

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

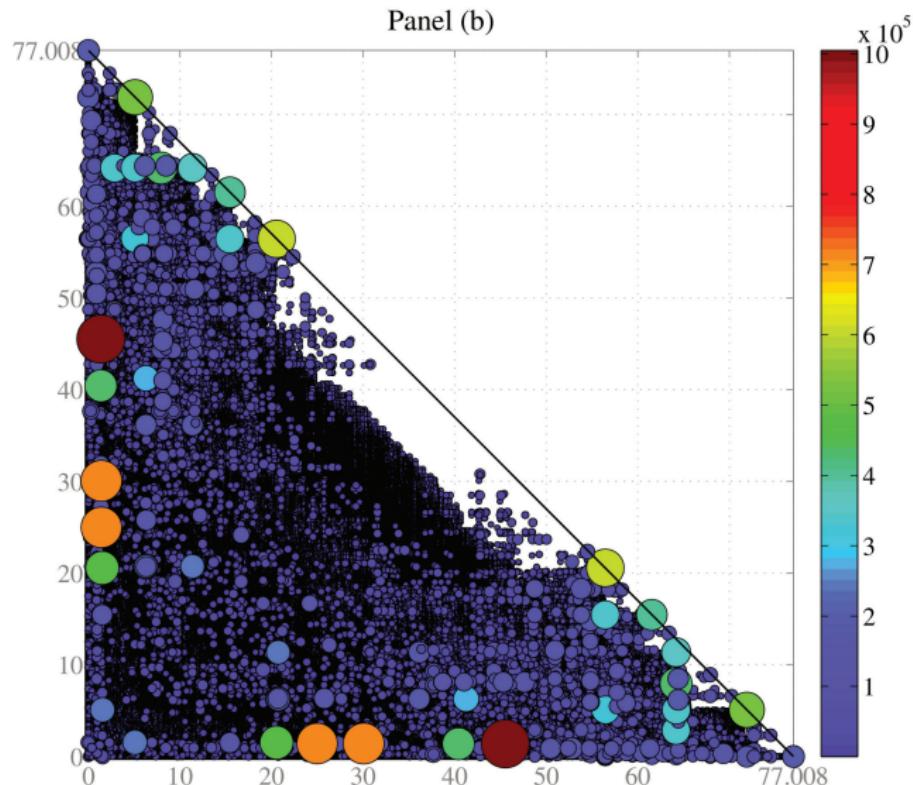
Fedor Iskhakov, Australian National University
John Rust, Georgetown University
Bertel Schjerning, University of Copenhagen

ZICE 2017
February 1, 2017

Stochastic dynamic games

- ① Several decision makers (*players*)
 - ② Maximize discounted expected lifetime utility
 - ③ Anticipate consequences of their current actions
 - ④ Anticipate actions by other players in current and future periods (*strategic interaction*)
 - ⑤ Operate in a stochastic environment (*state of the game*)
 - ⑥ Evolution of the environment (states) depend on the collective actions of the players
-
- **Solve** for equilibria of the game
 - Estimate structural parameters

Complete solution of a dynamic stochastic game



Why solve for all equilibria?

- Solving the model \Leftrightarrow finding all solutions
- Economic principles should be used to specify equilibrium selection rule
- Estimation of structural parameters
Failure to account for multiplicity leads to misspecification
- Counterfactual simulations
(Can equilibrium selection rule be reliably identified?)

Solution concept: Markov Perfect Equilibria (MPE)

 Maskin and Tirole (1988)

Solving dynamic games is challenging

Equilibria are characterized by a **very large** system of non-linear equations. There is **no general solution method**.

- ① Guess and verify
- ② Iteratively solve the dynamic optimization problem of each player
 -  Pakes and McGuire (1994, 2001)
- ③ Homotopy approach
 -  Besanko, Doraszelski and Kryukov (2010), Borkovsky, Doraszelski and Kryukov (2010)
- ④ Algebraic approach for games with special structure
 -  Datta (2010), Judd, Renner and Schmedders (2012)

Our approach: State recursion + RLS

- ① Partition the state space into linked **stages**
- ② Solve each **stage game** separately
- ③ Gradually build the equilibrium of the whole stochastic game from these solutions (**State recursion**)
- ④ Efficiently combine all solutions from each of the stage games to recover the full set of MPE of the whole stochastic game (**RLS**)

This is applicable to the class of **directional dynamic games (DDG)** which satisfy the intuitive notion of **directionality**

Examples of DDGs

- Any finite horizon game: directional passage of time
- Chess: the number of chess pieces still on the board only decreases and never increases
- Patent races: irreversible R&D investments
 - ❑ Judd, Schmedders and Yeltekin (2012) *IER*
- Tipping point game: growing market shares
 - ❑ Dubè, Hitsch and Chintagunta (2010) *Marketing Science*
- Bertrand price competition with leapfrogging investments: irreversible technological progress

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 ($t = 1, 2, \dots, \infty$)
- 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

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

Strategy profile

- $\sigma = (\sigma_1, \sigma_2)$ – pair of Markovian *behavior* strategies
- $$\sigma_i = (p_i(c_1, c_2, c), P_i^I(c_1, c_2, c)) \in \mathbb{R}_+ \times [0, 1]$$
- Pure strategies included as special case

Definition of Markov Perfect Equilibrium

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* $V(s) = (V_1(s), V_2(s))$, $V_i : S \rightarrow R$,

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

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

$$v_i^N(c_1, c_2, c) = r^i(c_1, c_2) + \beta EV_i(c_1, c_2, c|N)$$

$$v_i^I(c_1, c_2, c) = r^i(c_1, c_2) - K(c) + \beta EV_i(c_1, c_2, c|I)$$

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

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

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

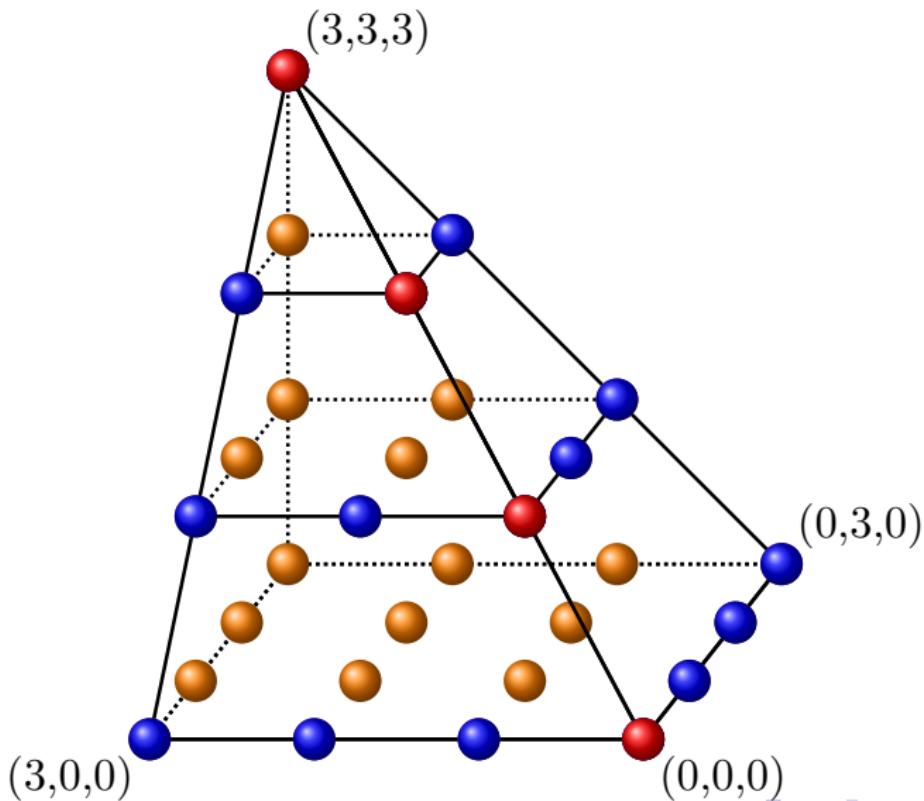
where

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

is the “smoothed max” or logsum function

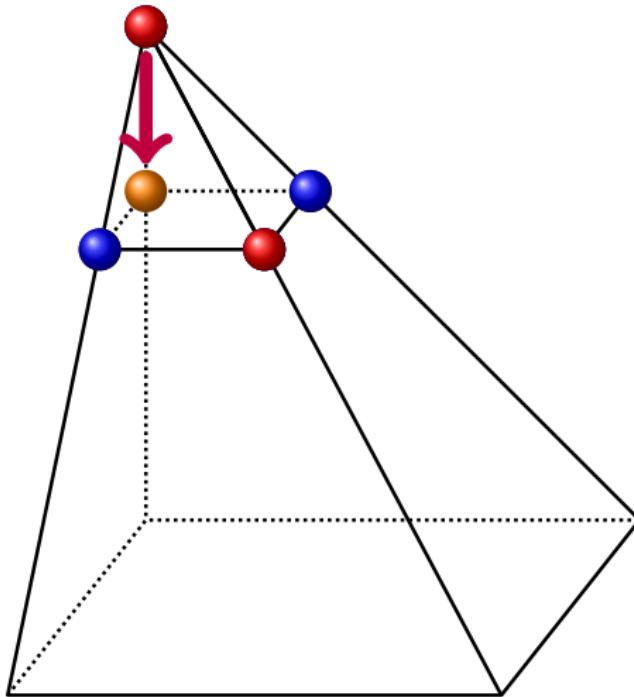
Discretized state space = a “quarter pyramid”

