Karlsruhe Institute of Technology

Algorithms for Graph Visualization

Flow Methods: Orthogonal Layouts – Part II

INSTITUT FÜR THEORETISCHE INFORMATIK · FAKULTÄT FÜR INFORMATIK

Torsten Ueckerdt

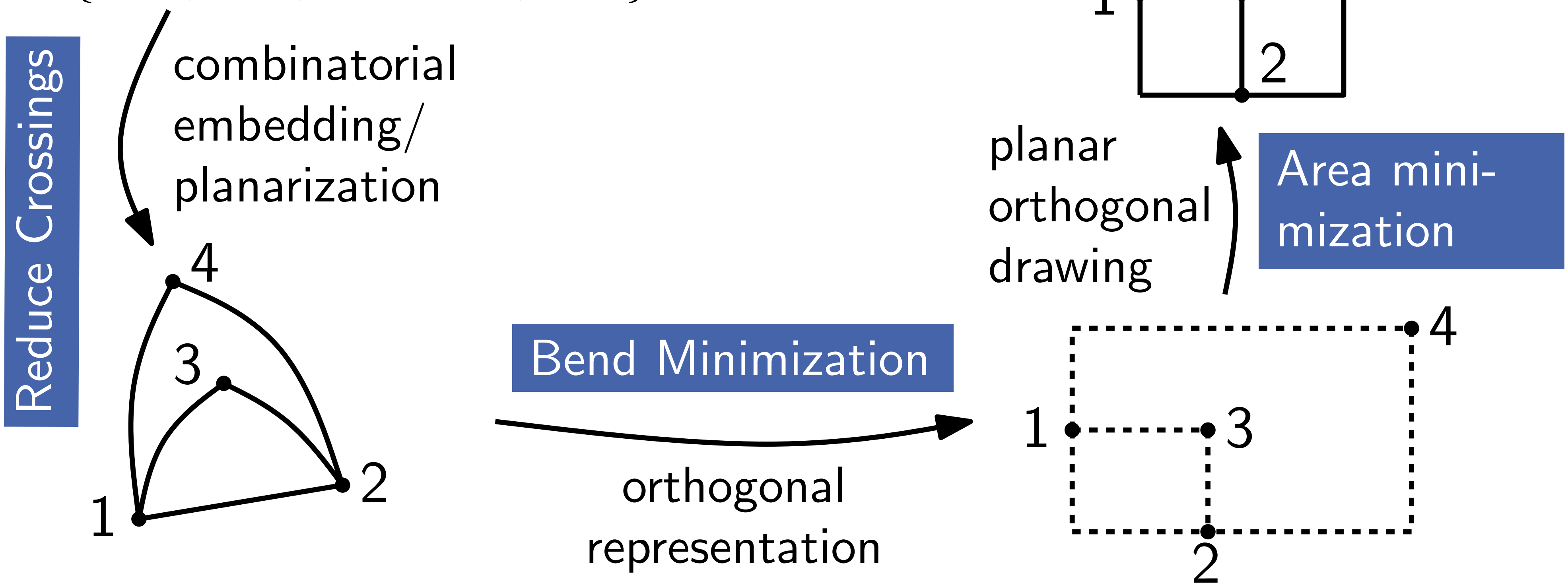
04.12.2019

(Planar) Orthogonal Drawings

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

[Tamassia SIAM J. Comput. 1987]

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

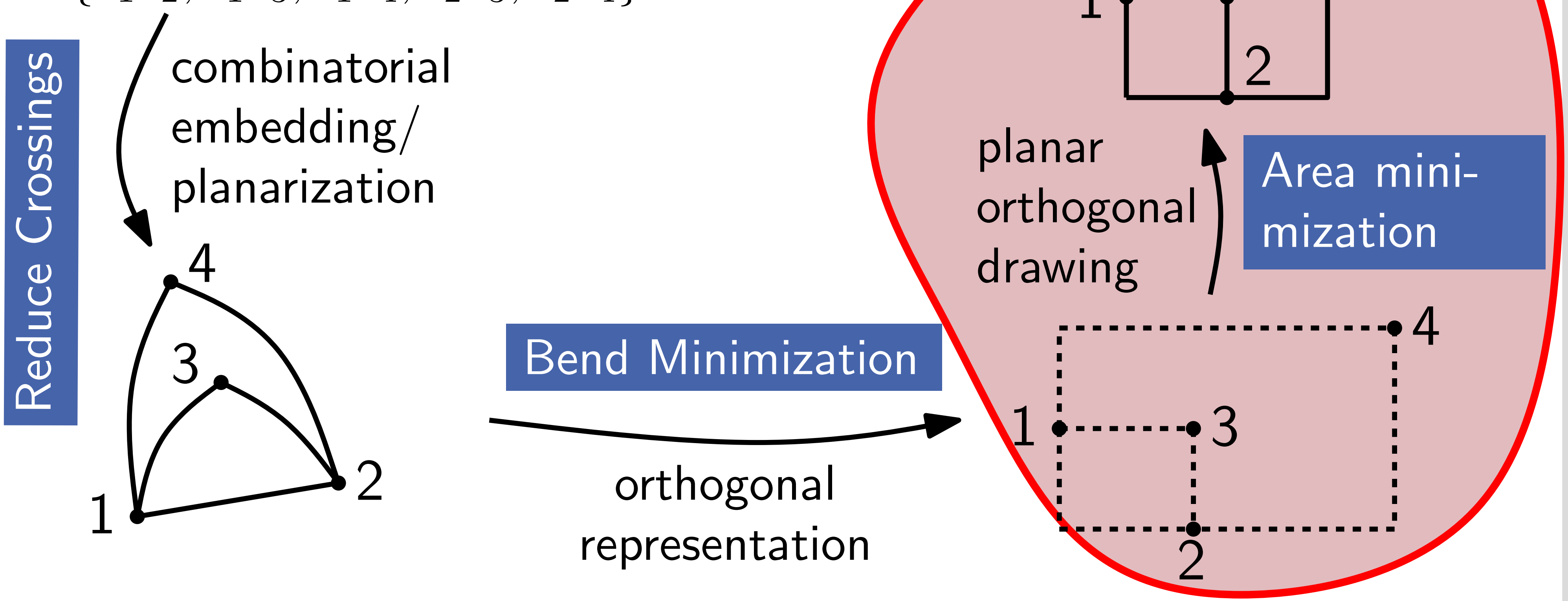


(Planar) Orthogonal Drawings

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

[Tamassia SIAM J. Comput. 1987]

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

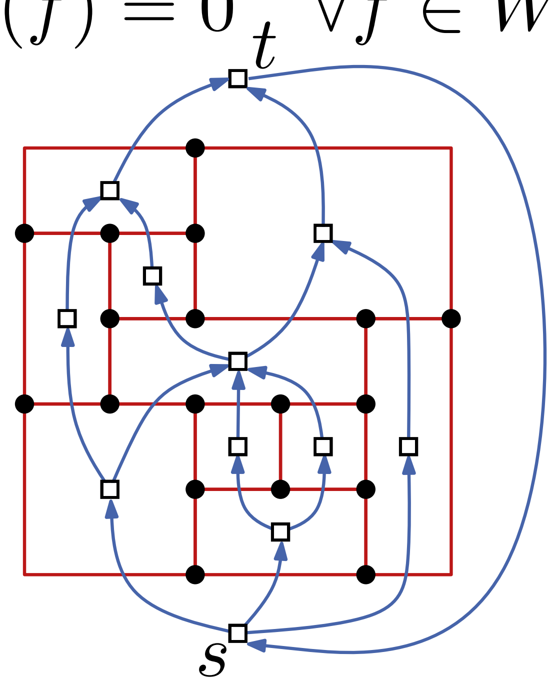


Flow Network for Edge Length Computation

Karlsruhe Institute of Technology

Def: Flow Network $N_{\text{hor}} = ((W_{\text{hor}}, A_{\text{hor}}); \ell; u; b; \text{cost})$

- $W_{\text{hor}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{hor}} = \{(f, g) \mid f, g$ share a horizontal segment and f lies below $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $u(a) = \infty \quad \forall a \in A_{\text{hor}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $b(f) = 0 \quad \forall f \in W_{\text{hor}}$

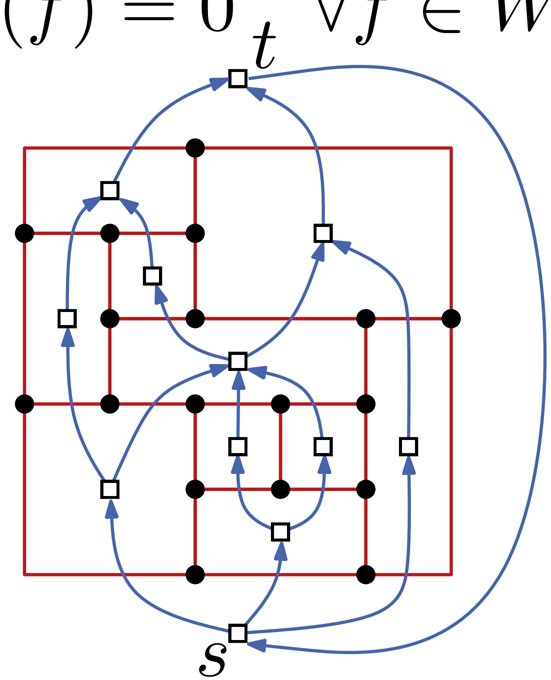


Flow Network for Edge Length Computation

Def: Flow Network $N_{\text{hor}} = ((W_{\text{hor}}, A_{\text{hor}}); \ell; u; b; \text{cost})$

- $W_{\text{hor}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{hor}} = \{(f, g) \mid f, g$ share a horizontal segment and f lies below $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $u(a) = \infty \quad \forall a \in A_{\text{hor}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{hor}}$
- $b(f) = 0 \quad \forall f \in W_{\text{hor}}$

s and t represent lower and upper side of f_0

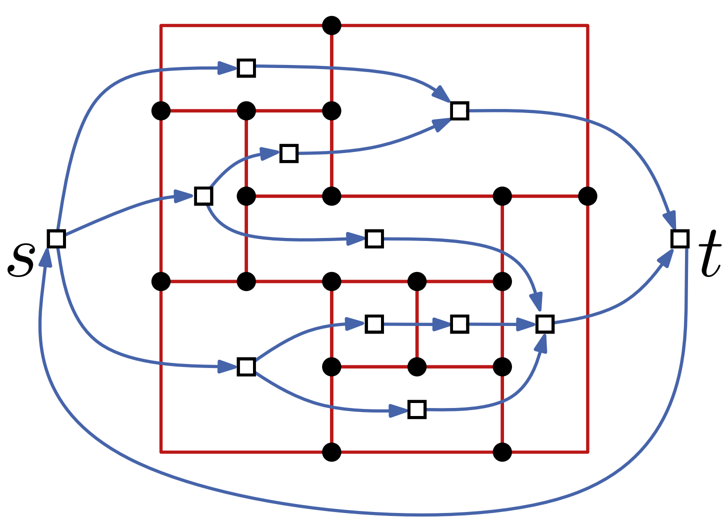


Flow Network for Edge Length Computation

Karlsruhe Institute of Technology

Def: Flow Network $N_{\text{ver}} = ((W_{\text{ver}}, A_{\text{ver}}); \ell; u; b; \text{cost})$

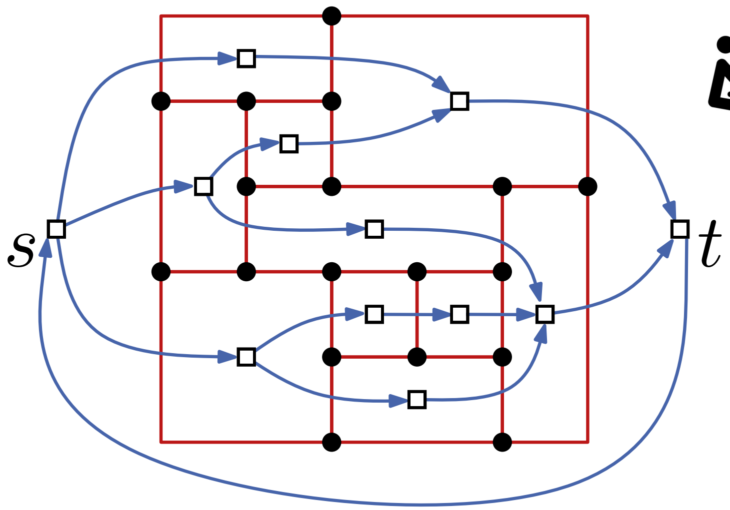
- $W_{\text{ver}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{ver}} = \{(f, g) \mid f, g$ share a vertical segment and f lies to the left of $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $u(a) = \infty \quad \forall a \in A_{\text{ver}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $b(f) = 0 \quad \forall f \in W_{\text{ver}}$



Flow Network for Edge Length Computation

Def: Flow Network $N_{\text{ver}} = ((W_{\text{ver}}, A_{\text{ver}}); \ell; u; b; \text{cost})$

- $W_{\text{ver}} = \mathcal{F} \setminus \{f_0\} \cup \{s, t\}$
- $A_{\text{ver}} = \{(f, g) \mid f, g$ share a vertical segment and f lies to the left of $g\} \cup \{(t, s)\}$
- $\ell(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $u(a) = \infty \quad \forall a \in A_{\text{ver}}$
- $\text{cost}(a) = 1 \quad \forall a \in A_{\text{ver}}$
- $b(f) = 0 \quad \forall f \in W_{\text{ver}}$

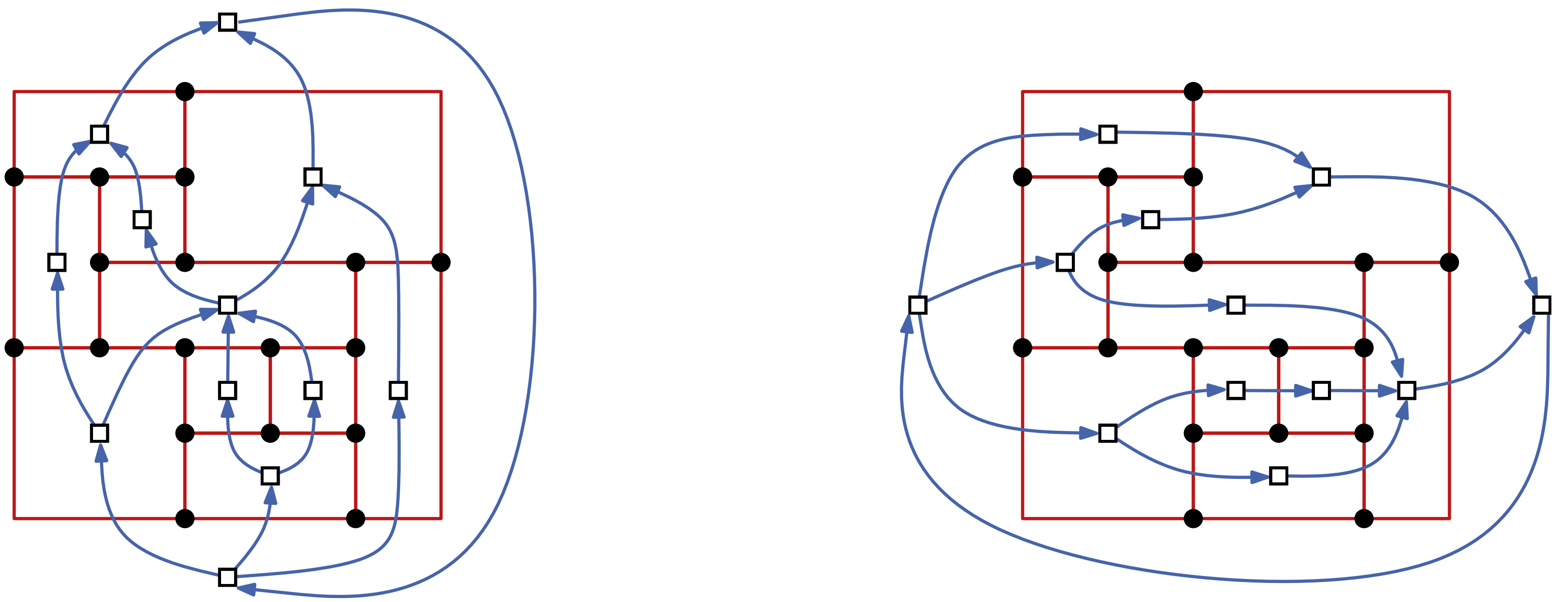


Pair, think, share: **3 min**

What values of the drawing represent the following?

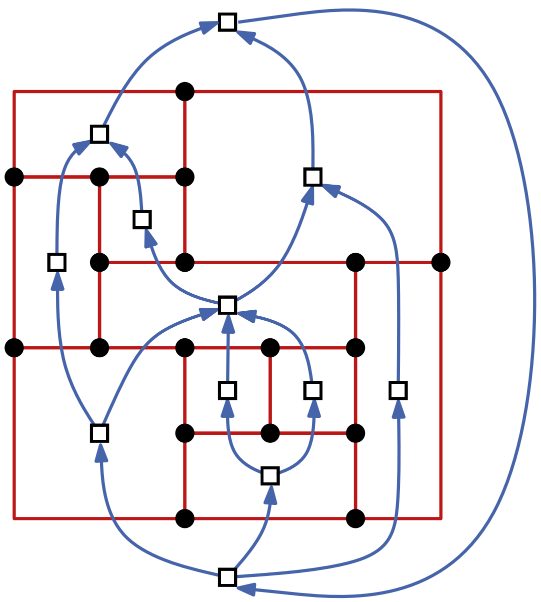
- $|X_{\text{hor}}(t, s)|$ and $|X_{\text{ver}}(t, s)|$?
- $\sum_{a \in A_{\text{hor}}} X_{\text{hor}}(a) + \sum_{a \in A_{\text{ver}}} X_{\text{ver}}(a)$

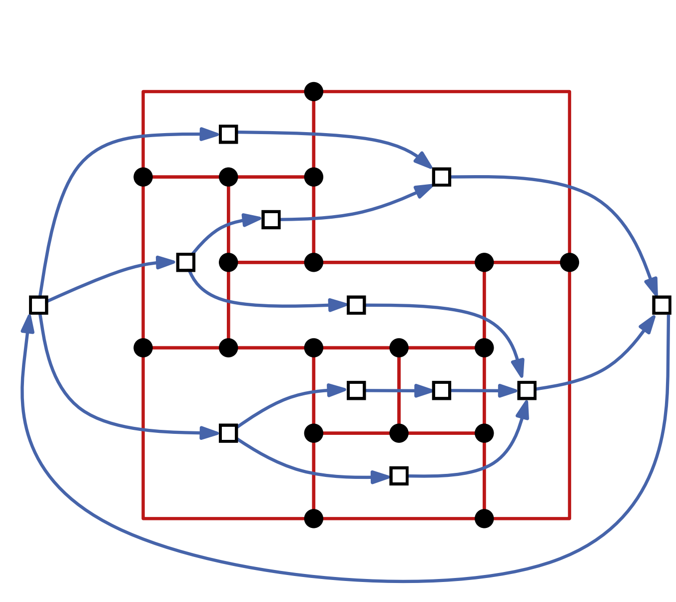
Optimal Layout



Thm 2: Integer flows X_{hor} and X_{ver} in N_{hor} and N_{ver} with minimum cost induce a valid orthogonal layout with minimum total edge length. The layout can be computed in $O(n^{3/2})$ time.

