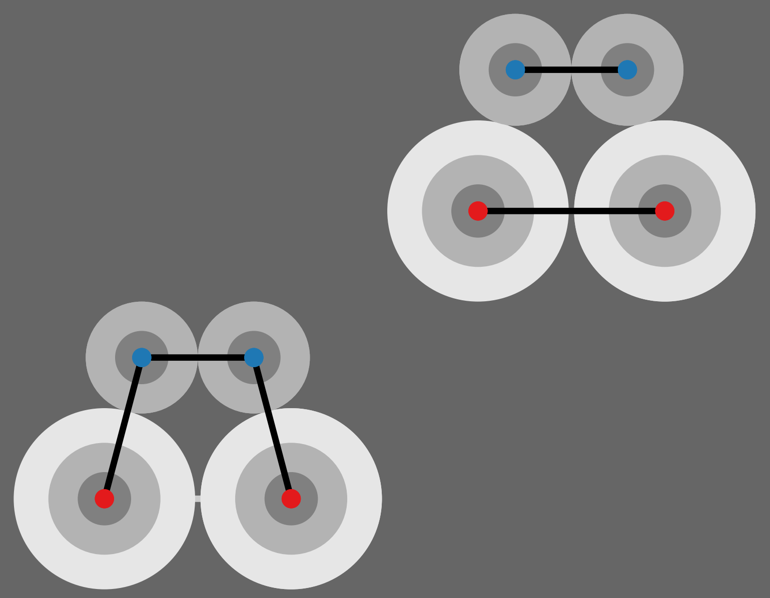
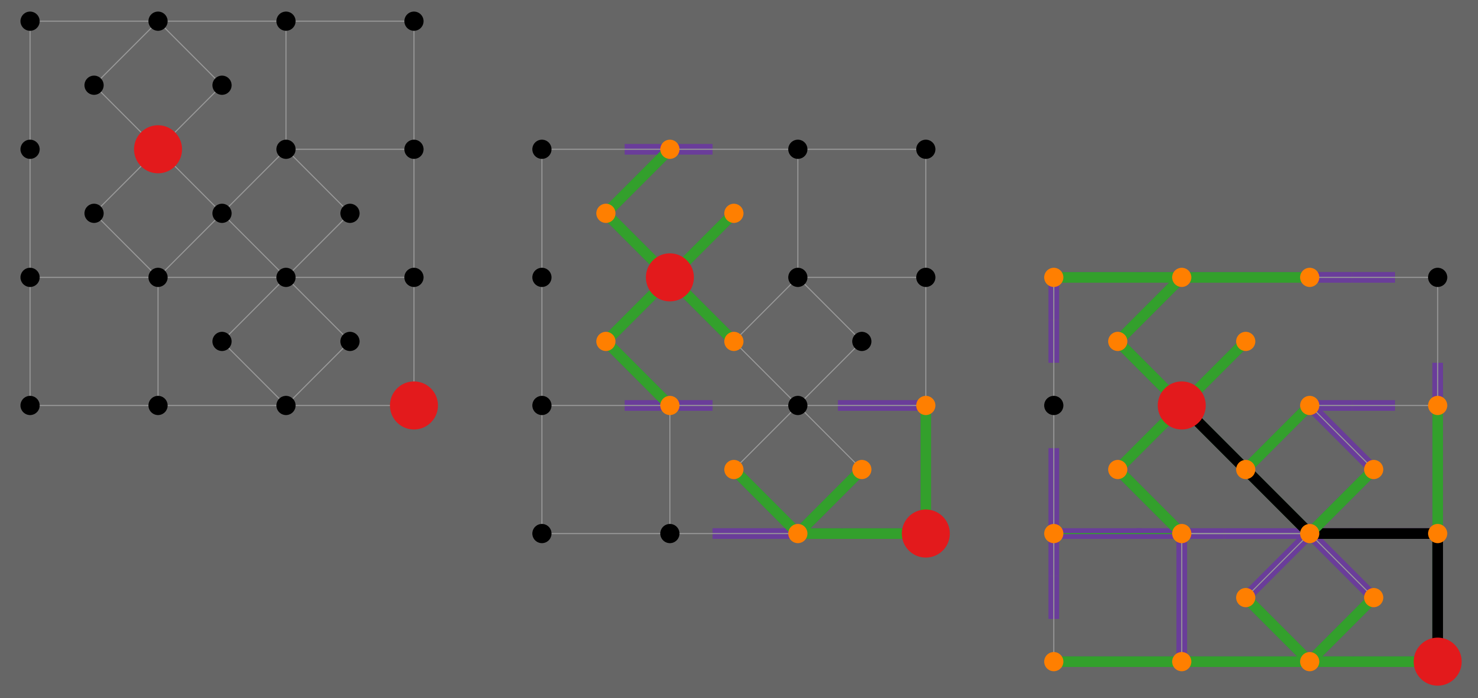




Model-Agnostic Approximation of Constrained Forest Problems

Corinna Coupette, Alipasha Montaseri and Christoph Lenzen

Motivation

Motivation

Steiner Forest: Generalization of Steiner Tree

Motivation

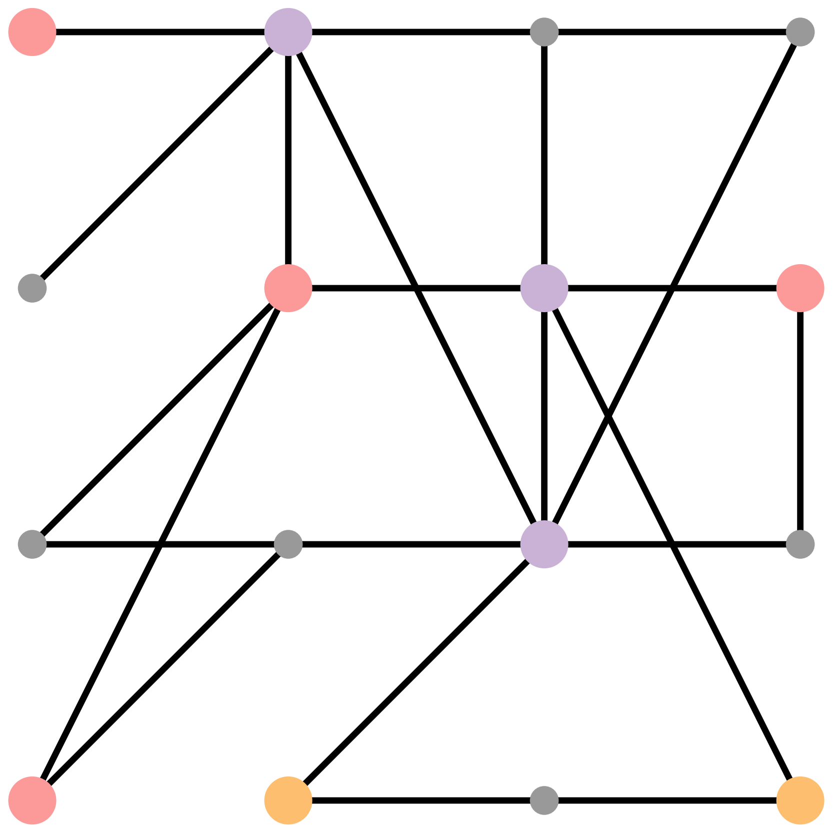
Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Motivation

Steiner Forest: Generalization of Steiner Tree

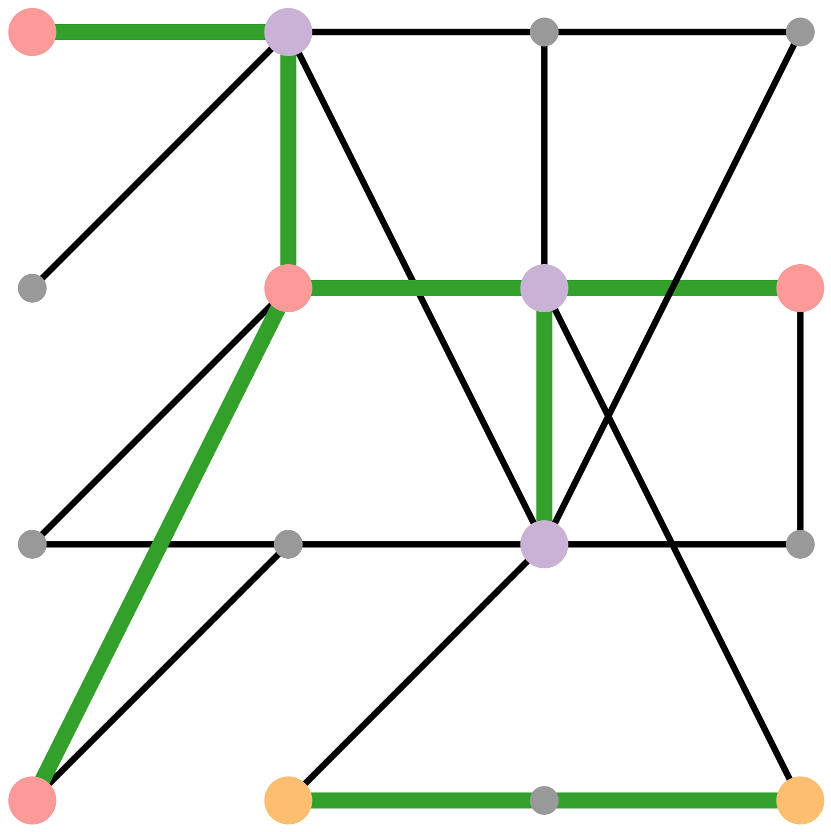
Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected



Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

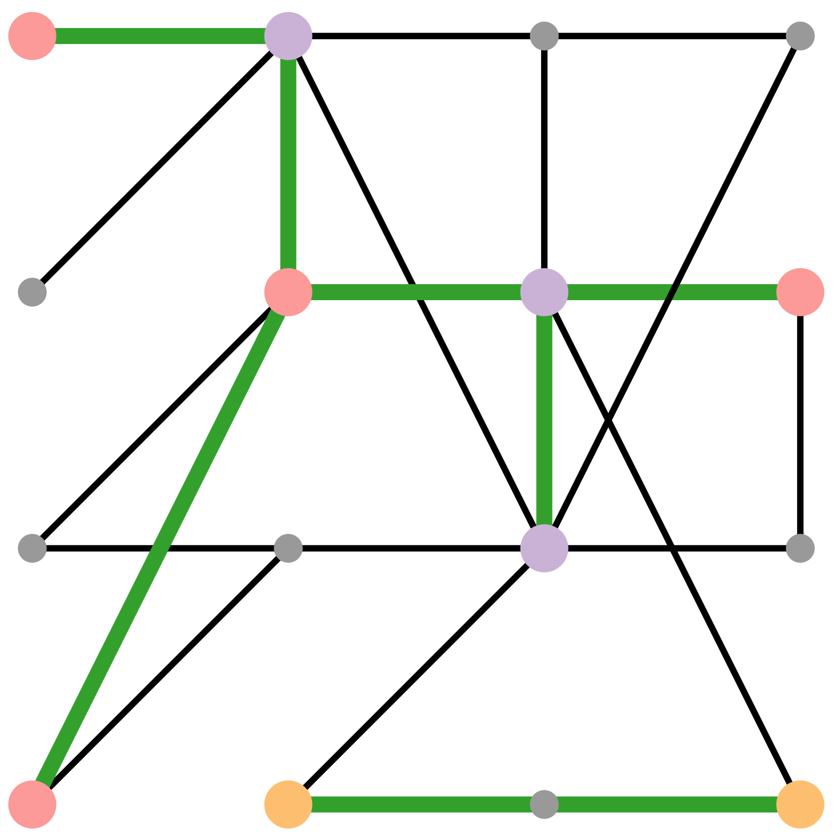


Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in Congest (Lenzen and Patt-Shamir 2014):



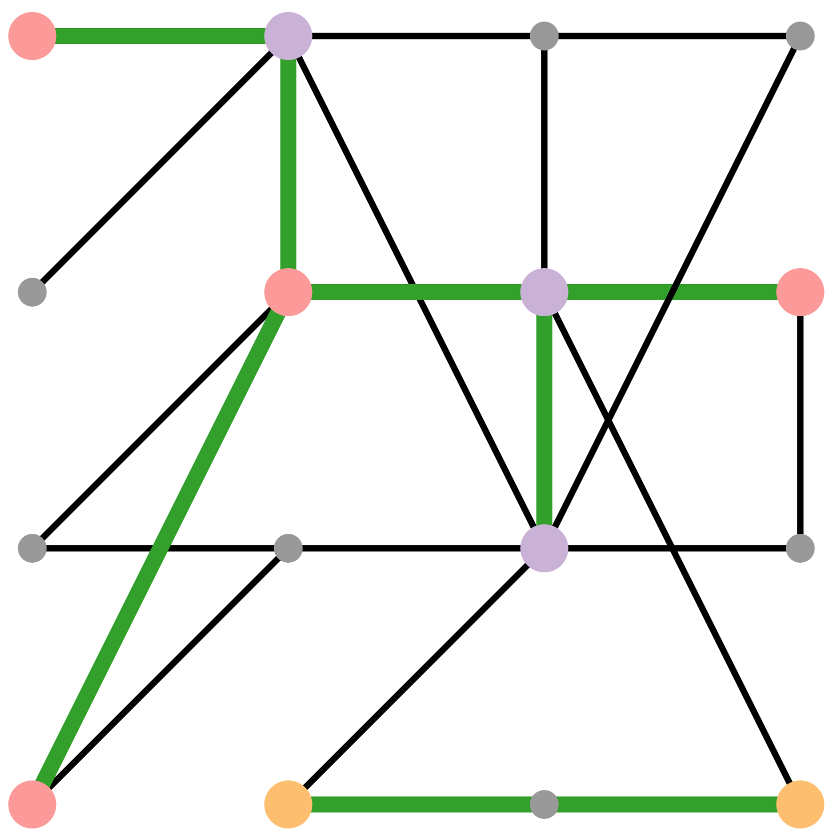
Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in Congest (Lenzen and Patt-Shamir 2014):

- $(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
- $O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]



Motivation

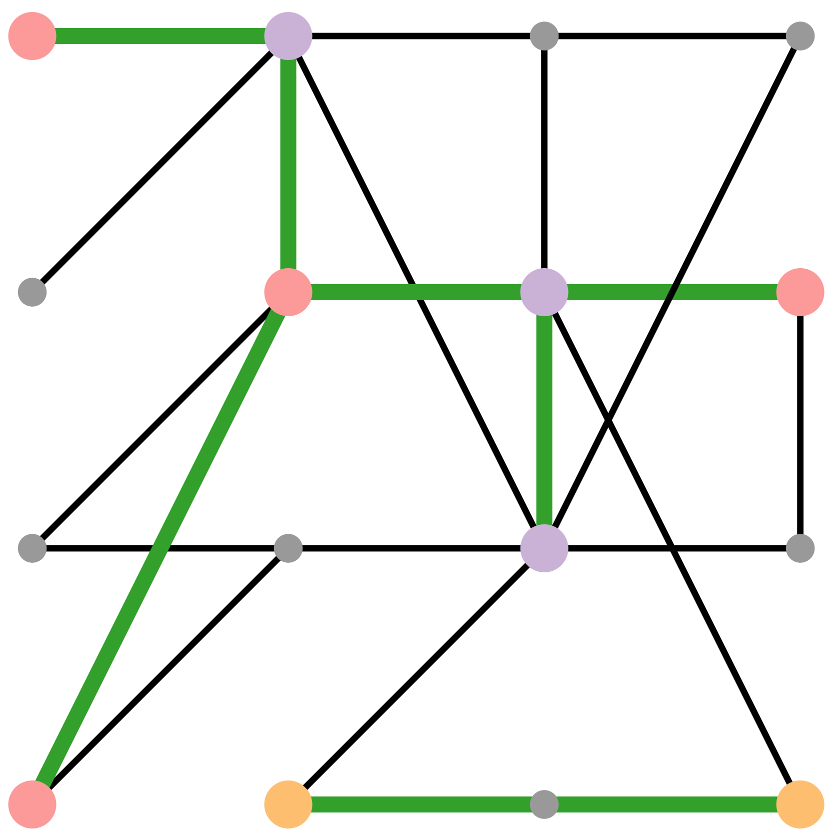
Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in Congest (Lenzen and Patt-Shamir 2014):

- $(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
- $O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]

Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$



Motivation

Steiner Forest: Generalization of Steiner Tree

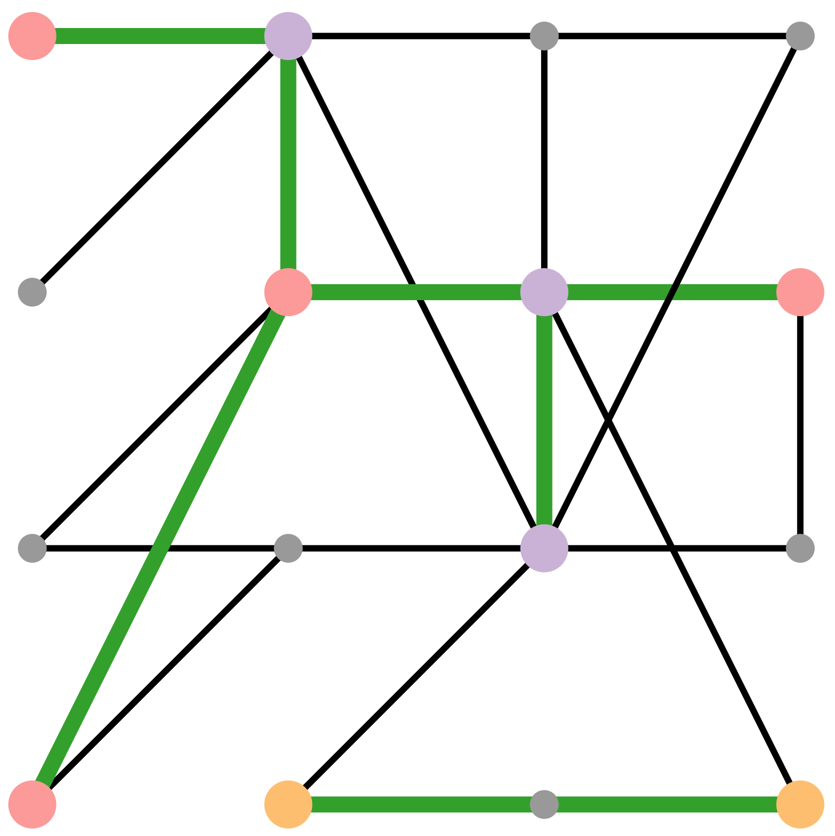
Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in Congest (Lenzen and Patt-Shamir 2014):

- $(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
- $O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]

Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$

Challenges:



Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

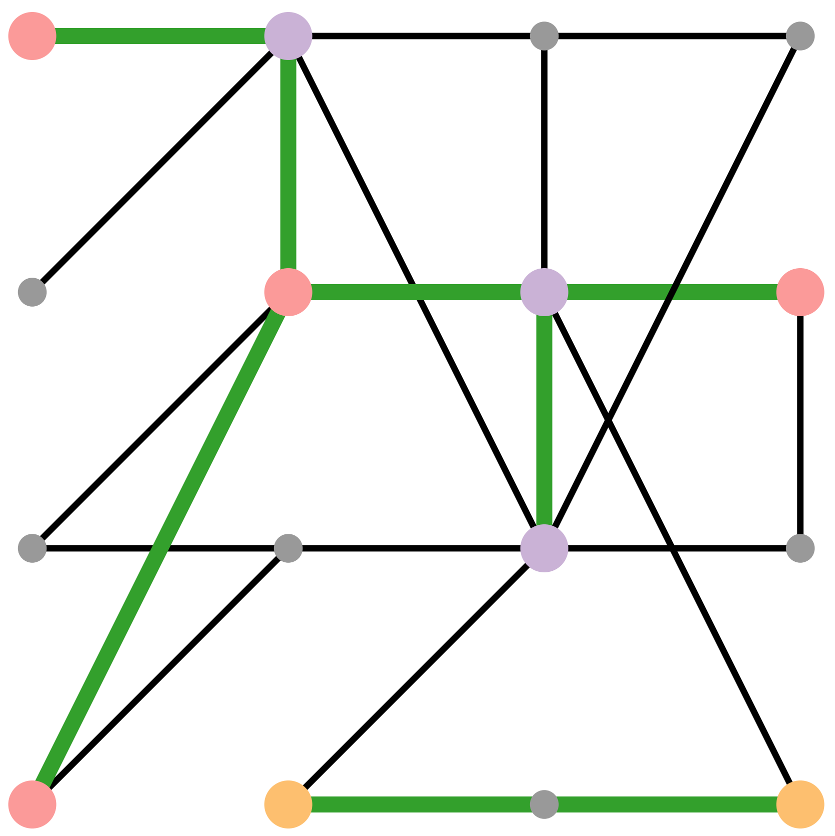
Prior Work in CONGEST (Lenzen and Patt-Shamir 2014):

$(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
$O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]

Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$

Challenges:

Model Specificity



Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

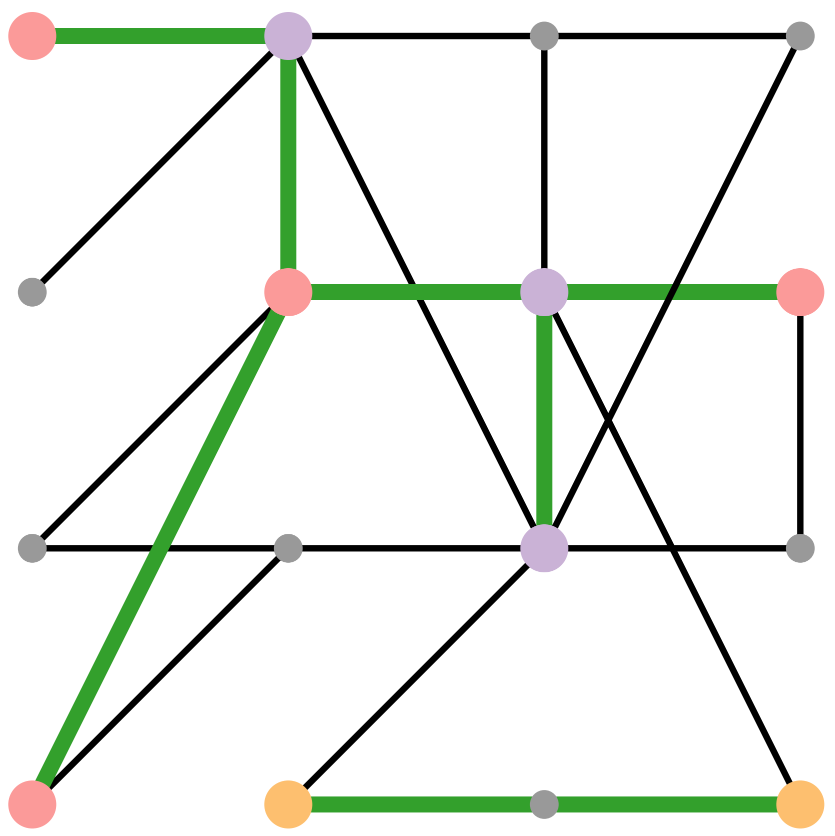
Prior Work in CONGEST (Lenzen and Patt-Shamir 2014):

$(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
$O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]

Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$

Challenges:

Model Specificity $\longrightarrow$ Model Agnosticism (works across computational models)



Motivation

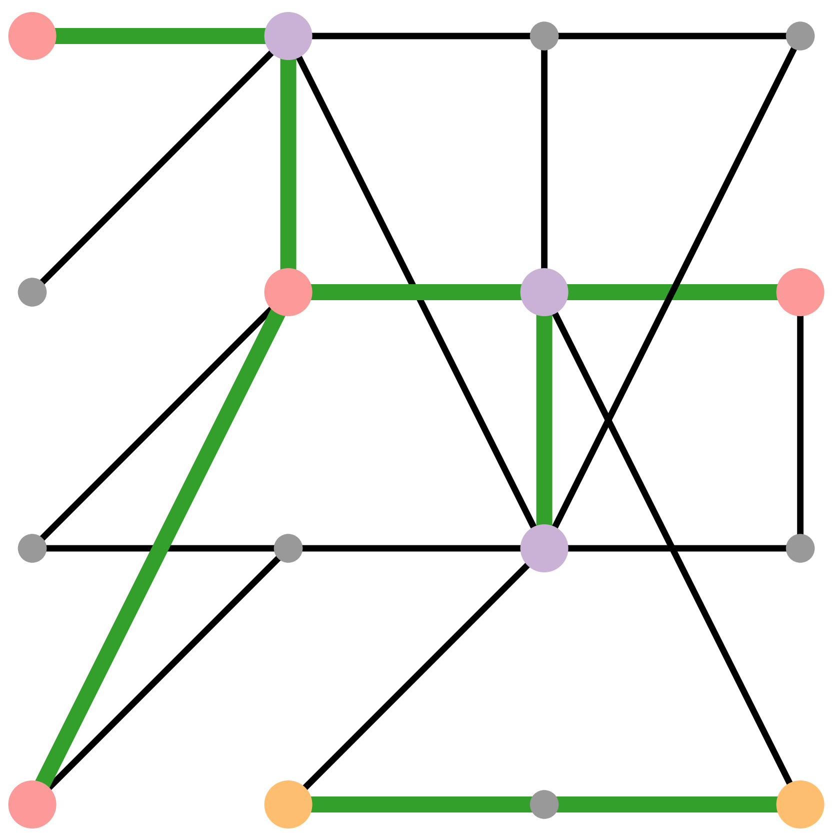
Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in Congest (Lenzen and Patt-Shamir 2014):

$(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
$O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]

Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$



Challenges:

Model Specificity → Model Agnosticism (works across computational models)

Existential Optimality

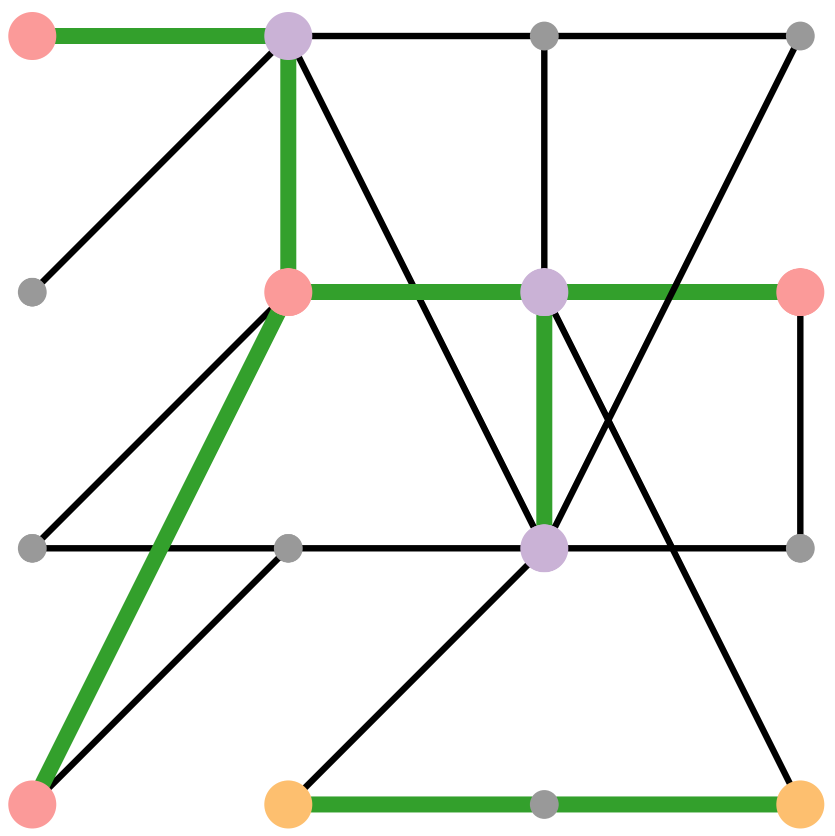
Motivation

Steiner Forest: Generalization of Steiner Tree

Classic Input: Components $\{V_i \mid i \in [k]\}$ to be connected

Prior Work in CONGEST (Lenzen and Patt-Shamir 2014):

$(2+\varepsilon)$-Approximation in $O(sk + \sqrt{\min\{st, n\}})$ rounds [D]
$O(\log n)$-Approximation in $O(\min\{s, \sqrt{n}\} + D + k)$ rounds [R]



Existential Lower Bound (Das Sarma et al. 2012): $\widetilde{\Omega}(\sqrt{n})$

Challenges:

Model Specificity	$\longrightarrow$ Model Agnosticism (works across computational models)
Existential Optimality	$\longrightarrow$ Universal Optimality (best on given topology)

Constrained Forest Problems (CFPs)

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$
\begin{aligned}
&\min \sum_{e \in E} c(e) x_e \\
\text{s.t.}\quad & x(\partial(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V \\
& x_e \in \{0,1\} \quad \forall\, e \in E
\end{aligned}
$$

Constrained Forest Problems (CFPs)

$\approx$ Problems on weighted graphs with forests as solutions

Primal IP

$$\begin{aligned} &\min \sum_{e \in E} c(e) x_e \\ \text{s.t.} \quad & x(\partial(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V \\ & x_e \in \{0, 1\} \quad \forall\, e \in E \end{aligned}$$

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$
\begin{aligned}
& \min \sum_{e \in E} c(e) x_e \\
\text{s.t.} \quad & x(\partial(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V \\
& x_e \in \{0, 1\} \quad \forall\, e \in E
\end{aligned}
$$

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$\begin{aligned} & \min \sum_{e \in E} c(e) x_e \\ \text{s.t.} \quad & x(\partial(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V \\ & x_e \in \{0,1\} \quad \forall\, e \in E \end{aligned}$$

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$\begin{aligned} \min \sum_{e \in E} c(e) x_e & \\ \text{s.t.} \quad x(\delta(S)) \geq f(S) & \quad \forall\, \emptyset \neq S \subset V \\ x_e \in \{0,1\} & \quad \forall\, e \in E \end{aligned}$$

Dual LP

$$\begin{aligned} \max \sum_{S \subset V} f(S) y_S & \\ \text{s.t.} \quad \sum_{S : e \in \delta(S)} y_S \leq c(e) & \quad \forall\, e \in E \\ y_S \geq 0 & \end{aligned}$$

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$\min \sum_{e \in E} c(e) x_e$$

$$\text{s.t.} \quad x(\delta(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V$$

$$x_e \in \{0,1\} \quad \forall\, e \in E$$

Dual LP

$$\max \sum_{S \subset V} f(S) y_S$$

$$\text{s.t.} \quad \sum_{S: e \in \delta(S)} y_S \leq c(e) \quad \forall\, e \in E$$

$$y_S \geq 0$$

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$\begin{aligned} \min \sum_{e \in E} c(e) x_e & \\ \text{s.t.} \quad x(\delta(S)) \geq f(S) & \quad \forall\, \emptyset \neq S \subset V \\ x_e \in \{0,1\} & \quad \forall\, e \in E \end{aligned}$$

Dual LP

$$\begin{aligned} \max \sum_{S \subset V} f(S) y_S & \\ \text{s.t.} \quad \sum_{S : e \in \delta(S)} y_S \leq c(e) & \quad \forall\, e \in E \\ y_S \geq 0 & \end{aligned}$$

We focus on CFPs with *proper* functions f (zero, symmetry, disjointness)

Constrained Forest Problems (CFPs)

≈ Problems on weighted graphs with forests as solutions

Primal IP

$$\min \sum_{e \in E} c(e) x_e$$

$$\text{s.t.} \quad x(\delta(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V$$

$$x_e \in \{0, 1\} \quad \forall\, e \in E$$

Dual LP

$$\max \sum_{S \subset V} f(S) y_S$$

$$\text{s.t.} \quad \sum_{S : e \in \delta(S)} y_S \leq c(e) \quad \forall\, e \in E$$

$$y_S \geq 0$$

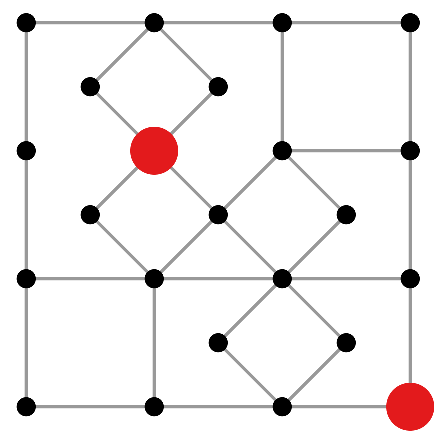
We focus on CFPs with *proper* functions f (zero, symmetry, disjointness)

Steiner Forest: $f(S) = 1 \Leftrightarrow \emptyset \neq S \cap V_i \neq V_i$ for some $i \in [k]$

The Goemans-Williamson Algorithm

The Goemans-Williamson Algorithm

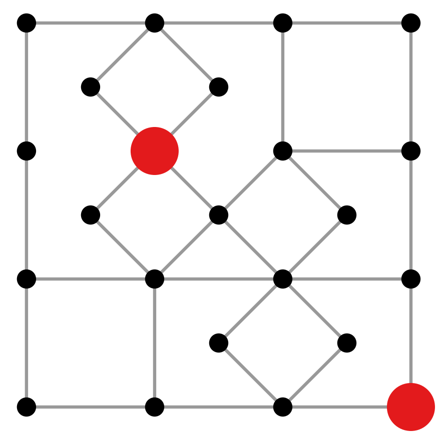
Input: Graph with edge costs c, proper forest function f



The Goemans-Williamson Algorithm

Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

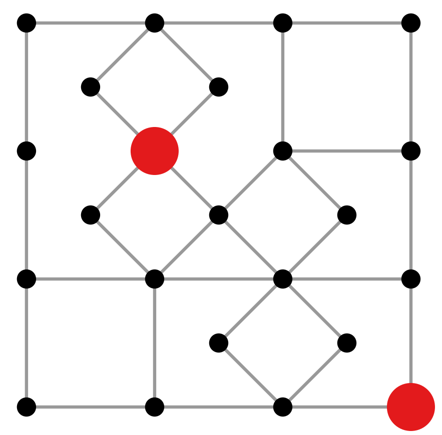


The Goemans-Williamson Algorithm

Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:



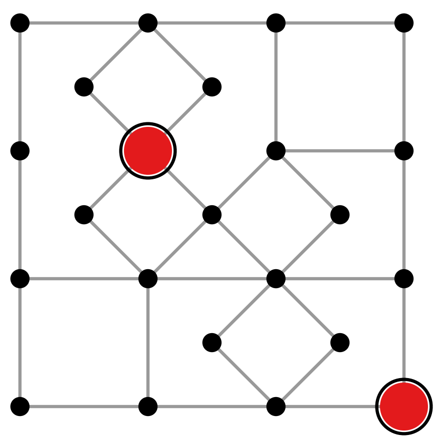
The Goemans-Williamson Algorithm

Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$



The Goemans-Williamson Algorithm

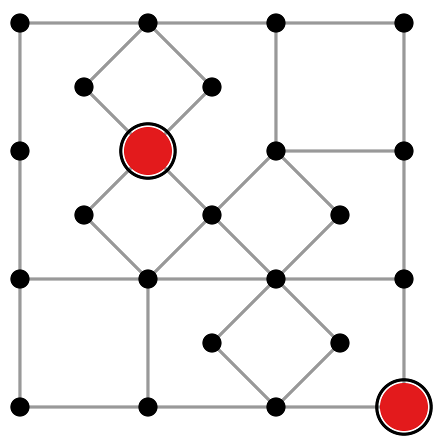
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

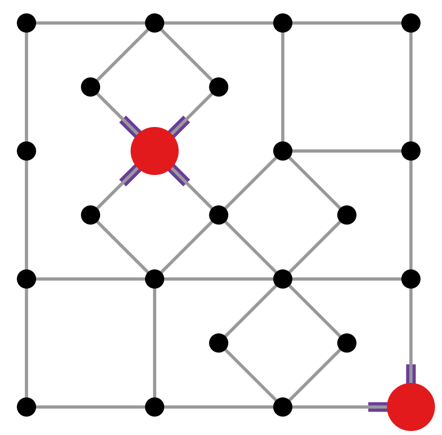
While there are active components ($f(C) = 1$)



The Goemans-Williamson Algorithm

Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB



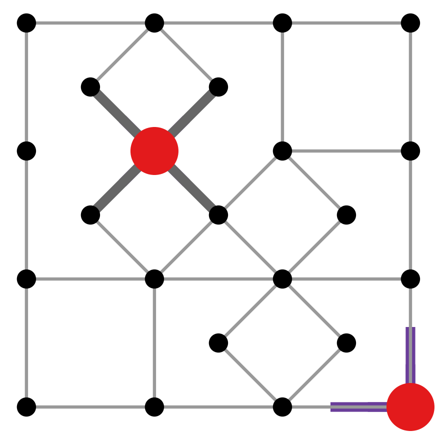
Procedure:

Start with each node v as its own component $C = \{v\}$

While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

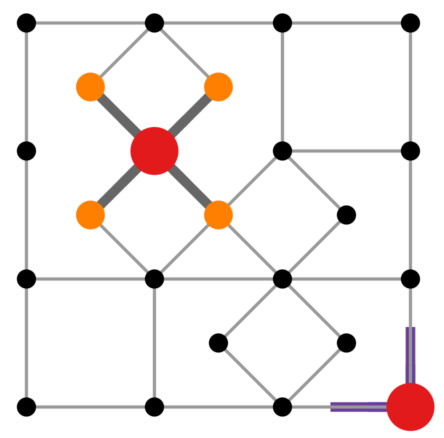
Start with each node v as its own component $C = \{v\}$

While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

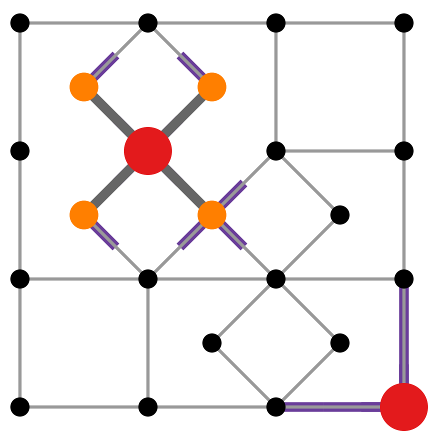
While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



