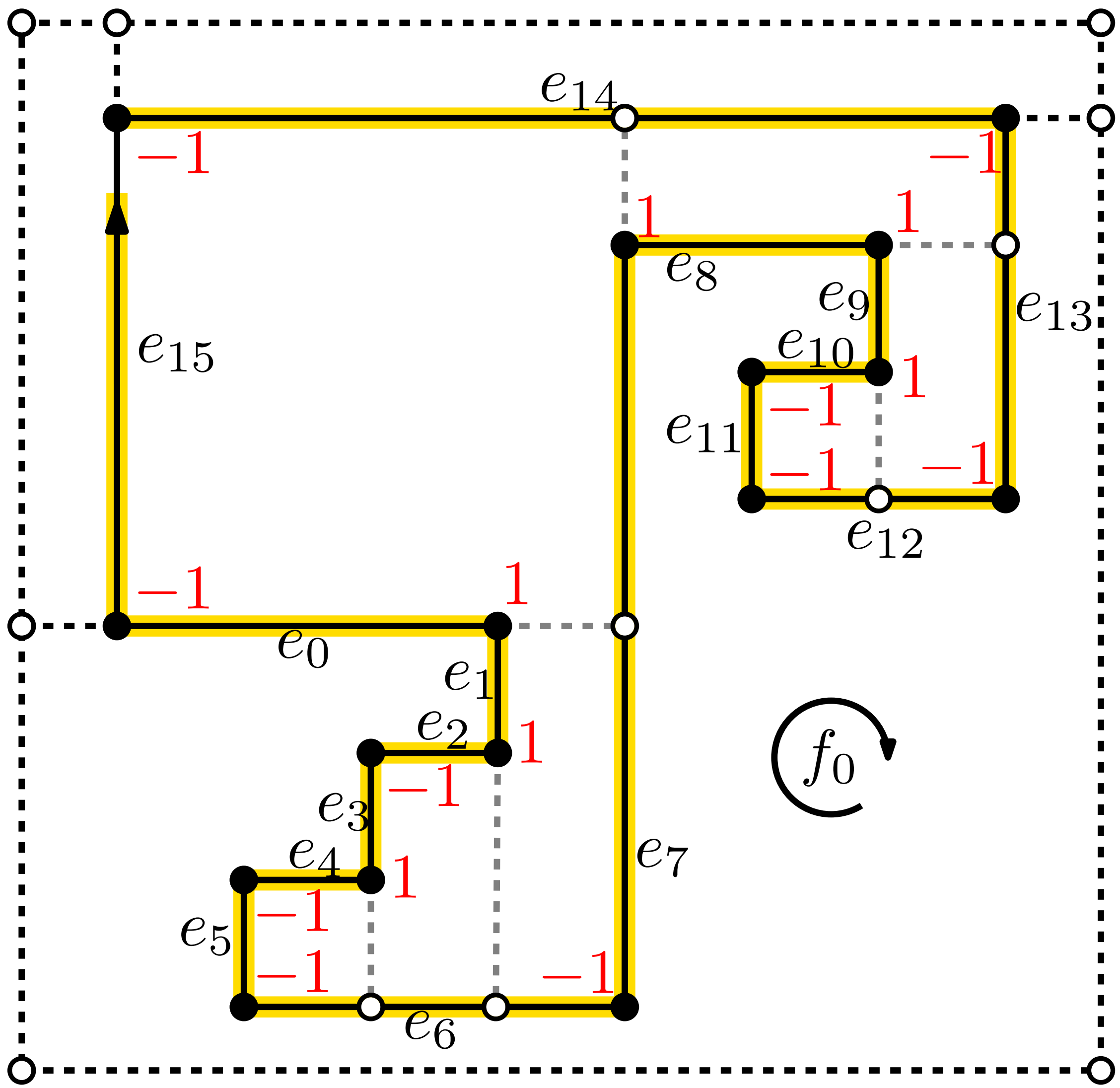
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum \text{turn}(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

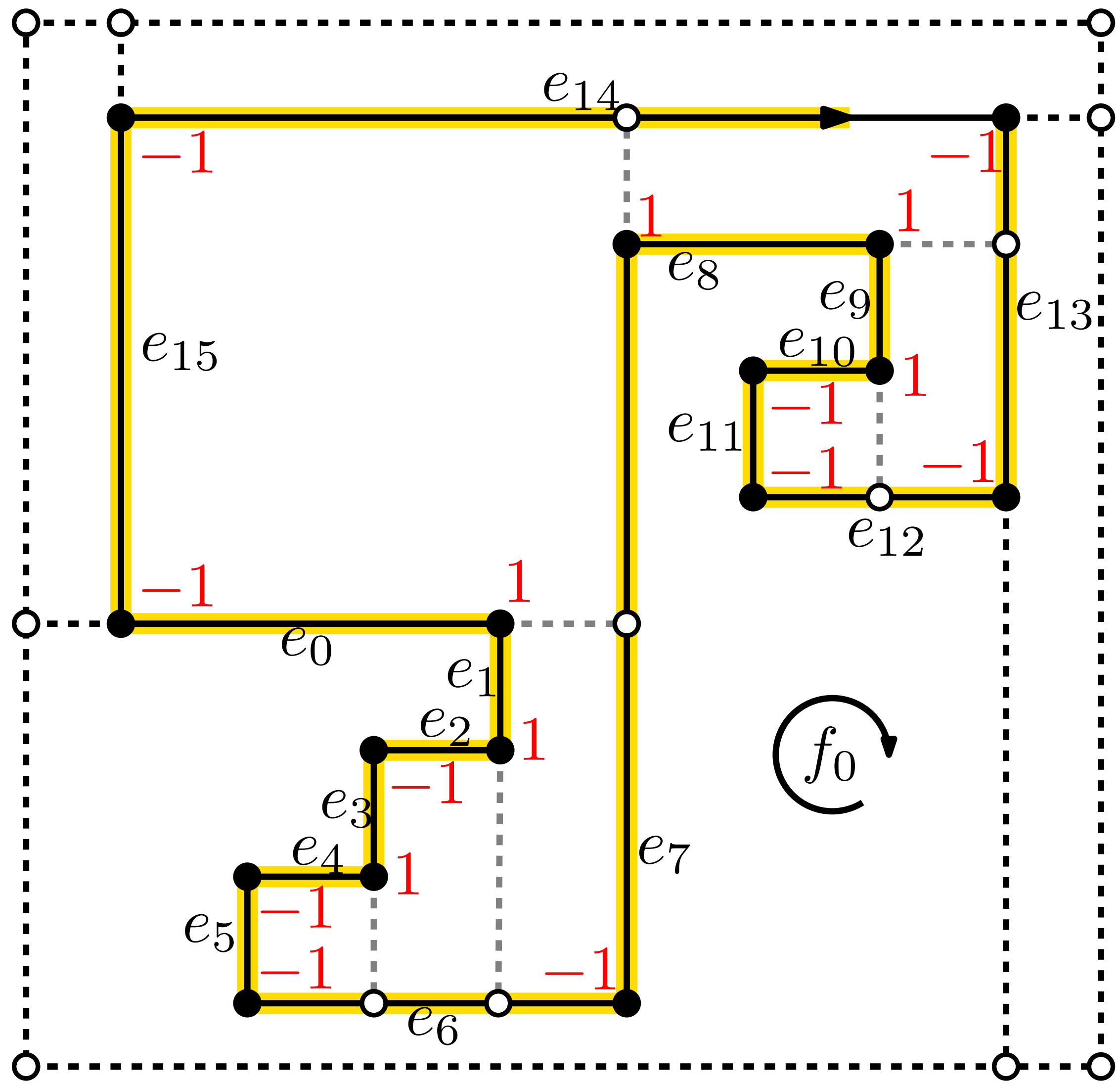
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn$(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

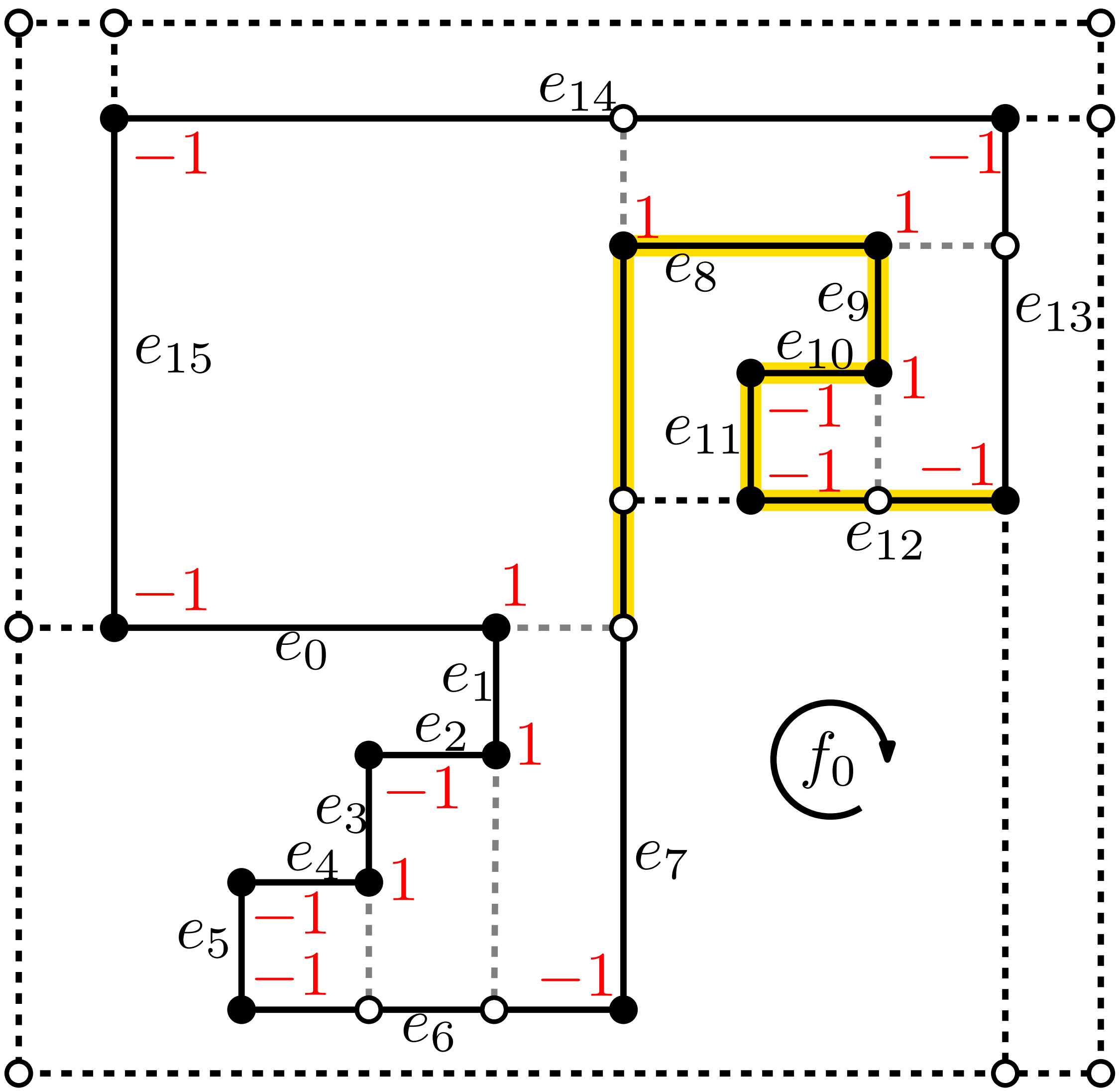
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum \text{turn}(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

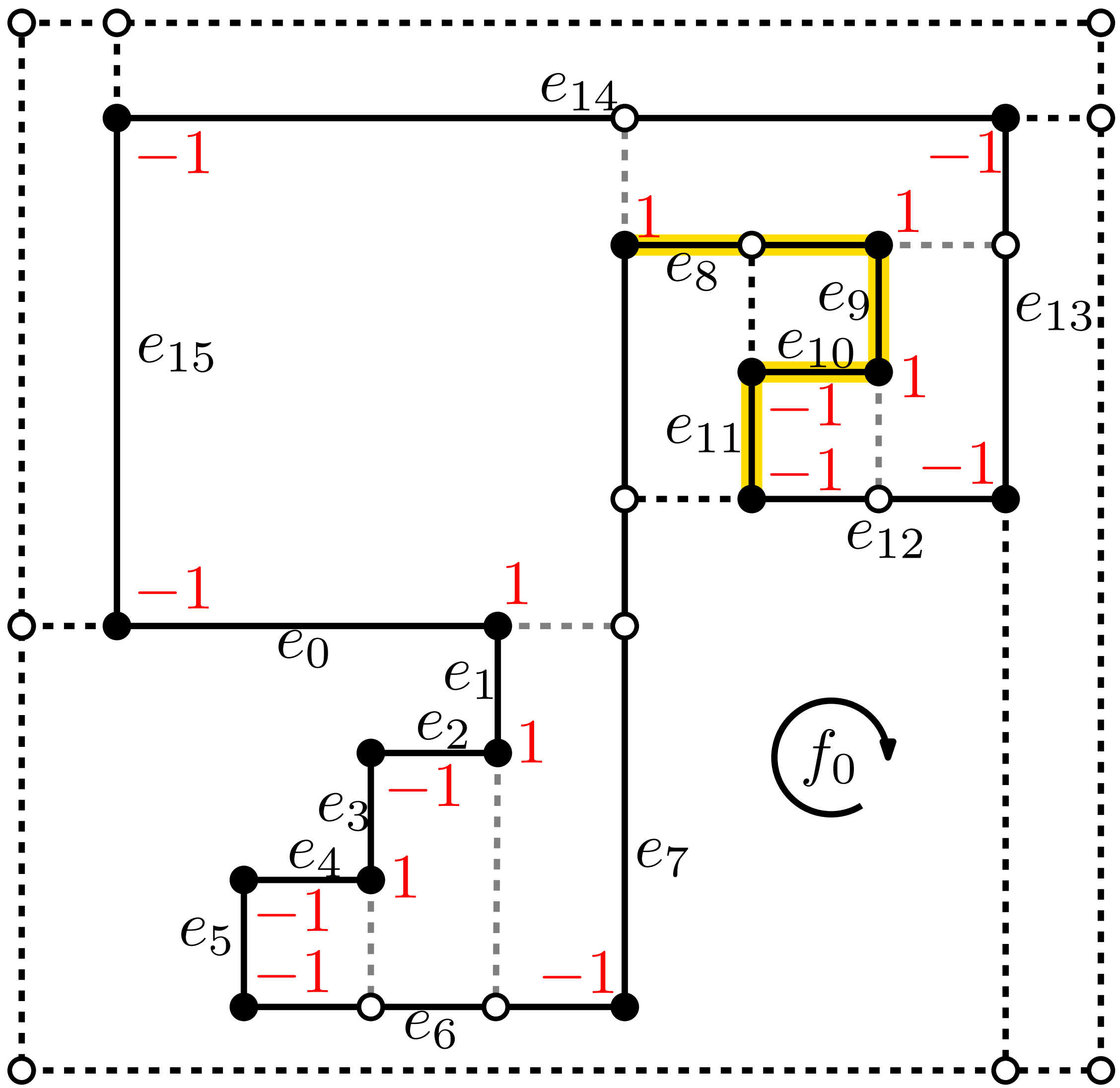
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn$(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

