Nested Recursive Lexicographical Search: Finding all Markov Perfect Equilibria and Structural Estimation of Directional Dynamic Games

Fedor Iskhakov, Australian National University
John Rust, Georgetown University
Bertel Schjerning, University of Copenhagen

ZICE 2017
February 1, 2017

How to Compute All Markov Perfect Equilibria?

The Markov Perfect Equilibrium (MPE) concept of Maskin and Tirole (1988) is now a widely used in *empirical IO.*

How to Compute All Markov Perfect Equilibria?

The Markov Perfect Equilibrium (MPE) concept of Maskin and Tirole (1988) is now a widely used in *empirical IO.* However computing MPE remains a daunting computation problem

How to Compute All Markov Perfect Equilibria?

The Markov Perfect Equilibrium (MPE) concept of Maskin and Tirole (1988) is now a widely used in *empirical IO.* However computing MPE remains a daunting computation problem

Quote (Hörner *et. al. Econometrica* 2011)

"Dynamic games are difficult to solve. In repeated games, finding some equilibrium is easy, as any repetition of a stage-game Nash equilibrium will do. This is not the case in stochastic games. The characterization of even the most elementary equilibria for such games, namely (stationary) Markov equilibria, in which continuation strategies depend on the current state only, turns out to be often challenging."

Finding even a single MPE is challenging!

- How do people "find" MPEs?

Finding even a single MPE is challenging!

- How do people "find" MPEs?
- Theorists:

Finding even a single MPE is challenging!

- How do people "find" MPEs?
- Theorists: Guess and Verify

Finding even a single MPE is challenging!

- How do people "find" MPEs?
- Theorists: Guess and Verify
- Applied people:

Finding even a single MPE is challenging!

- How do people "find" MPEs?
- Theorists: Guess and Verify
- Applied people: Iterate on the player Bellman equations

Finding even a single MPE is challenging!

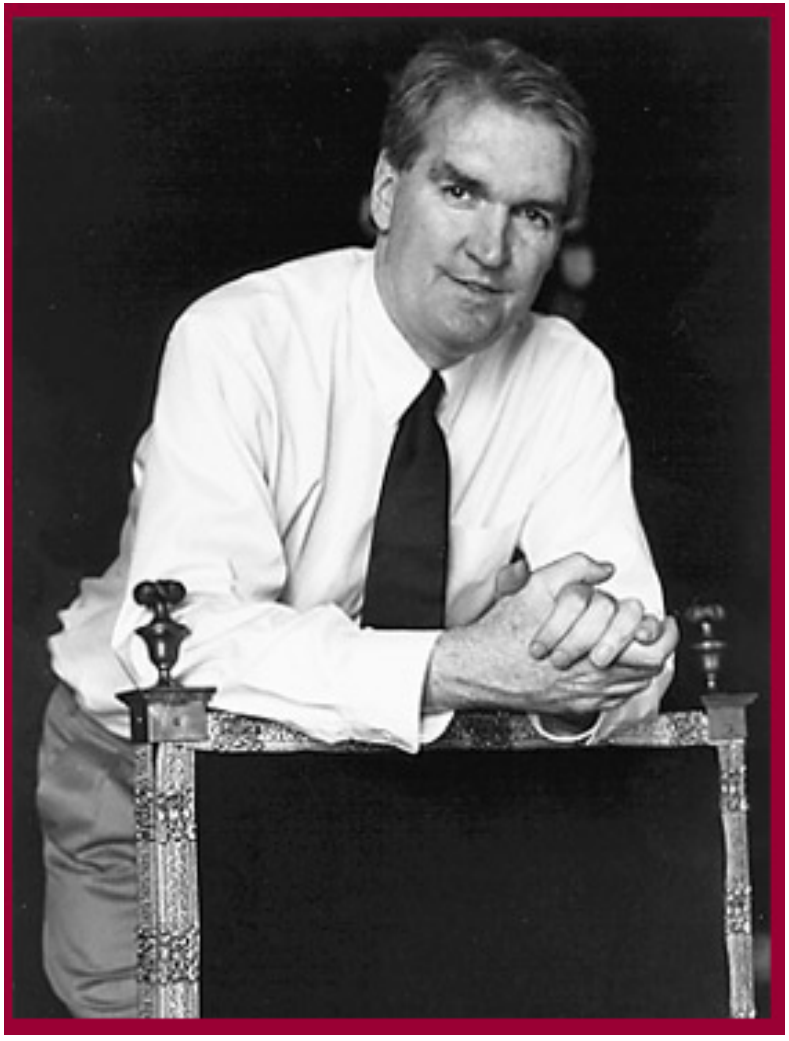
- How do people "find" MPEs?
- Theorists: Guess and Verify
- Applied people: Iterate on the player Bellman equations
- Pakes and McGuire (1994): some of the earliest work on computing MPE. Proposed a deterministic, iterative algorithm to compute MPE. Found a curse of dimensionality in trying to solve MPE model of firm dynamics with even moderate numbers of firms

Finding even a single MPE is challenging!

- How do people "find" MPEs?
- Theorists: Guess and Verify
- Applied people: Iterate on the player Bellman equations
- Pakes and McGuire (1994): some of the earliest work on computing MPE. Proposed a deterministic, iterative algorithm to compute MPE. Found a curse of dimensionality in trying to solve MPE model of firm dynamics with even moderate numbers of firms
- Pakes and McGuire (2001): Proposed a *stochastic algorithm* to approximate an MPE, in an attempt to break this curse of dimensionality

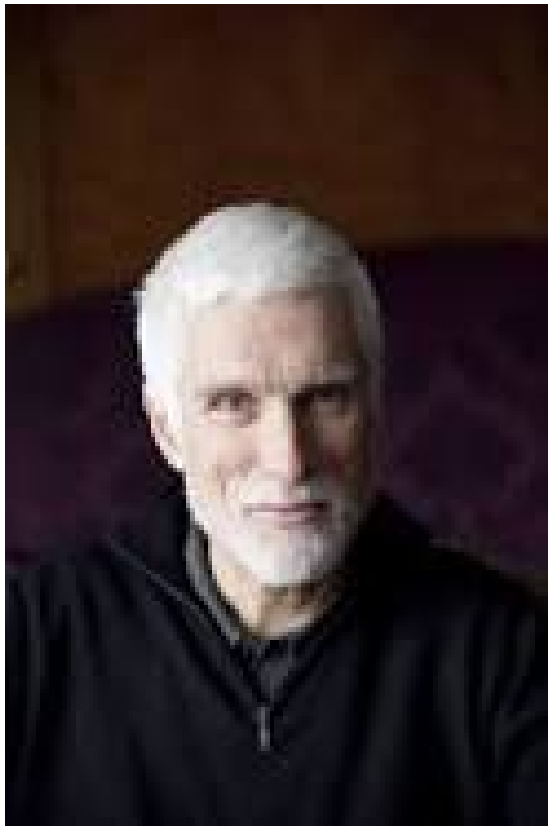
Eric Maskin: taught my game theory class at MIT

One of my classmates: Tim Kehoe



Another classmate: David Levine

Another classmate: Drew Fudenberg



Another classmate: Jean Tirole

Motivation: Collusion on the beach

Peter Brown: Amcor Managing Director

Harry Debney: Visy CEO

The Australian cardboard collusion case

- The Australian market for *cardboard* (CFP) is essentially a duopoly

The Australian cardboard collusion case

- The Australian market for *cardboard* (CFP) is essentially a duopoly
- Between 2000 and 2005 the two firms, *Visy* and *Amcor* colluded to raise the price of CFP

The Australian cardboard collusion case

- The Australian market for *cardboard* (CFP) is essentially a duopoly
- Between 2000 and 2005 the two firms, *Visy* and *Amcor* colluded to raise the price of CFP
- I was hired to estimate the damage caused by the collusion, which requires predicting what CFP prices would have been in the absence of collusion

The Australian cardboard collusion case

- The Australian market for *cardboard* (CFP) is essentially a duopoly
- Between 2000 and 2005 the two firms, *Visy* and *Amcor* colluded to raise the price of CFP
- I was hired to estimate the damage caused by the collusion, which requires predicting what CFP prices would have been in the absence of collusion
- My opinion was that the "but-for" CFP prices are those predicted by Bertrand price competition in the short run, with *leapfrogging investments* by the two firms over the longer run as they vie for low-cost leadership

Amcor's New B9 Paper Mill

Main Mill Site, Botany Bay Road, Botany Bay NSW

Source: Amcor

A cardboard corrugator

The Problem with "Iterating on the Value Function"

- Solving infinite horizon dynamic games is difficult in part because there is no "last period" to start a standard backward induction calculation.

The Problem with "Iterating on the Value Function"

- Solving infinite horizon dynamic games is difficult in part because there is no "last period" to start a standard backward induction calculation.
- The applied version of "guess and verify" is to make a guess about the players' value functions and then use this guess as a starting point for *sucessive approximations* of the players' Bellman equations (value function iteration)

The Problem with "Iterating on the Value Function"

- Solving infinite horizon dynamic games is difficult in part because there is no "last period" to start a standard backward induction calculation.
- The applied version of "guess and verify" is to make a guess about the players' value functions and then use this guess as a starting point for *sucessive approximations* of the players' Bellman equations (value function iteration)
- The problem is that in dynamic games, the requisite continuity conditions for the Bellman equations to constitute *contraction mappings* generally fails to hold.

The Problem with "Iterating on the Value Function"

- Solving infinite horizon dynamic games is difficult in part because there is no "last period" to start a standard backward induction calculation.
- The applied version of "guess and verify" is to make a guess about the players' value functions and then use this guess as a starting point for *sucessive approximations* of the players' Bellman equations (value function iteration)
- The problem is that in dynamic games, the requisite continuity conditions for the Bellman equations to constitute *contraction mappings* generally fails to hold.
- This implies that successive approximations is not guaranteed to converge to a fixed point/MPE, unlike in the case of single agent "game against nature" problems

Successive Approximations and Equilibrium Selection

- Sometimes successive approximations does converge, and if it does, one can prove that it converges to an set of value functions that constitutes an MPE of the game $\mathcal{G}$

Successive Approximations and Equilibrium Selection

- Sometimes successive approximations does converge, and if it does, one can prove that it converges to an set of value functions that constitutes an MPE of the game $\mathcal{G}$
- However if there are multiple MPE, successive approximations can converge to *different MPE* depending on how the algorithm is initialized

Successive Approximations and Equilibrium Selection

- Sometimes successive approximations does converge, and if it does, one can prove that it converges to an set of value functions that constitutes an MPE of the game $\mathcal{G}$
- However if there are multiple MPE, successive approximations can converge to *different MPE* depending on how the algorithm is initialized
- Despite these problems, applied work in economics typically relies on successive approximations as the method of choice.

Successive Approximations and Equilibrium Selection

- Sometimes successive approximations does converge, and if it does, one can prove that it converges to an set of value functions that constitutes an MPE of the game $\mathcal{G}$
- However if there are multiple MPE, successive approximations can converge to *different MPE* depending on how the algorithm is initialized
- Despite these problems, applied work in economics typically relies on successive approximations as the method of choice. People seem unaware that this algorithm can be inadvertently operating as an *equilibrium selection mechanism*

Existing methods for finding ALL MPE

- Much better to compute *all* MPE and select one of them on *economic grounds* rather than haphazardly selecting an MPE depending on how the solution algorithm is initialized

Existing methods for finding ALL MPE

- Much better to compute *all* MPE and select one of them on *economic grounds* rather than haphazardly selecting an MPE depending on how the solution algorithm is initialized
- Homotopy/path following methods (Borkovsky *et. al.* 2010 and Besanko *et. al.* 2010)

Existing methods for finding ALL MPE

- Much better to compute *all* MPE and select one of them on *economic grounds* rather than haphazardly selecting an MPE depending on how the solution algorithm is initialized
- Homotopy/path following methods (Borkovsky *et. al.* 2010 and Besanko *et. al.* 2010)
- Algebraic methods (Datta 2010, Judd *et. al.* 2012)

Existing methods for finding ALL MPE

- Much better to compute *all* MPE and select one of them on *economic grounds* rather than haphazardly selecting an MPE depending on how the solution algorithm is initialized
- Homotopy/path following methods (Borkovsky *et. al.* 2010 and Besanko *et. al.* 2010)
- Algebraic methods (Datta 2010, Judd *et. al.* 2012)
- Problem with the homotopy methods: may not be able to follow all the bifurcations in the path from a unique equilibrium of a "perturbed problem" to find all MPE of the problem you want to solve

Existing methods for finding ALL MPE

- Much better to compute *all* MPE and select one of them on *economic grounds* rather than haphazardly selecting an MPE depending on how the solution algorithm is initialized
- Homotopy/path following methods (Borkovsky *et. al.* 2010 and Besanko *et. al.* 2010)
- Algebraic methods (Datta 2010, Judd *et. al.* 2012)
- Problem with the homotopy methods: may not be able to follow all the bifurcations in the path from a unique equilibrium of a "perturbed problem" to find all MPE of the problem you want to solve
- Problem with the algebraic methods: limited to problems where the equations defining the state-specific Nash equilibria can be expressed as systems of polynomial equations that have specific forms.

Our approach: *Recursive Lexicographical Search*

- This is a systematic approach to "building" the set of all MPE from the MPE to simpler dynamic games we refer to as *stage games* of the original dynamic game $\mathcal{G}$.

Our approach: *Recursive Lexicographical Search*

- This is a systematic approach to "building" the set of all MPE from the MPE to simpler dynamic games we refer to as *stage games* of the original dynamic game $\mathcal{G}$.
- It is based on a generalization of backward induction that is applicable to a class of dynamic games that we call *directional dynamic games* (DDGs)

Our approach: *Recursive Lexicographical Search*

- This is a systematic approach to "building" the set of all MPE from the MPE to simpler dynamic games we refer to as *stage games* of the original dynamic game $\mathcal{G}$.
- It is based on a generalization of backward induction that is applicable to a class of dynamic games that we call *directional dynamic games* (DDGs)
- A DDG is a game where a subset of the state variables evolve in a manner that satisfies certain conditions including an intuitive notion of *directionality.*

Our approach: *Recursive Lexicographical Search*

- This is a systematic approach to "building" the set of all MPE from the MPE to simpler dynamic games we refer to as *stage games* of the original dynamic game $\mathcal{G}$.
- It is based on a generalization of backward induction that is applicable to a class of dynamic games that we call *directional dynamic games* (DDGs)
- A DDG is a game where a subset of the state variables evolve in a manner that satisfies certain conditions including an intuitive notion of *directionality.*
- When the state space is finite we can exploit this directionality and partition it into a finite number of elements we call *stages.*

Our approach: *Recursive Lexicographical Search*

- This is a systematic approach to "building" the set of all MPE from the MPE to simpler dynamic games we refer to as *stage games* of the original dynamic game $\mathcal{G}$.
- It is based on a generalization of backward induction that is applicable to a class of dynamic games that we call *directional dynamic games* (DDGs)
- A DDG is a game where a subset of the state variables evolve in a manner that satisfies certain conditions including an intuitive notion of *directionality.*
- When the state space is finite we can exploit this directionality and partition it into a finite number of elements we call *stages.*
- We can find the MPE of the game using a generalized version of backward induction over the stages of the game, a process we call *state recursion*

Examples of DDGs

- Any finite horizon game:

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess:

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases
- Bargaining over a stochastically shrinking pie:

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases
- Bargaining over a stochastically shrinking pie: the game is directional if the pie can only shrink

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases
- Bargaining over a stochastically shrinking pie: the game is directional if the pie can only shrink
- Patent races (Judd, Schmedders, Yeltekin, 2012)

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases
- Bargaining over a stochastically shrinking pie: the game is directional if the pie can only shrink
- Patent races (Judd, Schmedders, Yeltekin, 2012)
- Bertrand price competition with leapfrogging investments:

Examples of DDGs

- Any finite horizon game: time is always an obviously "directional variable"
- Chess: the number of chess pieces still on the board only decreases and never increases
- Bargaining over a stochastically shrinking pie: the game is directional if the pie can only shrink
- Patent races (Judd, Schmedders, Yeltekin, 2012)
- Bertrand price competition with leapfrogging investments: directionality comes from the fact that firms can "leapfrog" each other by investing in a state of the art production technology that steadily improves

State Recursion versus Backward Induction

- State recursion can be viewed as a generalization of the method of backward induction which is done using a *time variable* t where $t \to t+1$ with probability 1

State Recursion versus Backward Induction

- State recursion can be viewed as a generalization of the method of backward induction which is done using a *time variable* t where $t \to t+1$ with probability 1
- State recursion is backward induction over the *stages of the game* τ, and transitions between stages can be *stochastic.*

State Recursion versus Backward Induction

- State recursion can be viewed as a generalization of the method of backward induction which is done using a *time variable* t where $t \to t+1$ with probability 1
- State recursion is backward induction over the *stages of the game* τ, and transitions between stages can be *stochastic.* Thus τ can be thought of as a *stochastic time variable* which evolves unidirectionally, but is not restricted to satisfy the restriction that $\tau \to \tau + 1$ with probability 1

Relation to Subgame Perfection

- Kuhn (1953) and Selten (1965) showed that standard backward induction on the *game tree* (the extensive form representation of the game) can be used to compute all *subgame perfect equilibria* of finite games

Relation to Subgame Perfection

- Kuhn (1953) and Selten (1965) showed that standard backward induction on the *game tree* (the extensive form representation of the game) can be used to compute all *subgame perfect equilibria* of finite games
- In a DDG, we can represent the directionality of the game via a *directed acyclic graph* (DAG). While every game tree is a DAG, the DAGs that represent directionality in a dynamic game are *not game trees.*

Relation to Subgame Perfection

- Kuhn (1953) and Selten (1965) showed that standard backward induction on the *game tree* (the extensive form representation of the game) can be used to compute all *subgame perfect equilibria* of finite games
- In a DDG, we can represent the directionality of the game via a *directed acyclic graph* (DAG). While every game tree is a DAG, the DAGs that represent directionality in a dynamic game are *not game trees.* Instead, the DAG summarizes the directionality of the game in terms of the state space instead of the temporal ordering implied by the game tree.

Relation to Subgame Perfection

- Kuhn (1953) and Selten (1965) showed that standard backward induction on the *game tree* (the extensive form representation of the game) can be used to compute all *subgame perfect equilibria* of finite games
- In a DDG, we can represent the directionality of the game via a *directed acyclic graph* (DAG). While every game tree is a DAG, the DAGs that represent directionality in a dynamic game are *not game trees.* Instead, the DAG summarizes the directionality of the game in terms of the state space instead of the temporal ordering implied by the game tree.
- State recursion can be viewed as a generalization of standard backward induction, *but a type of backward induction that is performed on the DAG rather than on the game tree.*

Recursive Lexicographical Search (RLS)

- State recursion can be used to find a *single* MPE of a dynamic game $\mathcal{G}$

Recursive Lexicographical Search (RLS)

- State recursion can be used to find a *single* MPE of a dynamic game $\mathcal{G}$
- It depends on a specification of an *equilibrium selection rule* for selecting a particular MPE at each state of the game that constitute the *behavior strategies* used by the players

Recursive Lexicographical Search (RLS)

- State recursion can be used to find a *single* MPE of a dynamic game $\mathcal{G}$
- It depends on a specification of an *equilibrium selection rule* for selecting a particular MPE at each state of the game that constitute the *behavior strategies* used by the players
- *Recursive lexicographical search* (RLS) is an algorithm that repeatedly invokes state recursion in an efficient way to compute all MPE of the DDG by systematically cycling through all feasible equilibrium selection rules (ESRs) for each of the component stage games of the DDG.

Relation to Literature on Finitely Repeated Games

- The idea of how multiple equilibria of a stage game can be used to construct a much larger set of equilibria in the overall game was used by Benoit and Krishna (1985) to show that a version of the Folk Theorem can hold in *finitely* repeated games.

Relation to Literature on Finitely Repeated Games

- The idea of how multiple equilibria of a stage game can be used to construct a much larger set of equilibria in the overall game was used by Benoit and Krishna (1985) to show that a version of the Folk Theorem can hold in *finitely* repeated games.
- The prevailing view prior to their work was that the extreme multiplicity of equilibria implied by the Folk Theorem for infinitely repeated games cannot happen because backward induction from the last period results in a unique equilibrium in a finitely repeated game.

Relation to Literature on Finitely Repeated Games

- The idea of how multiple equilibria of a stage game can be used to construct a much larger set of equilibria in the overall game was used by Benoit and Krishna (1985) to show that a version of the Folk Theorem can hold in *finitely* repeated games.
- The prevailing view prior to their work was that the extreme multiplicity of equilibria implied by the Folk Theorem for infinitely repeated games cannot happen because backward induction from the last period results in a unique equilibrium in a finitely repeated game.
- Benoit and Krishna showed that multiplicity of equilibria in the stage games can be used to create a much larger set of subgame perfect equilibria in the finitely repeated game, so the Folk Theorem can emerge if the time horizon is sufficiently large.

Benoit and Krishna vs RLS

- However Benoit and Krishna did not propose an algorithm to enumerate all possible subgame perfect equilibria of a finitely repeated game, whereas the RLS algorithm we propose can find and enumerate all such equilibria.

Benoit and Krishna vs RLS

- However Benoit and Krishna did not propose an algorithm to enumerate all possible subgame perfect equilibria of a finitely repeated game, whereas the RLS algorithm we propose can find and enumerate all such equilibria.
- Though we do not claim that all dynamic games will have exploitable directional structure, we show there is a sense in which the RLS algorithm can approximate the set of all MPE for a wide class of finite and infinite-horizon dynamic games, even if there is no exploitable directionality in the game other than the passage of time.

Benoit and Krishna vs RLS

- However Benoit and Krishna did not propose an algorithm to enumerate all possible subgame perfect equilibria of a finitely repeated game, whereas the RLS algorithm we propose can find and enumerate all such equilibria.
- Though we do not claim that all dynamic games will have exploitable directional structure, we show there is a sense in which the RLS algorithm can approximate the set of all MPE for a wide class of finite and infinite-horizon dynamic games, even if there is no exploitable directionality in the game other than the passage of time.
- We show how backward induction *performed in the right way* can approximate all MPE of a fairly broad class of infinite horizon games. We view this as an analog of Benoit and Krishnas Folk Theorem approximation result for finitely repeated games.

Road Map for the Talk

1. Define the concept of a *directional dynamic game*

Road Map for the Talk

1. Define the concept of a *directional dynamic game*
2. Introduce the *state recursion algorithm* and show that it can compute *a single MPE* of the game $\mathcal{G}$

Road Map for the Talk

1. Define the concept of a *directional dynamic game*
2. Introduce the *state recursion algorithm* and show that it can compute *a single MPE* of the game $\mathcal{G}$
3. Introduce the *recursive lexicographical search algorithm* and show that is can find *all MPE* of $\mathcal{G}$

Road Map for the Talk

1. Define the concept of a *directional dynamic game*
2. Introduce the *state recursion algorithm* and show that it can compute *a single MPE* of the game $\mathcal{G}$
3. Introduce the *recursive lexicographical search algorithm* and show that is can find *all MPE* of $\mathcal{G}$
4. Illustrate how RLS works by using it to find all MPE of a dynamic model of Bertrand price competition with leapfrogging investments.

Road map of the talk

1. Original application: collusion of Australian corrugated fibre packaging (CFP) producers
   - Collusion between Amcor and Visy
   - Bertrand pricing and investment game
   - Solution concept: Markov perfect equilibrium (MPE)
2. State recursion algorithm
3. Recursive lexicographical search (RLS) algorithm
4. Solving the Bertrand investment game using RLS
5. Embedding full solution into the nested ML estimator

Amcor-Visy collusion case

- Australian market for cardboard is essentially a duopoly
- Between 2000 and 2005 *Visy* and *Amcor* colluded to divide the market of cardboard and to fix prices
- 2007: Visy admits to have been manipulating the market, issued with $36 million fine
- July 2009: Cadbury vs. Amcor, damages estimated at $235.8 million, settles out of court
- March 2011: Class action suit against both Amcor and Visy settles out of court for $95 million

Dynamic Bertrand price competition

Stochastic dynamic game

- Two Bertrand competitors, $n = 2$, no entry or exit
- Discrete time, infinite horizon ($t = 1, 2, \ldots, \infty$)
- Each firm maximizes expected discounted profits, common discount factor $\beta \in (0, 1)$
- Each firm has two choices in each period:
  1. Price for the product
  2. Whether or not to buy the state of the art technology

Static Bertrand price competition in each period

- Continuum of consumers make static purchase decision
- No switching costs: buy from the lower price supplier

Cost-reducing investments

State-of-the-art production cost c process

- Initial value c_0, lowest value 0: $0 \leq c \leq c_0$
- Discretized with n points
- Follows exogenous Markov process and only improves
- Markov transition probability $\pi(c_{t+1}|c_t)$
  $\pi(c_{t+1}|c_t) = 0$ if $c_{t+1} > c_t$

Investment choice: dichotomous

- Investment cost of $K(c)$ to obtain marginal cost c
- One period construction time: production with technology obtained at t starts at $t+1$

State space and information structure

Common knowledge

- State of the game: cost structure (c_1, c_2, c)
- State space is $S = (c_1, c_2, c) \subset R^3$: $c_1 \geq c$, $c_2 \geq c$
- Actions are observable

Private information

- In each period each firm incurs additive costs (benefits) from not investing and investing $\eta\epsilon_{i,I}$ and $\eta\epsilon_{i,N}$
- $\epsilon_{i,I}$ and $\epsilon_{i,N}$ are extreme value distributed, independent across choice, time and firms
- $\eta \geq 0$ is a scaling parameter
- Investment choice probabilities have logit form for $\eta > 0$

Timing of moves

Pricing decisions are made simultaneously

Expected one period profit of firm i from Bertrand game ($j \neq i$)

$$r_i(c_1, c_2) = \begin{cases} 0 & \text{if } c_i \geq c_j \\ c_j - c_i & \text{if } c_i < c_j \end{cases}$$

Two versions regarding investment decisions

1. Simultaneous moves:
   - Investment decisions are made simultaneously
2. Alternating moves:
   - The "right to move" state variable $m \in \{1, 2\}$,
   - When $m = i$, only firm i can make a cost reducing investment
   - m follows an own Markov process (deterministic alternation as a special case).

Actions and behavior strategies

Two choices in each period

- $p_i(c_1, c_2, c) = \max(c_1, c_2)$ – Bertrand pricing decision
- $P_i^I(c_1, c_2, c)$ – probability of firm i to invest in state-of-the-art production technology

  $P_i^N(c_1, c_2, c) = 1 - P_i^I(c_1, c_2, c)$ – probability not to invest

Strategy profile

- $\sigma = (\sigma_1, \sigma_2)$ – pair of Markovian *behavior* strategies

  $\sigma_i = \Big(p_i(c_1, c_2, c), P_i^I(c_1, c_2, c)\Big) \in \mathbb{R}_+ \times [0, 1]$
- Pure strategies included as special case

Definition of Markov Perfect Equilibium

Definition (Markov perfect equilibrium (MPE))

MPE of Bertrand investment stochastic game is a pair of

- strategy profile $\sigma^* = (\sigma_1^*, \sigma_2^*)$, and
- pair of *value functions* $V(s) = \Big(V_1(s), V_2(s)\Big)$, $V_i : S \to R$,

such that

1. Bellman equations (below) are satisfied for each firm, and
2. strategies σ_1^* and σ_2^* constitute mutual best responses, and assign positive probabilities only to the actions in the set of maximizers of the Bellman equations.

Bellman equations, firm $i = 1$, simultaneous moves

$$V_i(c_1, c_2, c) = \max\left[v_i^I(c_1, c_2, c) + \eta\epsilon_{i,I}, v_i^N(c_1, c_2, c) + \eta\epsilon_{i,N}\right]$$

$$\begin{aligned} v_i^N(c_1, c_2, c) &= r^i(c_1, c_2) + \beta EV_i(c_1, c_2, c|N) \\ v_i^I(c_1, c_2, c) &= r^i(c_1, c_2) - K(c) + \beta EV_i(c_1, c_2, c|I) \end{aligned}$$

With extreme value shocks, the investment probability is

$$P_i^I(c_1, c_2, c) = \frac{\exp\{v_I^I(c_1, c_2, c)/\eta\}}{\exp\{v_i^I(c_1, c_2, c)/\eta\} + \exp\{v_i^N(c_1, c_2, c)/\eta\}}$$

Bellman equations, firm $i = 1$, simultaneous moves

The expected values are given by

$$EV_i(c_1, c_2, c|N) = \int_0^c \left[P_j^I(c_1, c_2, c)H_i(c_1, c, c') + P_j^N(c_1, c_2, c)H_i(c_1, c_2, c')\right] \pi(dc'|c)$$

$$EV_i(c_1, c_2, c|I) = \int_0^c \left[P_j^I(c_1, c_2, c)H_i(c, c, c') + P_j^N(c_1, c_2, c)H_i(c, c_2, c')\right] \pi(dc'|c)$$

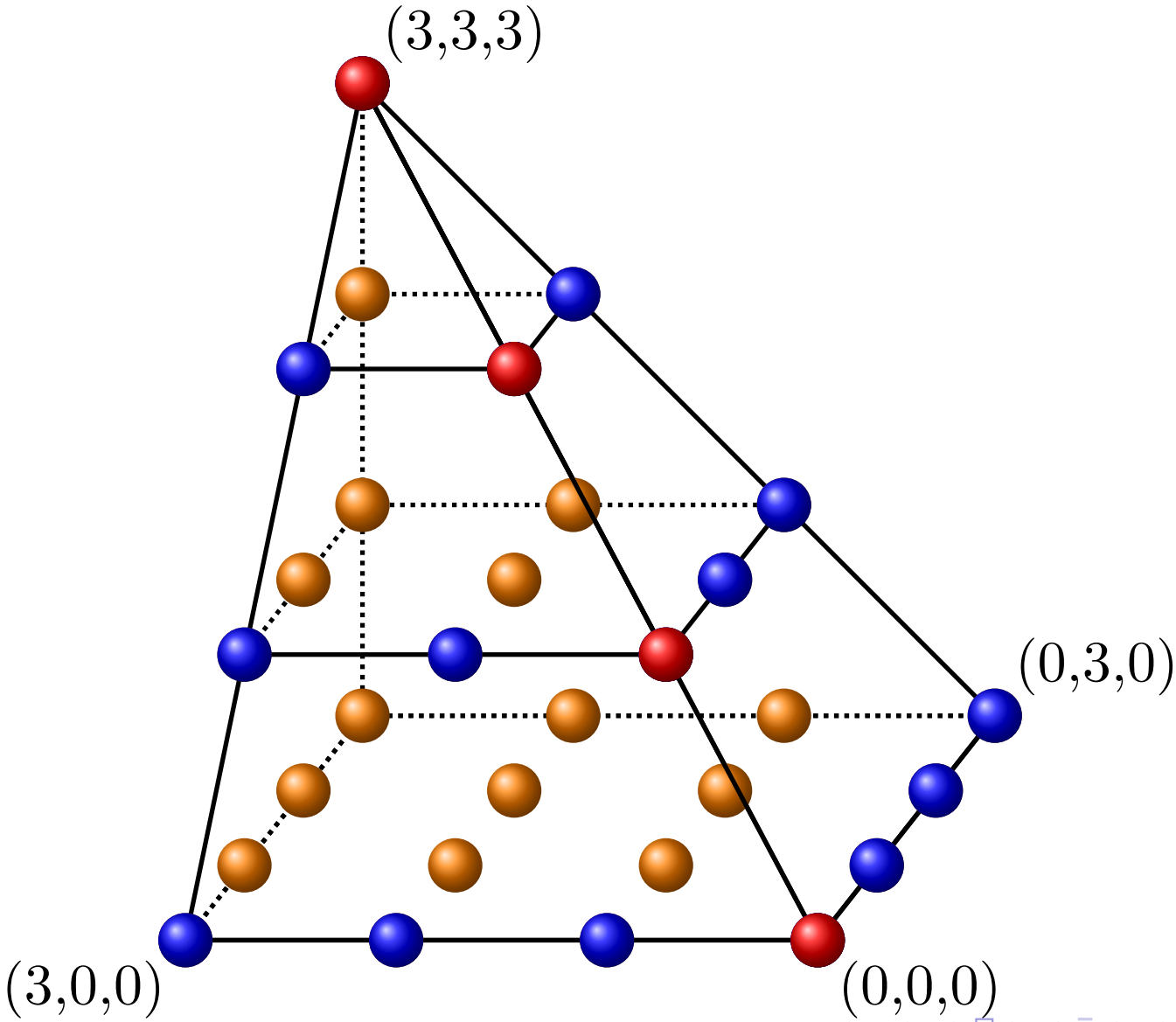
where

$H_i(c_1, c_2, c) = \eta \log\left[\exp\left(v_i^N(c_1, c_2, c)/\eta\right) + \exp\left(v_i^I(c_1, c_2, c)/\eta\right)\right]$

is the "smoothed max" or logsum function

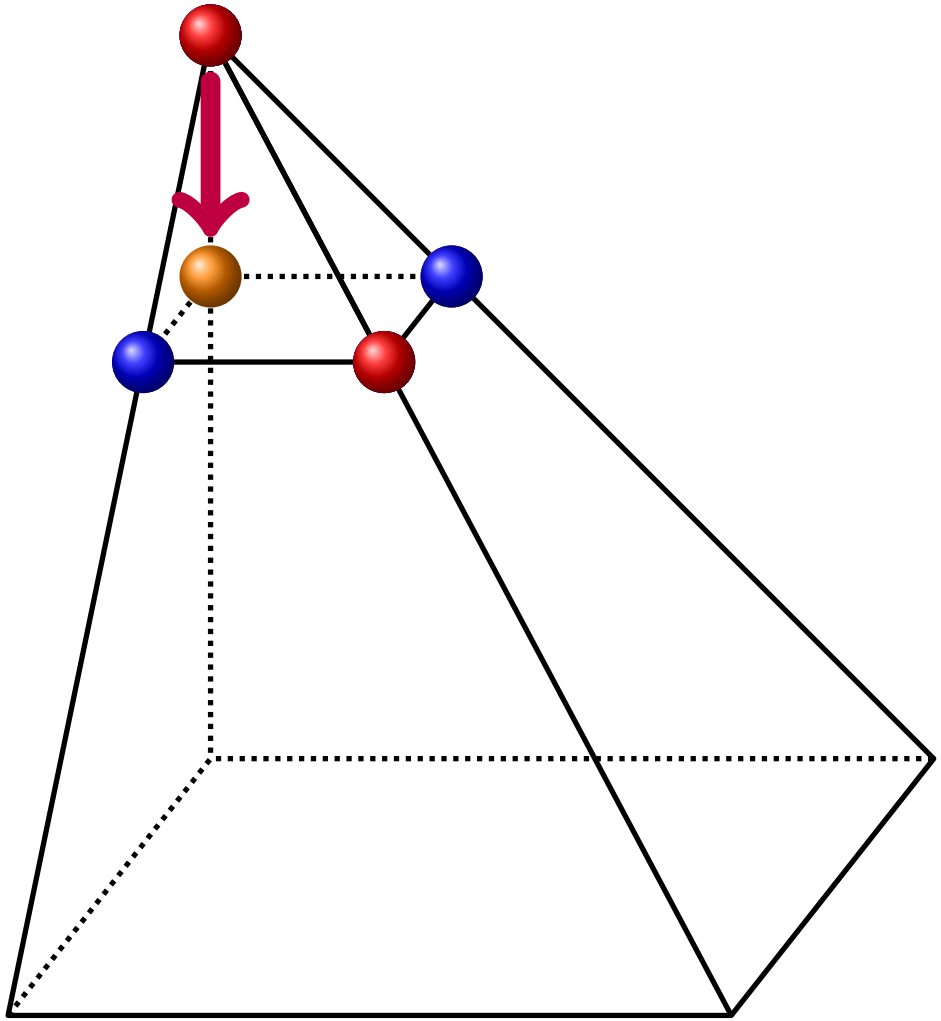
Discretized state space = a "quarter pyramid"

$S = \{(c_1, c_2, c) | c_1 \geq c,\ c_2 \geq c, c \in [0, \overline{c}]\},\ N = 4$



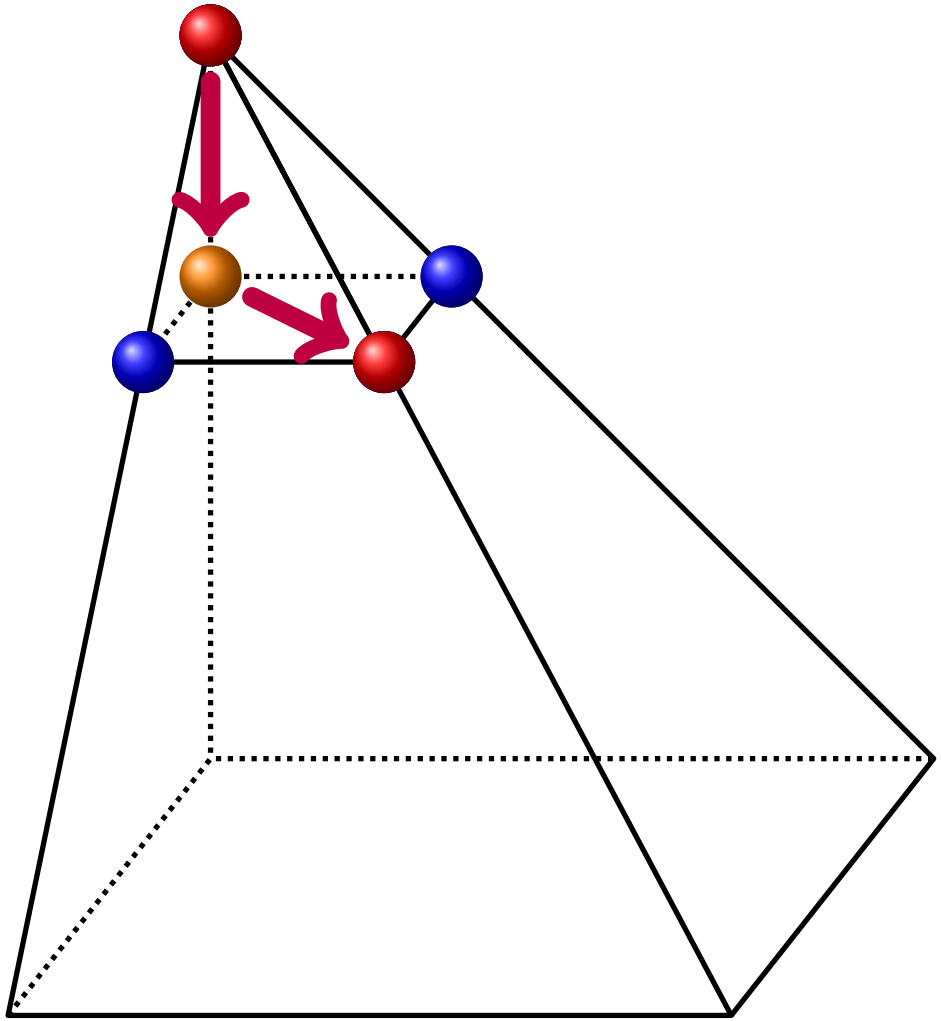
Game dynamics: example

The game starts at the apex, as some point technology improves



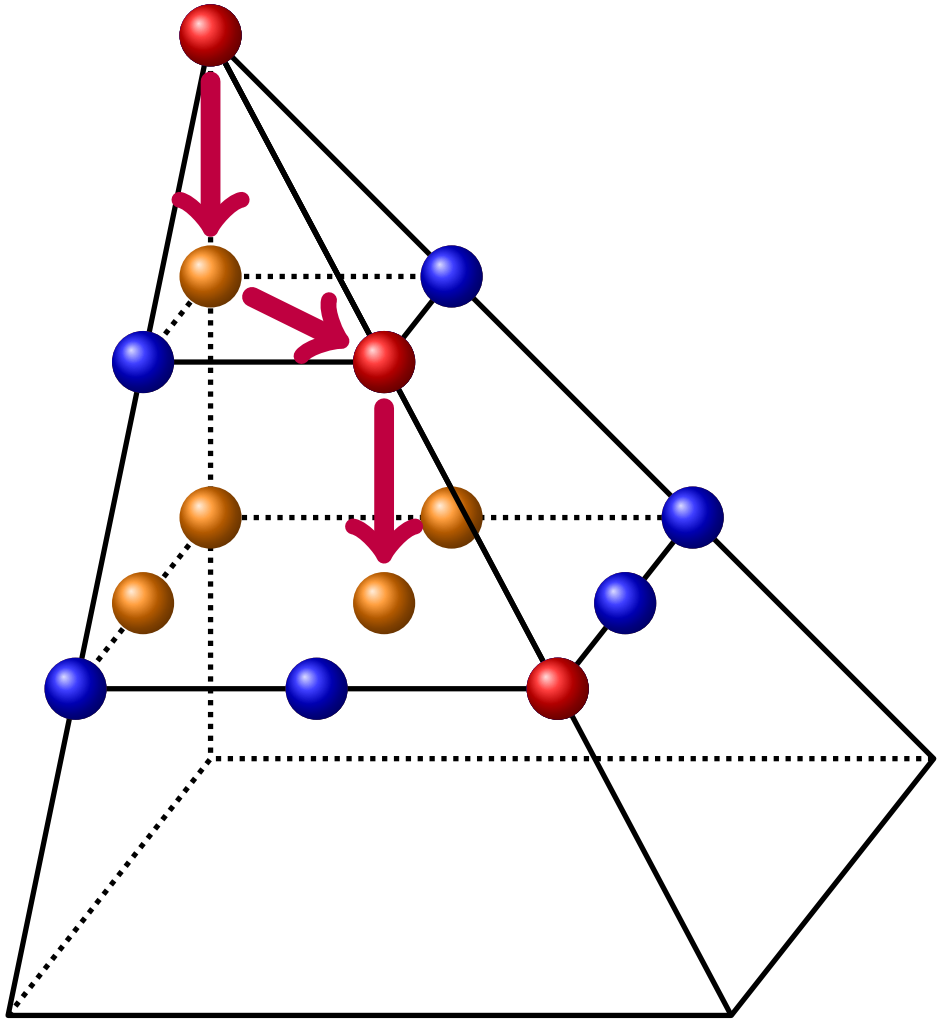
Game dynamics: example

Both firms buy new technology $c = 2 \rightsquigarrow (c_1, c_2, c) = (2, 2, 2)$



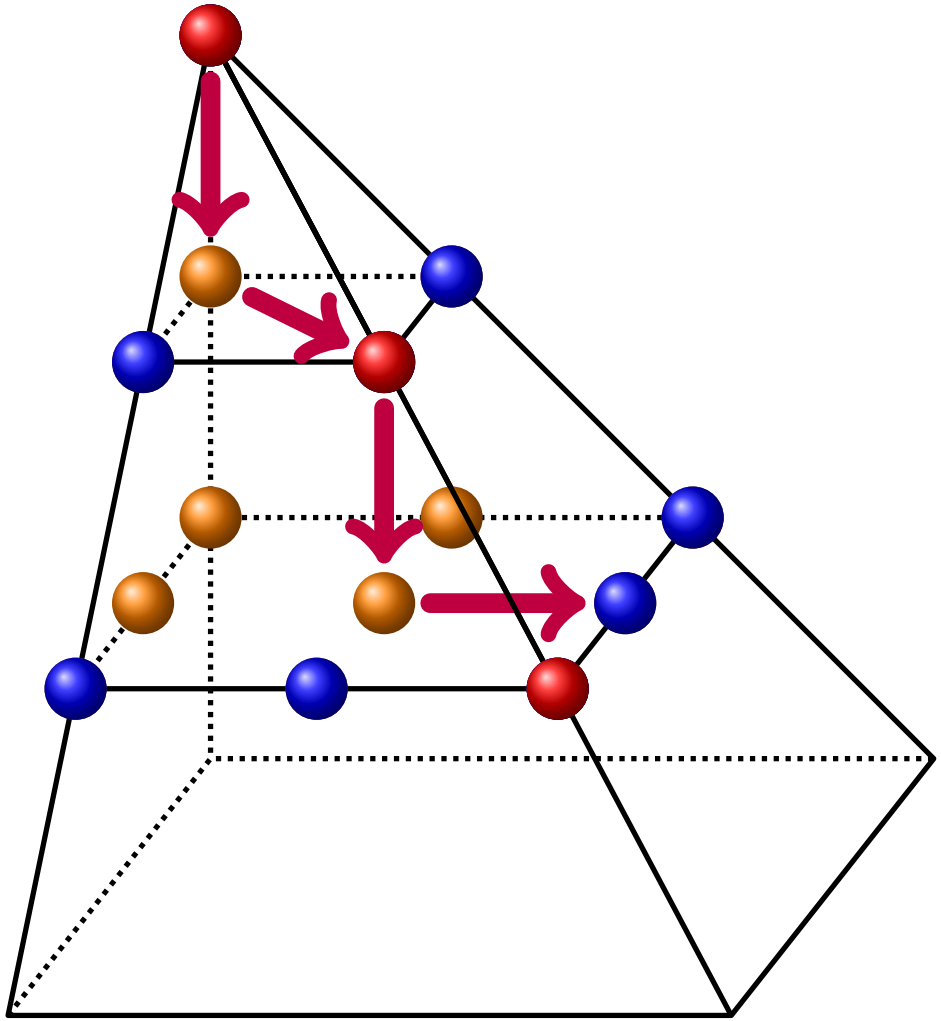
Game dynamics: example

State-of-the-art technology becomes $c = 1 \rightsquigarrow (c_1, c_2, c) = (2, 2, 1)$



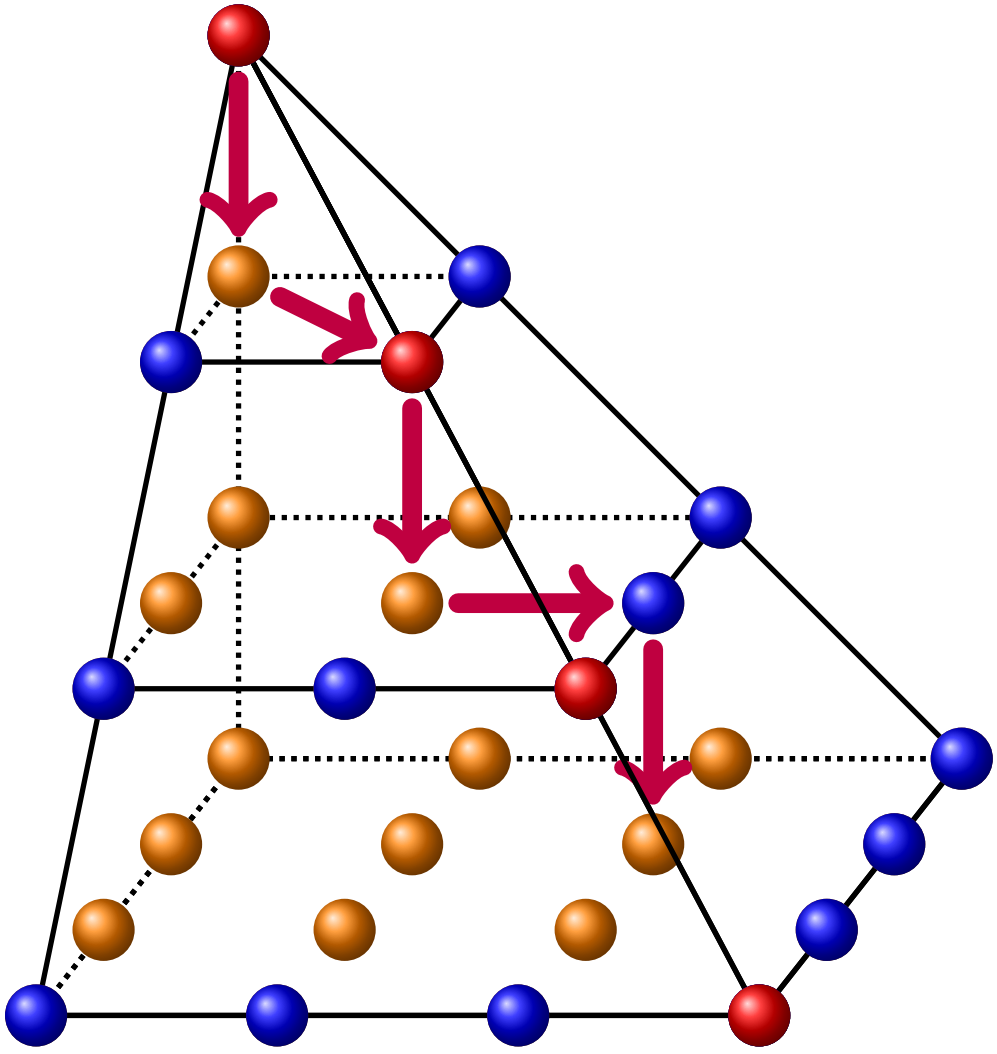
Game dynamics: example

Firm 1 invests and becomes cost leader $\rightsquigarrow (c_1, c_2, c) = (1, 2, 1)$