Faster Flow Computation

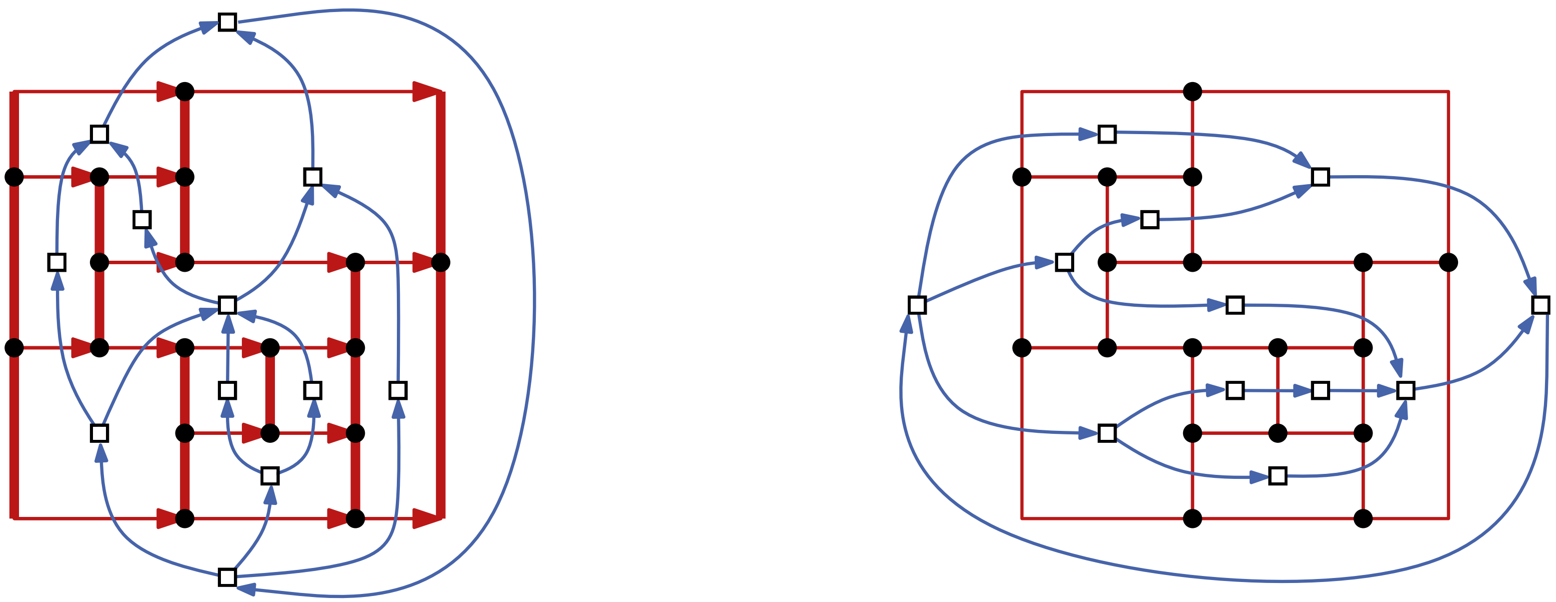




- construct the duals $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$
- for edge (f, g) of N_{hor} set flow $X_{\text{hor}}(f, g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f, g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

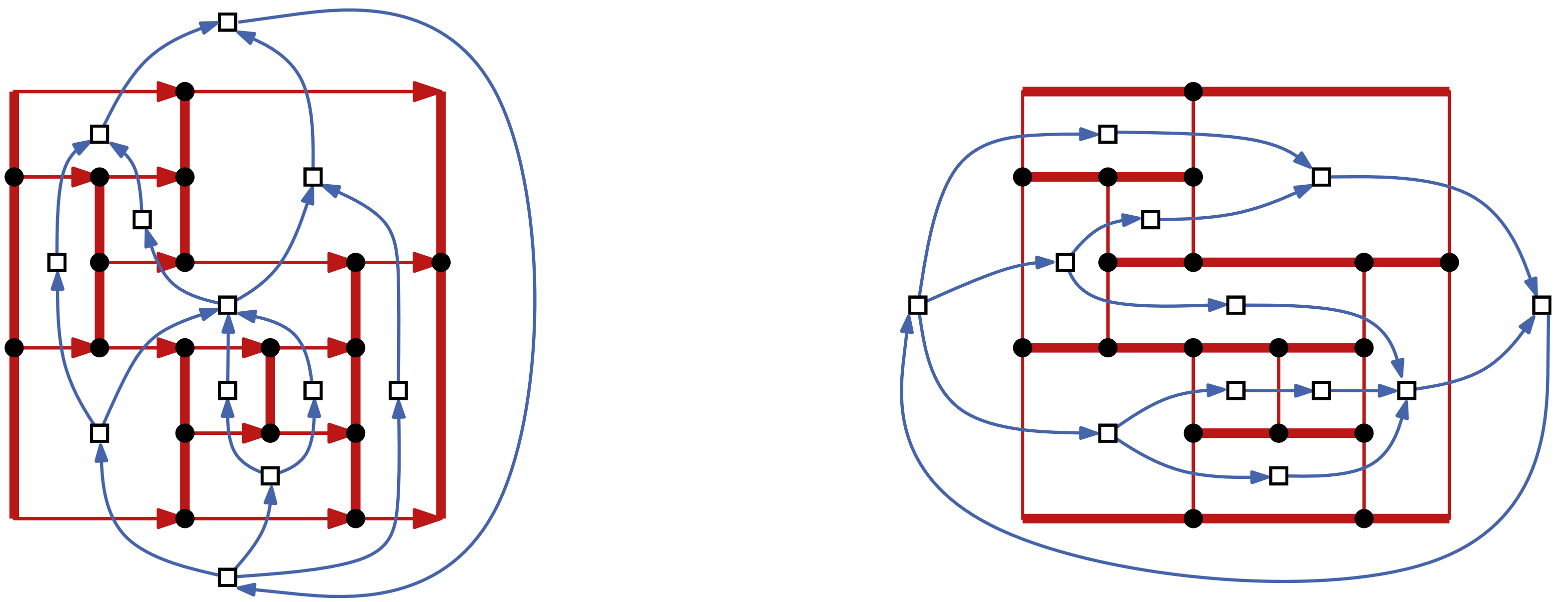
Faster Flow Computation

Karlsruhe Institute of Technology



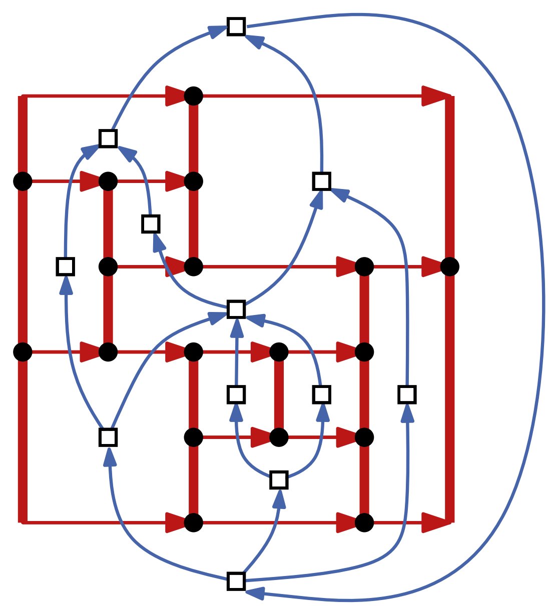
- construct the duals $N^{\star}_{\text{hor}}$ and $N^{\star}_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^{\star}_{\text{hor}}$ and $N^{\star}_{\text{ver}}$
- for edge (f, g) of N_{hor} set flow $X_{\text{hor}}(f, g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f, g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

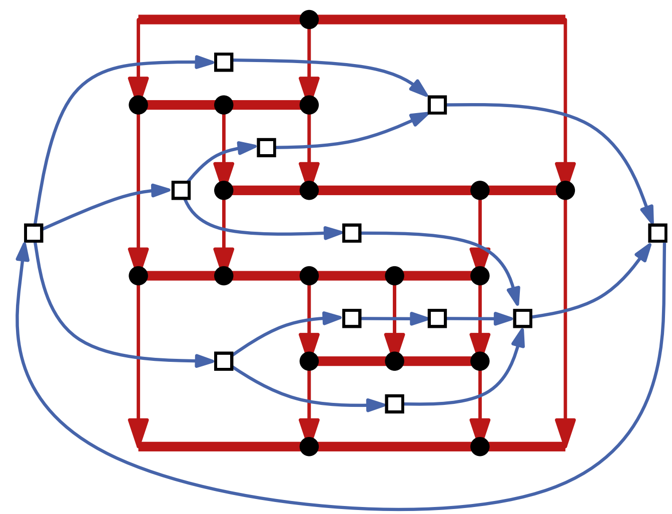
Faster Flow Computation



- construct the duals $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$
- for edge (f, g) of N_{hor} set flow $X_{\text{hor}}(f, g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f, g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

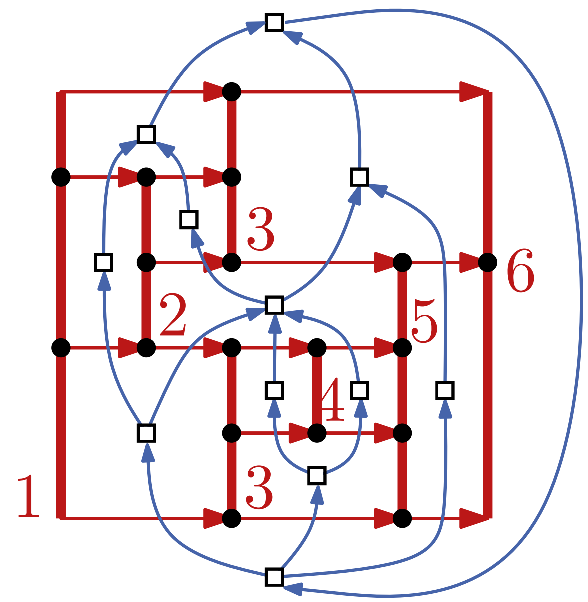
Faster Flow Computation

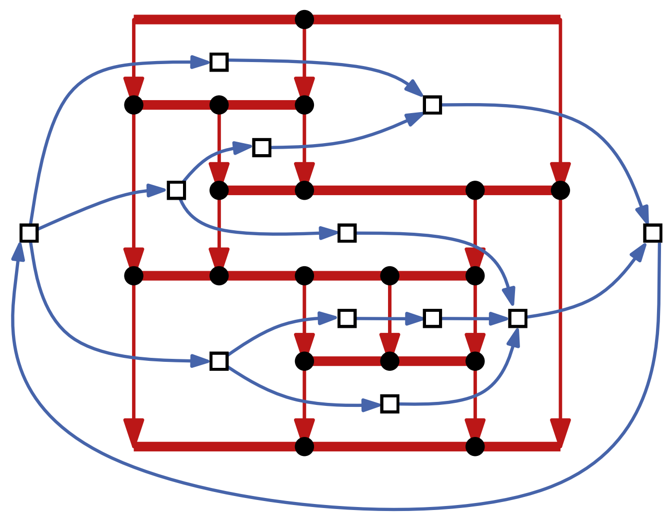




- construct the duals $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$
- for edge (f, g) of N_{hor} set flow $X_{\text{hor}}(f, g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f, g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

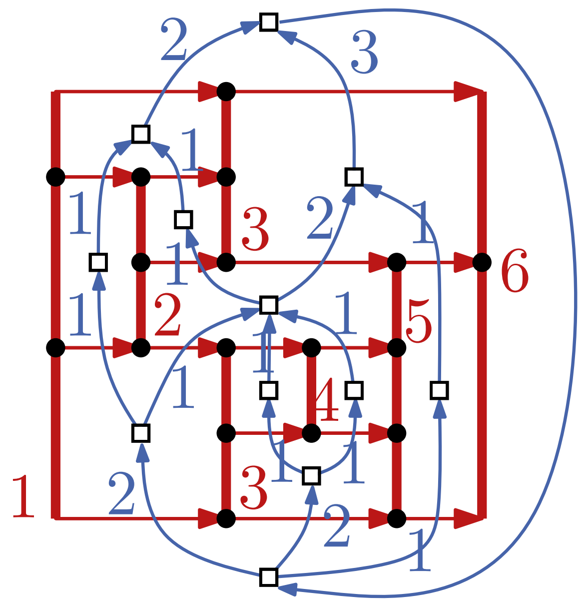
Faster Flow Computation

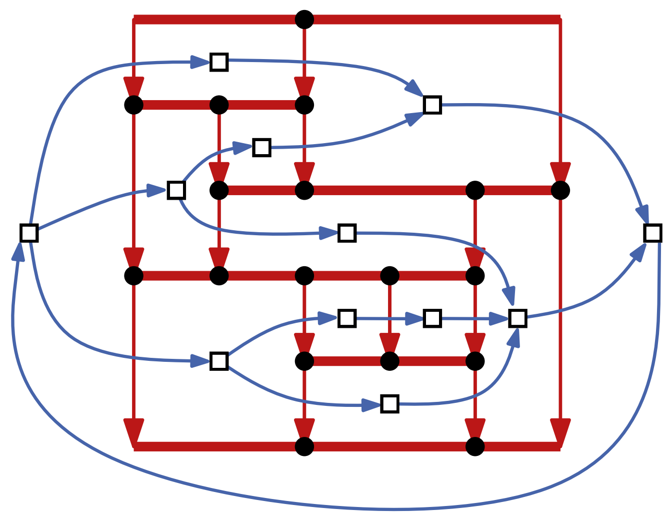




- construct the duals $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^\star_{\text{hor}}$ and $N^\star_{\text{ver}}$
- for edge (f, g) of N_{hor} set flow $X_{\text{hor}}(f, g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f, g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

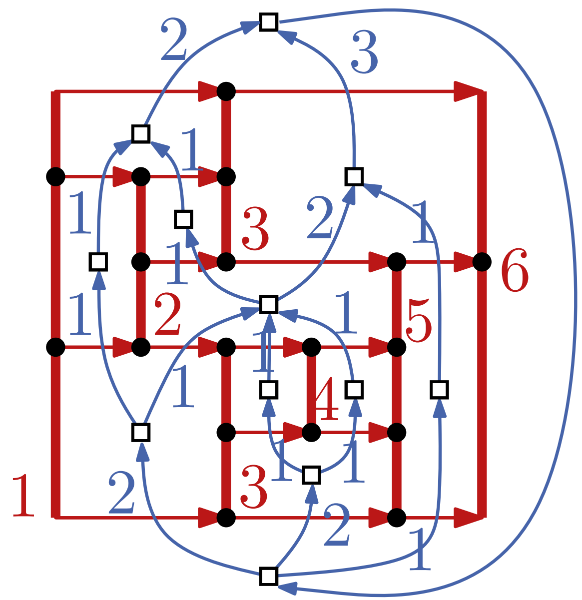
Faster Flow Computation

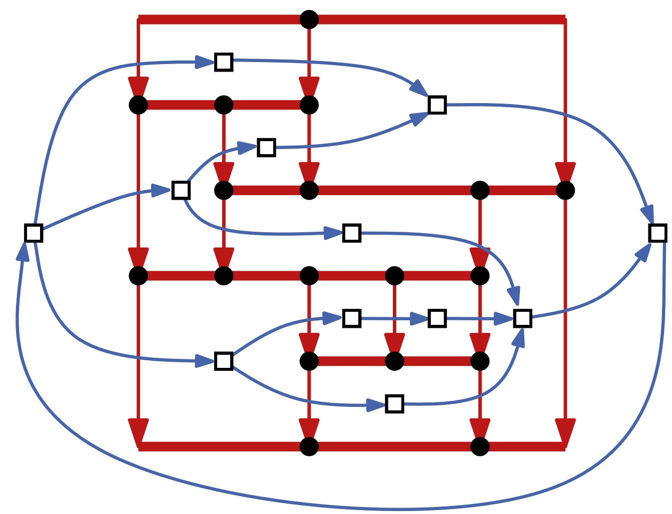




- construct the duals $N^{\star}_{\text{hor}}$ and $N^{\star}_{\text{ver}}$ of N_{hor} and N_{ver}
- topologial numbering T_{hor} and T_{ver} of $N^{\star}_{\text{hor}}$ and $N^{\star}_{\text{ver}}$
- for edge (f,g) of N_{hor} set flow $X_{\text{hor}}(f,g) = T_{\text{hor}}(u) - T_{\text{hor}}(v)$, where u is dual vertex on the left and v is dual vertex on the right of (f,g), similar for X_{ver}
- the constructed functions X_{hor}, X_{ver} have minimum cost

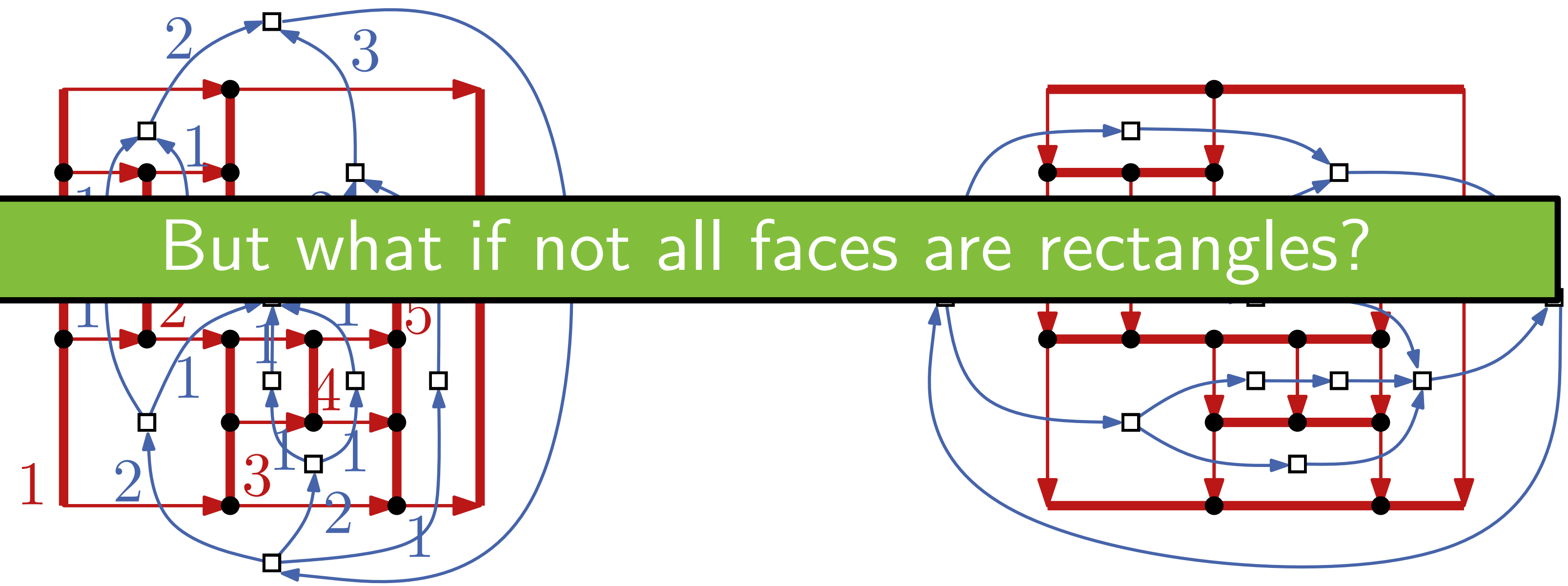
Faster Flow Computation





- This approach finds minimum width, height, area, but does not guarantee minimum total edge length
- Time complexity $O(n)$