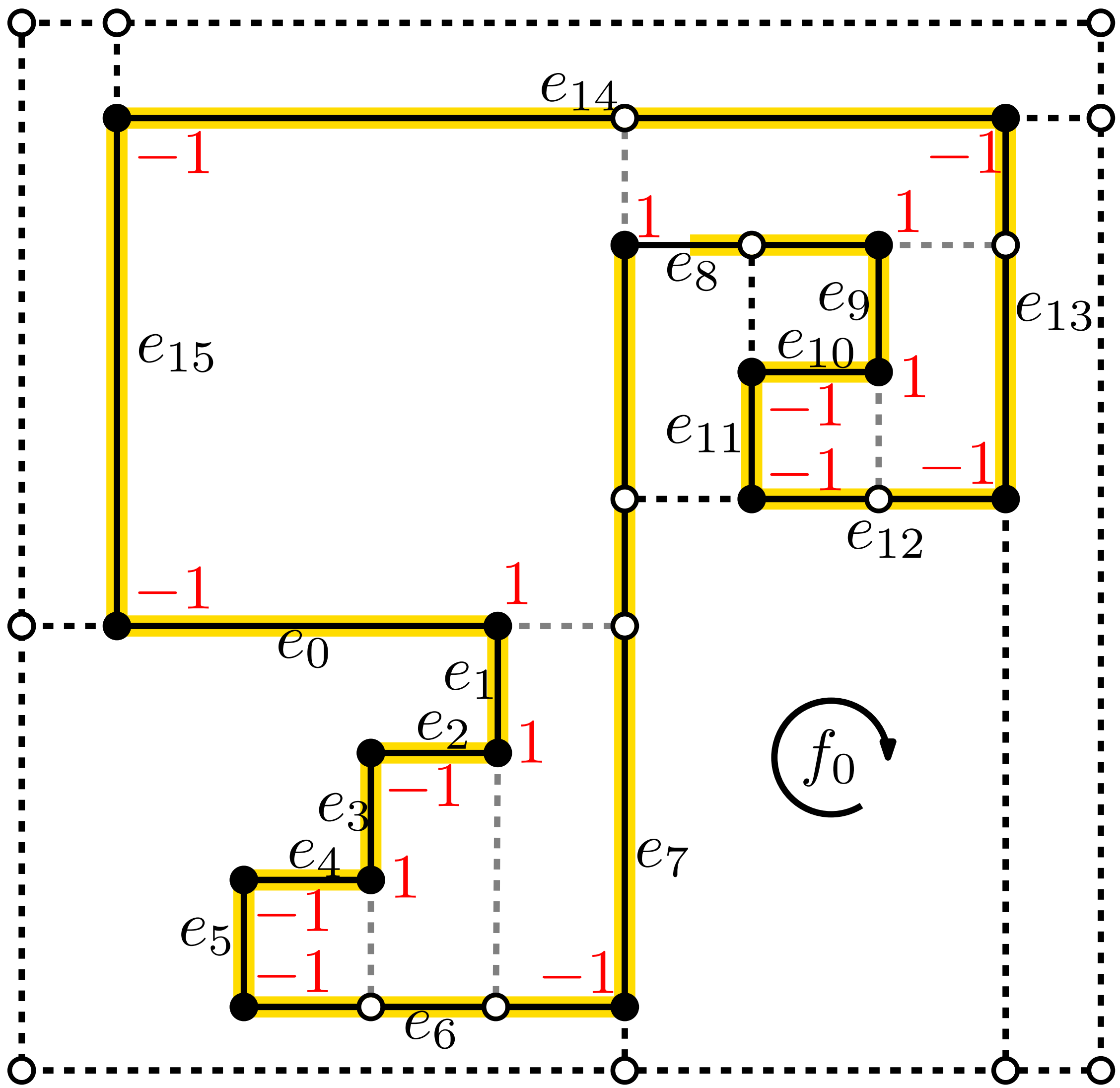
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn$(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

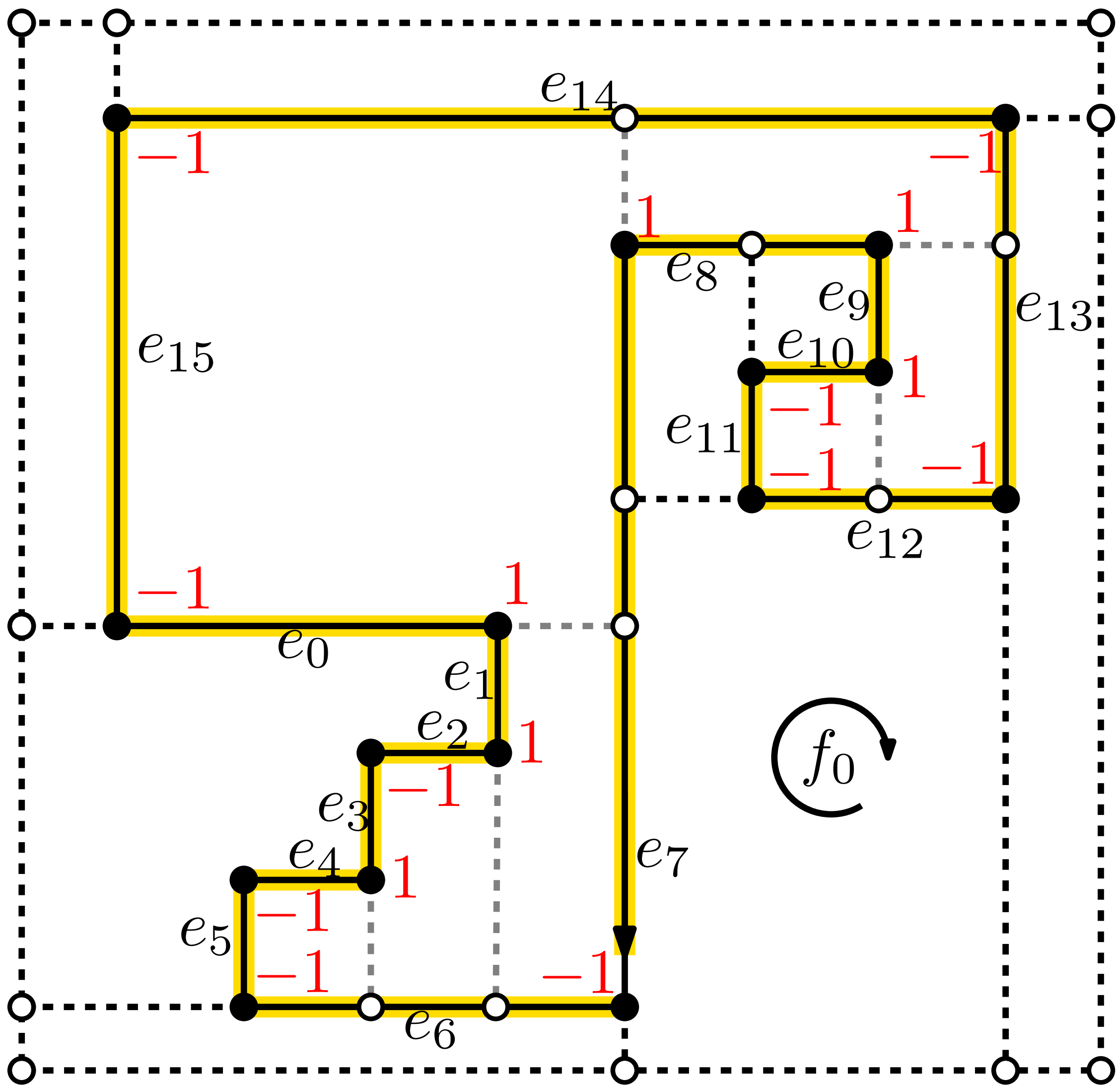
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum \text{turn}(e) < 1$ for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

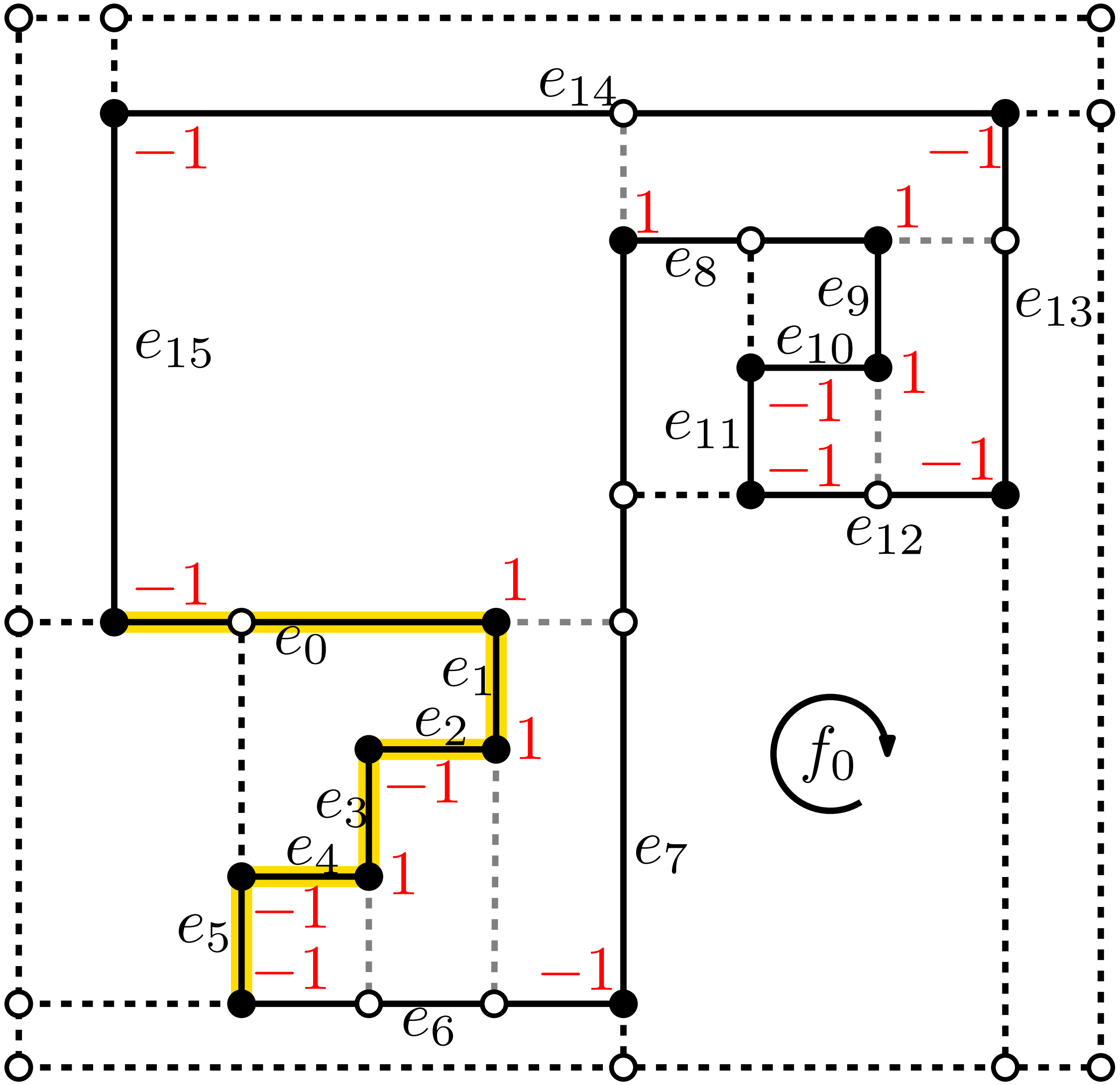
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn(e) < 1 for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