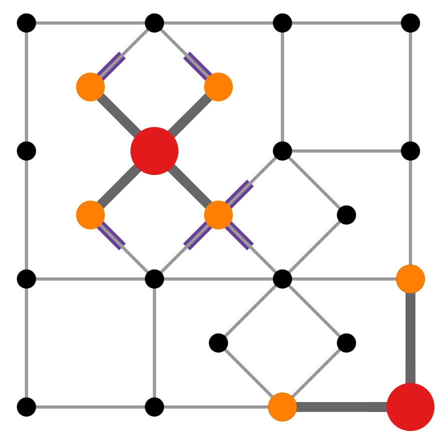
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

- Start with each node v as its own component $C = \{v\}$
- While there are active components ($f(C) = 1$)
  - Increase dual variables of all edges incident with active components, adding to LB
  - Once a dual variable $y(e)$ becomes tight, add e to F
  - Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

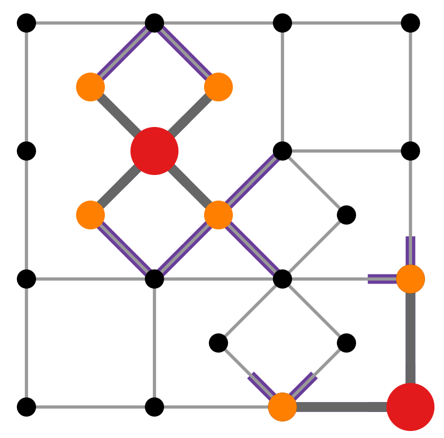
While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



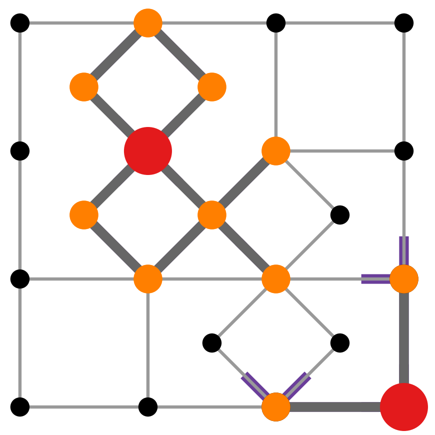
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

- Start with each node v as its own component $C = \{v\}$
- While there are active components ($f(C) = 1$)
  - Increase dual variables of all edges incident with active components, adding to LB
  - Once a dual variable $y(e)$ becomes tight, add e to F
  - Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



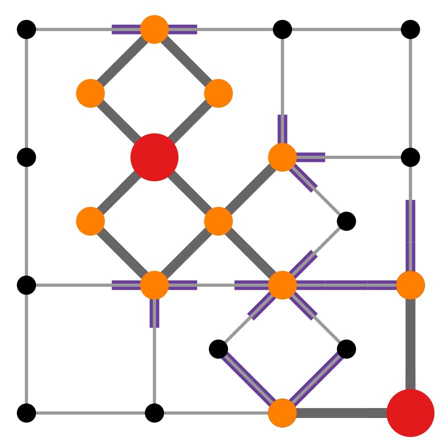
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

- Start with each node v as its own component $C = \{v\}$
- While there are active components ($f(C) = 1$)
    - Increase dual variables of all edges incident with active components, adding to LB
    - Once a dual variable $y(e)$ becomes tight, add e to F
    - Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

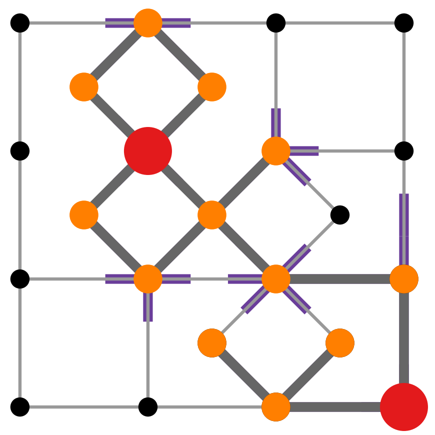
While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



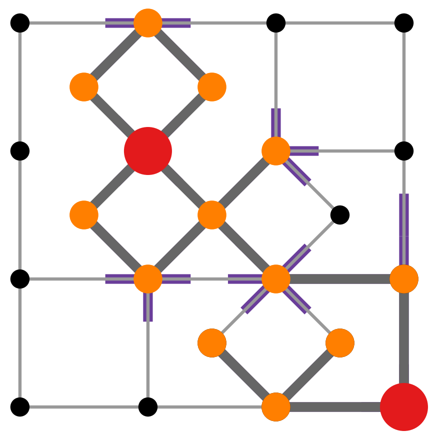
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

- Start with each node v as its own component $C = \{v\}$
- While there are active components ($f(C) = 1$)
  - Increase dual variables of all edges incident with active components, adding to LB
  - Once a dual variable $y(e)$ becomes tight, add e to F
  - Merge components represented by endpoints of e

The Goemans-Williamson Algorithm



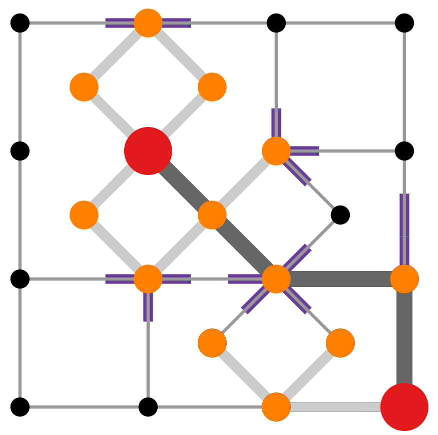
Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

- Start with each node v as its own component $C = \{v\}$
- While there are active components ($f(C) = 1$)
  - Increase dual variables of all edges incident with active components, adding to LB
  - Once a dual variable $y(e)$ becomes tight, add e to F
  - Merge components represented by endpoints of e
- Remove unnecessary edges from F

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

While there are active components ($f(C) = 1$)

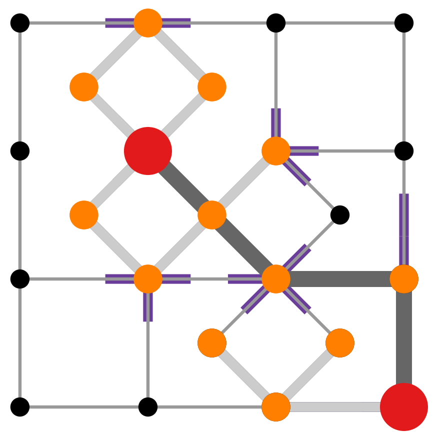
Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

Merge components represented by endpoints of e

Remove unnecessary edges from F

The Goemans-Williamson Algorithm



Input: Graph with edge costs c, proper forest function f

Output: Forest F and lower-bound value LB

Procedure:

Start with each node v as its own component $C = \{v\}$

While there are active components ($f(C) = 1$)

Increase dual variables of all edges incident with active components, adding to LB

Once a dual variable $y(e)$ becomes tight, add e to F

Merge components represented by endpoints of e

Remove unnecessary edges from F

Approximation Guarantee: $2 - {}^{2}/_{t}$

Toward Model Agnosticism

Toward Model Agnosticism

Challenge

Goemans-Williamson

Our Approach

Toward Model Agnosticism

Challenge	Goemans-Williamson	Our Approach
Solution-Set Construction	Filtering	Incremental

Toward Model Agnosticism

Challenge	Goemans-Williamson	Our Approach
Solution-Set Construction	Filtering	Incremental
Pairwise Distance Computations	Exact	Approximate

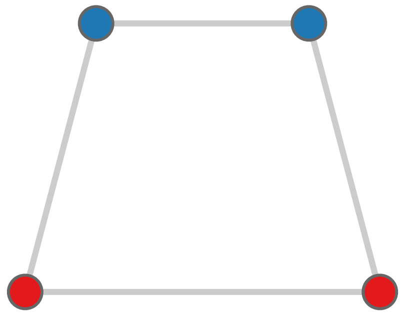
Toward Model Agnosticism

Challenge	Goemans-Williamson	Our Approach
Solution-Set Construction	Filtering	Incremental
Pairwise Distance Computations	Exact	Approximate
Forest-Function Evaluation	Many (Each Edge)	Deferred (Each Phase)

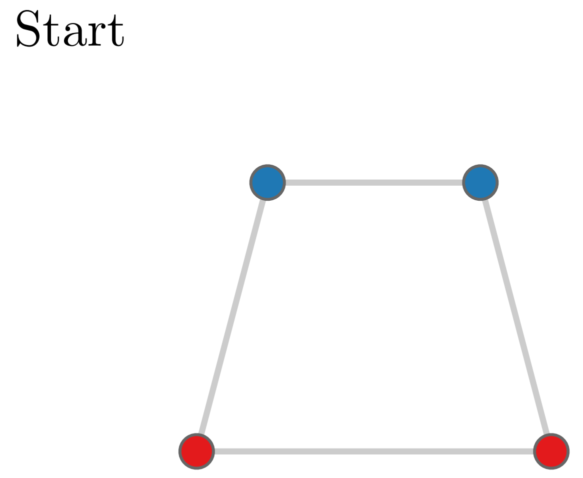
The Shell-Decomposition Algorithm: Approximation Intuition

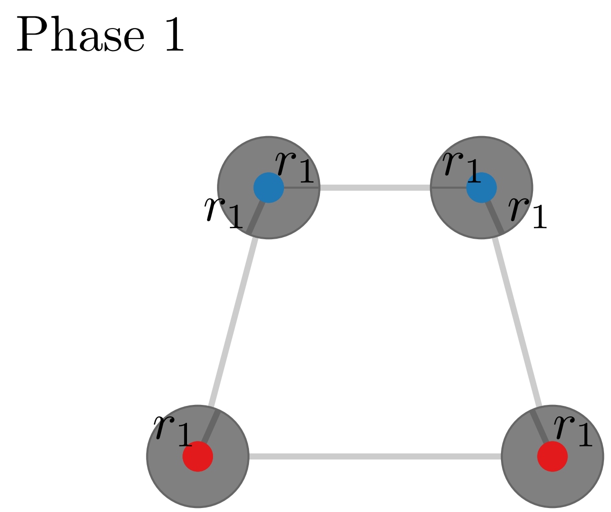
The Shell-Decomposition Algorithm: Approximation Intuition

Start

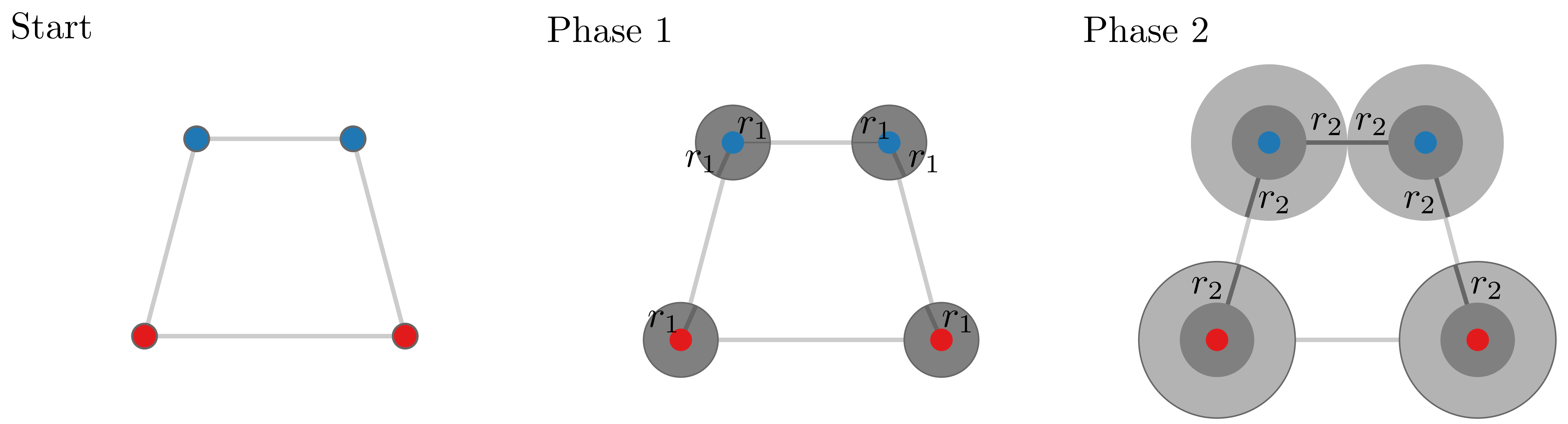


The Shell-Decomposition Algorithm: Approximation Intuition

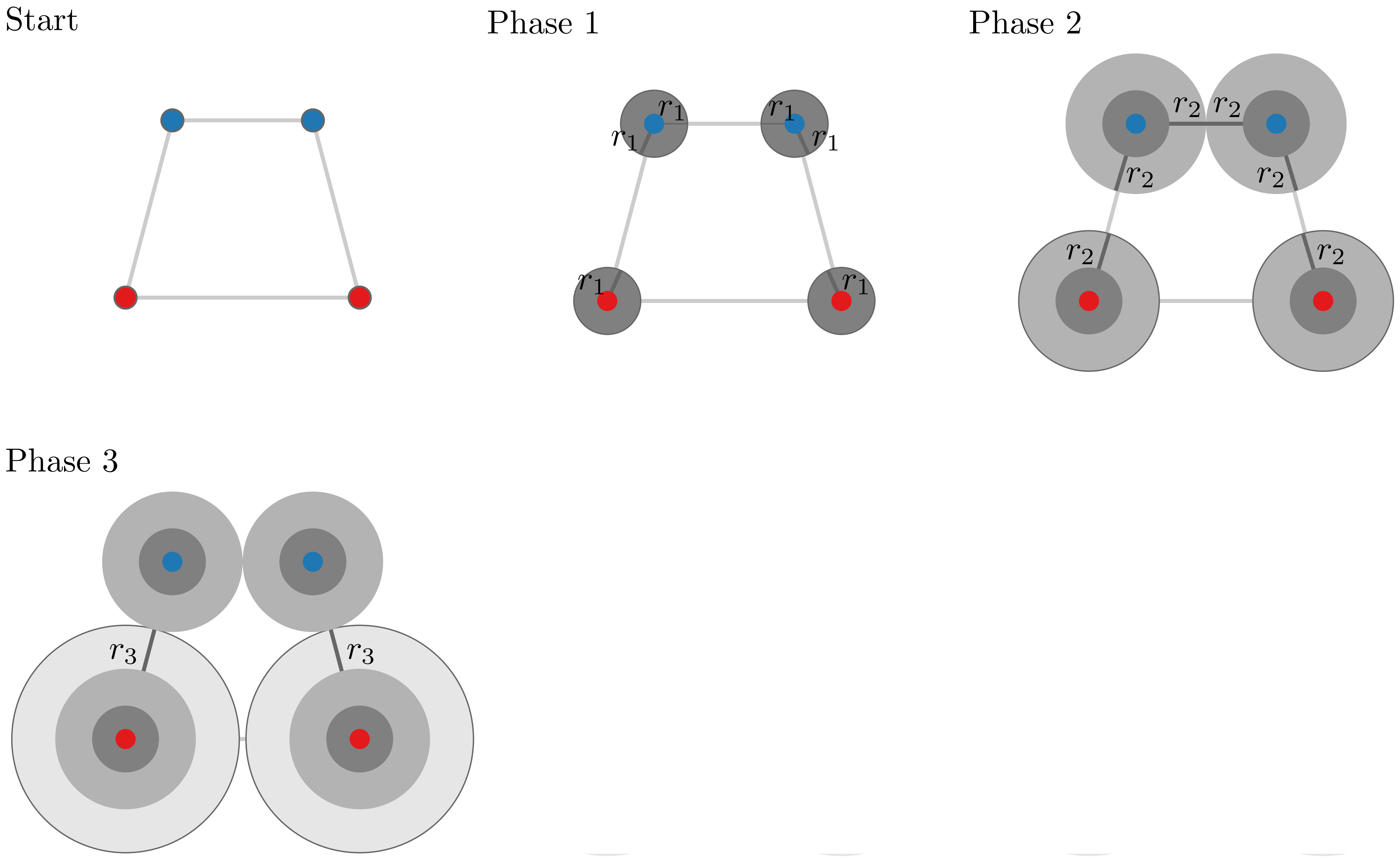




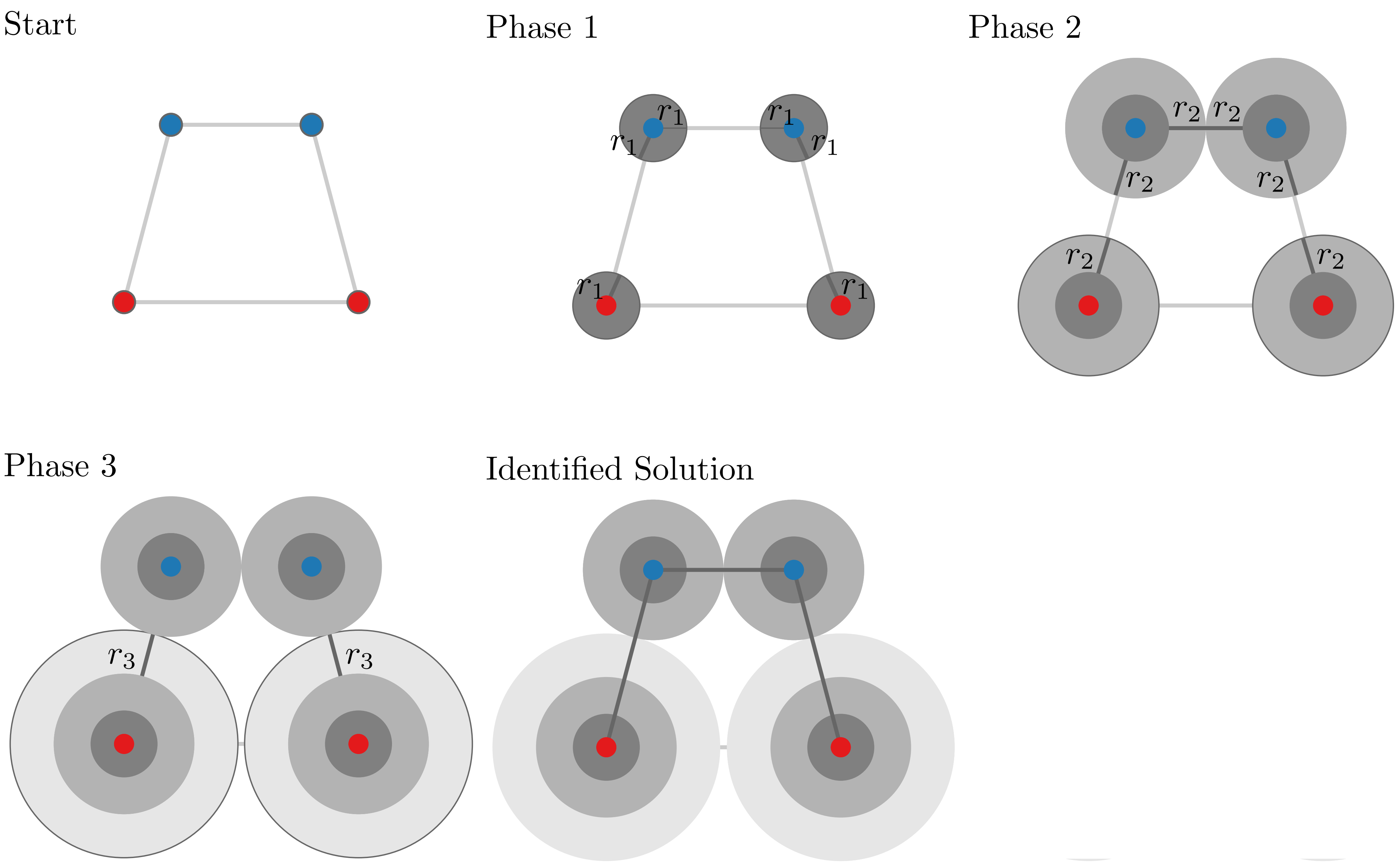
The Shell-Decomposition Algorithm: Approximation Intuition