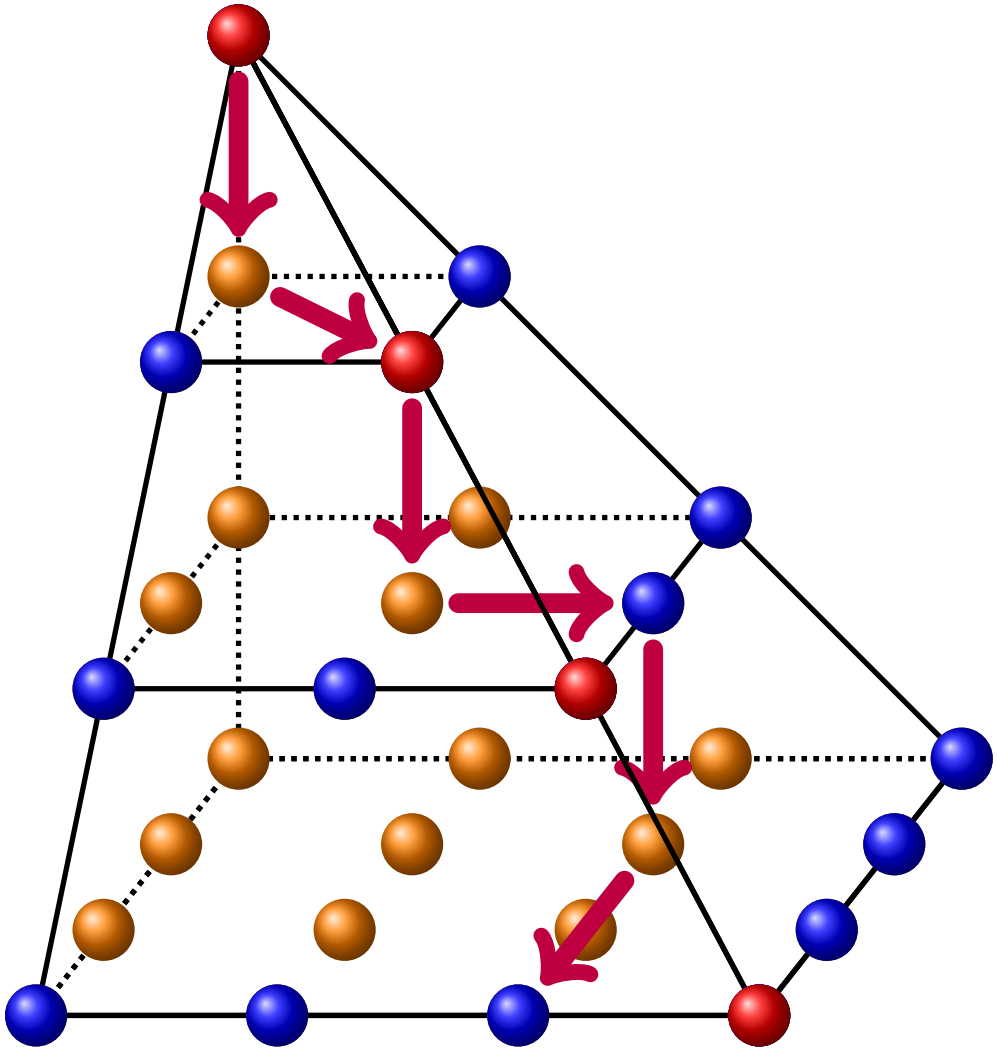
Game dynamics: example

State-of-the-art technology becomes $c = 0 \rightsquigarrow (c_1, c_2, c) = (1, 2, 0)$



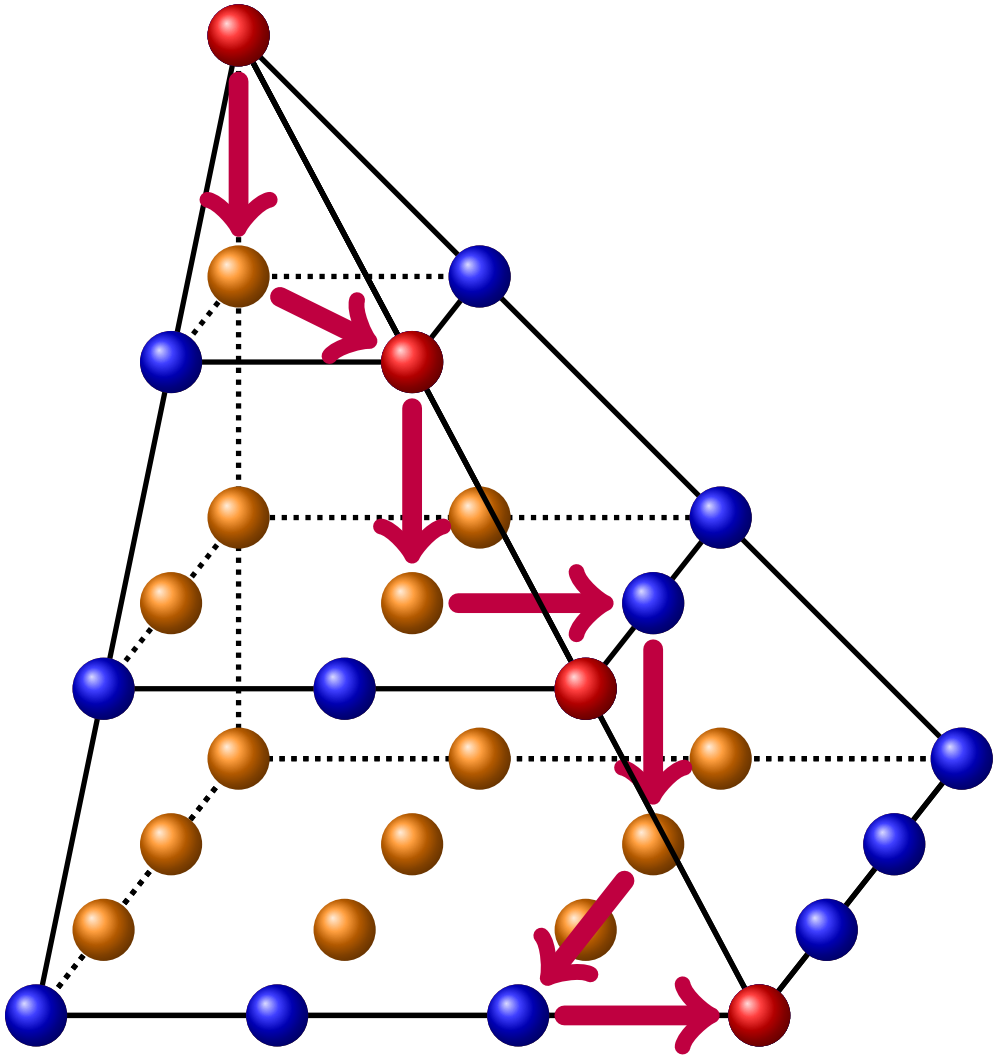
Game dynamics: example

Firm 2 leapfrogs firm 1 to become new cost leader $\rightsquigarrow (c_1, c_2, c) = (1, 0, 0)$



Game dynamics: example

Firm 1 invests, and the game reaches terminal state $\rightsquigarrow (c_1, c_2, c) = (0, 0, 0)$

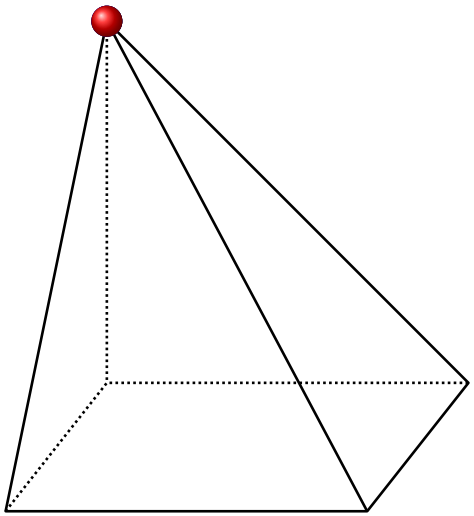


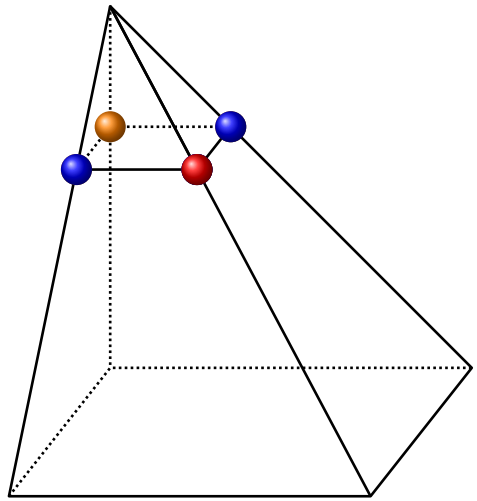
Road map of the talk

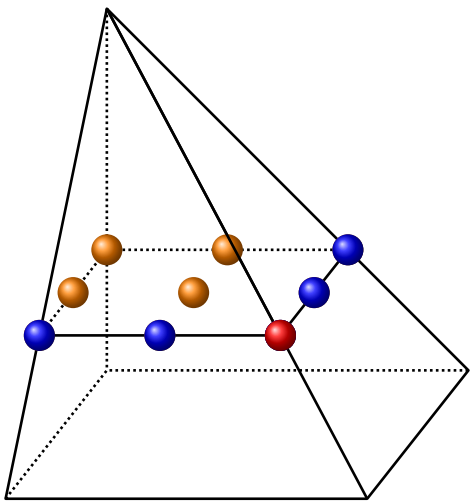
1. Original application: collusion of Australian corrugated fibre packaging (CFP) producers
2. State recursion algorithm
   - Formalization of the notion of directionality
   - Definition of directional dynamic games (DDG)
   - Directed acyclic graph (DAG)
   - Stage games and continuation strategies
   - Equilibrium selection and state recursion
3. Recursive lexicographical search (RLS) algorithm
4. Solving the Bertrand investment game using RLS
5. Embedding full solution into the nested ML estimator

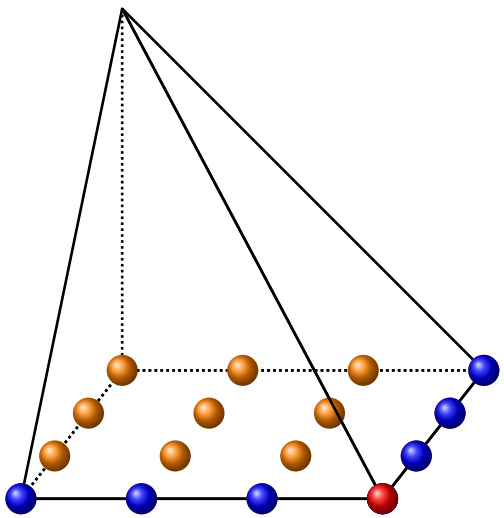
Transitions due to technological progress

As c decreases, the game falls through the layers of the pyramid



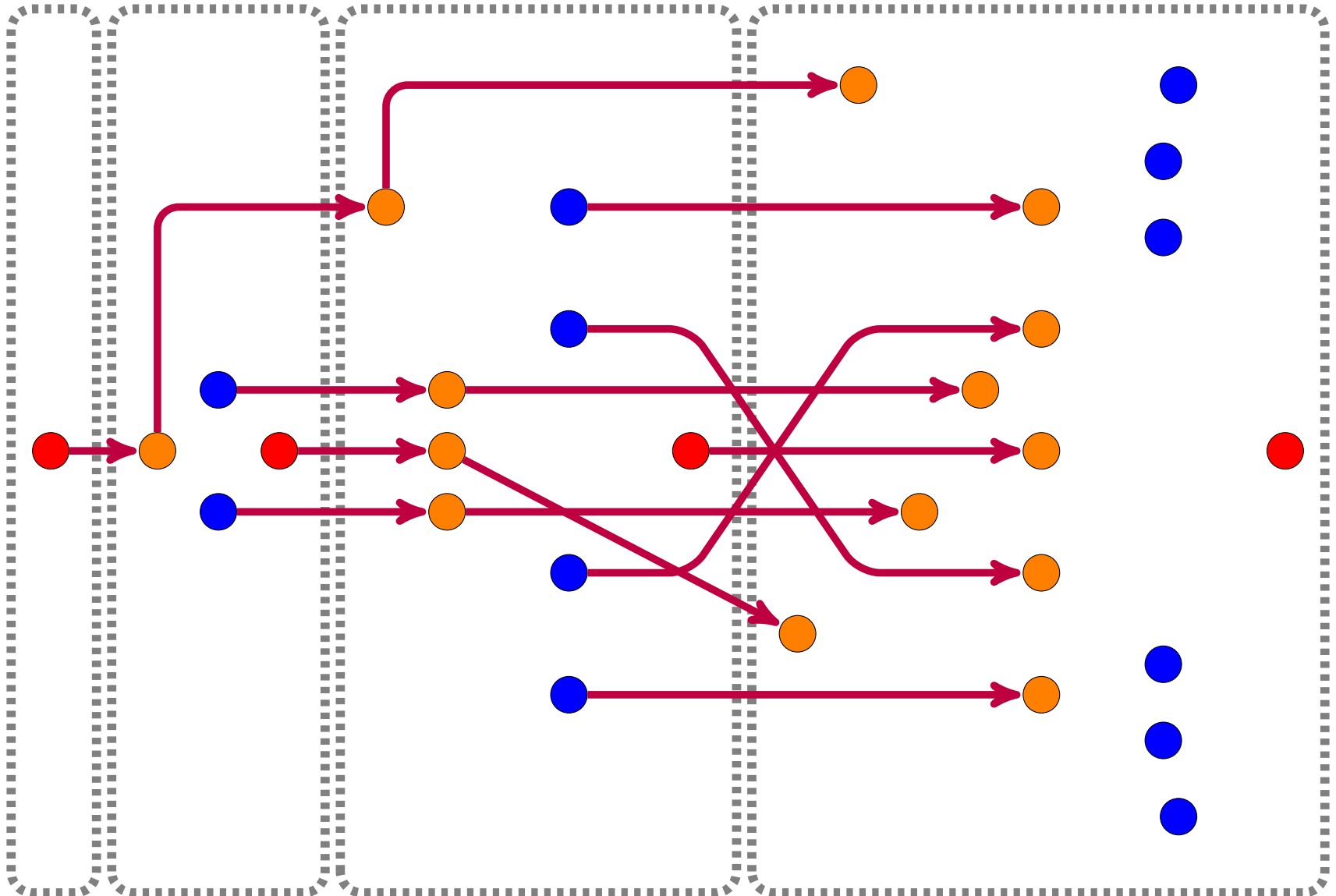






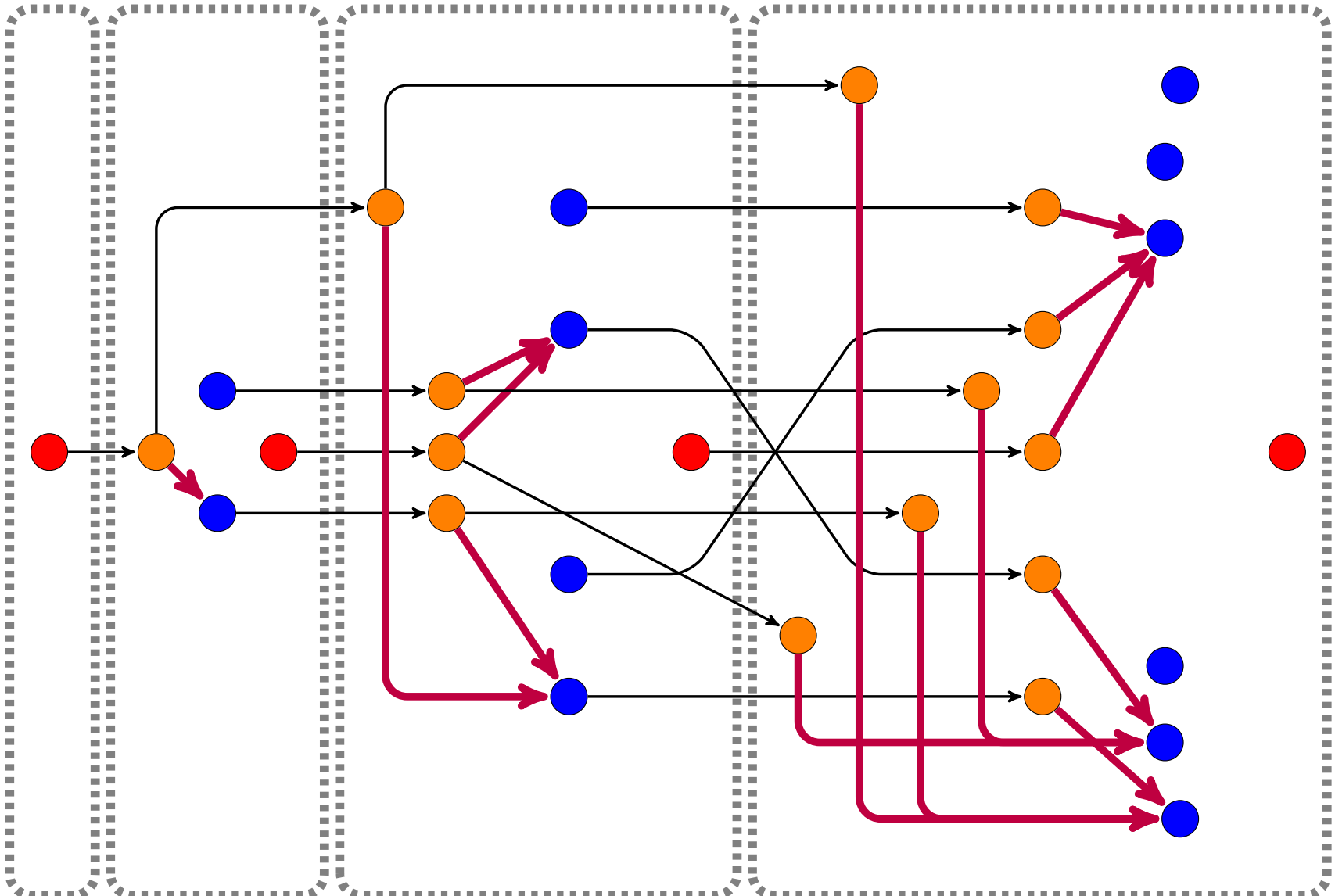
Transitions due to technological progress

As c decreases, the game falls through the layers of the pyramid



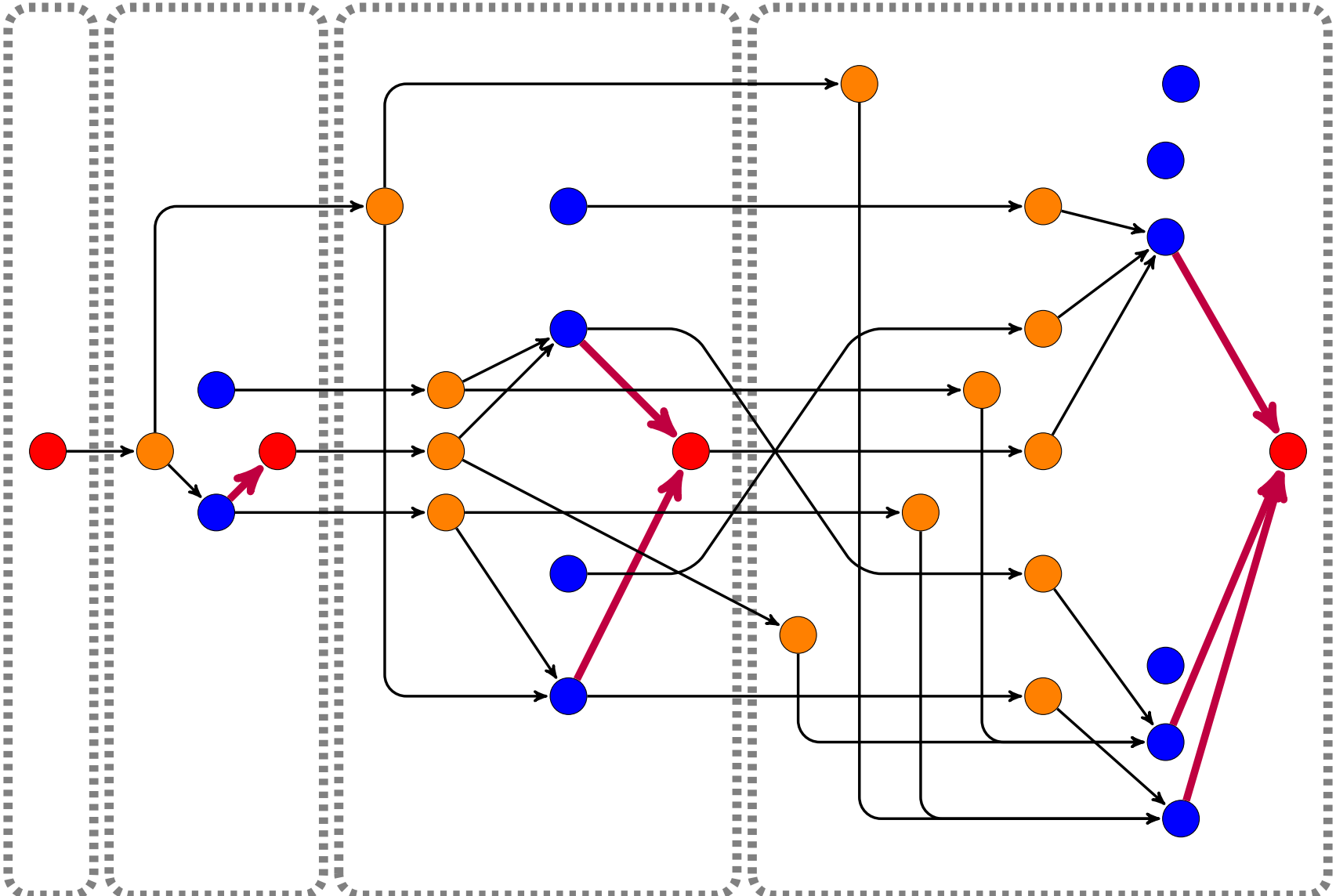
Strategy-specific partial order on S

Strategy σ_1 of firm 1: invest at all interior points



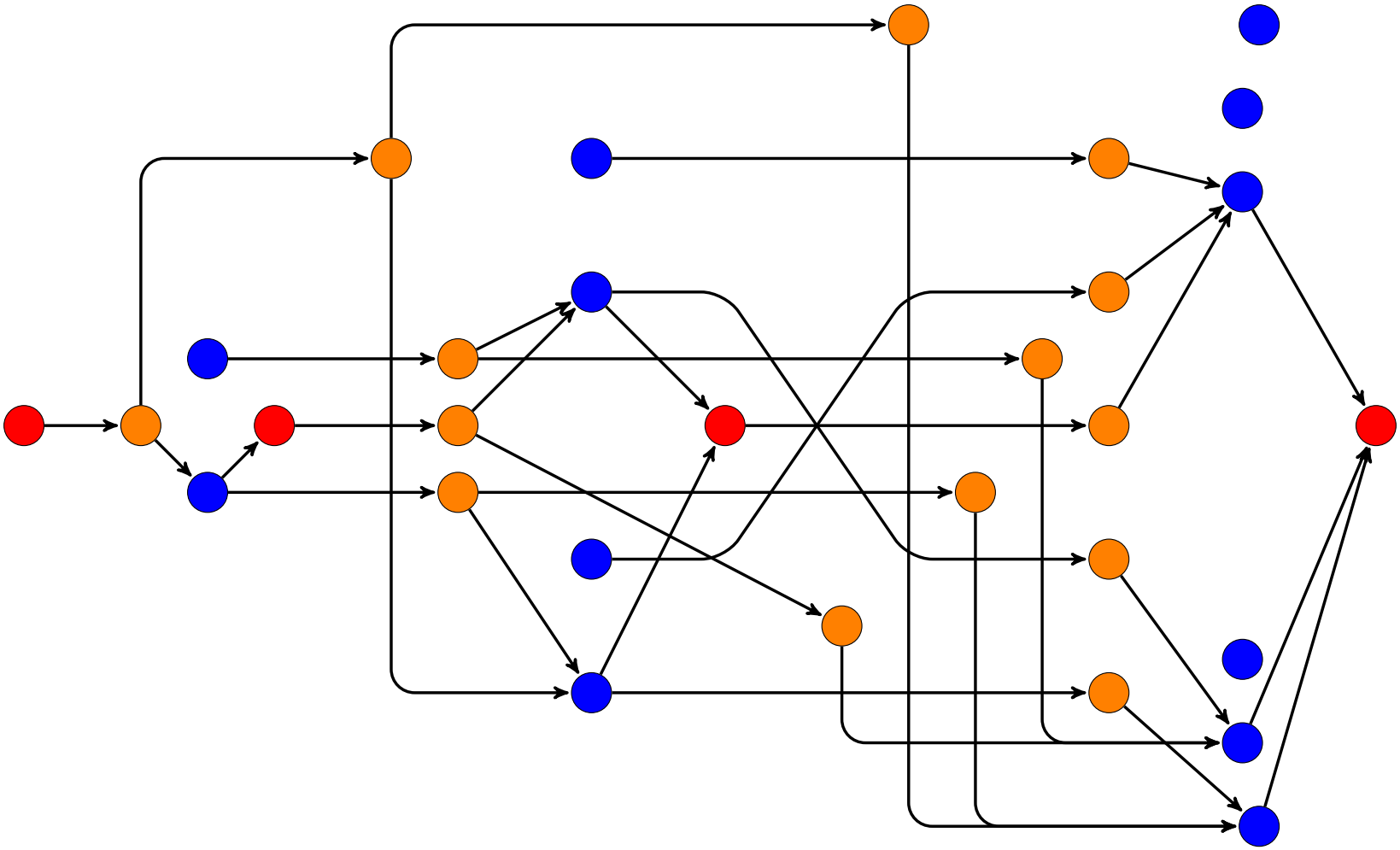
Strategy-specific partial order on S

Strategy σ_2 of firm 2: invest at all edge points



Strategy-specific partial order on S

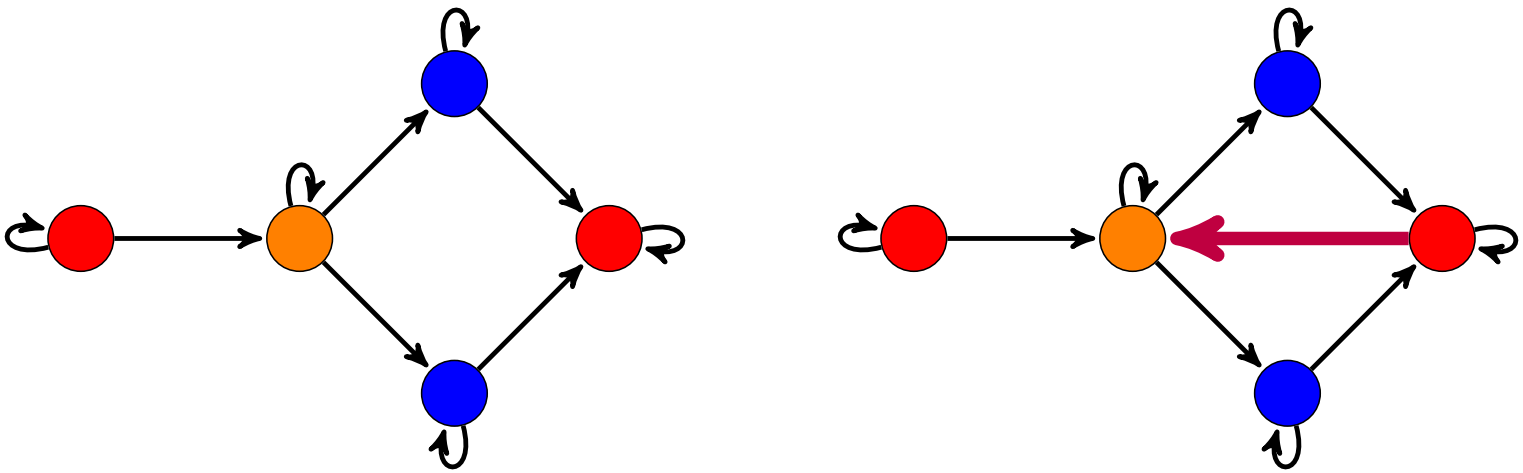
Strategy $\sigma = (\sigma_1, \sigma_2)$ of both firms



No loop (anti-cycling) condition

Hypothetical strategy profile inducing cycles

Self-loops appear when the game remains in the same state for two or more consecutive periods of time

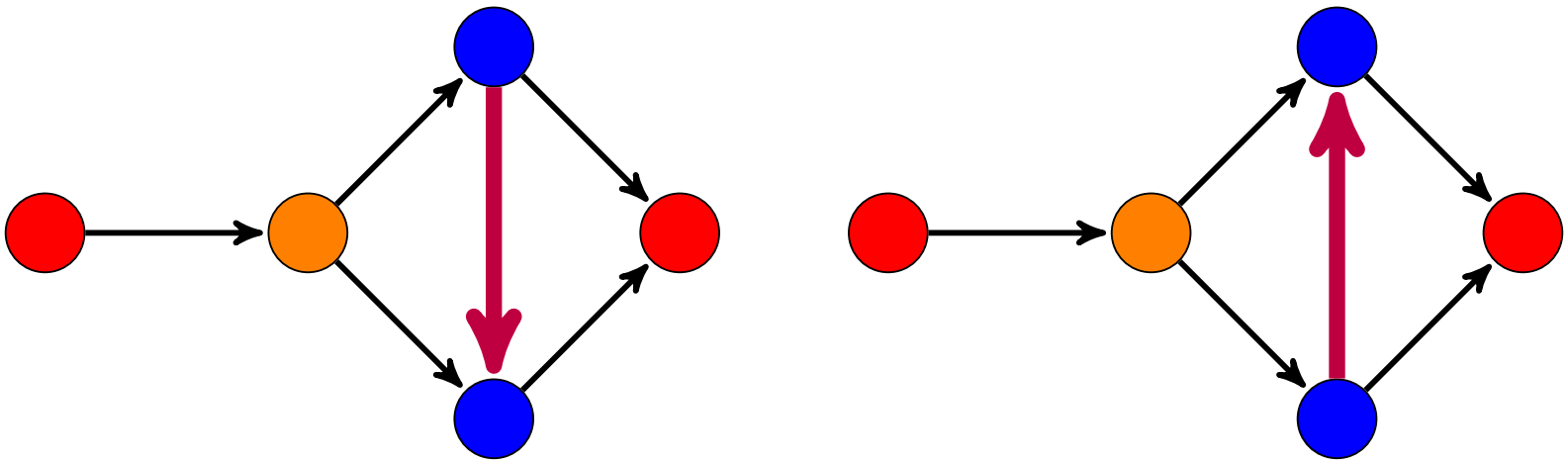


But loops between different states are not allowed

Consistency of strategy specific partial orders

Two hypothetical inconsistent strategies

Two strategies that induce opposite transitions are inconsistent



Note that in both cases the no-loop condition is satisfied

Definition of the Dynamic Directional Games

Definition (Dynamic Directional Games, DDG)

Finite state Markovian stochastic game is a DDG if it holds:

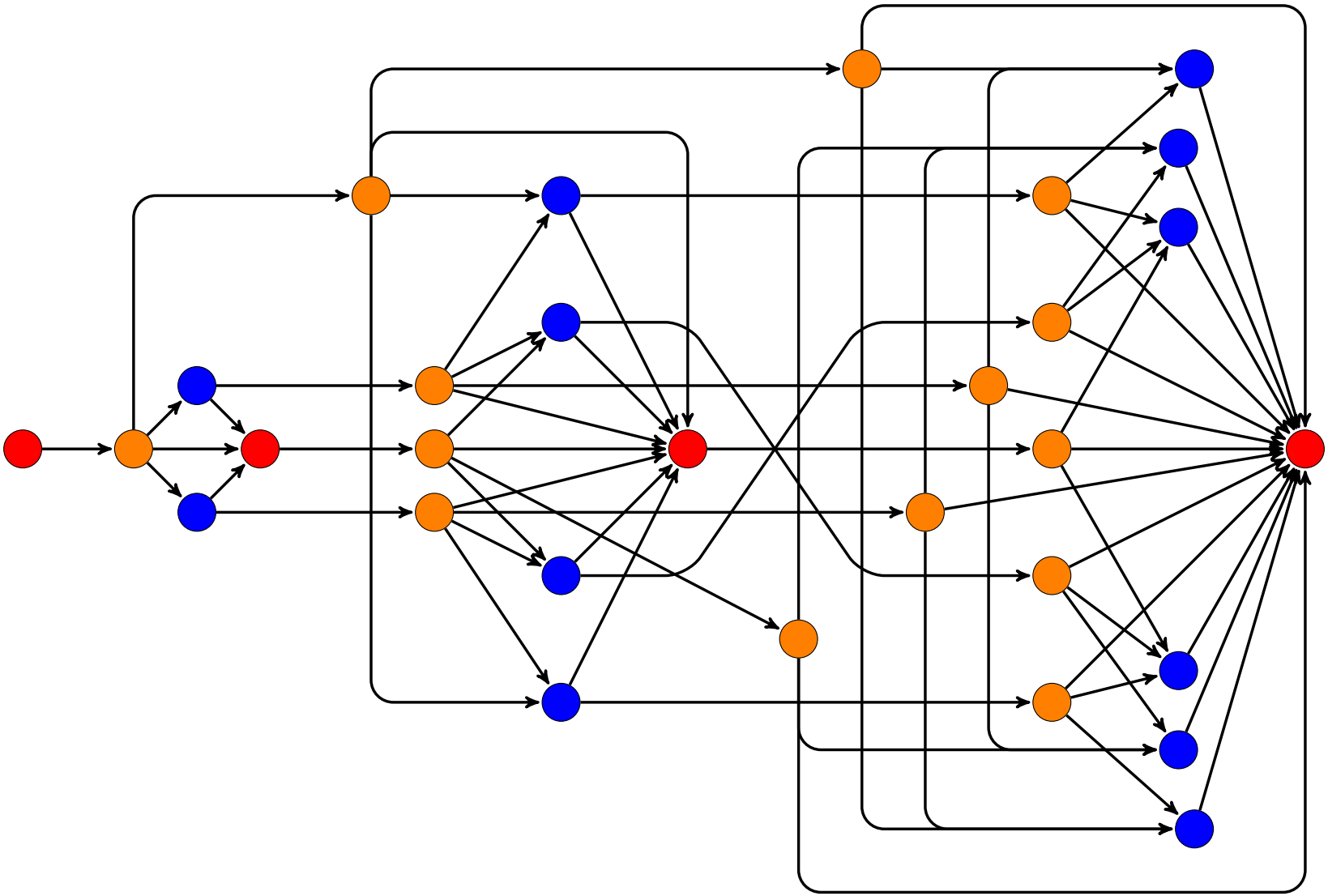
1. Every feasible Markovian strategy σ satisfies the no loop condition.
2. Every pair of feasible Markovian strategies σ and σ' induce consistent partial orders on the state space.

Next: Stages, stage games and state recursion

1. Strategy independent partial order over S
2. DAG to represent the directionality of the game
3. Recursion on the game DAG to form partition of totally ordered subsets of S
4. Stages on the state space and induced subgames of DDG
5. Continuation strategies and stage games

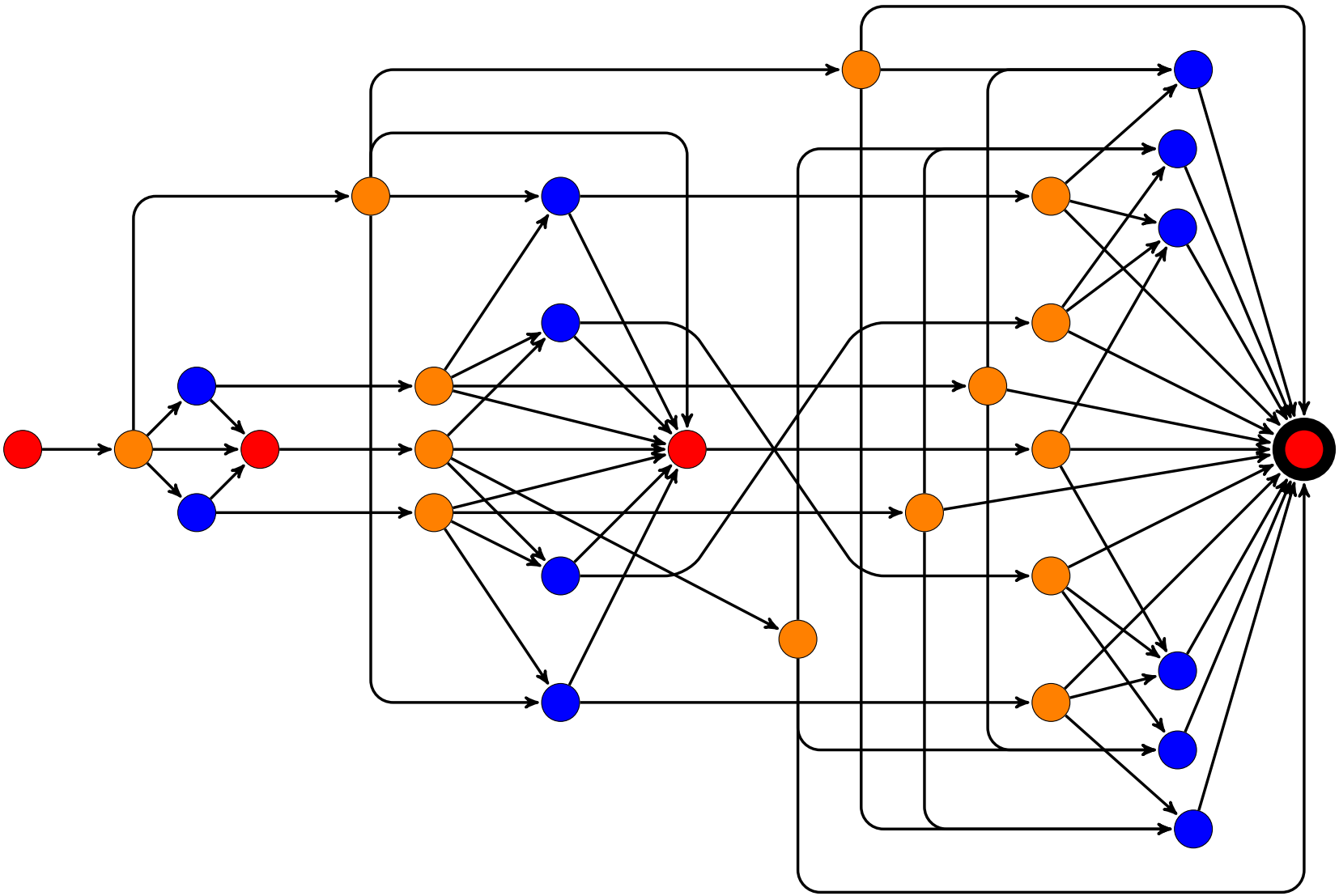
Strategy independent partial order on S

Coarsest common refinement of partial orders induced by all strategies



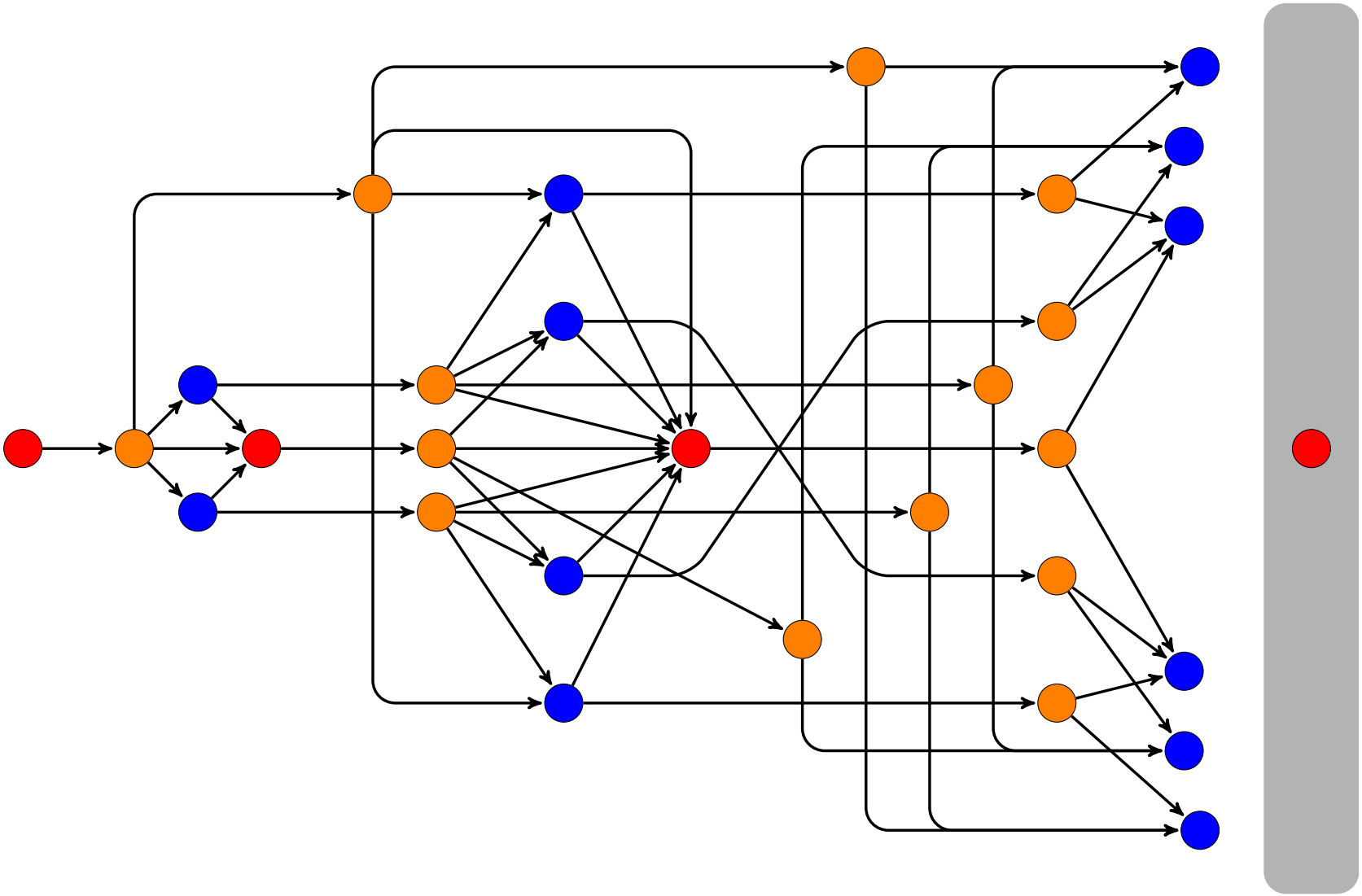
DAG recursion to partition S into stages

Identify terminal states



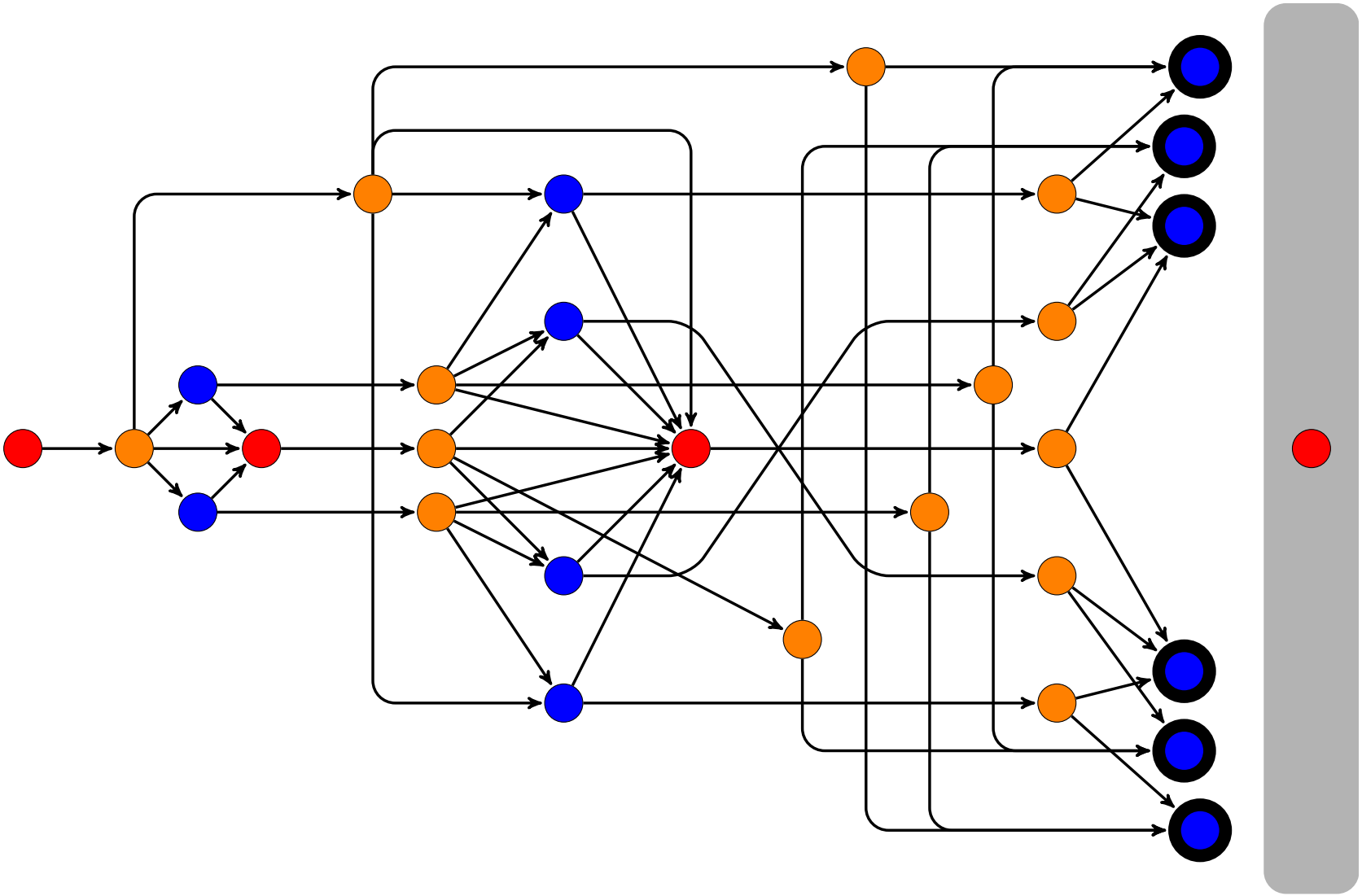
DAG recursion to partition S into stages

Remove terminal states



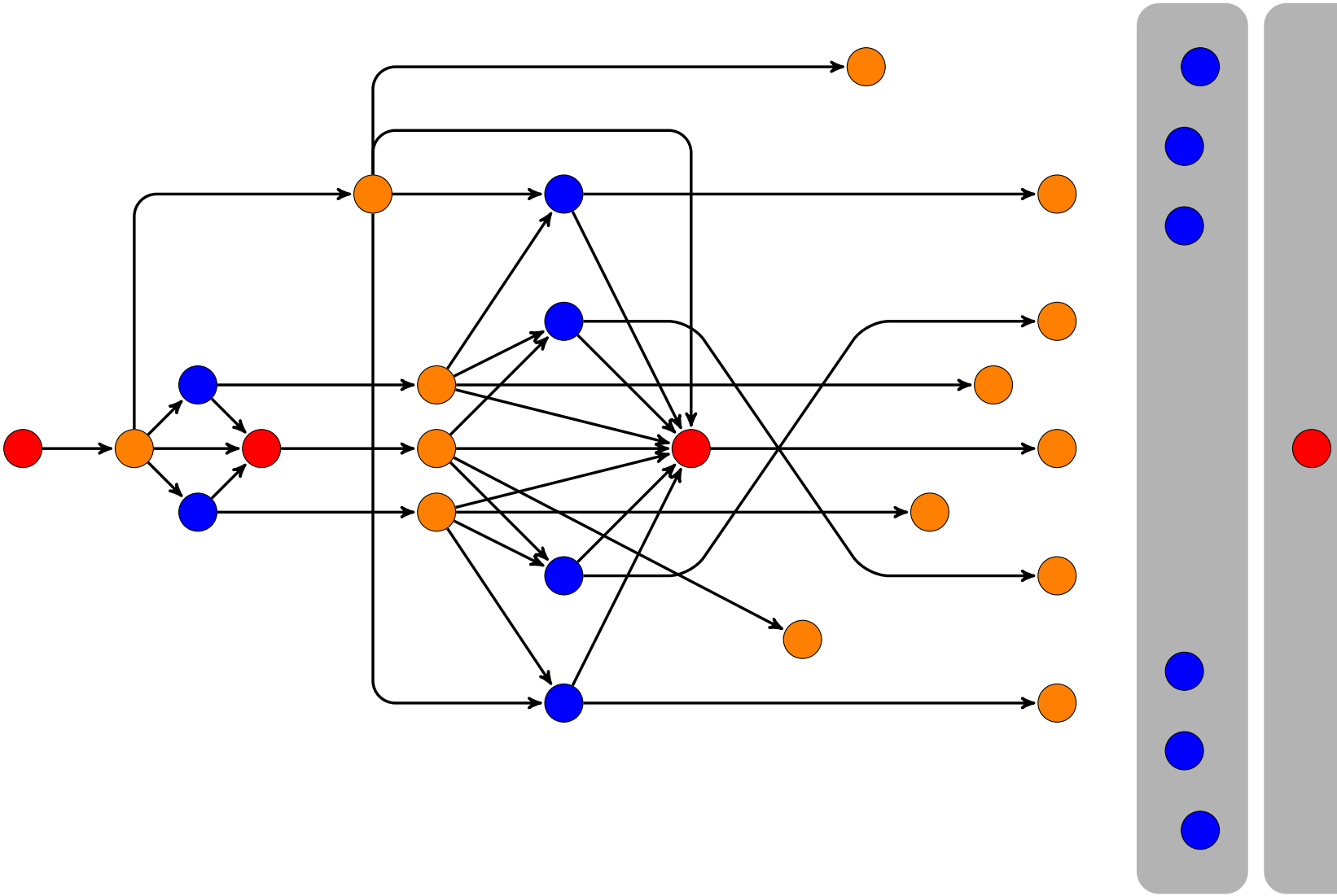
DAG recursion to partition S into stages