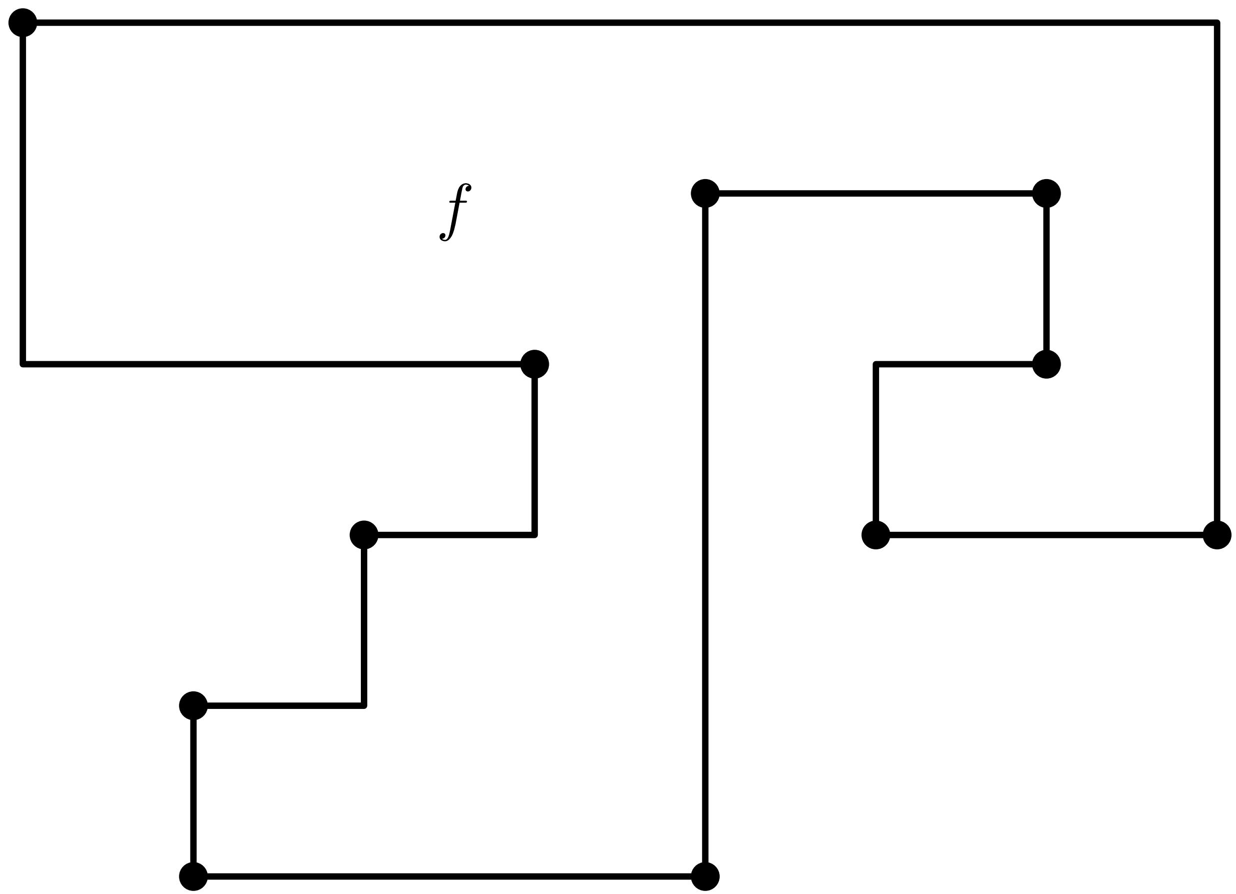
Faster Flow Computation



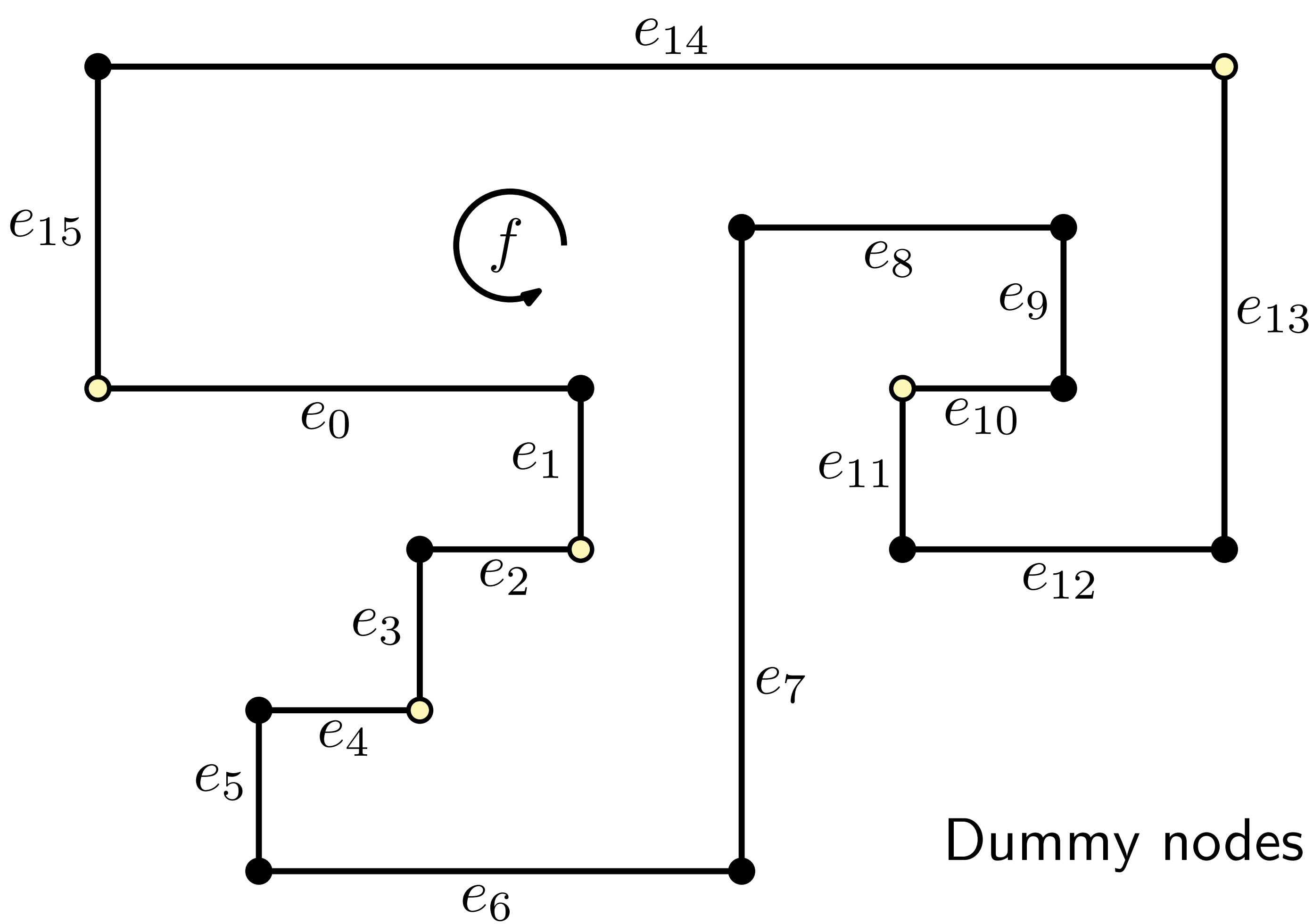
But what if not all faces are rectangles?

- This approach finds minimum width, height, area, but does not guarantee minimum total edge length
- Time complexity $O(n)$

Refinement of (G, H) – Inner Face

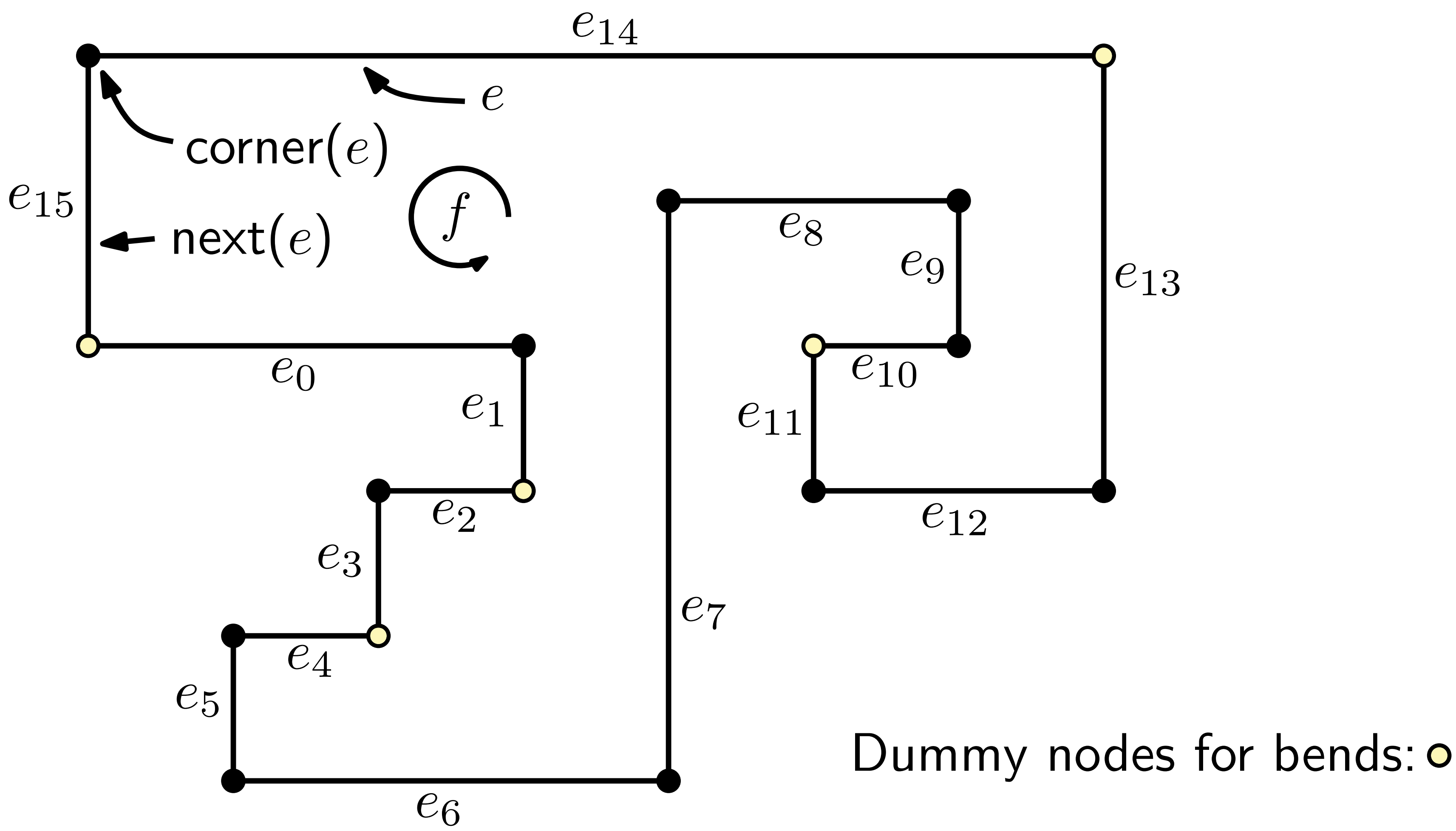


Refinement of (G, H) – Inner Face

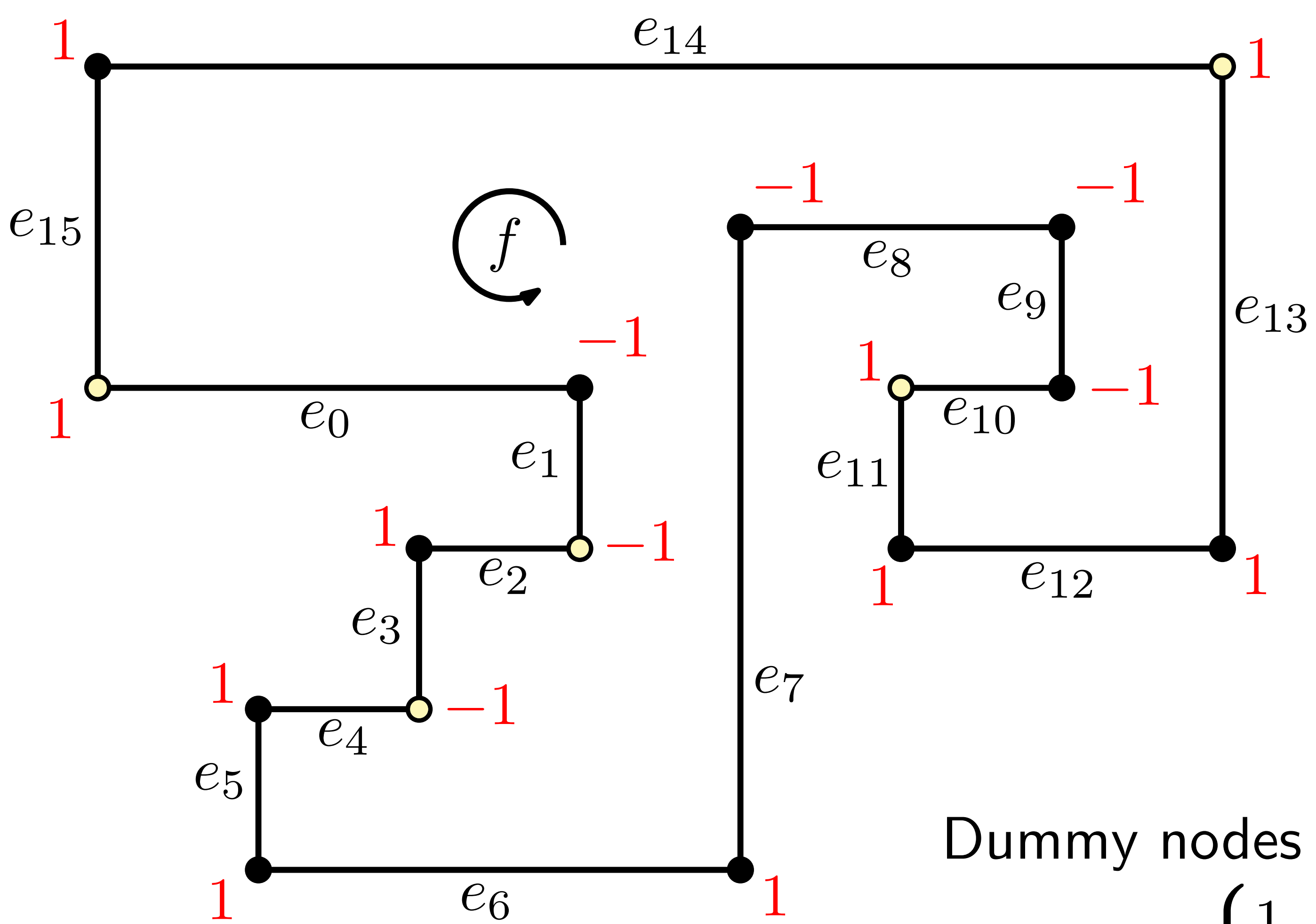


Dummy nodes for bends:

Refinement of (G, H) – Inner Face



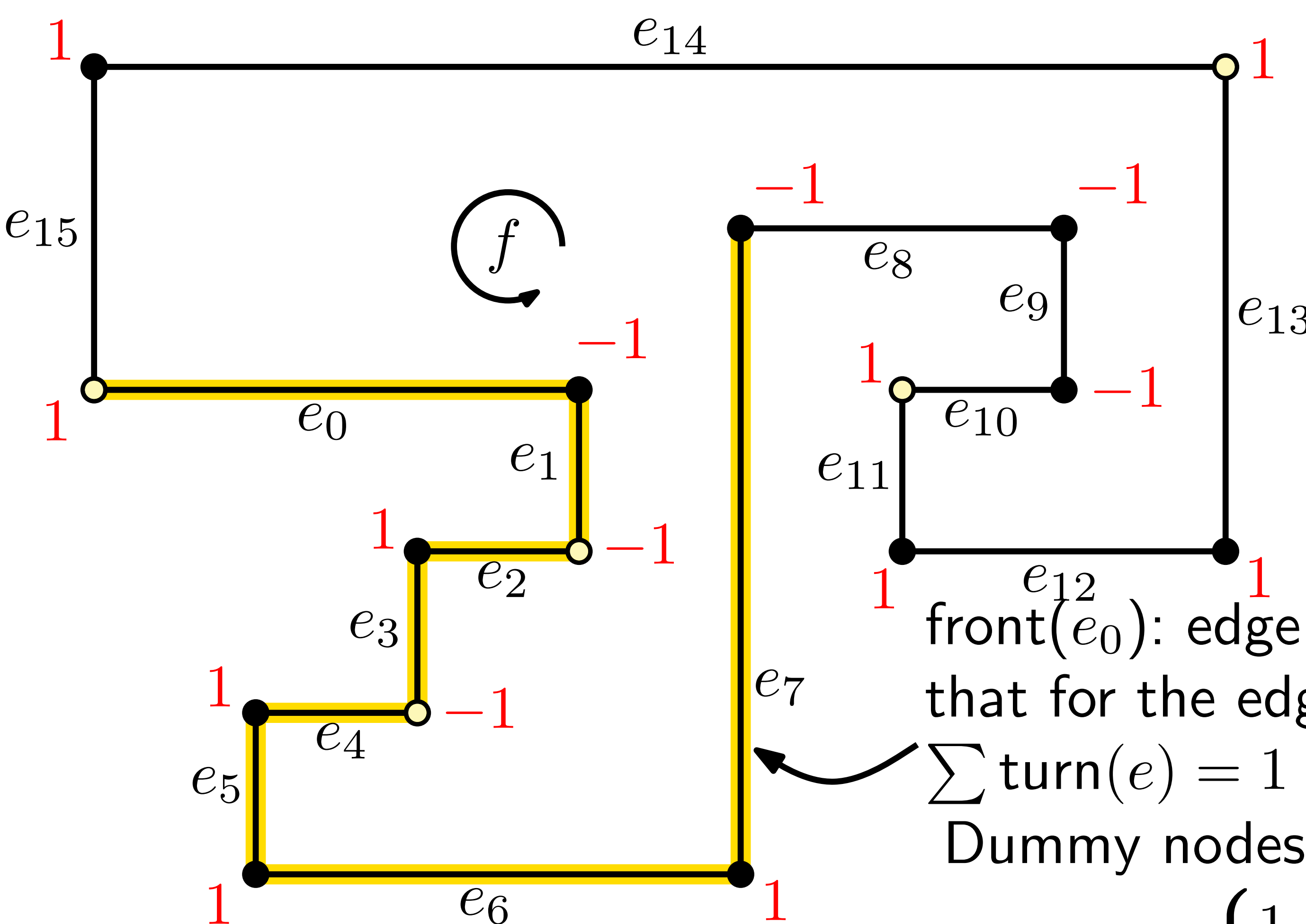
Refinement of (G, H) – Inner Face



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) – Inner Face

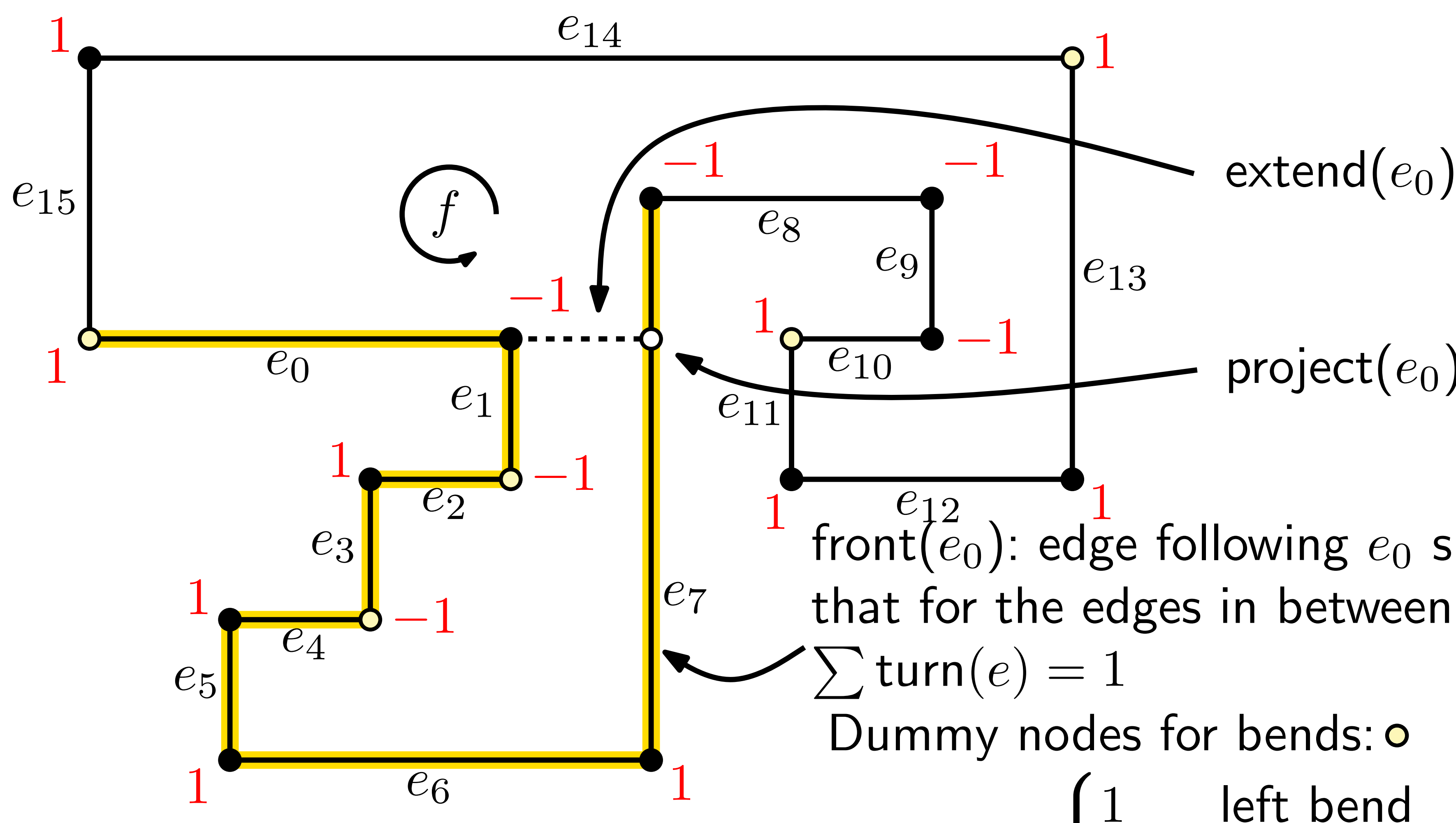


front(e_0): edge following e_0 such that for the edges in between $\sum \text{turn}(e) = 1$

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) – Inner Face

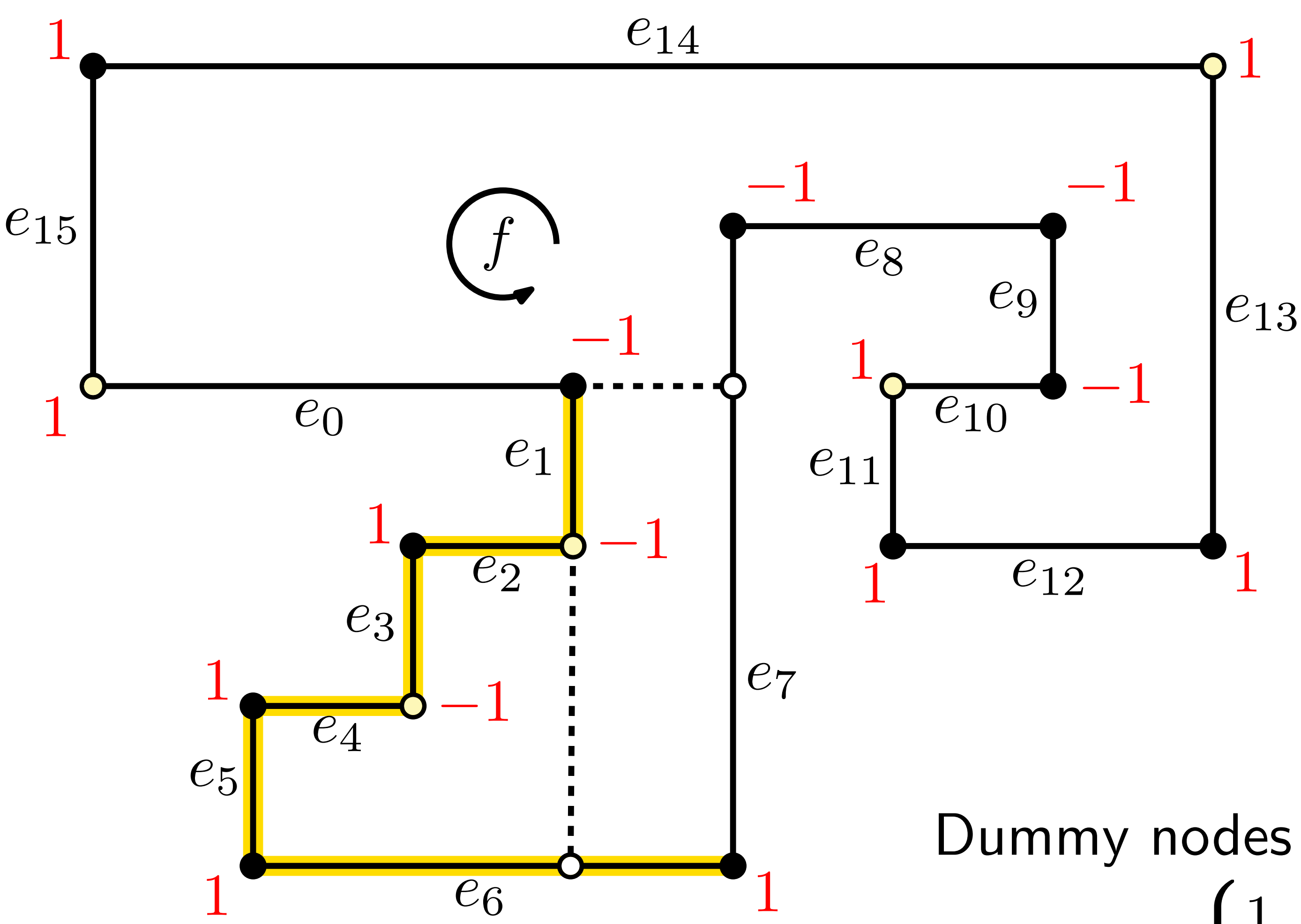


front(e_0): edge following e_0 such that for the edges in between $\sum$ turn$(e) = 1$

Dummy nodes for bends: ○

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

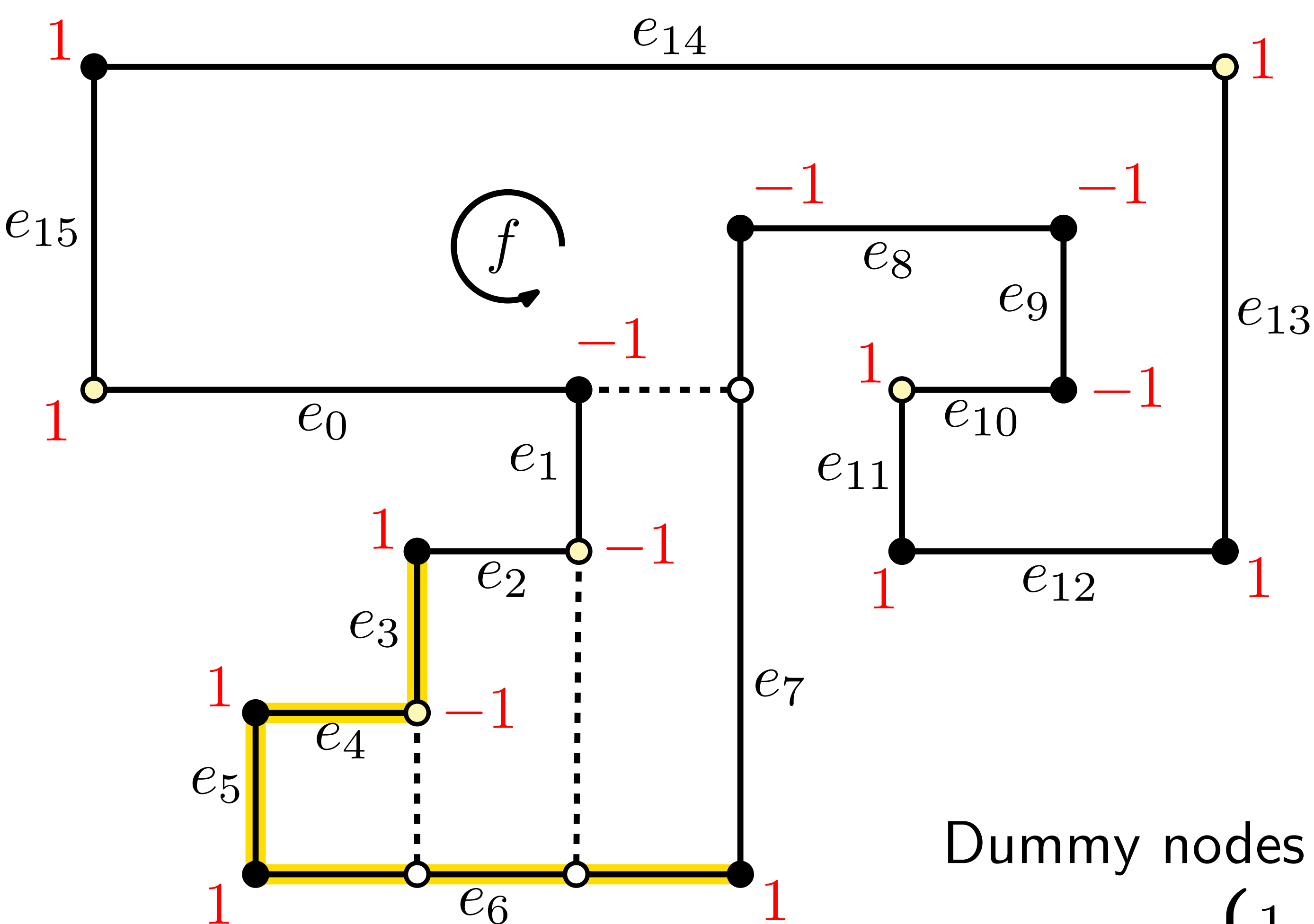
Refinement of (G, H) – Inner Face



Dummy nodes for bends:

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

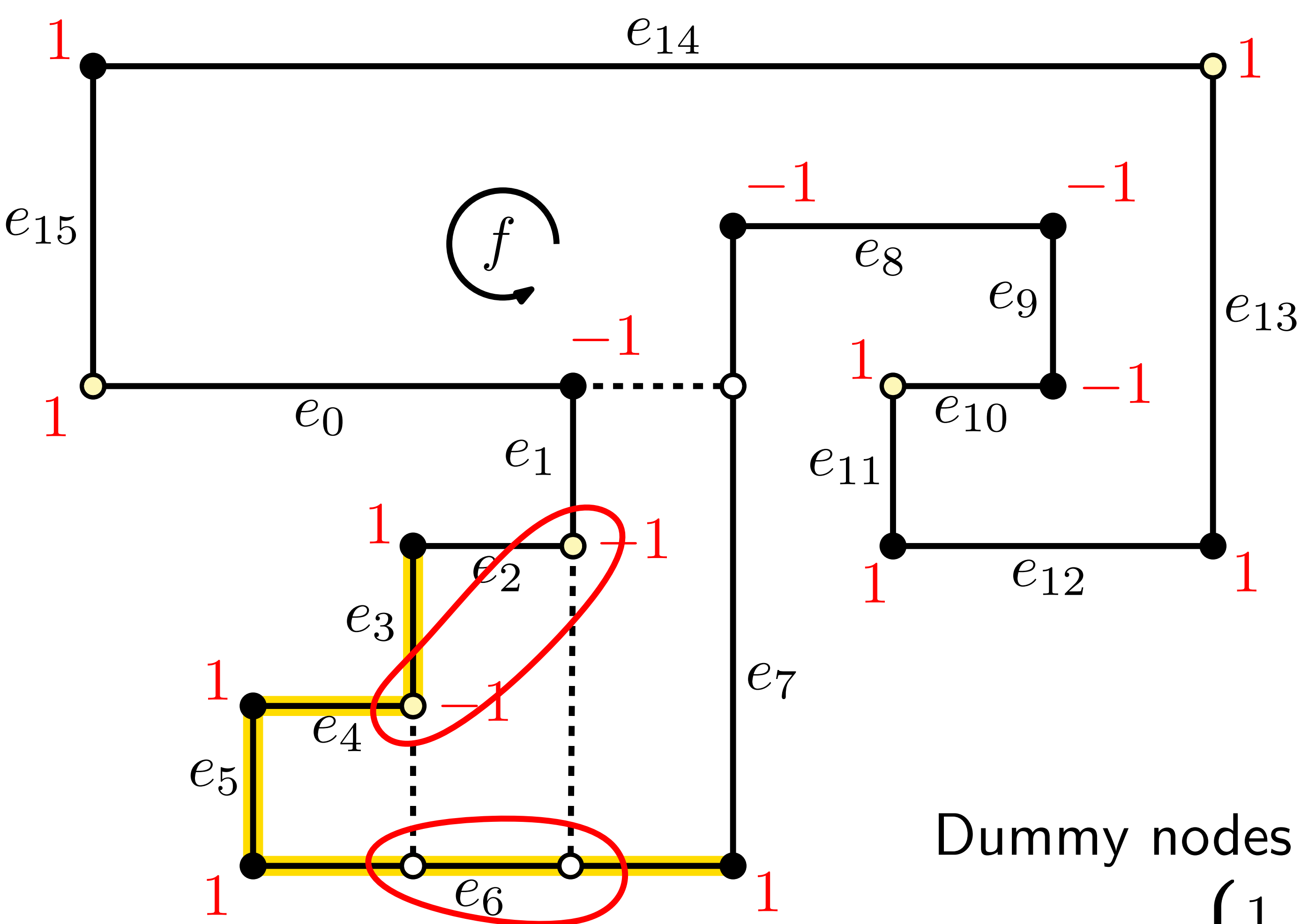
Refinement of (G, H) – Inner Face



Dummy nodes for bends: ●

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

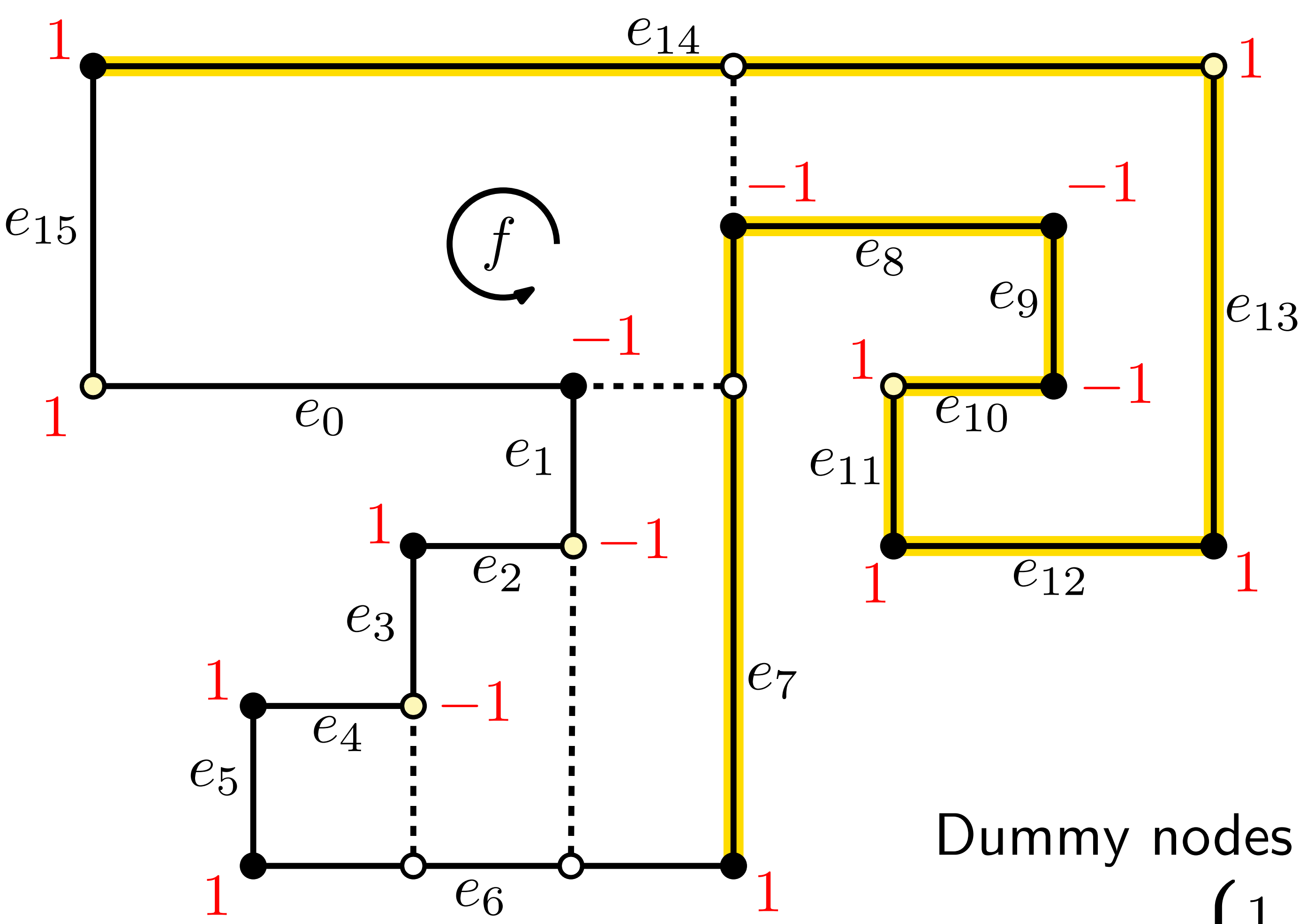
Refinement of (G, H) – Inner Face



Dummy nodes for bends:

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

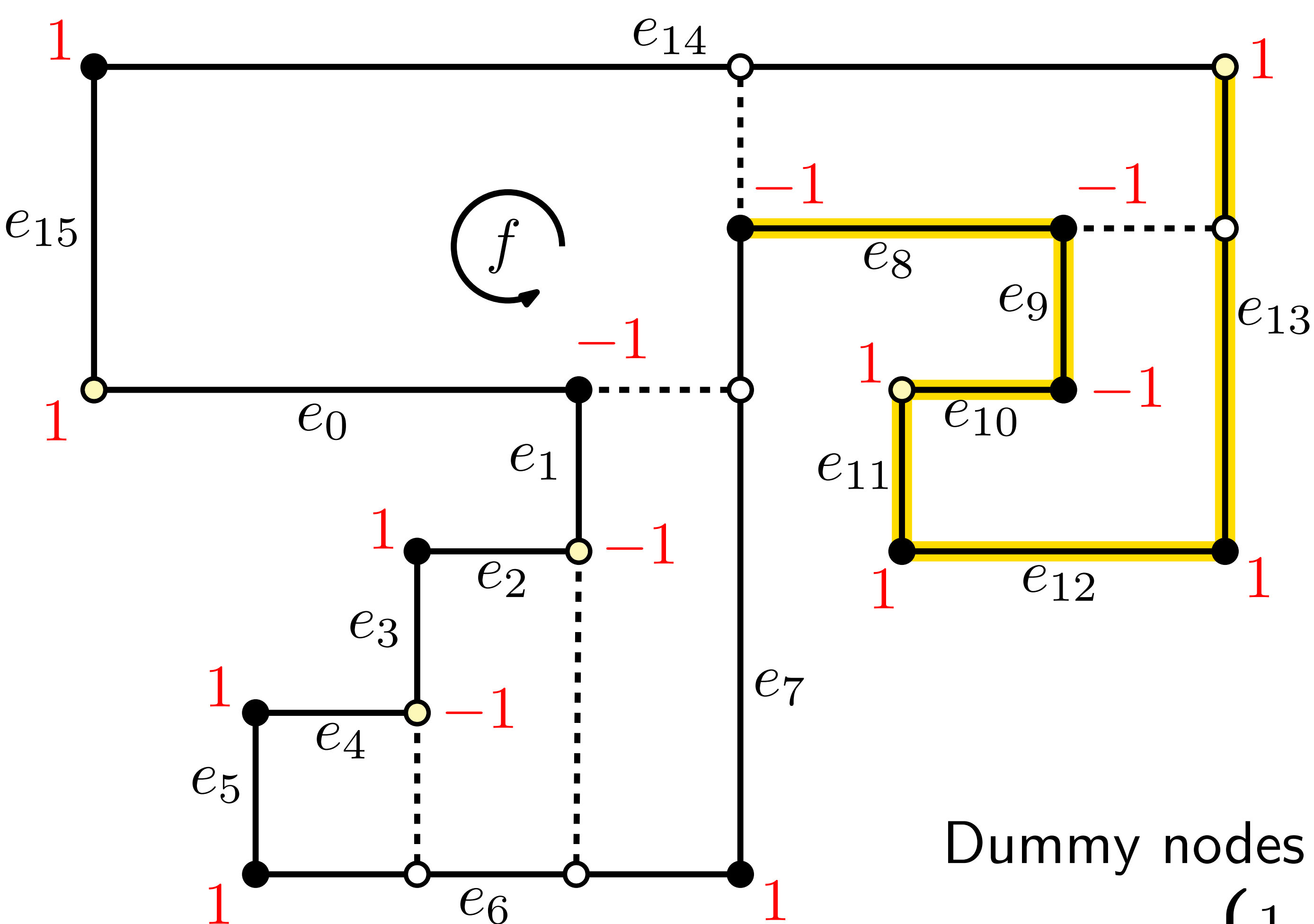
Refinement of (G, H) – Inner Face



Dummy nodes for bends:

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

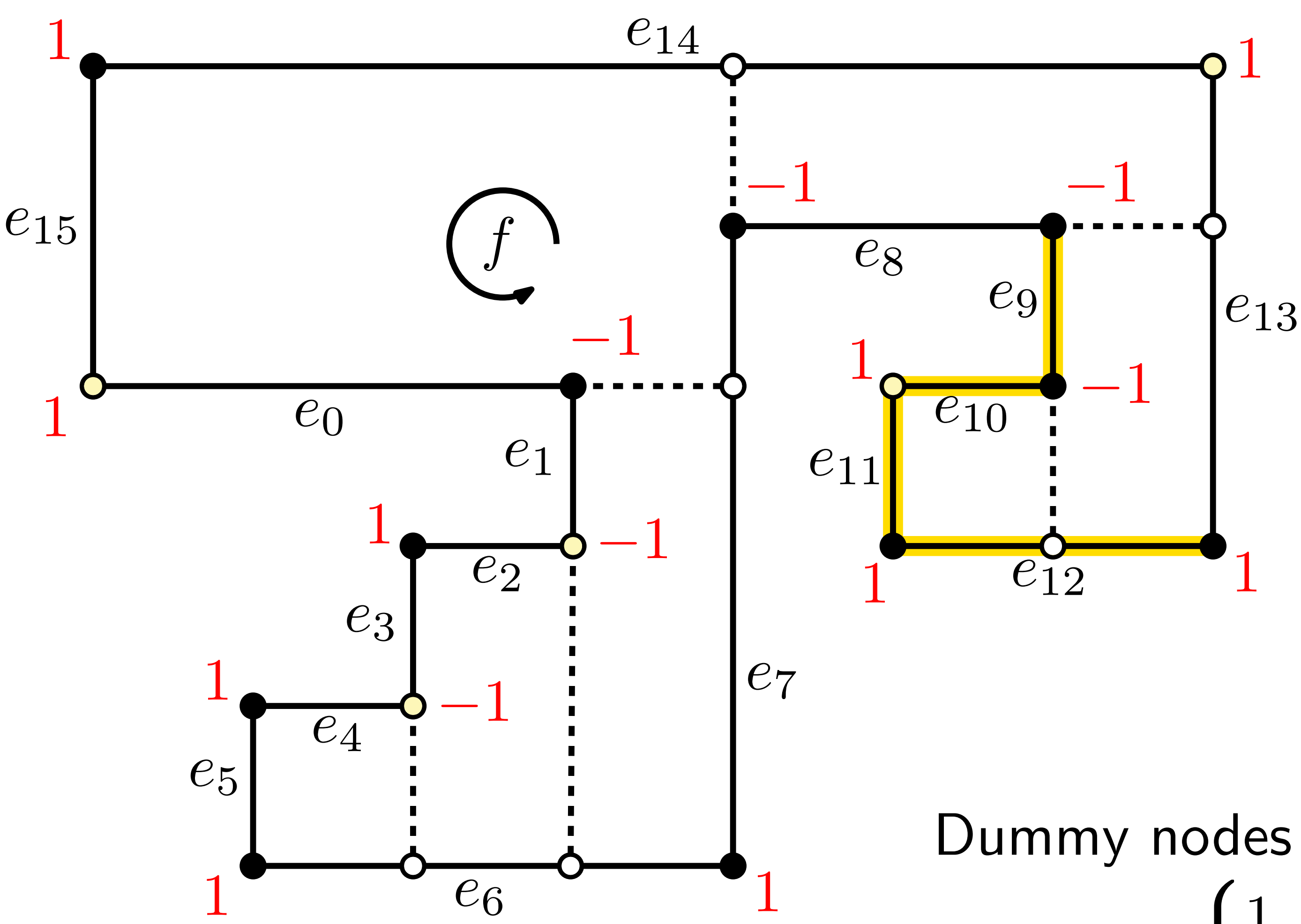
Refinement of (G, H) – Inner Face



Dummy nodes for bends:

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

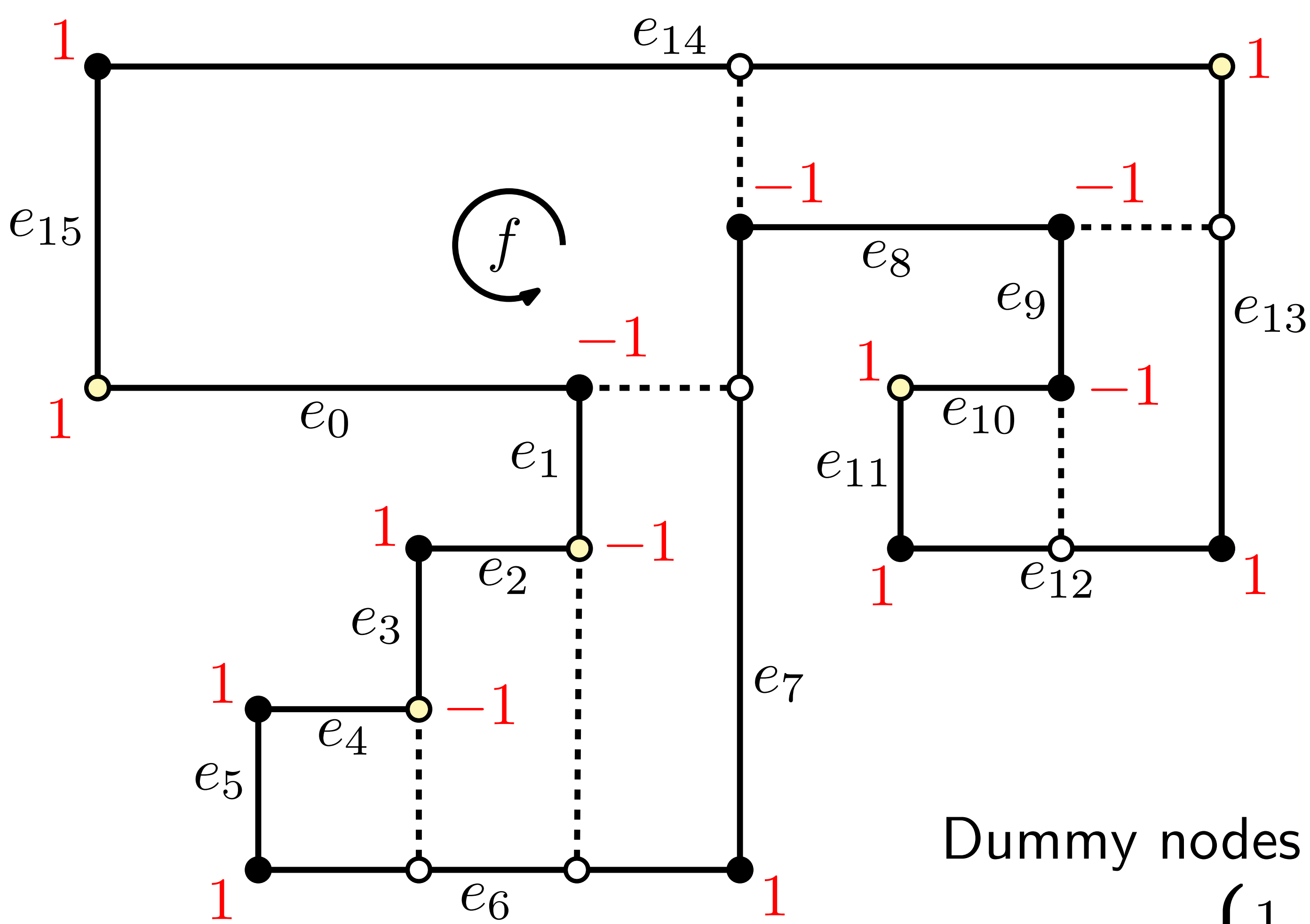
Refinement of (G, H) – Inner Face



Dummy nodes for bends:

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

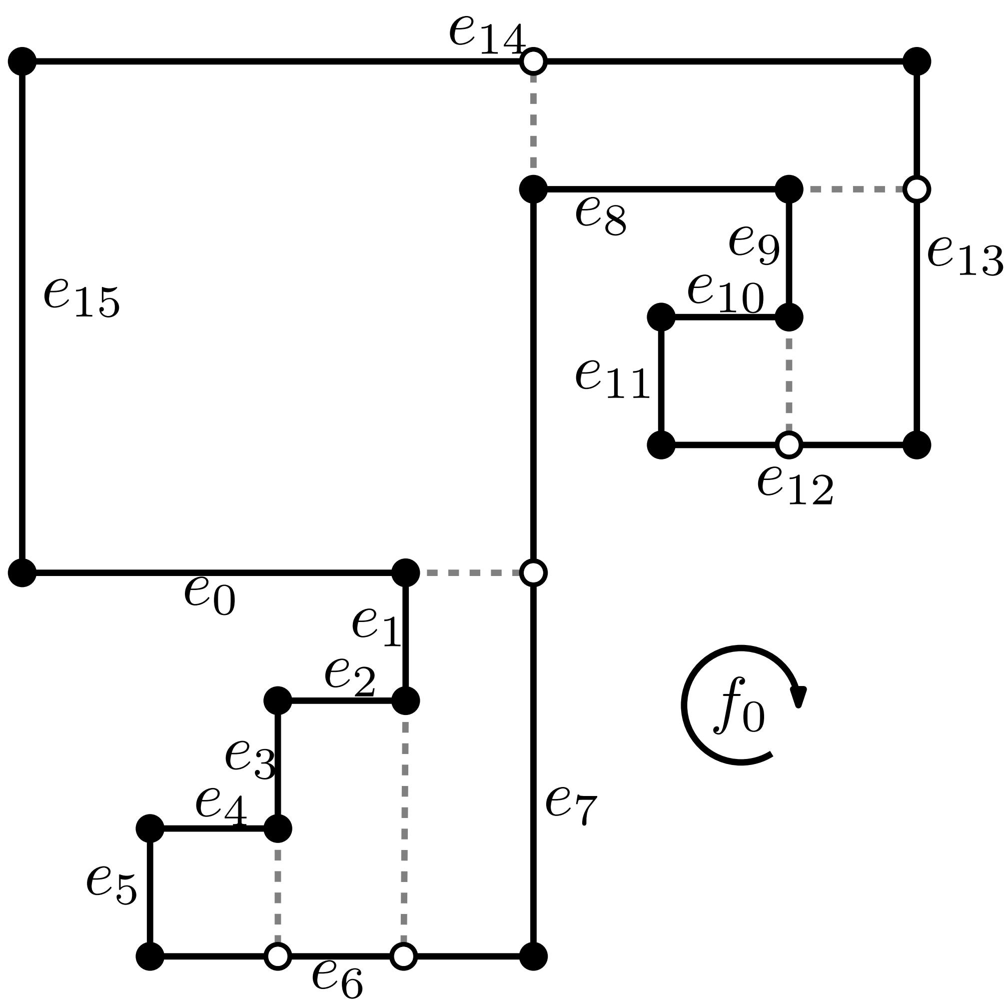
Refinement of (G, H) – Inner Face



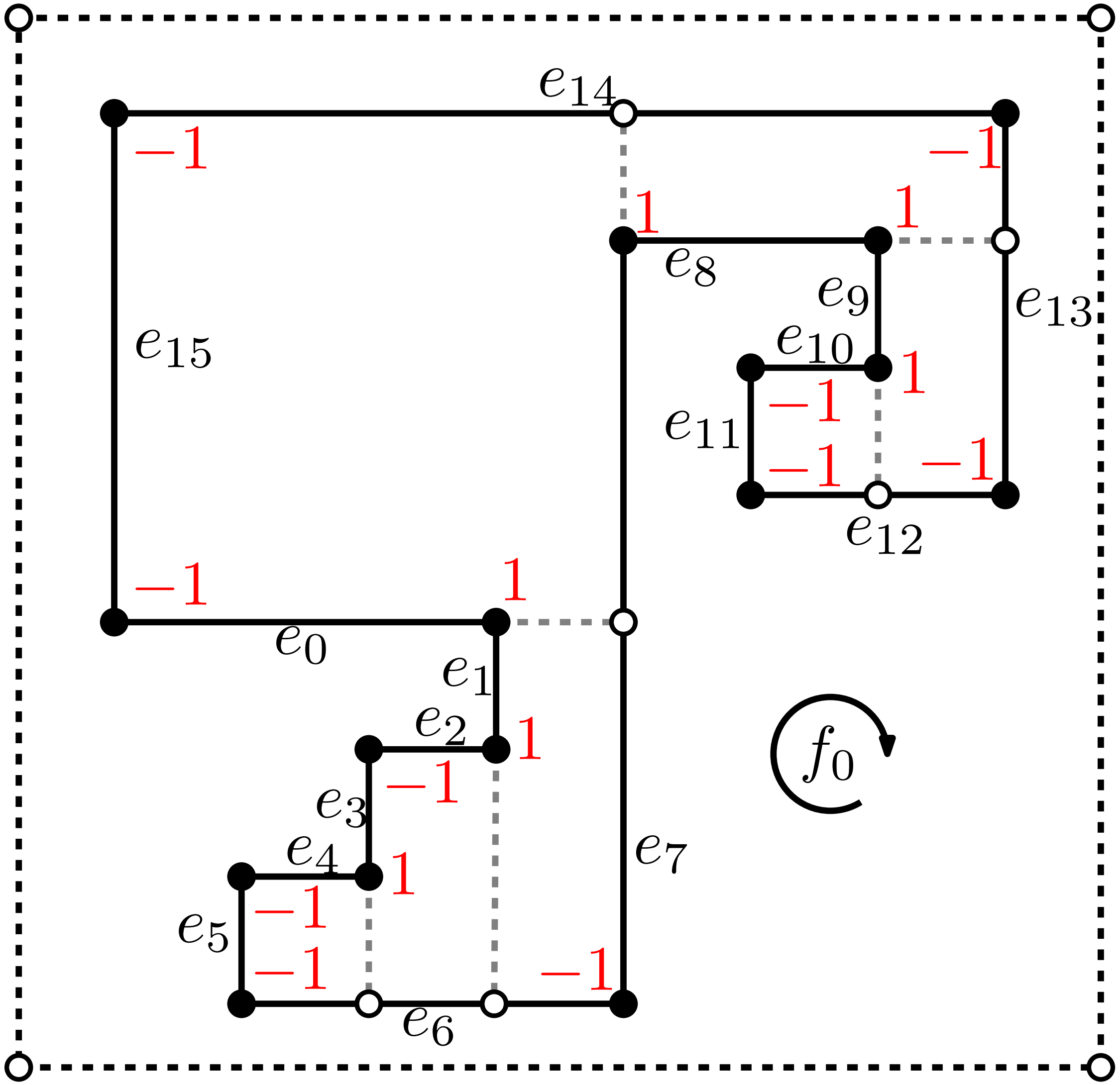
Dummy nodes for bends: ○

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

Refinement of (G, H) – Outer Face



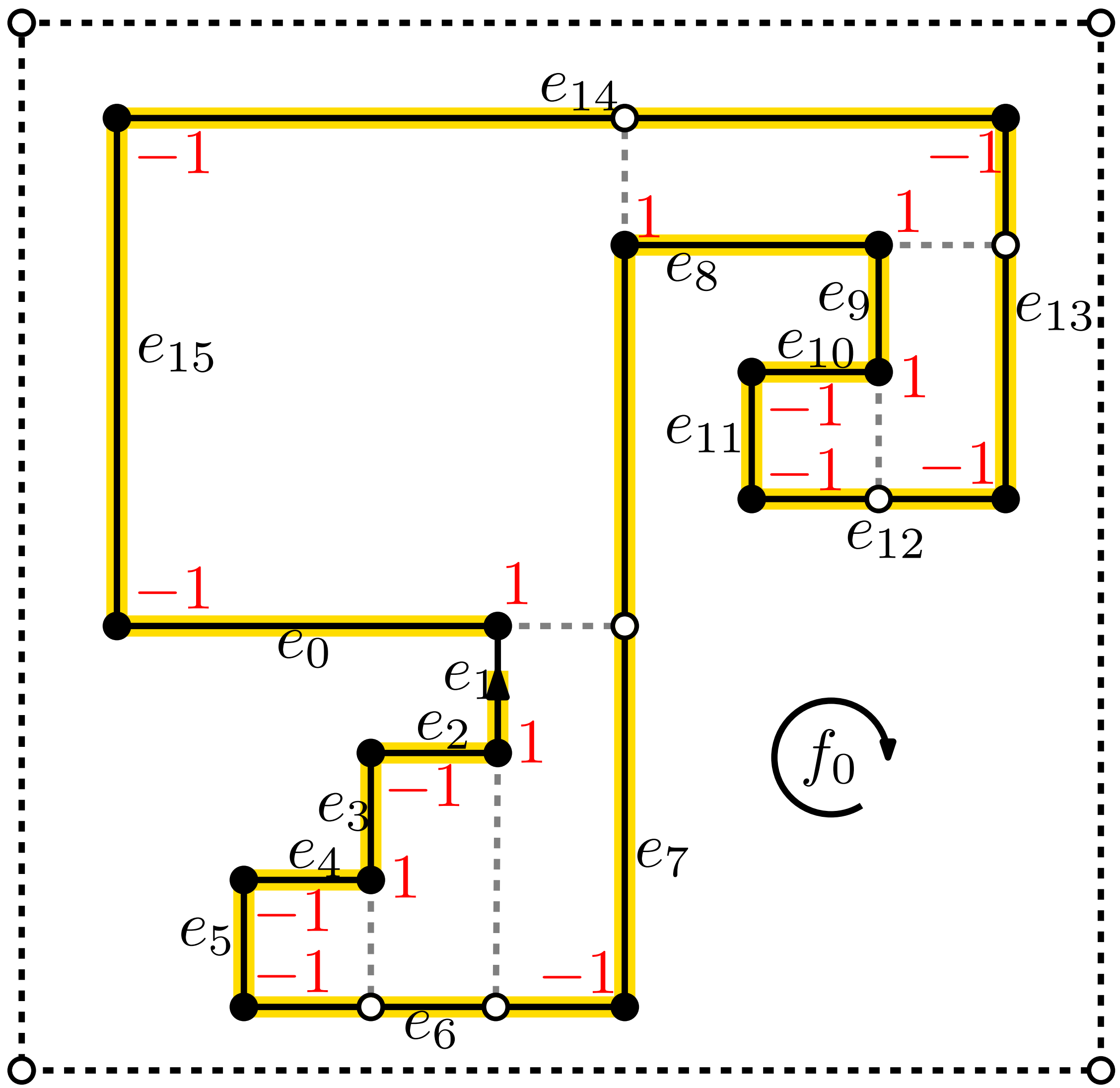
Refinement of (G, H) – Outer Face



- front(e) may be undefined

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

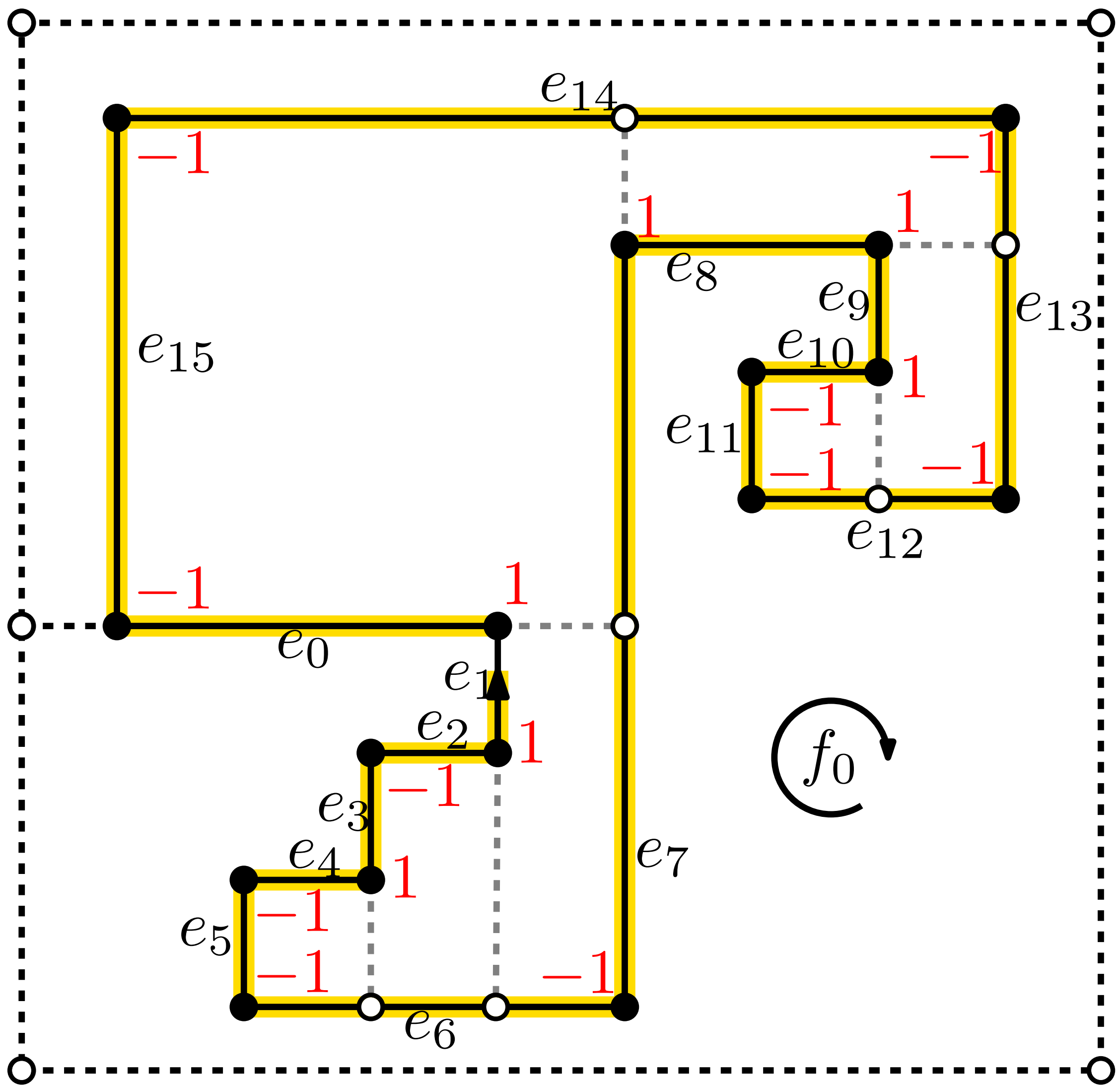
Refinement of (G, H) – Outer Face



- front(e) may be undefined

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

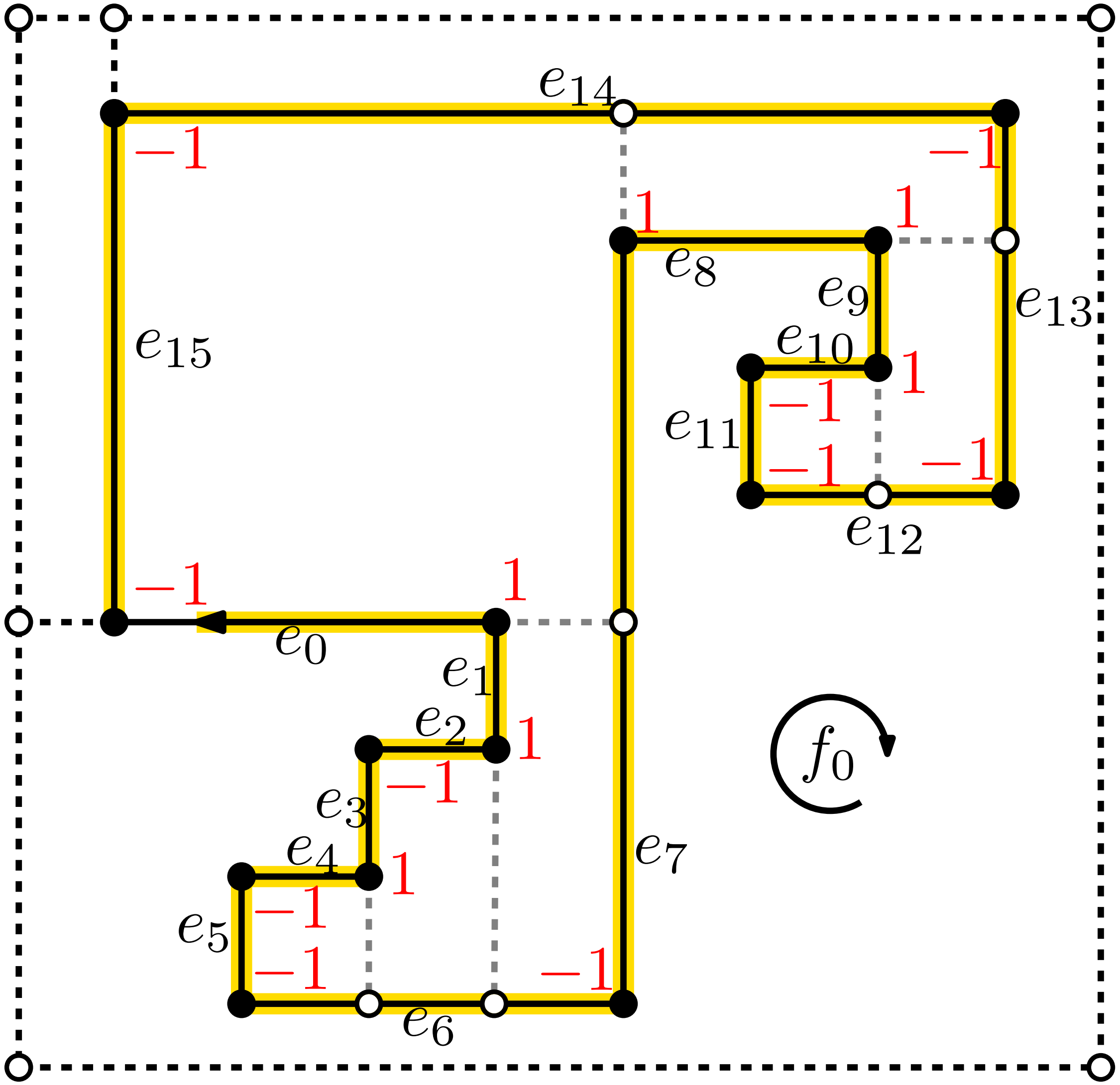
Refinement of (G, H) – Outer Face



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

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

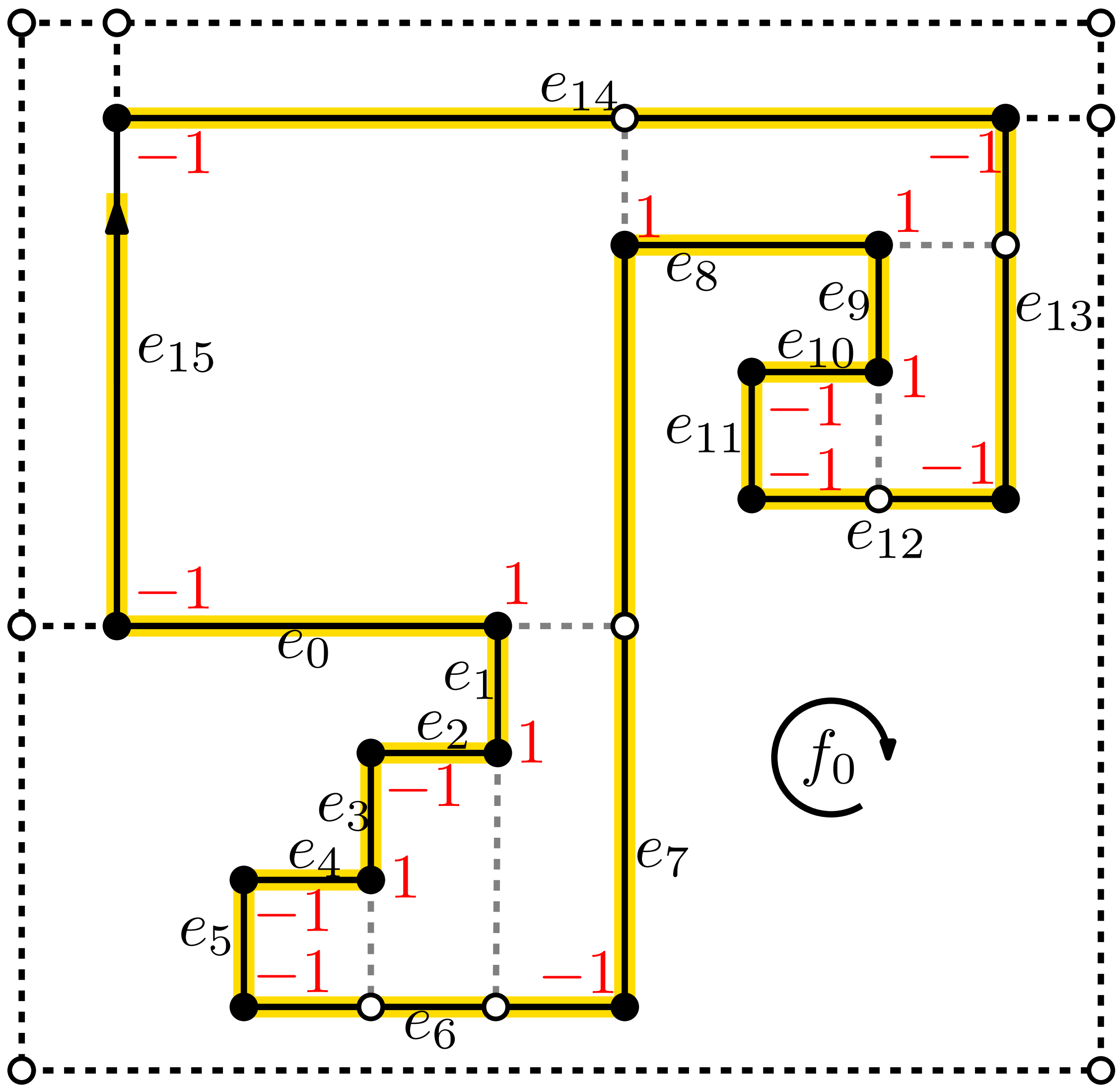
Refinement of (G, H) – Outer Face



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

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

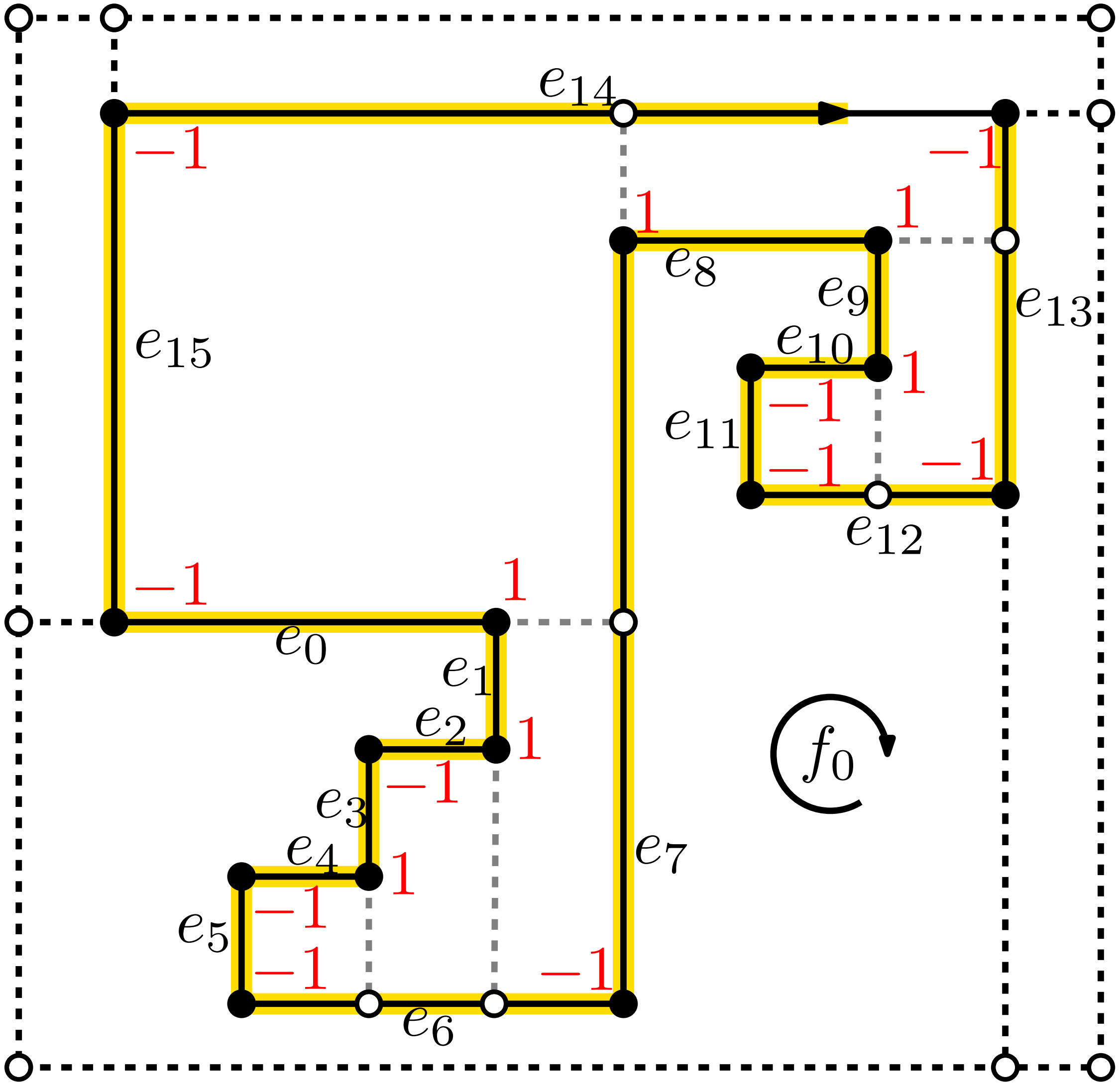
Refinement of (G, H) – Outer Face



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

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

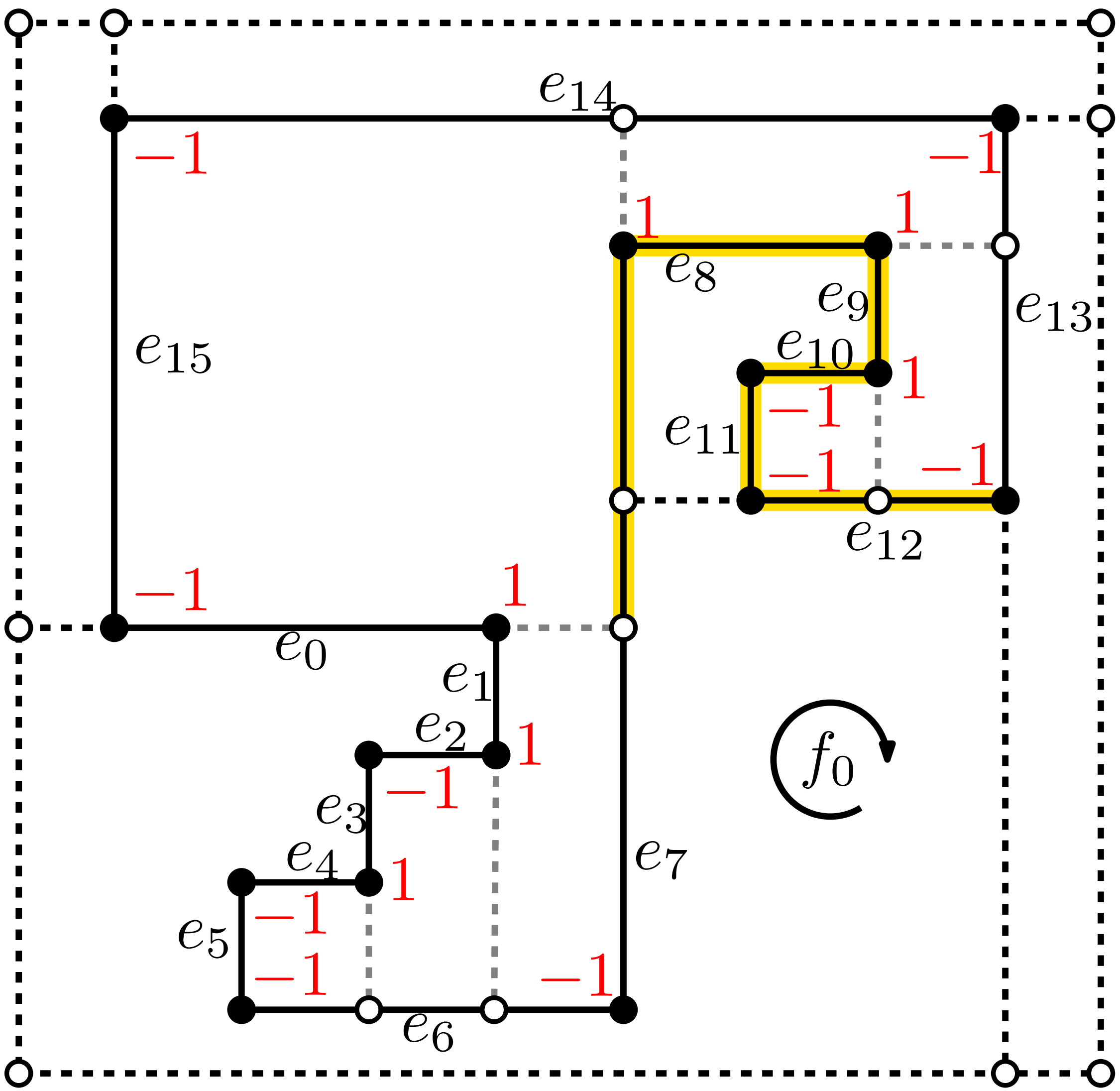