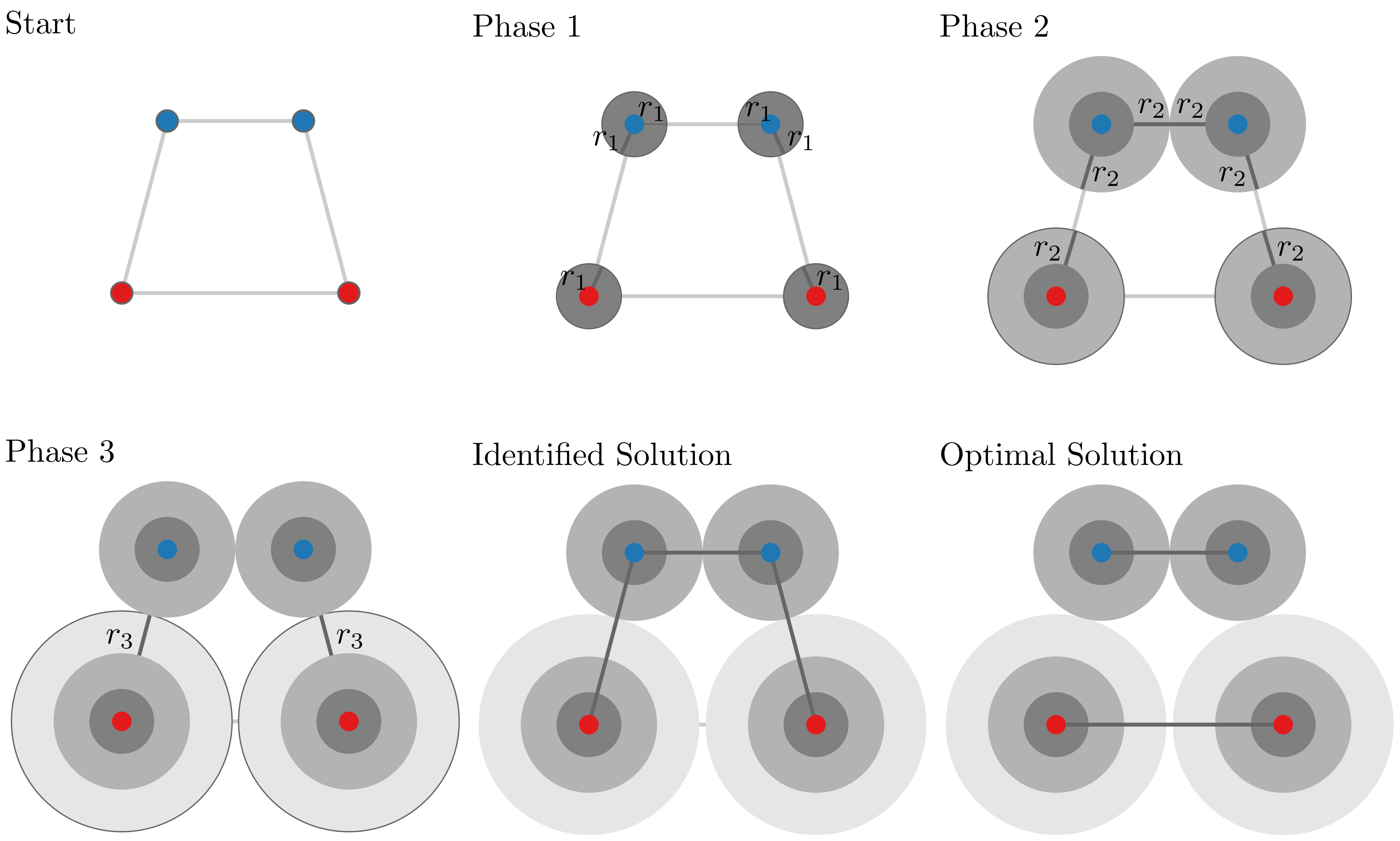
The Shell-Decomposition Algorithm: Approximation Intuition



The Shell-Decomposition Algorithm: Approximation Intuition



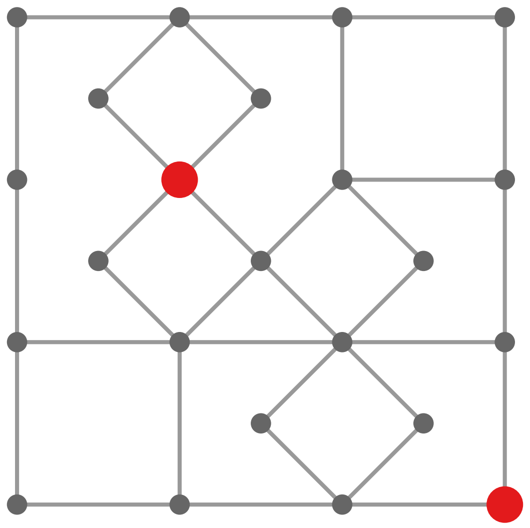
The Shell-Decomposition Algorithm: Approximation Intuition



The Shell-Decomposition Algorithm: Building Blocks

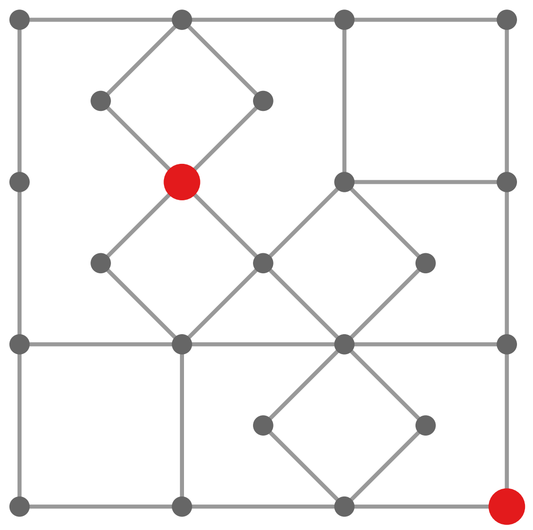
The Shell-Decomposition Algorithm: Building Blocks

Initialization

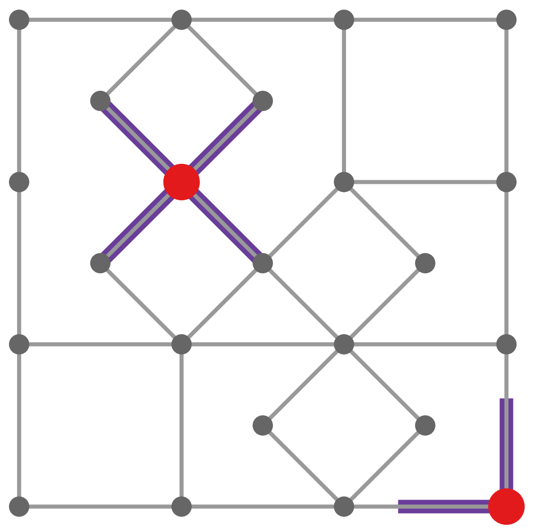


The Shell-Decomposition Algorithm: Building Blocks

Initialization



(0) Ball Growth

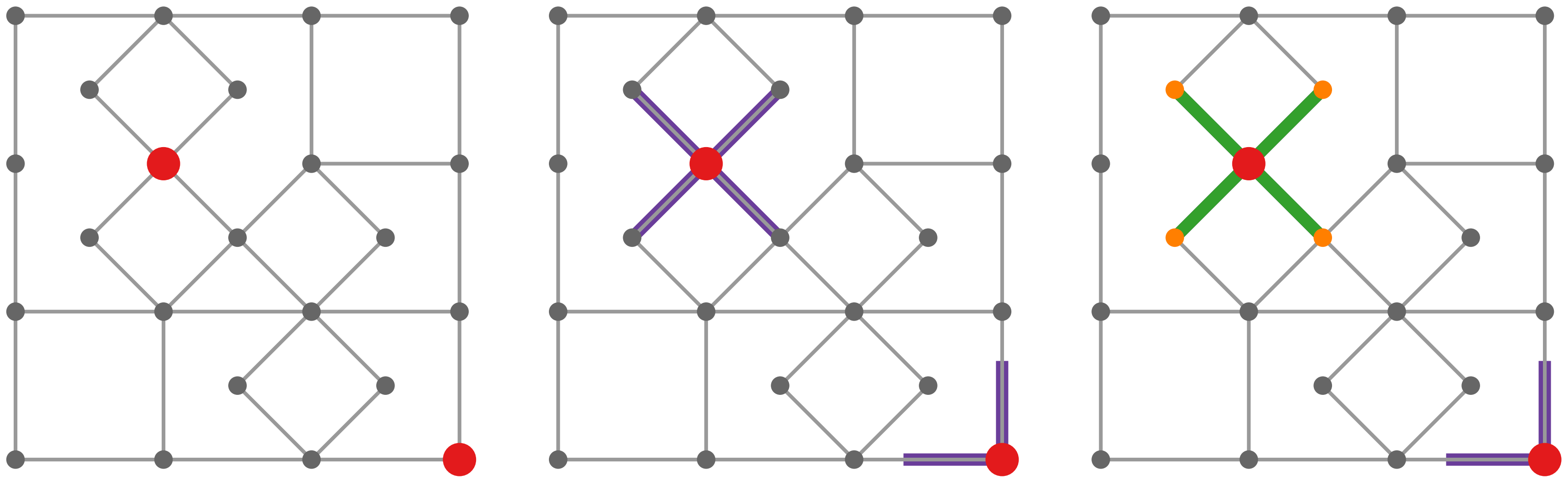


The Shell-Decomposition Algorithm: Building Blocks

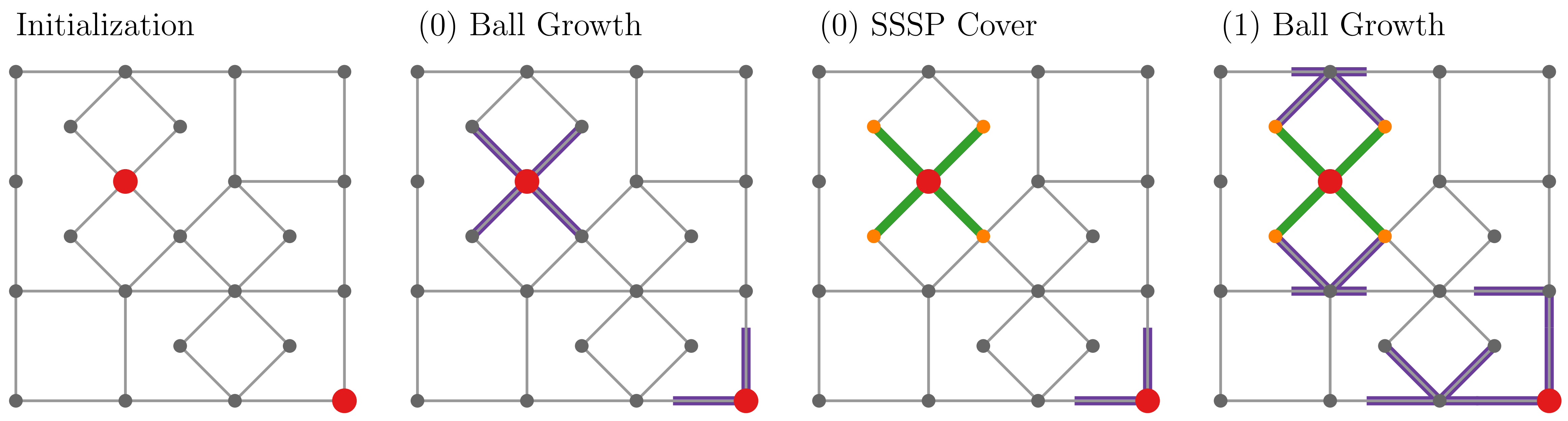
Initialization

(0) Ball Growth

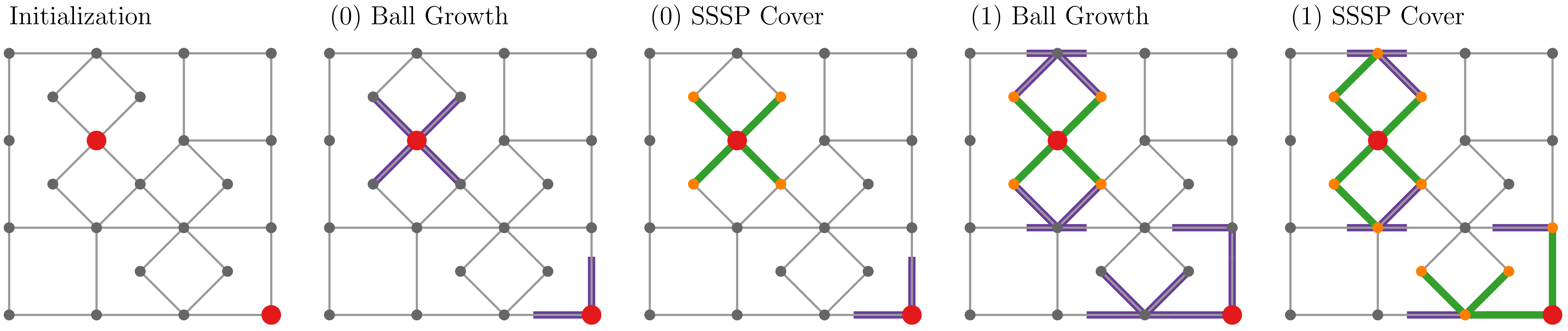
(0) SSSP Cover



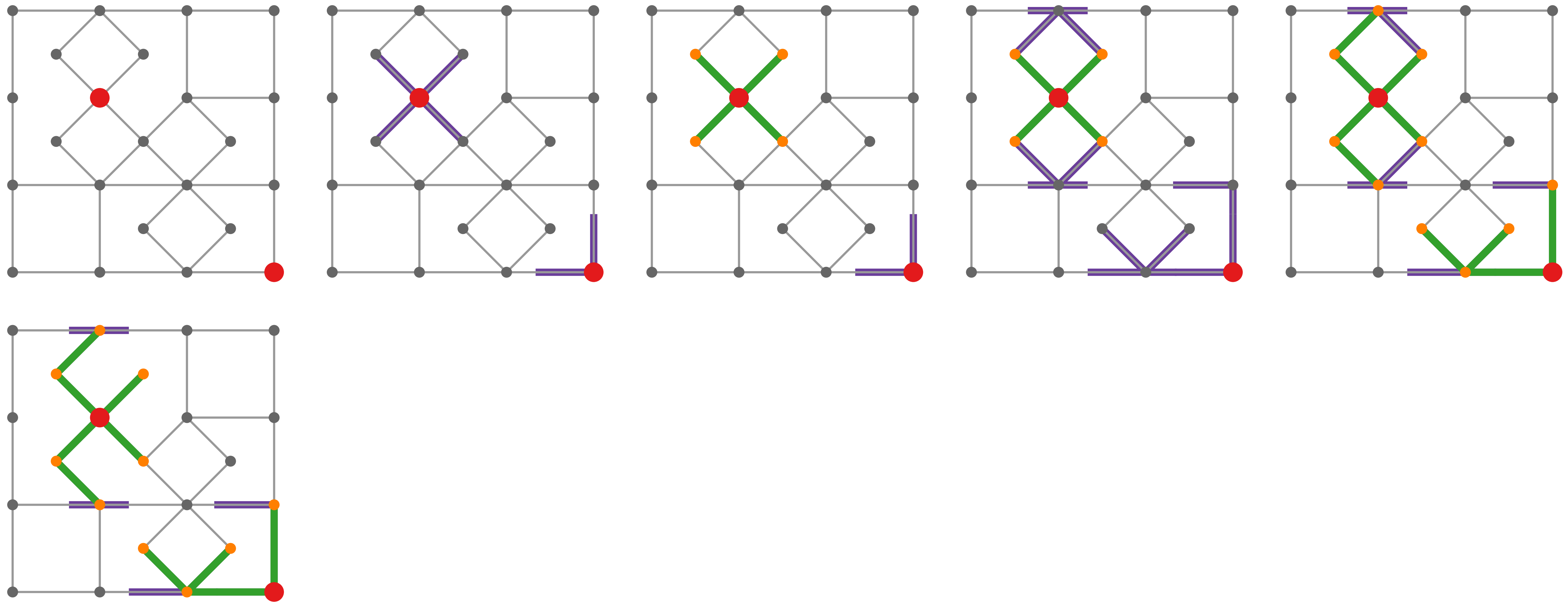
The Shell-Decomposition Algorithm: Building Blocks