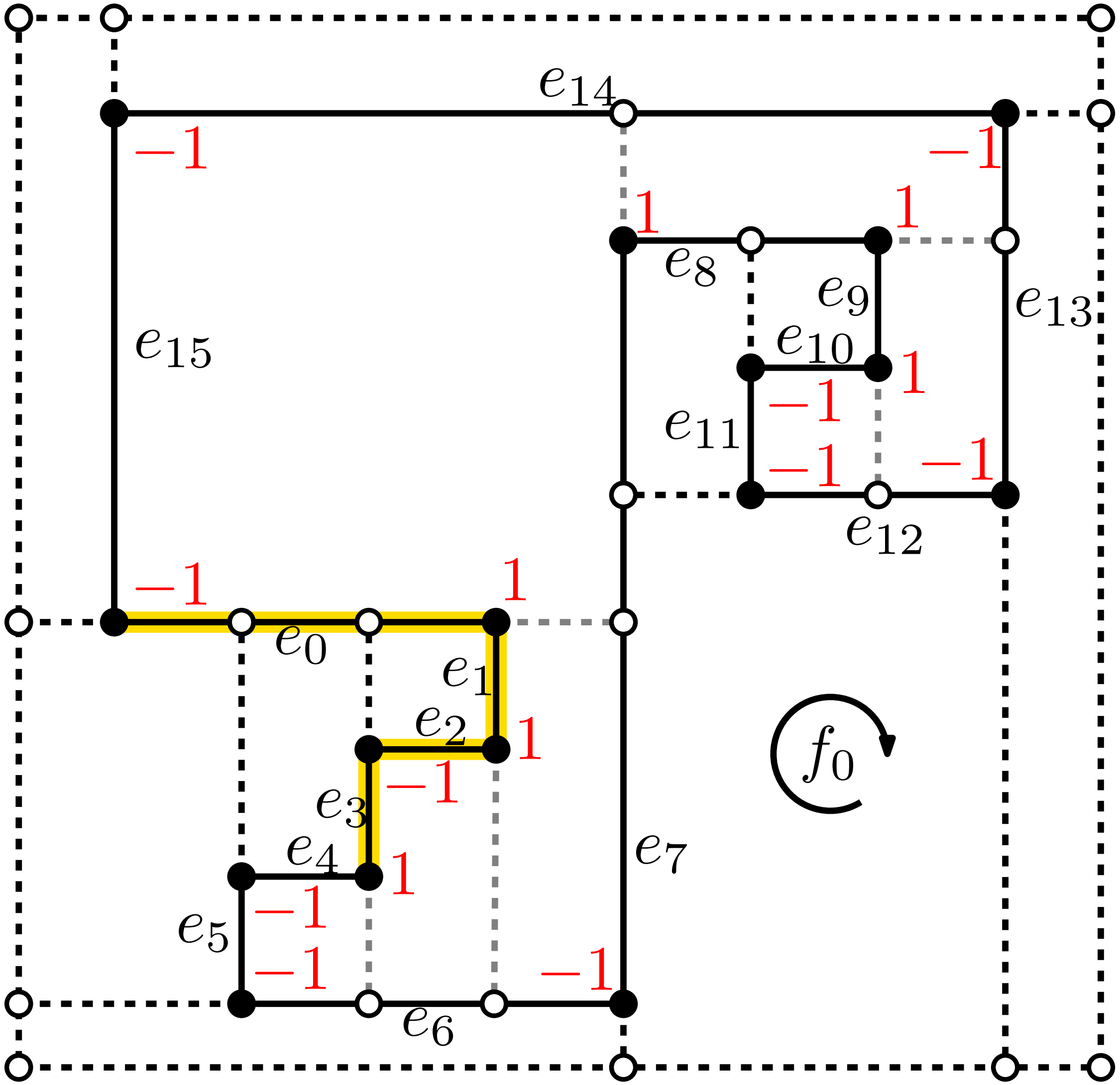
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn(e) < 1 for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

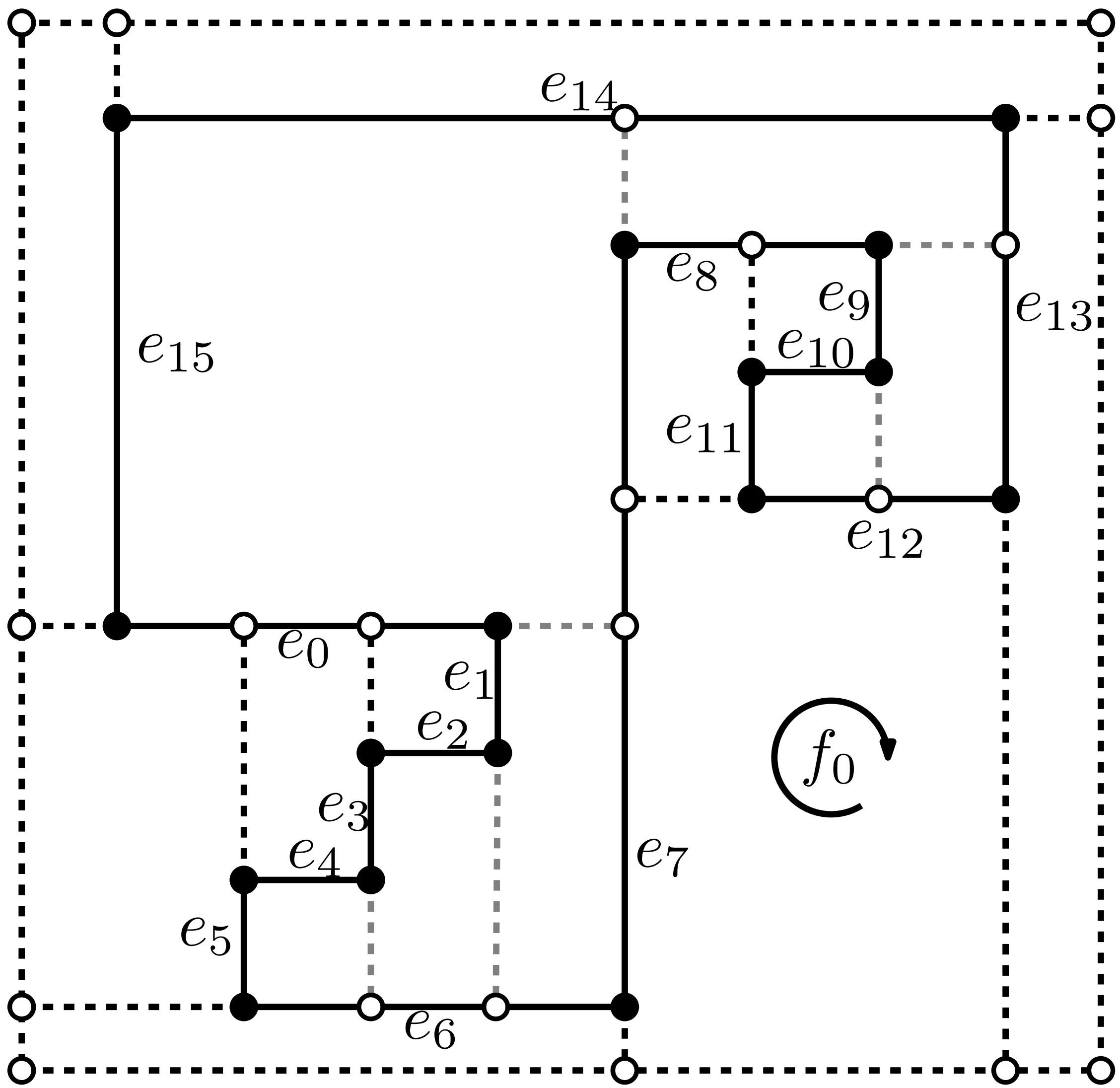
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn(e) < 1 for the complete turn around f_0, project on R

$$\text{turn}(e) = \begin{cases} 1 & \text{left bend} \\ 0 & \text{no bend} \\ -1 & \text{right bend} \end{cases}$$

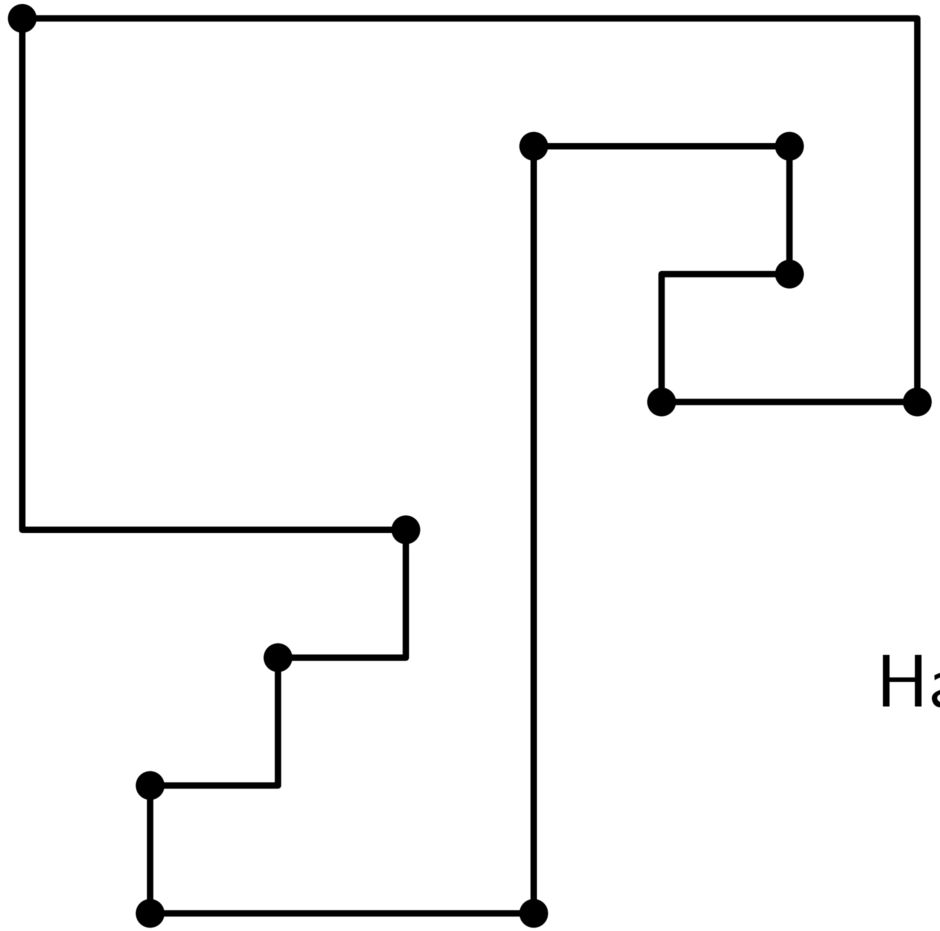
Refinement of (G, H) – Outer Face



- front(e) may be undefined
- when $\sum$ turn$(e) < 1$ for the complete turn around f_0, project on R

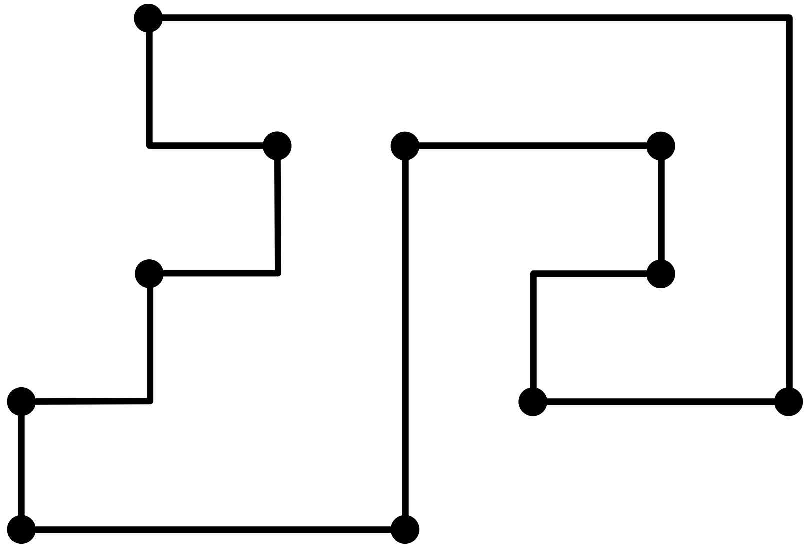
all faces are rectangles $\rightarrow$ apply flow network

Refinement of (G, H) – Outer Face



Has minimum area?

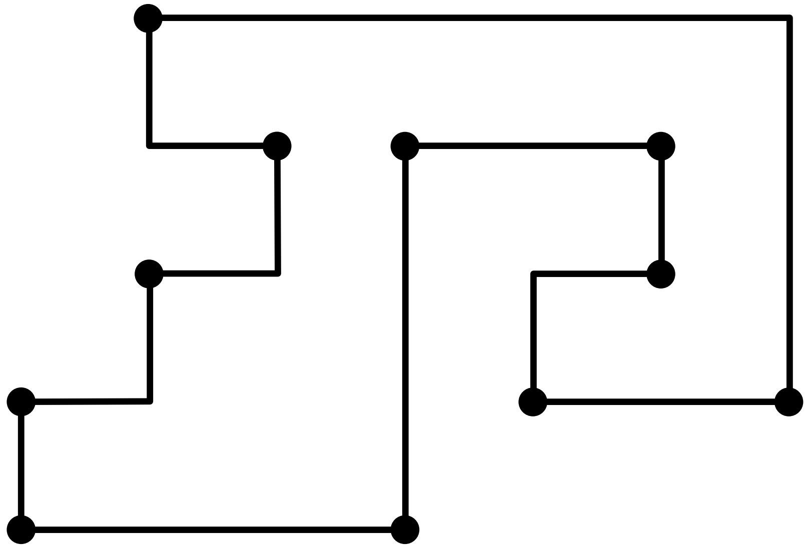
Refinement of (G, H) – Outer Face



Has minimum area?

NO!

Refinement of (G, H) – Outer Face



Has minimum area?

NO!

Area Minimization with a given orthogonal representation is an NP-hard problem!