Identify terminal states



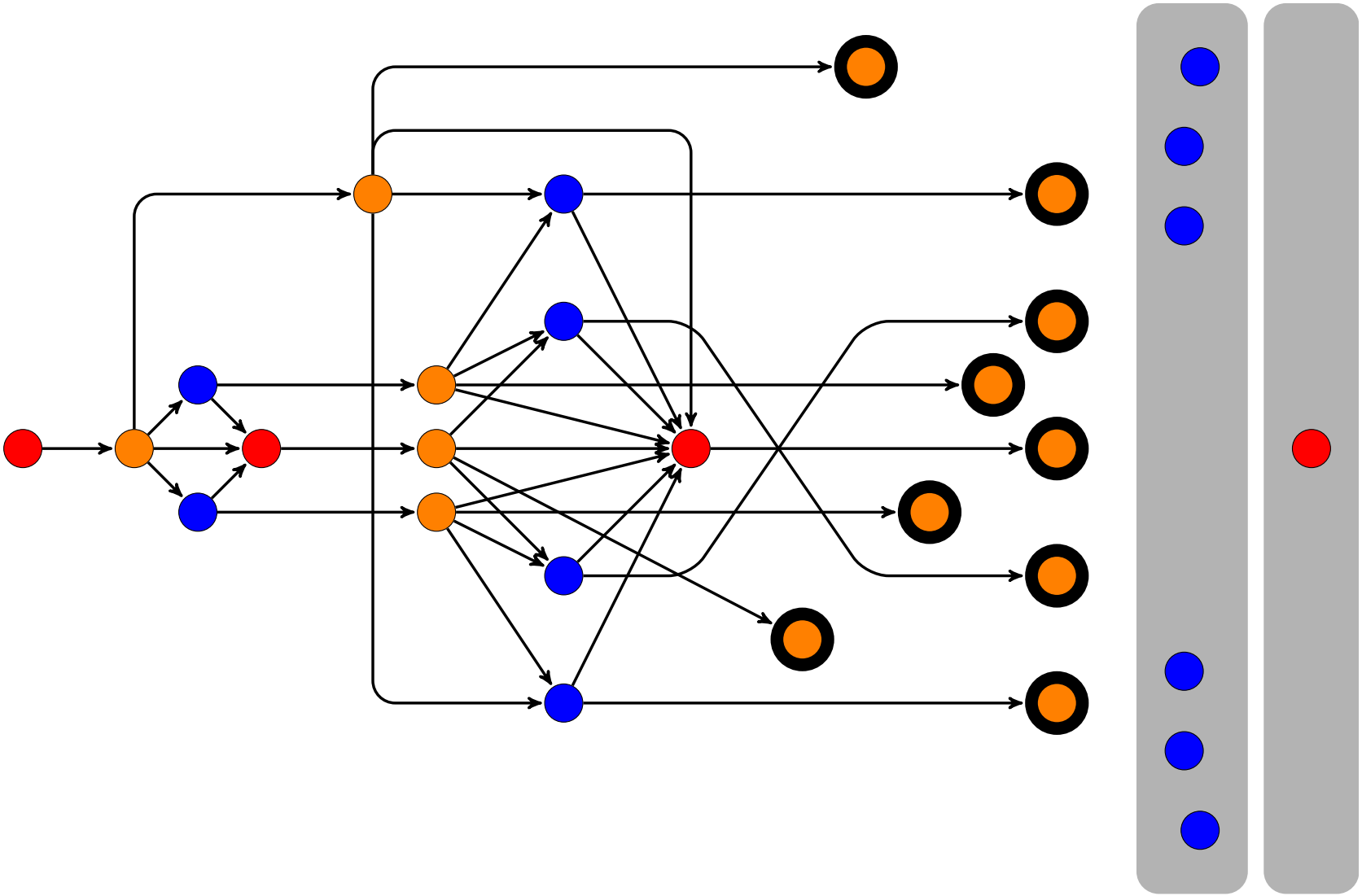
DAG recursion to partition S into stages

Remove terminal states



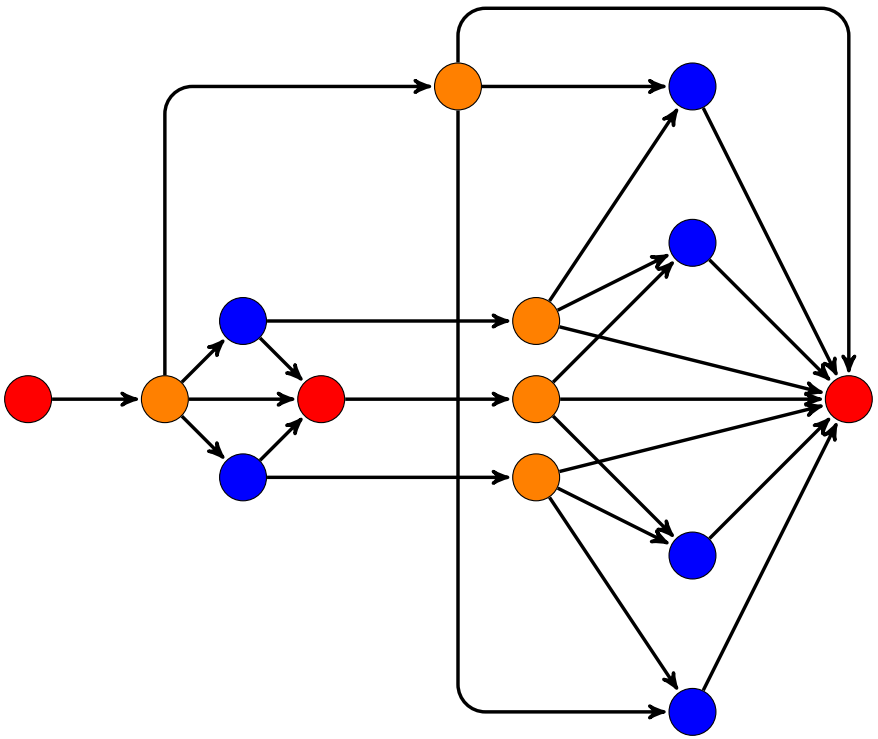
DAG recursion to partition S into stages

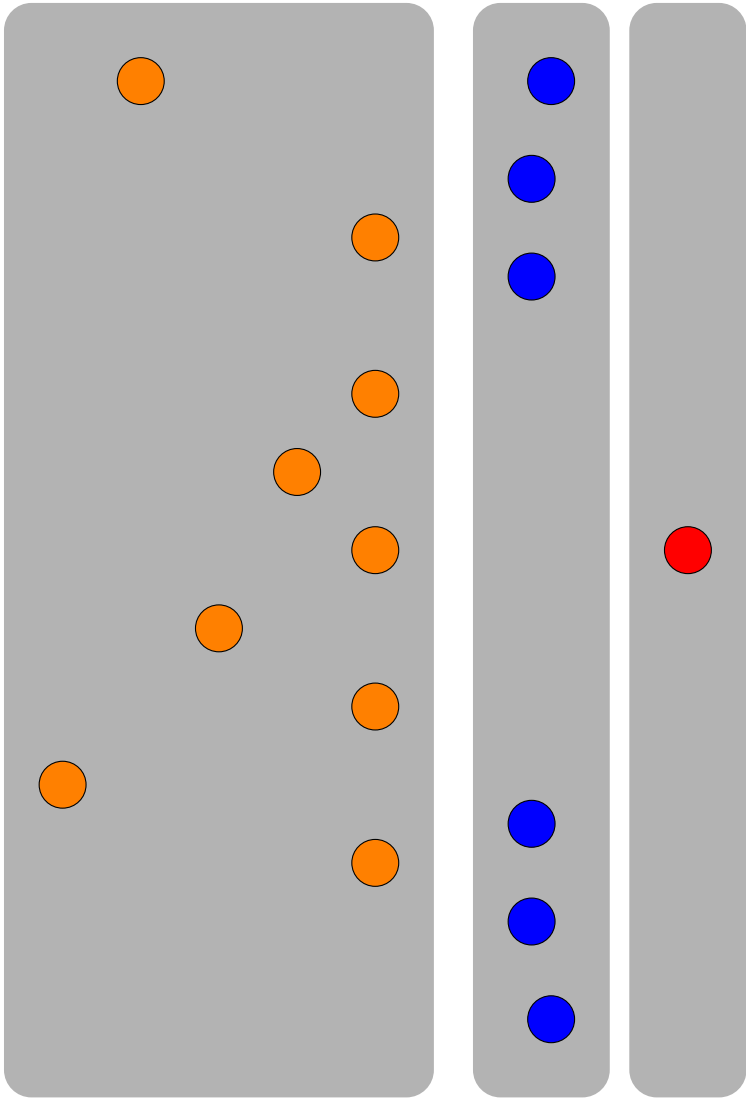
Identify terminal states



DAG recursion to partition S into stages

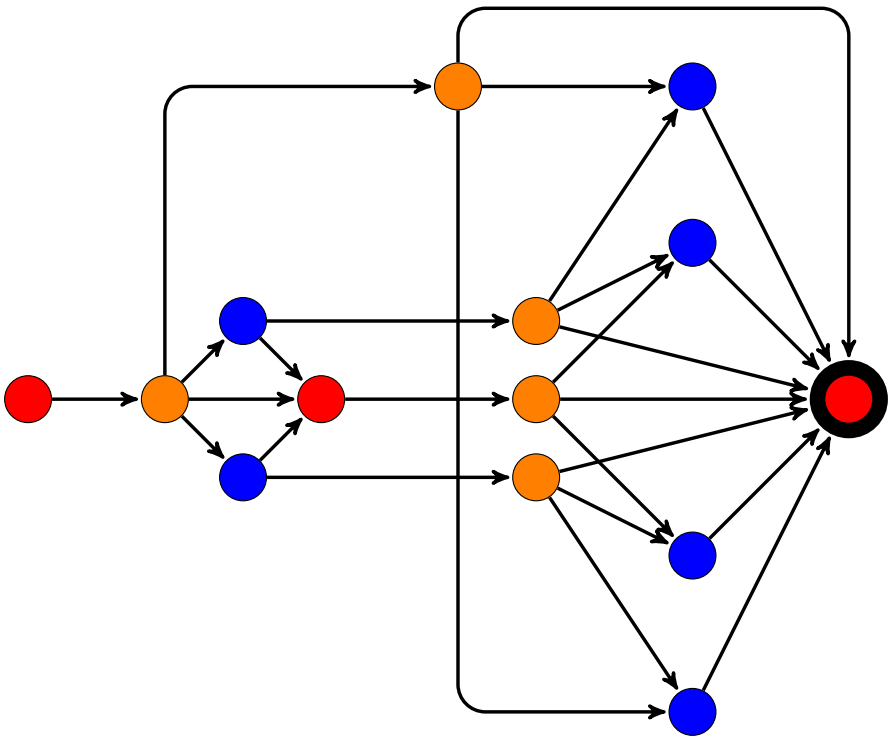
Remove terminal states

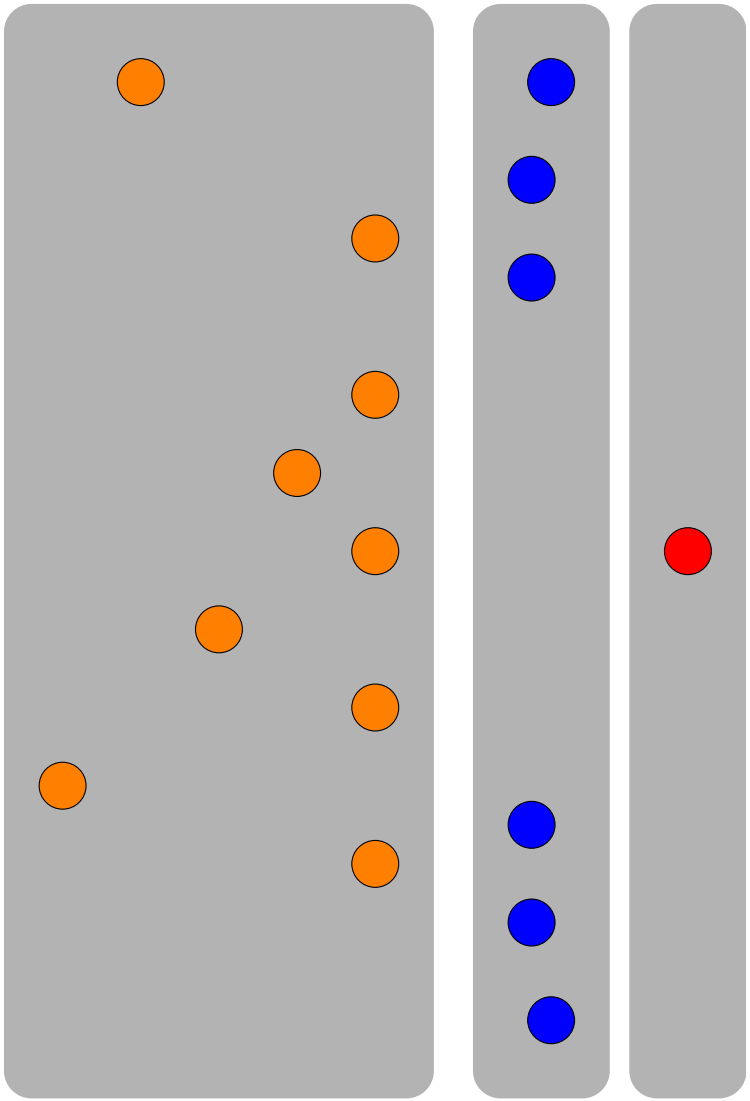




DAG recursion to partition S into stages

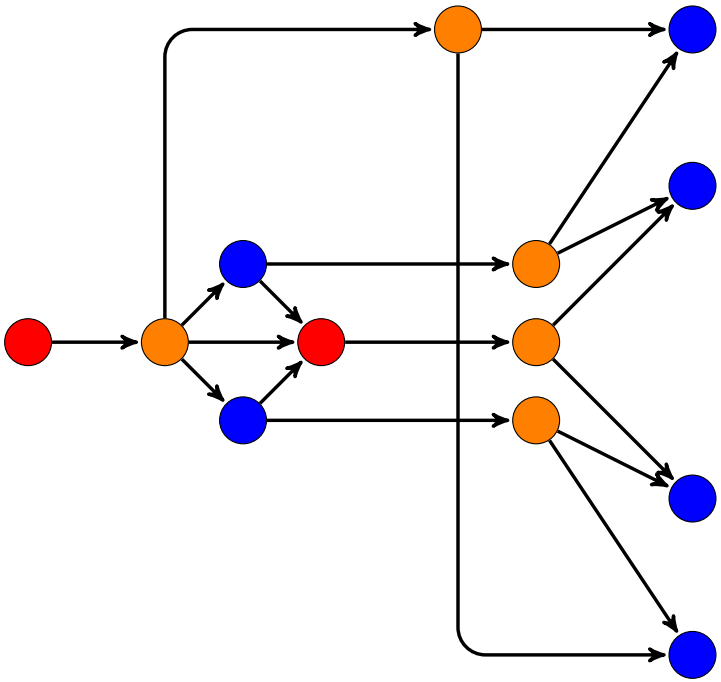
Identify terminal states

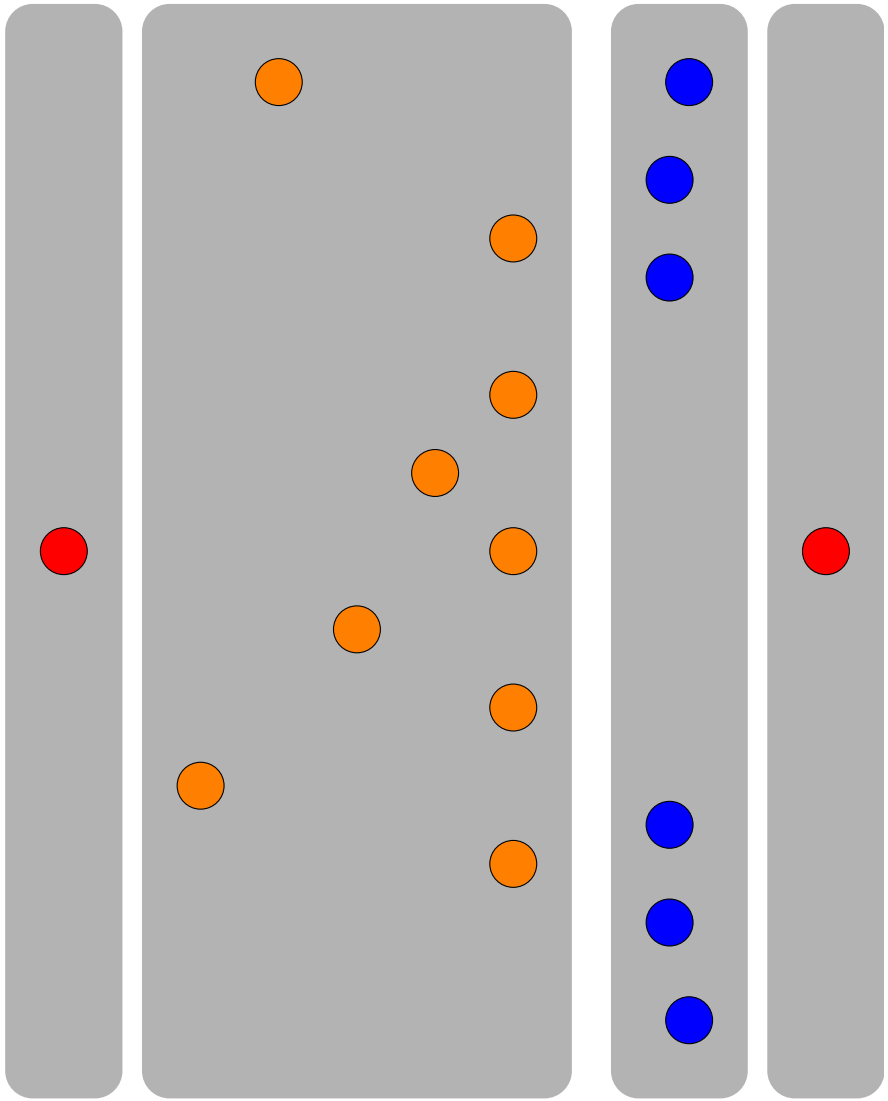




DAG recursion to partition S into stages

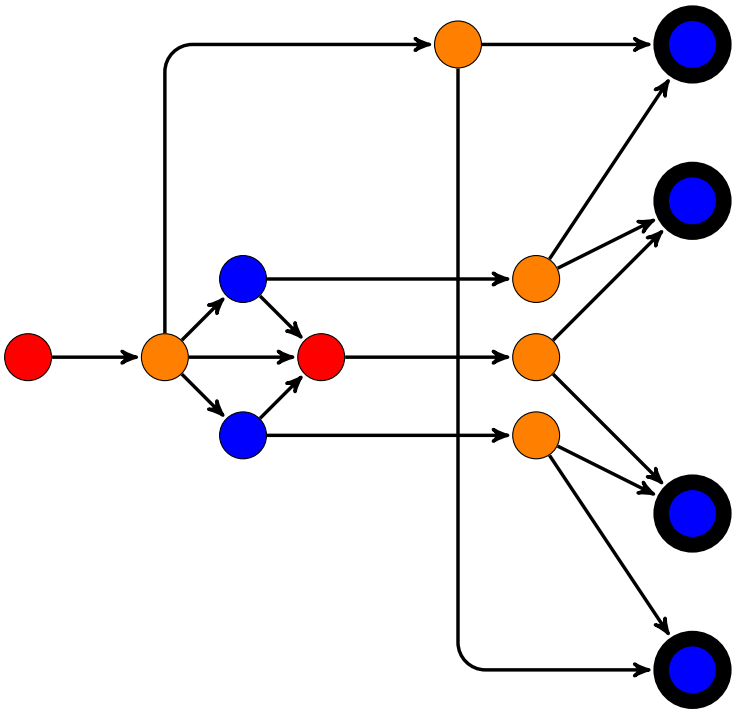
Remove terminal states

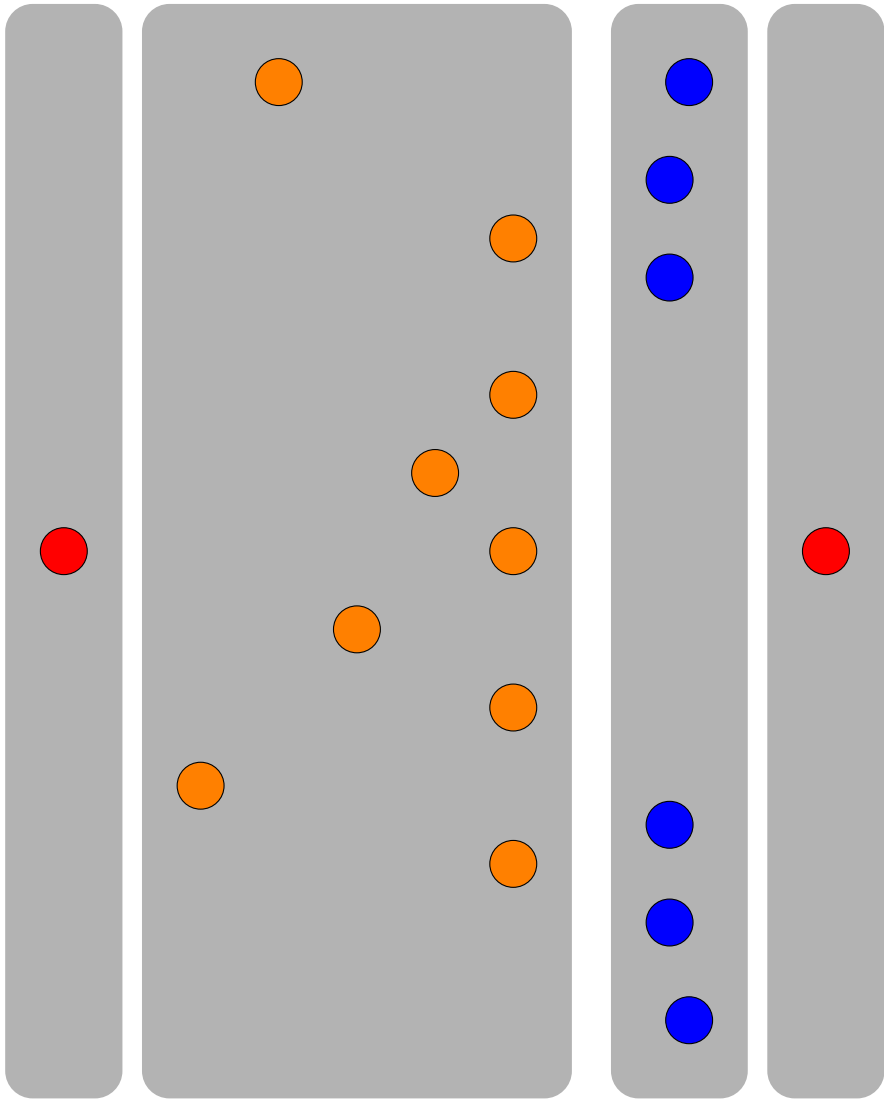




DAG recursion to partition S into stages

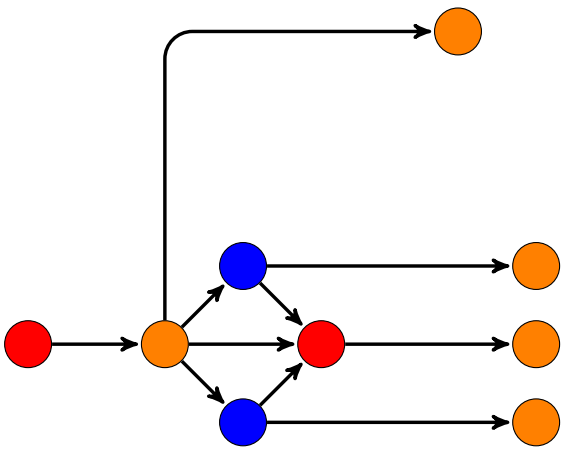
Identify terminal states

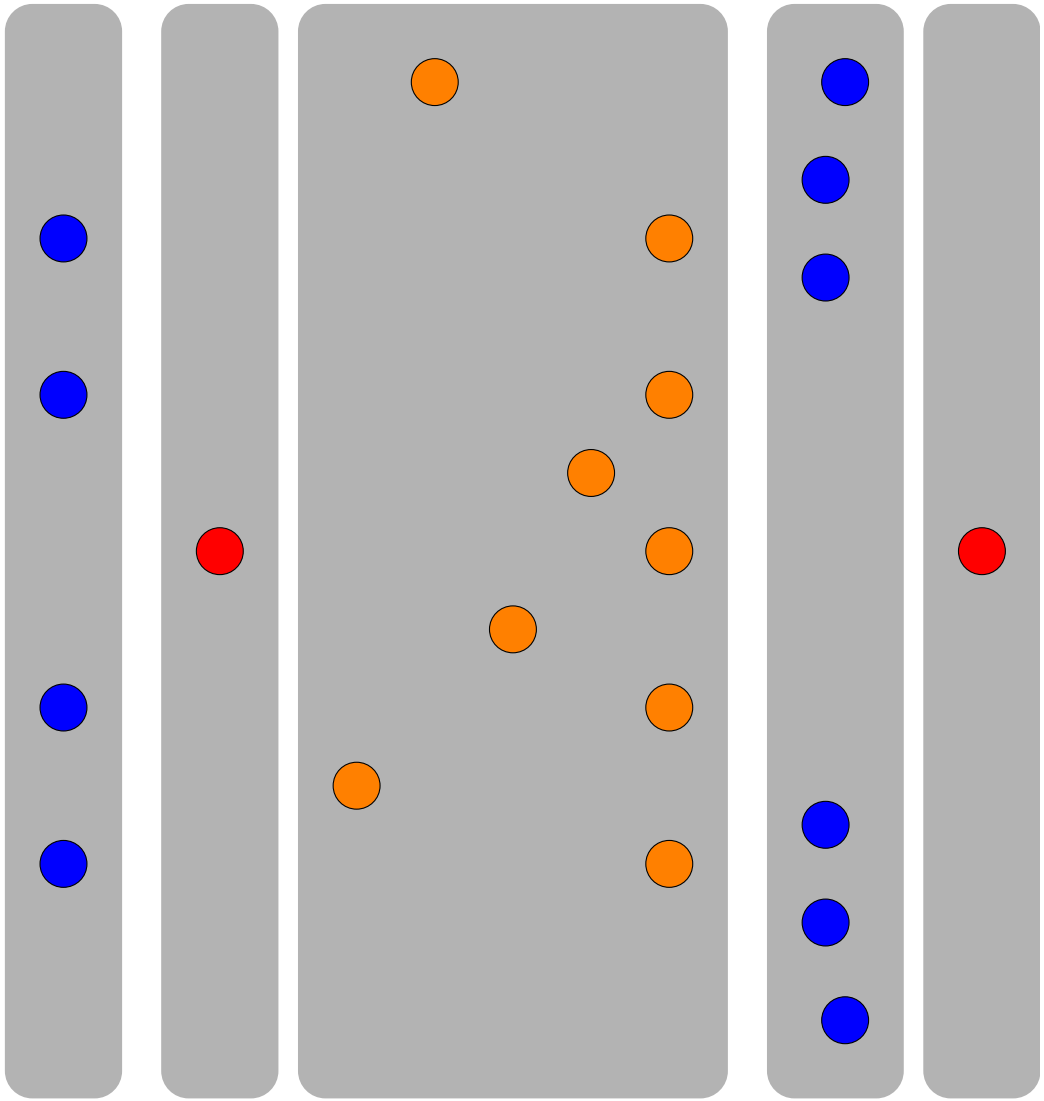




DAG recursion to partition S into stages

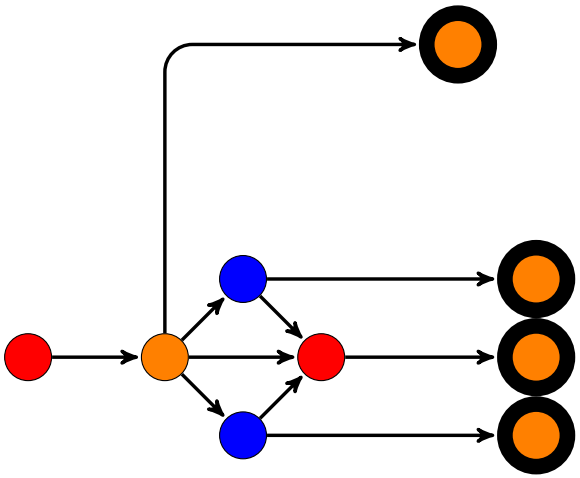
Remove terminal states

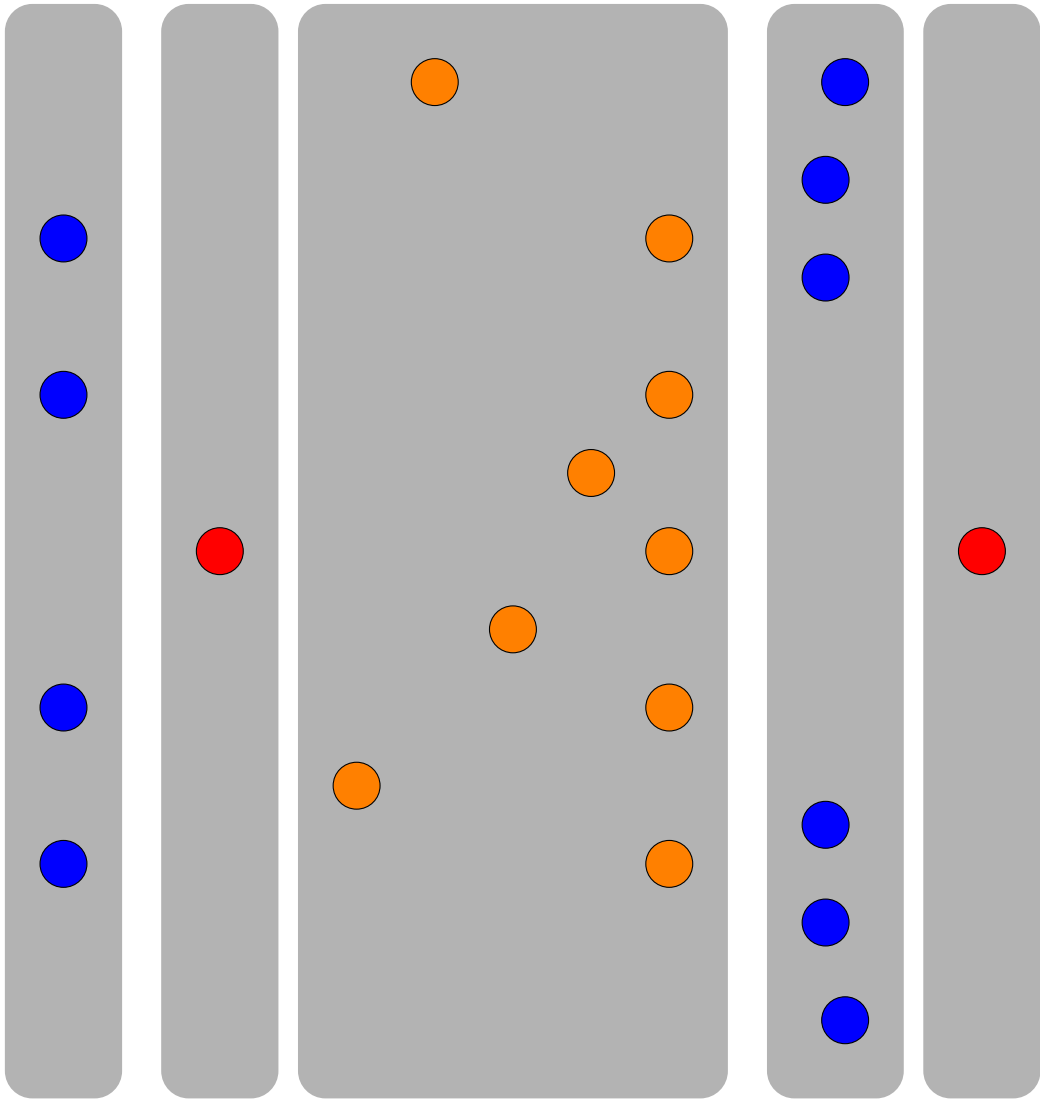




DAG recursion to partition S into stages

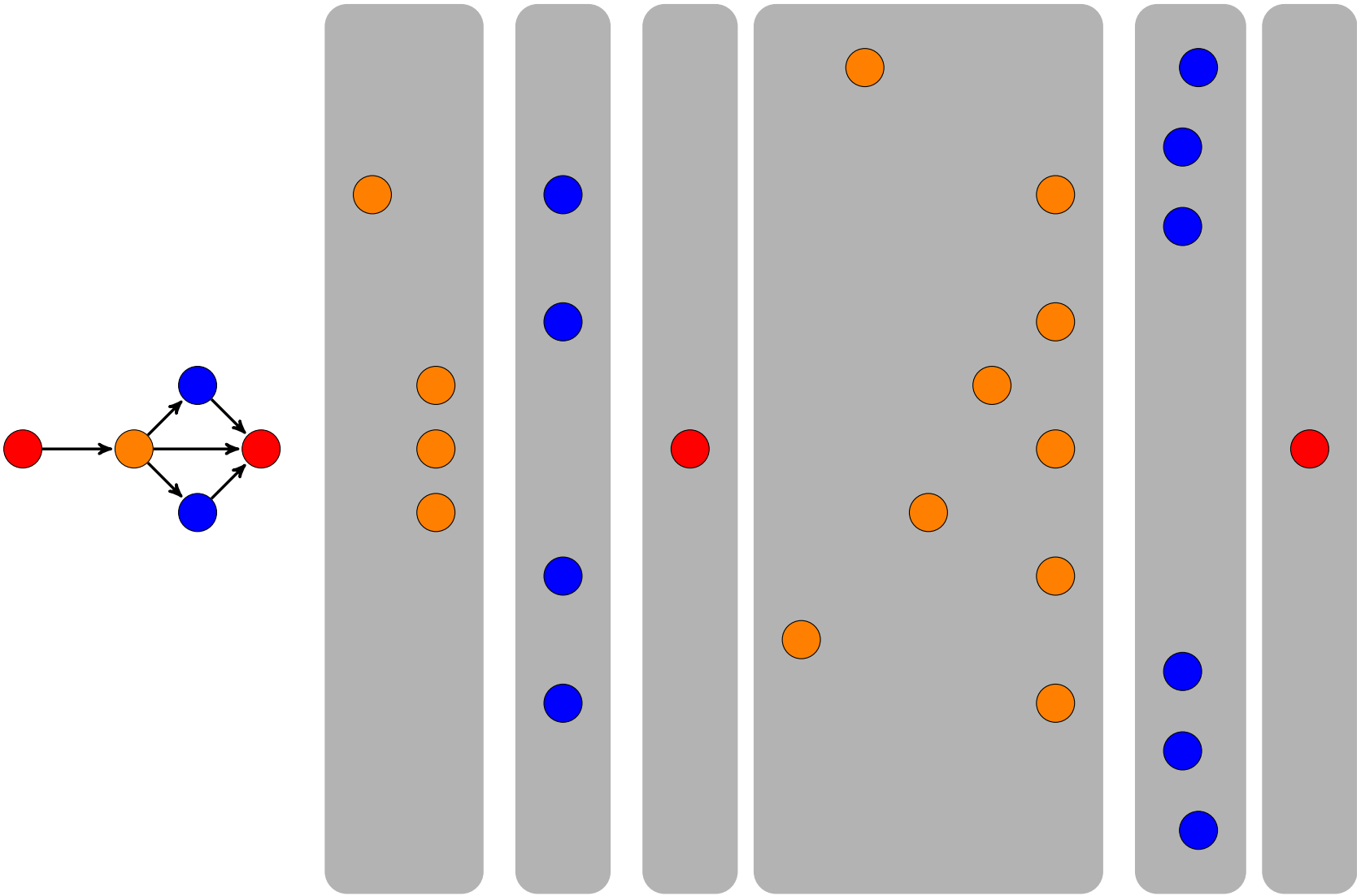
Identify terminal states





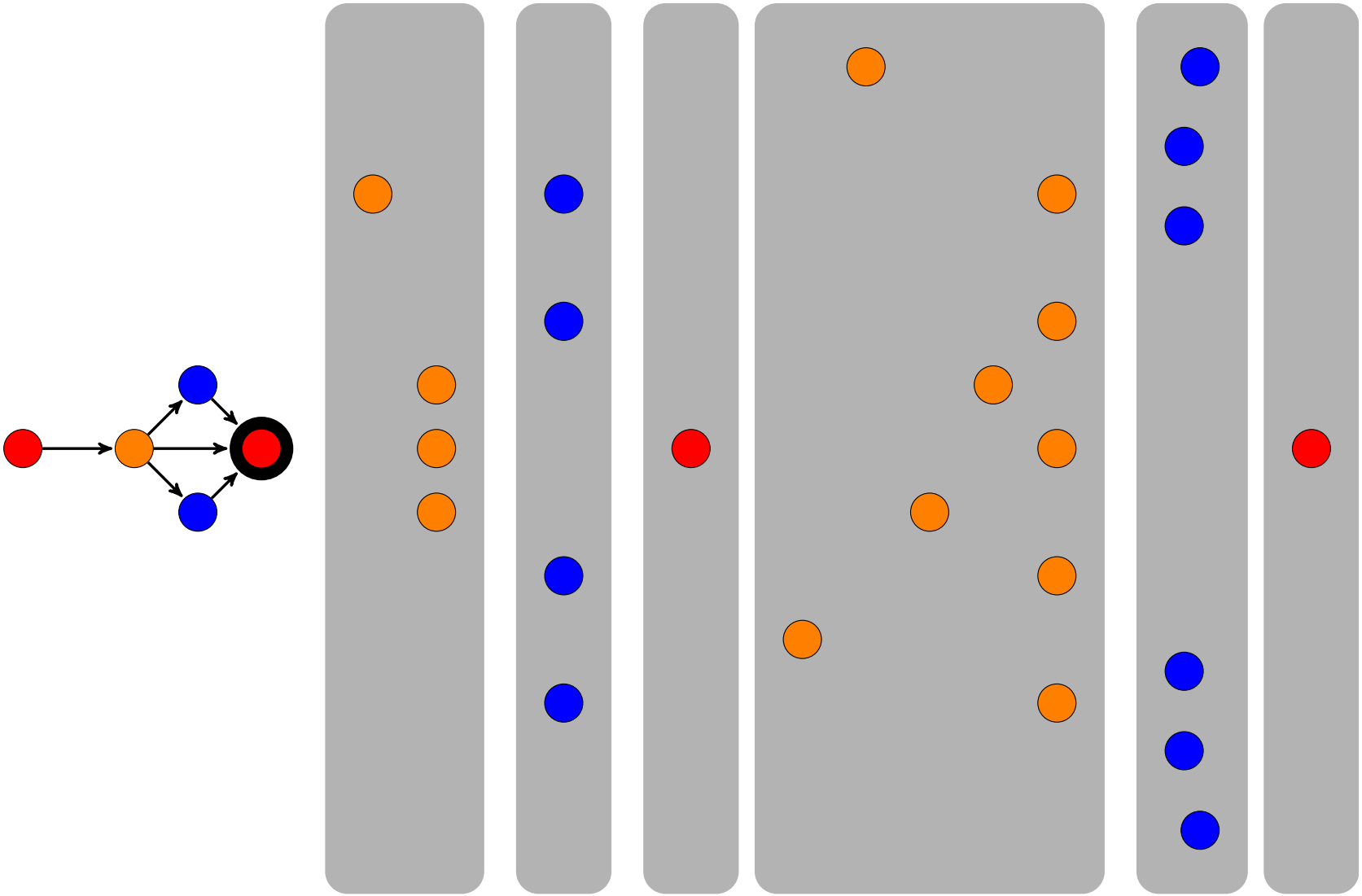
DAG recursion to partition S into stages

Remove terminal states



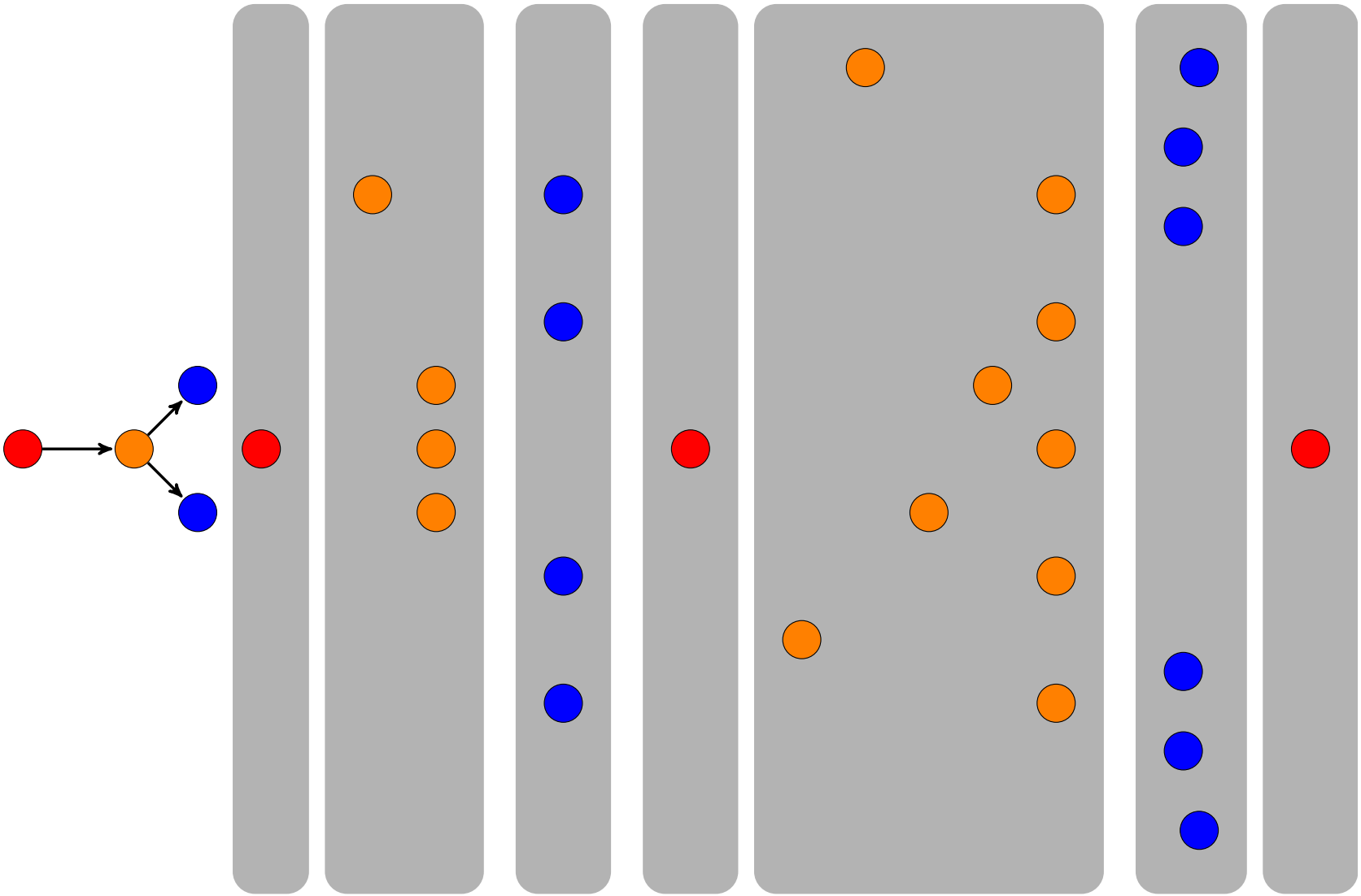
DAG recursion to partition S into stages

Identify terminal states



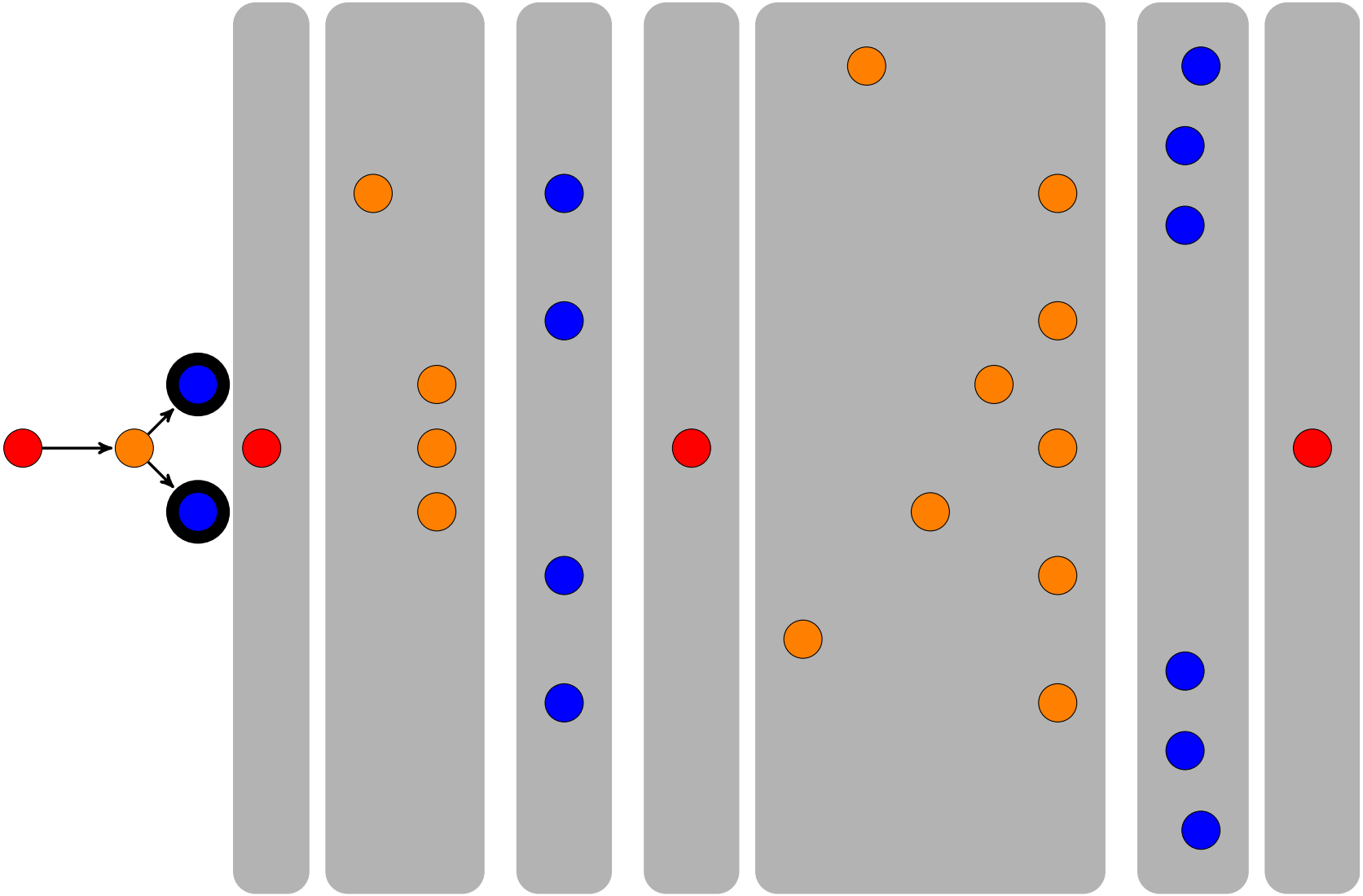
DAG recursion to partition S into stages

Remove terminal states



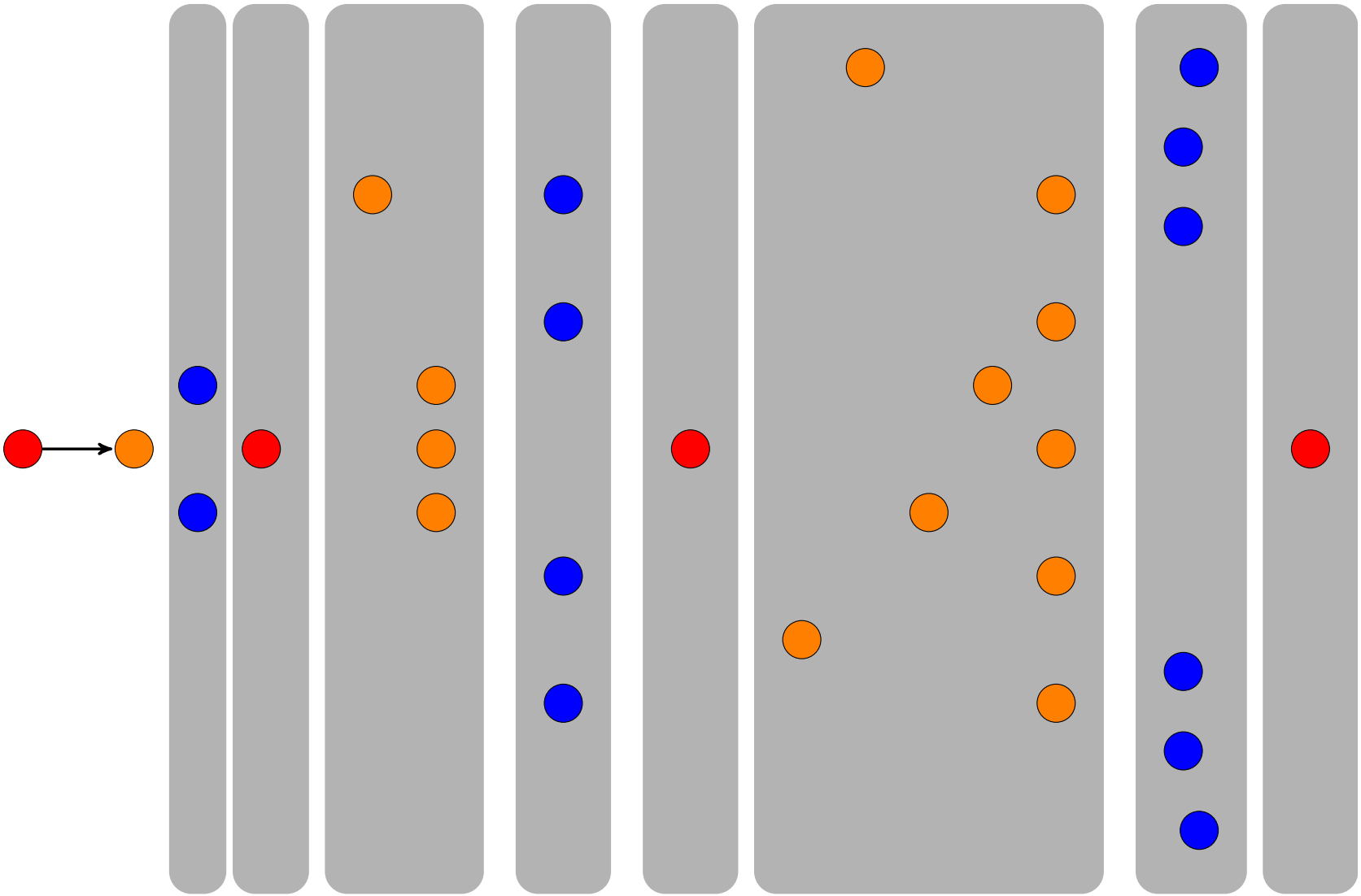
DAG recursion to partition S into stages