The Shell-Decomposition Algorithm: Building Blocks

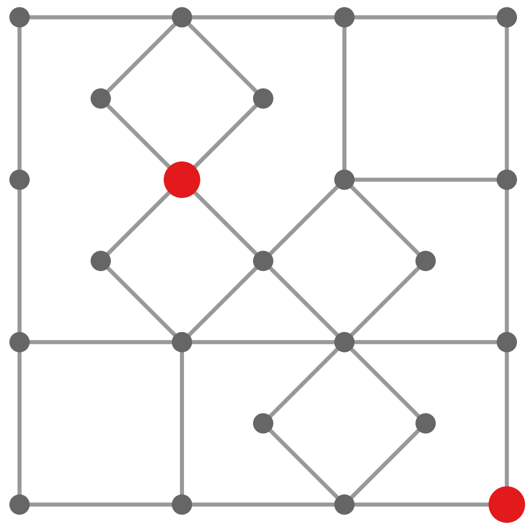


The Shell-Decomposition Algorithm: Building Blocks

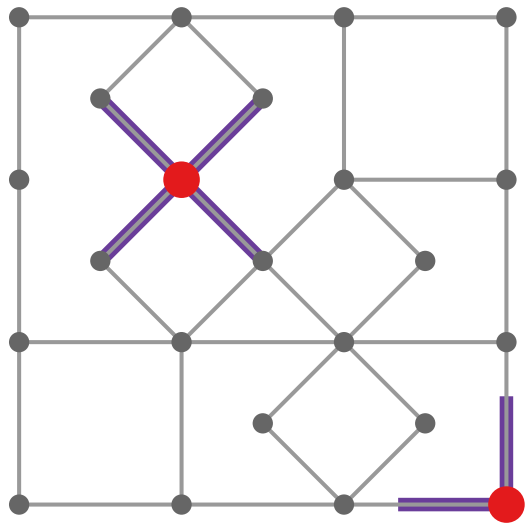


The Shell-Decomposition Algorithm: Building Blocks

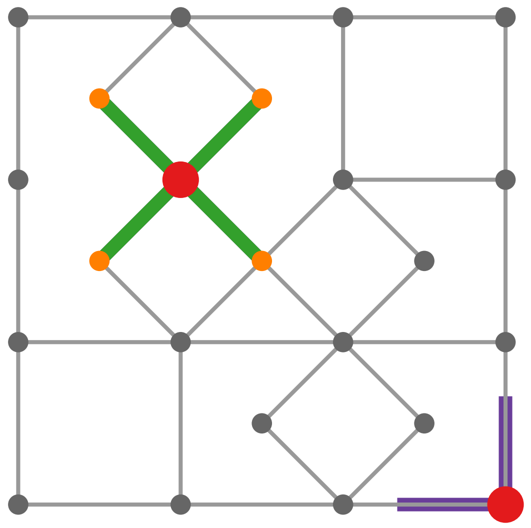
Initialization



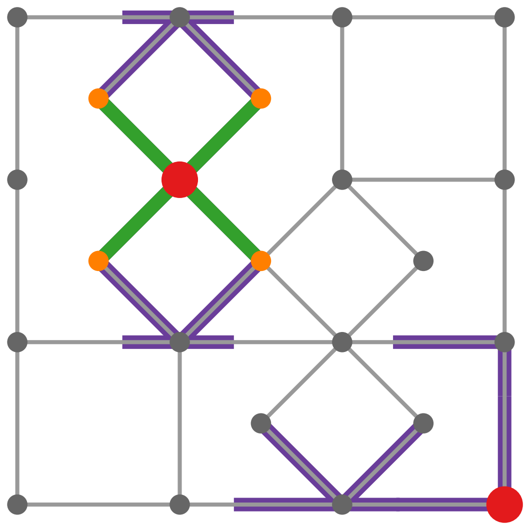
(0) Ball Growth



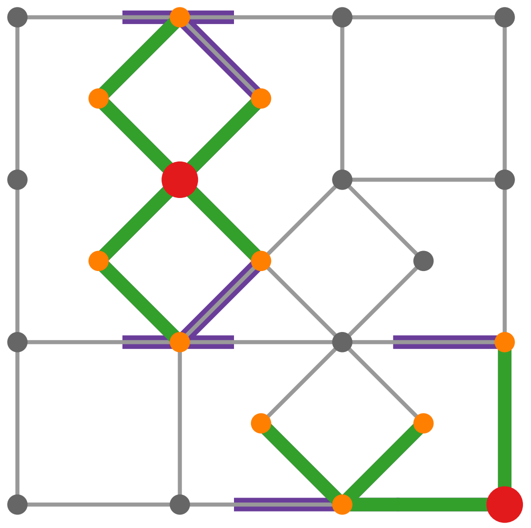
(0) SSSP Cover

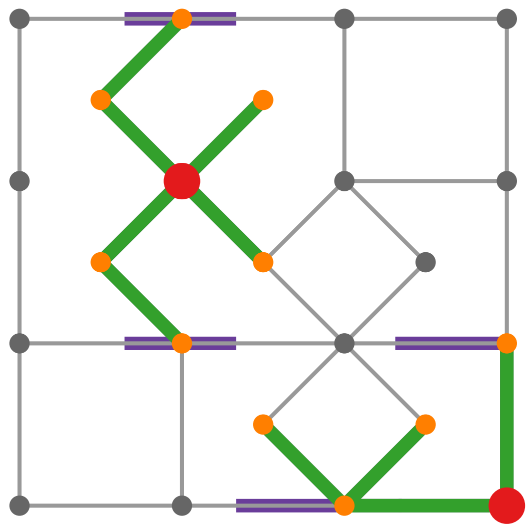


(1) Ball Growth

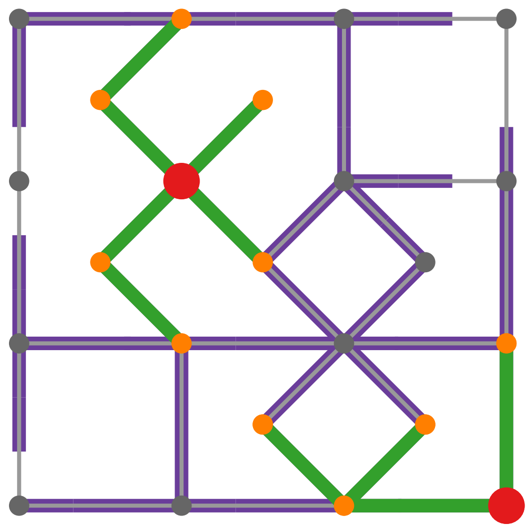


(1) SSSP Cover





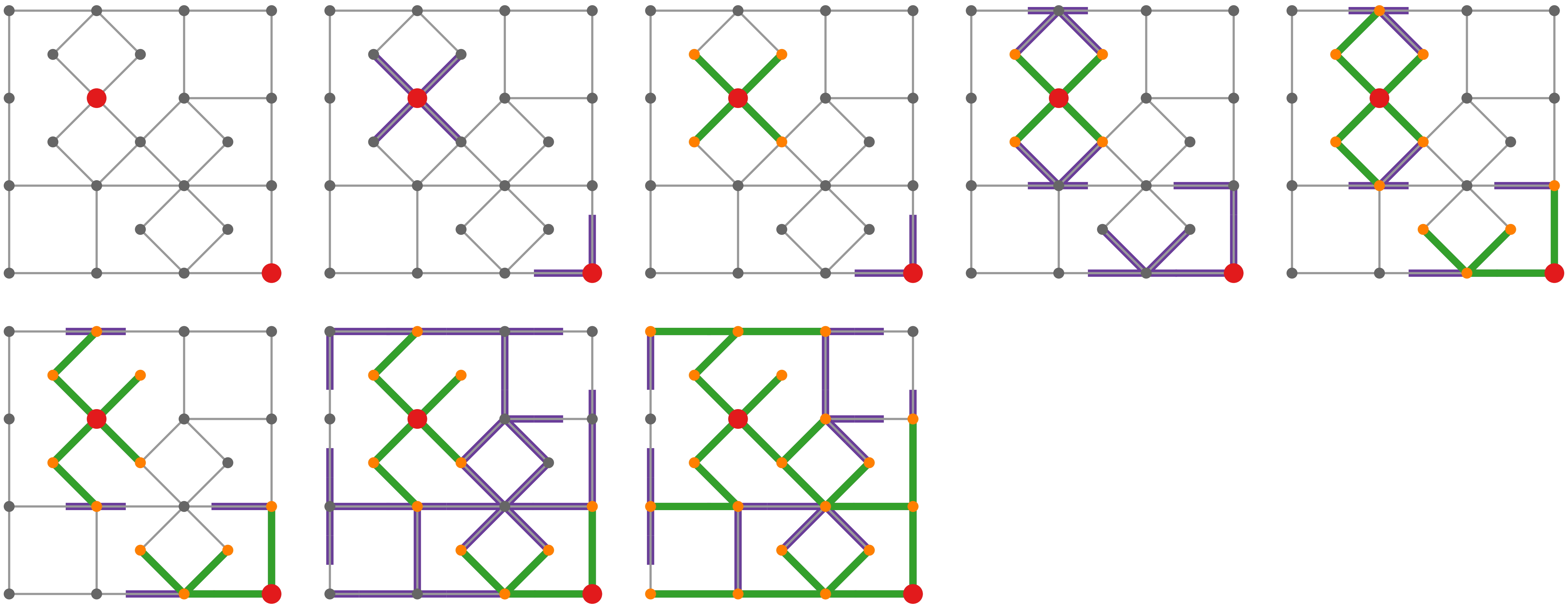
(1) Edge Deletion



(2) Ball Growth

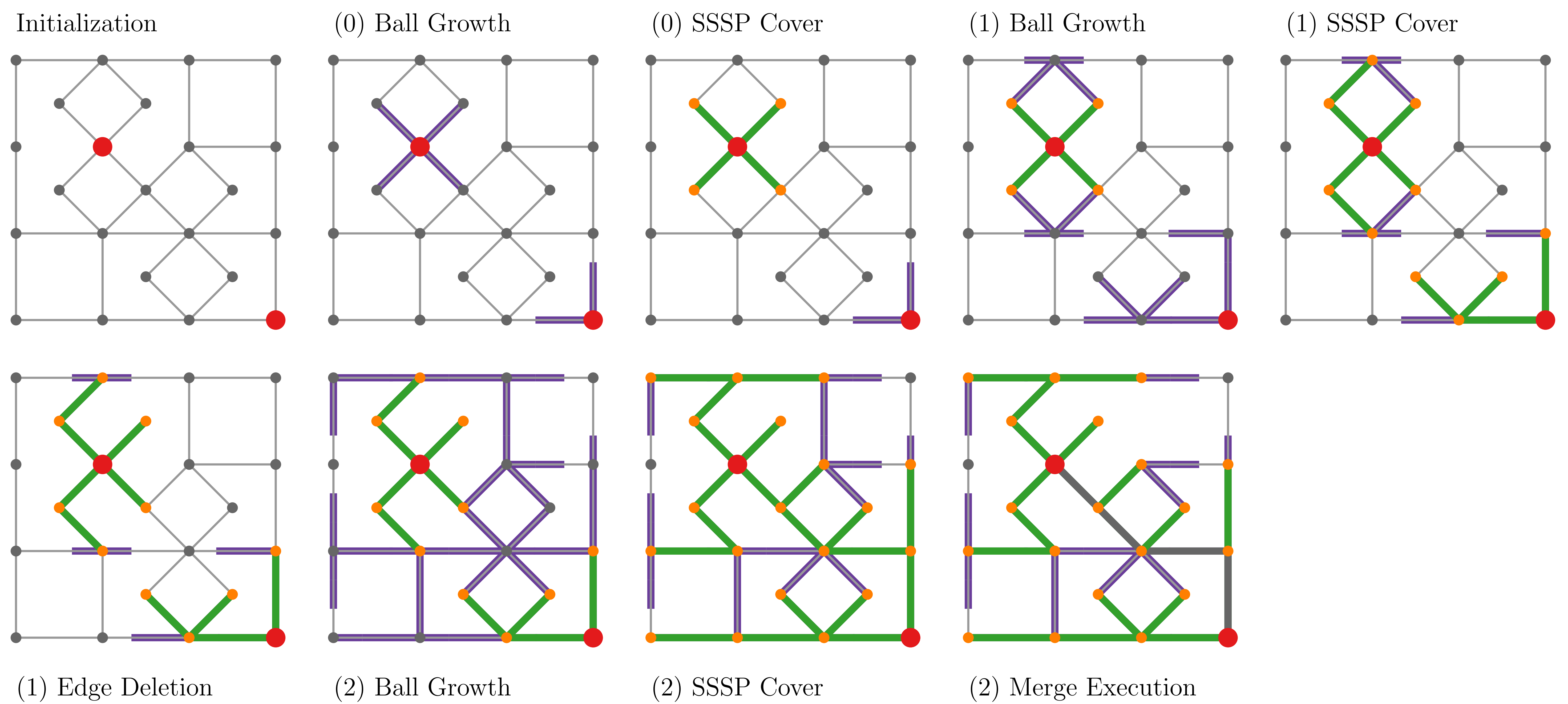
The Shell-Decomposition Algorithm: Building Blocks

Initialization (0) Ball Growth (0) SSSP Cover (1) Ball Growth (1) SSSP Cover



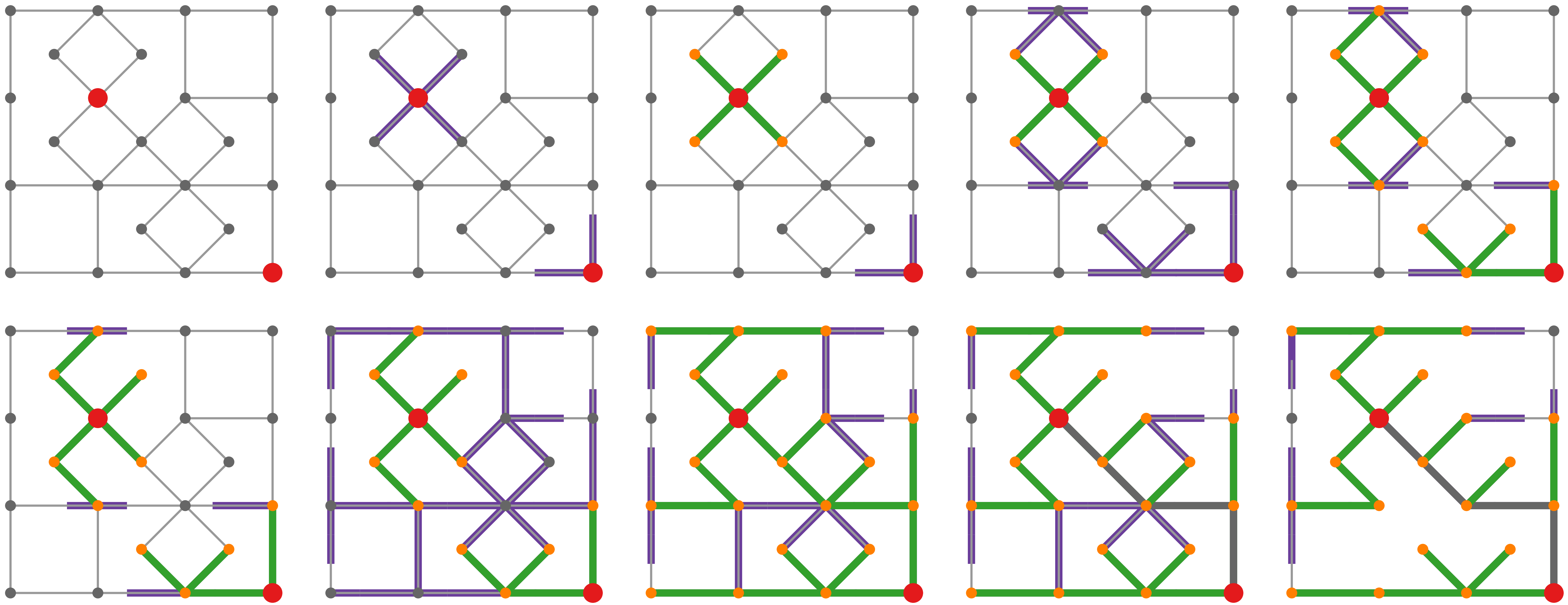
(1) Edge Deletion (2) Ball Growth (2) SSSP Cover

The Shell-Decomposition Algorithm: Building Blocks



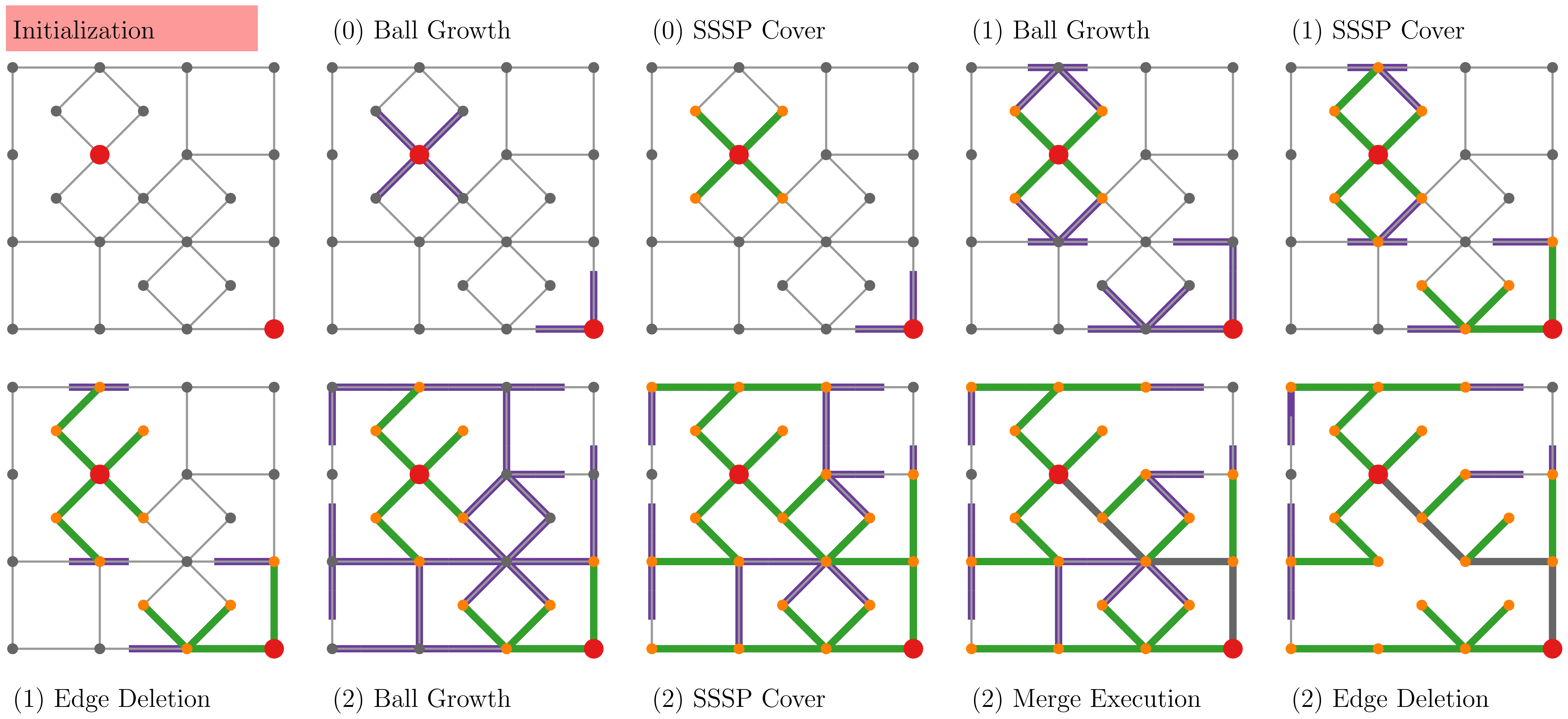
The Shell-Decomposition Algorithm: Building Blocks

Initialization · (0) Ball Growth · (0) SSSP Cover · (1) Ball Growth · (1) SSSP Cover