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



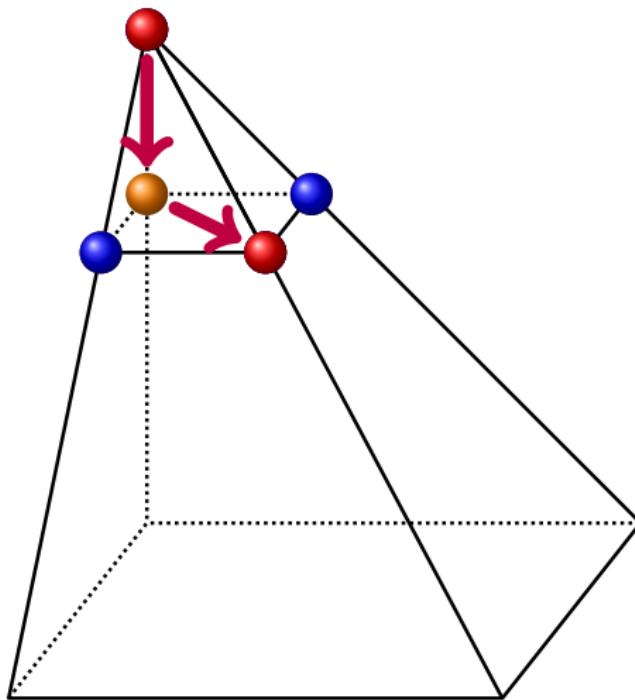
Game dynamics: example

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



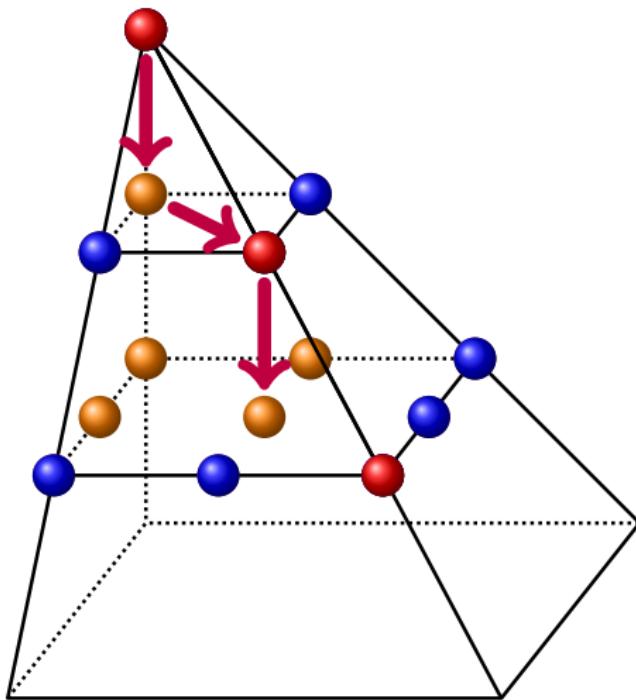
Game dynamics: example

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



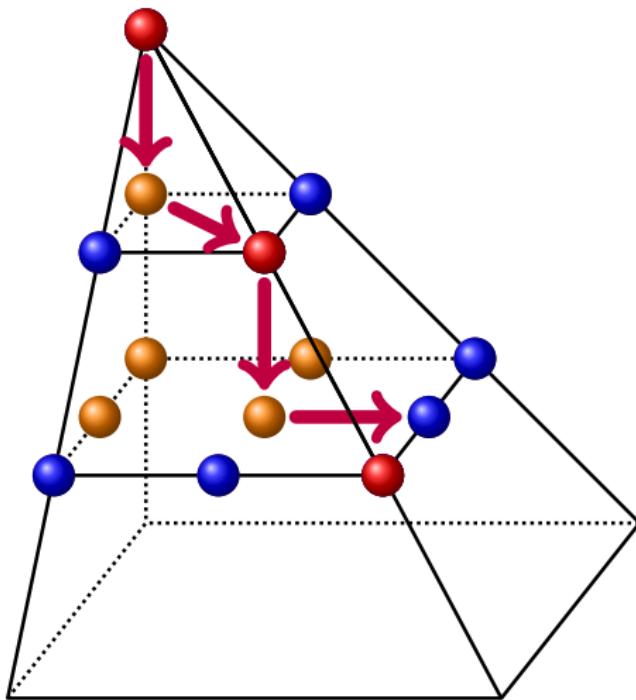
Game dynamics: example

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



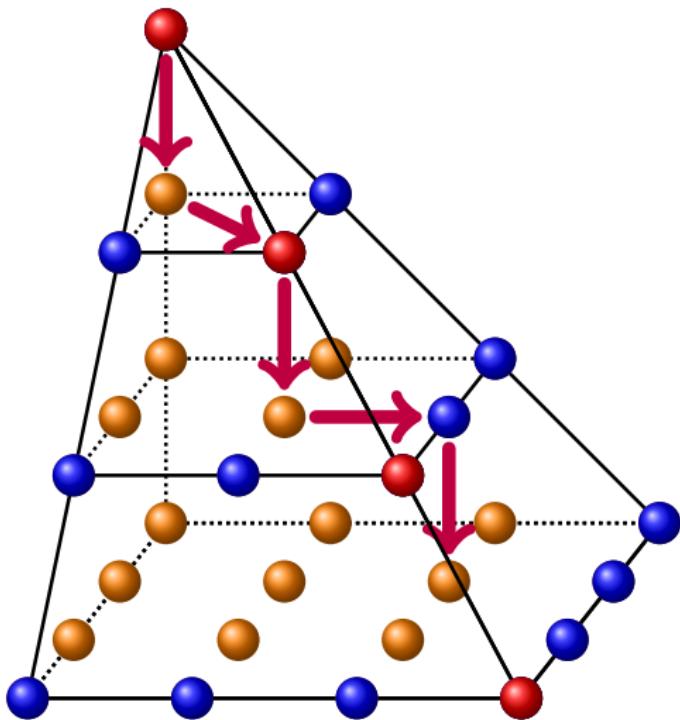
Game dynamics: example

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



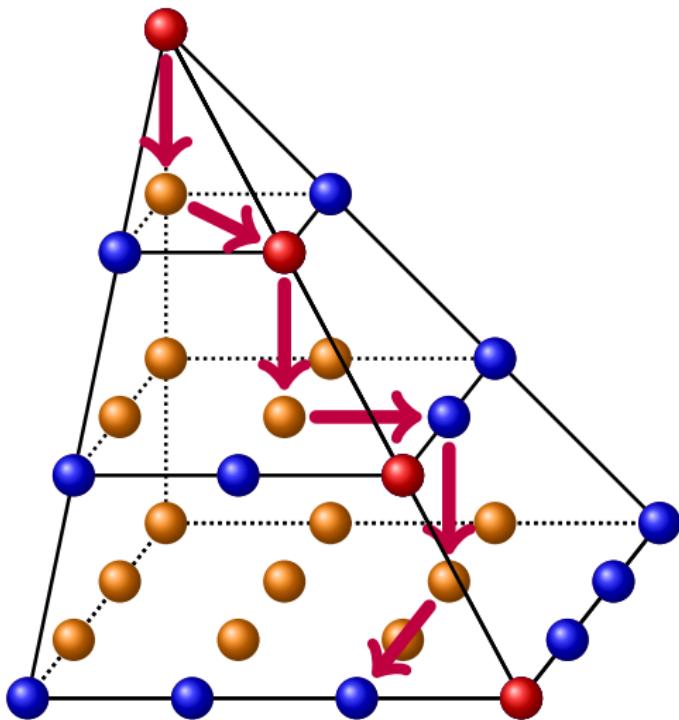
Game dynamics: example

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