Identify terminal states



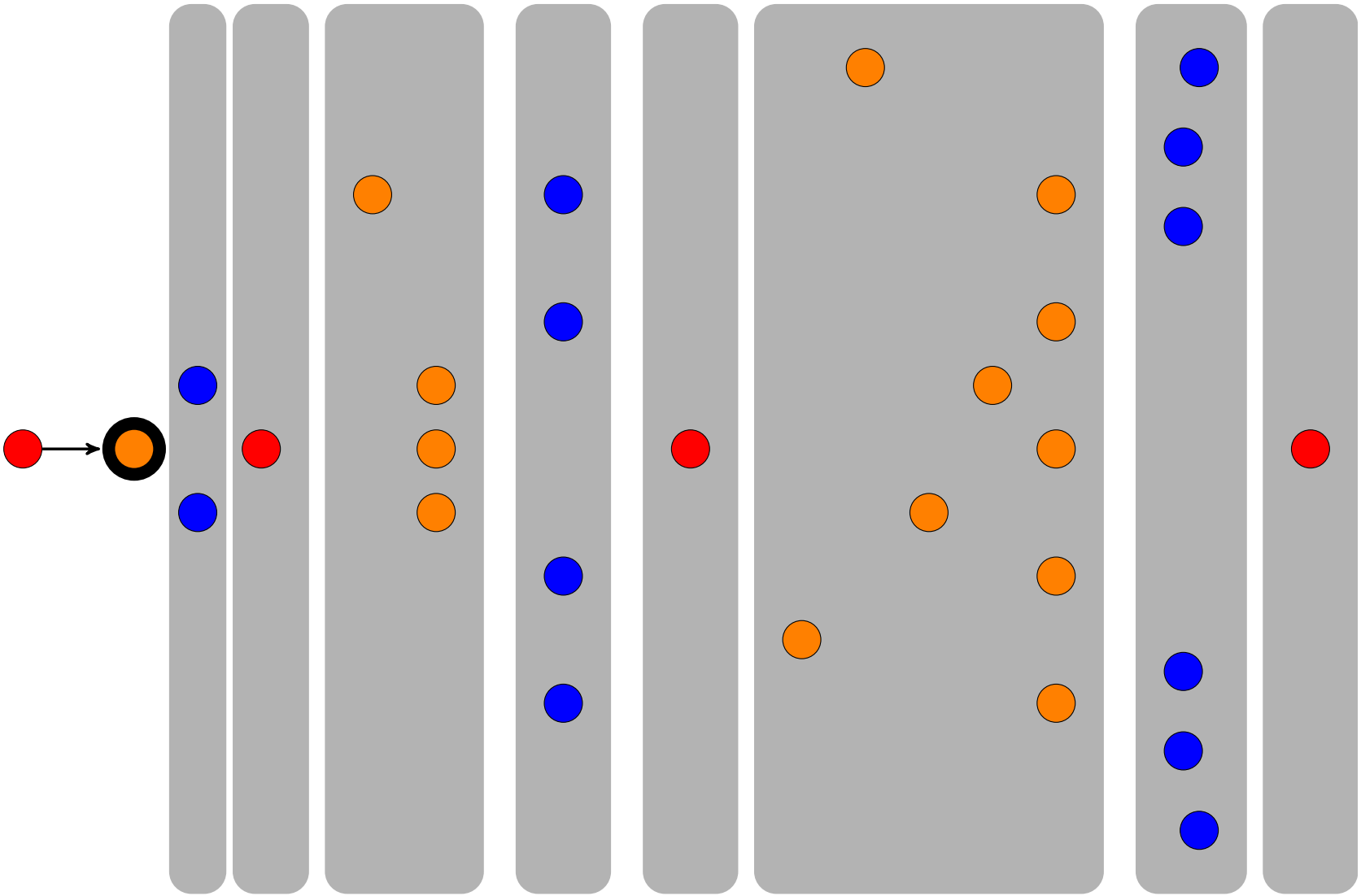
DAG recursion to partition S into stages

Remove terminal states



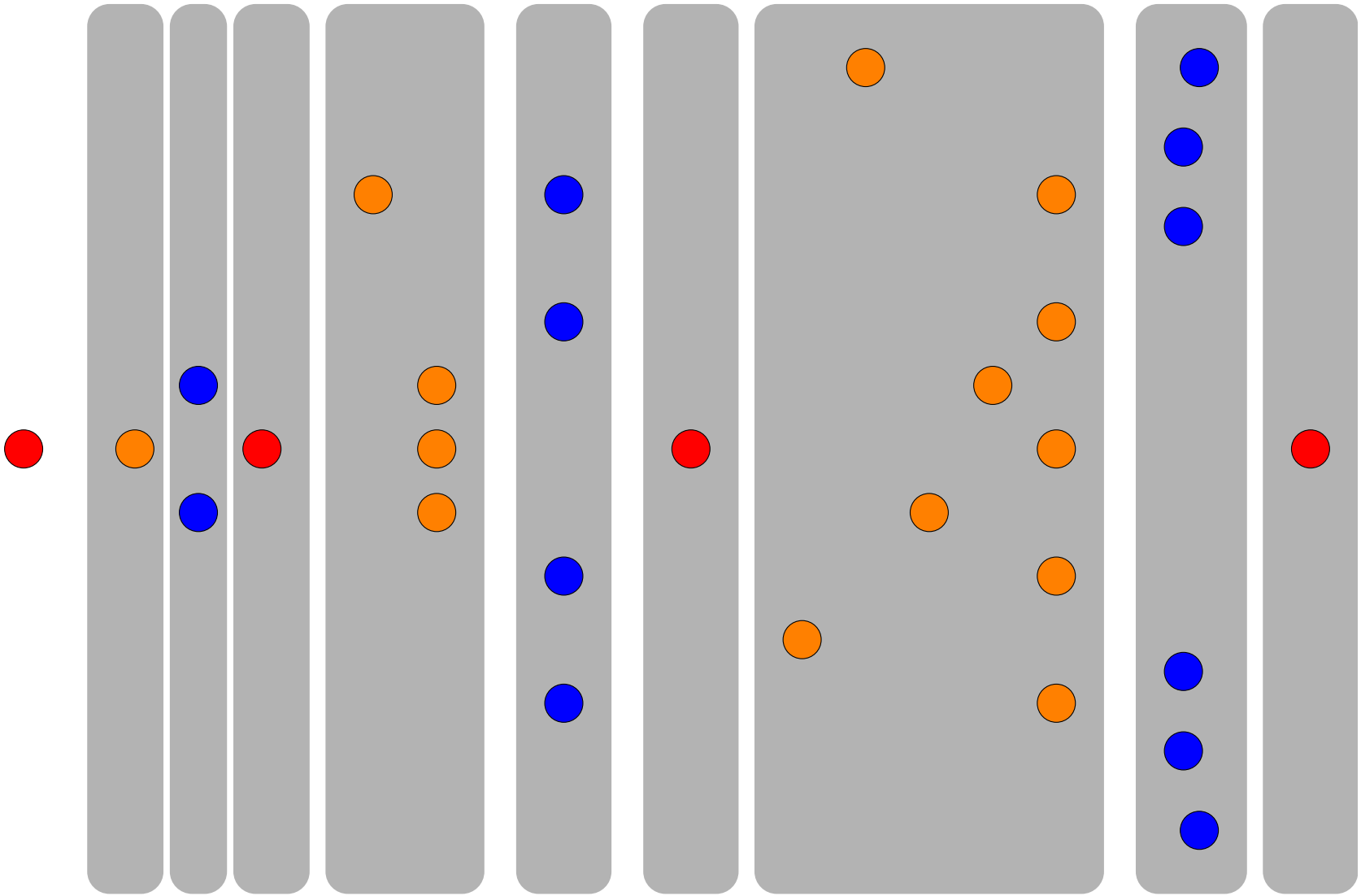
Resulting partition of S into stages

Identify terminal states



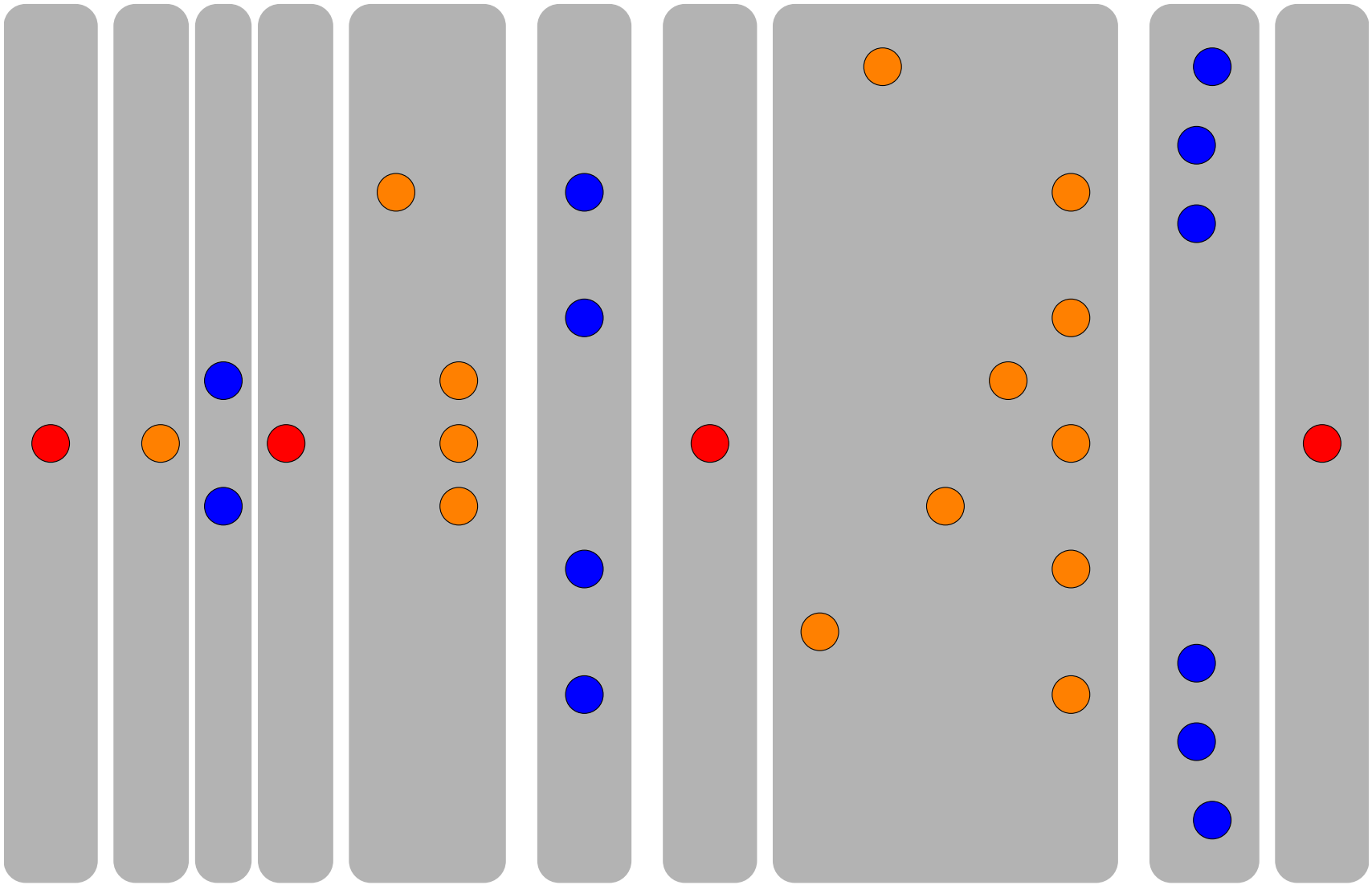
Resulting partition of S into stages

Remove terminal states



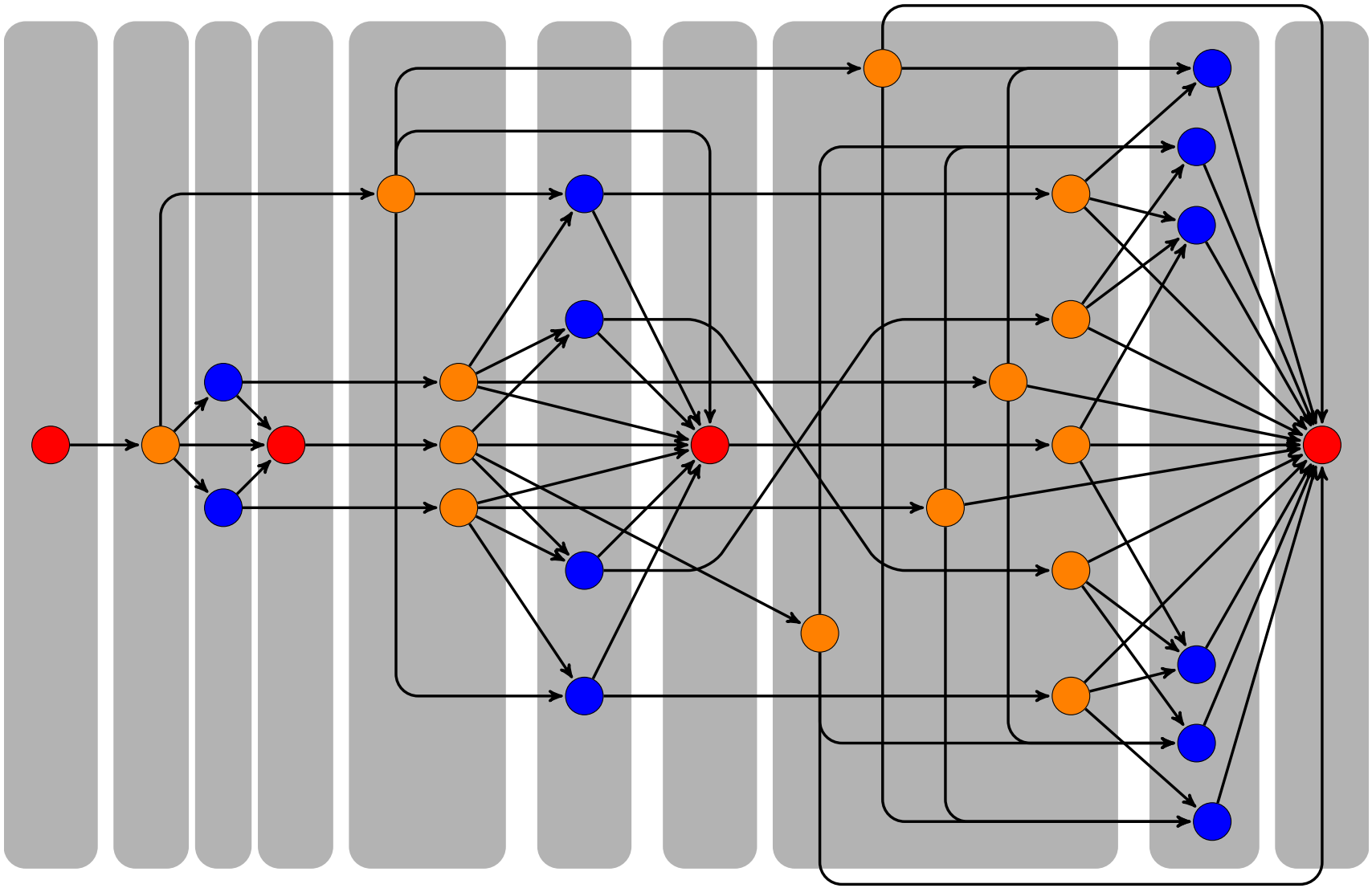
Resulting partition of S into stages

The stages of the game are totally ordered



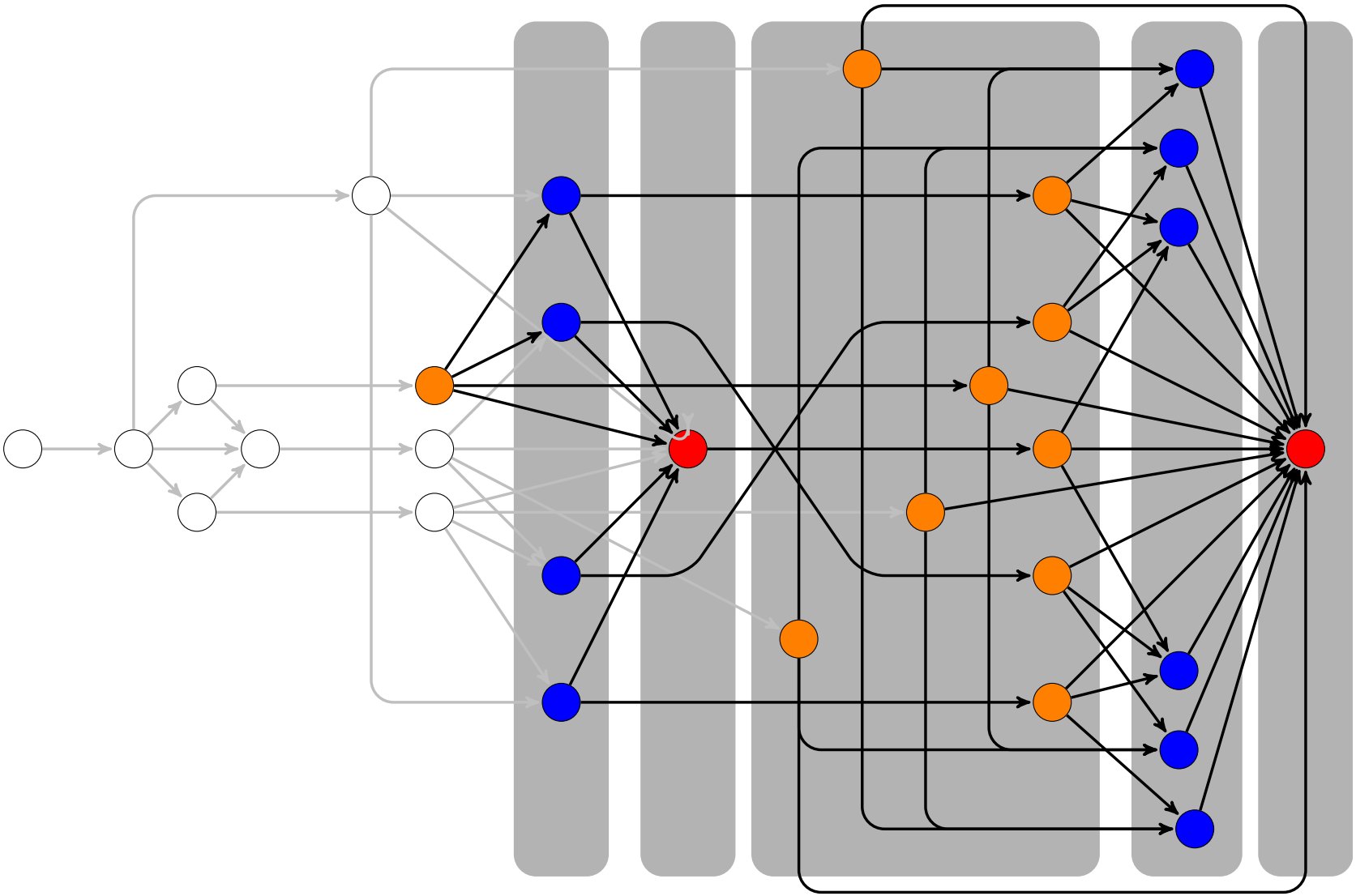
Subgames of DDG and continuation strategies

Subgames of DDG follow the order of stages



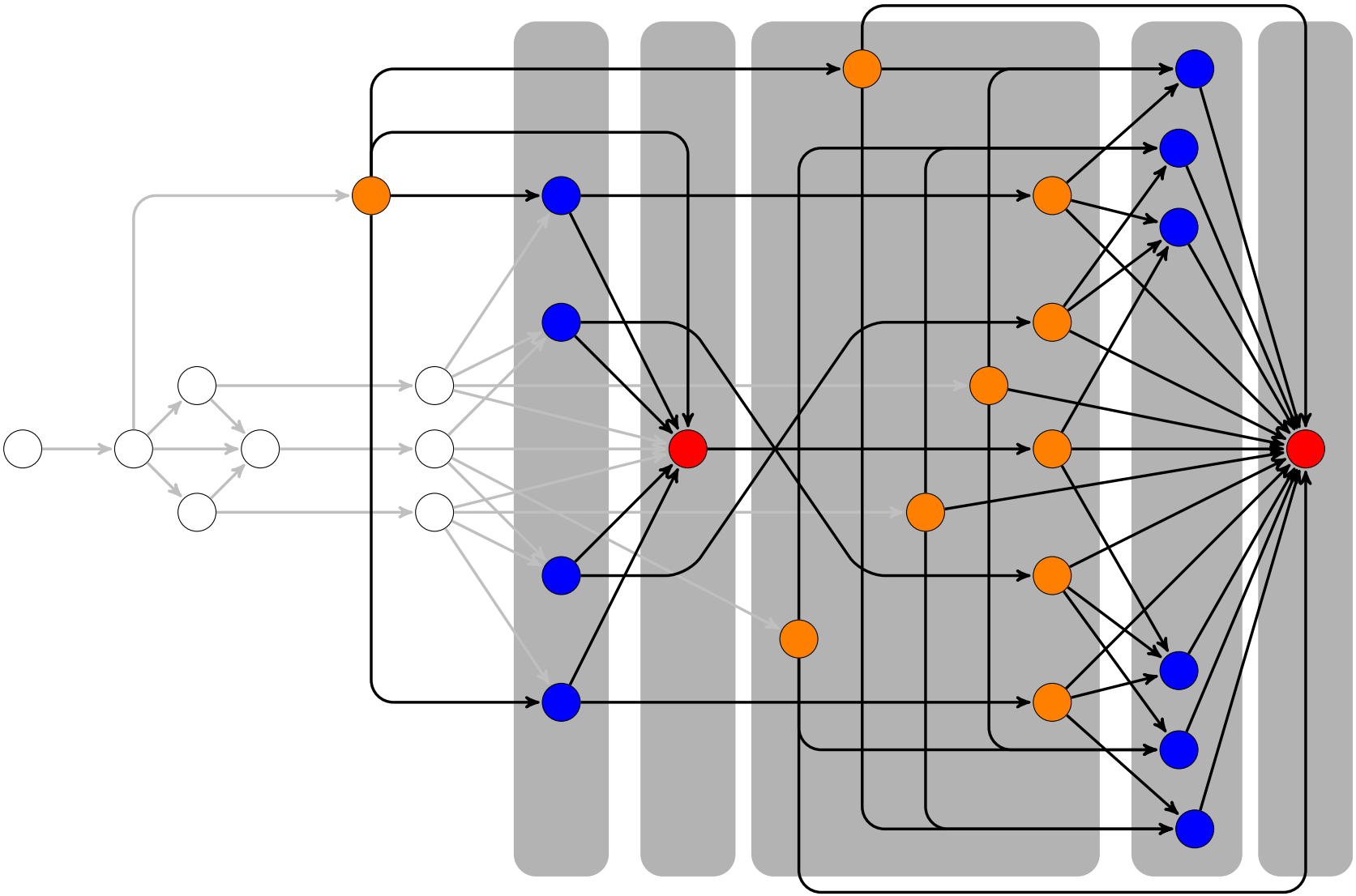
Subgames of DDG and continuation strategies

Solutions depends on later stages



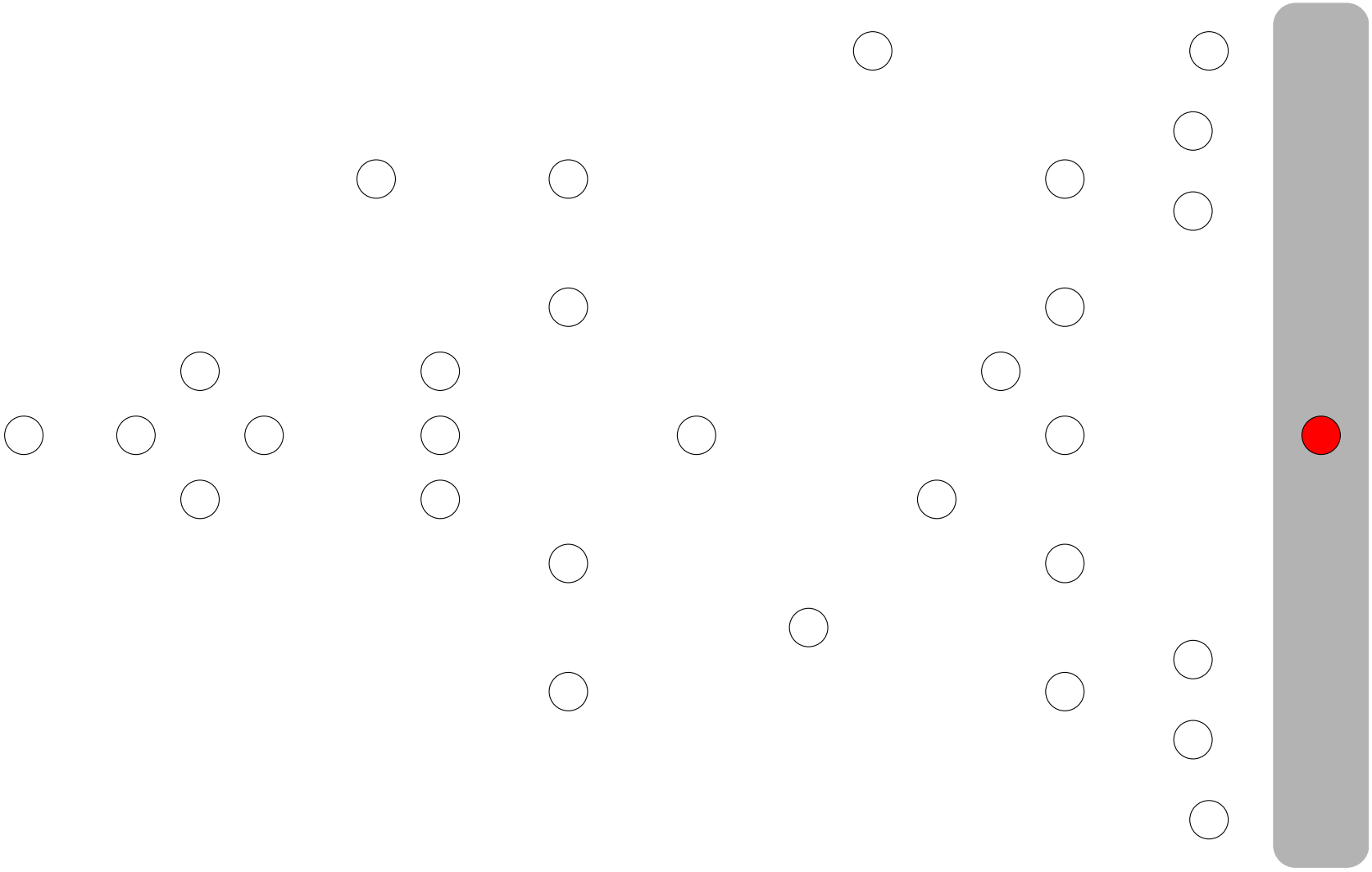
Subgames of DDG and continuation strategies