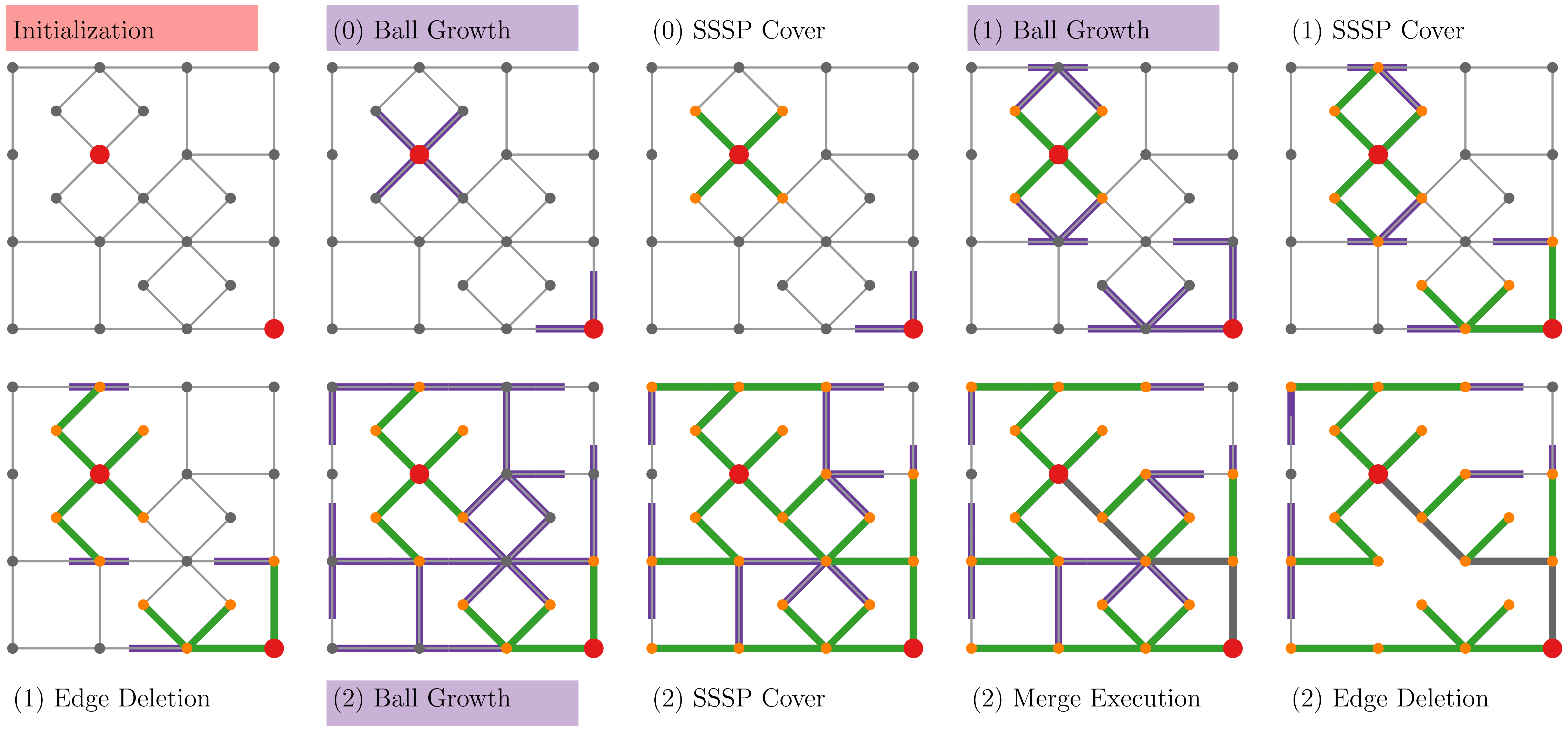


(1) Edge Deletion · (2) Ball Growth · (2) SSSP Cover · (2) Merge Execution · (2) Edge Deletion

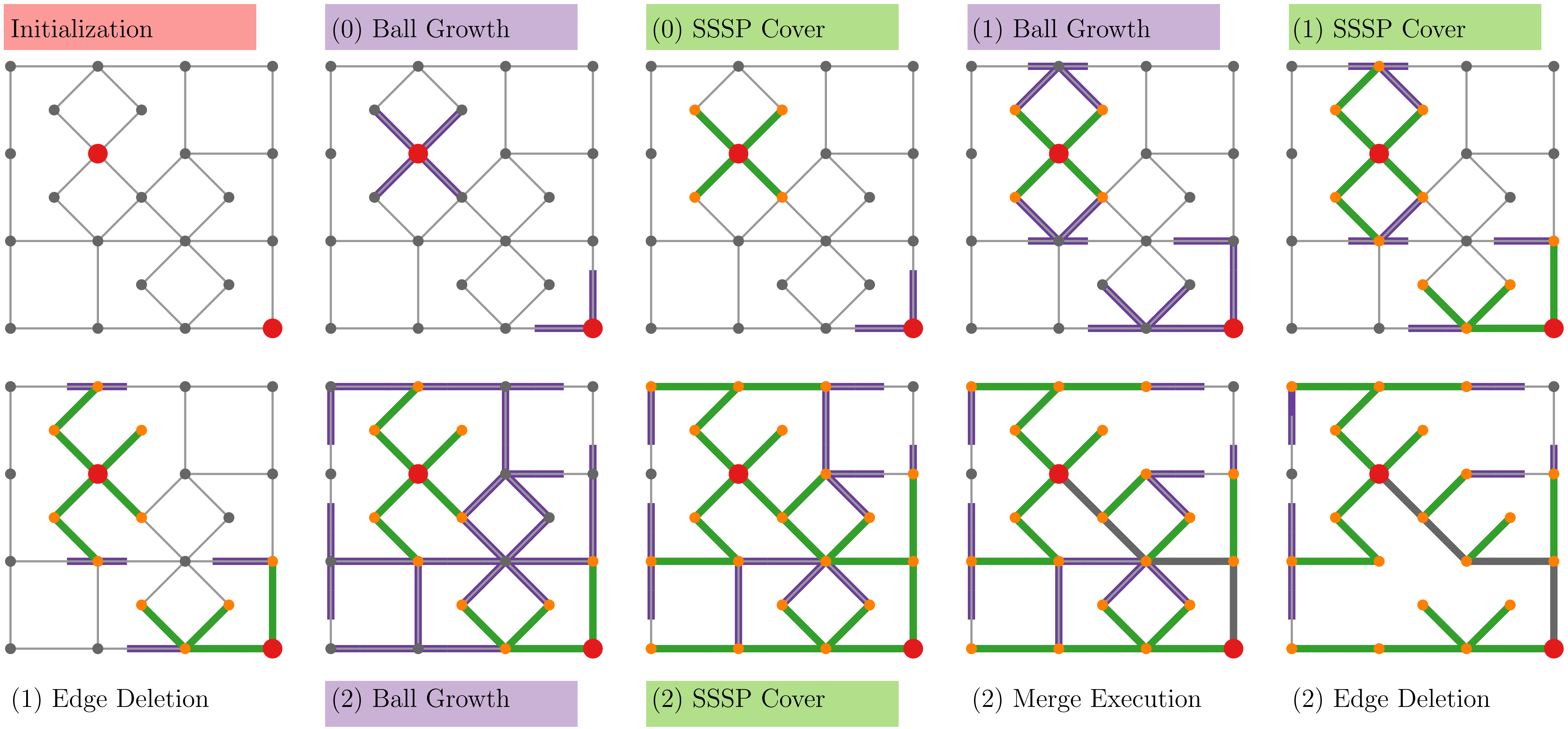
The Shell-Decomposition Algorithm: Building Blocks



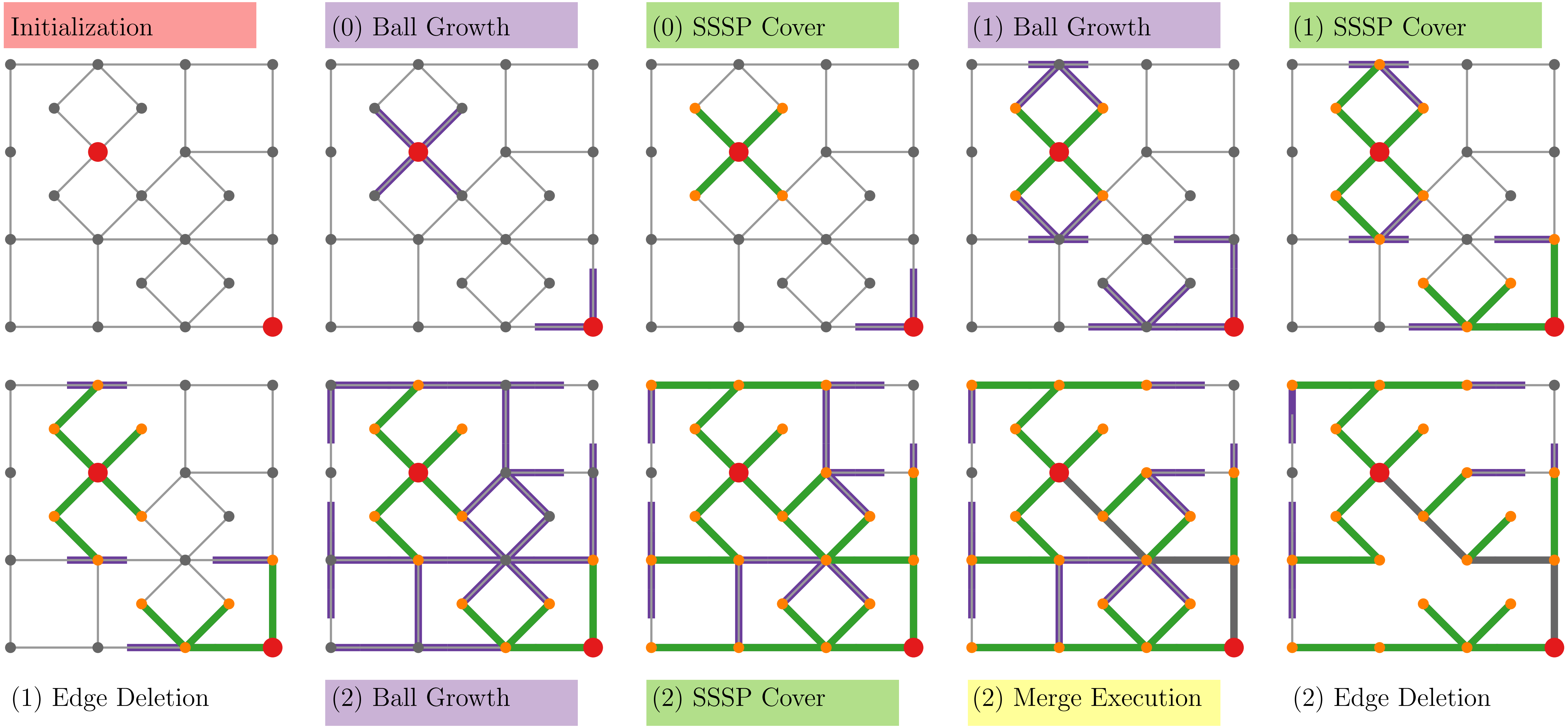
The Shell-Decomposition Algorithm: Building Blocks



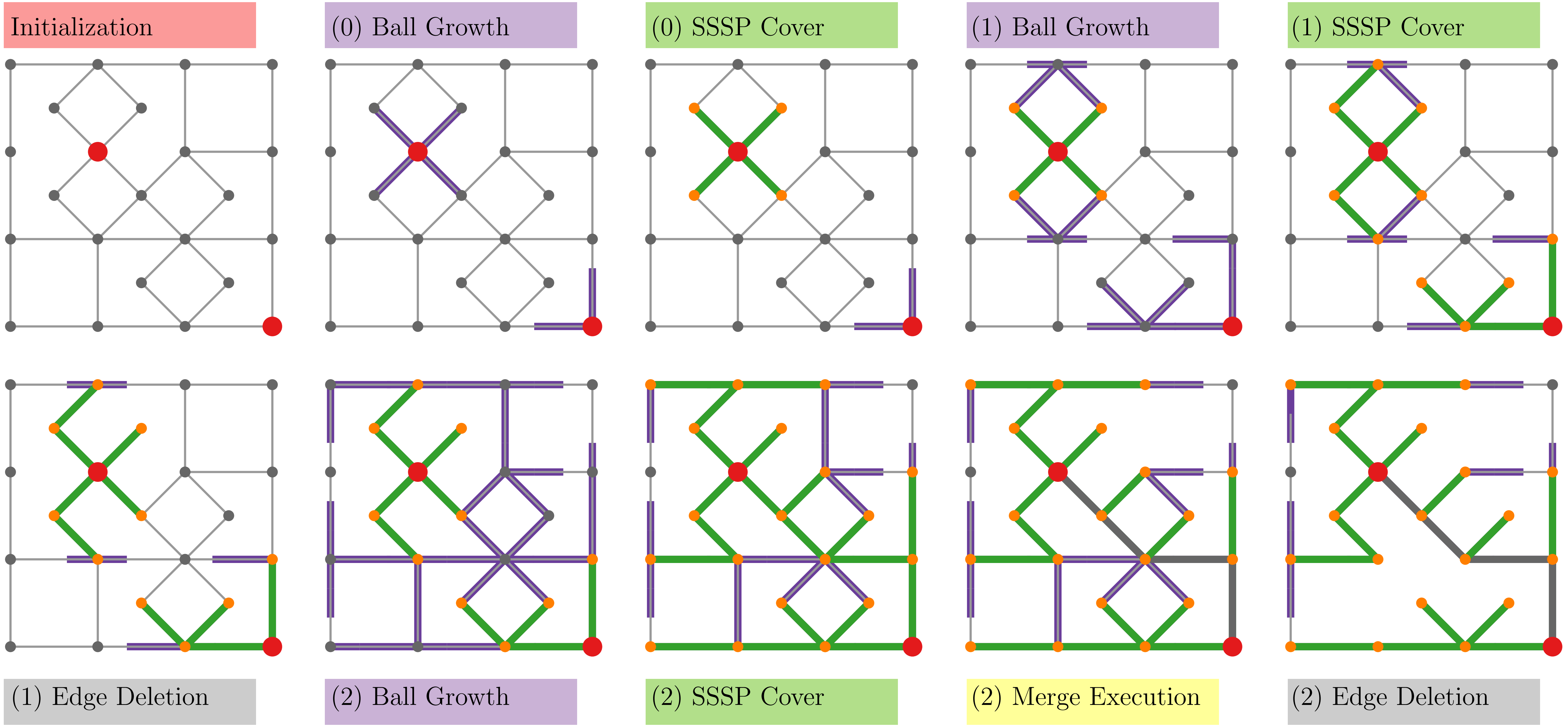
The Shell-Decomposition Algorithm: Building Blocks



The Shell-Decomposition Algorithm: Building Blocks

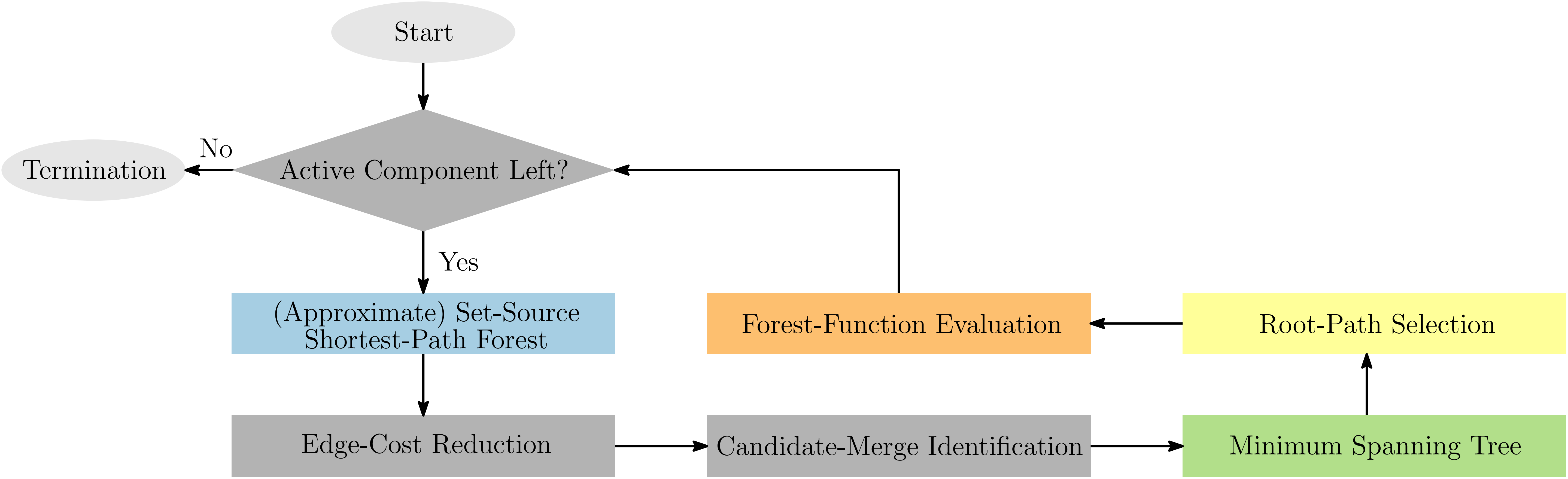


The Shell-Decomposition Algorithm: Building Blocks



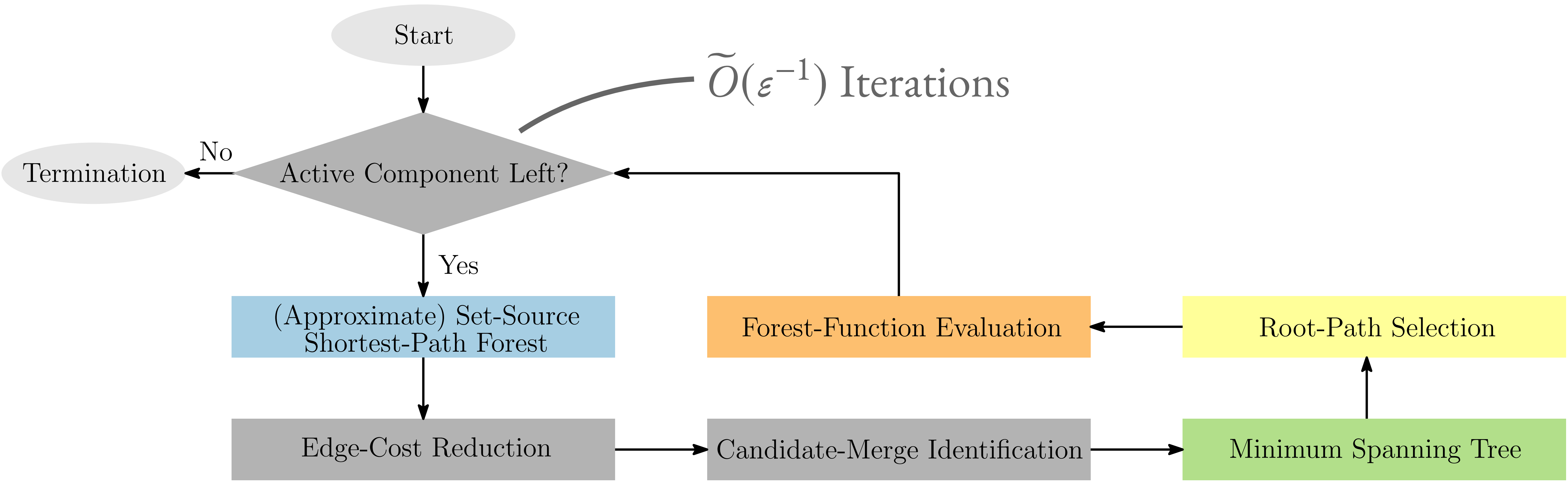
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