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area and total edge length is achievable for the case of rectangular faces
- In case of non rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area and total edge length is achievable for the case of rectangular faces
- In case of non rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is an NP-hard problem. [Patrignany CGTA 2001]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area and total edge length is achievable for the case of rectangular faces
- In case of non rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is an NP-hard problem. [Patrignany CGTA 2001]
- Solvable with an integer linear program (ILP) [Klau, Mutzel IPCO 1999]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area and total edge length is achievable for the case of rectangular faces
- In case of non rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is an NP-hard problem. [Patrignany CGTA 2001]
- Solvable with an integer linear program (ILP) [Klau, Mutzel IPCO 1999]
- Various heuristics have been implemented and experimentally evaluated wrt running time and quality [Klau, Klein, Mutzel GD 2001]

Summary

- An orthogonal representation with minimum number of bends can be found in $O(n^{3/2})$ time
- Given an orthogonal representation a layout with minimum area and total edge length is achievable for the case of rectangular faces
- In case of non rectangular faces, reduce the problem to rectangular case. The resulting area is not minimum.
- Area minimization with a given orthogonal representation is an NP-hard problem. [Patrignany CGTA 2001]
- Solvable with an integer linear program (ILP) [Klau, Mutzel IPCO 1999]
- Various heuristics have been implemented and experimentally evaluated wrt running time and quality [Klau, Klein, Mutzel GD 2001]
- for non-planar graphs the area minimization is hard to approximate [Bannister, Eppstein, Simons JGAA 2012]

Upward Planar Drawings

Upward Planarity

Karlsruhe Institute of Technology

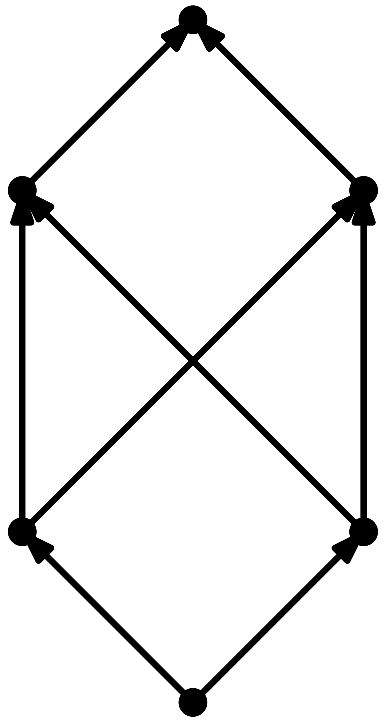
Def: A directed acyclic graph $D = (V, A)$ is called **upward planar**, when D admits a drawing (vertices points, edges simple curves), which is planar and each edge is a monotone curve increasing in y-direction.

Upward Planarity

Karlsruhe Institute of Technology

Def: A directed acyclic graph $D = (V, A)$ is called **upward planar**, when D admits a drawing (vertices points, edges simple curves), which is planar and each edge is a monotone curve increasing in y-direction.

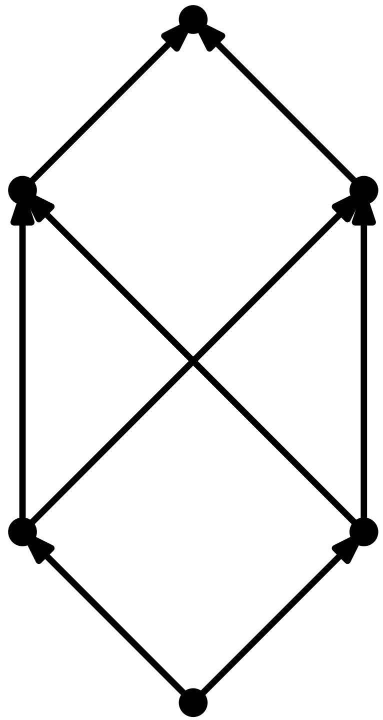
Example:

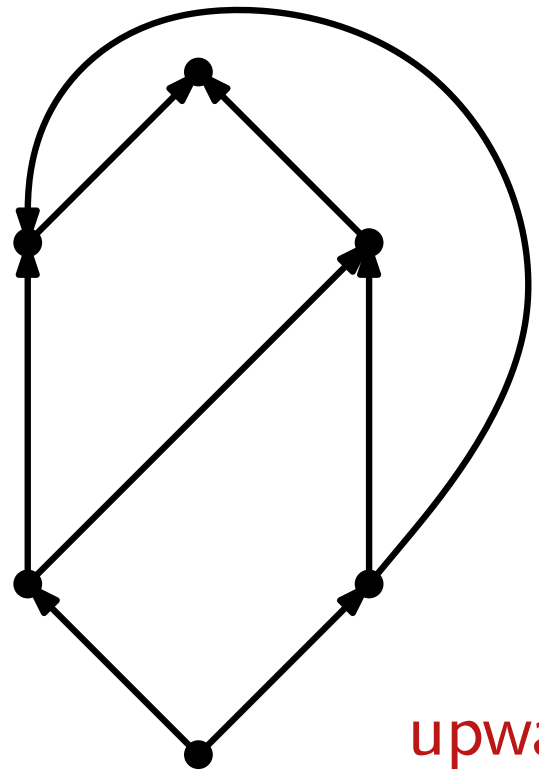


Upward Planarity

Def: A directed acyclic graph $D = (V, A)$ is called **upward planar**, when D admits a drawing (vertices points, edges simple curves), which is planar and each edge is a monotone curve increasing in y-direction.

Example:





planar!

upward planar? – NO!

Complexity

Thm 1: For a directed acyclis graph it is NP-hard to decide whether it is upward planar.

[Garg, Tamassia GD 1995]

Complexity

Thm 1: For a directed acyclis graph it is NP-hard to decide whether it is upward planar.

[Garg, Tamassia GD 1995]

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

[Bertolazzi et al.Algorithmica, 1994]

Complexity

Karlsruhe Institute of Technology

Thm 1: For a directed acyclis graph it is NP-hard to decide whether it is upward planar.

[Garg, Tamassia GD 1995]

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar. [Bertolazzi et al.Algorithmica, 1994]

Corol: For a triconnected planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

[Bertolazzi et al.Algorithmica, 1994]

Complexity

Thm 1: For a directed acyclis graph it is NP-hard to decide whether it is upward planar.

[Garg, Tamassia GD 1995]

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

[Bertolazzi et al.Algorithmica, 1994]

Corol: For a triconnected planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

[Bertolazzi et al.Algorithmica, 1994]

Thm 3: For a single-source acyclic digraph it can be tested whether it is upward planar in $O(n)$ time.

[Hutton, Lubiw, SIAM J.Comp.,1996]

Complexity

Thm 1: For a directed acyclis graph it is NP-hard to decide whether it is upward planar.

[Garg, Tamassia GD 1995]

NEXT

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar. [Bertolazzi et al.Algorithmica, 1994]

Corol: For a triconnected planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

[Bertolazzi et al.Algorithmica, 1994]

Thm 3: For a single-source acyclic digraph it can be tested whether it is upward planar in $O(n)$ time.

[Hutton, Lubiw, SIAM J.Comp.,1996]

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Proof:
- (2) $\Rightarrow$ (1) obvious

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Proof:
- (2) $\Rightarrow$ (1) obvious
- (1) $\Leftrightarrow$ (3) simple augmentation of a layout (blackboard)

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Proof:

- (2) $\Rightarrow$ (1) obvious
- (1) $\Leftrightarrow$ (3) simple augmentation of a layout (blackboard)
- (3) $\Rightarrow$ (2) triangulation and construction of straight-line drawing (blackboard)

Characterization

Karlsruhe Institute of Technology

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Proof:

- (2) $\Rightarrow$ (1) obvious
- (1) $\Leftrightarrow$ (3) simple augmentation of a layout (blackboard)
- (3) $\Rightarrow$ (2) triangulation and construction of straight-line drawing (blackboard)

- Step (3) $\Rightarrow$ (2) implies a $O(n)$ algorithm to construct a planar straight-line drawing of an st-digraph.

Characterization

Thm 4: For a directed graph $D = (V, A)$ the following statements are equivalent:

1. D is upward planar
2. D admits an upward planar straight-line drawing
3. D is the spanning subgraph of a planar st-digraph

[Di Battista, Tamassia TCS 1988]

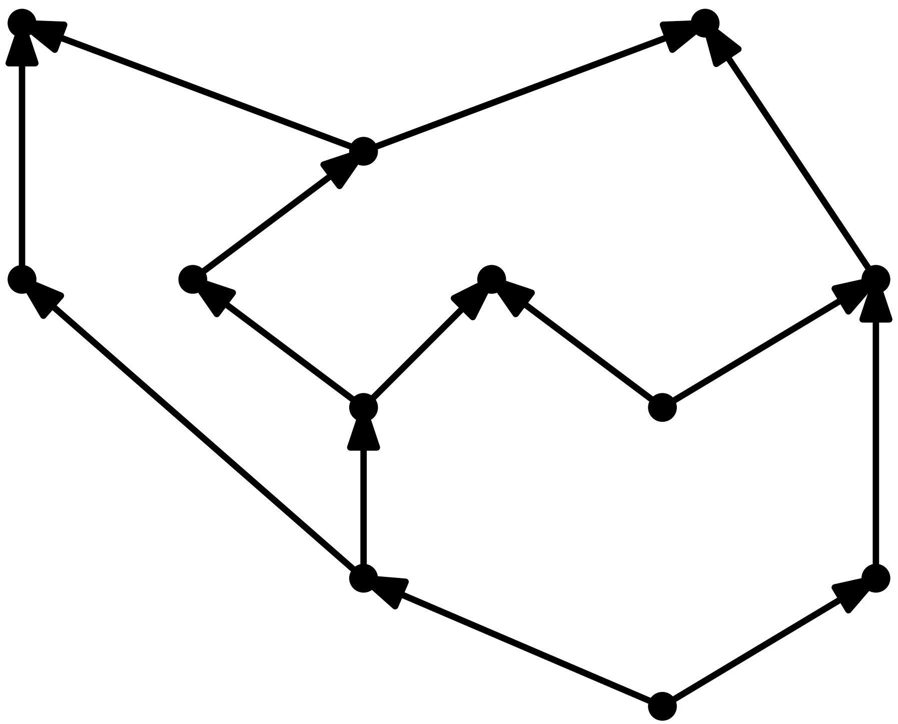
st-digraph: (i) single source s and sink t, (ii) edge $(s, t) \in E$

Proof:
- (2) $\Rightarrow$ (1) obvious
- (1) $\Leftrightarrow$ (3) simple augmentation of a layout (blackboard)
- (3) $\Rightarrow$ (2) triangulation and construction of straight-line drawing (blackboard)

- Step (3) $\Rightarrow$ (2) implies a $O(n)$ algorithm to construct a planar straight-line drawing of an st-digraph.
- May have exponential area. There are examples where this is unavoidable (see Lecture 3).

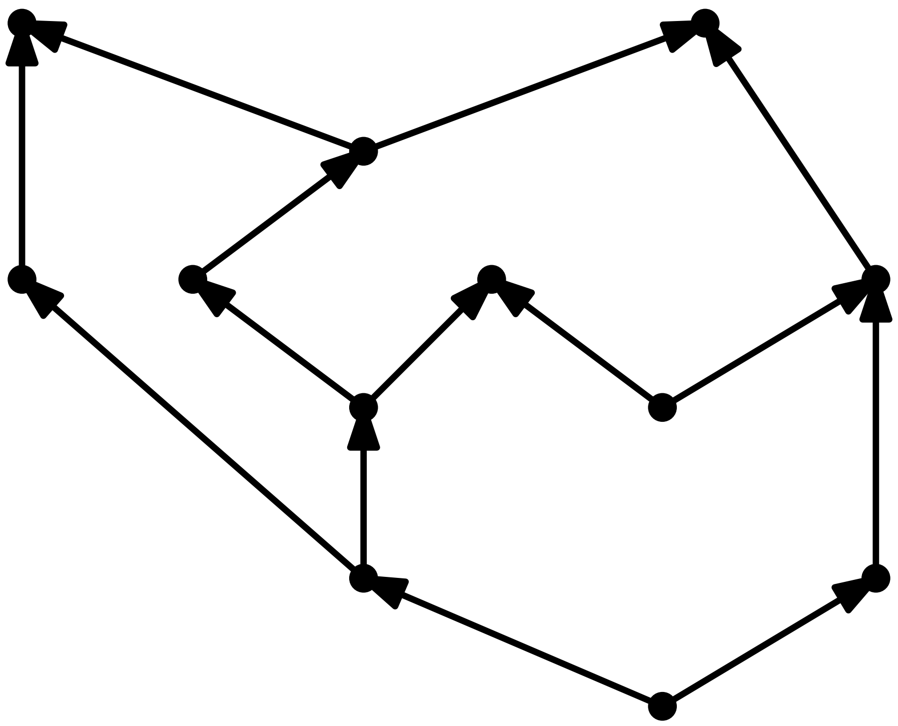
Fixed Outer Face: Angles

Problem: Consider a directed acyclic graph $D = (V, A)$ with embedding $\mathcal{F}, f_0$. Test whether $D, \mathcal{F}, f_0$ is upward planar and construct corresponding drawing.

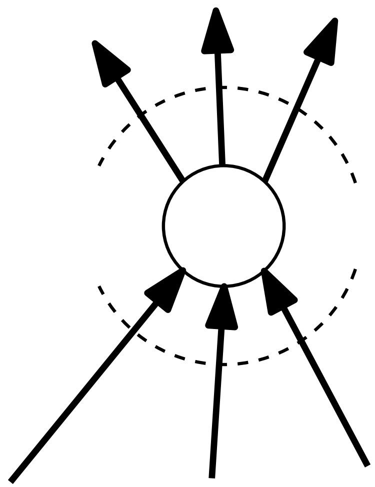


Fixed Outer Face: Angles

Problem: Consider a directed acyclic graph $D = (V, A)$ with embedding $\mathcal{F}, f_0$. Test whether $D, \mathcal{F}, f_0$ is upward planar and construct corresponding drawing.