Within a given stage dependence is the same



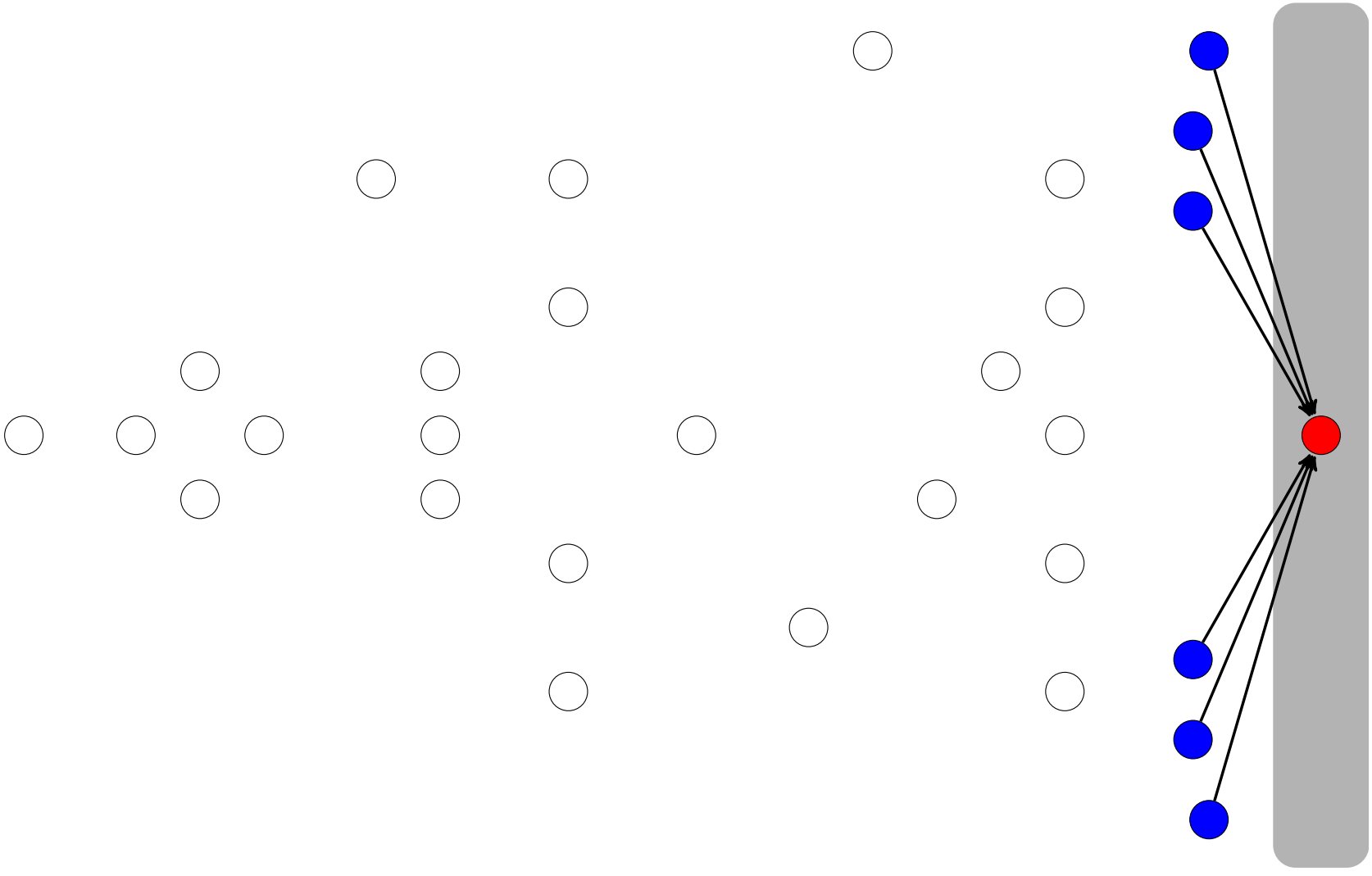
State recursion algorithm

Backward induction on stages of DDG



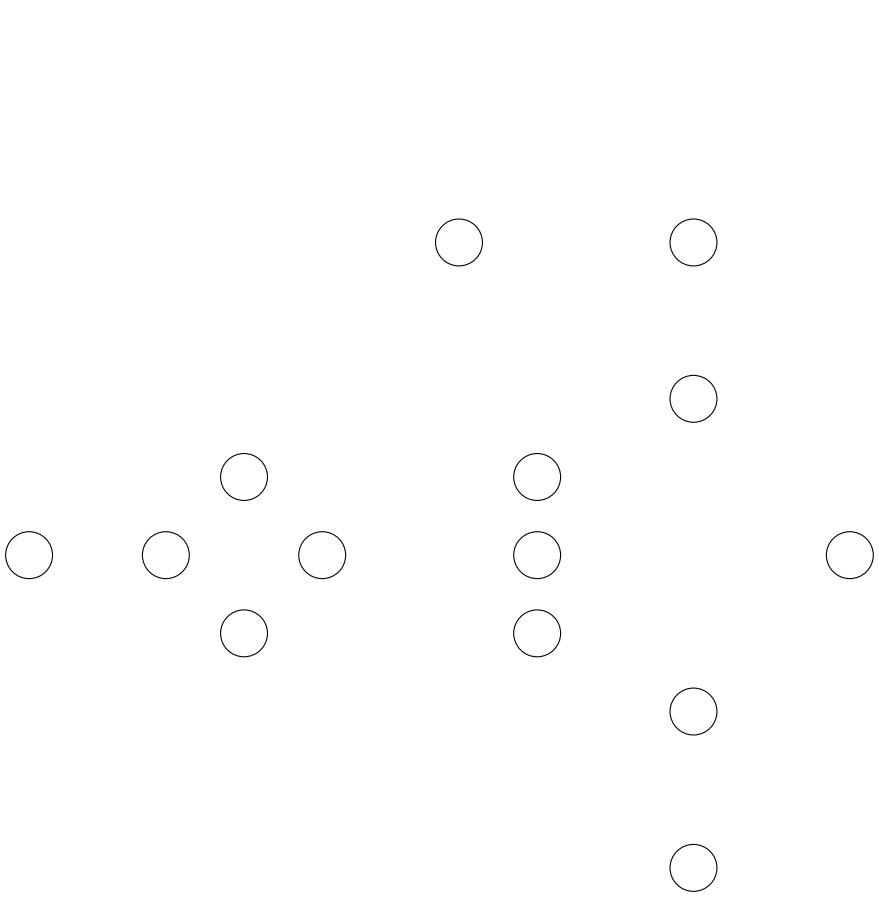
State recursion algorithm

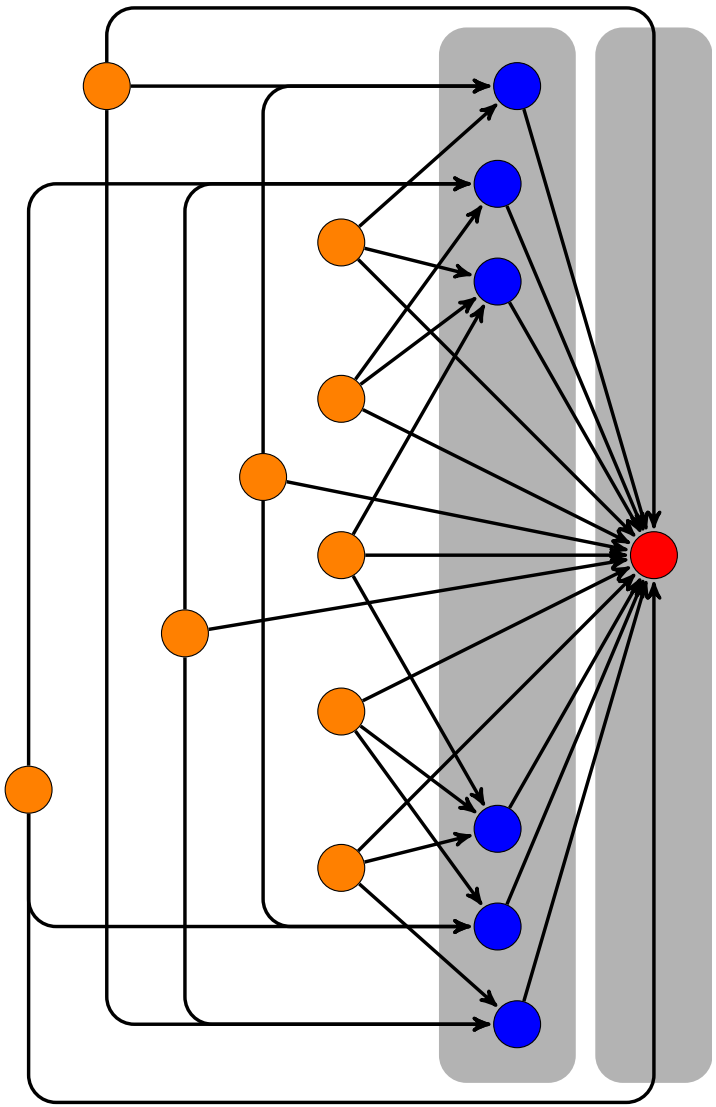
Backward induction on stages of DDG



State recursion algorithm

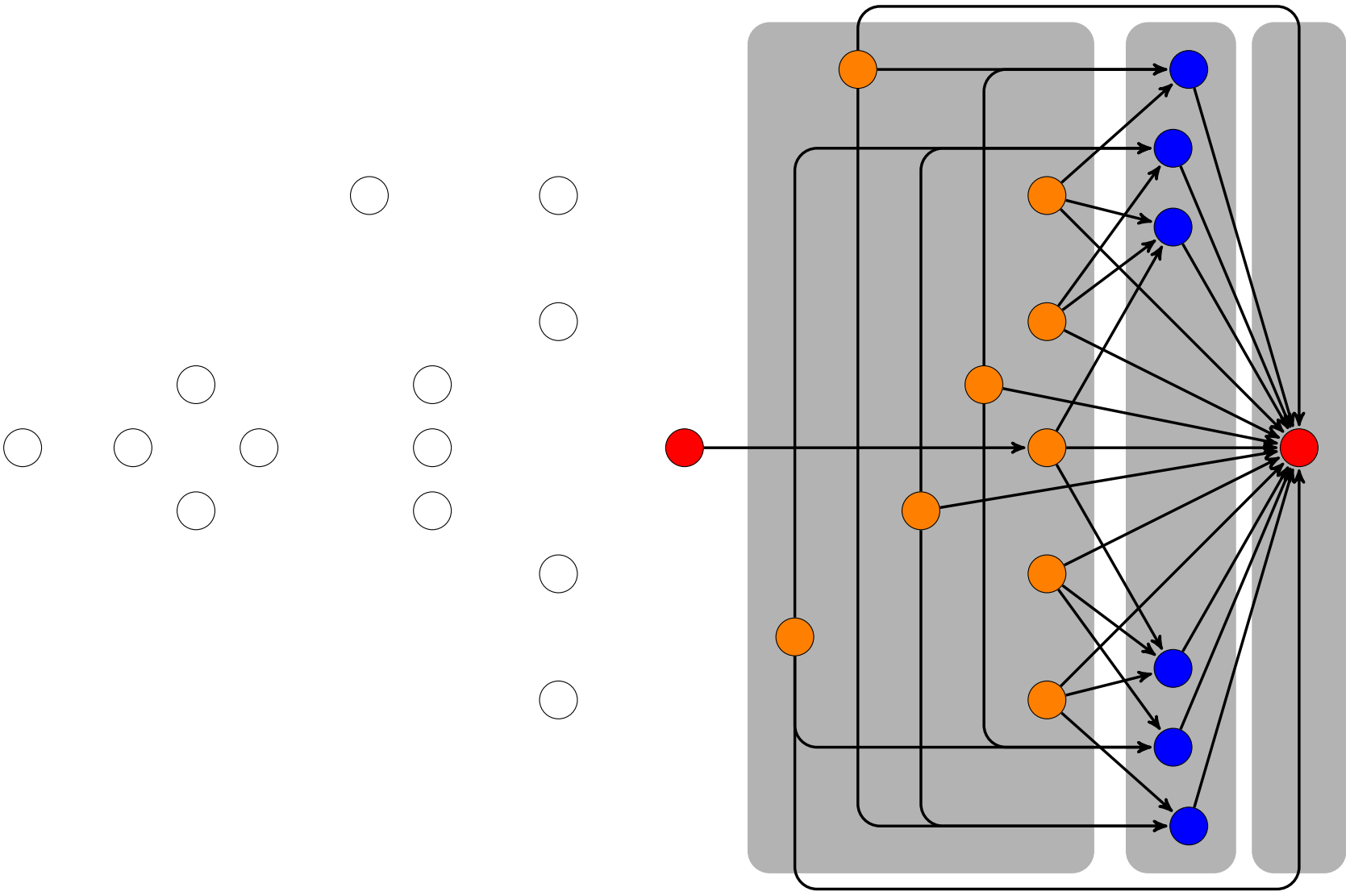
Backward induction on stages of DDG





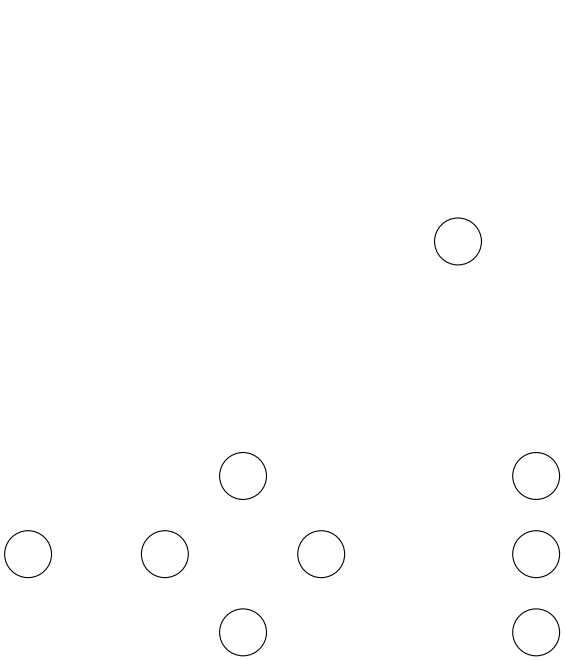
State recursion algorithm

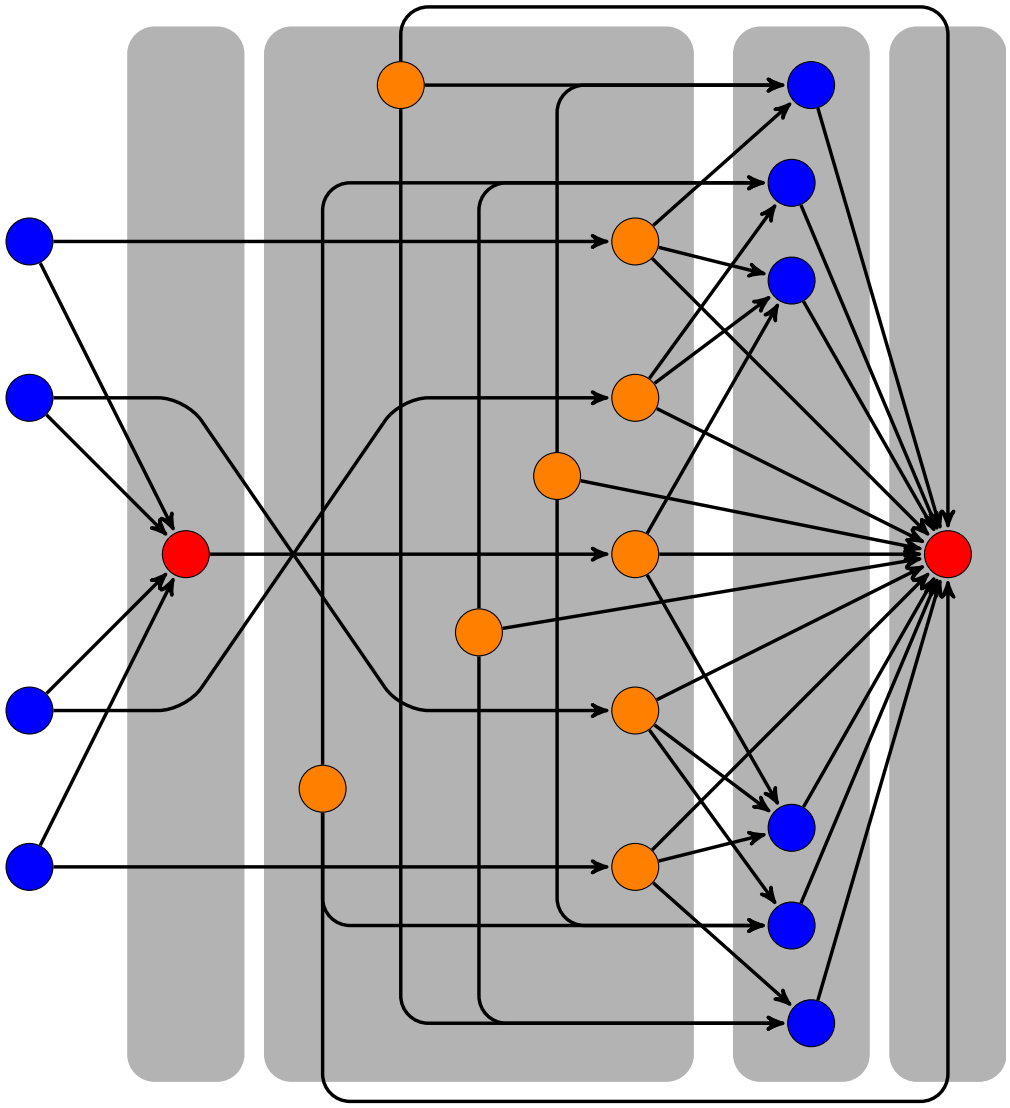
Backward induction on stages of DDG



State recursion algorithm

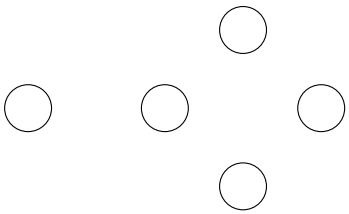
Backward induction on stages of DDG

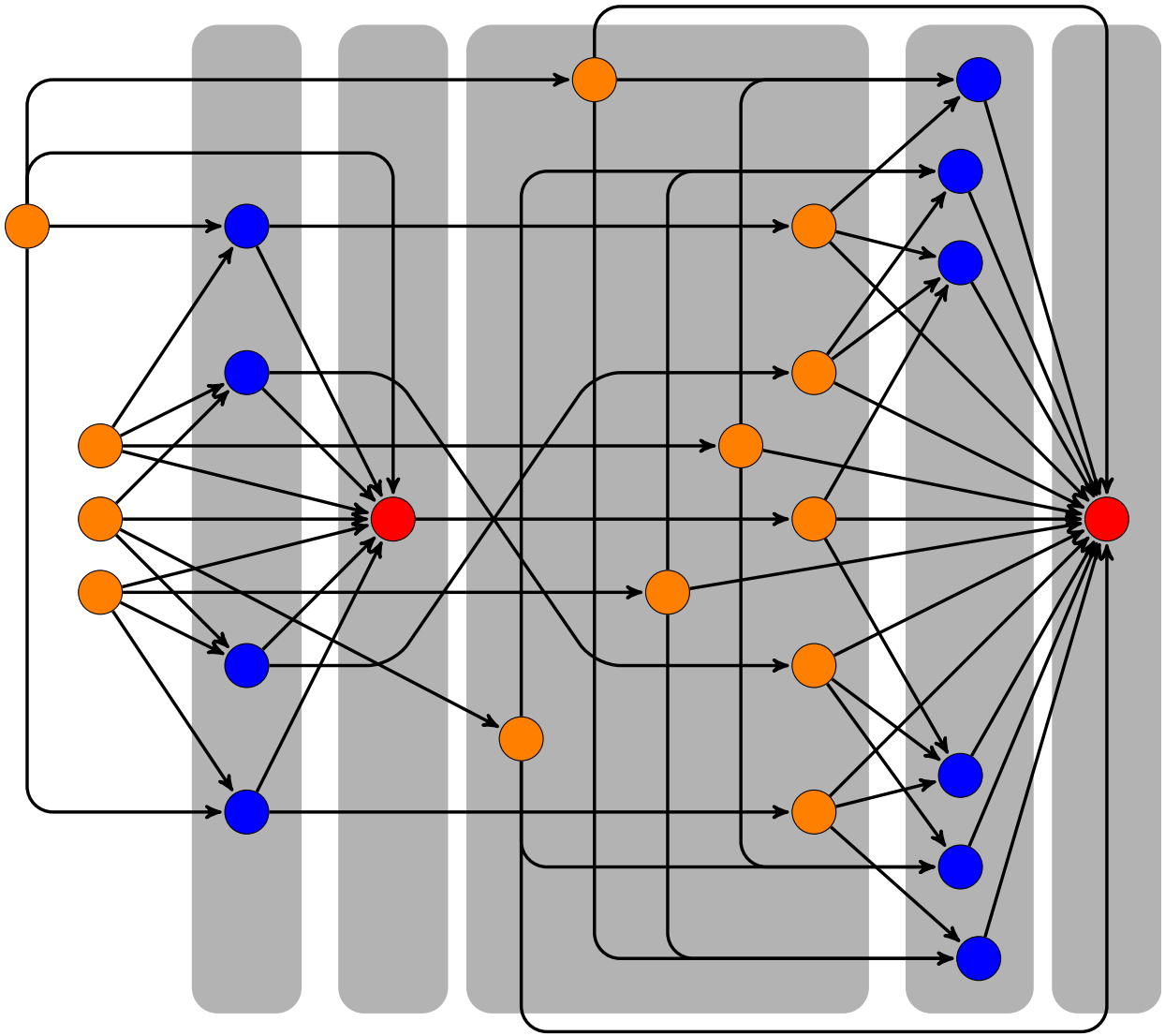




State recursion algorithm

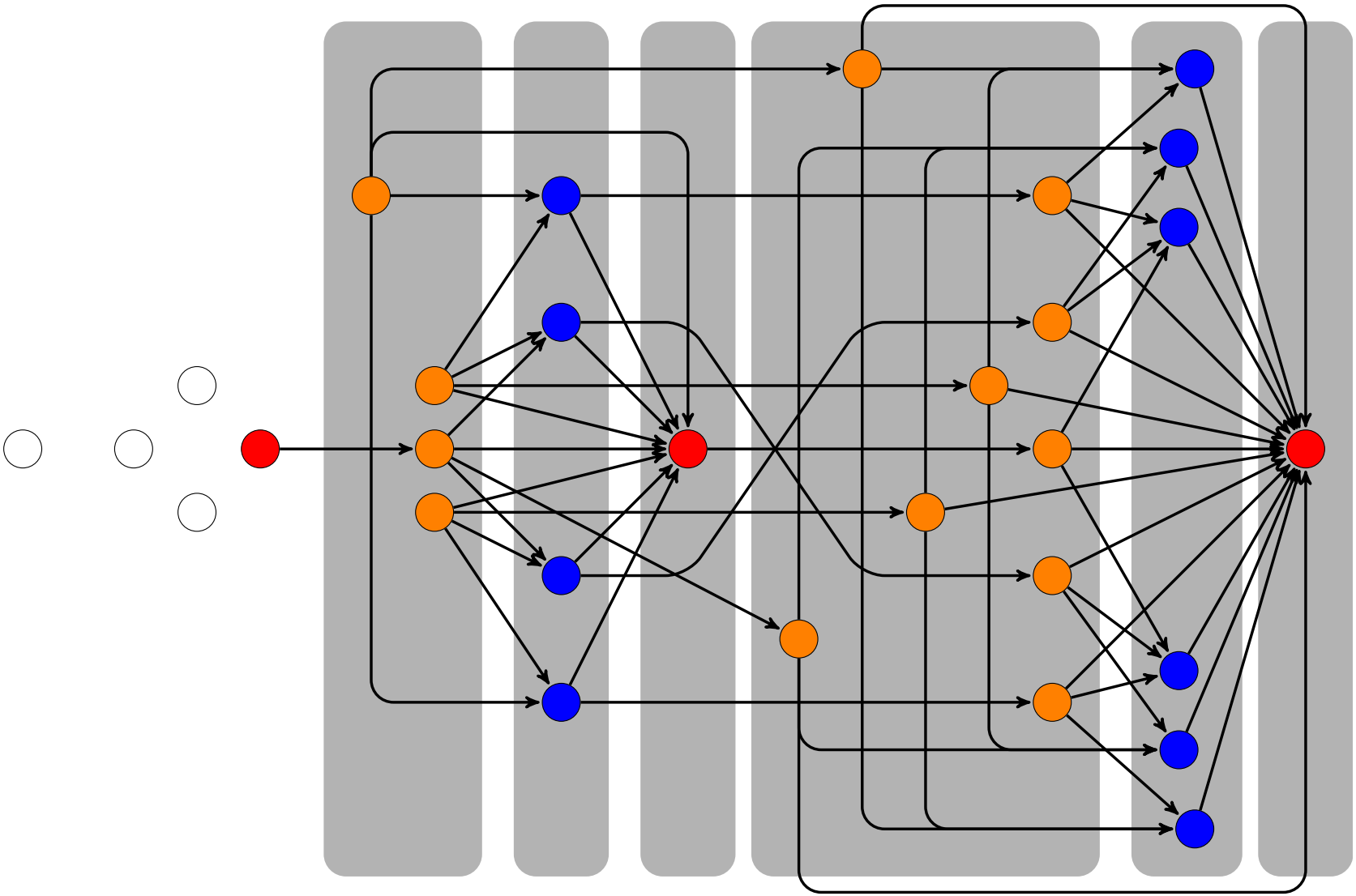
Backward induction on stages of DDG





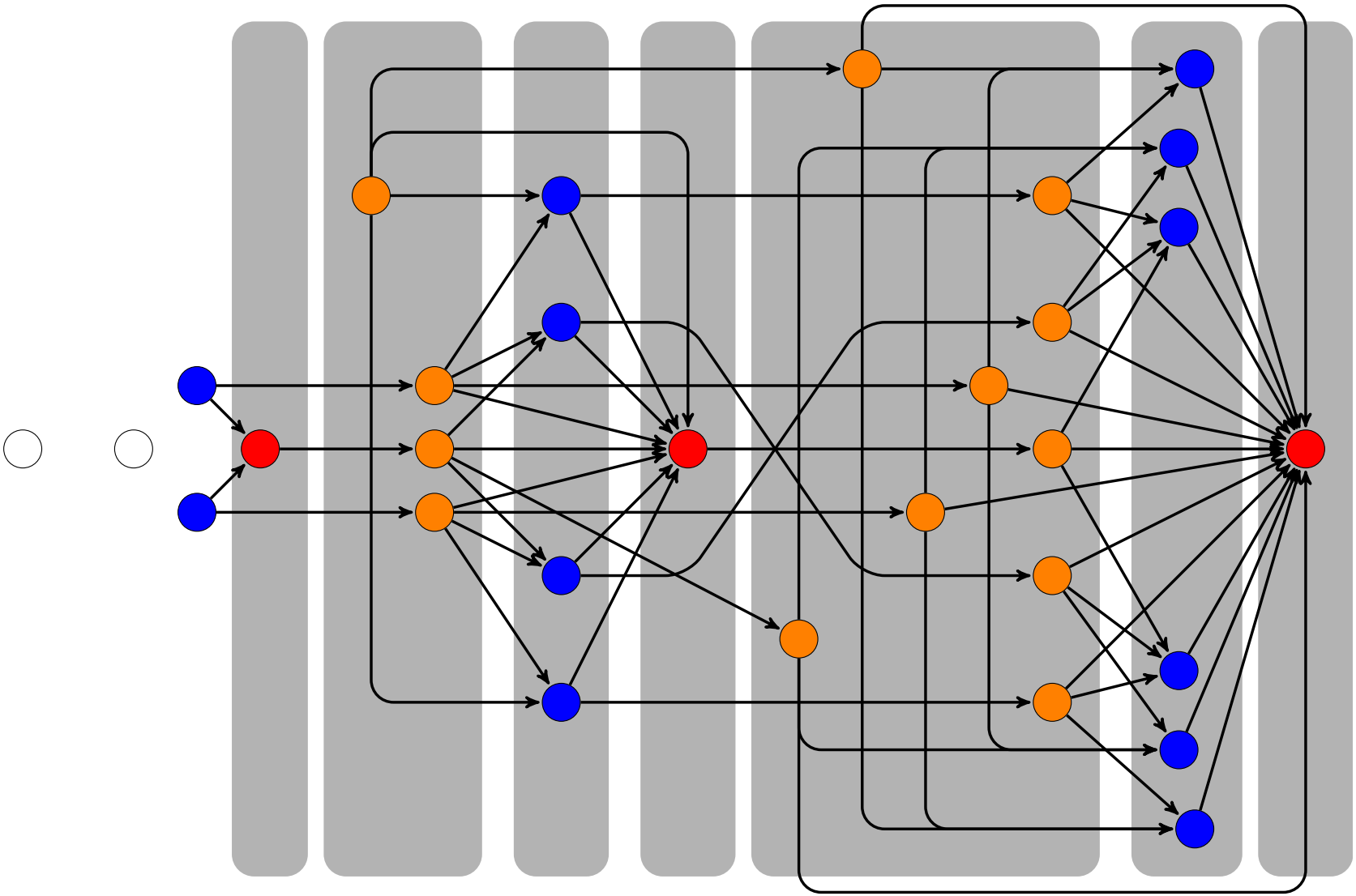
State recursion algorithm

Backward induction on stages of DDG



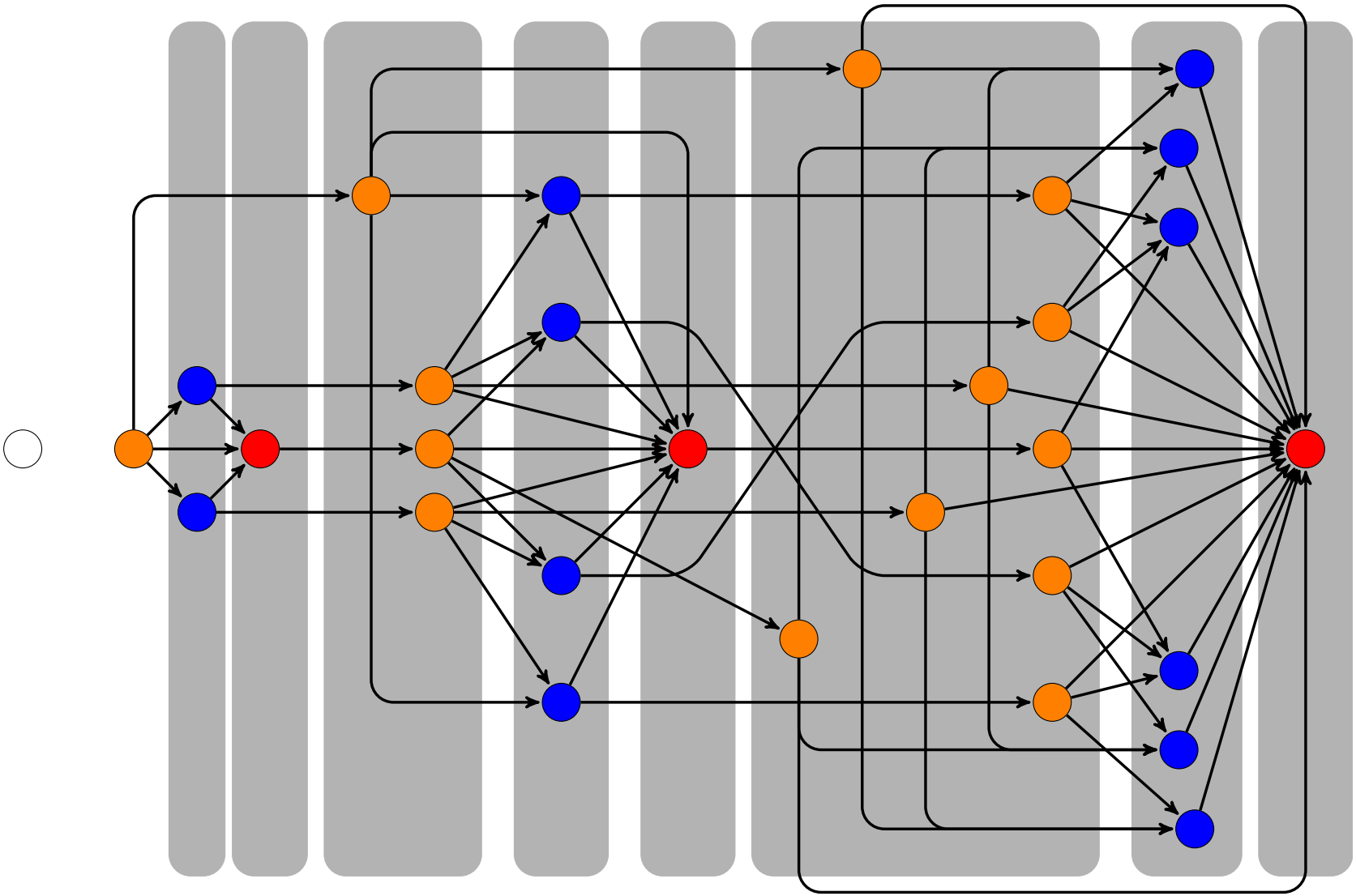
State recursion algorithm

Backward induction on stages of DDG



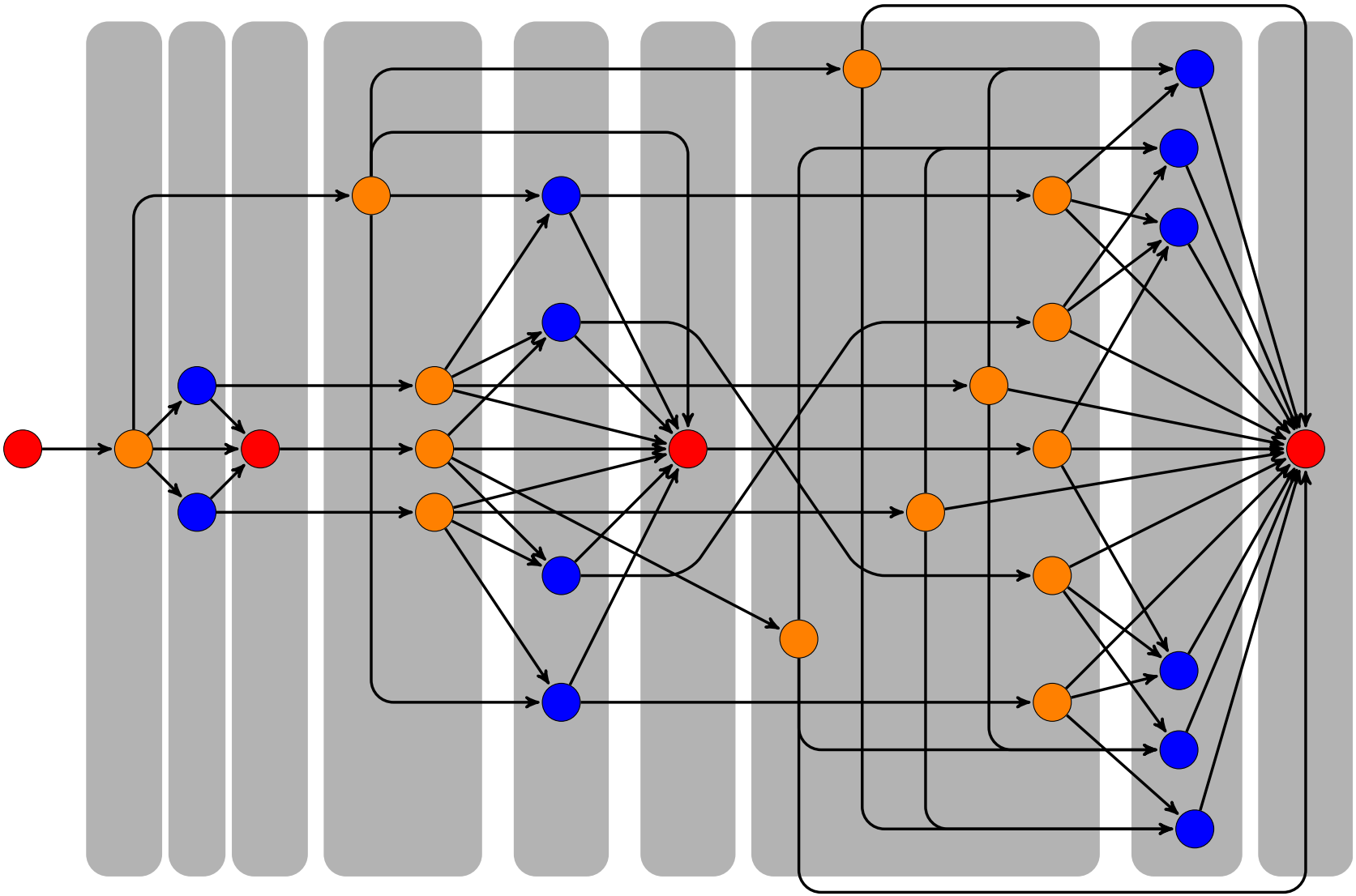
State recursion algorithm

Backward induction on stages of DDG



State recursion algorithm

Backward induction on stages of DDG

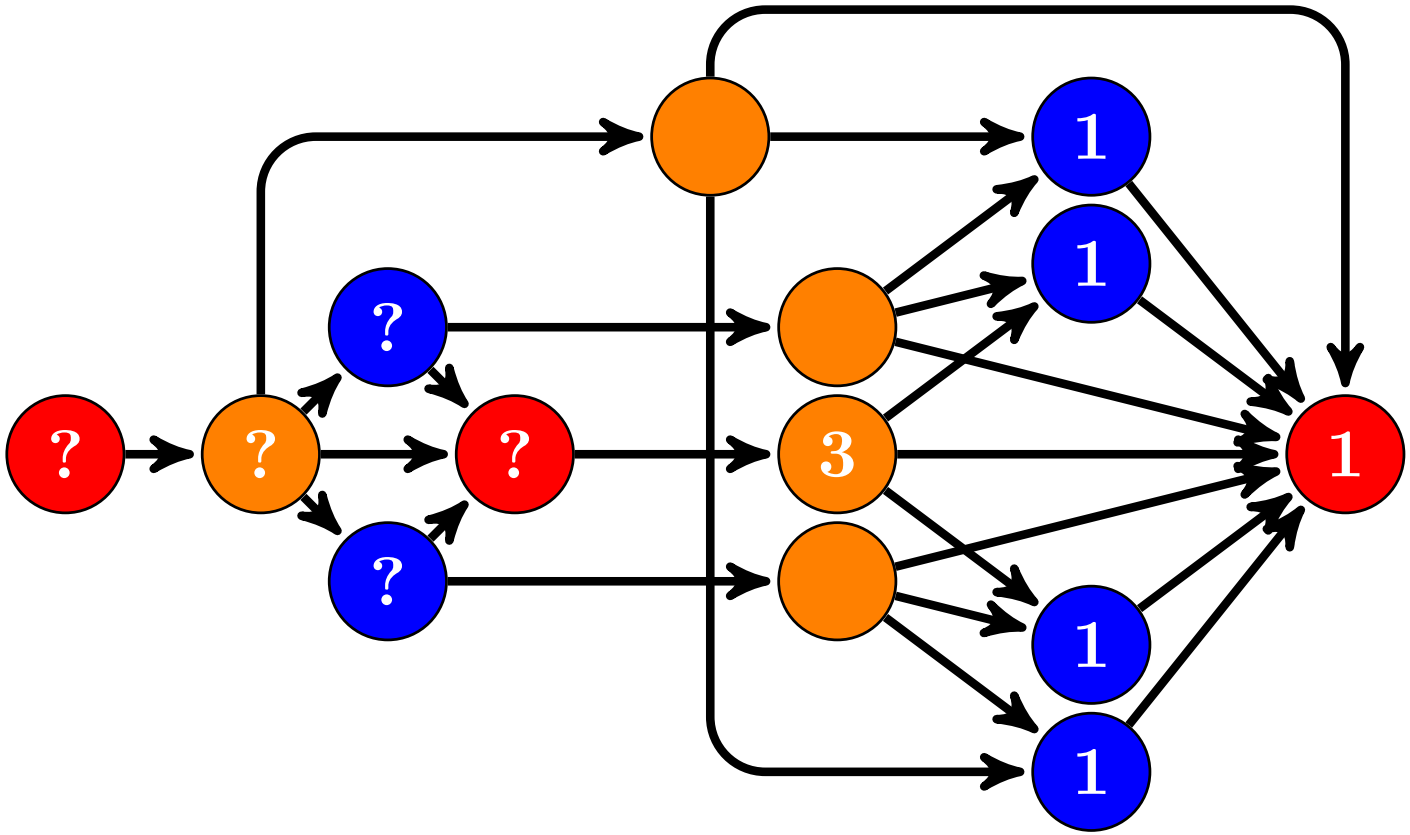


State Recursion versus Backward Induction

- State recursion – generalization of backward induction
- Runs on state space instead of time periods
- Time (t) evolves as $t \to t+1$ with probability 1
- For stages of state space (τ) transitions are stochastic and not necessarily sequential
- Yet, probability of going $\tau \to \tau'$ is zero when $\tau' < \tau$
- With multiplicity, state recursion is performed conditional of a particular equilibrium selection rule (ESR)

Multiplicity of equibiria

Number of equilibria in the higher stages depends on the selected equilibria



Road map of the talk

1. Original application: collusion of Australian corrugated fibre packaging (CFP) producers
2. State recursion algorithm
3. Recursive lexicographical search (RLS) algorithm
   - Encoding of equilibrium selection rules (ESR)
   - Number of MPE in stage games and feasibility of ESR
   - Variable base arithmetics and successor function
   - Weak and strong versions of RLS algorithm
4. Solving the Bertrand investment game using RLS
5. Embedding full solution into the nested ML estimator

Recursive Lexicographic Search Algorithm

Building blocks of RLS algorithm:

1. State recursion algorithm solves the game conditional on equilibrium selection rule (ESR)
2. RLS algorithm efficiently cycles through all feasible ESRs

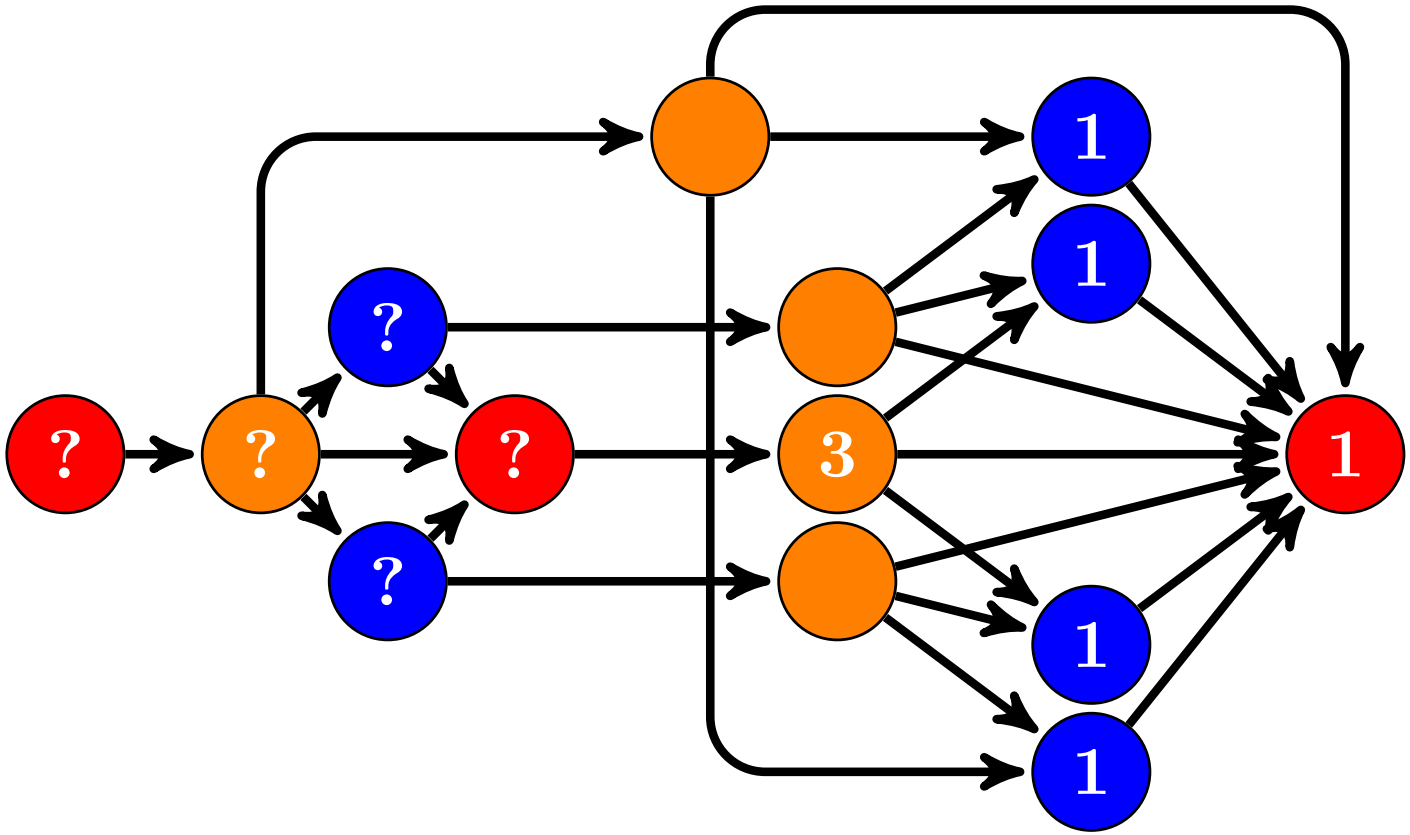
Challenge:

- Choice of a particular MPE for any stage game at any stage
- may alter the set and even the number of stage equilibria at earlier stages

Need to find feasible ESRs

Multiplicity of equibiria

Number of equilibria in the higher stages depends on the selected equilibria

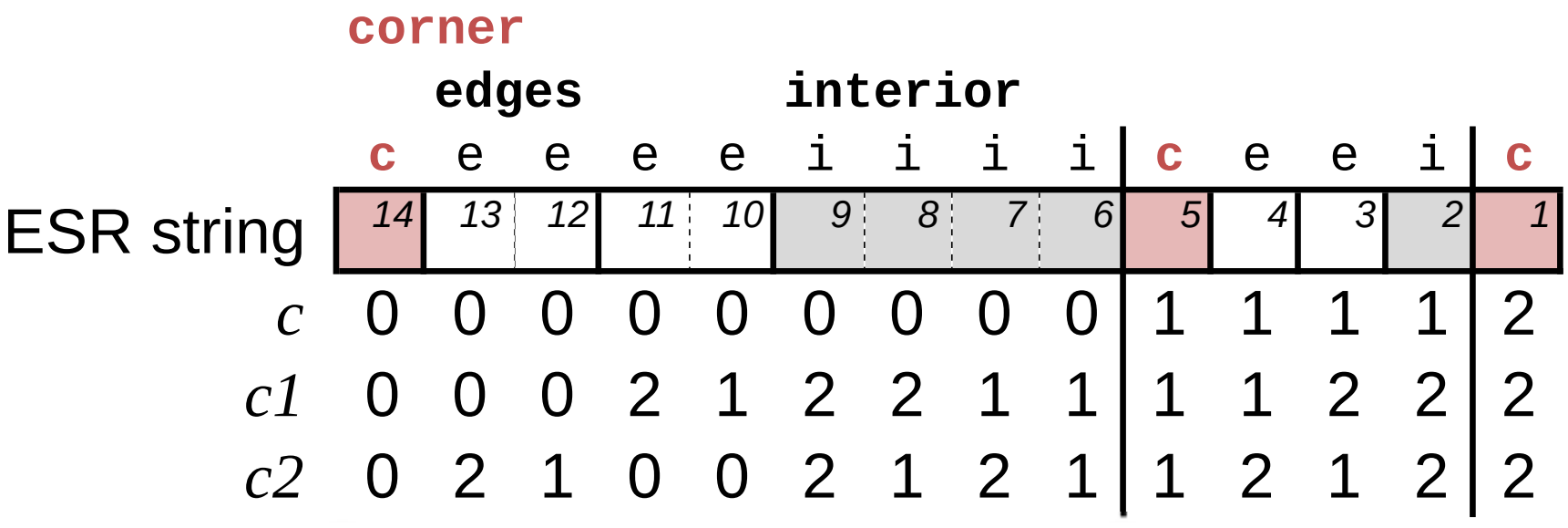


Represent ESR as string of digits

Use numbers in base-K number system with digits $0, 1, .., K-1$

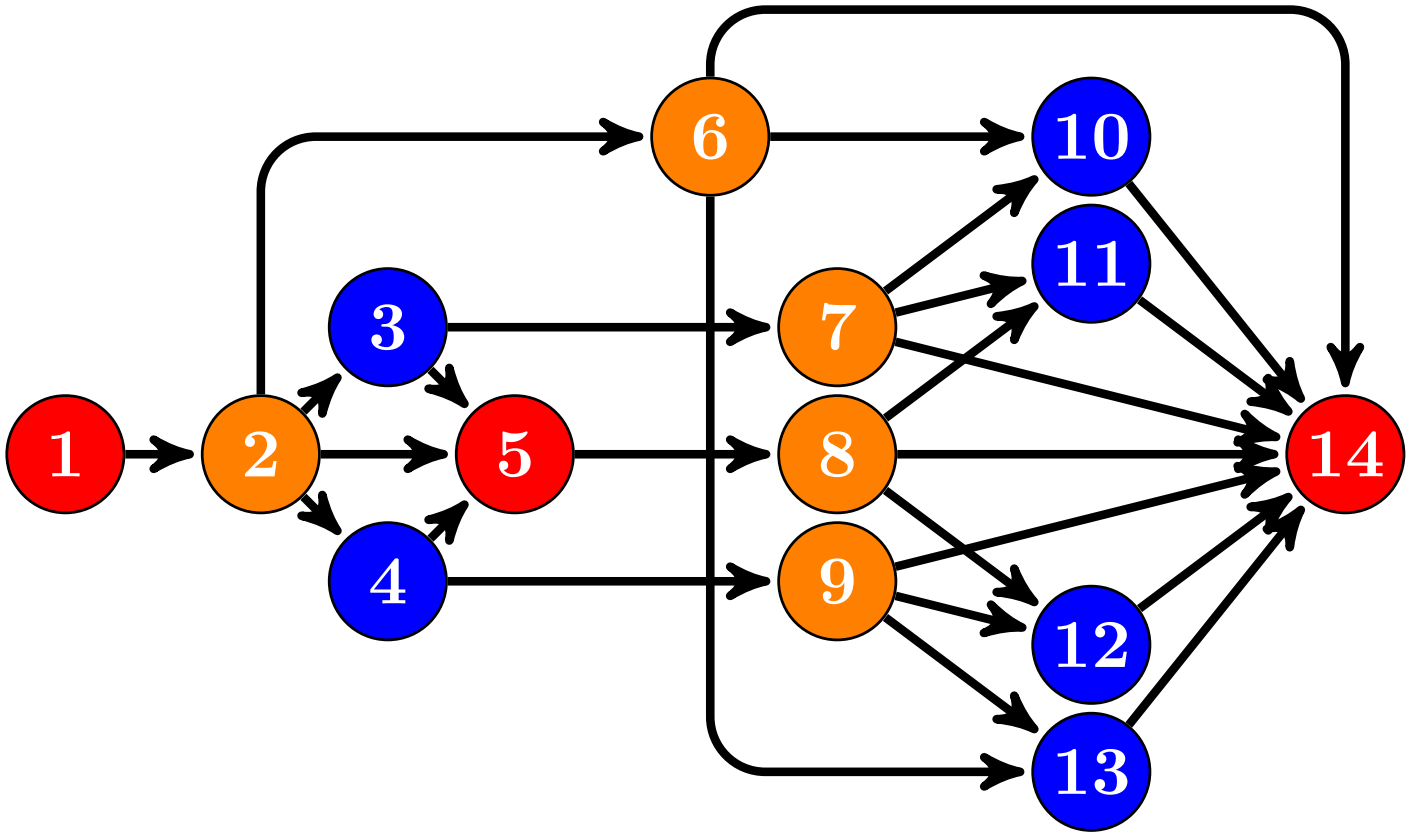
Dependence preserving property:

Any point of the state space may depend on the points to the left (higher digits) and not the points to the right (lower digits)



Digits of the ESR string mapped to the state space

Dependent preserving property



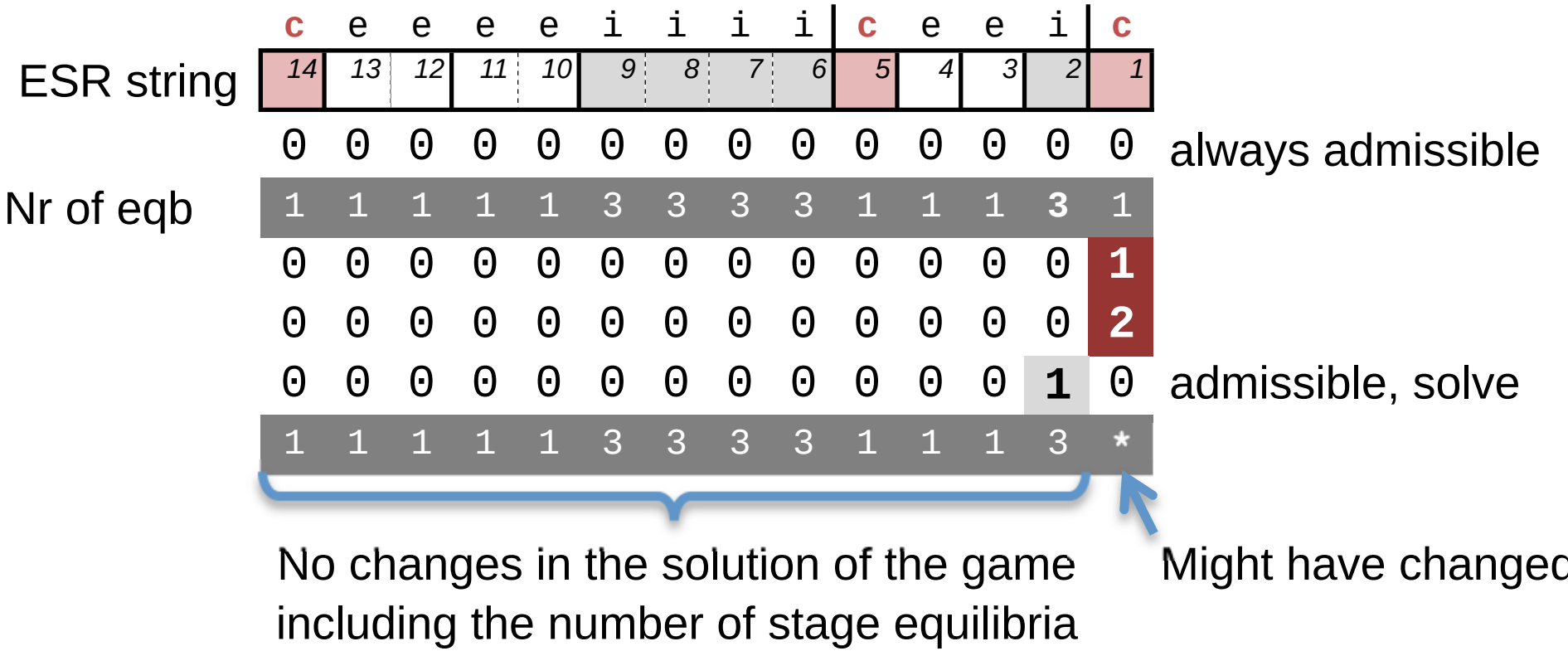
All possible ESR in lexicographic order

	c	e	e	e	e	i	i	i	i	c	e	e	i	c
ESR string	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Lexicograph	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	1	2
	0	0	0	0	0	0	0	0	0	0	0	0	2	0
	0	0	0	0	0	0	0	0	0	0	0	0	2	1
	0	0	0	0	0	0	0	0	0	0	0	0	2	2
	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	0	0	0	0	0	0	0	0	0	0	0	1	0	1
														...
	2	2	2	2	2	2	2	2	2	2	2	2	2	0
	2	2	2	2	2	2	2	2	2	2	2	2	2	1
	2	2	2	2	2	2	2	2	2	2	2	2	2	2

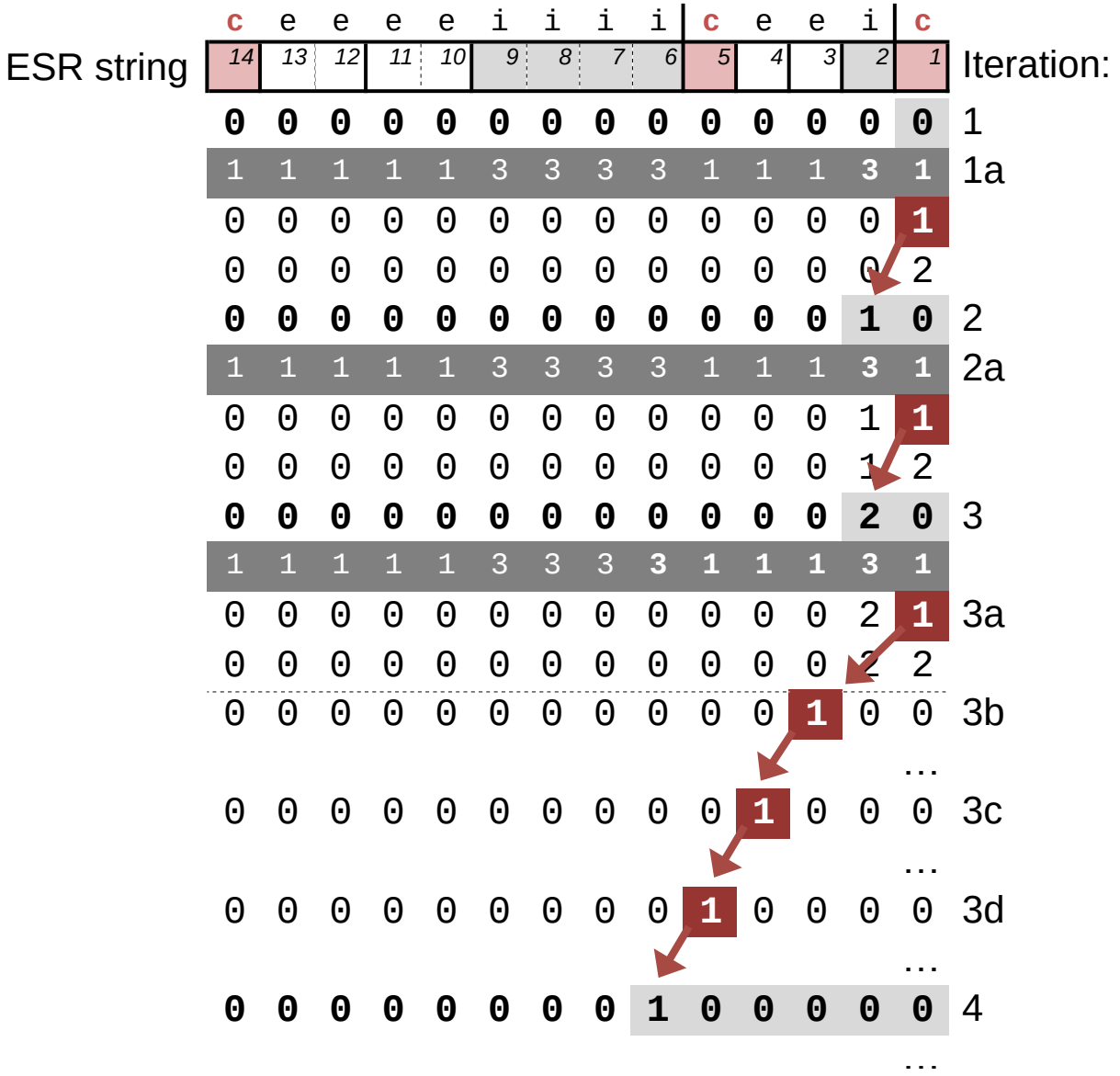
4,782,969

Recalculation of fesibility condition for new ESR

Avoid recalculation of subgames



Jumping over blocks of infeasibles ESRs



Variable base arithmetics

- Replace the base-K ($mod(K)$) arithmetics with variable base arithmetics
- Let (3 1 2) be bases $\rightarrow$
- Allowed digits in the numbers are $\{0, 1, 2\}$, $\{0\}$ and $\{0, 1\}$
- The 3-digit numbers in this system are:

0 0 0	$+1$	$\rightarrow$
0 0 1	$+1$	$\rightarrow$
1 0 0	$+1$	$\rightarrow$
1 0 1	$+1$	$\rightarrow$
2 0 0	$+1$	$\rightarrow$
2 0 1		

RLS Algorithm

1. Set ESR $= (0, \ldots, 0)$
2. Run State Recursion using the current ESR
3. Save the number of equilibria in every stage game as ne(ESR)
4. Add 1 to the ESR in bases ne(ESR) to obtain new feasible ESR
5. Stopping rule: successor function exceeds the maximum number with given number of digits
6. Return to step 2

Main result of the RLS Algorithm

Theorem (Decomposition theorem, strong)

Assume there exists an algorithm that can find all MPE of every stage game of the DDG, and that the number of these equilibria is finite in every stage game.
Then the RLS algorithm finds all MPE of the DDG in a finite number of steps, which equals the total number of MPE.

Main result of the RLS Algorithm

Theorem (Decomposition theorem, weak)

Assume there exists an algorithm that can at least one MPE of every stage game of the DDG, and that the number of these equilibria is finite in every stage game.
Then the RLS algorithm finds some (at least one) MPE of the DDG in a finite number of steps, which does not exceed the total number of MPE.

RLS algorithm: running times

$K = 3$

Simultaneous moves	$n = 3$	$n = 4$
Total number ESRs	4,782,969	3,948,865,611
Number of feasible ESRs	127	46,707
Time used	0.008 sec.	0.334 sec.

RLS algorithm: running times

$K = 3$

Simultaneous moves	$n = 3$	$n = 4$
Total number ESRs	4,782,969	3,948,865,611
Number of feasible ESRs	127	46,707
Time used	0.008 sec.	0.334 sec.

Simultaneous moves	$n = 5$
Total number ESRs	174,449,211,009,120,166,087,753,728
Number of feasible ESRs	192,736,405
Time used	45 min.

RLS algorithm: running times

$K = 3$

Simultaneous moves	$n = 3$	$n = 4$
Total number ESRs	4,782,969	3,948,865,611
Number of feasible ESRs	127	46,707
Time used	0.008 sec.	0.334 sec.

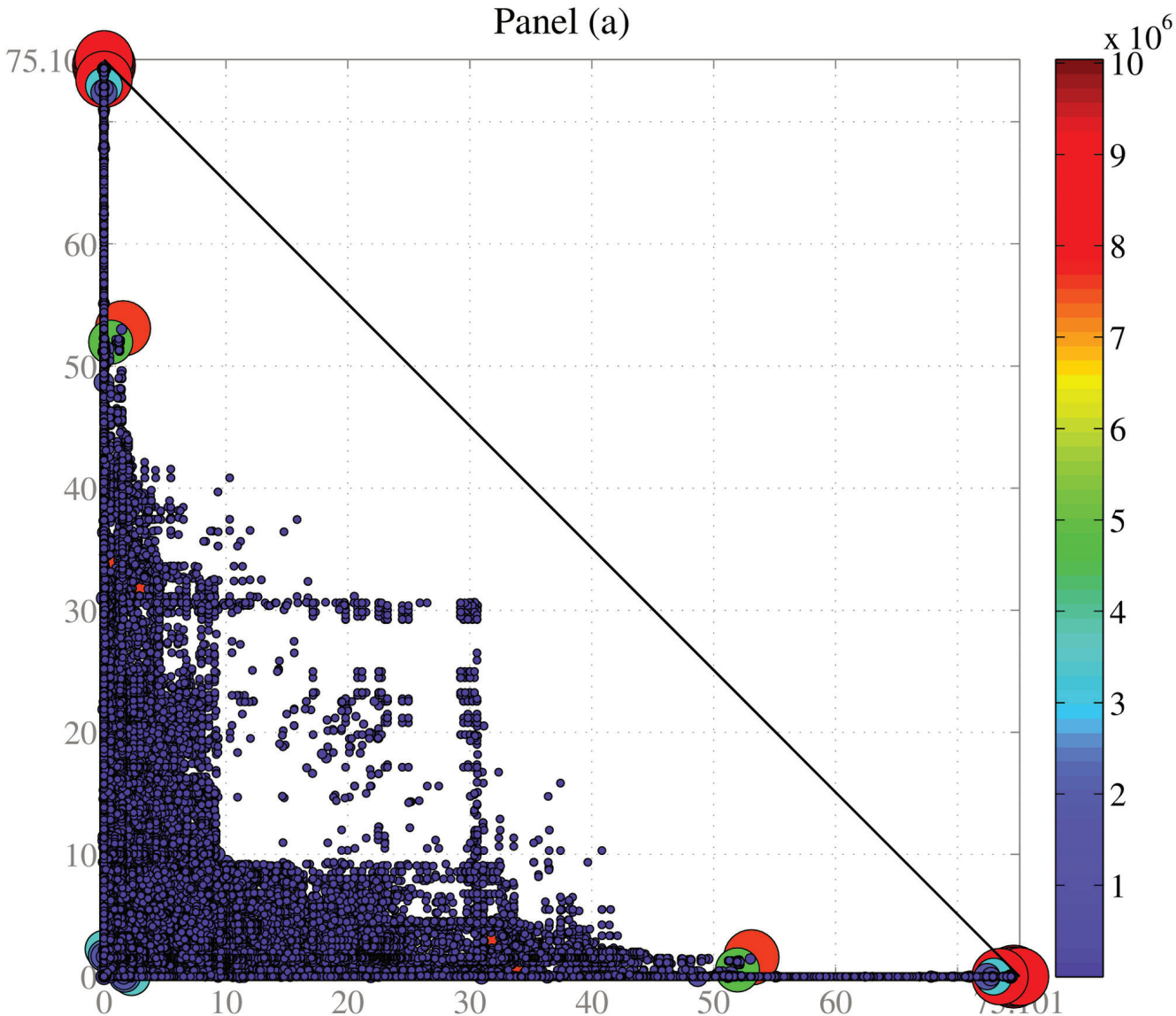
Simultaneous moves		$n = 5$
Total number ESRs	174,449,211,009,120,166,087,753,728	
Number of feasible ESRs		192,736,405
Time used		45 min.

Alternating moves		$n = 5$
Total number ESRs	174,449,211,009,120,166,087,753,728	
Number of feasible ESRs		1
Time used		0.006 sec.