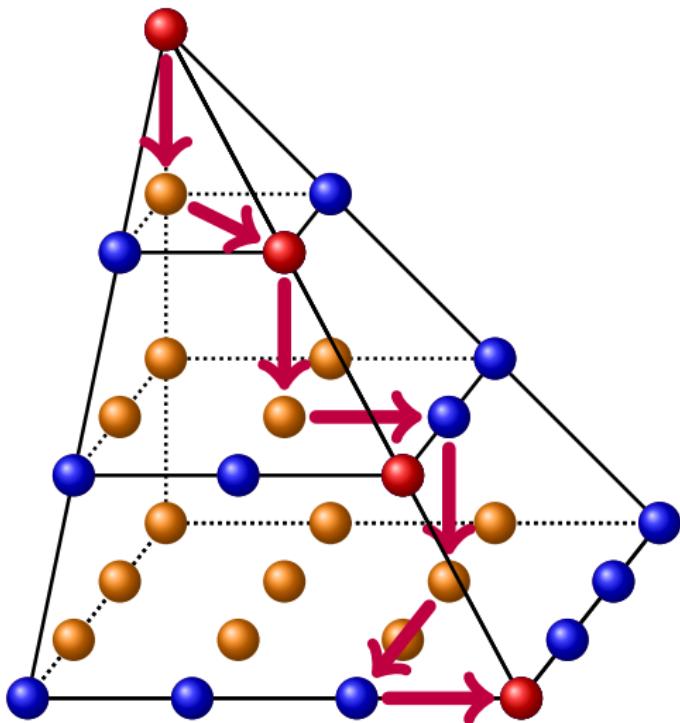
Game dynamics: example

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 $\rightsquigarrow (c_1, c_2, c) = (0, 0, 0)$

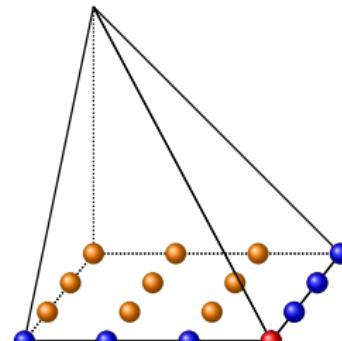
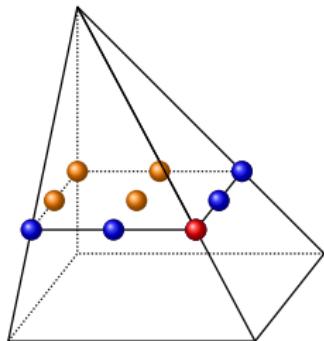
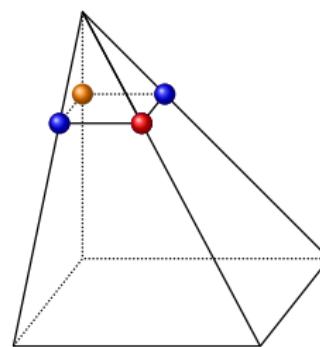
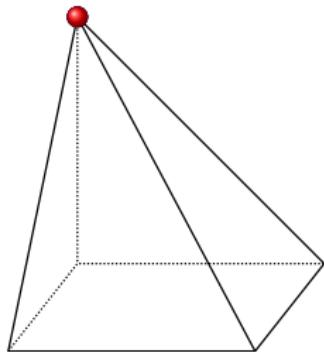


Road map of the talk

- 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

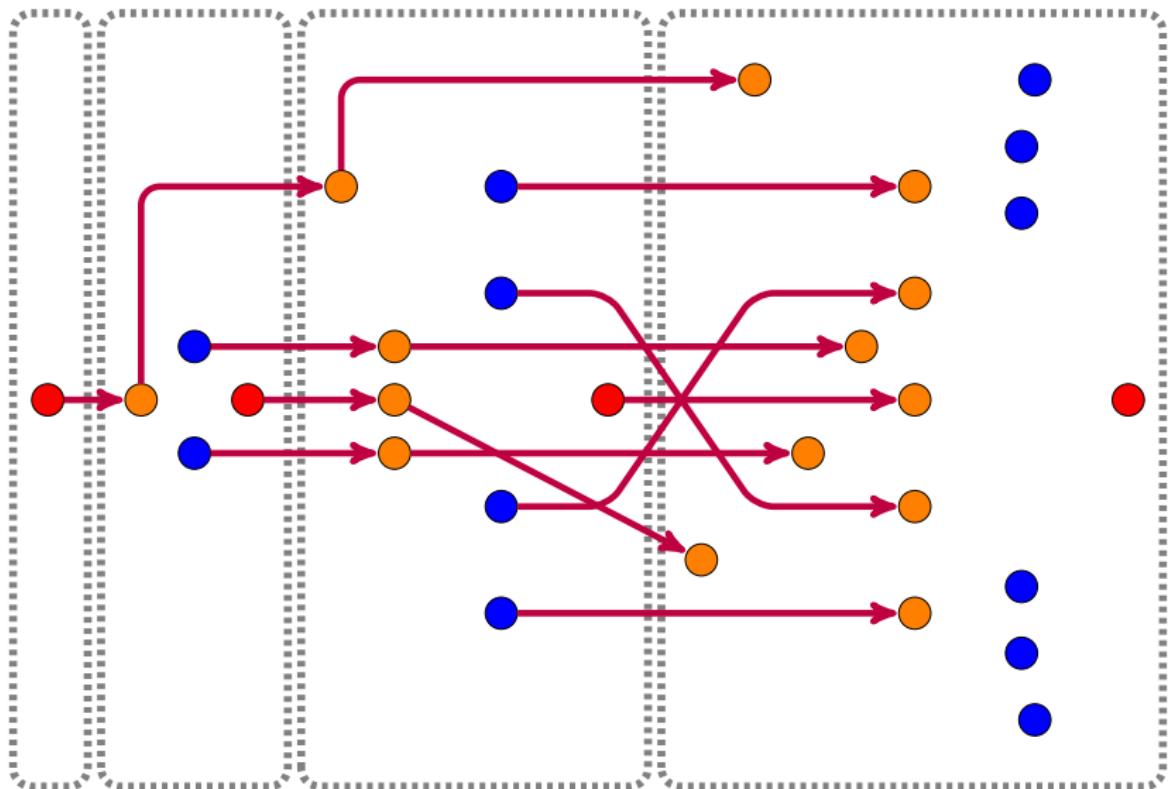
Transitions due to technological progress

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



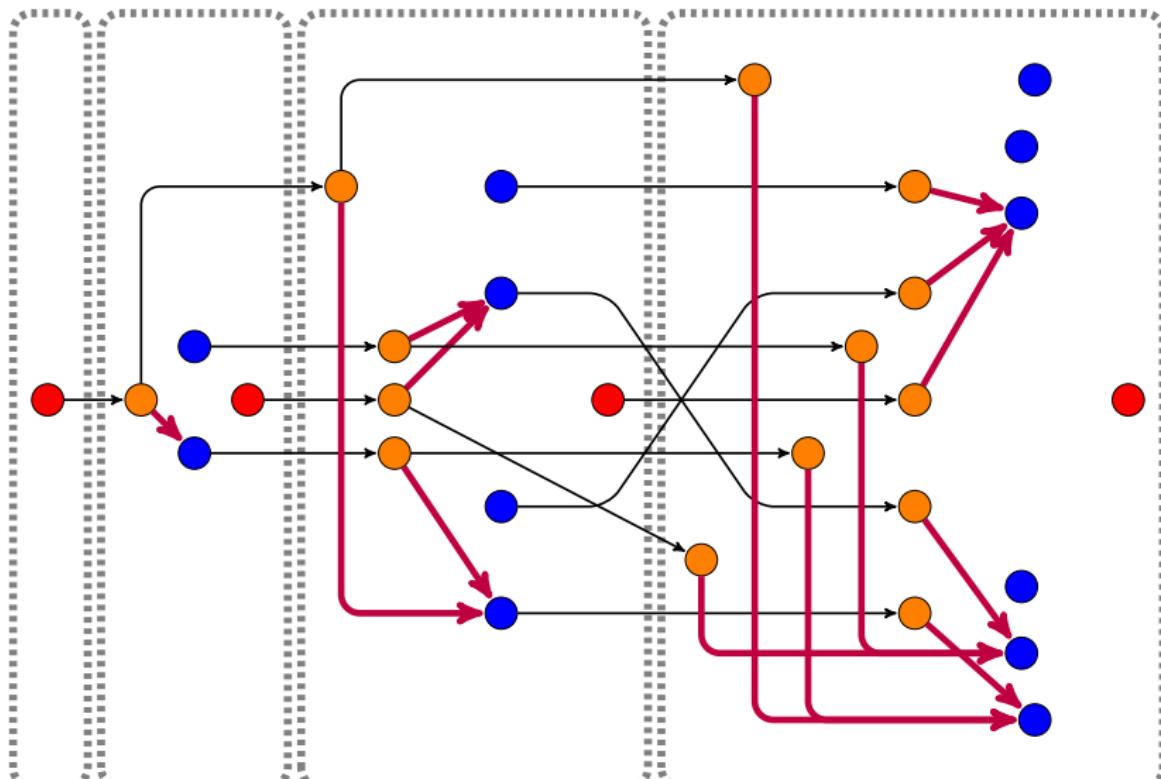
Transitions due to technological progress