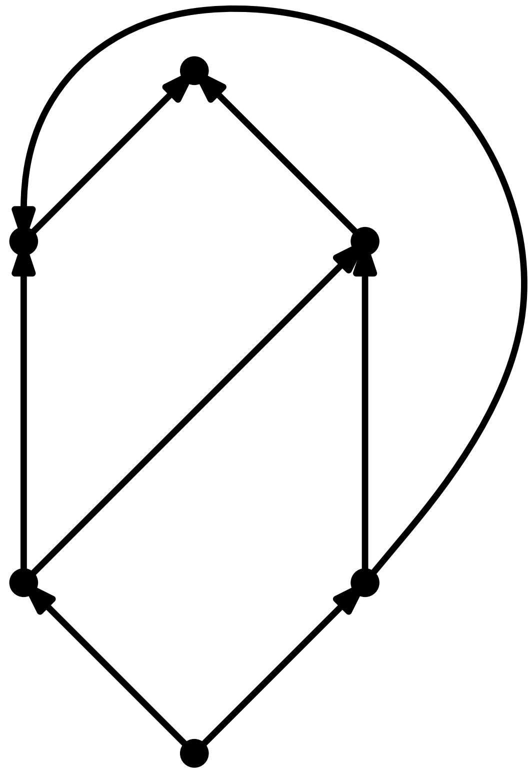


Embedding is **bimodal** if for each node:



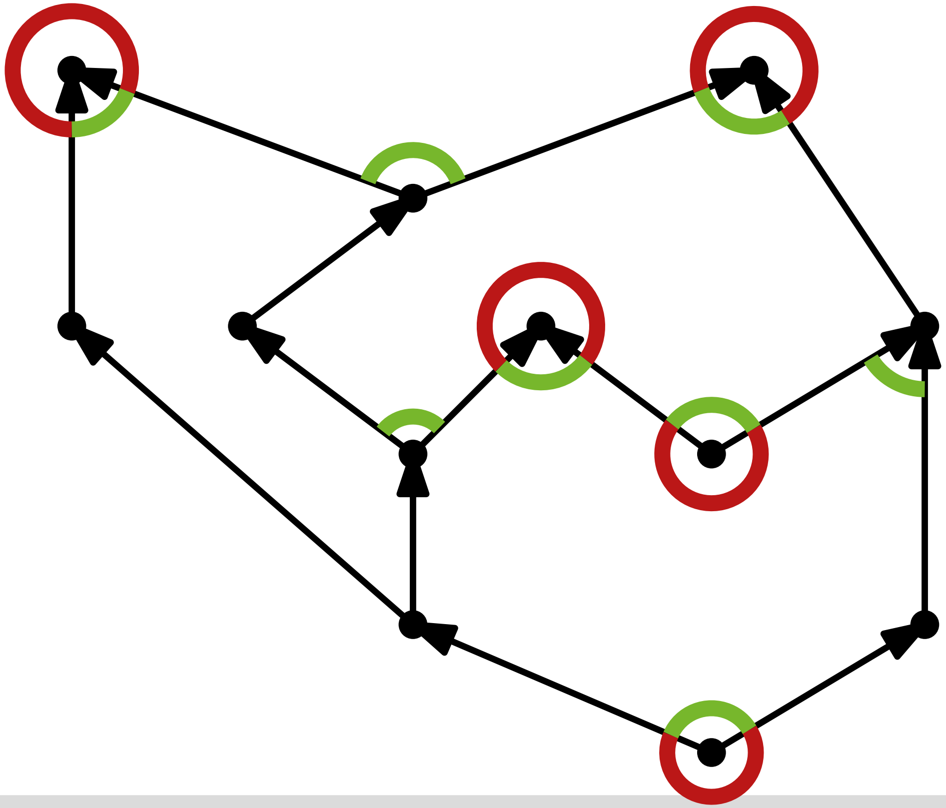
Fixed Outer Face: Observations

- Bimodality is necessary but not sufficient



Fixed Outer Face: Observations

- Bimodality is necessary but not sufficient
- measure angles between two incomming/outgoing edges
  Angle α is **large** when $\alpha > \pi$, **small** otherwise
  $L(v)$:= # large angles at node v
  $L(f)$:= # large angles in face f
  $S(v)$ resp. $S(f)$: # **small** angles



Fixed Outer Face: Observations

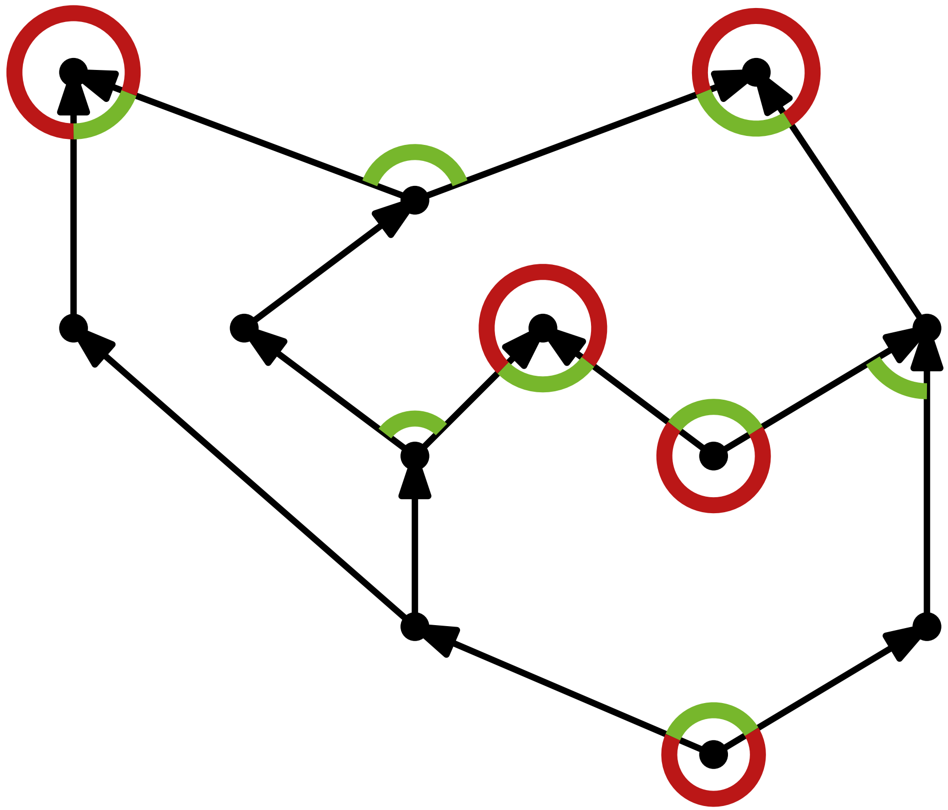
- Bimodality is necessary but not sufficient
- measure angles between two incomming/outgoing edges
  Angle α is **large** when $\alpha > \pi$, **small** otherwise
  $L(v)$:= # large angles at node v
  $L(f)$:= # large angles in face f
  $S(v)$ resp. $S(f)$: # **small** angles

Lemma 1: In any upward layout of D holds:

$$(1)\ \forall v \in V : L(v) = \begin{cases} 0 & v \text{ non source/sink} \\ 1 & v \text{ source/sink} \end{cases}$$

$$(2)\ \forall f \in \mathcal{F} : L(f) - S(f) = \begin{cases} -2 & f \neq f_0 \\ 2 & f = f_0 \end{cases}$$

Fixed Outer Face: Observations



Lemma 1: In any upward layout of D holds:

(1) $\forall v \in V : L(v) = \begin{cases} 0 & v \text{ non source/sink} \\ 1 & v \text{ source/sink} \end{cases}$

(2) $\forall f \in \mathcal{F} : L(f) - S(f) = \begin{cases} -2 & f \neq f_0 \\ 2 & f = f_0 \end{cases}$

Fixed Outer Face: Observations

- $A(f) := \#$ sources in face f (equal to the number of sinks)
  It holds that: $L(f) + S(f) = 2A(f)$ for all faces.
- in any upward planar layout of D holds:
  $$\forall f \in \mathcal{F} : L(f) = \begin{cases} A(f) - 1 & f \neq f_0 \\ A(f) + 1 & f = f_0 \end{cases}$$

Fixed Outer Face: Observations

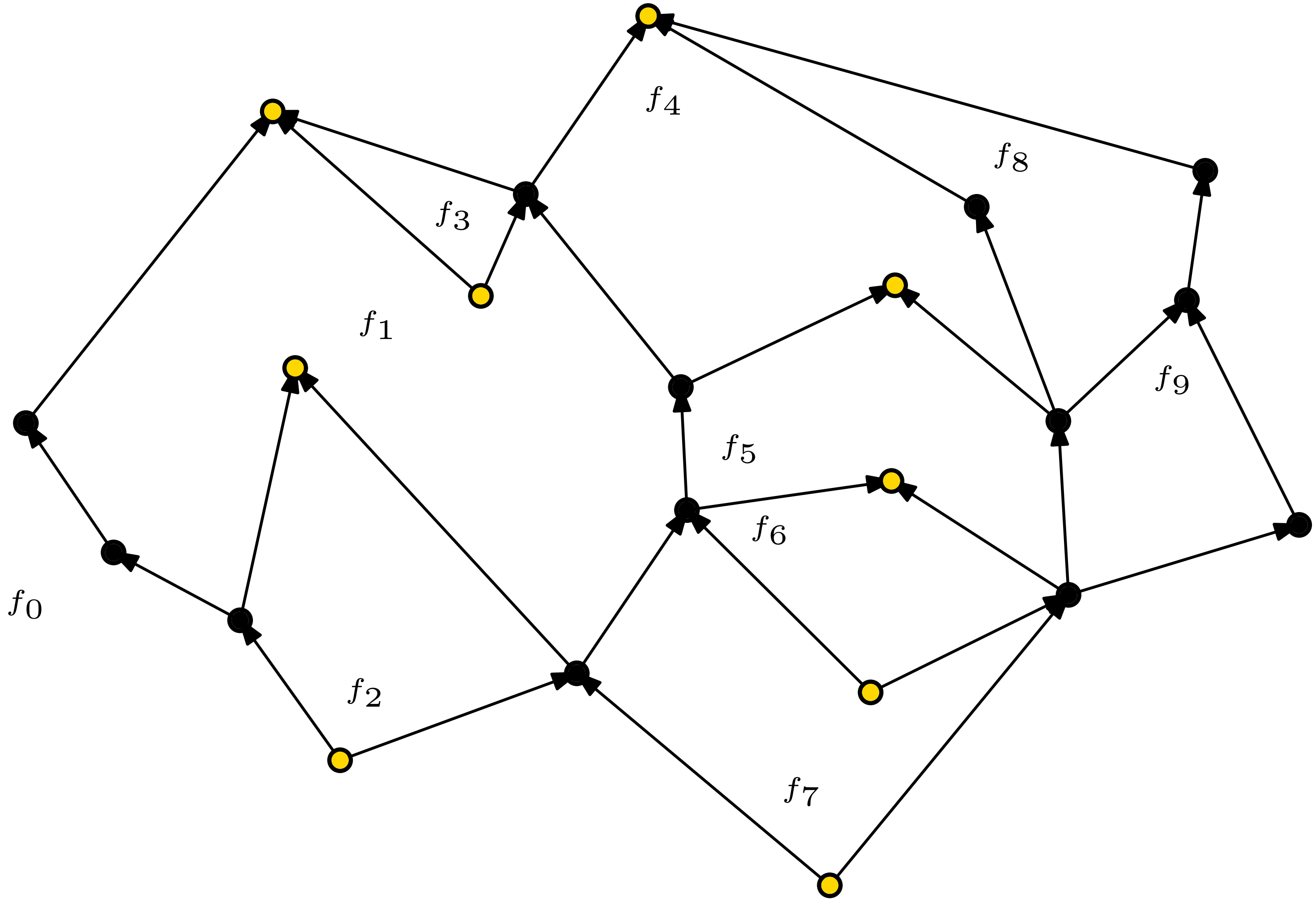
- $A(f) := \#$ sources in face f (equal to the number of sinks)
  It holds that: $L(f) + S(f) = 2A(f)$ for all faces.