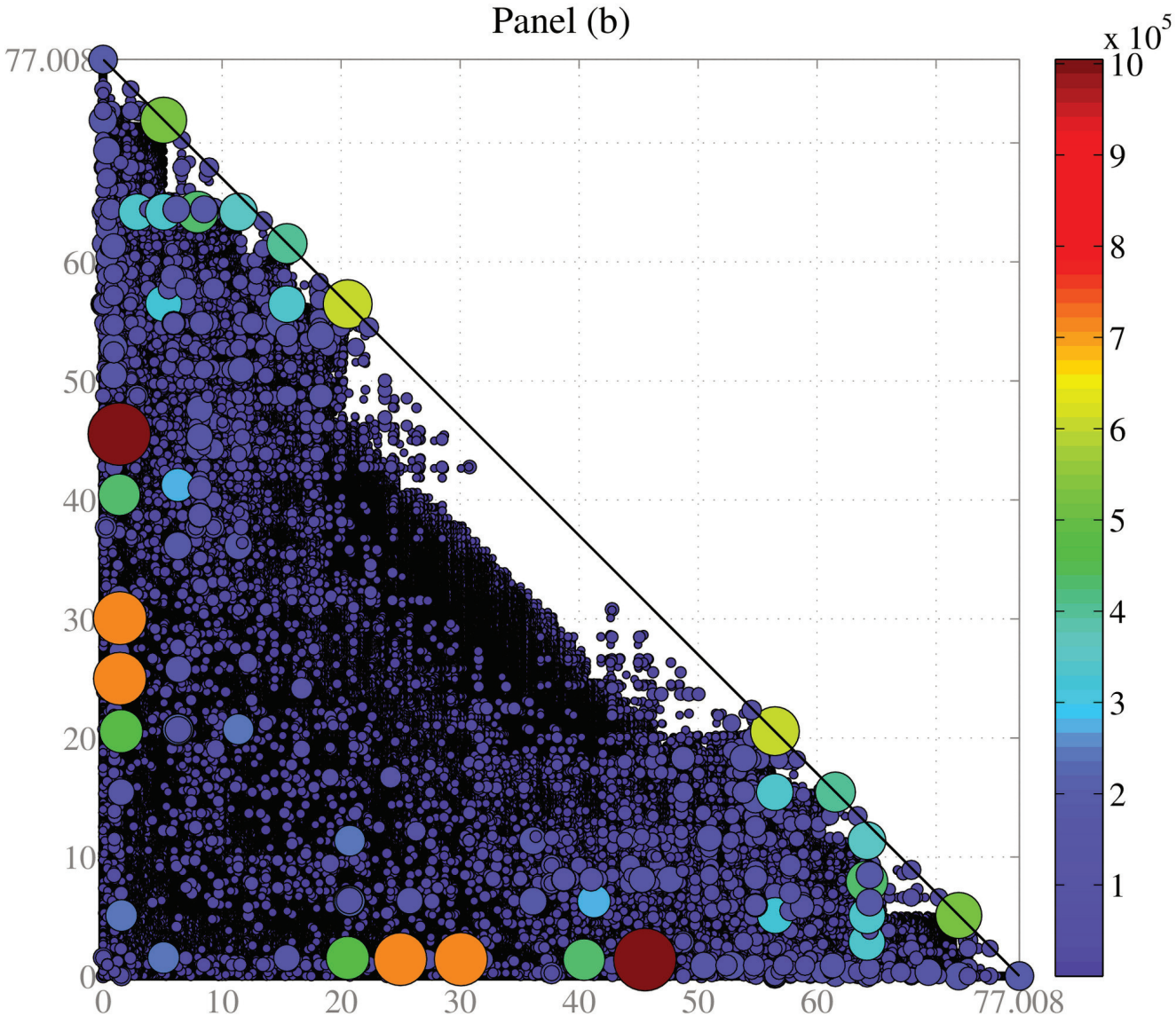
Road map of the talk

1. Original application: collusion of Australian corrugated fibre packaging (CFP) producers
2. State recursion algorithm
3. Recursive lexicographical search (RLS) algorithm
4. Solving the Bertrand investment game using RLS
   - Resolution to the Bertrand investment paradox
   - Sufficient conditions for uniqueness of equilibria
   - Characterization of the set of equilibrium payoffs
   - Efficiency of equilibria
5. Embedding full solution into the nested ML estimator

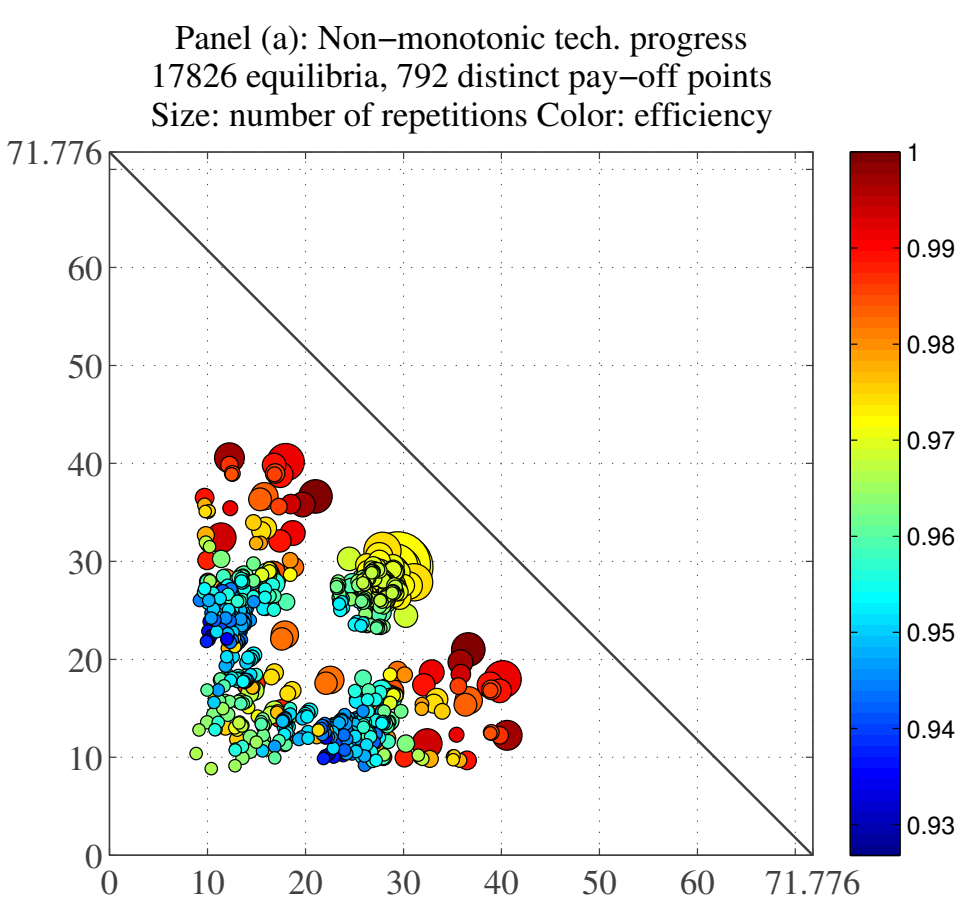
Pay-offs (deterministic tech progress)

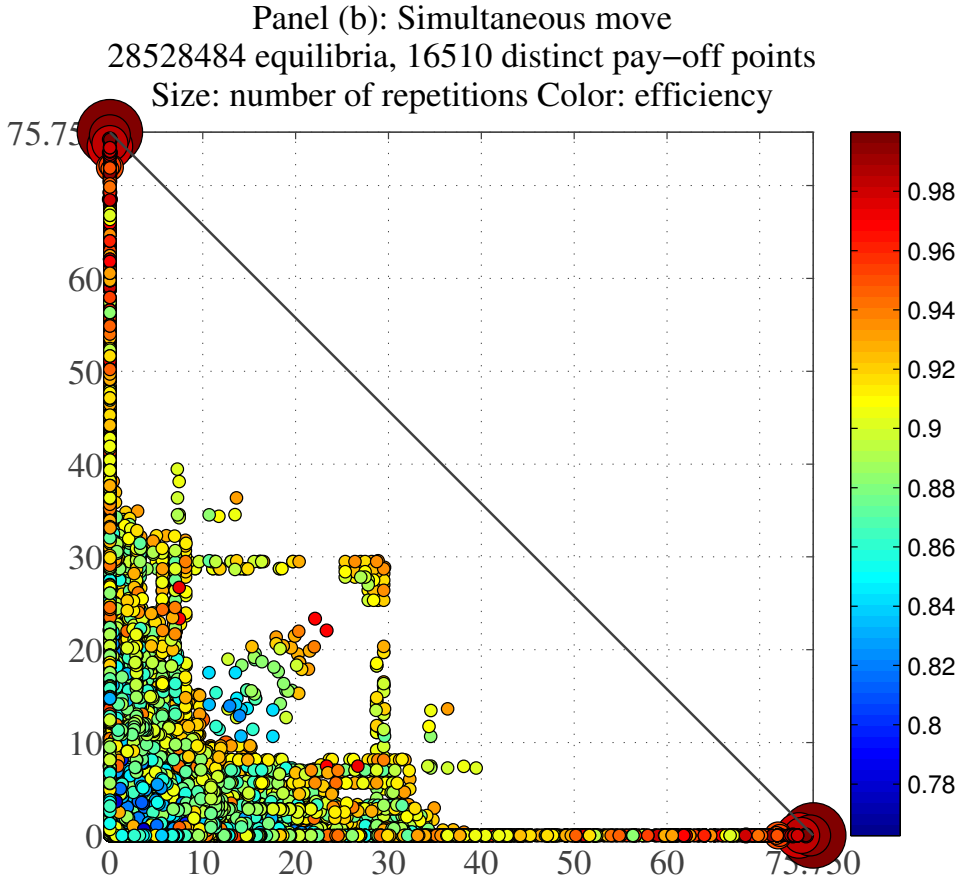


Pay-offs (stochastic tech progress)

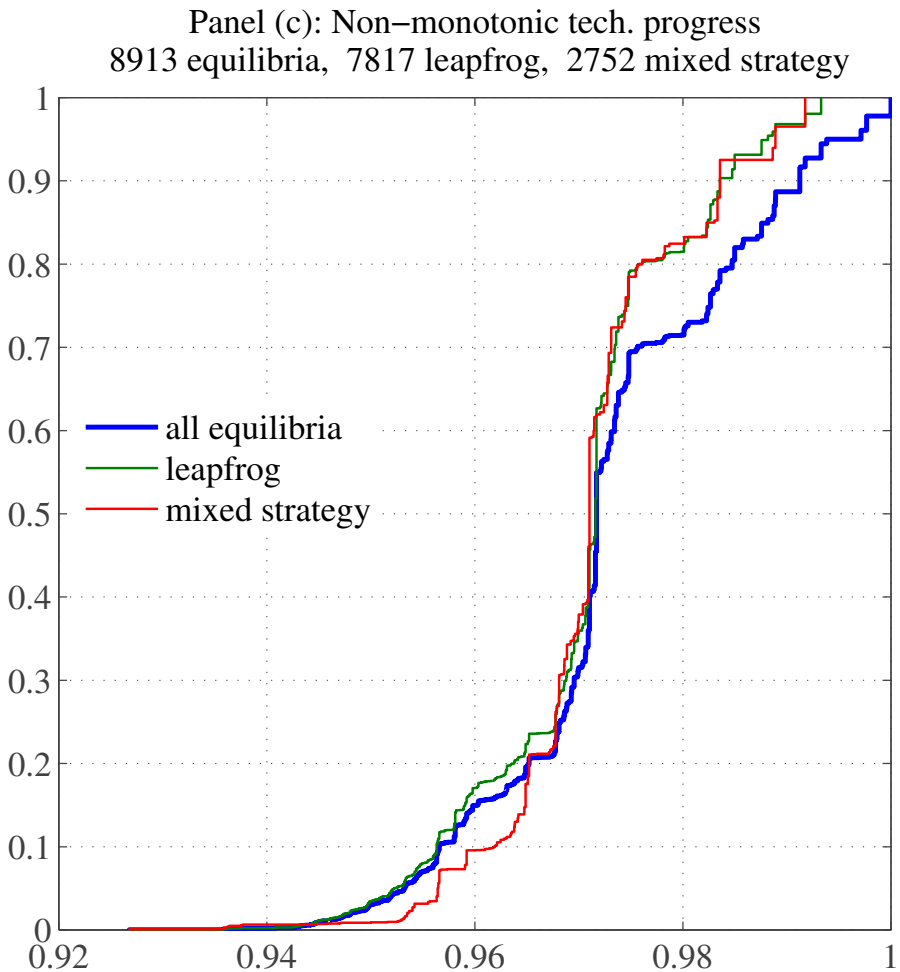


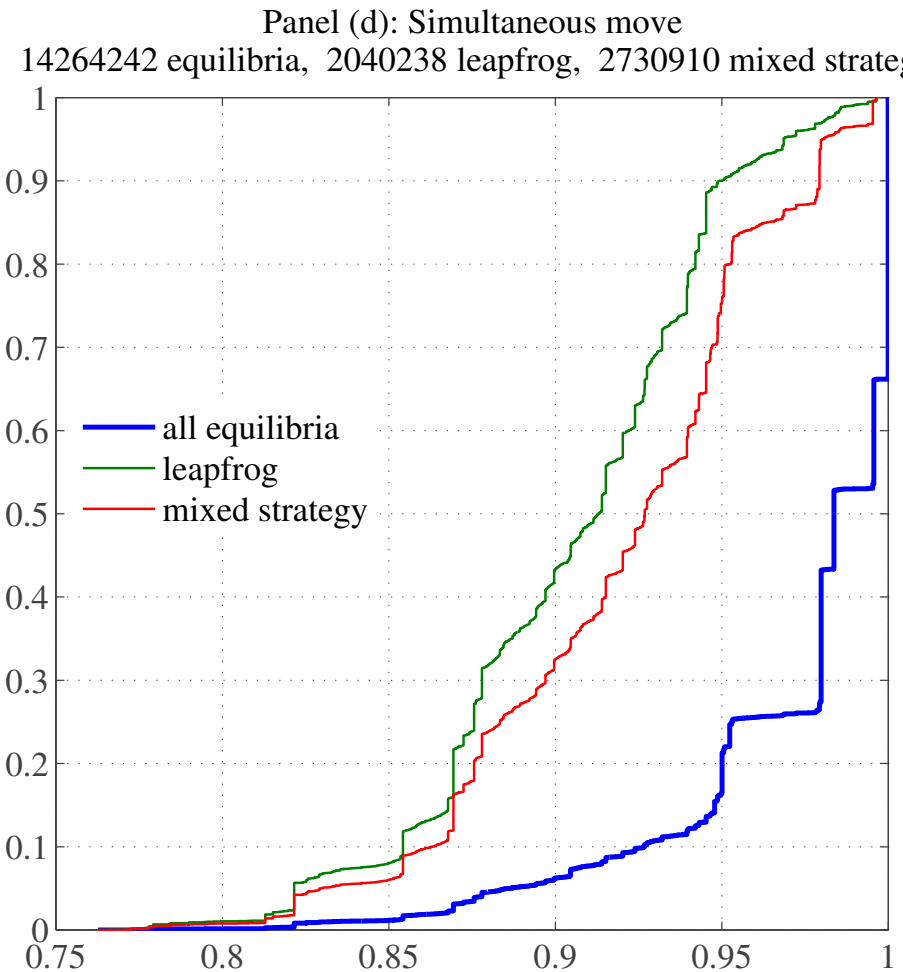
Pay-offs: alternating vs simultaneous move games





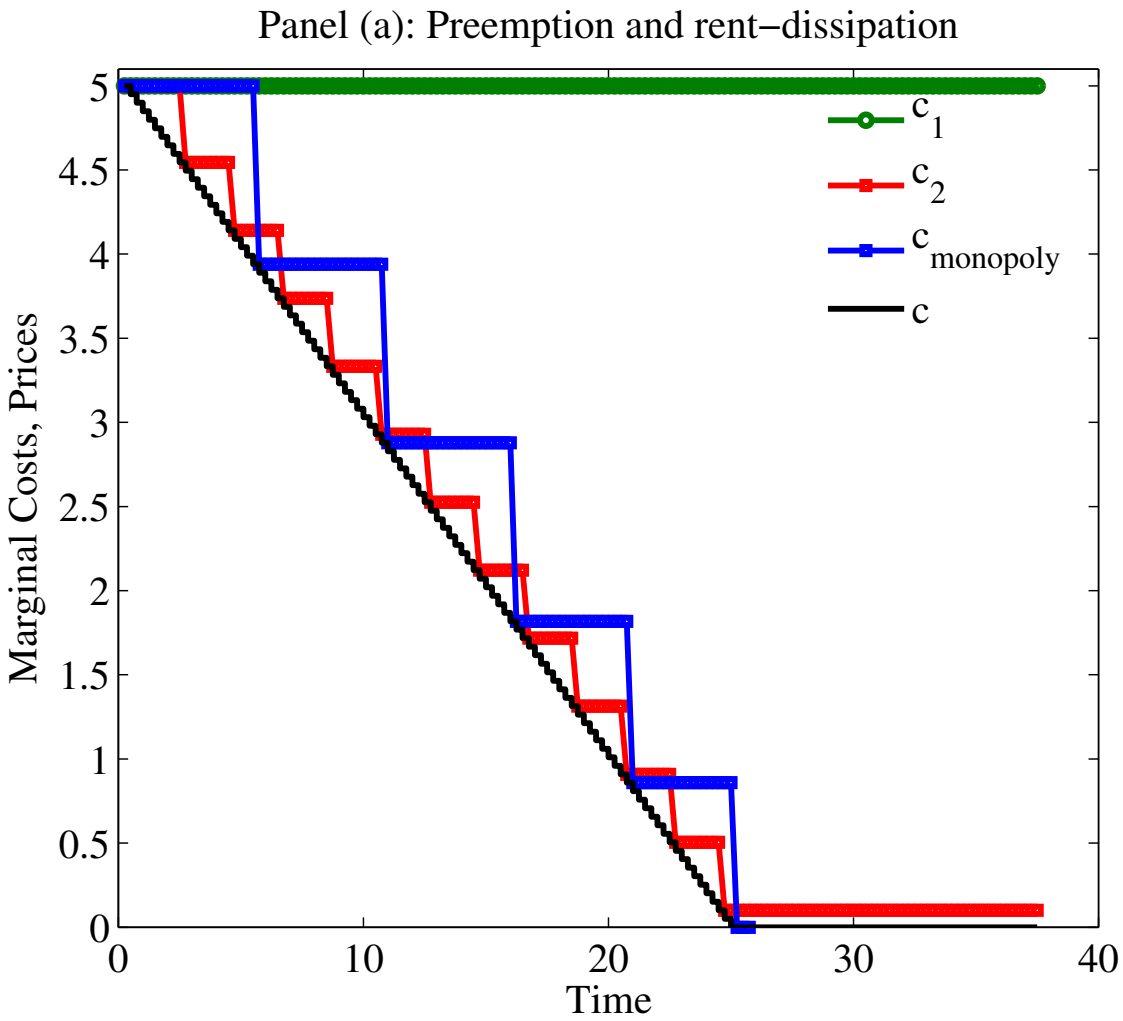
Efficiency: alternating vs simultaneous move games





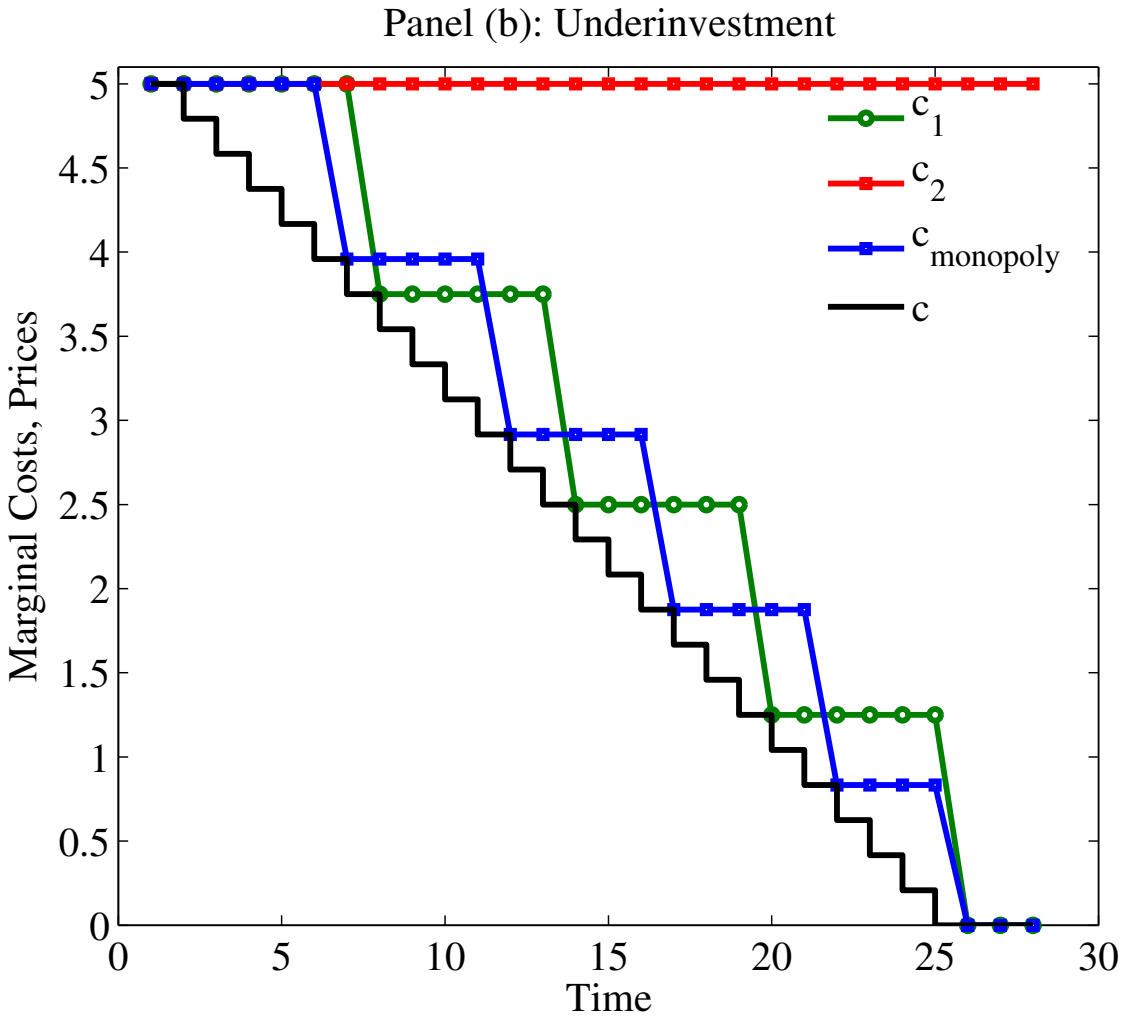
Full preemption and rent dissipation

Confirm R&S the result with high K and small dt



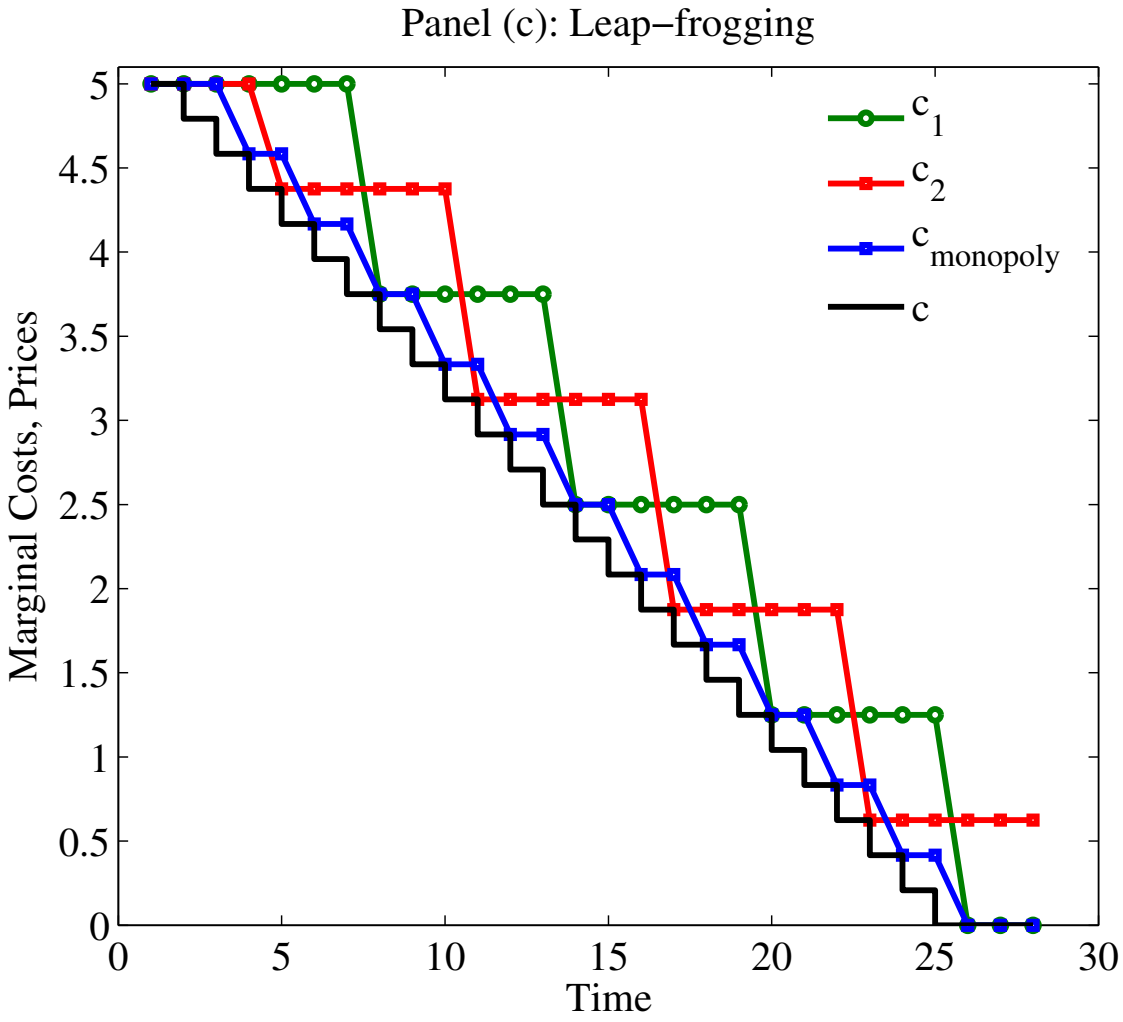
Underinvestment

Rent-dissapation is not a general outcome - disappears when K is low relative dt



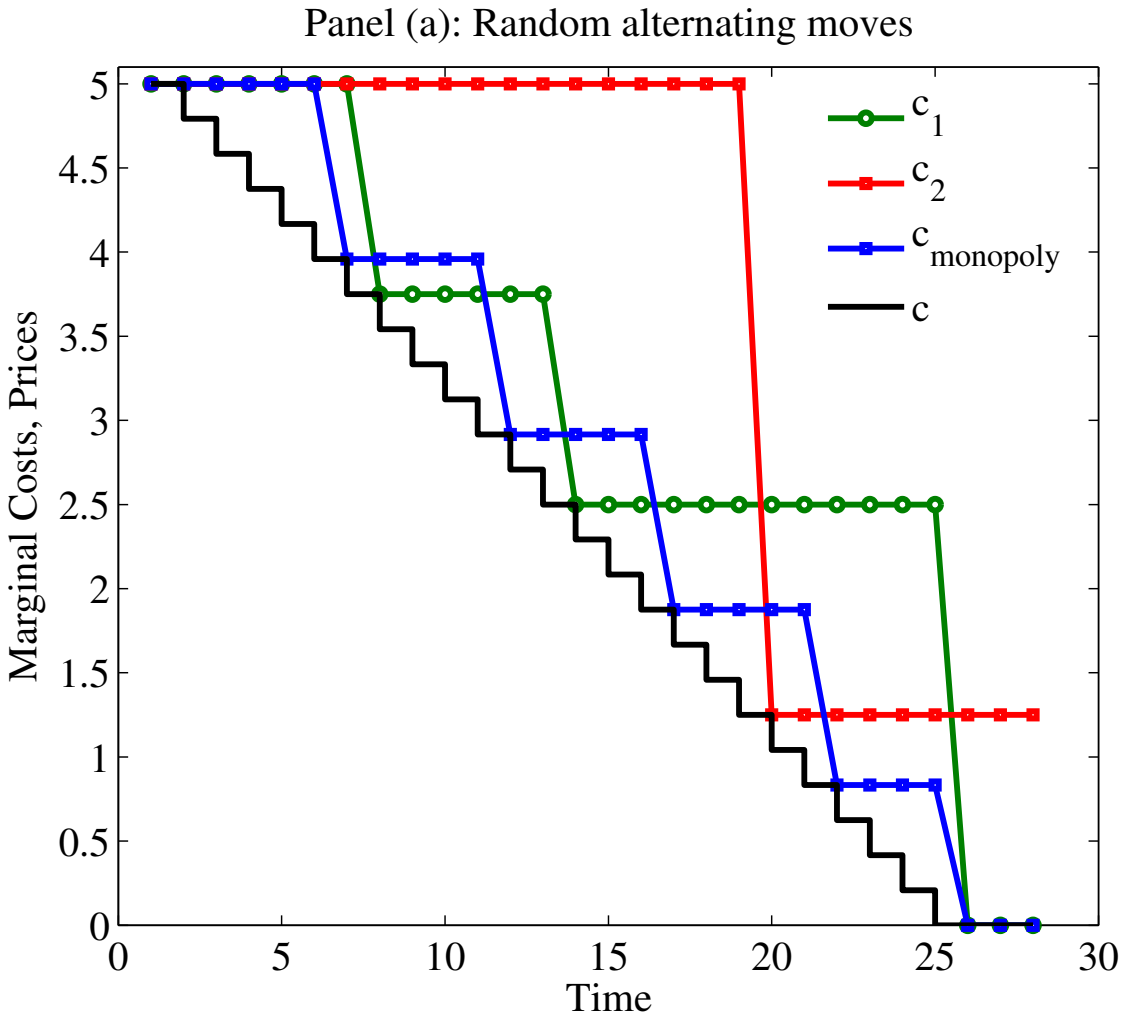
Leap-frogging

Preemption is not the general outcome - disappears when K is even lower



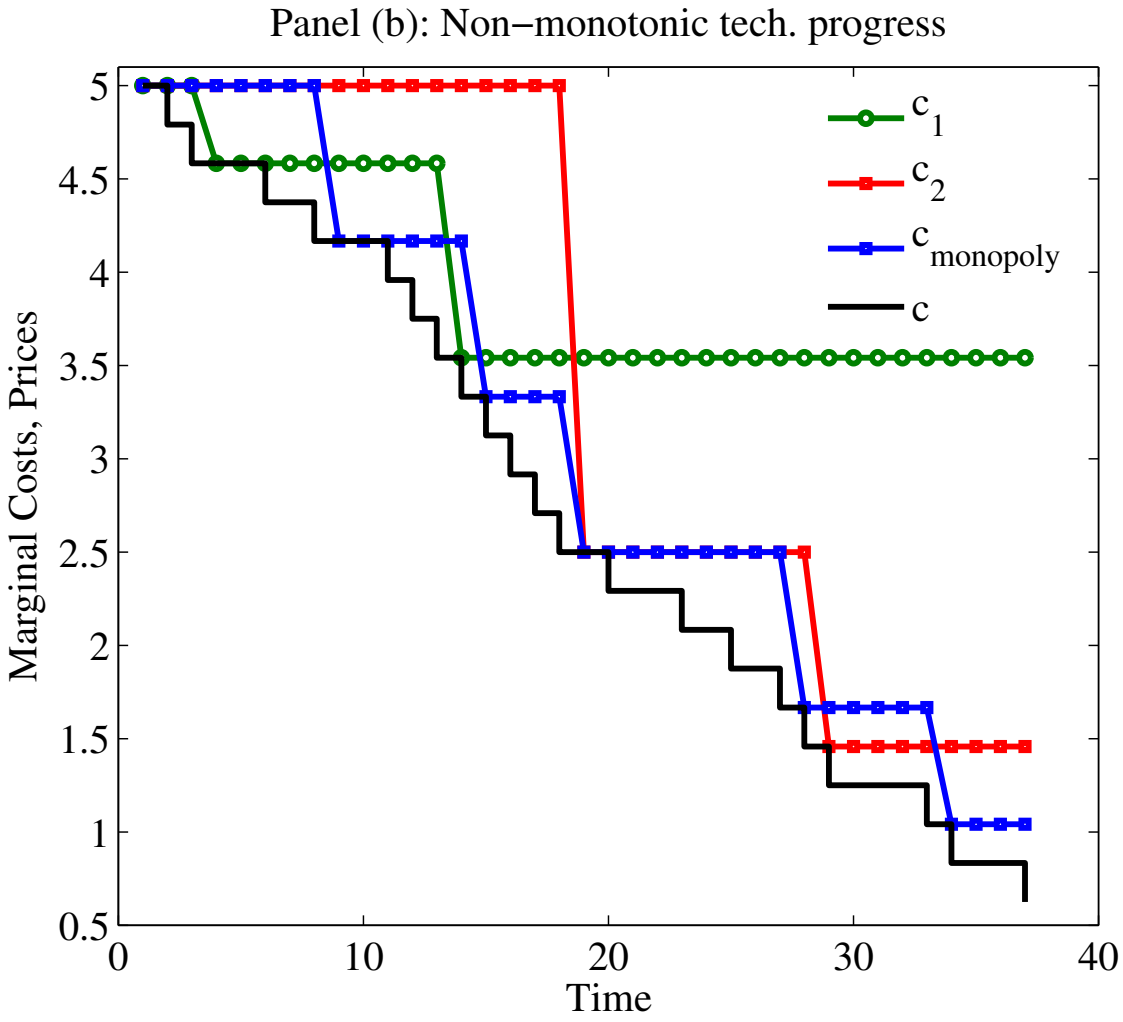
Random alternation $\rightarrow$ Leapfrogging

Riordan and Salant's result is not robust



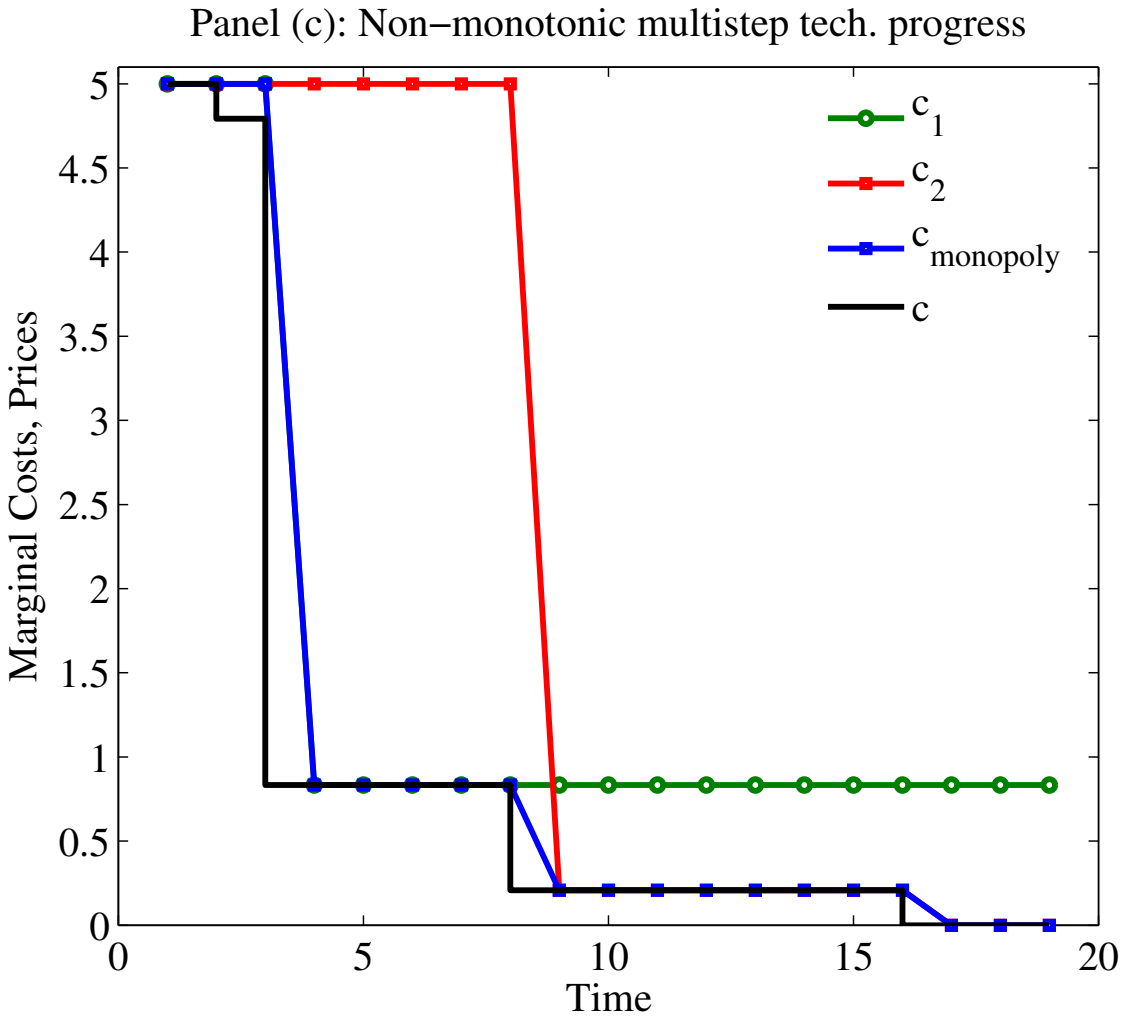
Random onestep technology → Leapfrogging

Riordan and Salant's result is not robust



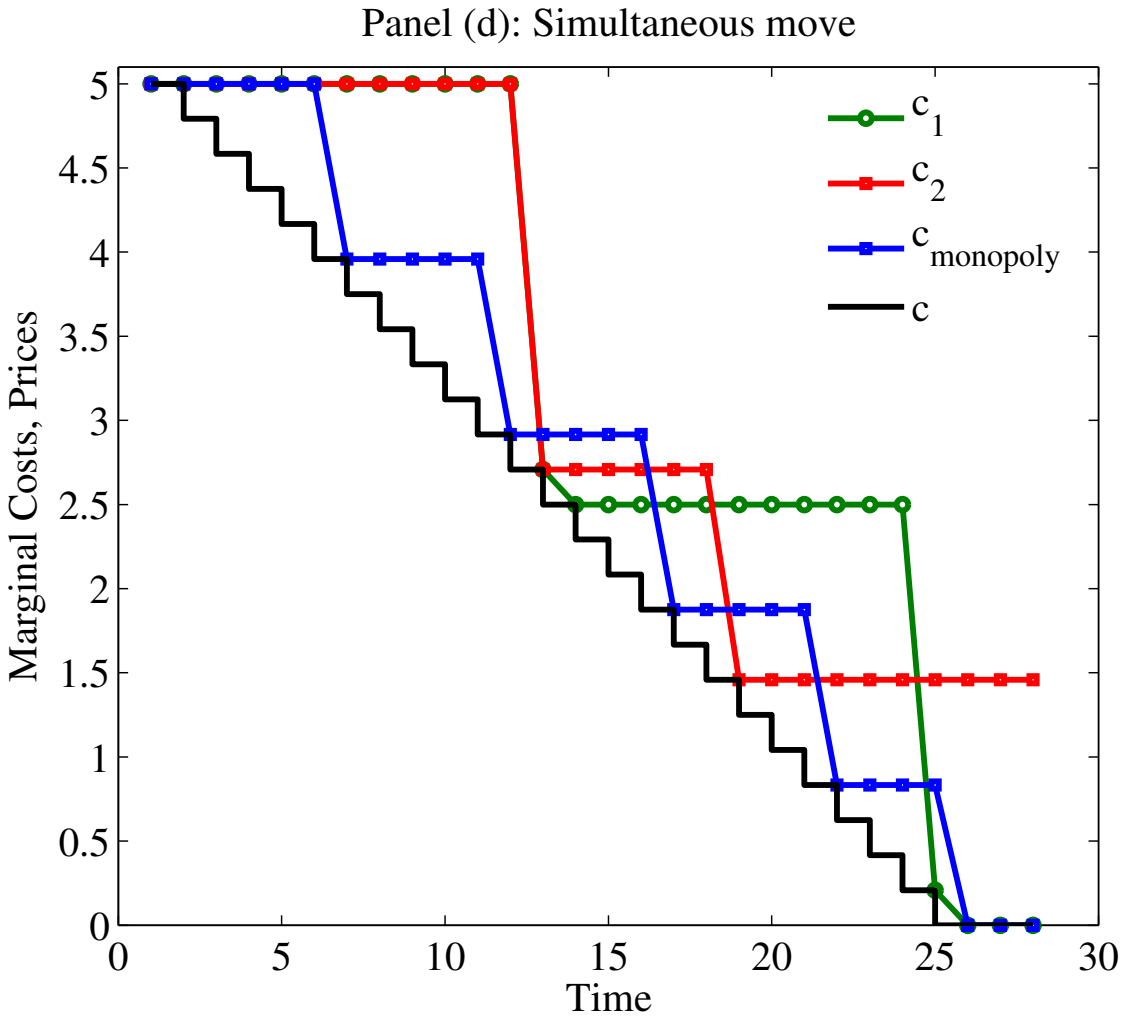
Random multistep technology → Leapfrogging

Riordan and Salant's result is not robust



Simultaneous moves: Leapfrogging

Riordan and Salant's conjecture is wrong



Road map of the talk

1. Original application: collusion of Australian corrugated fibre packaging (CFP) producers
2. State recursion algorithm
3. Recursive lexicographical search (RLS) algorithm
4. Solving the Bertrand investment game using RLS
5. Embedding full solution into the nested ML estimator
   - Nested RLS
   - RLS as Tree Traversal
   - Branch-N-Bound example

Nested Recursive Lexicographical Search (NRLS)

1. **Outer loop**
   Maximization of the likelihood function w.r.t. to structural parameters θ

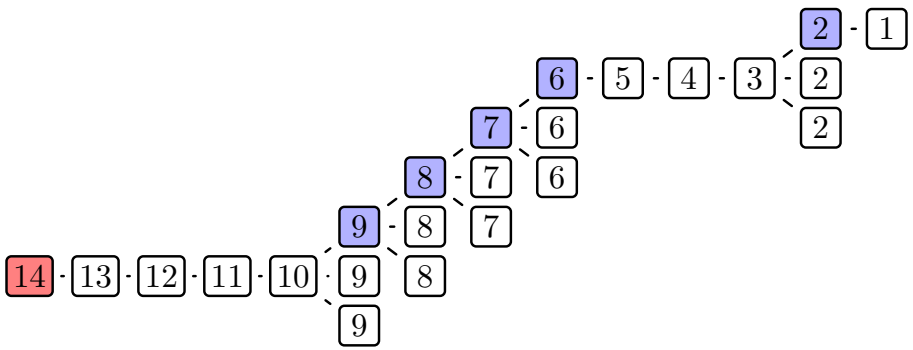
$$\theta^{ML} = \arg\max_{\theta} \mathcal{L}(Z, \theta)$$

2. **Inner loop**
   Maximization of the likelihood function w.r.t. equilibrium selection

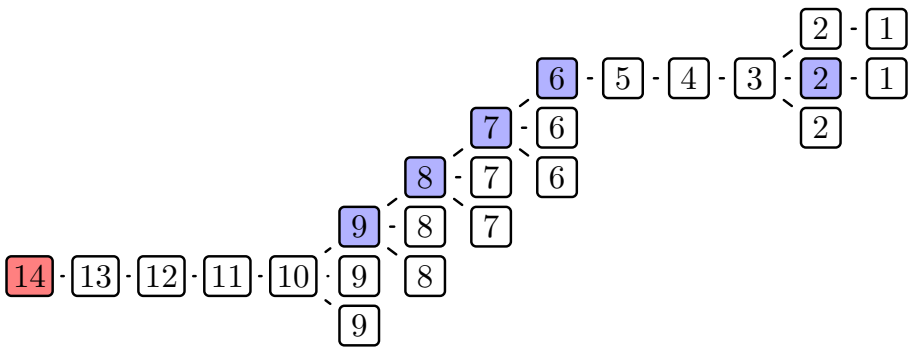
$$\mathcal{L}(Z, \theta) = \max_{(\mathbf{P}^{\ell}(\theta), \mathbf{V}^{\ell}(\theta) \in SOL(\Psi, \theta)} \frac{1}{M} \sum_{i=1}^{N} \sum_{m=1}^{M} \sum_{t=1}^{T} \log P_i^{\ell}(\bar{a}_i{}^{mt} | \bar{\mathbf{x}}^{mt}; \theta)$$