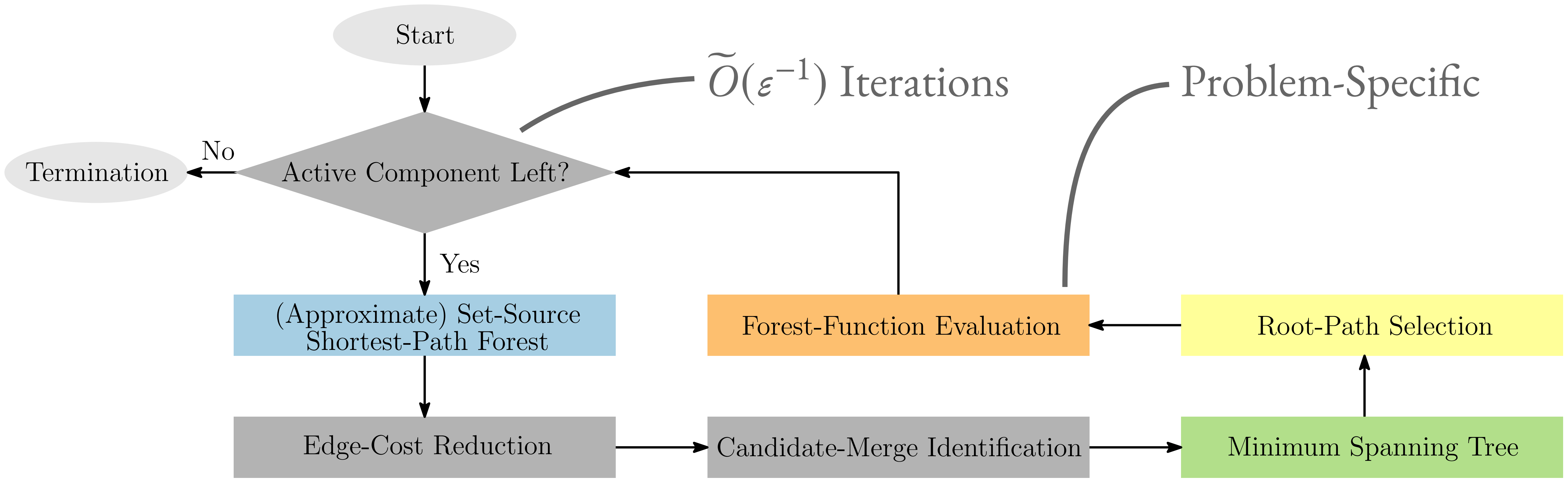
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



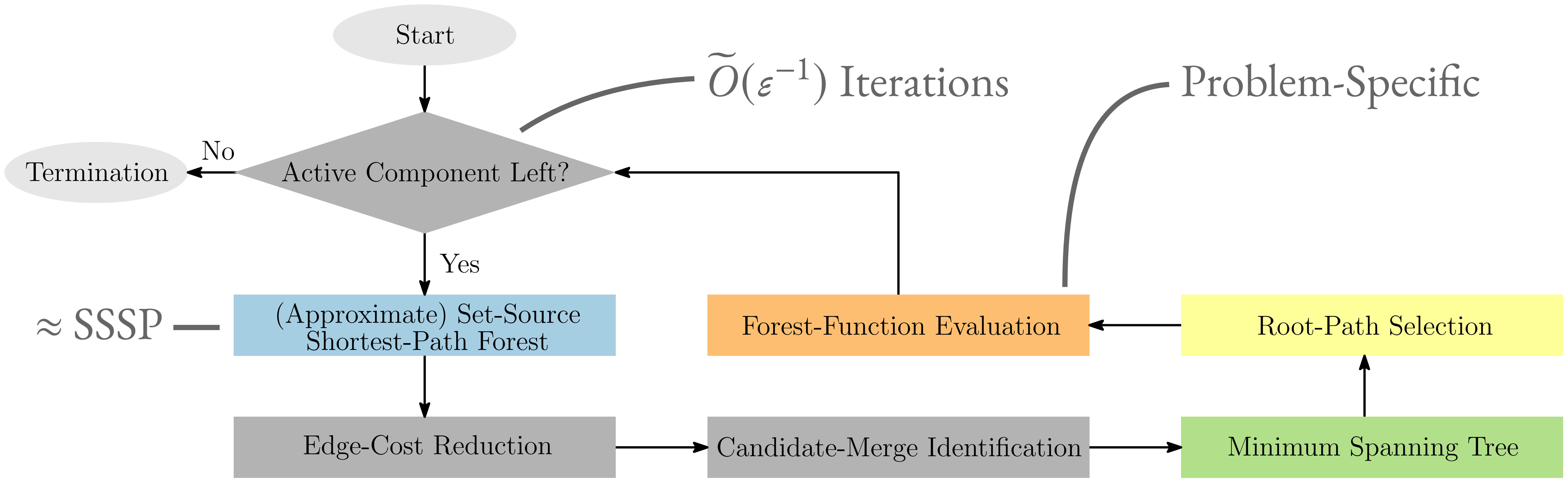
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



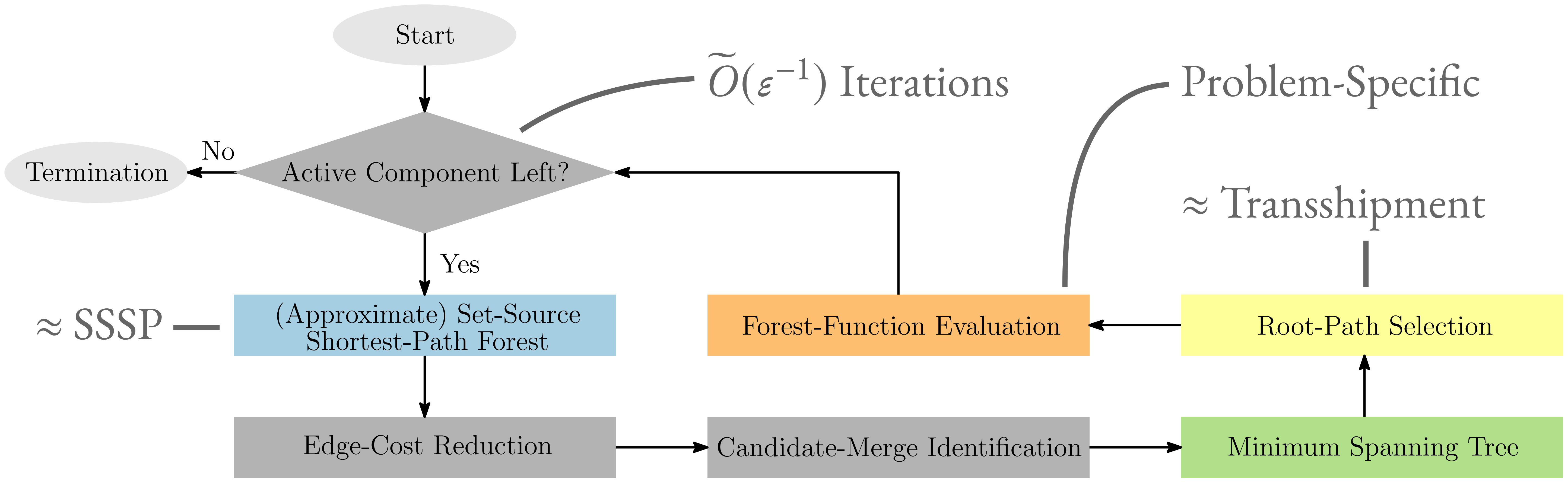
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



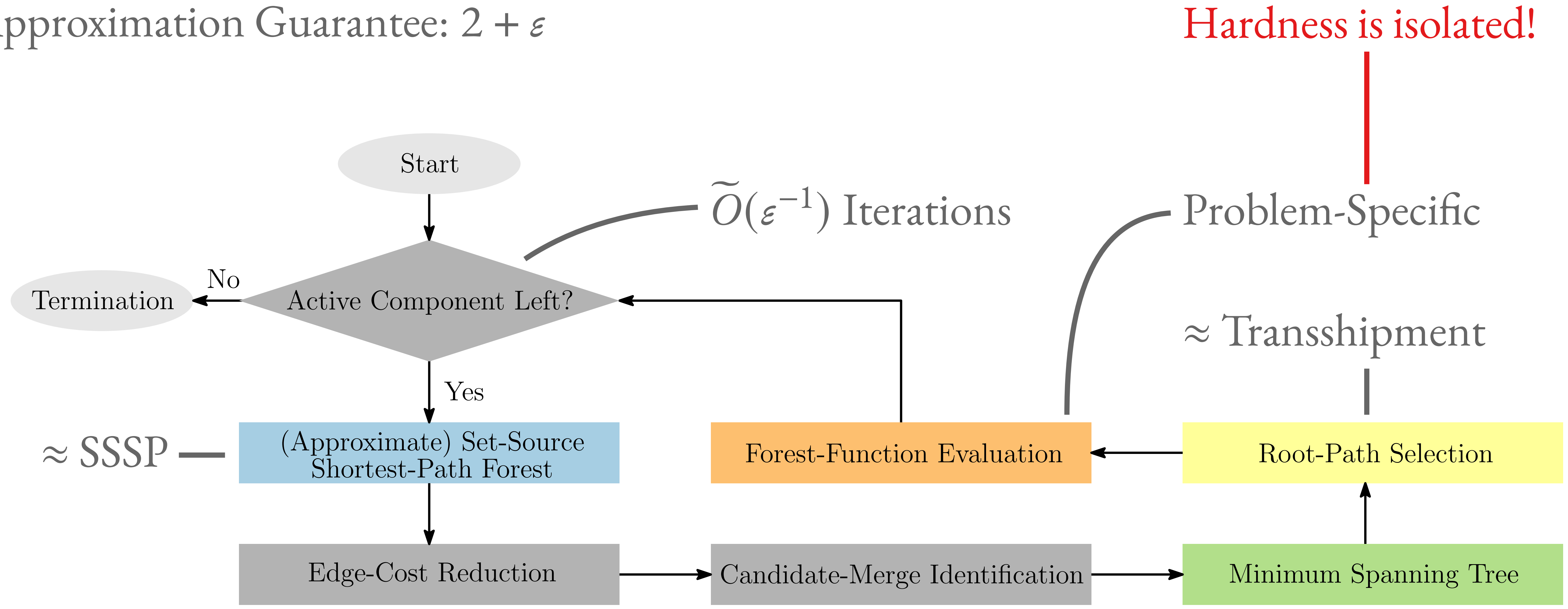
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



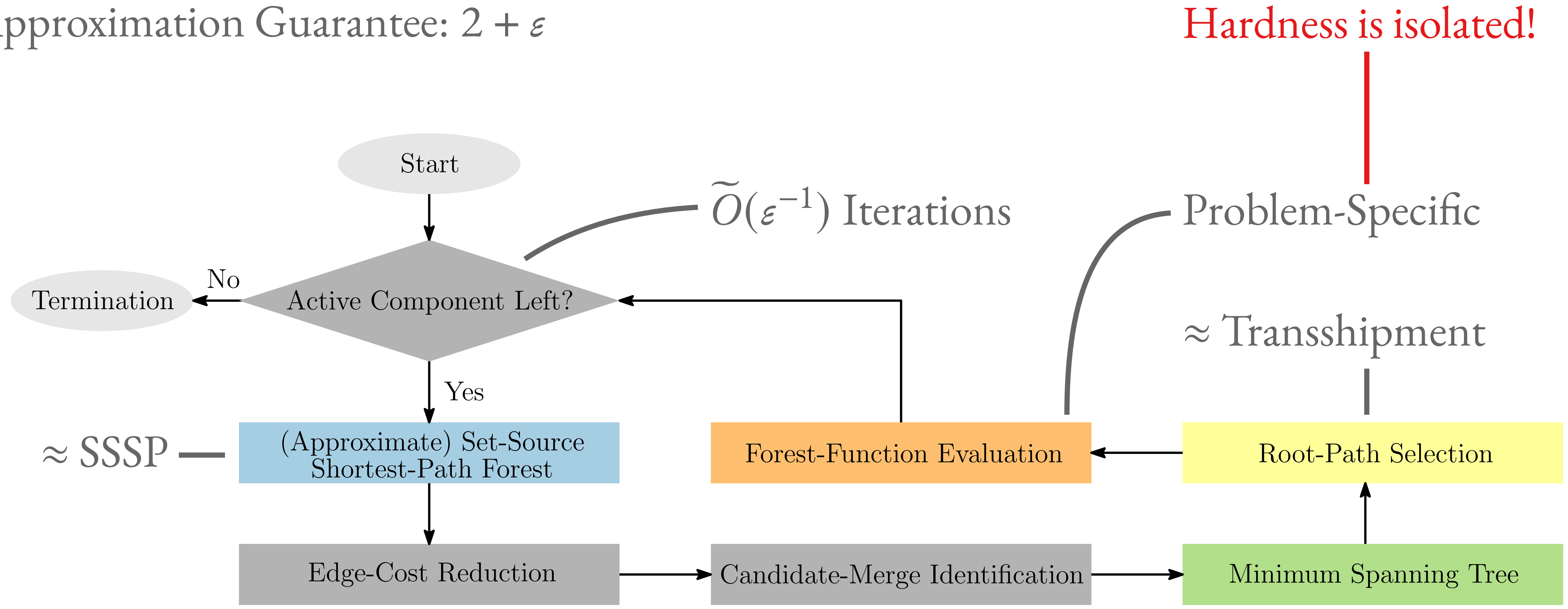
Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$



Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$

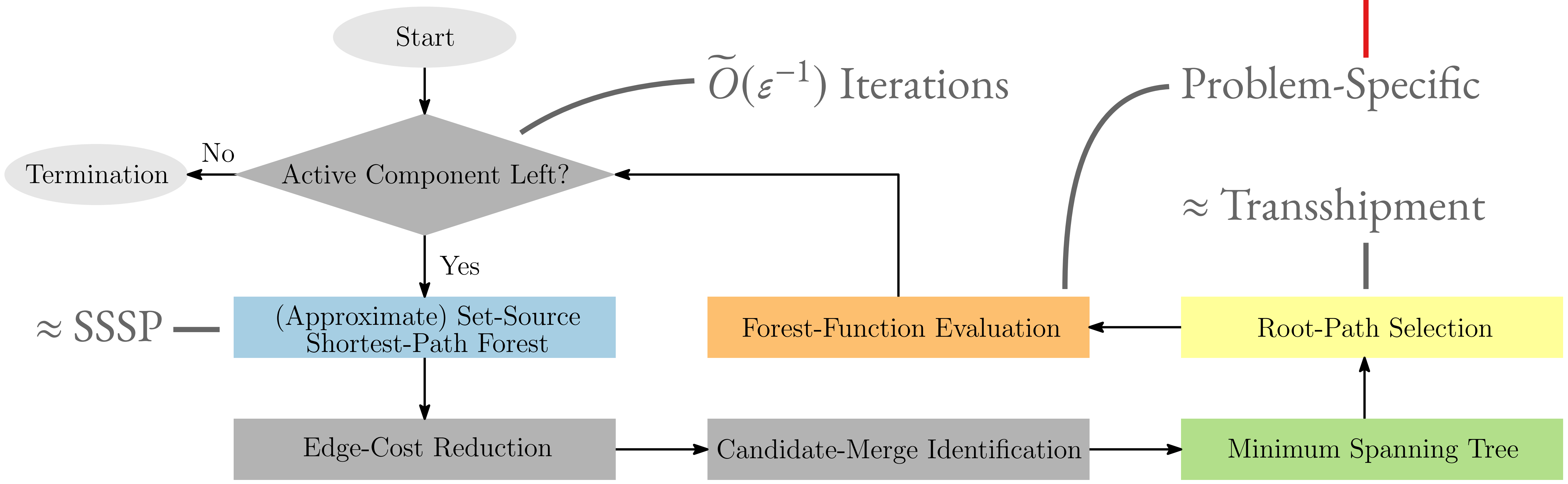


Complexity: $\widetilde{O}\left((\text{aSSSP} + \text{MST} + \text{RPS} + \text{FFE})\varepsilon^{-1}\right)$

Model-Agnostic Specification

Approximation Guarantee: $2 + \varepsilon$

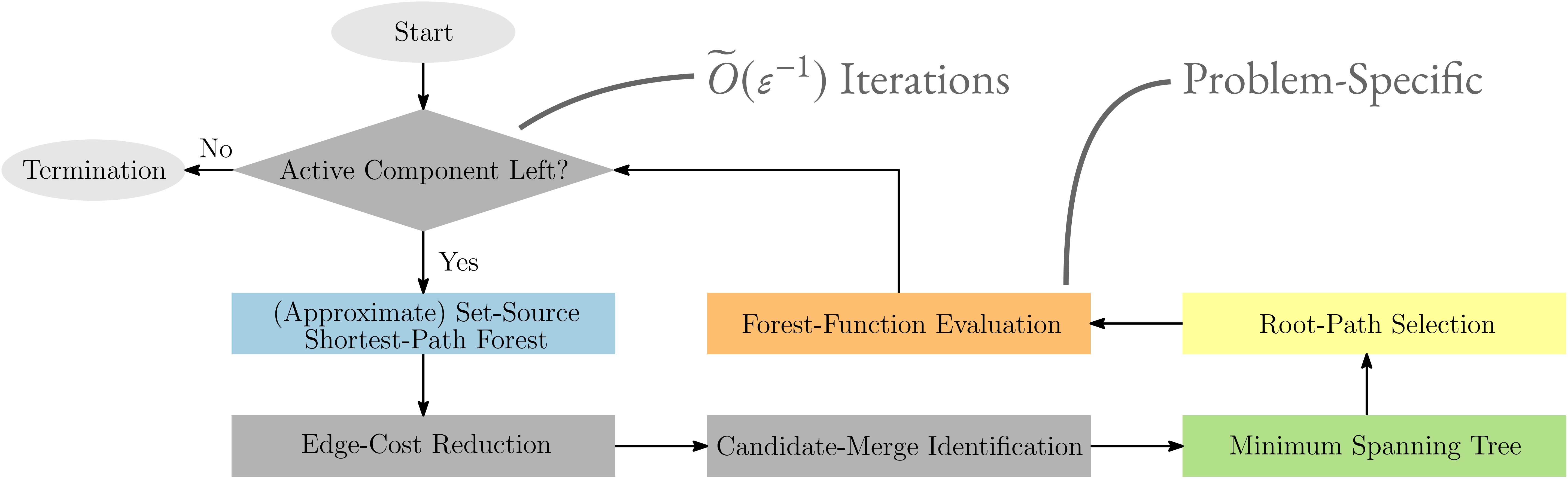
Hardness is isolated!



Complexity: $\widetilde{O}\left((\text{aSSSP} + \text{MST} + \text{RPS} + \text{FFE})\varepsilon^{-1}\right)$ — Model-agnostic!