- in any upward planar layout of D holds:
  $$\forall f \in \mathcal{F} : L(f) = \begin{cases} A(f) - 1 & f \neq f_0 \\ A(f) + 1 & f = f_0 \end{cases}$$

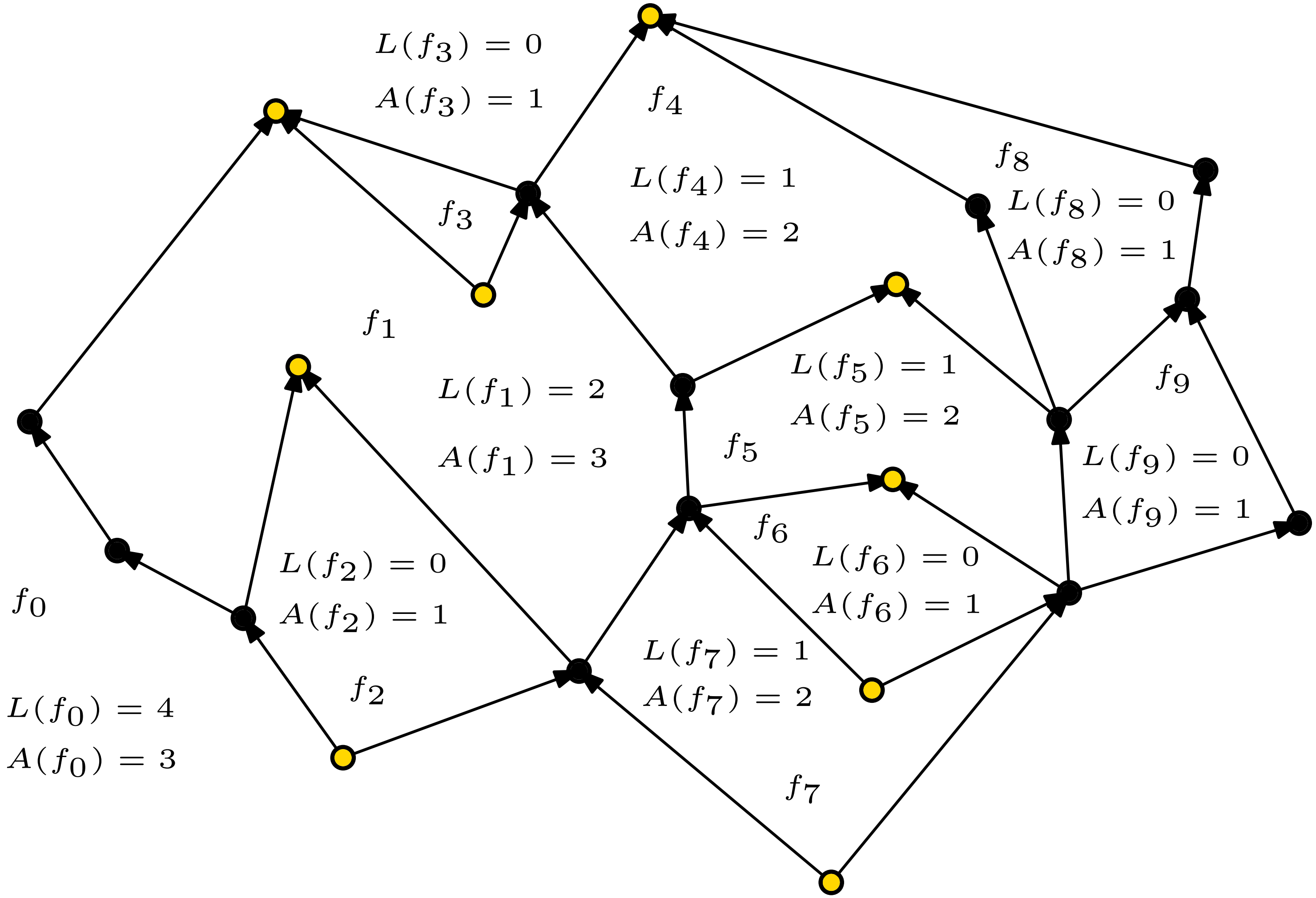
- Define assignement $\Phi : S \cup T \to \mathcal{F}$
  (S set of sources, T sinks), where
  $\Phi : v \mapsto$ incident face, where v is forms large angle

- Φ is called **consistent**, if: $|\Phi^{-1}(f)| = \begin{cases} A(f) - 1 & f \neq f_0 \\ A(f) + 1 & f = f_0 \end{cases}$

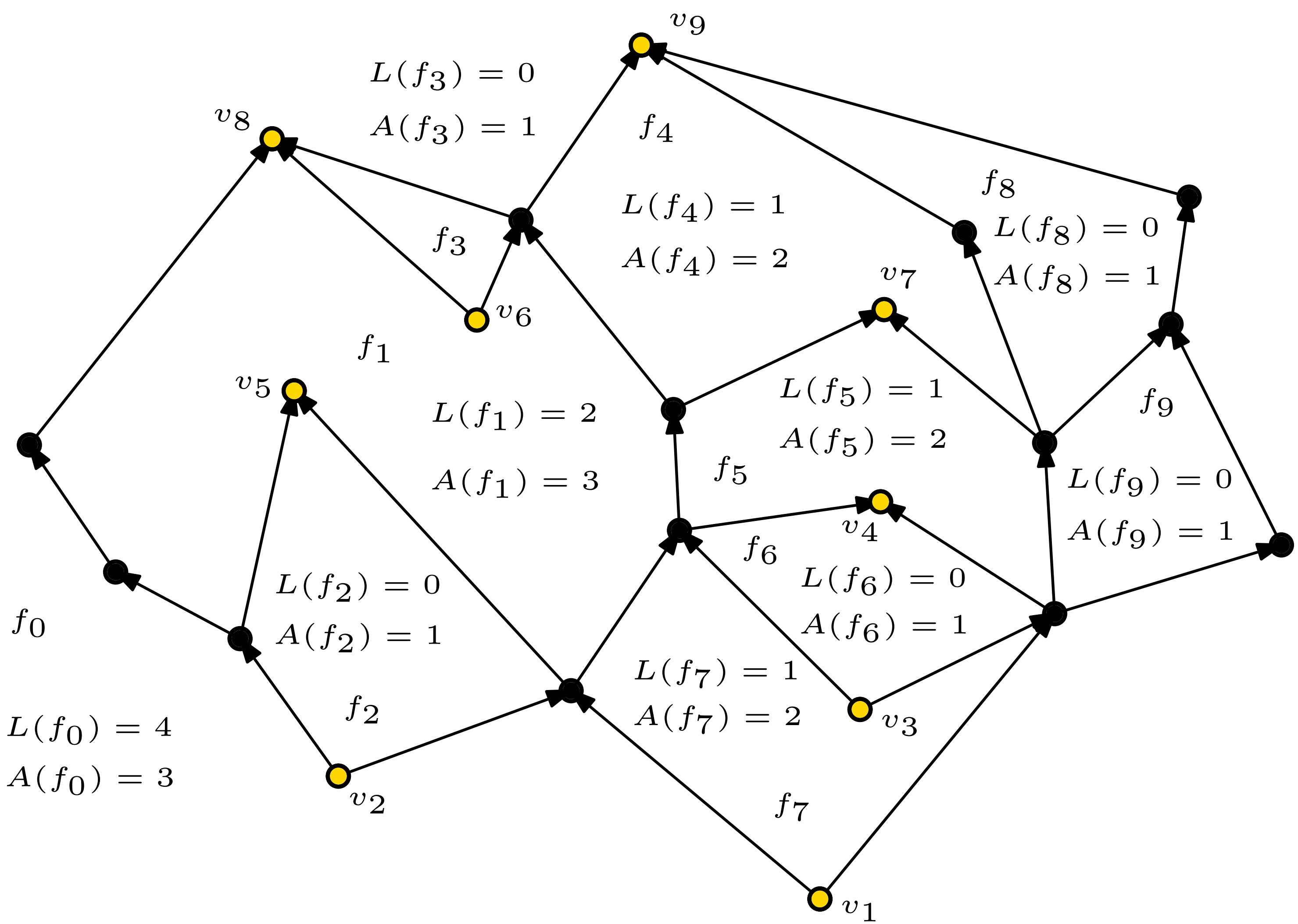
Example: Vertex-Face Assignement



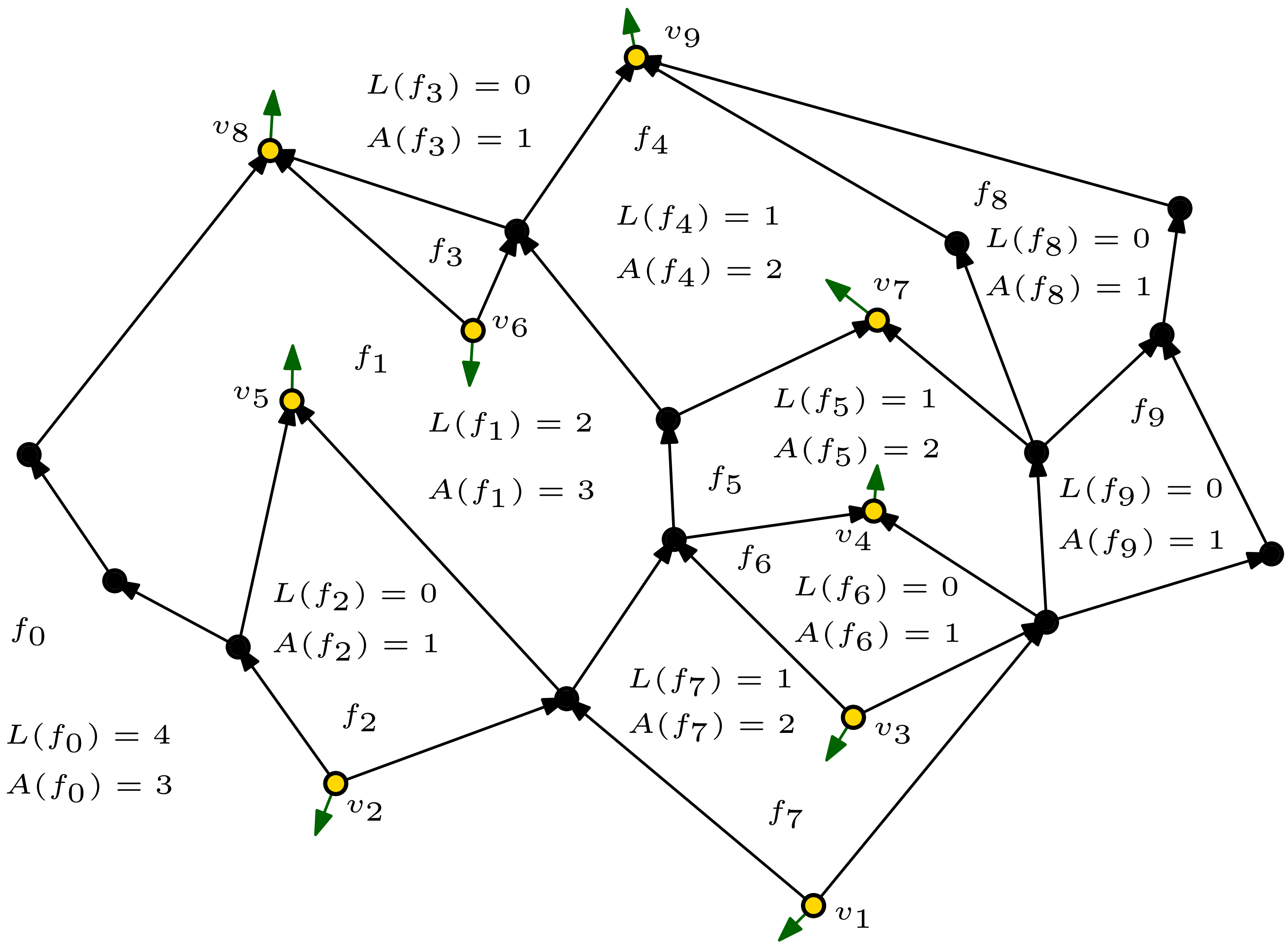
Example: Vertex-Face Assignement



Example: Vertex-Face Assignement



Example: Vertex-Face Assignement



$\Phi(v_1) = f_0$
$\Phi(v_2) = f_0$
$\Phi(v_3) = f_7$
$\Phi(v_4) = f_5$
$\Phi(v_5) = f_1$
$\Phi(v_6) = f_1$
$\Phi(v_7) = f_4$
$\Phi(v_8) = f_0$
$\Phi(v_9) = f_0$

Characterization

Thm 5: For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

Characterization

Thm 5: For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

Proof:

$\Rightarrow$ already clear

Characterization

Thm 5: For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

Proof:

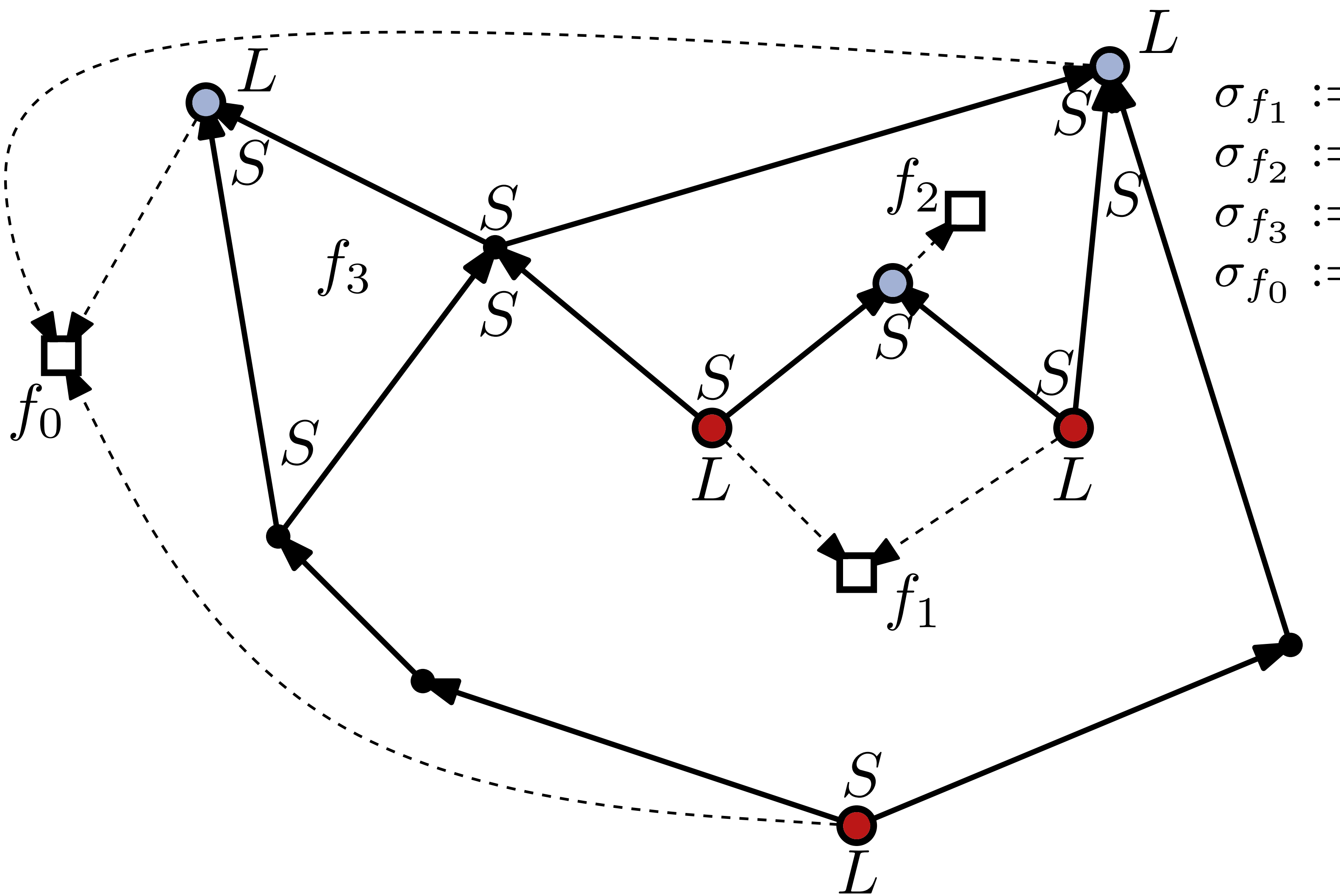
$\Rightarrow$ already clear

$\Leftarrow$ construct an st-digraph that contains D as spanning subgraph:

- insert edges in faces until they have single source and sink
- proof acyclicity, planarity and bimodality

Proof of Theorem 5

Assign labels s_L, t_L, s_S, t_S to each source/sink of each face f. Sequence σ_f.