Max of a function on a discrete set organized into RLS tree

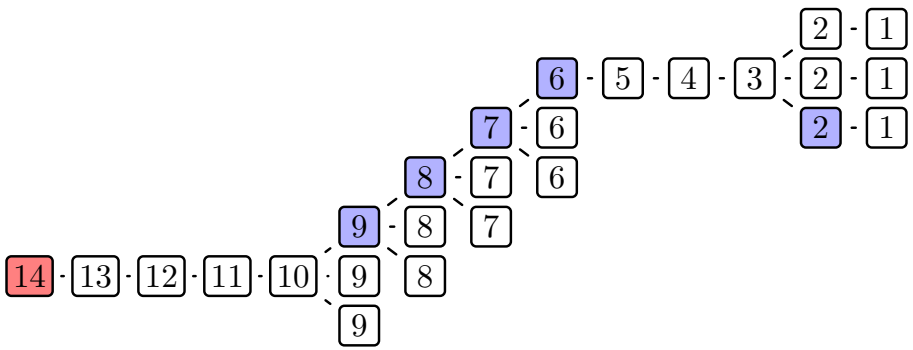
RLS Tree Traversal, step 1



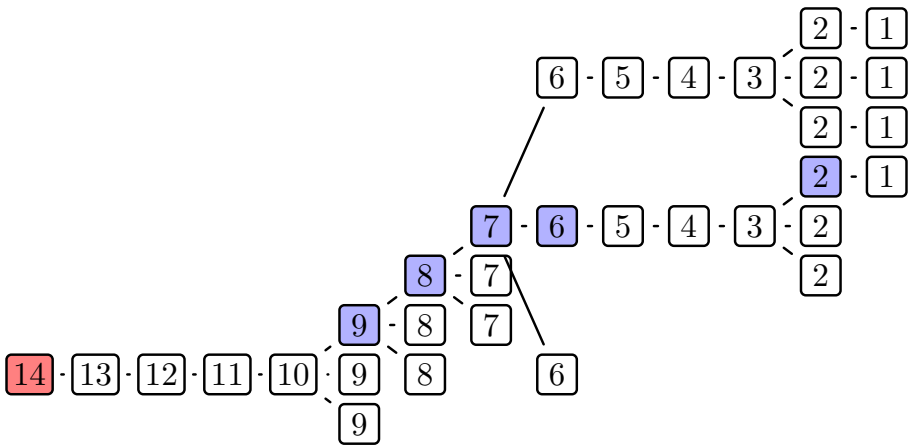
RLS Tree Traversal, step 2



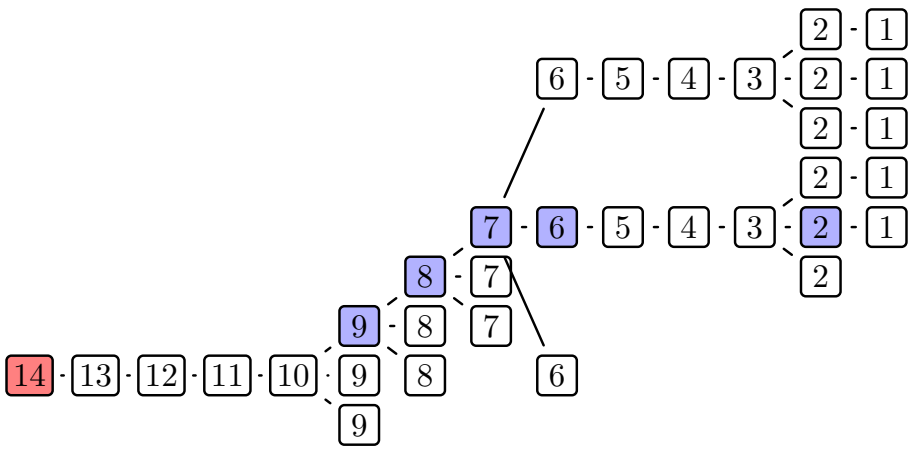
RLS Tree Traversal, step 3



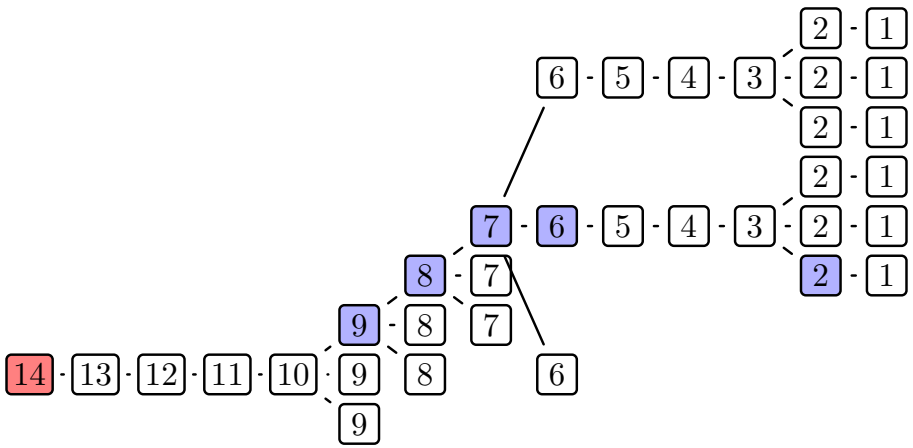
RLS Tree Traversal, step 4



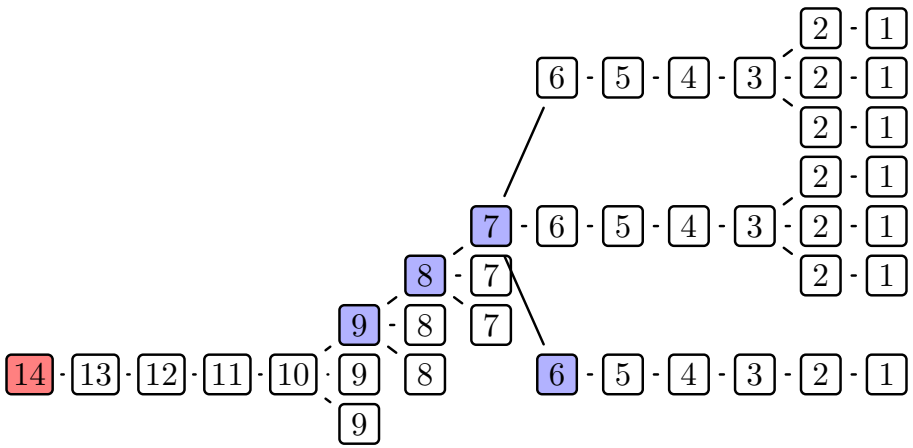
RLS Tree Traversal, step 5



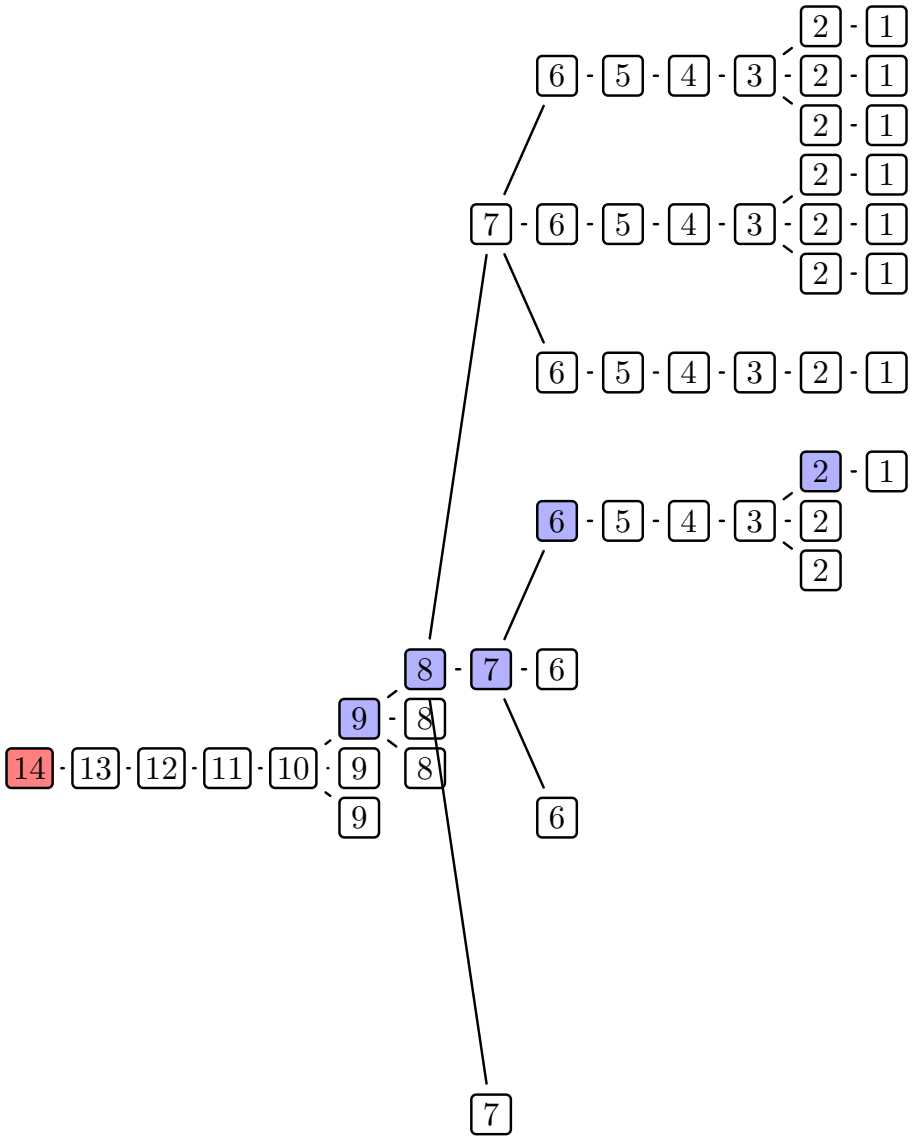
RLS Tree Traversal, step 6



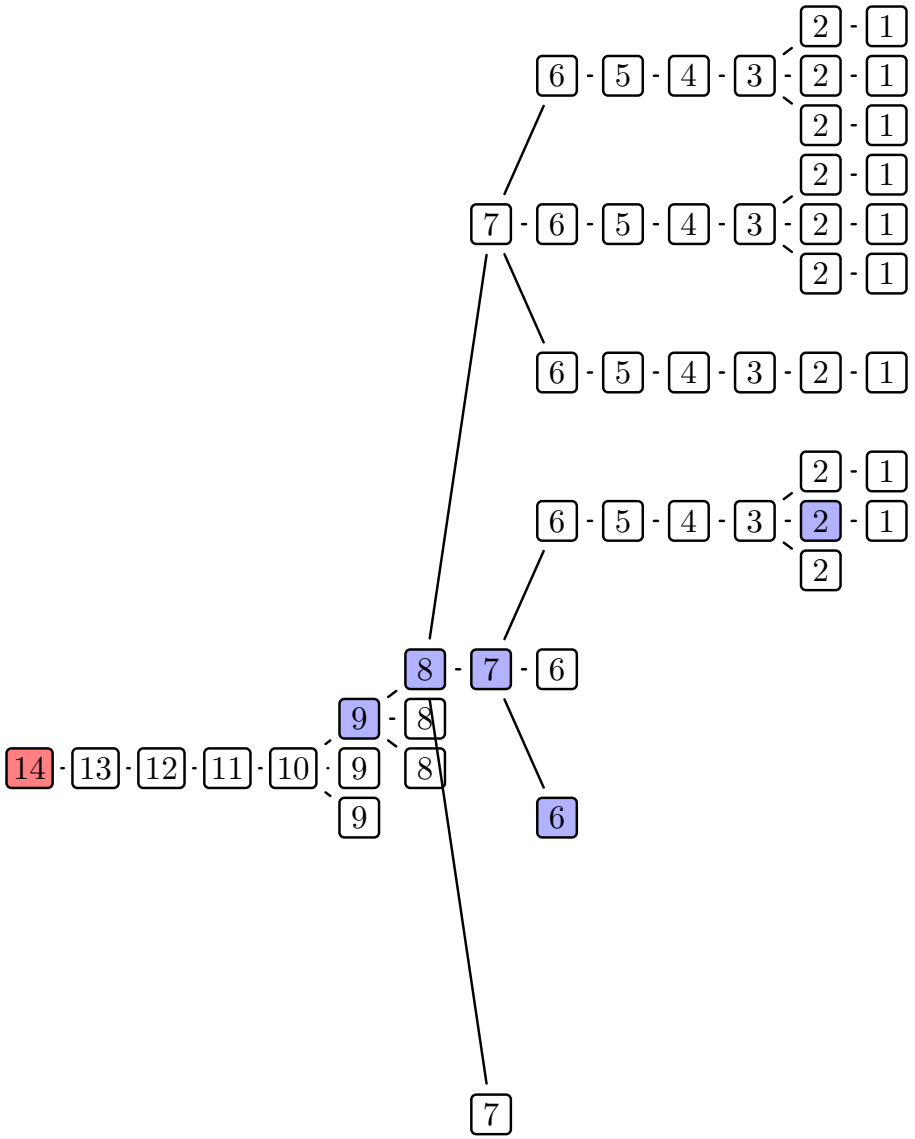
RLS Tree Traversal, step 7



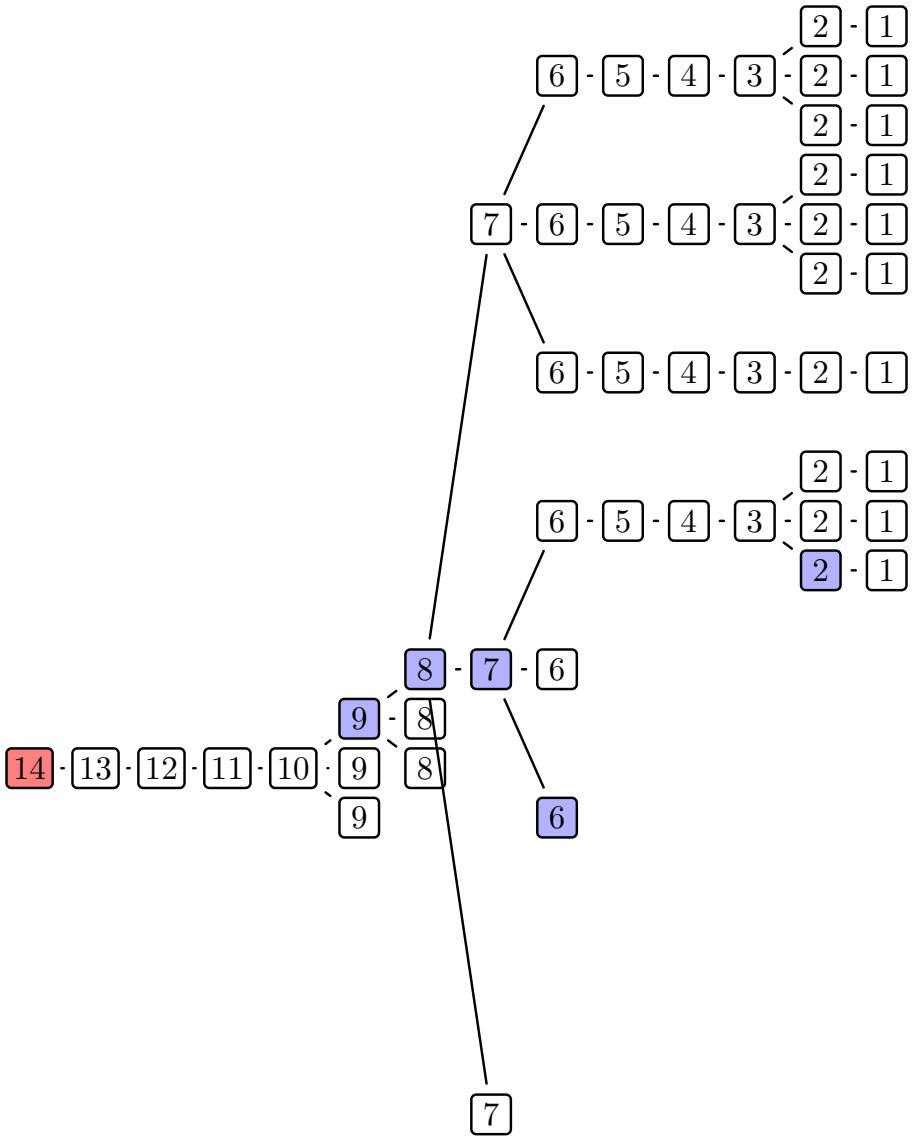
RLS Tree Traversal, step 8



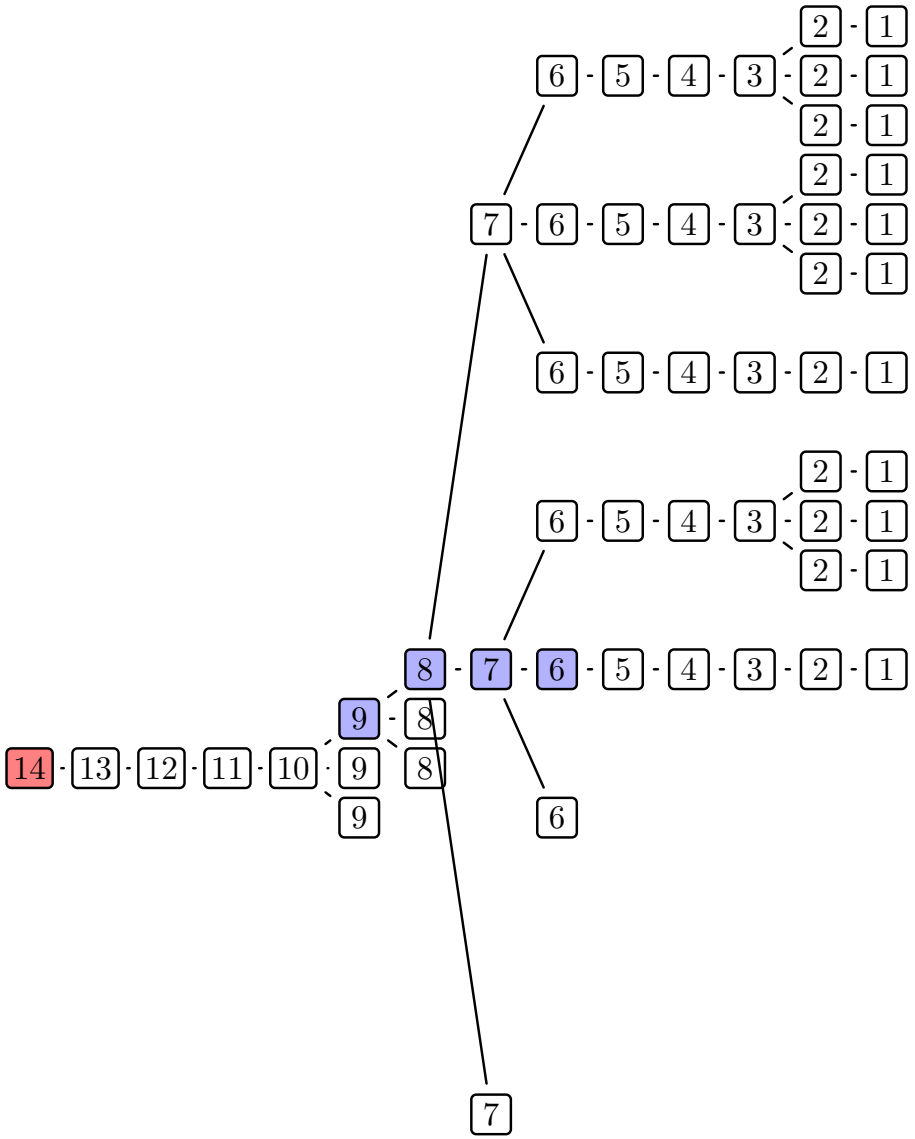
RLS Tree Traversal, step 9



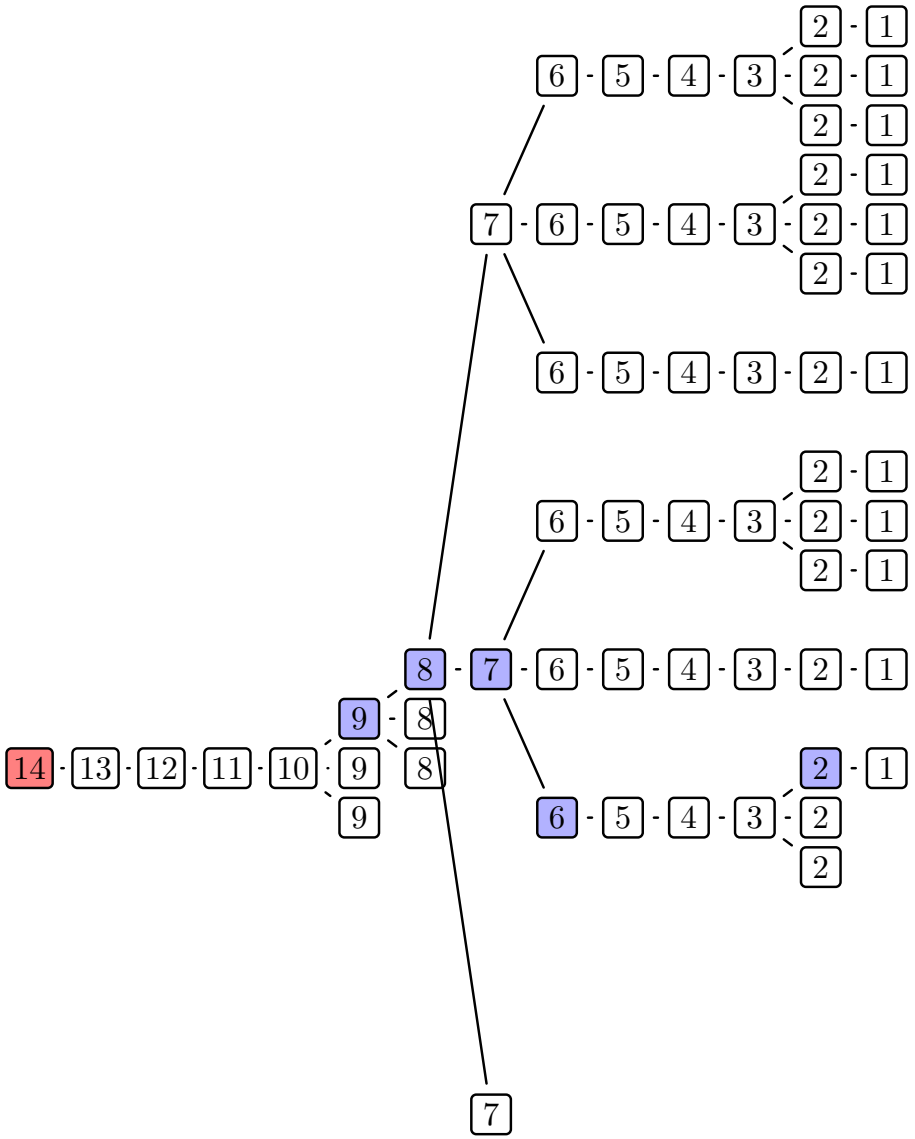
RLS Tree Traversal, step 10



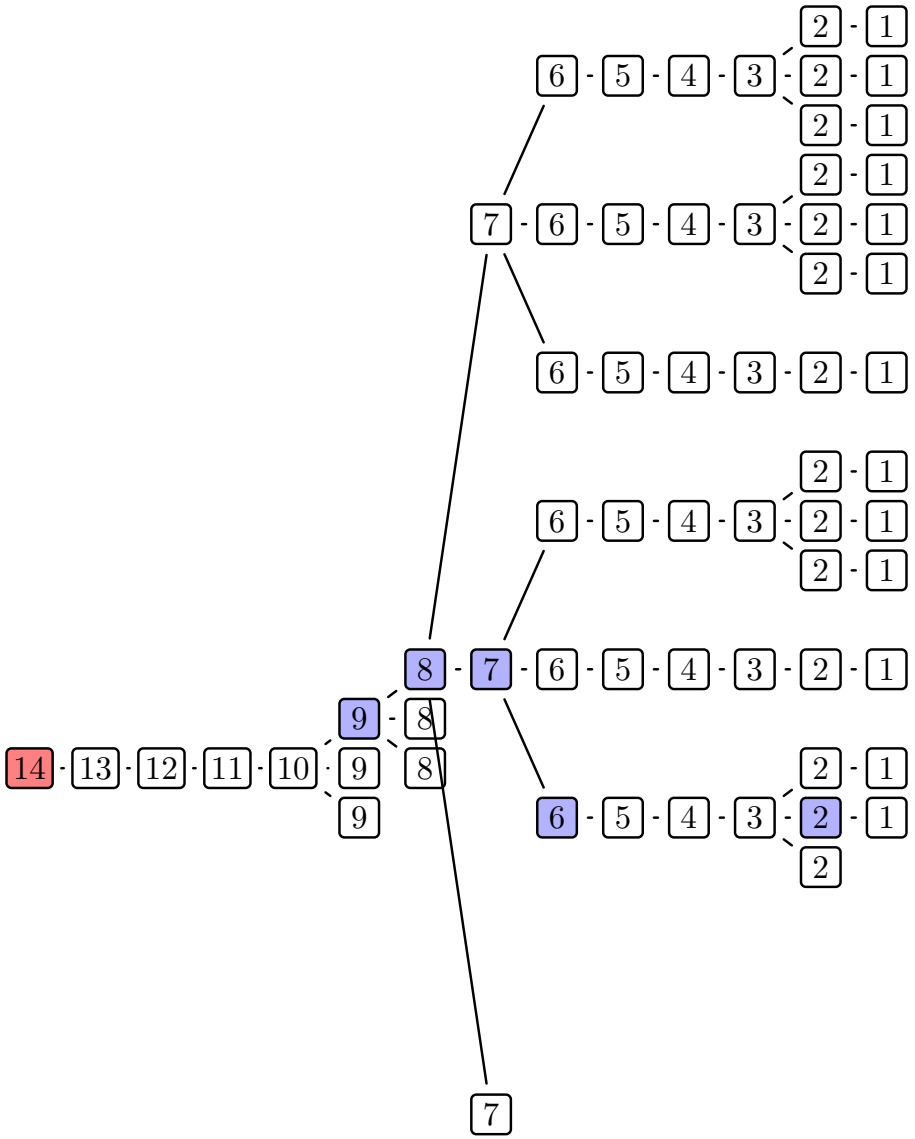
RLS Tree Traversal, step 11



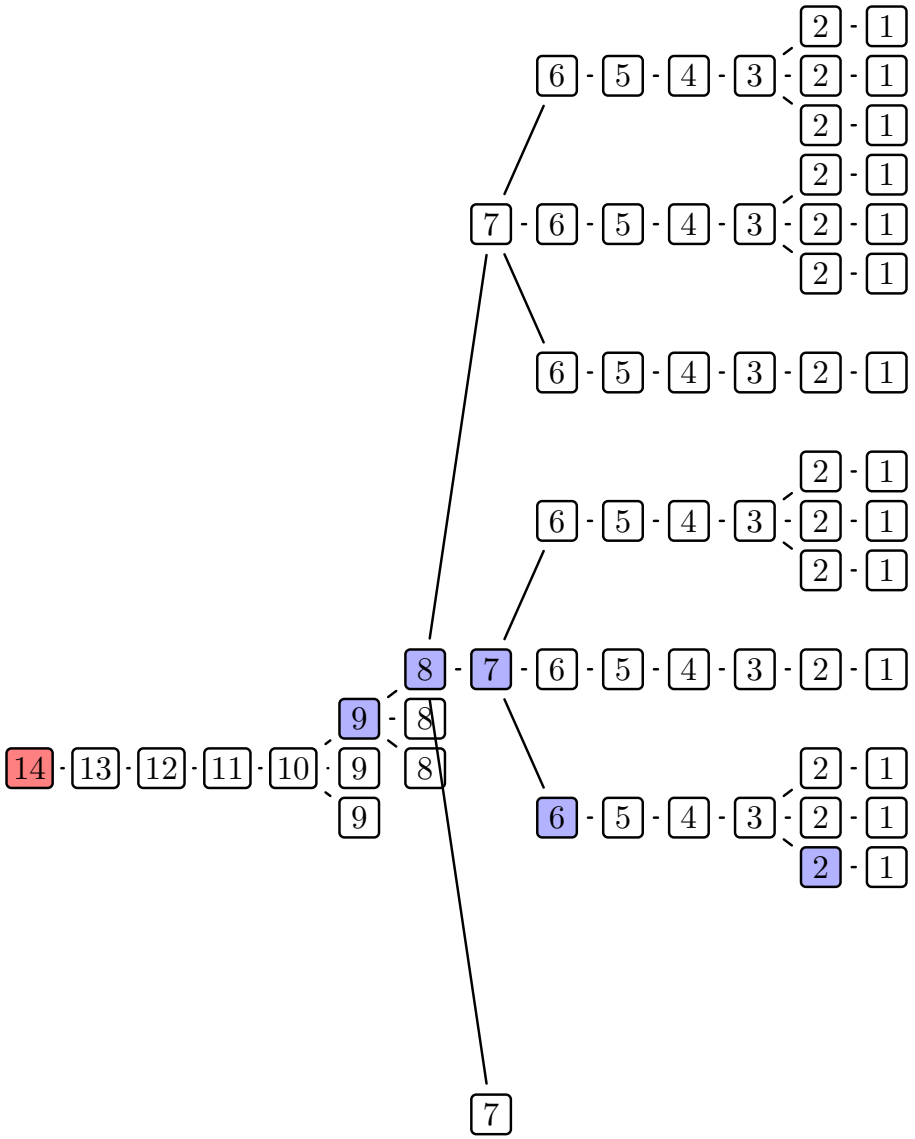
RLS Tree Traversal, step 12



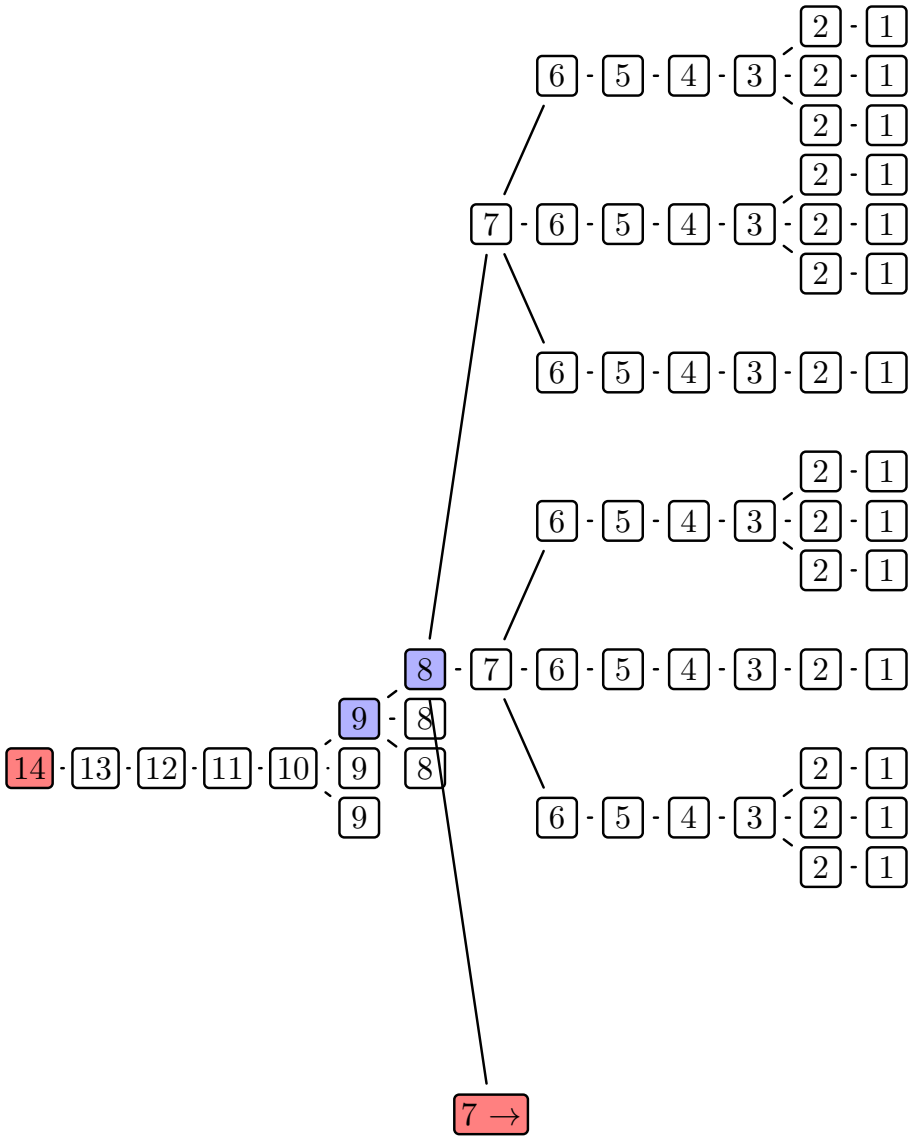
RLS Tree Traversal, step 13



RLS Tree Traversal, step 14



RLS Tree Traversal, step 15



Branch and bound (BnB) method

Land and Doig, 1960 *Econometrica*

- Old method for solving discrete programming problems

1. Form a tree of subdivisions of the set of admissible plans
2. Specify a bounding function representing the best attainable objective on a given subset
3. Dismiss the subsets of the plans where the bound is below the current best attained value of the objective

- There are several flavors of BnB method, differences in implementation
- There are several extensions to the BnB method

Example: Travelling Salesman Problem (TSP)

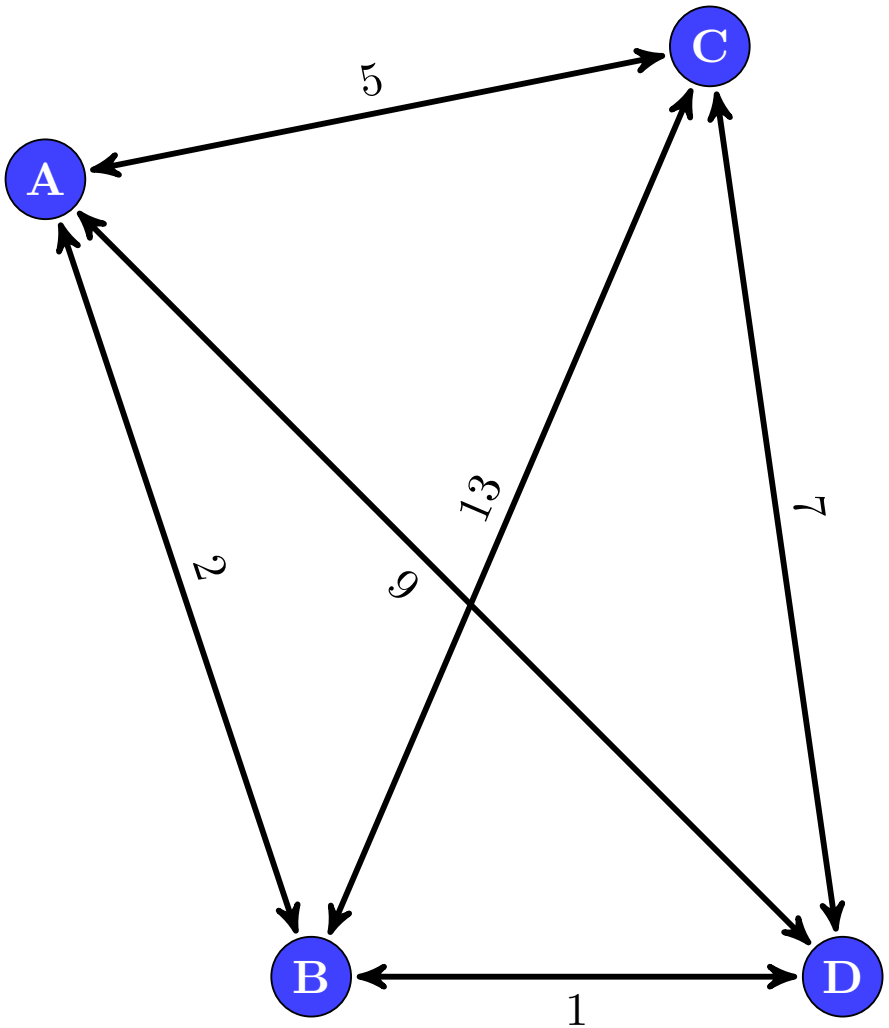
- Classic problem in integer programming
- NP-hard $\leftrightarrow$ Every problem in NP can be reduced in polynomial time to TSP
- Several version exist, we look at a couple

Definition (Classic TSP)

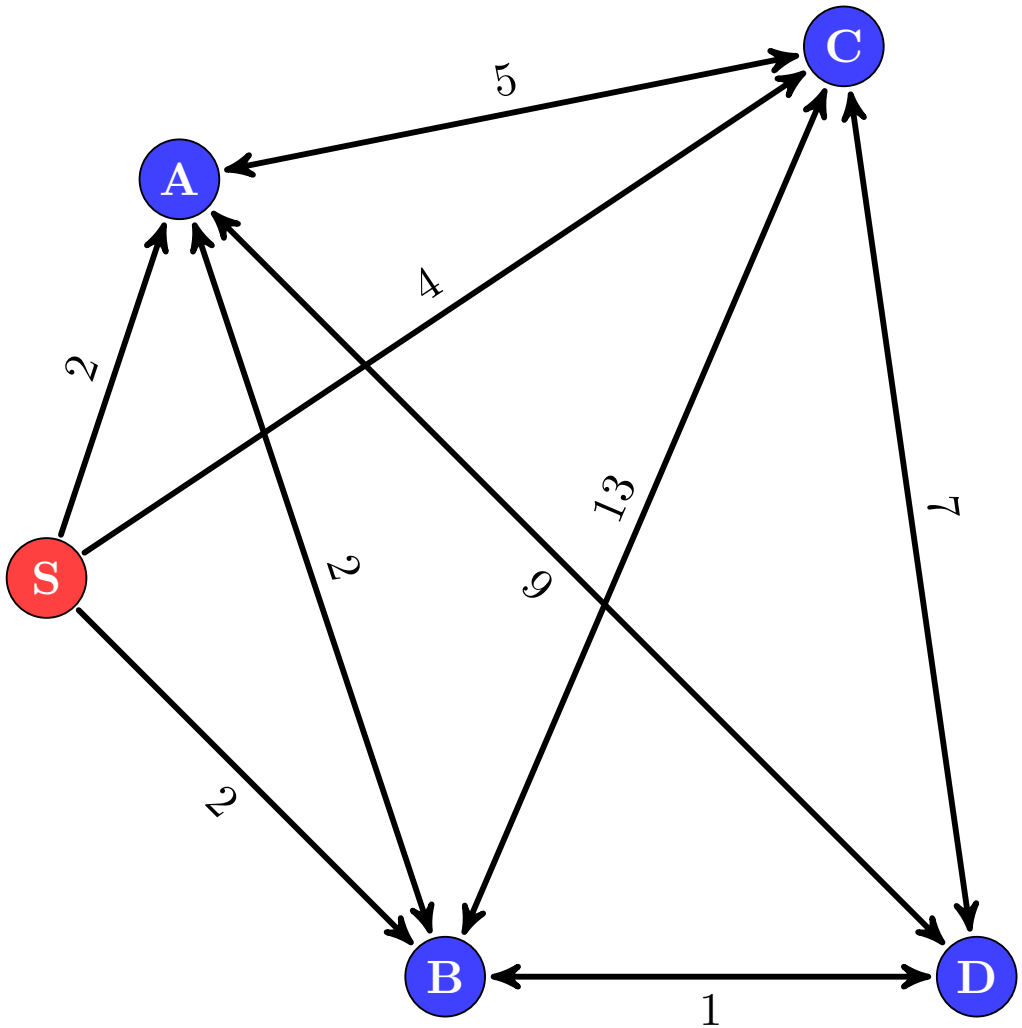
A travelling salesman has to visit every city c in a set C one and only one time, and return to the city visited first. What is the shortest path?

Papadimitriou, 1977 *Theoretical Computer Science*

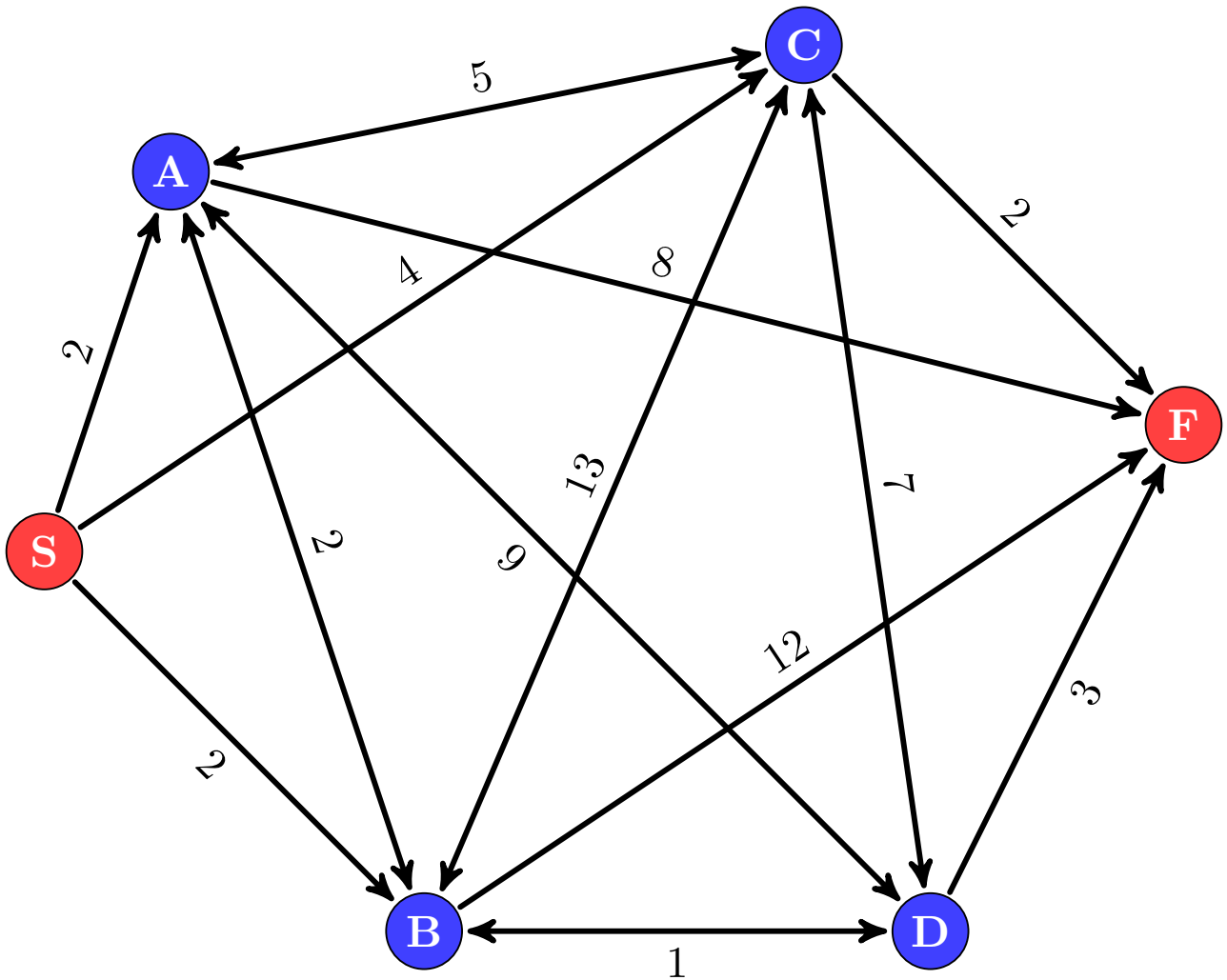
TSP: four cities to visit



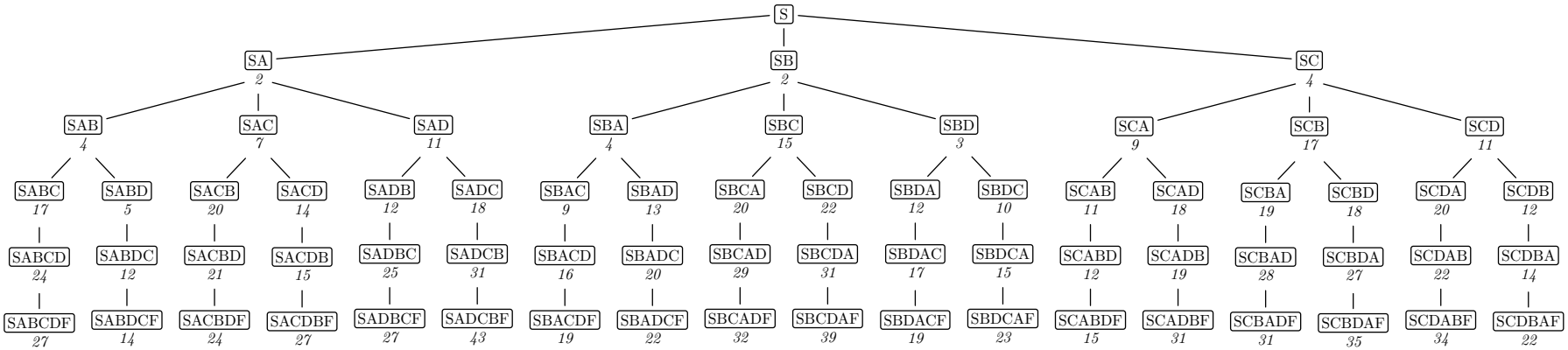
TSP: can start at three cities



TSP: what is the shortest path?

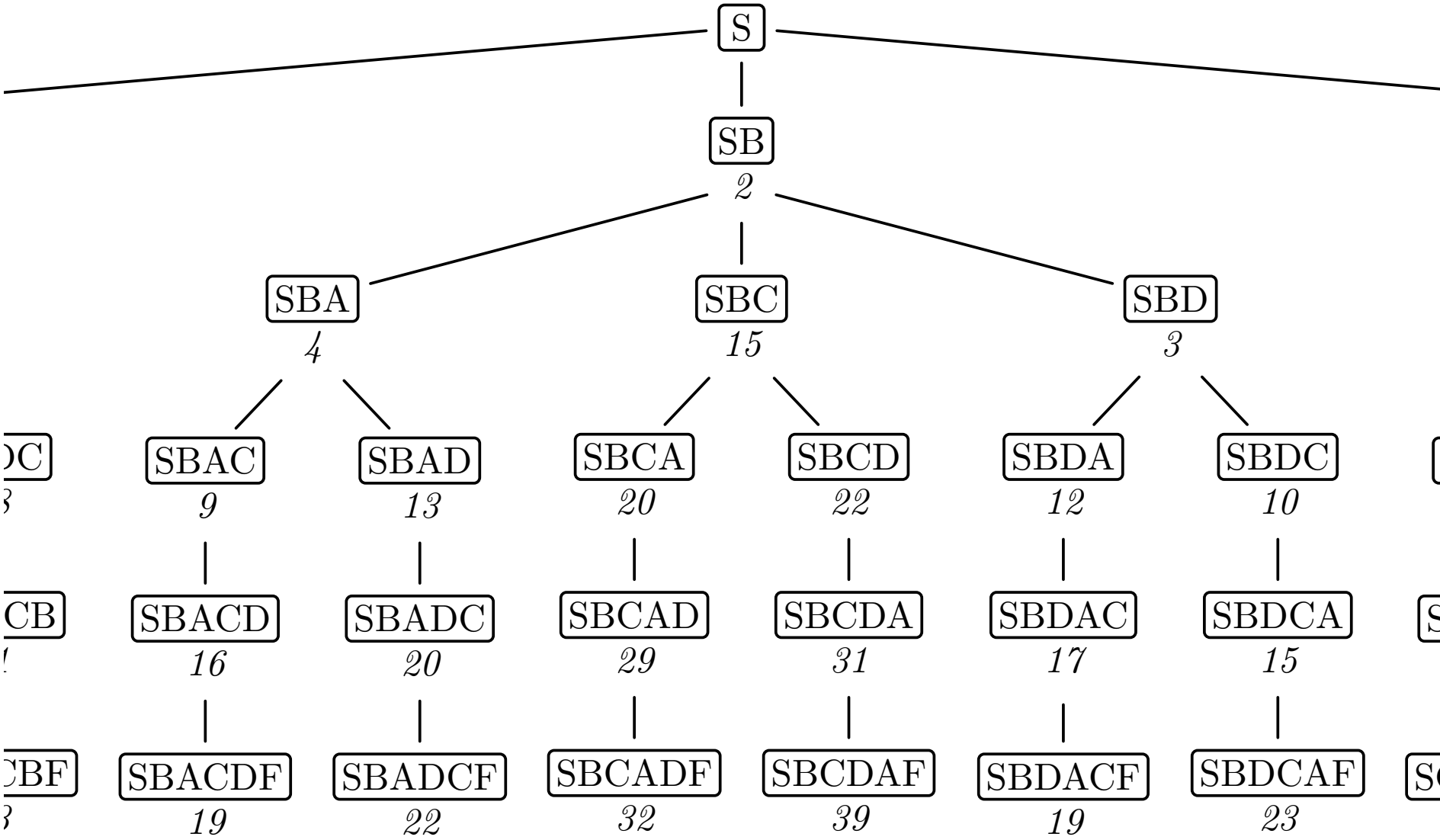


TSP: full tree of choices

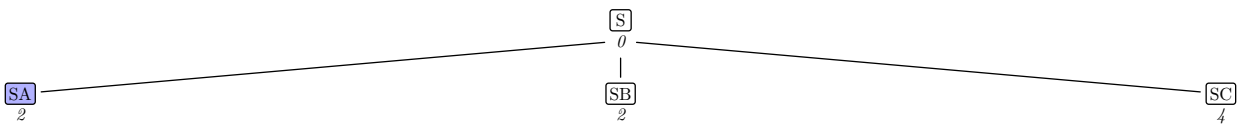


- 67 nodes
- 6 layers
- 18 nodes on the bottom layer $\leftrightarrow$ 18 paths to choose from

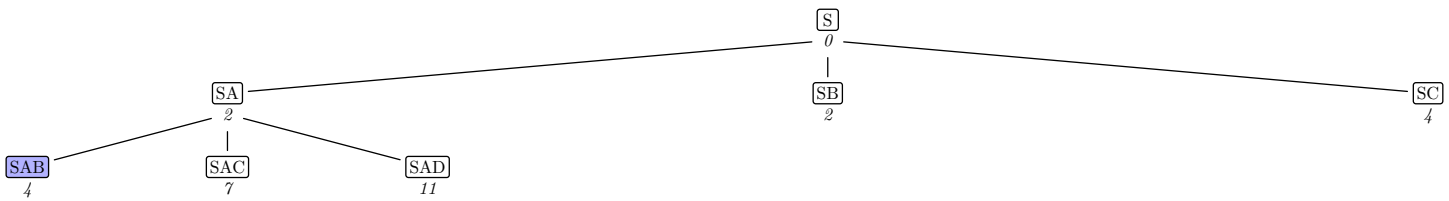
TSP: full tree of choices



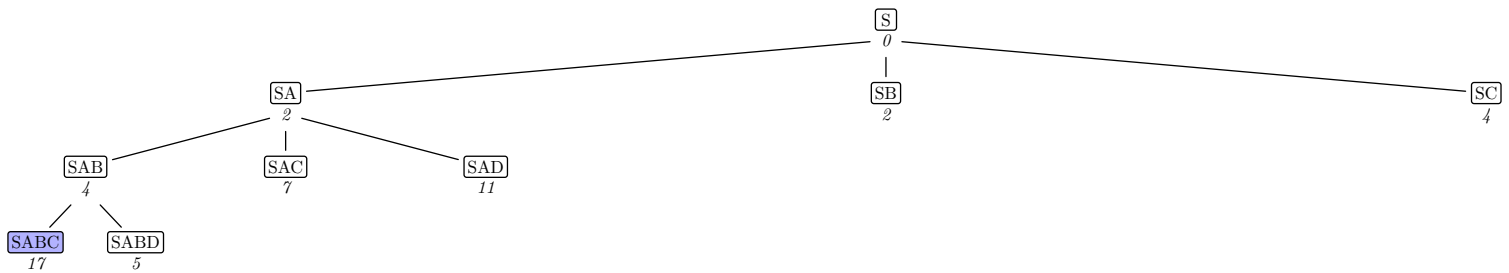
BnB step 1



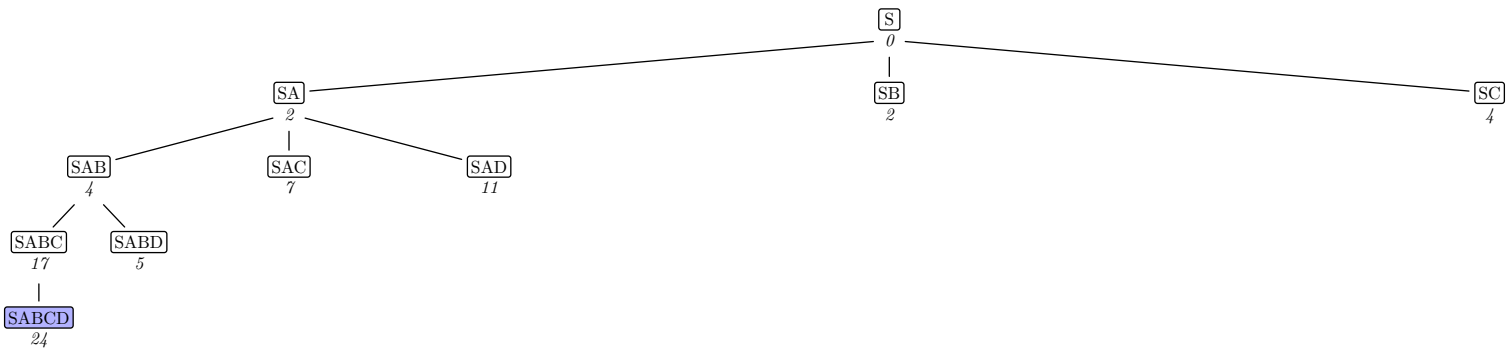
BnB step 2



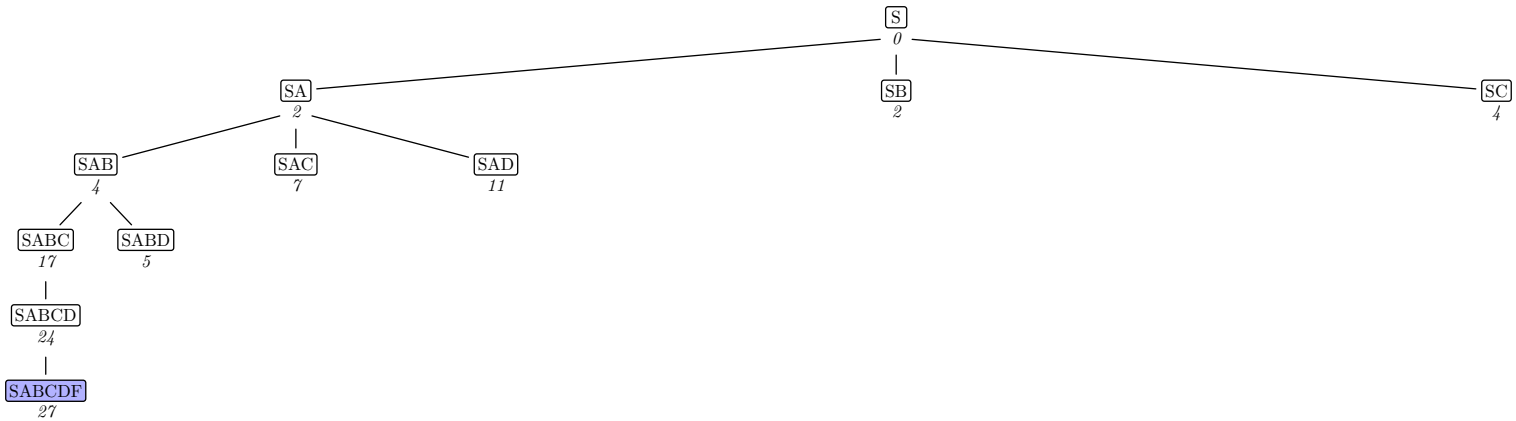
BnB step 3



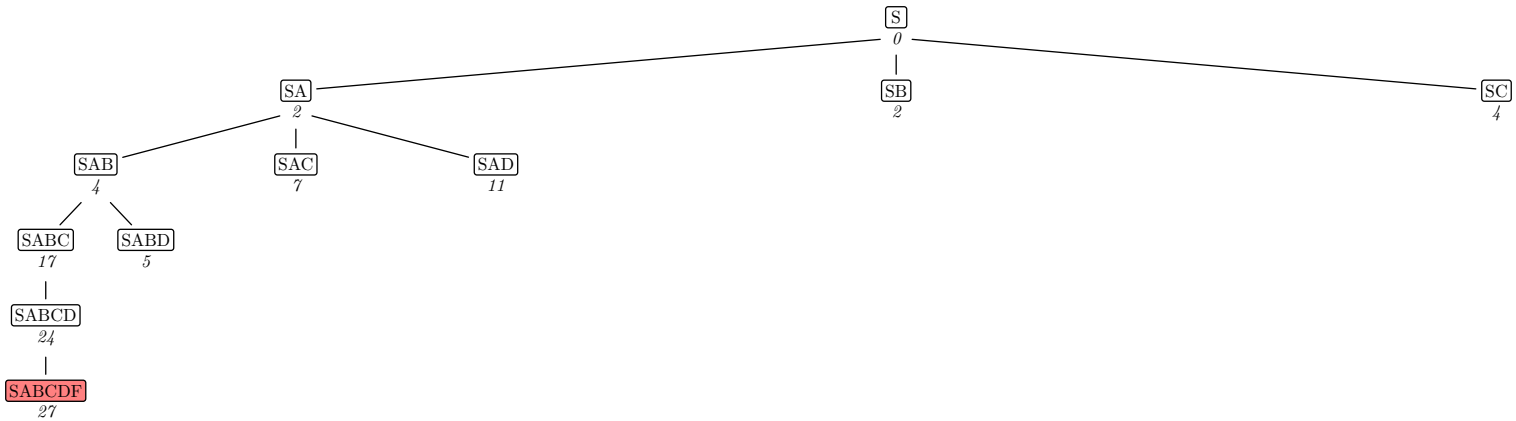
BnB step 4



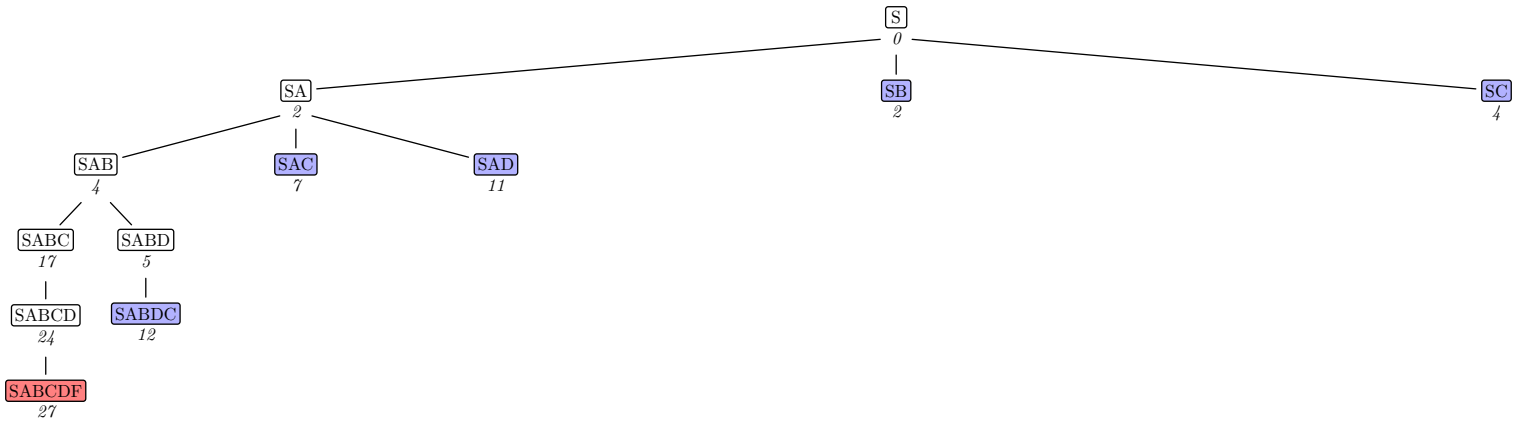
BnB step 5


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SABCD
24
SABCDF
27

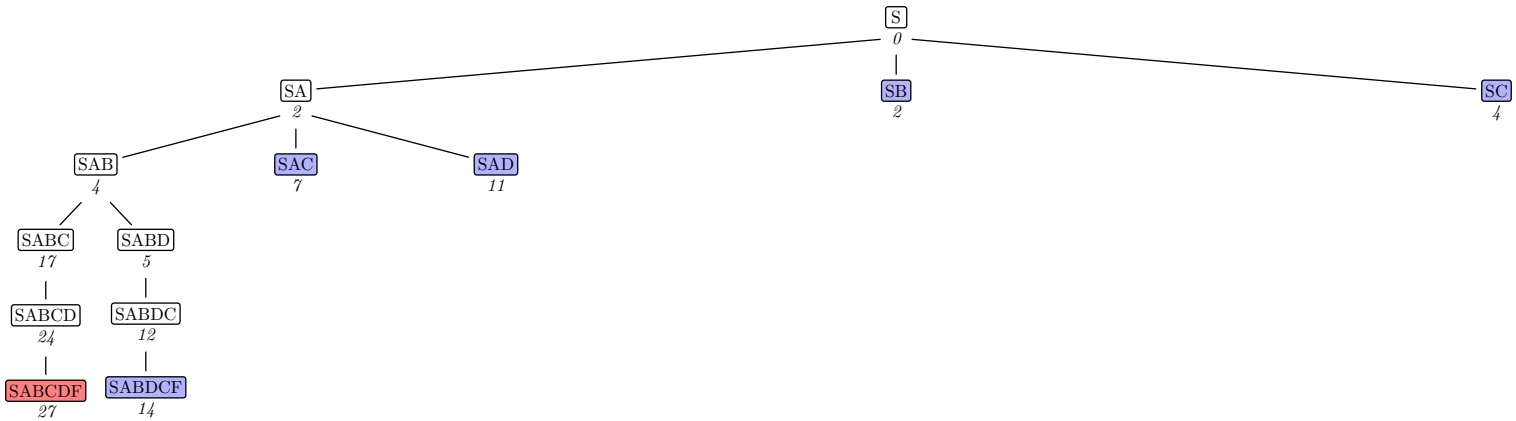
BnB step 6


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SABCD
24
SABCDF
27

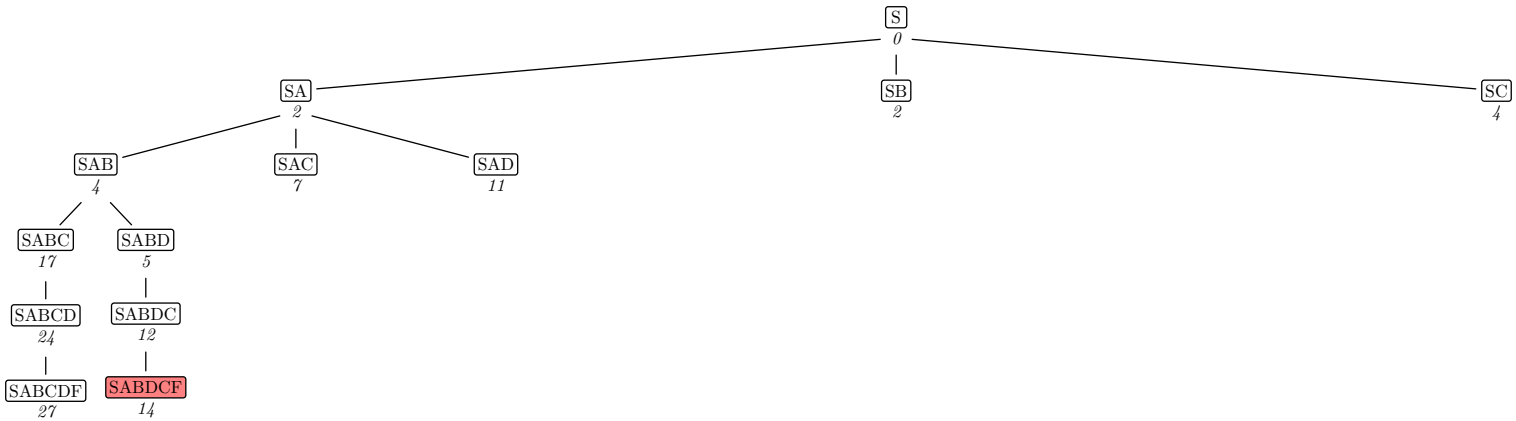
BnB step 7


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SABCD
24
SABDC
12
SABCDF
27

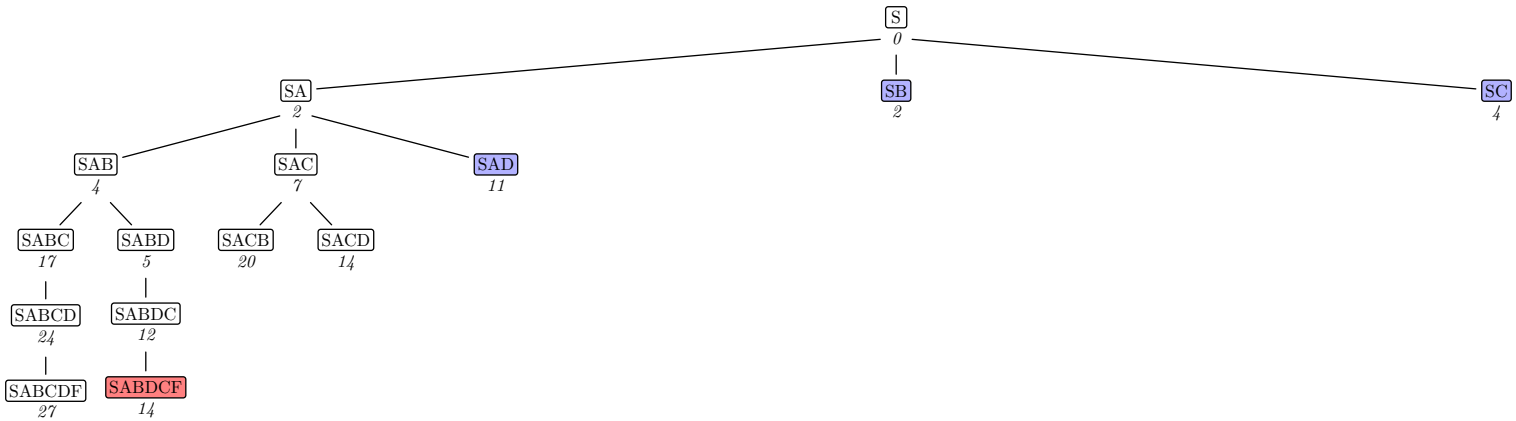
BnB step 8


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SABCD
24
SABDC
12
SABCDF
27
SABDCF
14