Refinement of (G, H) – Outer Face



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

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

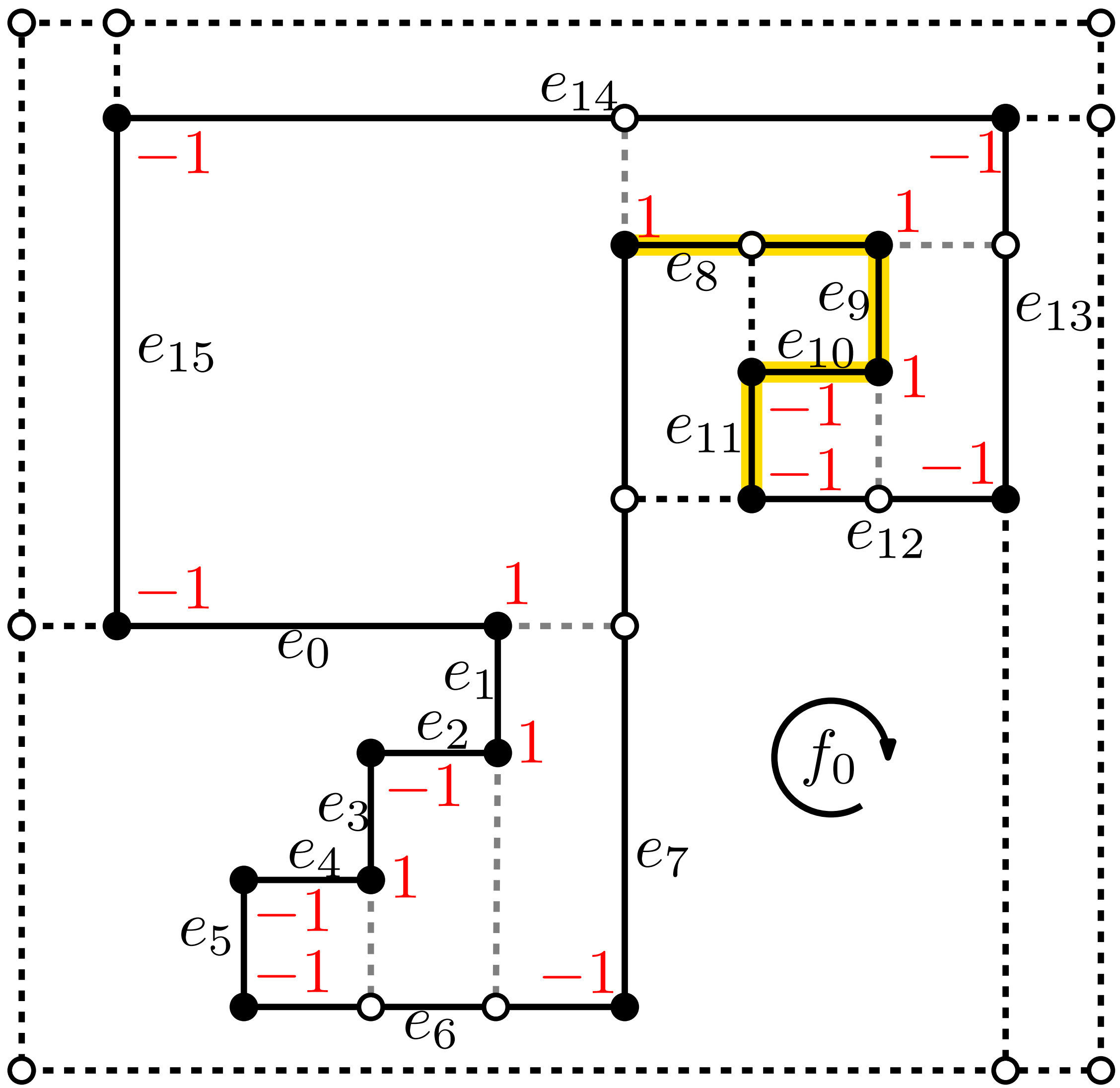
Refinement of (G, H) – Outer Face



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

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

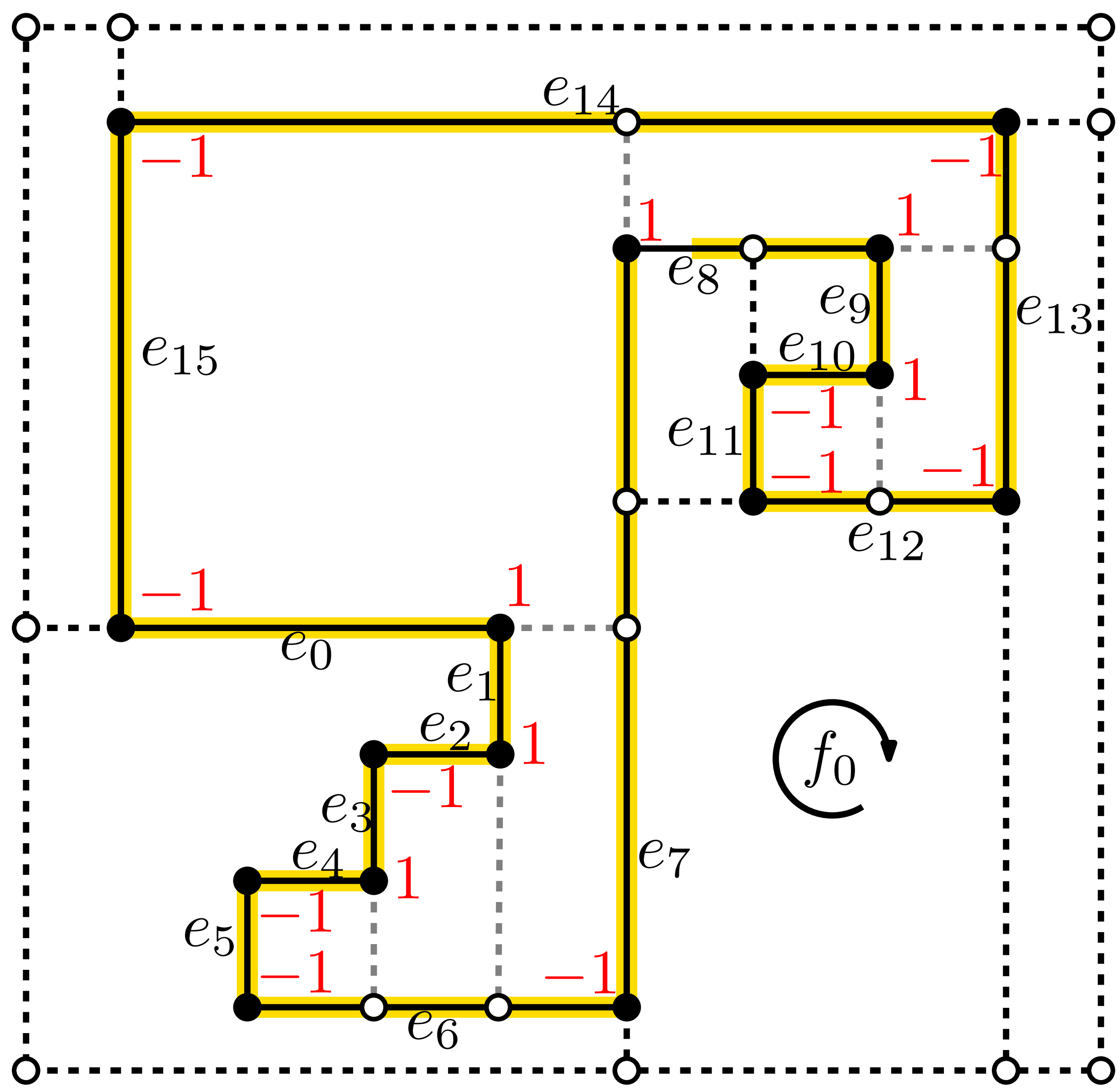
Refinement of (G, H) – Outer Face



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

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

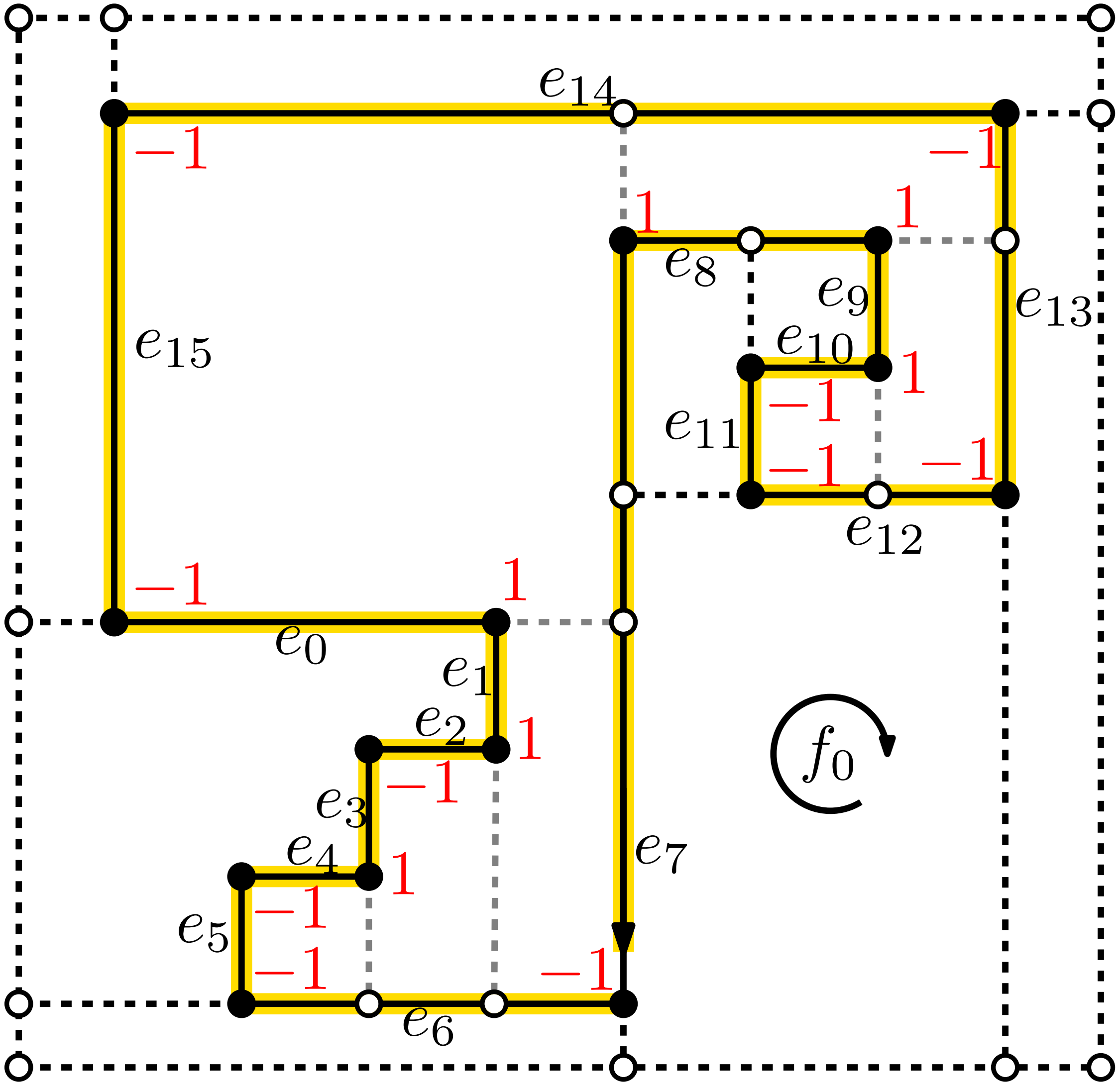
Refinement of (G, H) – Outer Face



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

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

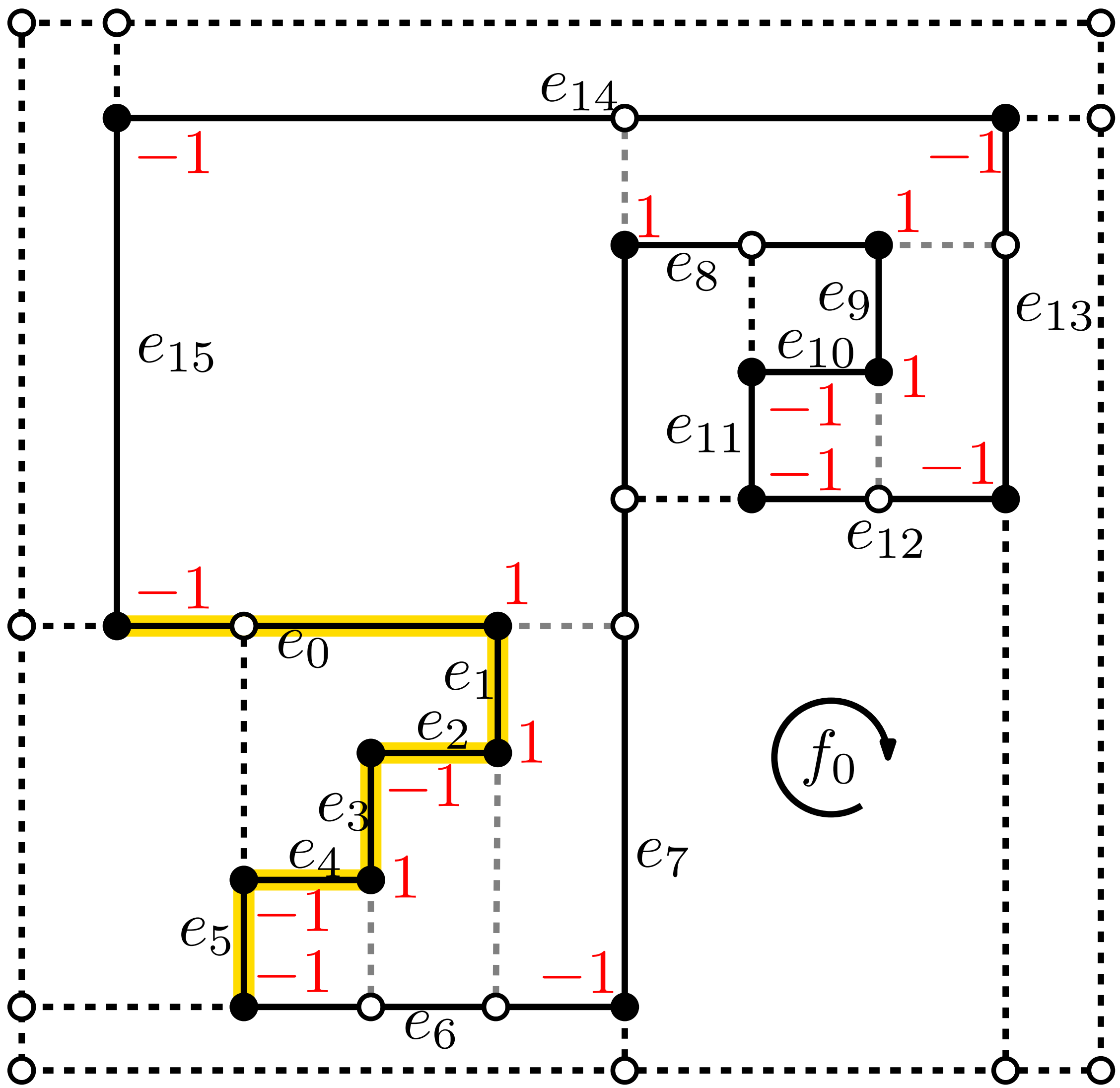
Refinement of (G, H) – Outer Face



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

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

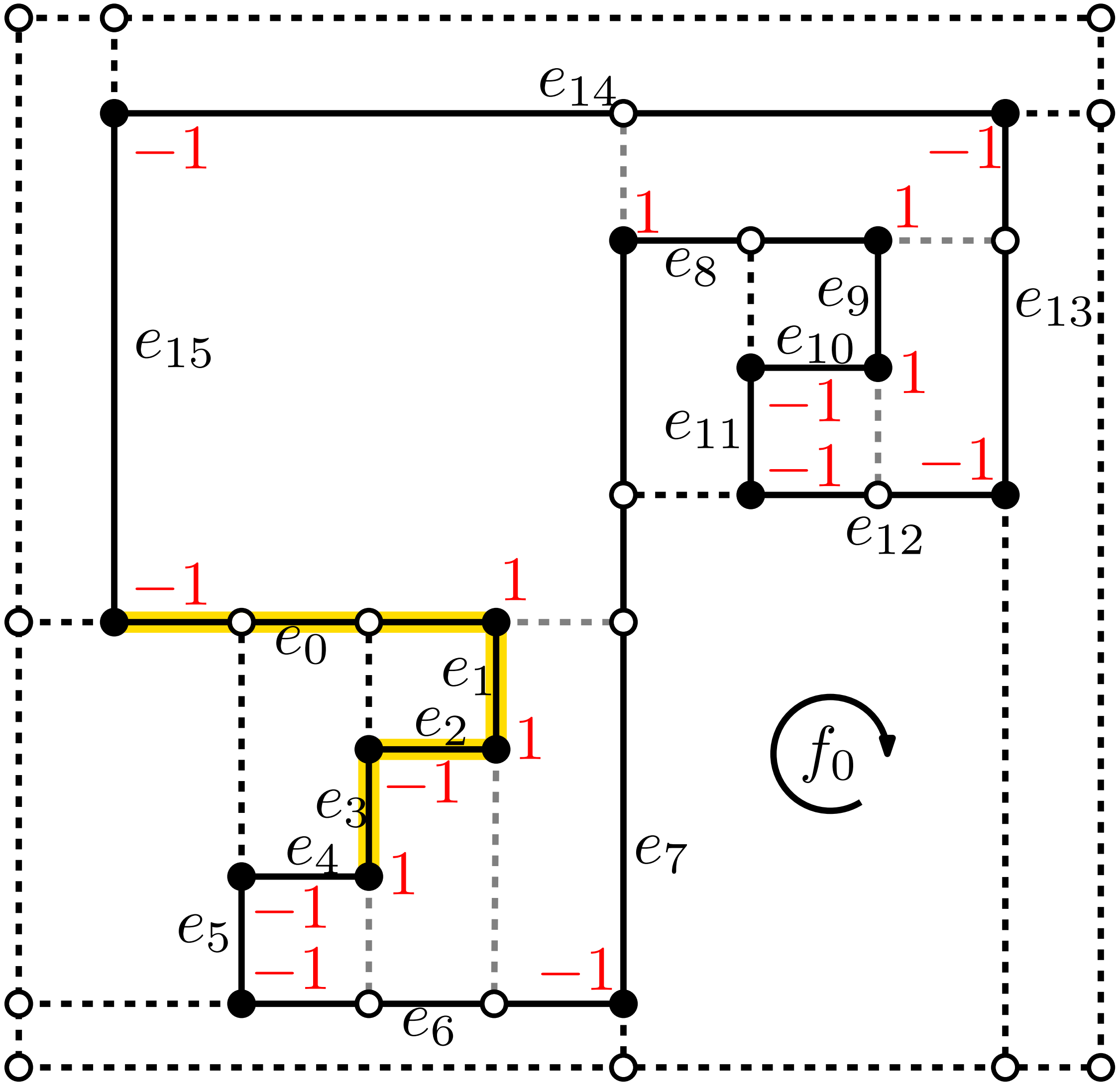
Refinement of (G, H) – Outer Face



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

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

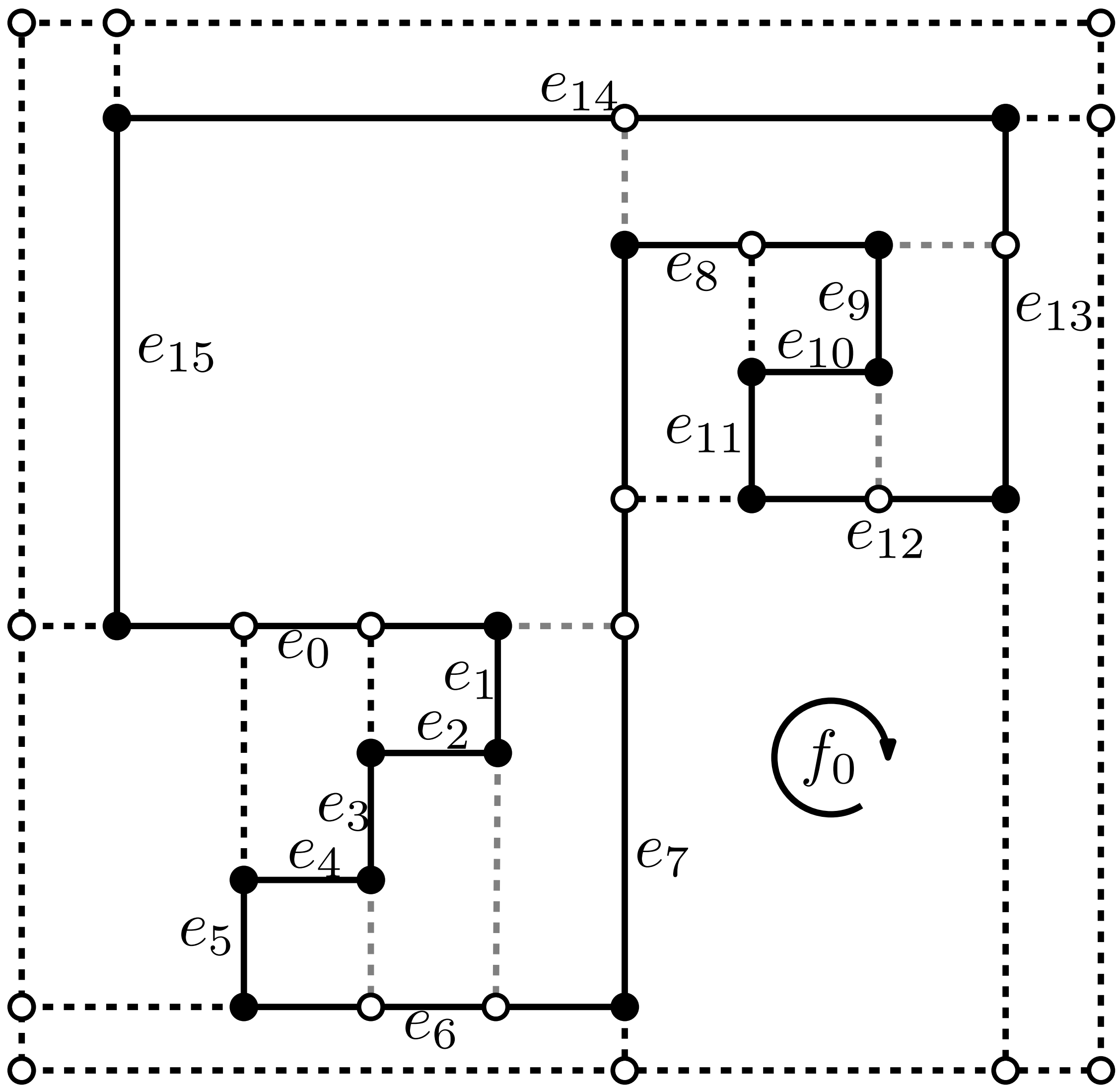
Refinement of (G, H) – Outer Face



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