$\sigma_{f_1} := (S, S, L, S, L, S)$

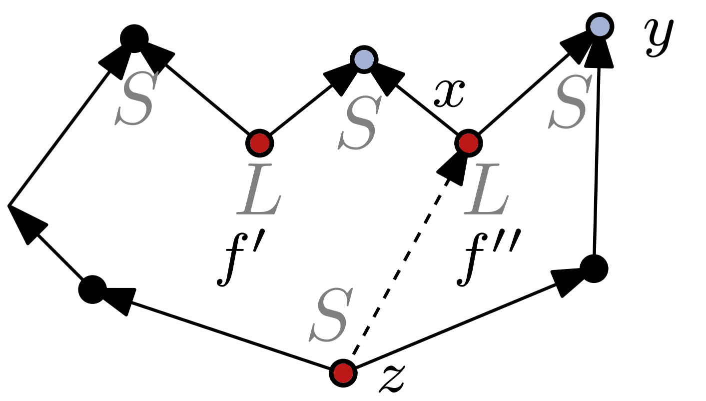
$\sigma_{f_2} := (L, S, S, S)$

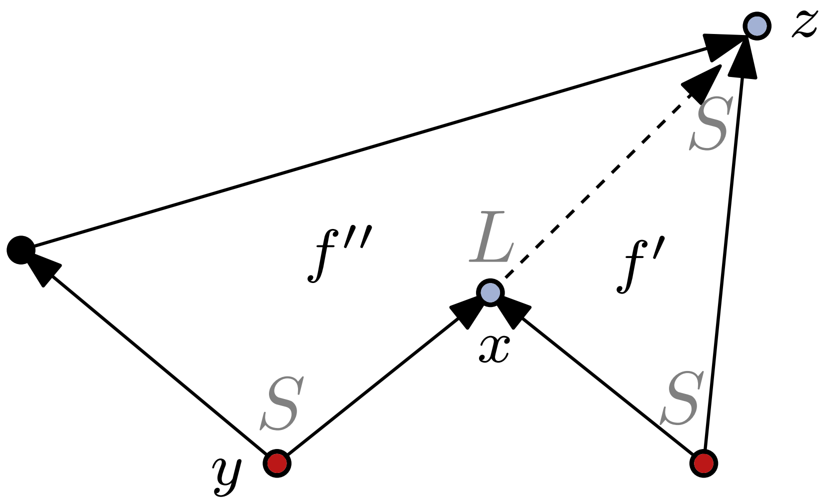
$\sigma_{f_3} := (S, S)$

$\sigma_{f_0} := (L, L, S, L)$

Proof of Theorem 5

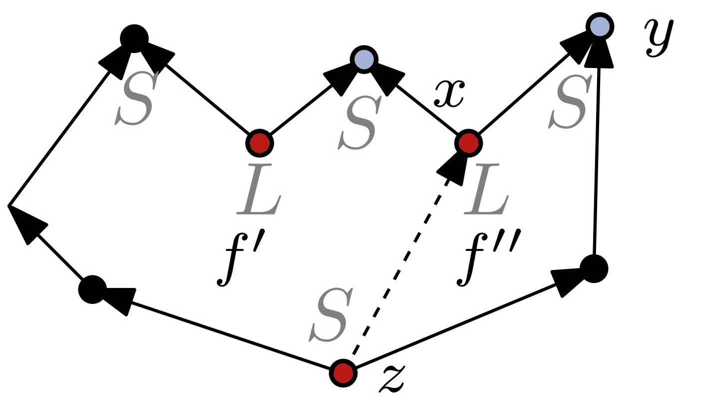
- Cancel all sources and sinks: search for subsequence LSS.

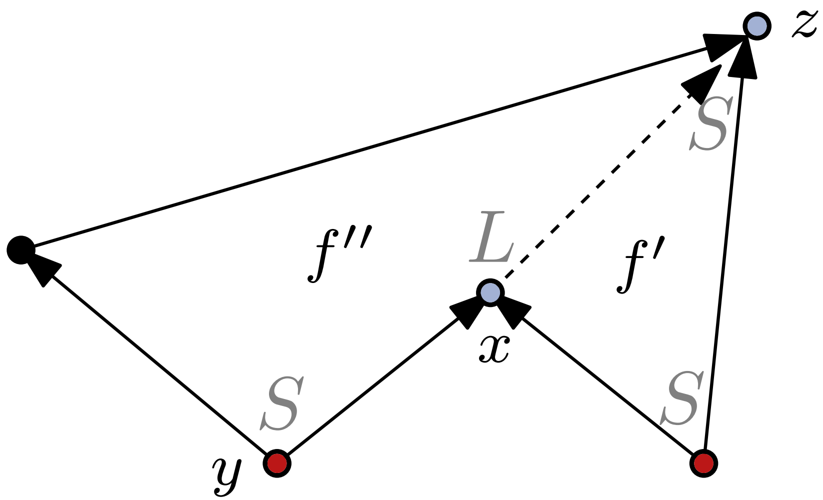




Proof of Theorem 5

- Cancel all sources and sinks: search for subsequence LSS.

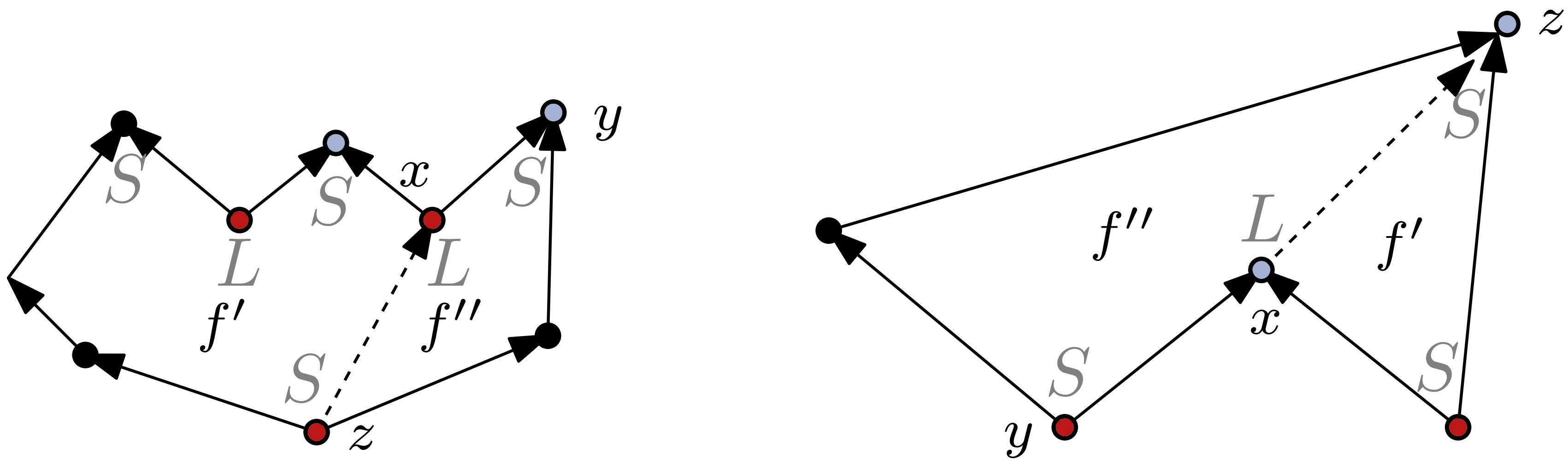




- Invariants of construction: planarity, acyclicity, bimodality
- In the outerface: select super source (resp. super sink) and add edges to (from) other sources (resp. sinks)

Proof of Theorem 5

- Cancel all sources and sinks: search for subsequence LSS.



- Invariants of construction: planarity, acyclicity, bimodality
- In the outerface: select super source (resp. super sink) and add edges to (from) other sources (resp. sinks)

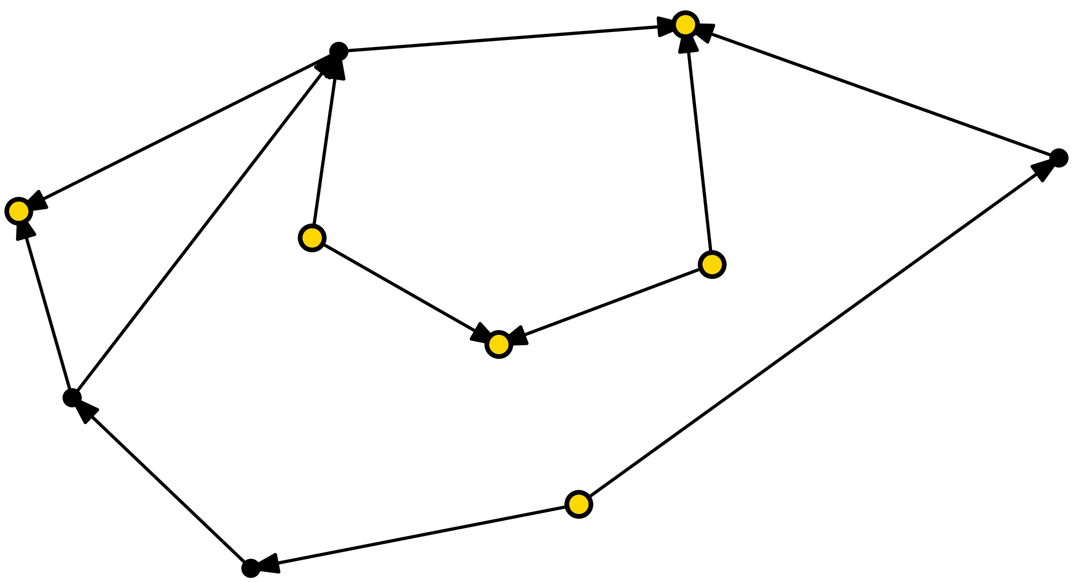
How to check whether a consistent asignement exists?

Flow Network

Def: Flow network $N(D, \mathcal{F}, f_0) = ((W, A_N); \ell; u; b)$

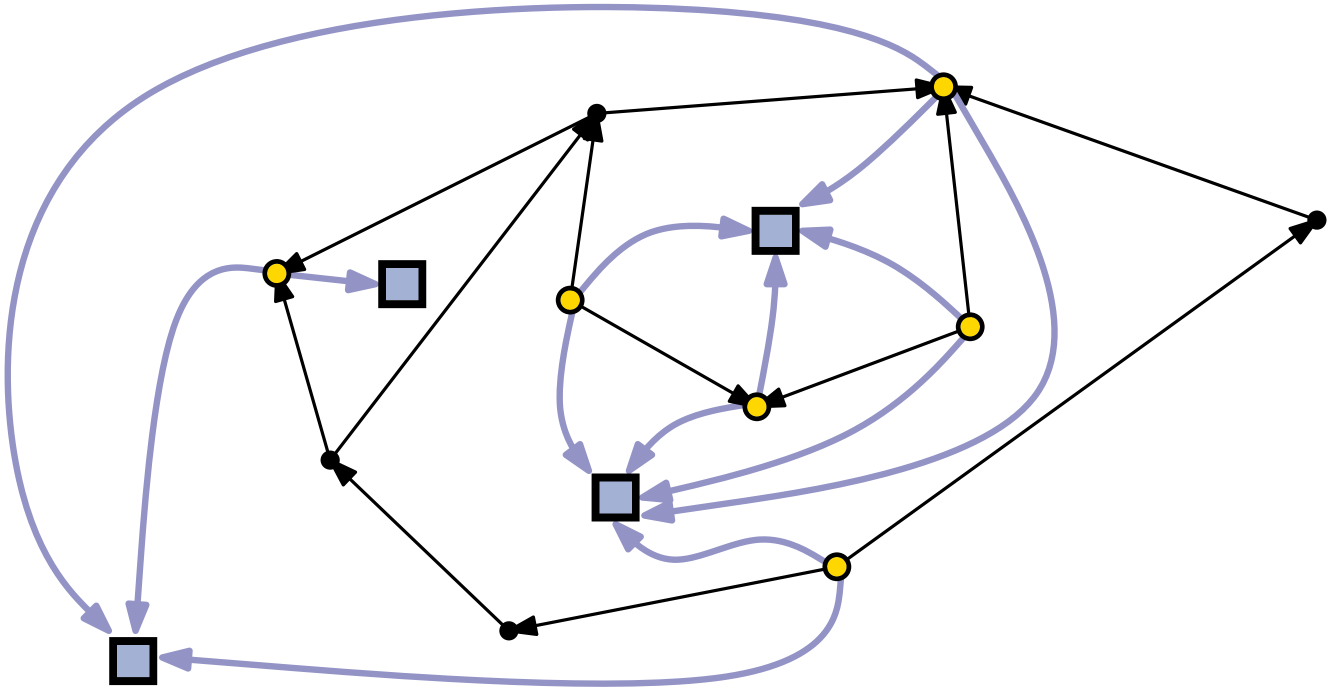
- $W = \{v \in V \mid v \text{ is source or sink }\} \cup \mathcal{F}$
- $A_N = \{(v, f) \mid v \text{ incident to } f\}$
- $l(a) = 0 \quad \forall a \in A_N$
- $u(a) = 1 \quad \forall a \in A_N$
- $b(q) = \begin{cases} 1 & \forall q \in W \cap V \\ -(A(q) - 1) & \forall q \in \mathcal{F} \setminus \{f_0\} \\ -(A(q) + 1) & q = f_0 \end{cases}$