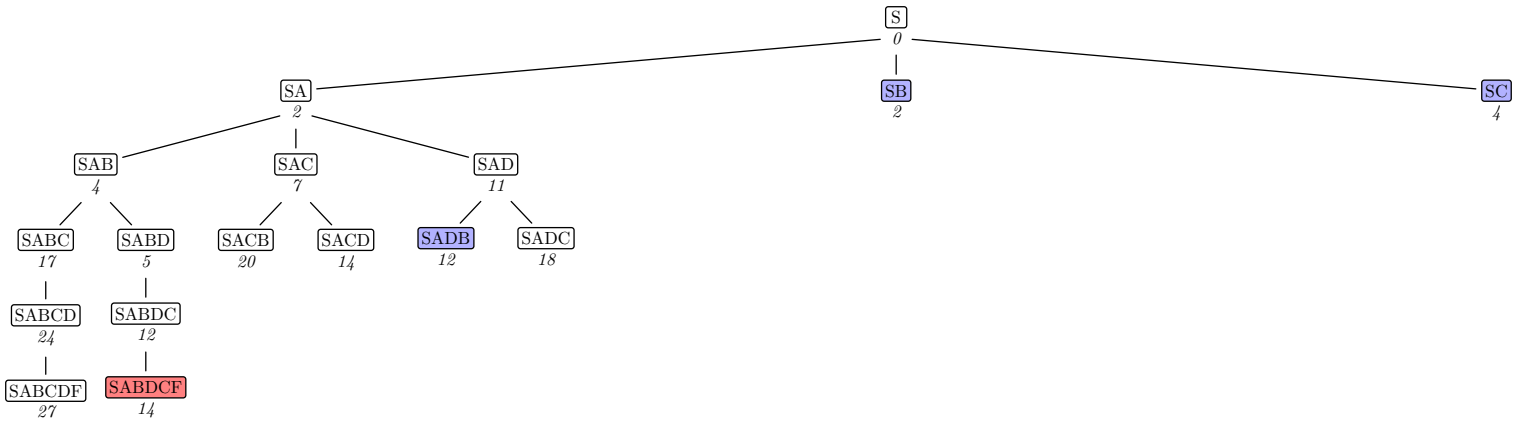
BnB step 9


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SABCD
24
SABDC
12
SABCDF
27
SABDCF
14

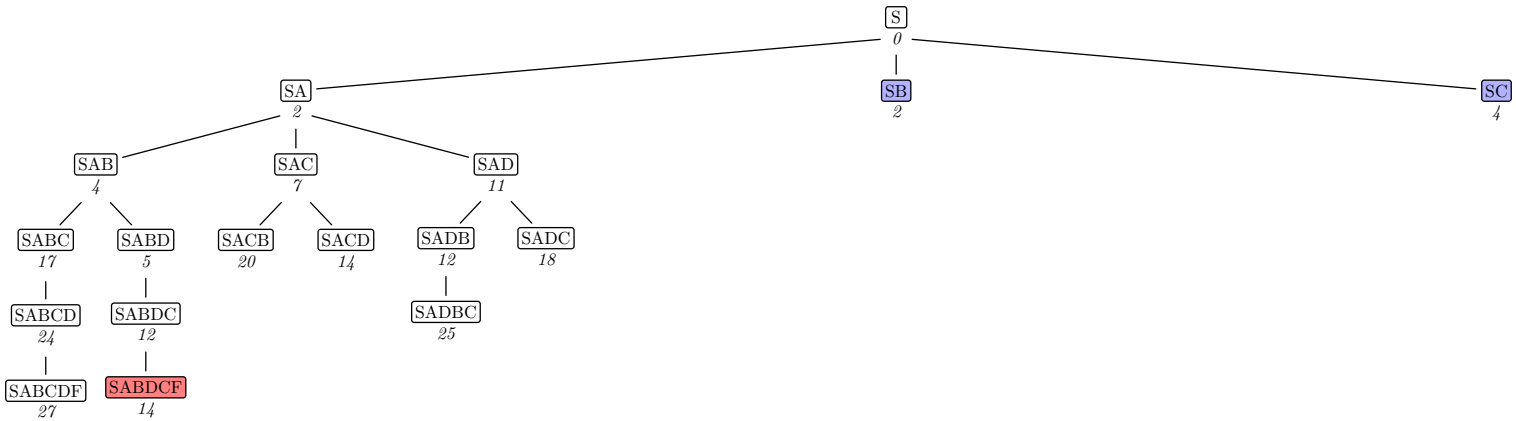
BnB step 10


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SACB
20
SACD
14
SABCD
24
SABDC
12
SABCDF
27
SABDCF
14

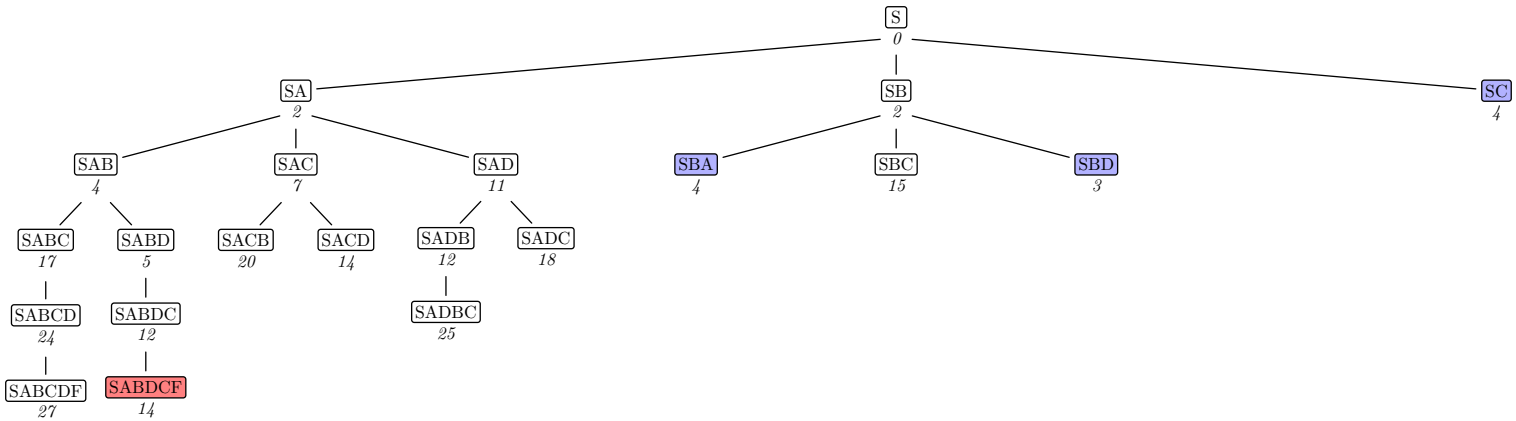
BnB step 11


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SABCD
24
SABDC
12
SABCDF
27
SABDCF
14

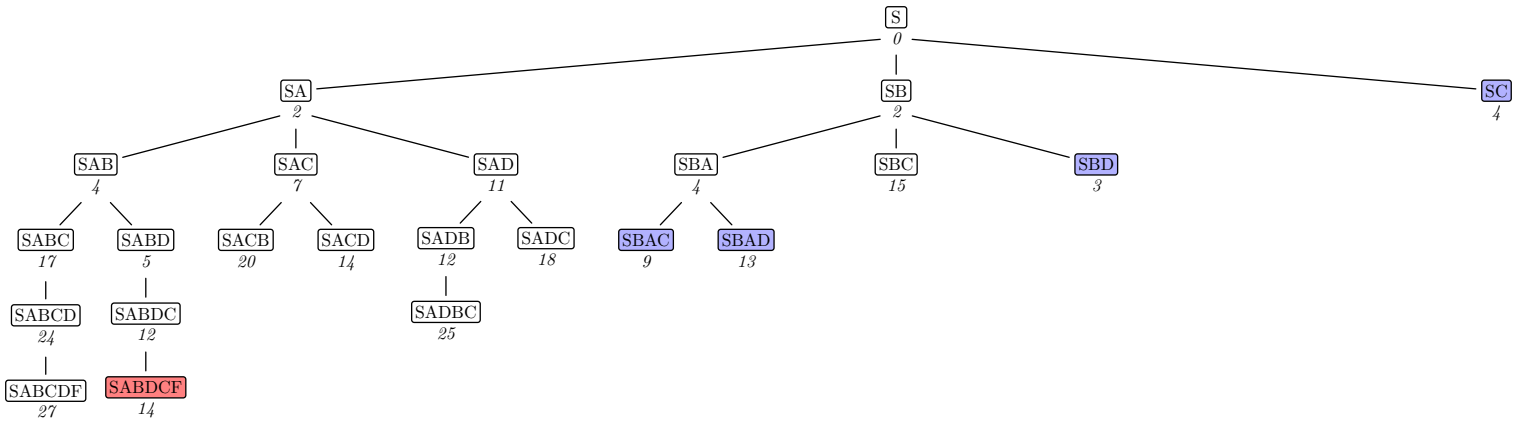
BnB step 12


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SABCD
24
SABDC
12
SADBC
25
SABCDF
27
SABDCF
14

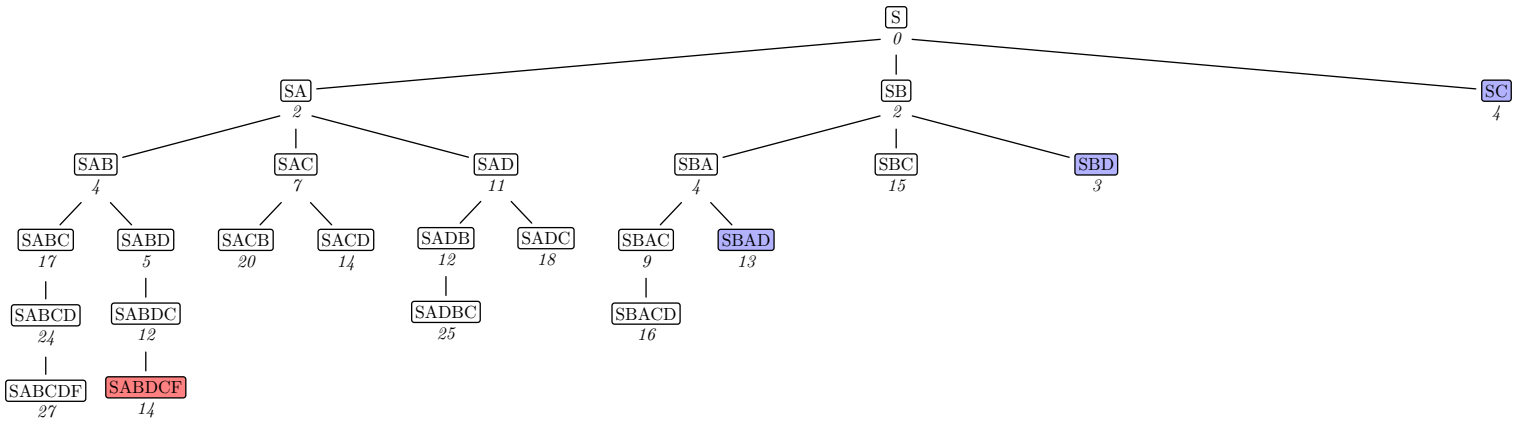
BnB step 13


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SABCD
24
SABDC
12
SADBC
25
SABCDF
27
SABDCF
14

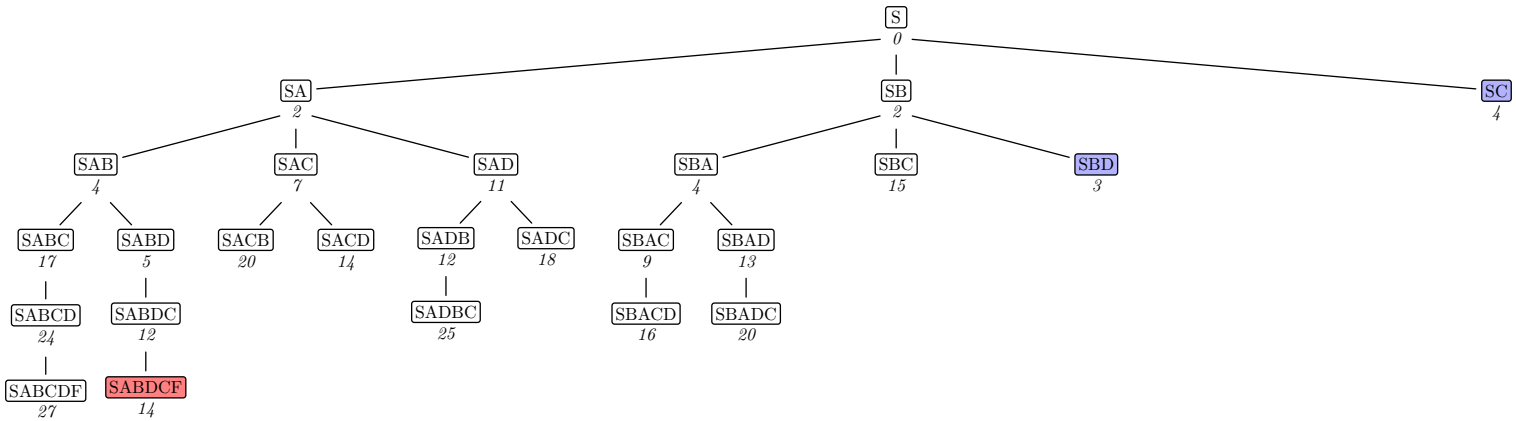
BnB step 14


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SABCD
24
SABDC
12
SADBC
25
SABCDF
27
SABDCF
14

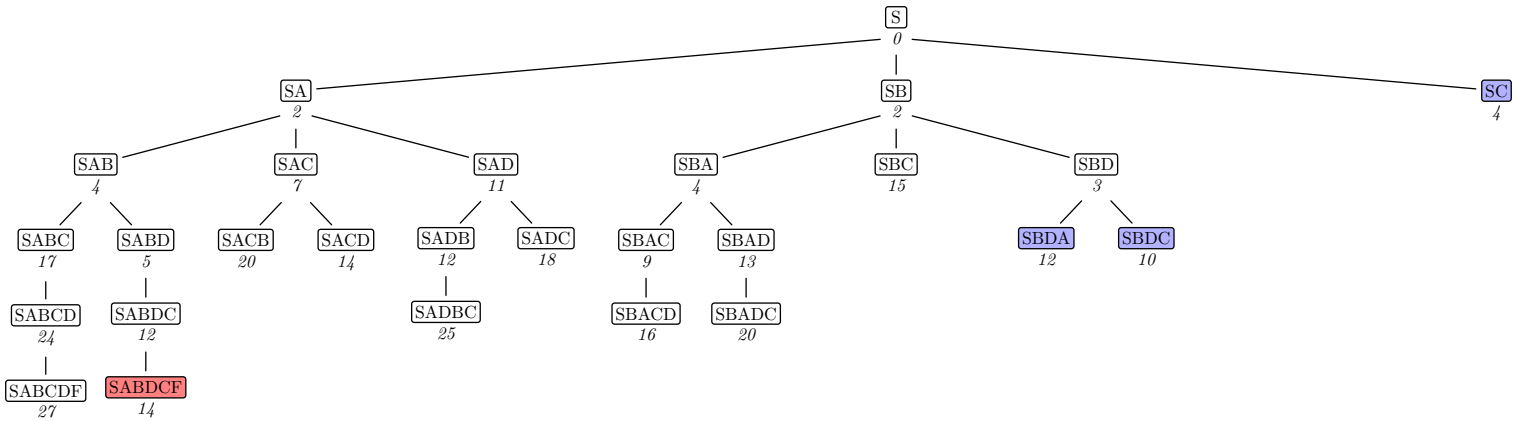
BnB step 15


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SABCDF
27
SABDCF
14

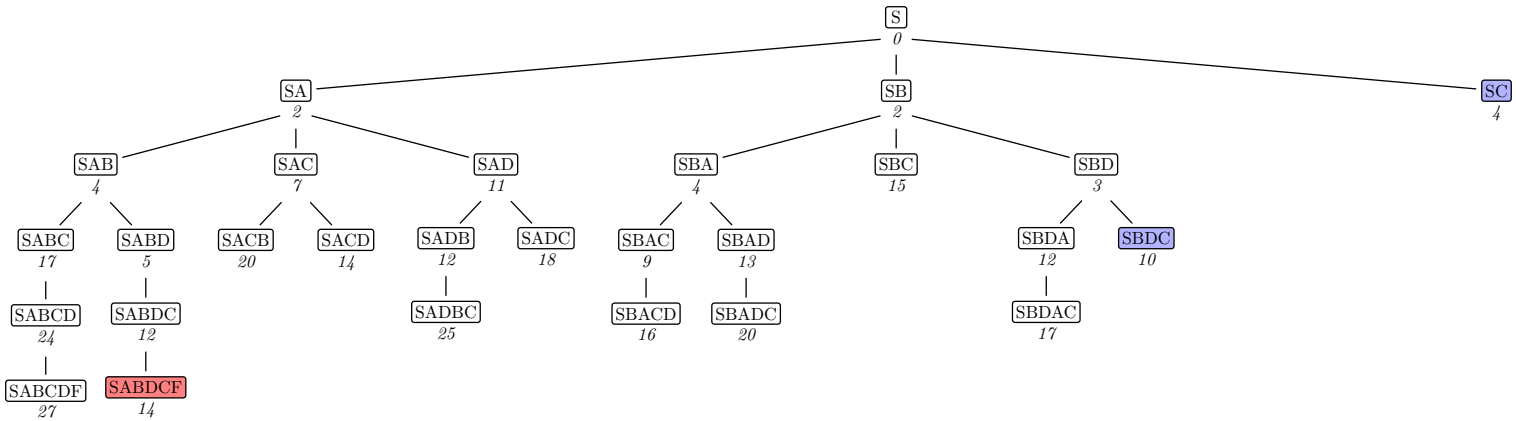
BnB step 16


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SABCDF
27
SABDCF
14

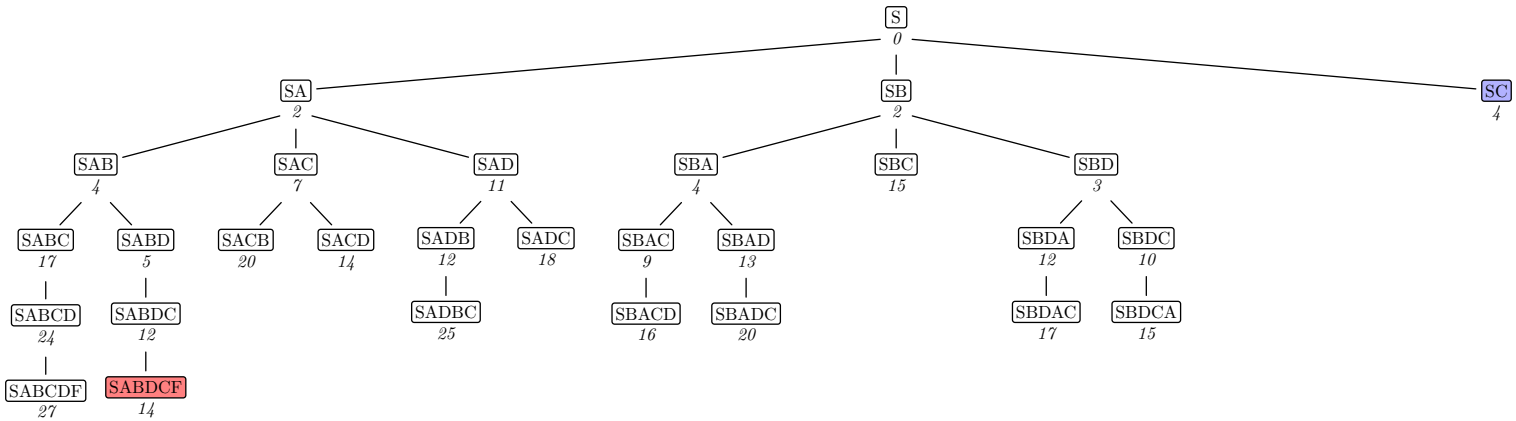
BnB step 17


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SABCDF
27
SABDCF
14

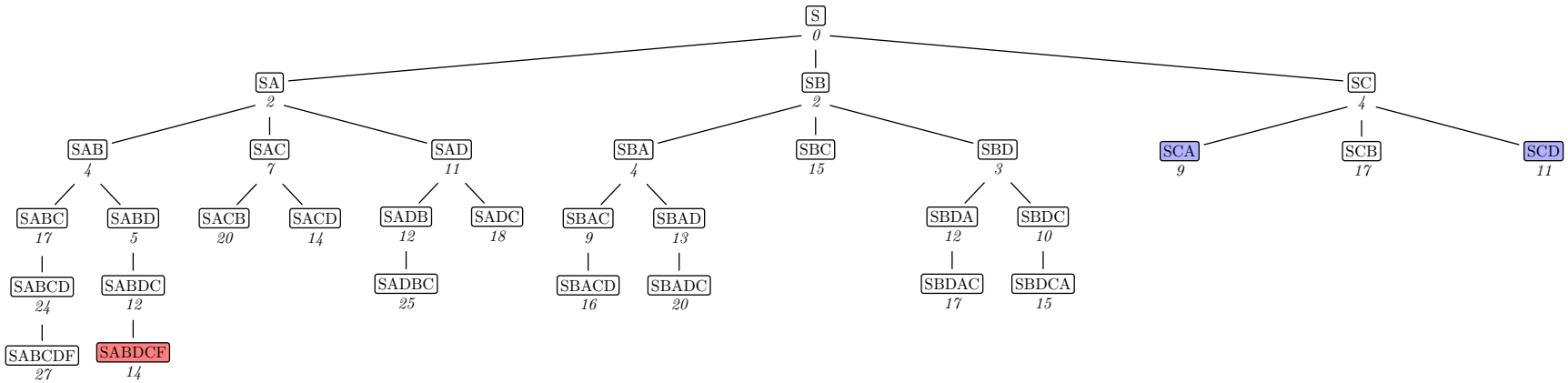
BnB step 18


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SBDAC
17
SABCDF
27
SABDCF
14

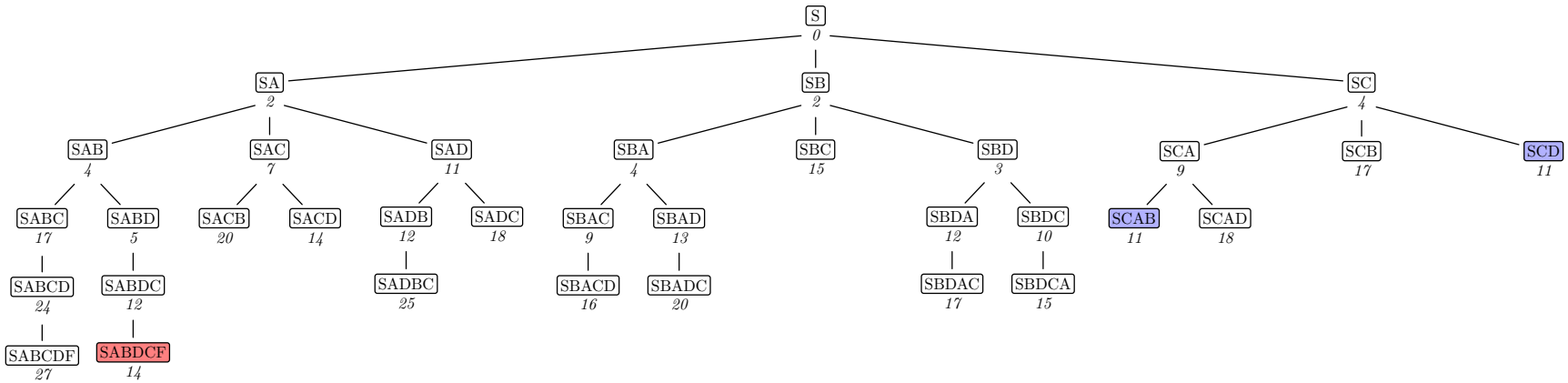
BnB step 19


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SBDAC
17
SBDCA
15
SABCDF
27
SABDCF
14

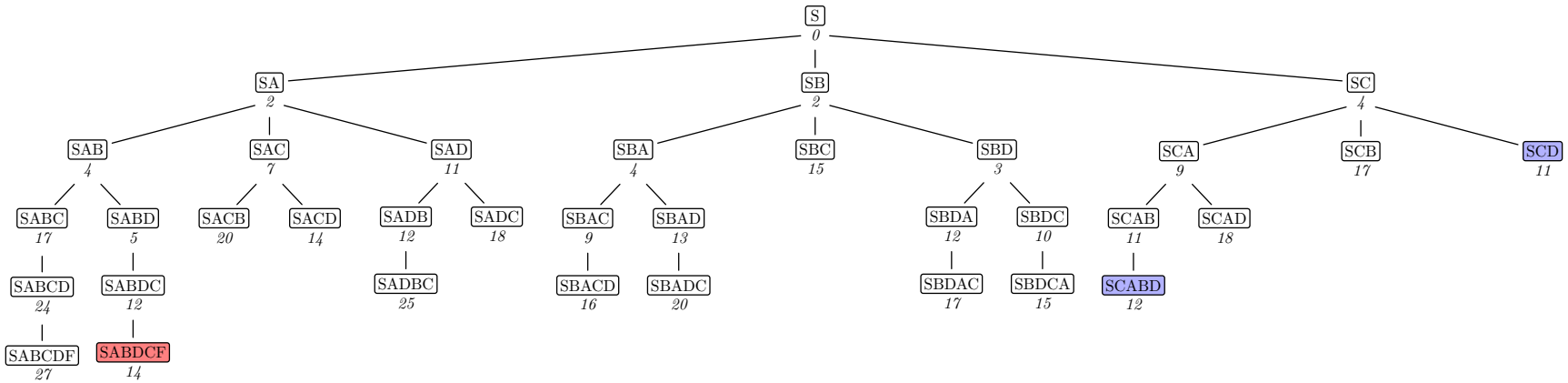
BnB step 20


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SCA
9
SCB
17
SCD
11
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SBDAC
17
SBDCA
15
SABCDF
27
SABDCF
14

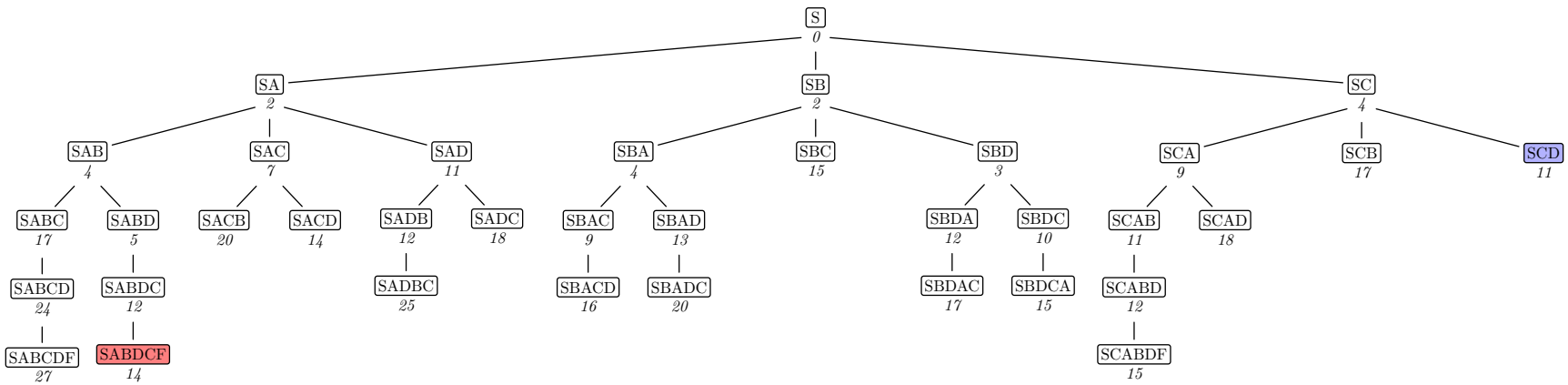
BnB step 21


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SCA
9
SCB
17
SCD
11
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SCAB
11
SCAD
18
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SBDAC
17
SBDCA
15
SABCDF
27
SABDCF
14

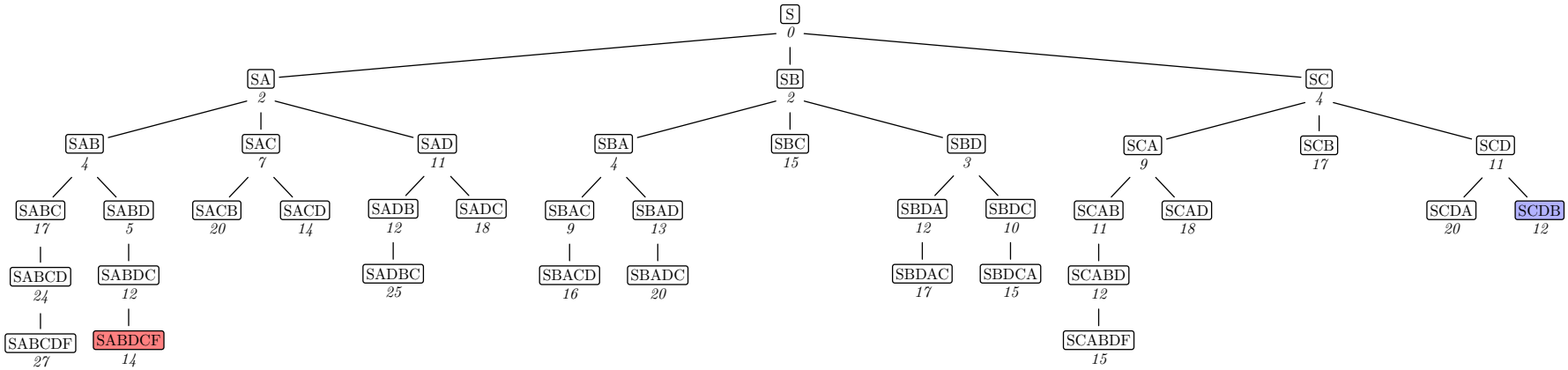
BnB step 22


S 0
SA 2
SB 2
SC 4
SAB 4
SAC 7
SAD 11
SBA 4
SBC 15
SBD 3
SCA 9
SCB 17
SCD 11
SABC 17
SABD 5
SACB 20
SACD 14
SADB 12
SADC 18
SBAC 9
SBAD 13
SBDA 12
SBDC 10
SCAB 11
SCAD 18
SABCD 24
SABDC 12
SADBC 25
SBACD 16
SBADC 20
SBDAC 17
SBDCA 15
SCABD 12
SABCDF 27
SABDCF 14

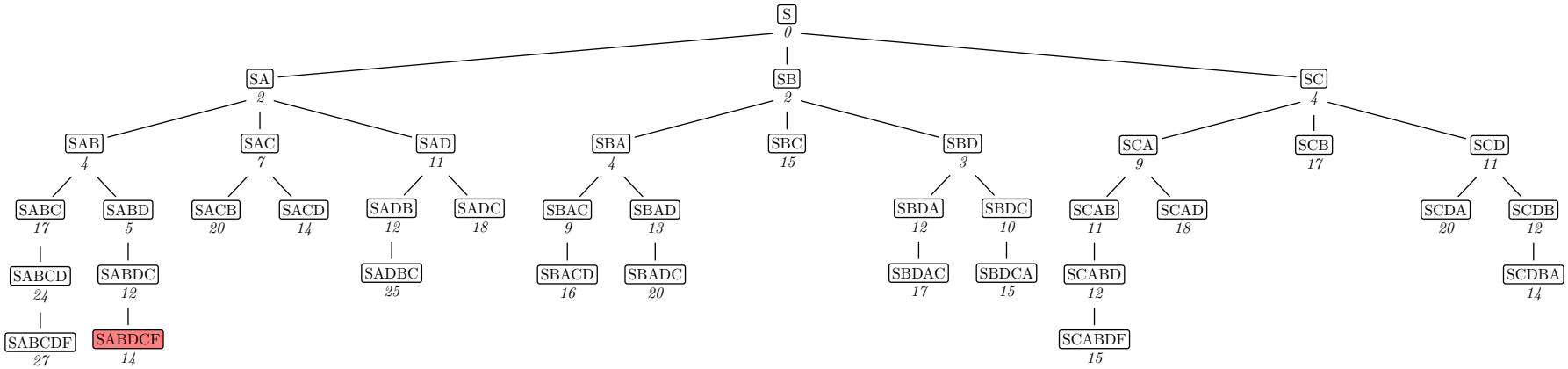
BnB step 23


S
0
SA
2
SB
2
SC
4
SAB
4
SAC
7
SAD
11
SBA
4
SBC
15
SBD
3
SCA
9
SCB
17
SCD
11
SABC
17
SABD
5
SACB
20
SACD
14
SADB
12
SADC
18
SBAC
9
SBAD
13
SBDA
12
SBDC
10
SCAB
11
SCAD
18
SABCD
24
SABDC
12
SADBC
25
SBACD
16
SBADC
20
SBDAC
17
SBDCA
15
SCABD
12
SABCDF
27
SABDCF
14
SCABDF
15

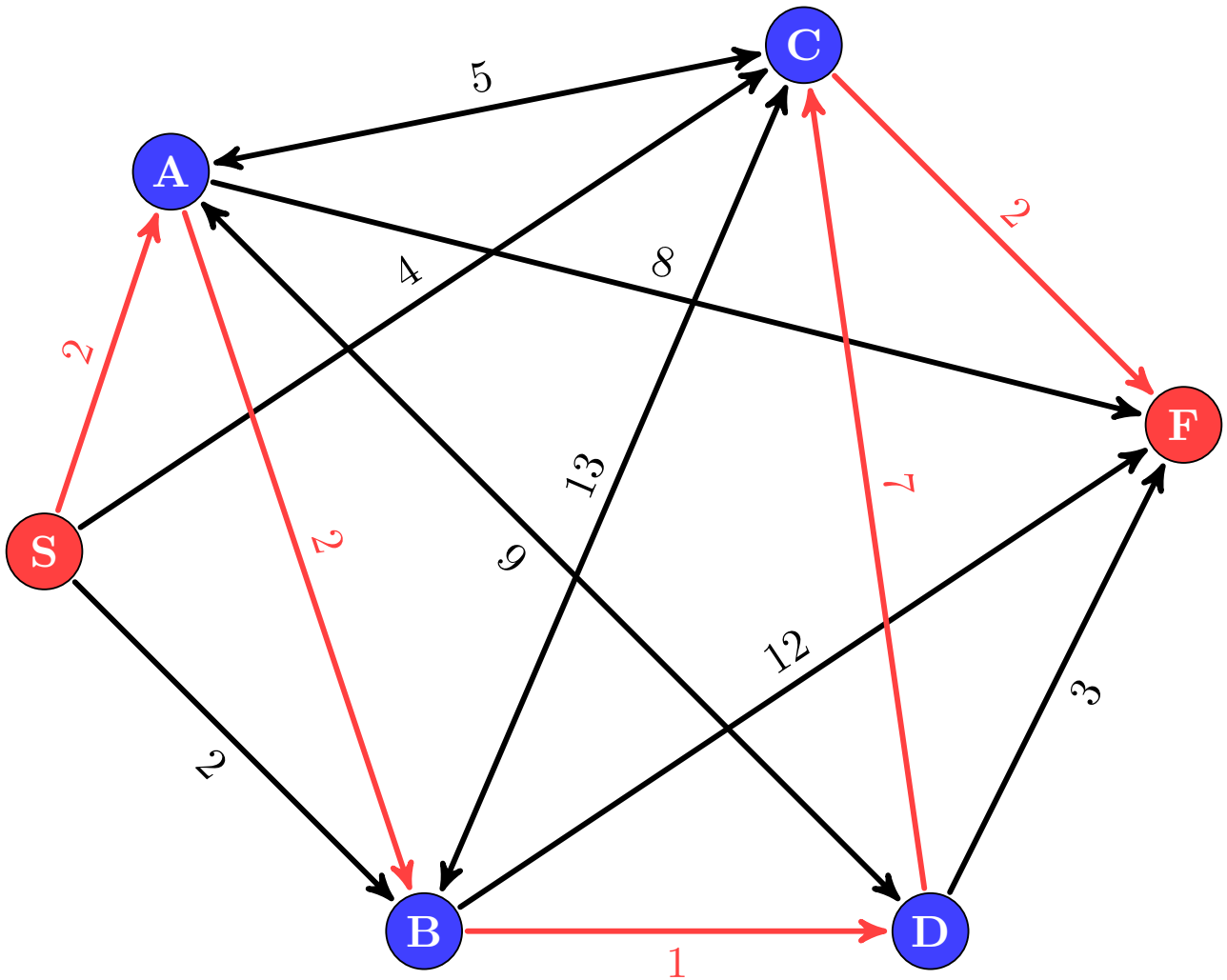
BnB step 24


S 0
SA 2
SB 2
SC 4
SAB 4
SAC 7
SAD 11
SBA 4
SBC 15
SBD 3
SCA 9
SCB 17
SCD 11
SABC 17
SABD 5
SACB 20
SACD 14
SADB 12
SADC 18
SBAC 9
SBAD 13
SBDA 12
SBDC 10
SCAB 11
SCAD 18
SCDA 20
SCDB 12
SABCD 24
SABDC 12
SADBC 25
SBACD 16
SBADC 20
SBDAC 17
SBDCA 15
SCABD 12
SABCDF 27
SABDCF 14
SCABDF 15

BnB step 25


S 0
SA 2
SB 2
SC 4
SAB 4
SAC 7
SAD 11
SBA 4
SBC 15
SBD 3
SCA 9
SCB 17
SCD 11
SABC 17
SABD 5
SACB 20
SACD 14
SADB 12
SADC 18
SBAC 9
SBAD 13
SBDA 12
SBDC 10
SCAB 11
SCAD 18
SCDA 20
SCDB 12
SABCD 24
SABDC 12
SADBC 25
SBACD 16
SBADC 20
SBDAC 17
SBDCA 15
SCABD 12
SCDBA 14
SABCDF 27
SABDCF 14
SCABDF 15

TSP: shortest path is length 14



Theory of BnB: branching

$$\min f(x) \text{ s.t. } x \in \Omega$$

$f(x)$ objective function

Ω set of feasible x

$\mathcal{P}_j(\Omega)$ partition of Ω into k_j subsets

$$\mathcal{P}_j(\Omega) = \{\Omega_{j1}, \ldots, \Omega_{jk_j} : \Omega_{ji} \cap \Omega_{ji'} = \varnothing, i \neq i', \cup_{i=1}^{k_j} \Omega_{ji} = \Omega\}$$

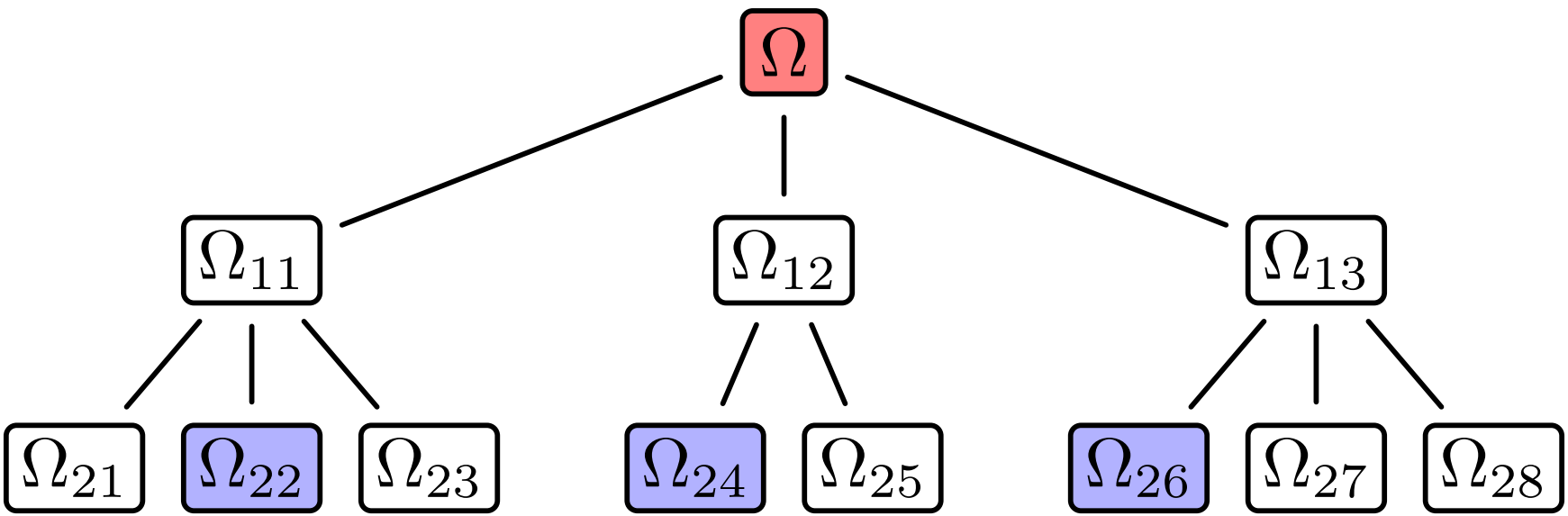
$\{\mathcal{P}_j(\Omega)\}_{j=1,\ldots,J}$ forms a sequence of J gradually refined partitions

$$k_1 \leq \cdots \leq k_j \leq \cdots \leq k_J$$

$$\forall j = 1, \ldots, J, \forall i = 1, \ldots, k_j : \quad \exists i' \text{ such that } \Omega_{ij} \subset \Omega_{i'j'} \; \forall j' < j$$

Theory of BnB: branching

$$\min f(x) \text{ s.t. } x \in \Omega$$



Theory of BnB: bounding

$$\min f(x) \text{ s.t. } x \in \Omega$$

$g(\Omega_{ij})$ bounding function: from subsets of Ω to real line

$g(\Omega_{ij}) = f(x)$ for singletons, i.e. when $\Omega_{ij} = \{x\}$

Monotonicity of bounding function

$$\forall j \; \forall \Omega_{i_1 1} \supset \Omega_{i_2 2} \supset \cdots \supset \Omega_{i_j j}$$

$$g(\Omega_{i_1 1}) \leq g(\Omega_{i_2 2}) \leq \cdots \leq g(\Omega_{i_j j})$$

- Inequalities are reversed for the minimization problem
- For TSP $g(\Omega_{ij})$ is accumulated distance

BnB with NRLS

- **Branching**: RLS tree
- **Bounding**: The bound function is partial likelihood calculated on the subset of states

$$\mathcal{L}(Z,\theta,s) = \max_{(\mathbf{P}^{\ell}(\theta),\mathbf{V}^{\ell}(\theta)\in SOL(\Psi,\theta)} \frac{1}{M}\sum_{i=1}^{N}\sum_{m=1}^{M}\sum_{t=1}^{T}\log P_i^{\ell}(\bar{a}_i{}^{mt}|\bar{\mathbf{x}}^{mt};\theta)$$

$$\text{s.t. } \bar{a}_i{}^{mt} \in s \subset S$$

Preliminary numerical results on NRLS with BnB

This slide has unintentionally been left blank

Conclusions: Bertrand investments model

- Many types of endogenous coordination is possible in equilibrium
  - Leapfrogging (alternating investments)
  - Preemption (investment by cost leader)
  - Duplicative (simultaneous investments)
- Full rent dissipation and monopoly outcomes are supported as MPE.
- Numerous MPE equilibria and "Folk theorem"-like result
- The equilibria are generally inefficient due to over-investment
  - Duplicative or excessively frequent investments

Conclusions: Solution of dynamic games

- When equilibrium is not unique the computation algorithm inadvertently acts as an equilibrium selection mechanism
- When directionality in the state space is present, state recursion algorithm is preferred to time iterations
- Plethora of Markov perfect equilibria poses new challenges:
  - How firms manage to coordinate on a particular equilibrium?
  - Increased difficulties for empirical applications.
  - Daunting perspectives for identification of equilibrium selection rule from the data.
- Next: estimation of dynamic games with multiple equilibria