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



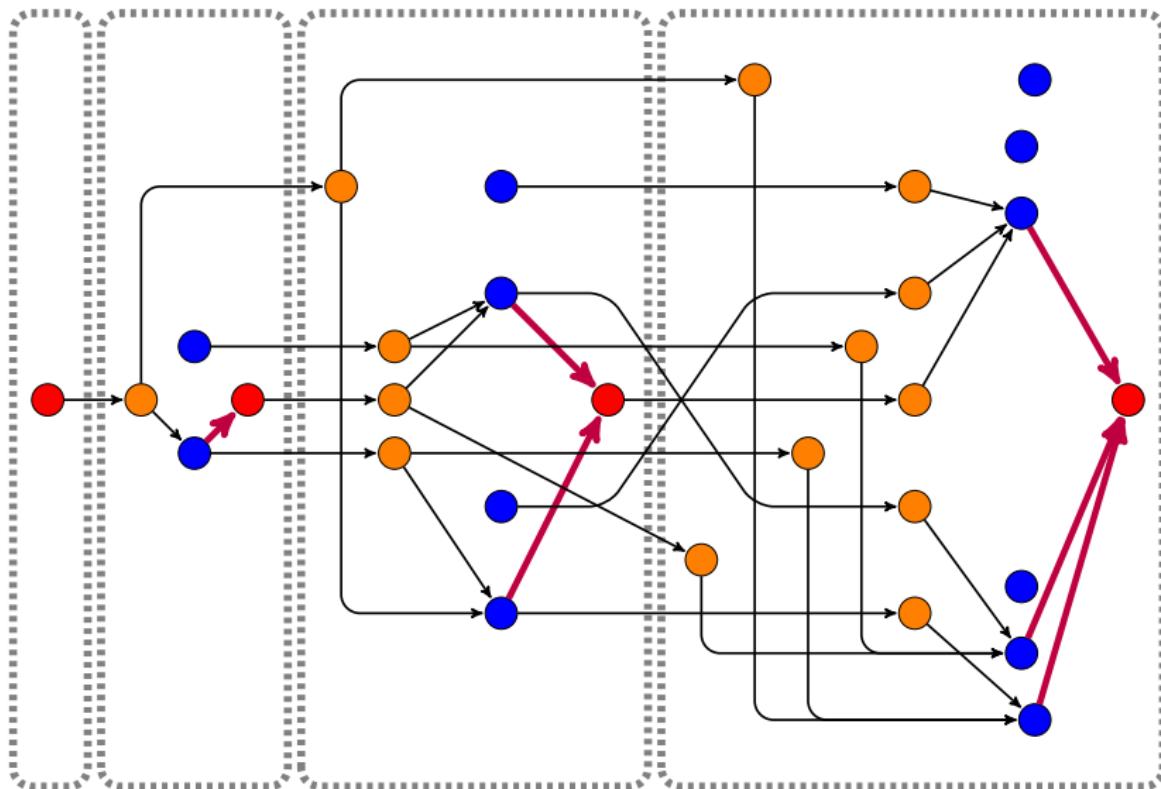
Strategy-specific partial order on S

Strategy σ_1 of firm 1: invest at all interior points



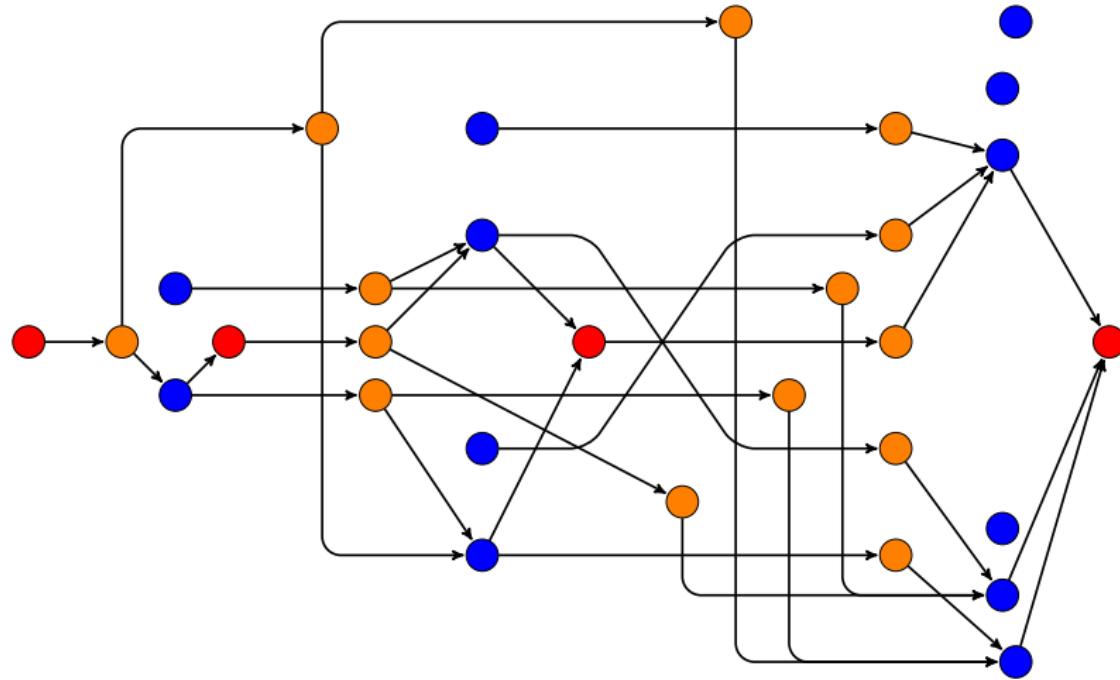
Strategy-specific partial order on S

Strategy σ_2 of firm 2: invest at all edge points



Strategy-specific partial order on S

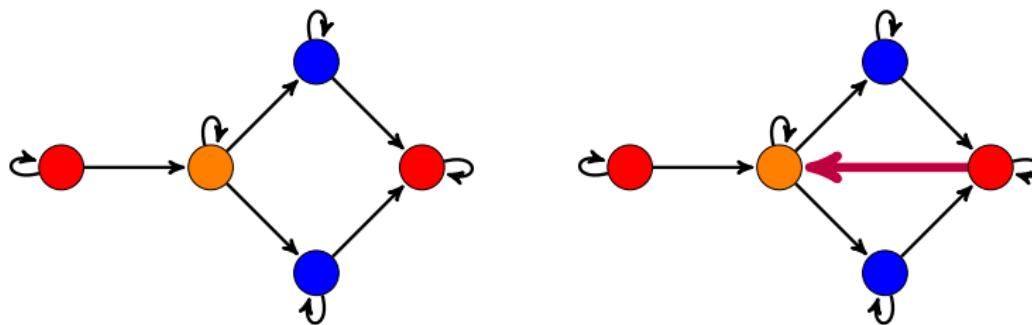
Strategy $\sigma = (\sigma_1, \sigma_2)$ of both firms



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

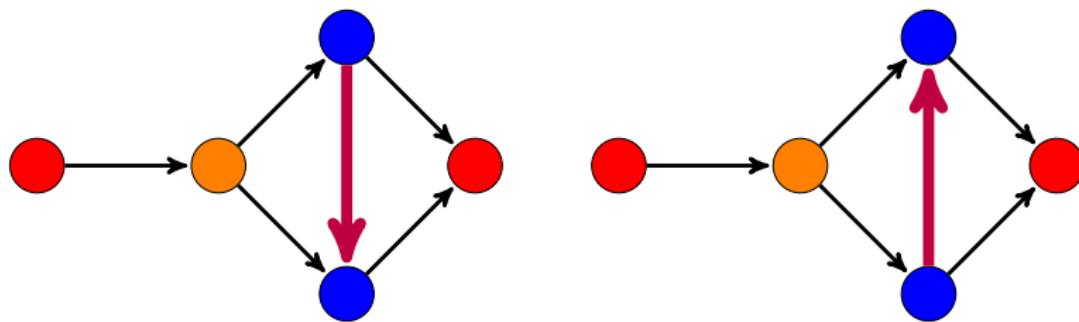


But loops between different states are not allowed

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:

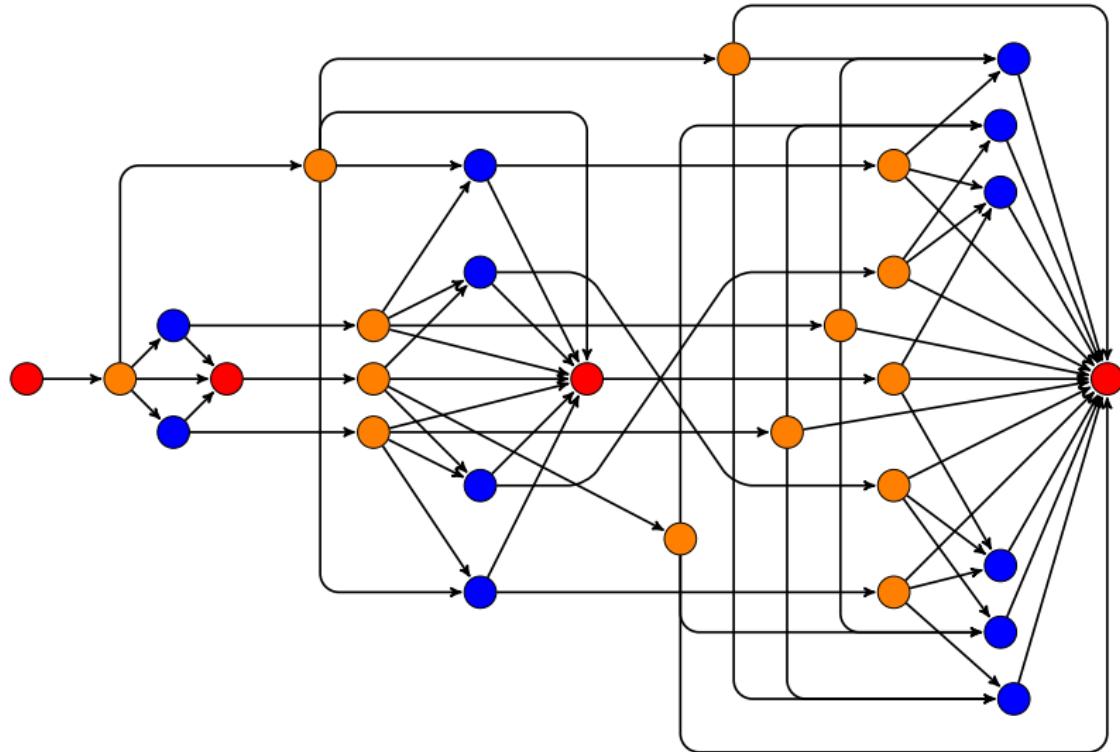
- ① Every feasible Markovian strategy σ satisfies the no loop condition.
- ② Every pair of feasible Markovian strategies σ and σ' induce consistent partial orders on the state space.

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

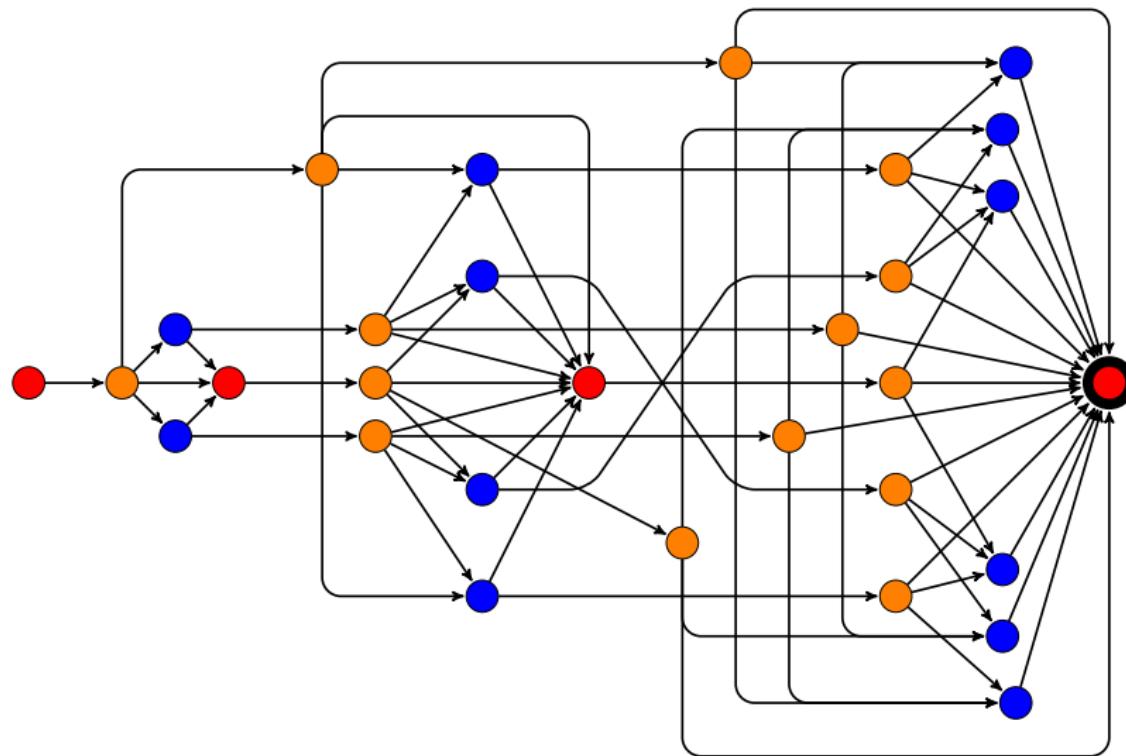
Strategy independent partial order on S

Coarsest common refinement of partial orders induced by all strategies



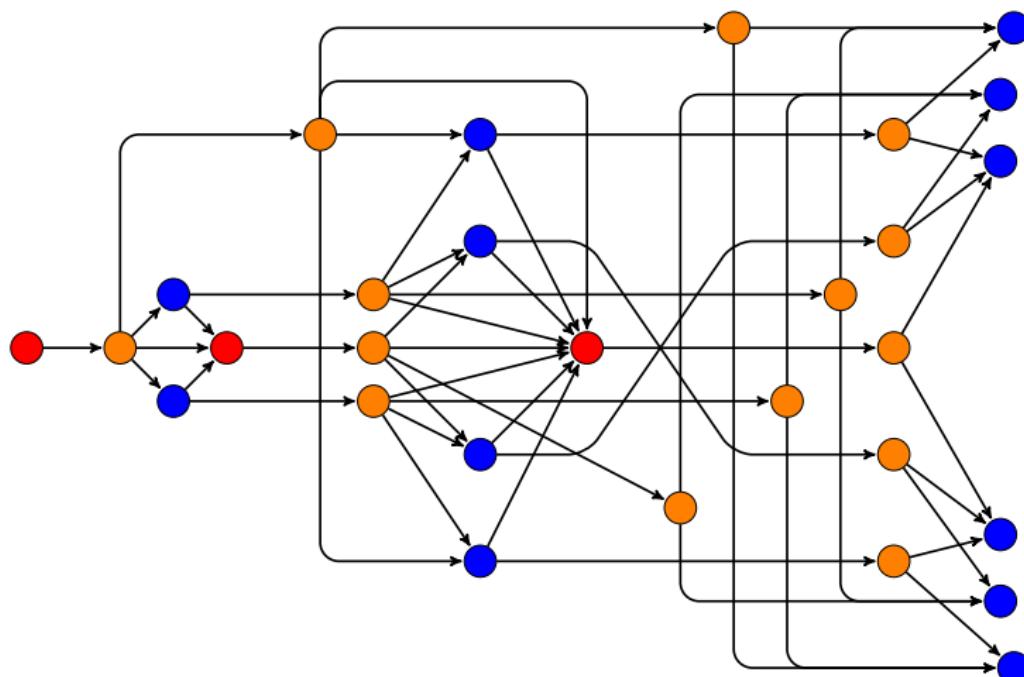
DAG recursion to partition S into stages