Example



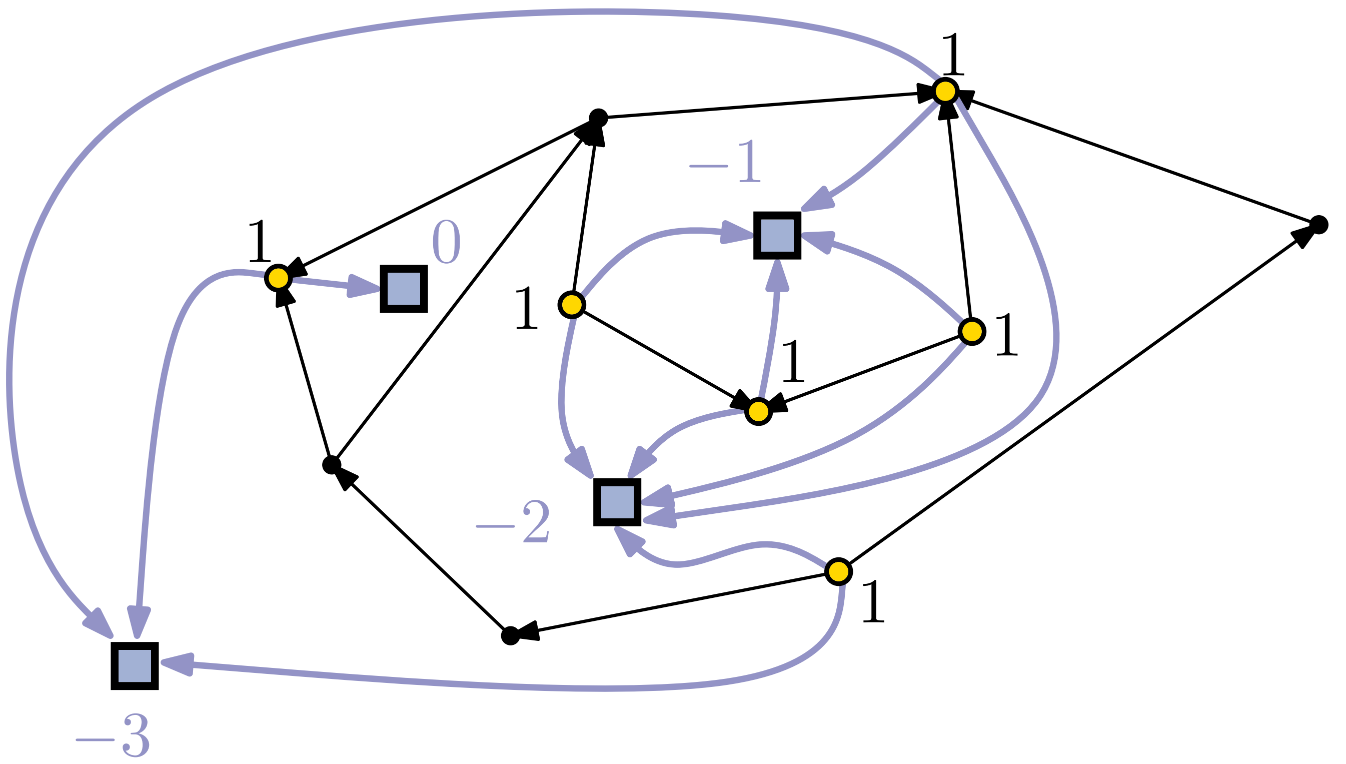
- normal nodes
- sources/sinks

Example



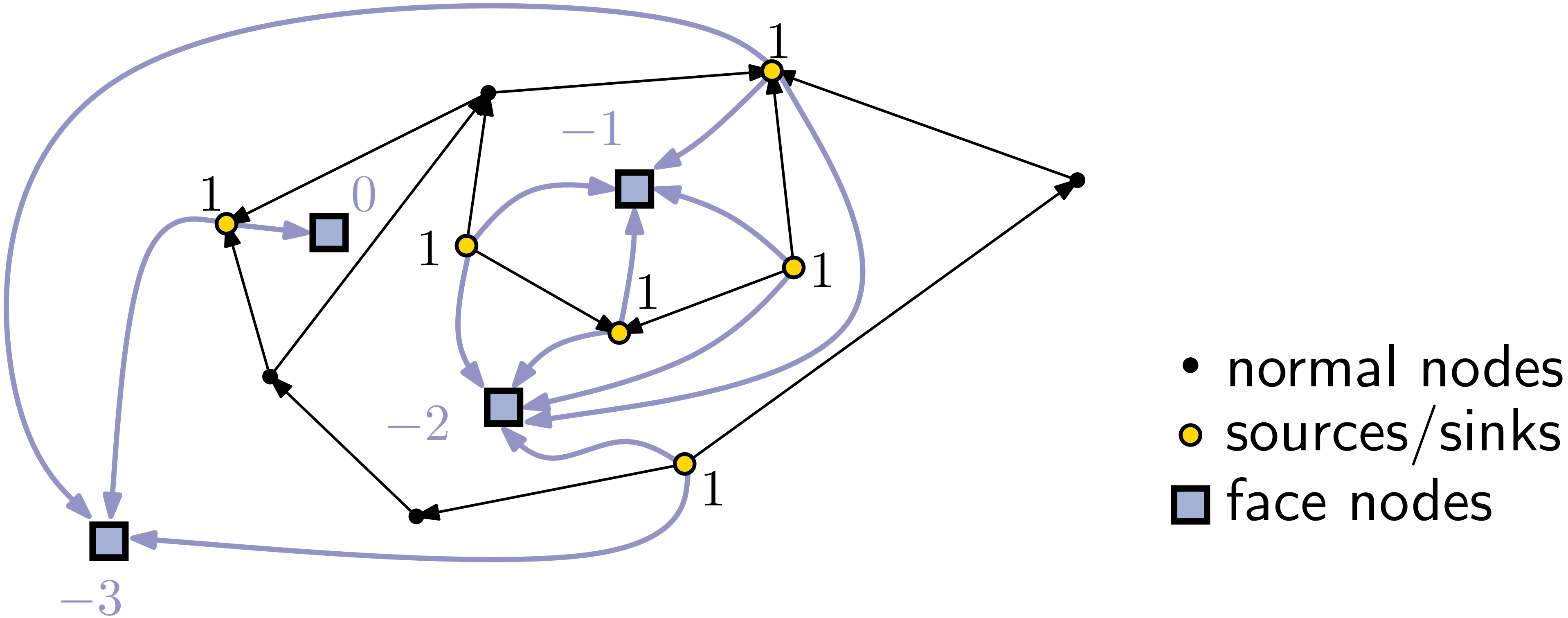
- normal nodes
- sources/sinks
- face nodes

Example



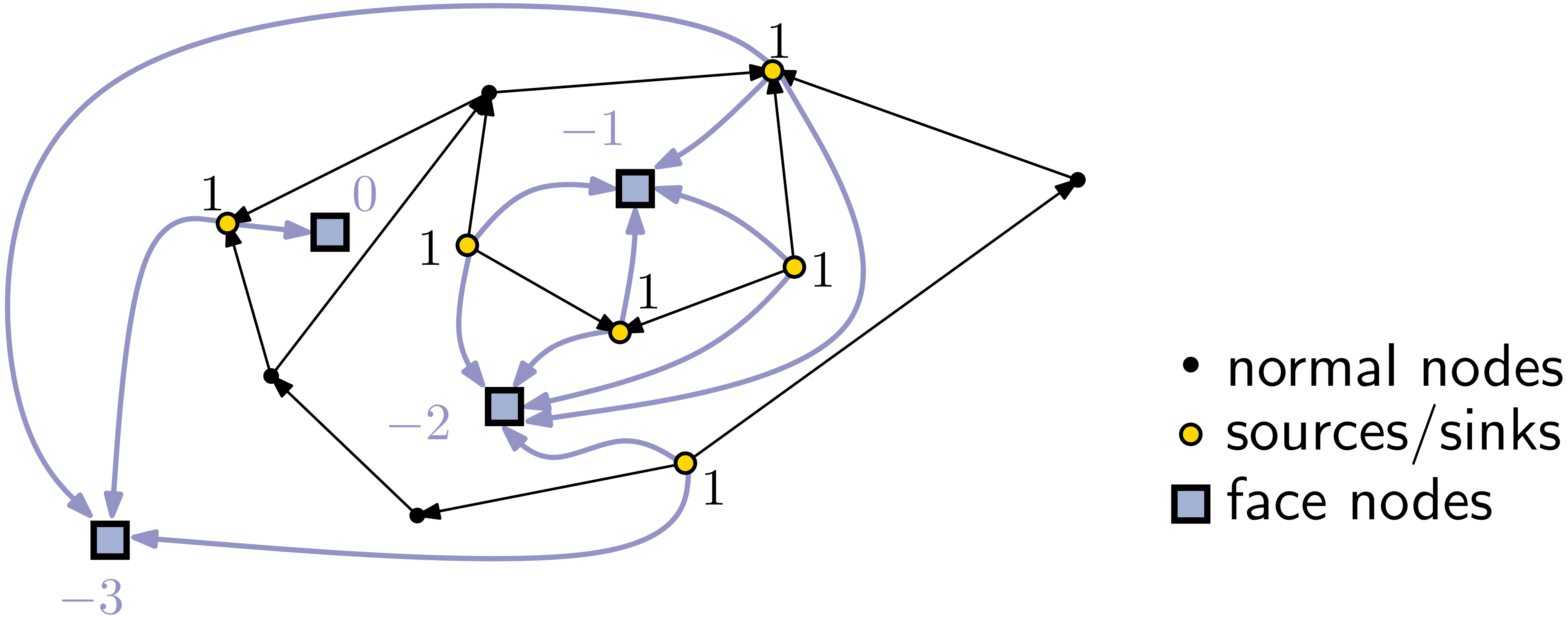
- normal nodes
- sources/sinks
- face nodes

Example



Thm 6: Let G be a directed acyclic digraph with embedding F and outer face f_0. Bipartite flow network $N(D, \mathcal{F}, f_0)$ admits a valid flow of value r (# of sources/sinks) iff G has a consistent asignement of sources and sinks to faces.

Example



- start with zero flow
- search for augmenting path (r times for total of r sources and sinks)

Final Remarks

- $O(rn)$ to decide whether consistent assignement exists
- works also without fixed outer face f_0: first compute all faces as internal and then add two units of demand to a face vertex and test whether the total flow can be augmented by two units. Do it for every face.

Final Remarks

- $O(rn)$ to decide whether consistent assignement exists
- works also without fixed outer face f_0: first compute all faces as internal and then add two units of demand to a face vertex and test whether the total flow can be augmented by two units. Do it for every face.

Thm 5: For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
(recall)
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

Final Remarks

- $O(rn)$ to decide whether consistent assignement exists
- works also without fixed outer face f_0: first compute all faces as internal and then add two units of demand to a face vertex and test whether the total flow can be augmented by two units. Do it for every face.

Thm 5: (recall) For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

$+$ algorithm to test the existence of assignement, imply:

Final Remarks

- $O(rn)$ to decide whether consistent assignement exists
- works also without fixed outer face f_0: first compute all faces as internal and then add two units of demand to a face vertex and test whether the total flow can be augmented by two units. Do it for every face.

Thm 5: (recall) For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

$+$ algorithm to test the existence of assignement, imply:

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

Final Remarks

- $O(rn)$ to decide whether consistent assignement exists
- works also without fixed outer face f_0: first compute all faces as internal and then add two units of demand to a face vertex and test whether the total flow can be augmented by two units. Do it for every face.

Thm 5: (recall) For a directed acyclic graph $D = (V, A)$ with combinatorial embedding $\mathcal{F}, f_0$ it holds:
D is upward planar $\Leftrightarrow$ D bimodal and $\exists$ consistent Φ

$+$ algorithm to test the existence of assignement, imply:

Thm 2: For a **combinatorially embedded*** planar directed graph it can be tested in $O(n^2)$ time whether it is upward planar.

The layout can be constructed in the same time: $O(n)$ to augment to st-digraphs and $O(n)$ to draw the st-digraph

Discussion

Discussion

- There exists a fixed parameter tractable algorithm to test upward planarity, with parameter number of triconnected components [Healy, Lynch SOFSEM 2005]
- The decision of Theorem 2 can be done in $O(n + r^{1.5})$ time where $r = \#$ sources/sinks [Abbasi, Healy, Rextin IPL 2010]
- many related concepts have been studied recently: quasi-planarity, upward drawings of mixed graphs