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

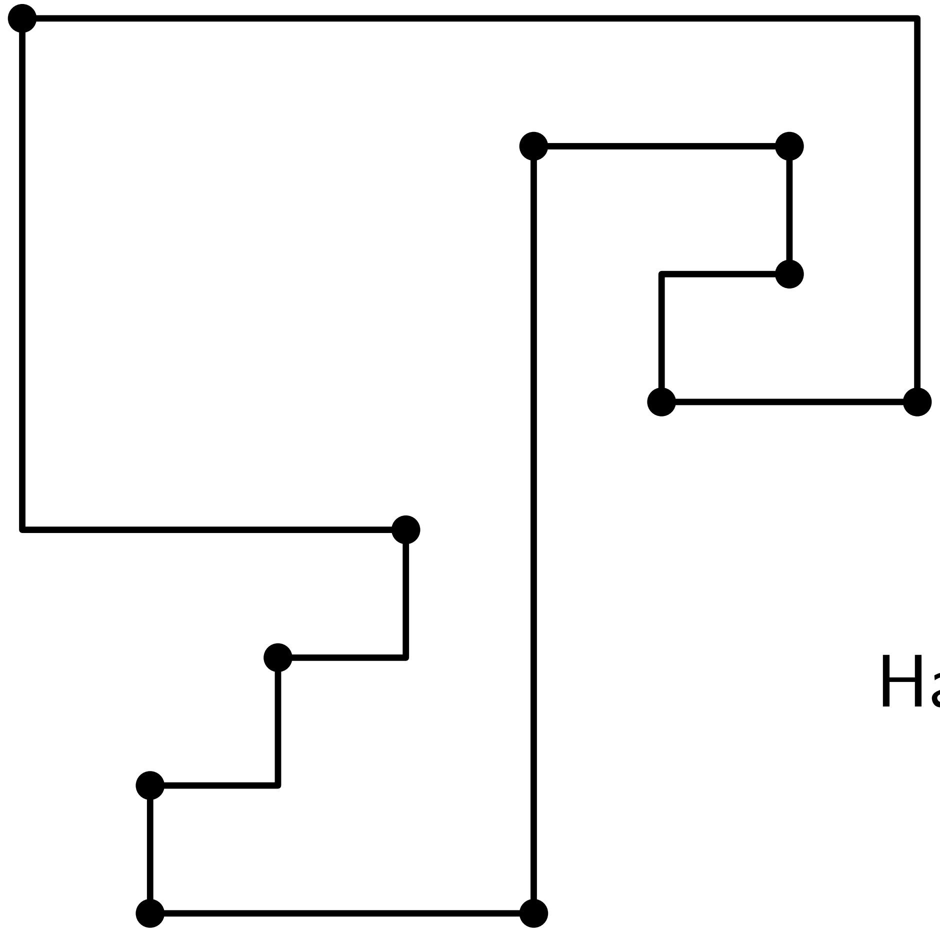
Refinement of (G, H) – Outer Face



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

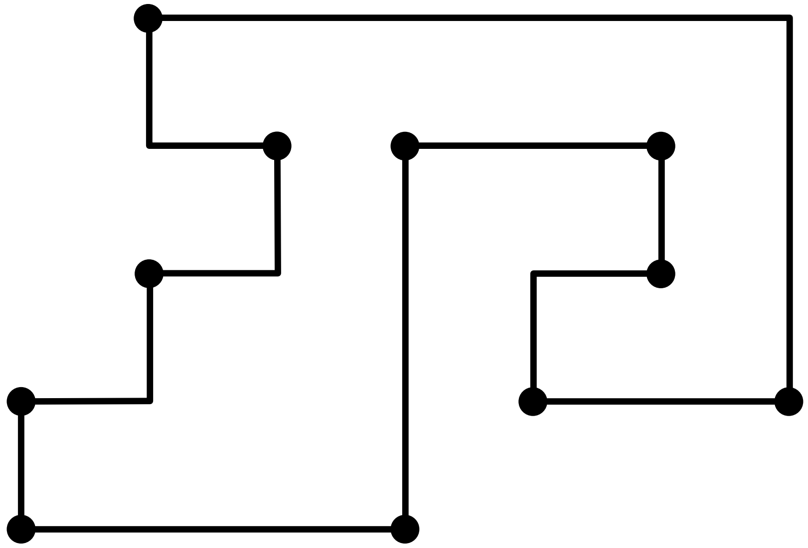
all faces are rectangles $\rightarrow$ apply flow network

Refinement of (G, H) – Outer Face



Has minimum area?

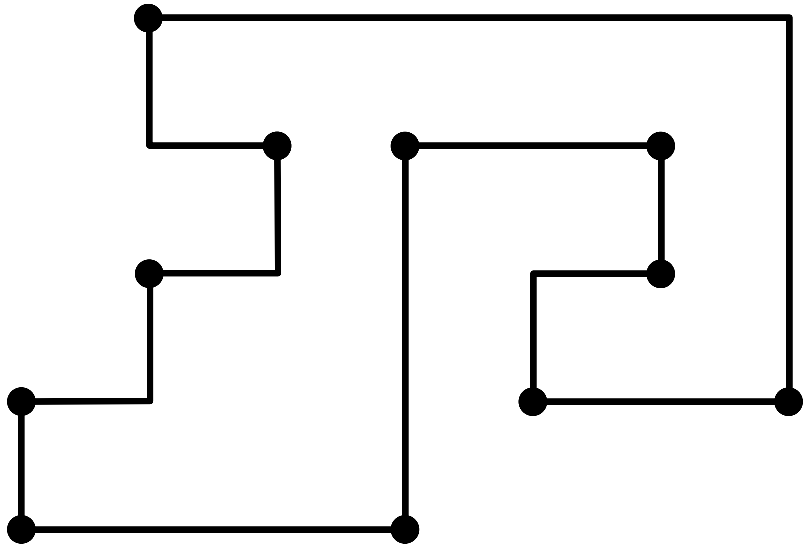
Refinement of (G, H) – Outer Face



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

Karlsruhe Institute of Technology

- 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

Karlsruhe Institute of Technology

- 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 w.r.t. running time and quality [Klau, Klein, Mutzel GD 2001]

Summary

Karlsruhe Institute of Technology

- 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 w.r.t. 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]

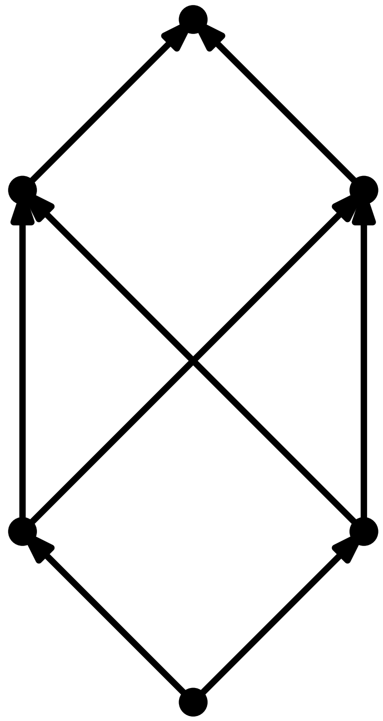
Upward Planarity

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.

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.

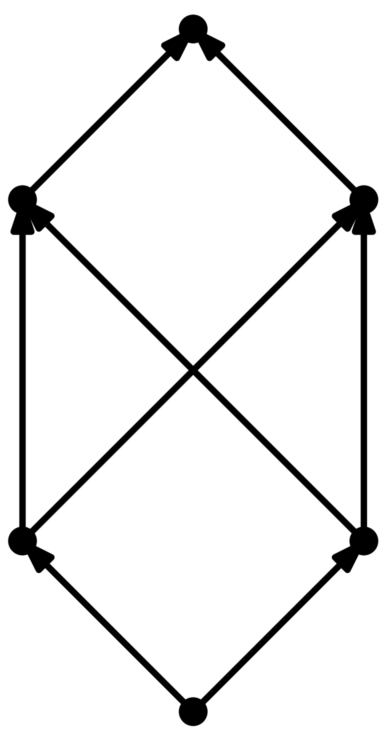
Example:

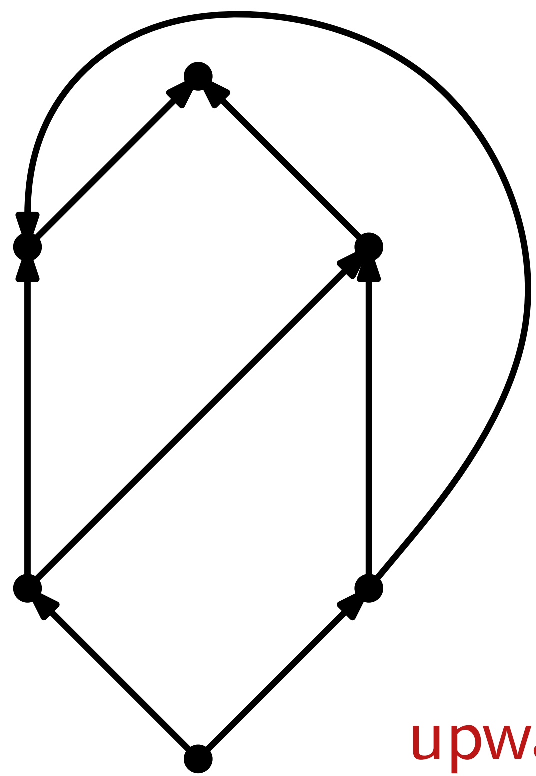


Upward Planarity

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!