Identify terminal states



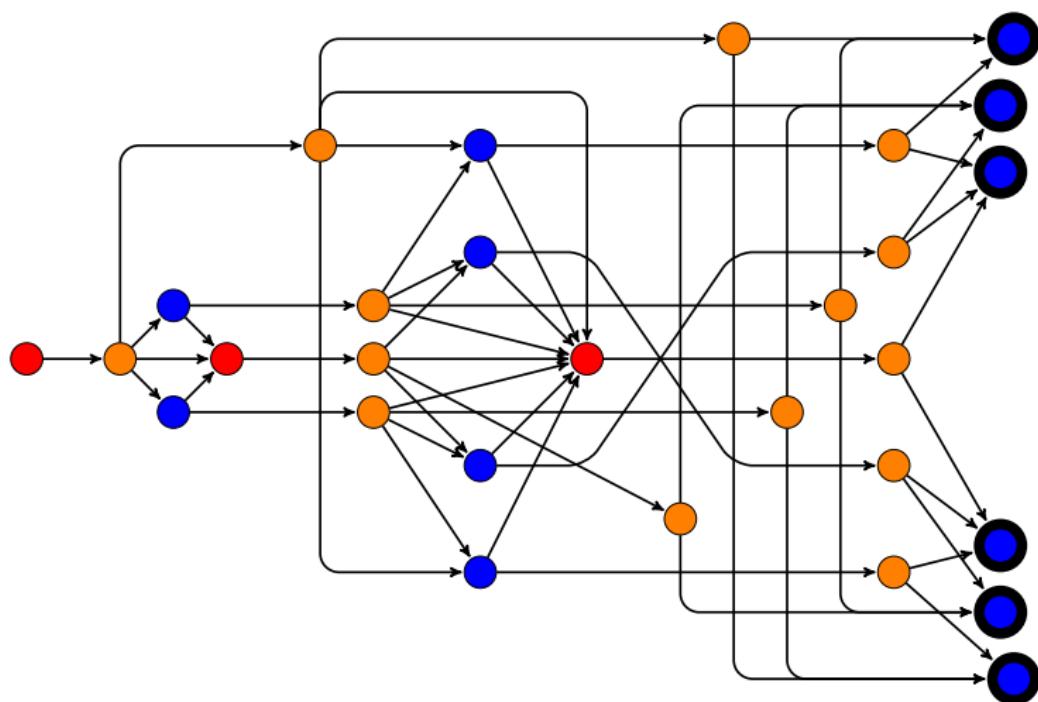
DAG recursion to partition S into stages

Remove terminal states



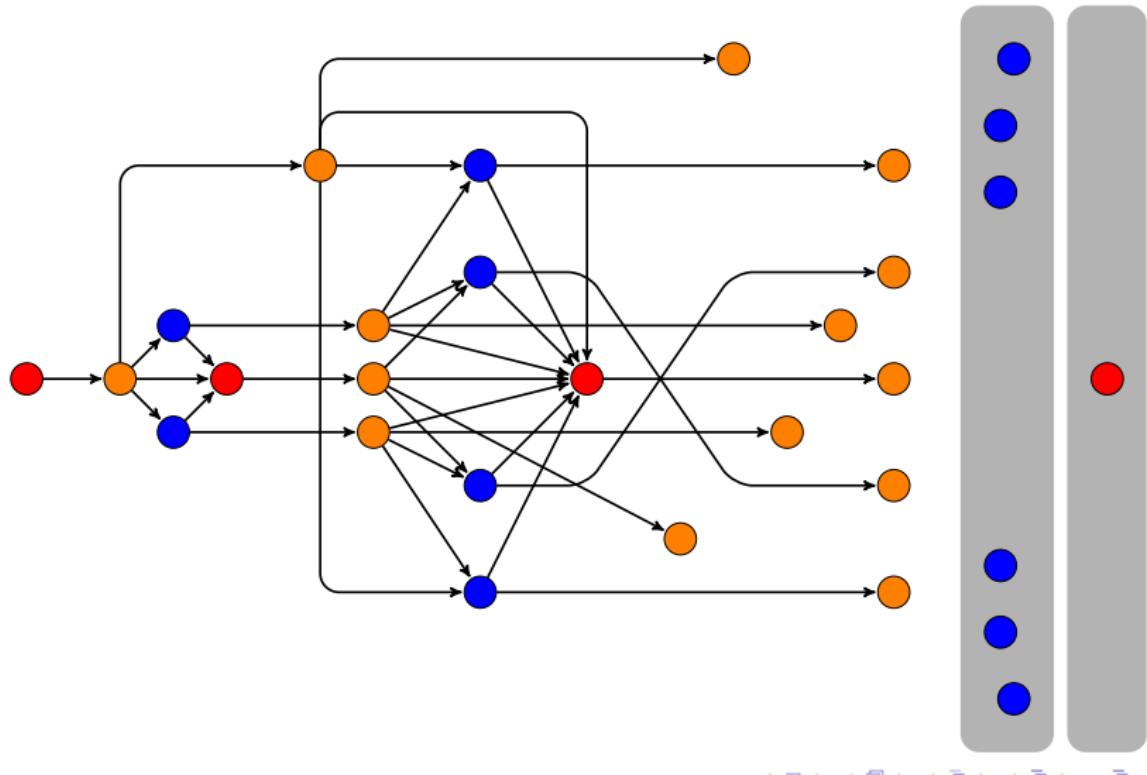
DAG recursion to partition S into stages

Identify terminal states



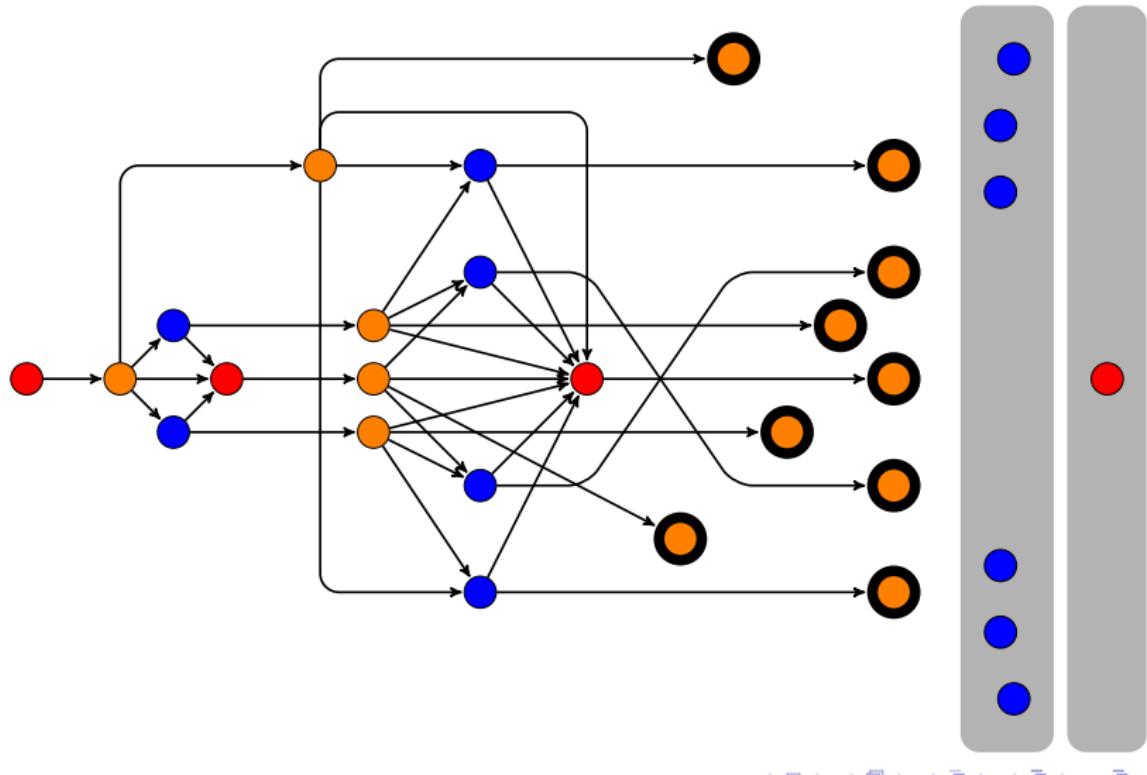
DAG recursion to partition S into stages