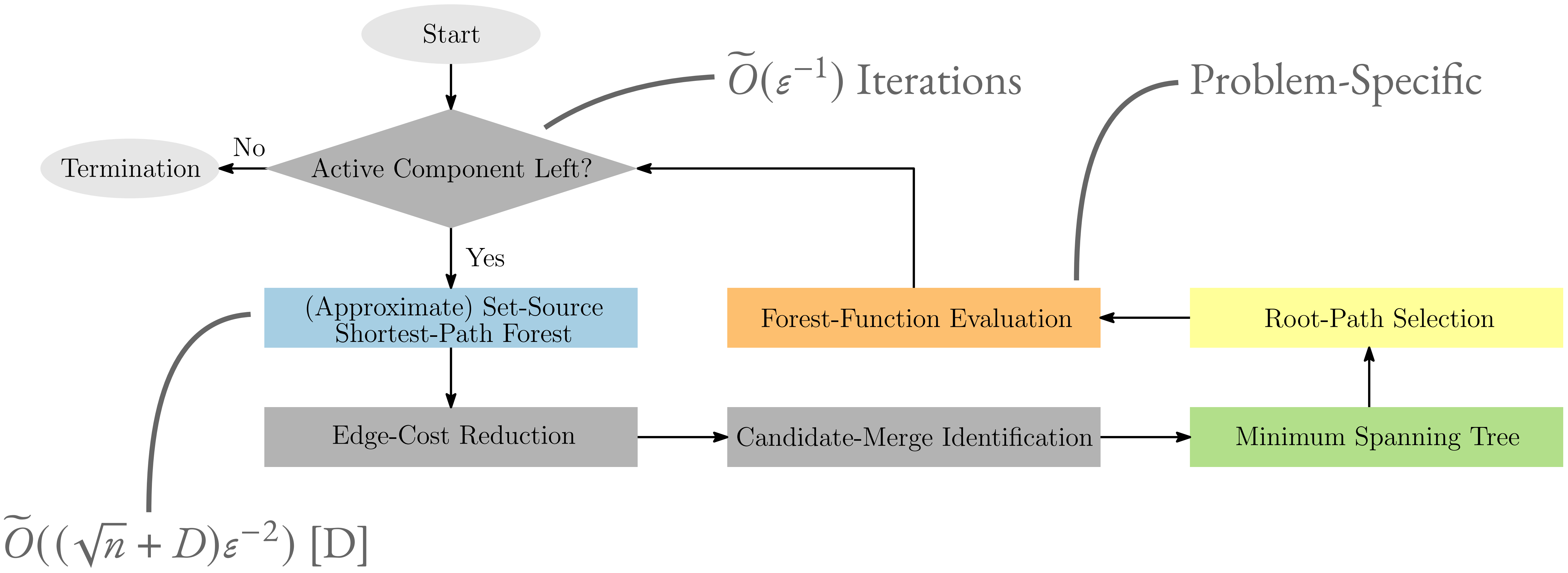
Model-Specific Implementation: CONGEST

$(2+\varepsilon)$-approximation with…



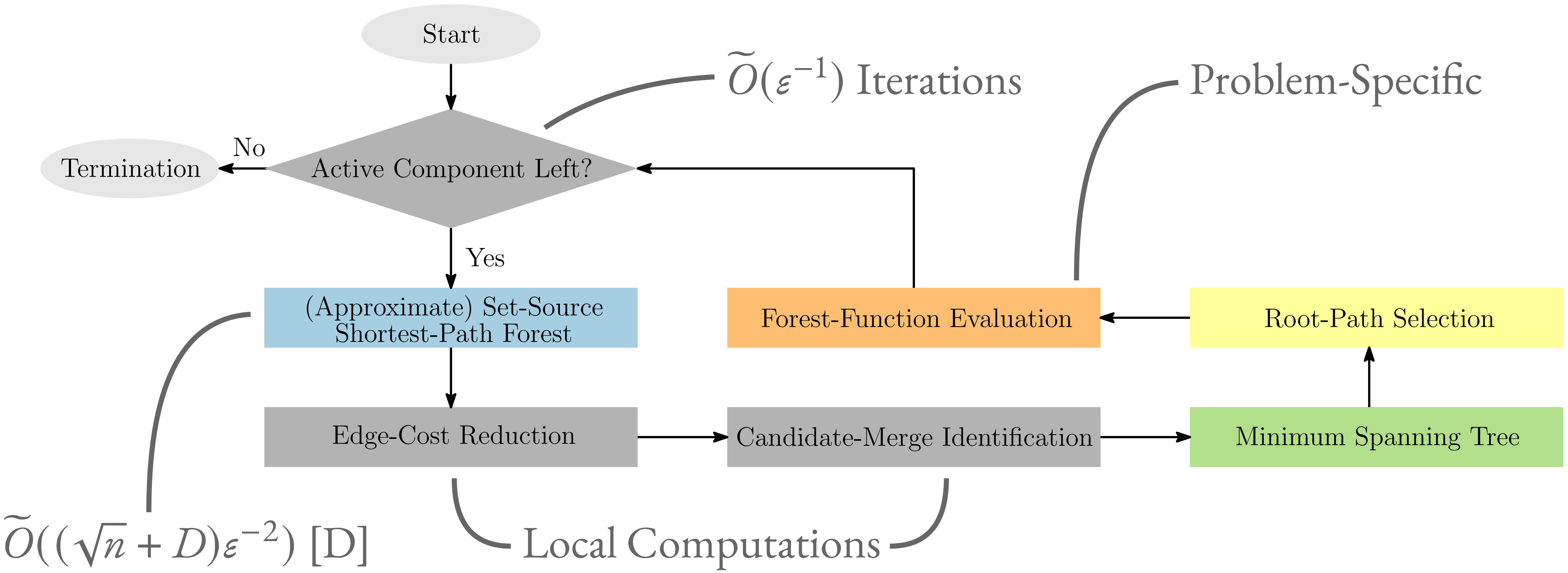
Model-Specific Implementation: Congest

$(2+\varepsilon)$-approximation with…



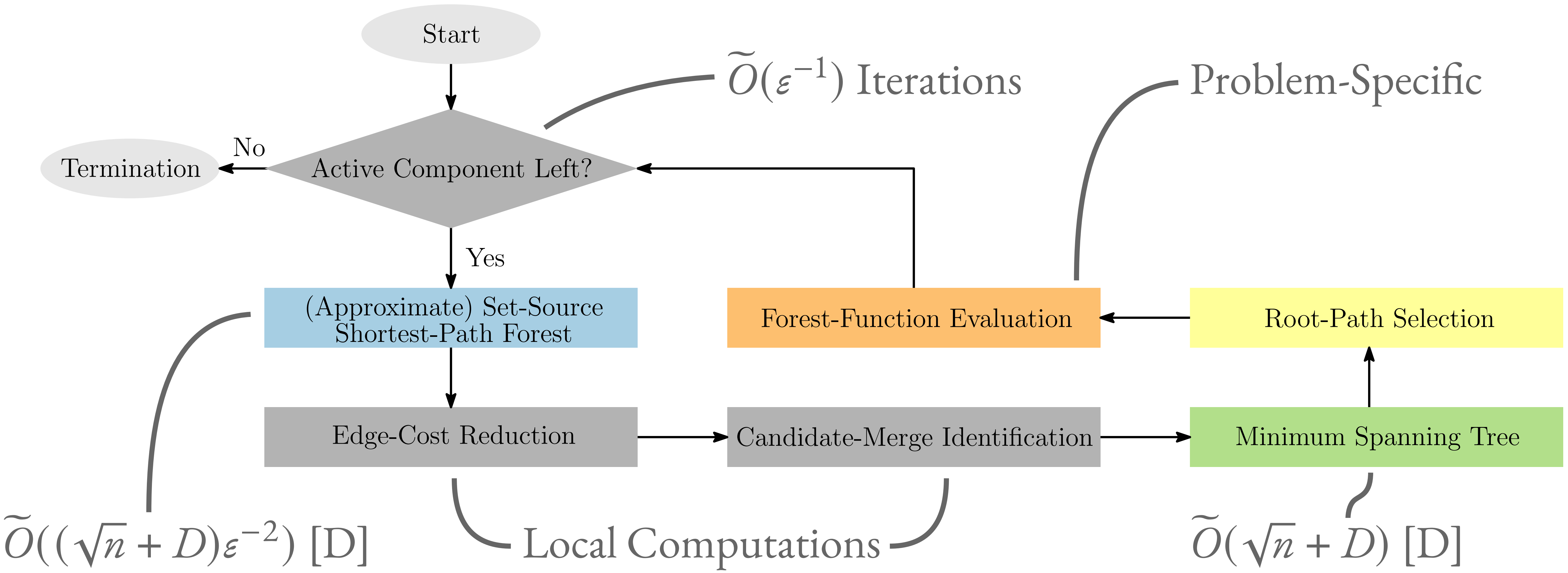
Model-Specific Implementation: Congest

(2 + ε)-approximation with…



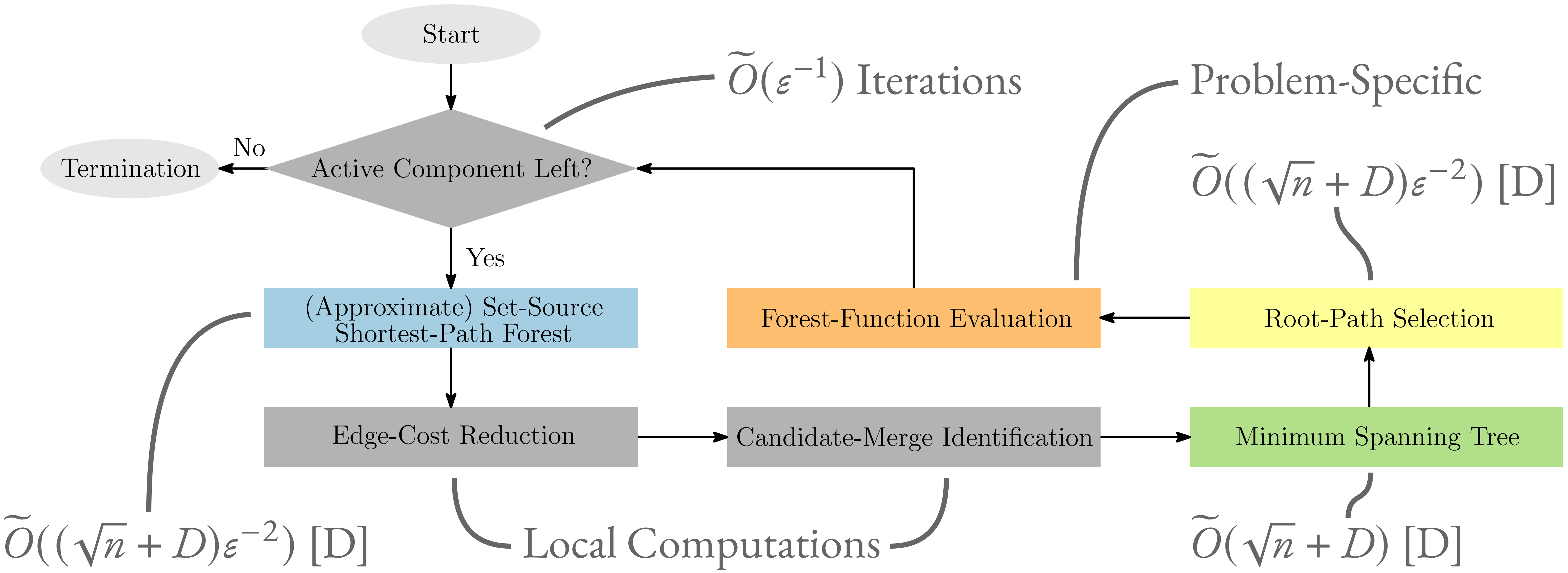
Model-Specific Implementation: Congest

$(2+\varepsilon)$-approximation with…



Model-Specific Implementation: Congest

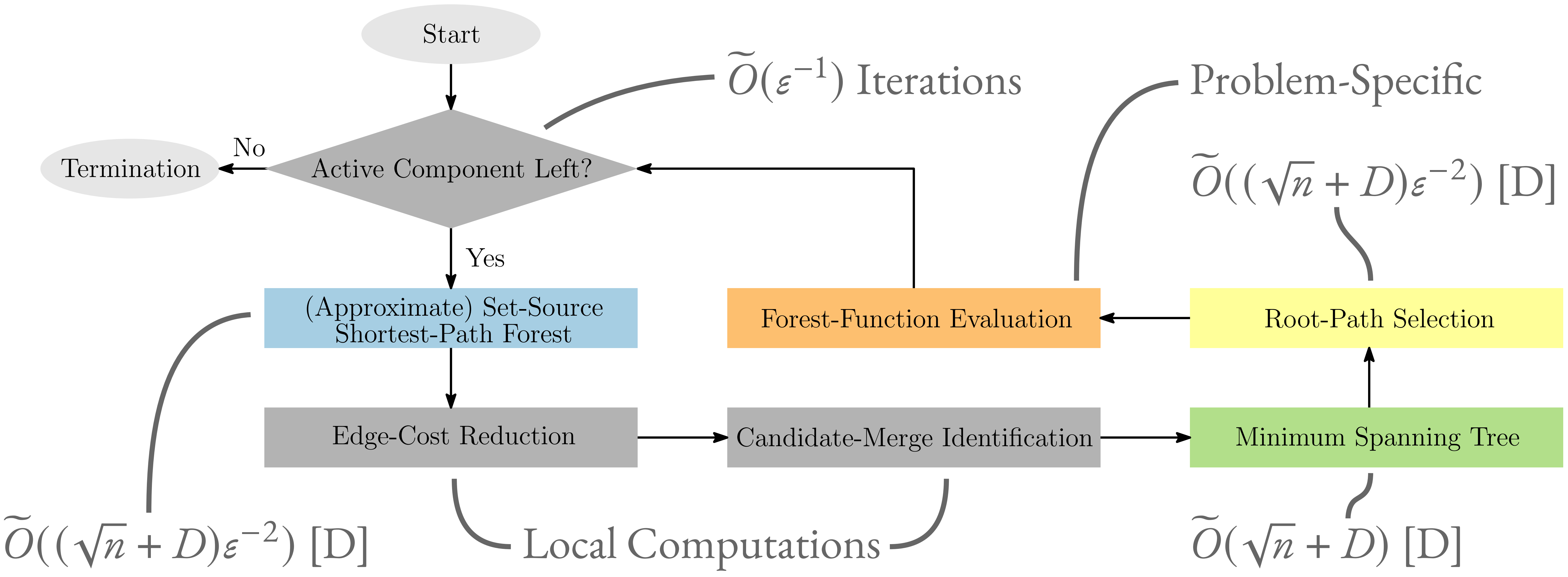
$(2+\varepsilon)$-approximation with…



Model-Specific Implementation: Congest

$(2+\varepsilon)$-approximation with…

Toward universal optimality,
we can replace $\sqrt{n}+D$ by $T^{PA}n^{o(1)}$



Problem-Specific Implementation: Steiner Forest in Congest

Interestingly, Congest complexity differs depending on the SF input representation!

Problem-Specific Implementation: Steiner Forest in Congest

Interestingly, Congest complexity differs depending on the SF input representation!

Problem	Input	LB	APX	Complexity
SF–IC	Component identifiers $\lambda\colon V \to [k] \cup \{\perp\}$; node v knows λ_v	$\widetilde{\Omega}(Q + k)$ R	$(2 + \varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n} + D\} + k)$

Problem-Specific Implementation: Steiner Forest in Congest

Interestingly, Congest complexity differs depending on the SF input representation!

Problem	Input	LB	APX	Complexity
SF–IC	Component identifiers $\lambda\colon V \to [k] \cup \{\perp\}$; node v knows λ_v	$\widetilde{\Omega}(Q+k)$ R	$(2+\varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\}+k)$
SF–CIC	As in SF–IC, but node v knows λ_v *and* $\lvert\{u \in V \mid \lambda_u = \lambda_v\}\rvert$	$\widetilde{\Omega}(Q+k)$ D	$(2+\varepsilon)$ R	$\widetilde{O}(n^{2/3}+D)$

Problem-Specific Implementation: Steiner Forest in Congest

Interestingly, Congest complexity differs depending on the SF input representation!

Problem	Input	LB	APX	Complexity
SF–IC	Component identifiers $\lambda\colon V \to [k] \cup \{\perp\}$; node v knows λ_v	$\widetilde{\Omega}(Q+k)$ R	$(2+\varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\}+k)$
SF–CIC	As in SF–IC, but node v knows λ_v *and* $\lvert\{u \in V \mid \lambda_u = \lambda_v\}\rvert$	$\widetilde{\Omega}(Q+k)$ D	$(2+\varepsilon)$ R	$\widetilde{O}(n^{2/3}+D)$
SF–CR	Each node v is given $\mathcal{R}_v \subseteq V \setminus \{v\}$	$\widetilde{\Omega}(Q+t)$ R	$(2+\varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\}+t)$

Problem-Specific Implementation: Steiner Forest in Congest

Interestingly, Congest complexity differs depending on the SF input representation!

Problem	Input	LB	APX	Complexity
SF–IC	Component identifiers $\lambda\colon V \to [k] \cup \{\perp\}$; node v knows λ_v	$\widetilde{\Omega}(Q+k)$ R	$(2+\varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\}+k)$
SF–CIC	As in SF–IC, but node v knows λ_v *and* $\lvert\{u \in V \mid \lambda_u = \lambda_v\}\rvert$	$\widetilde{\Omega}(Q+k)$ D	$(2+\varepsilon)$ R	$\widetilde{O}(n^{2/3}+D)$
SF–CR	Each node v is given $\mathcal{R}_v \subseteq V \setminus \{v\}$	$\widetilde{\Omega}(Q+t)$ R	$(2+\varepsilon)$ D	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\}+t)$
SF–SCR	$\mathcal{R} \subseteq \binom{V}{2}$; node v knows $\mathcal{R}_v = \{u \in V \mid \{u,v\} \in \mathcal{R}\}$	$\widetilde{\Omega}(Q+t)$ D	$(2+\varepsilon)$ R	$\widetilde{O}(\min\{T^{PA} n^{o(1)}, \sqrt{n}+D\})$

Outlook

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

General Framework for Model-Agnostic Approximation of CFPs

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in CONGEST

Open Questions

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Open Questions

- Problem Classes

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Open Questions

- Problem Classes
  - CFPs on Hypergraphs?

Outlook

- Our Main Contribution: Shell-Decomposition Algorithm
  - General Framework for Model-Agnostic Approximation of CFPs
  - Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest
- Open Questions
  - Problem Classes
    - CFPs on Hypergraphs?
    - CFPs with Non-Proper Forest Functions?

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Open Questions

- Problem Classes
  - CFPs on Hypergraphs?
  - CFPs with Non-Proper Forest Functions?
- Universal Optimality

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Open Questions

- Problem Classes
  - CFPs on Hypergraphs?
  - CFPs with Non-Proper Forest Functions?
- Universal Optimality
  - Hardness of *Disjoint* Aggregation (analogous to *Partwise* Aggregation)?

Outlook

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in Congest

Open Questions

- Problem Classes
  - CFPs on Hypergraphs?
  - CFPs with Non-Proper Forest Functions?
- Universal Optimality
  - Hardness of *Disjoint* Aggregation (analogous to *Partwise* Aggregation)?
  - Analog to Universal Optimality in Multi-Pass Streaming?

Thank You!

Our Main Contribution: Shell-Decomposition Algorithm

- General Framework for Model-Agnostic Approximation of CFPs
- Instantiated in 3 Models for 3 Problems, improving SOTA esp. for SF in CONGEST

Open Questions

- Problem Classes
  - CFPs on Hypergraphs?
  - CFPs with Non-Proper Forest Functions?
- Universal Optimality
  - Hardness of *Disjoint* Aggregation (analogous to *Partwise* Aggregation)?
  - Analog to Universal Optimality in Multi-Pass Streaming?