Remove terminal states



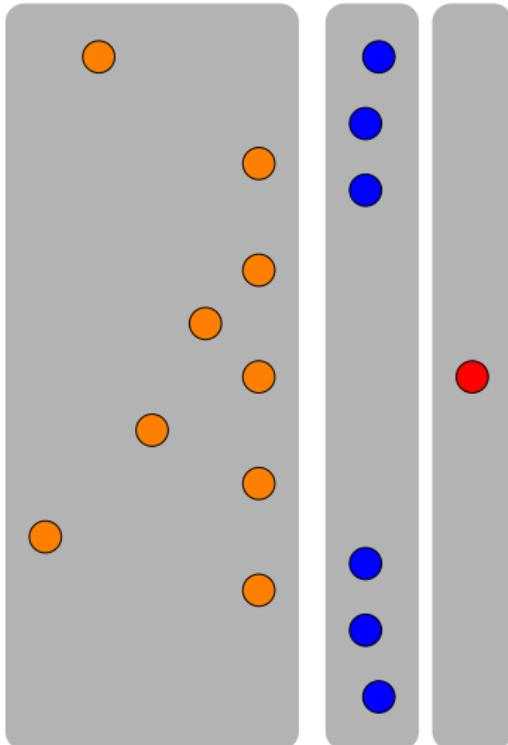
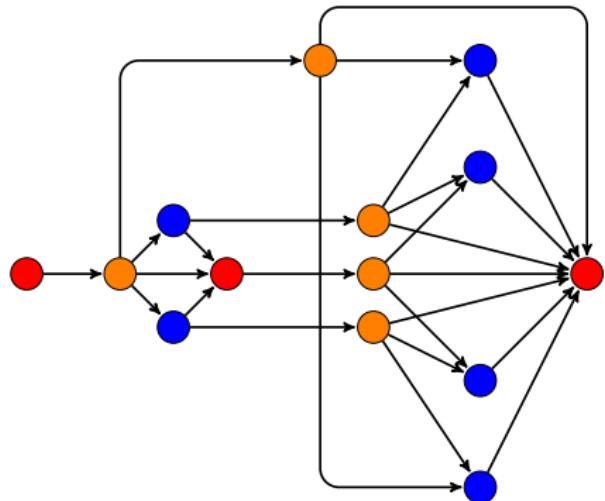
DAG recursion to partition S into stages

Identify terminal states



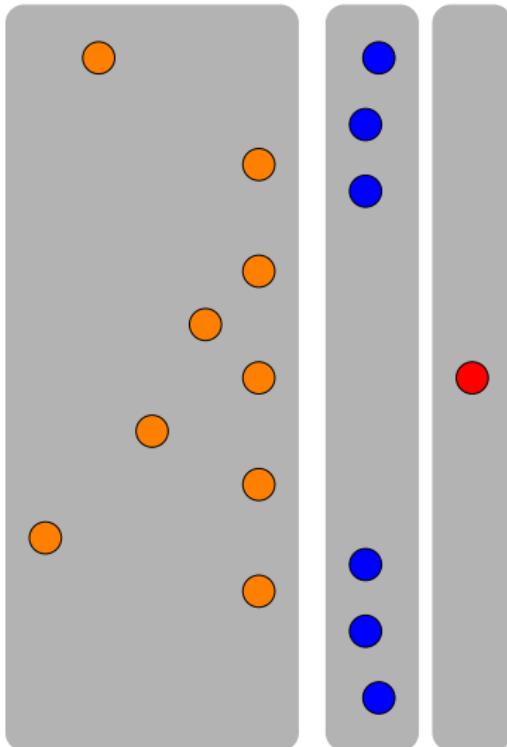
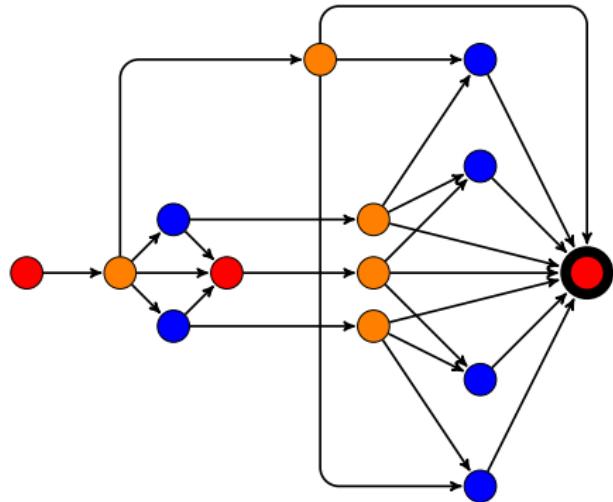
DAG recursion to partition S into stages

Remove terminal states



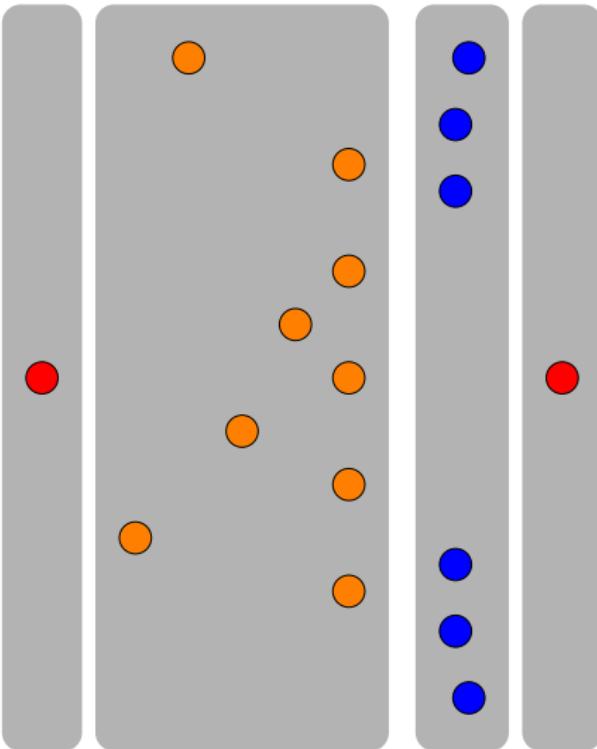
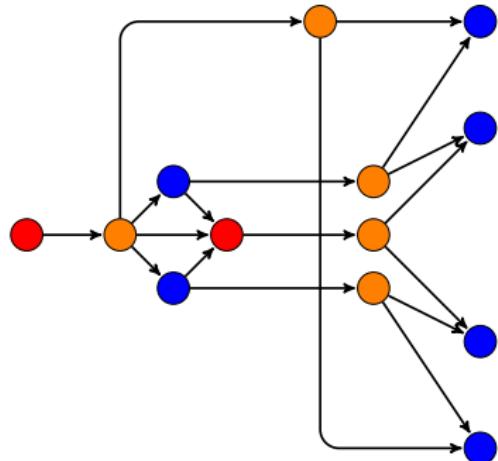
DAG recursion to partition S into stages

Identify terminal states



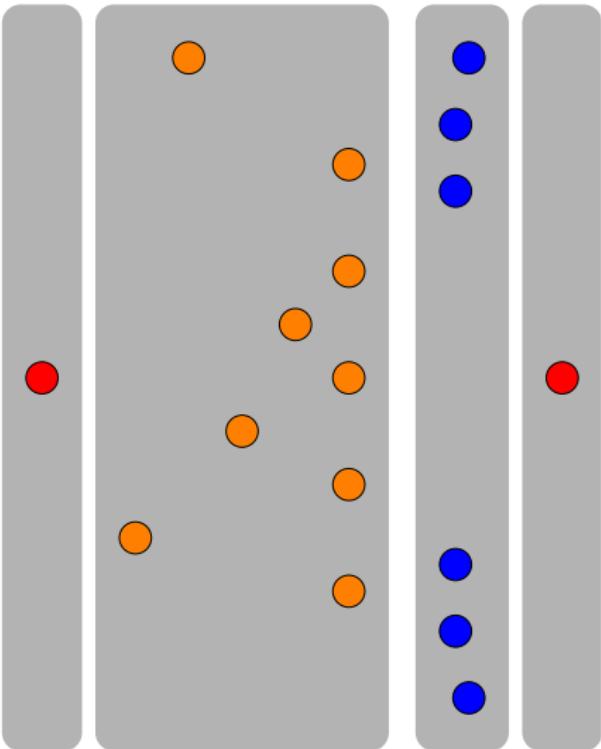
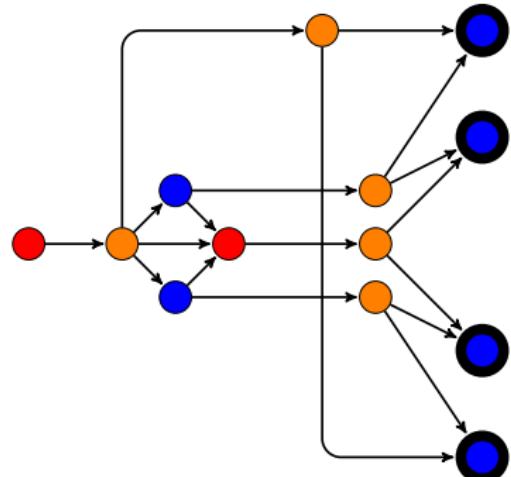
DAG recursion to partition S into stages

Remove terminal states



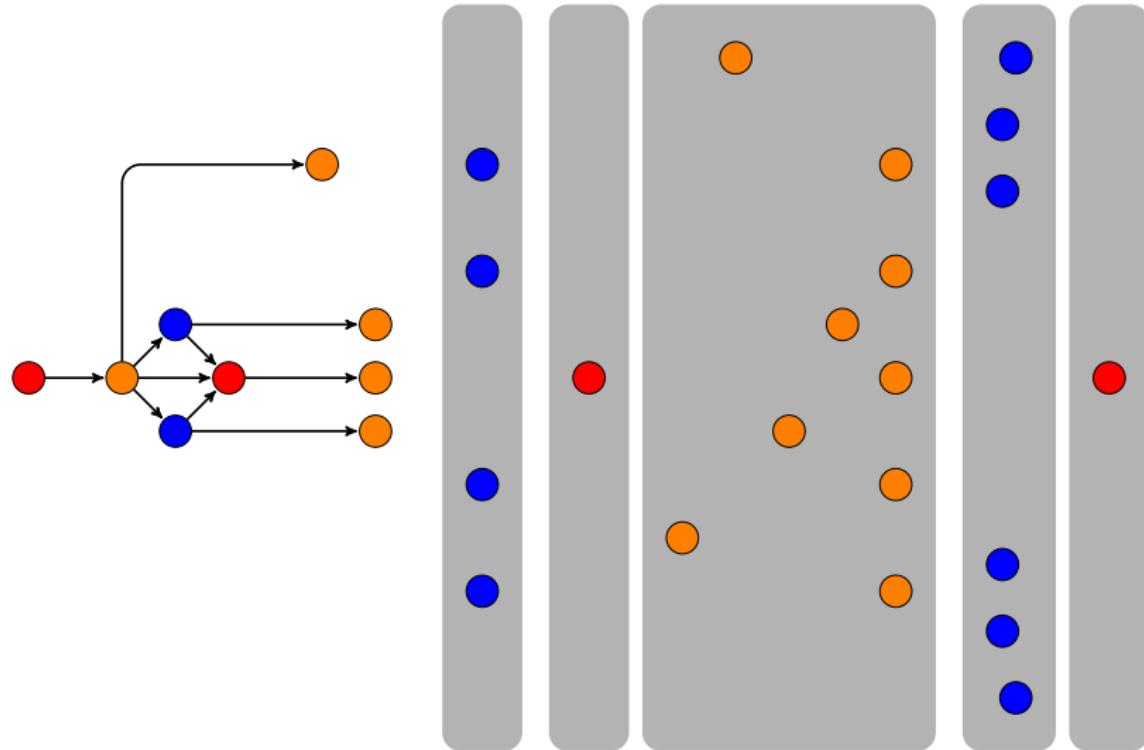
DAG recursion to partition S into stages

Identify terminal states



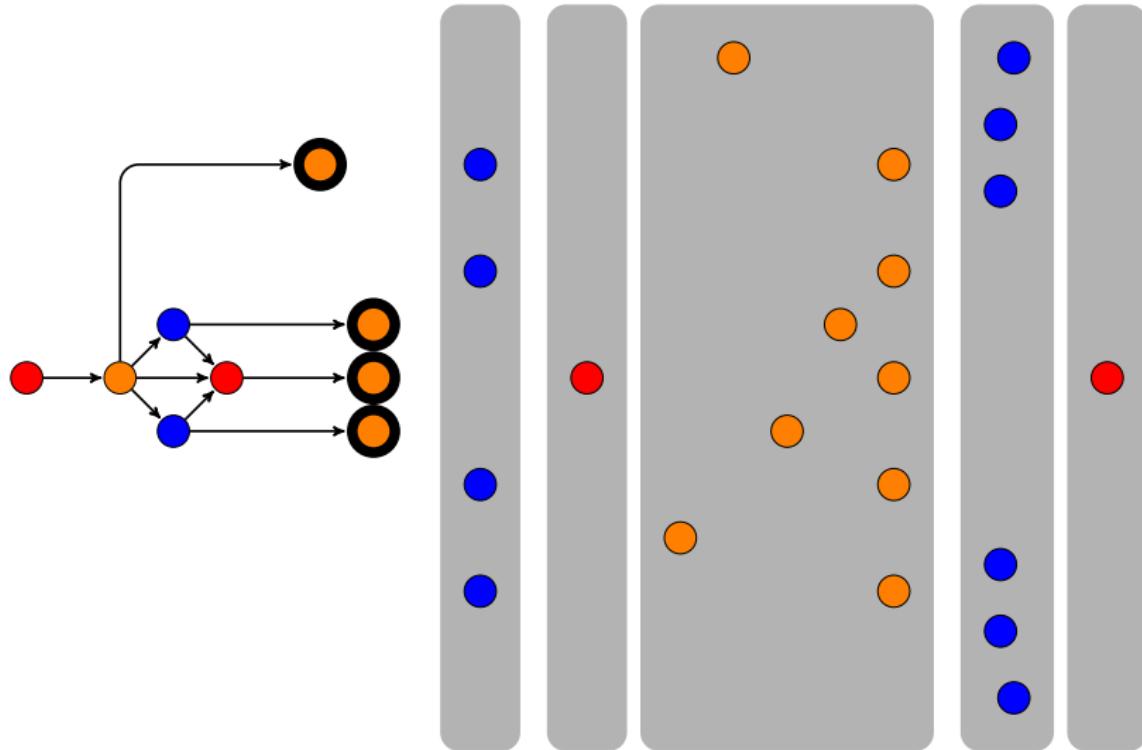
DAG recursion to partition S into stages

Remove terminal states



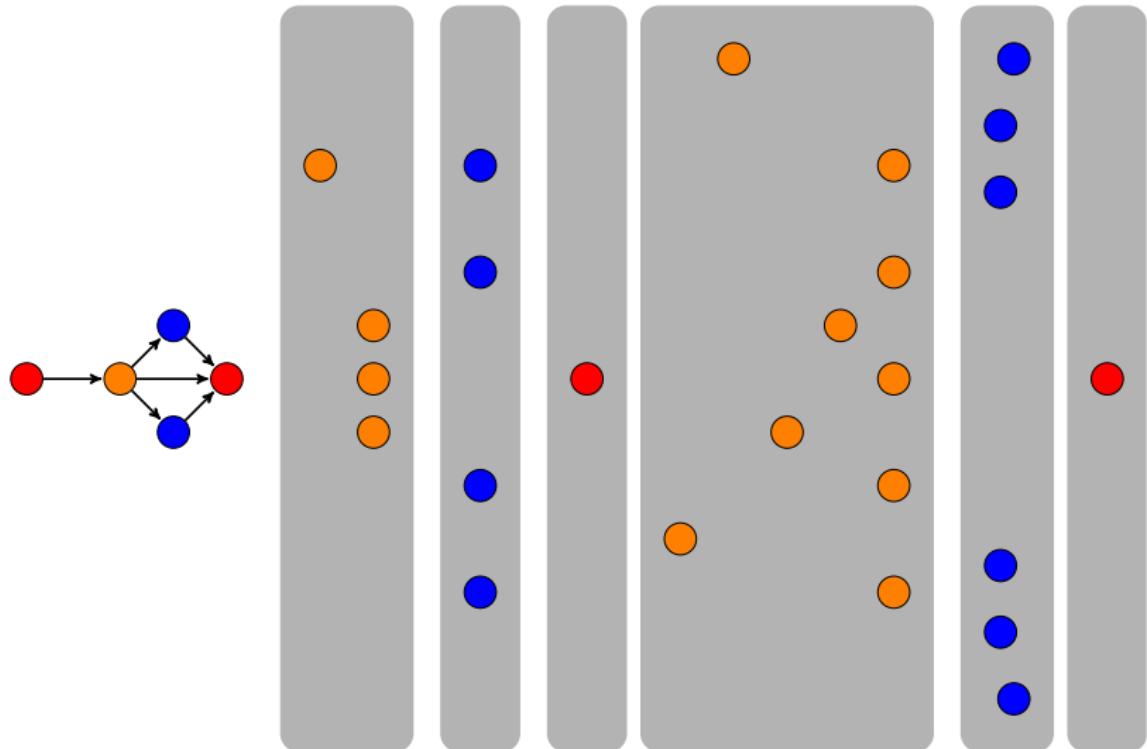
DAG recursion to partition S into stages

Identify terminal states



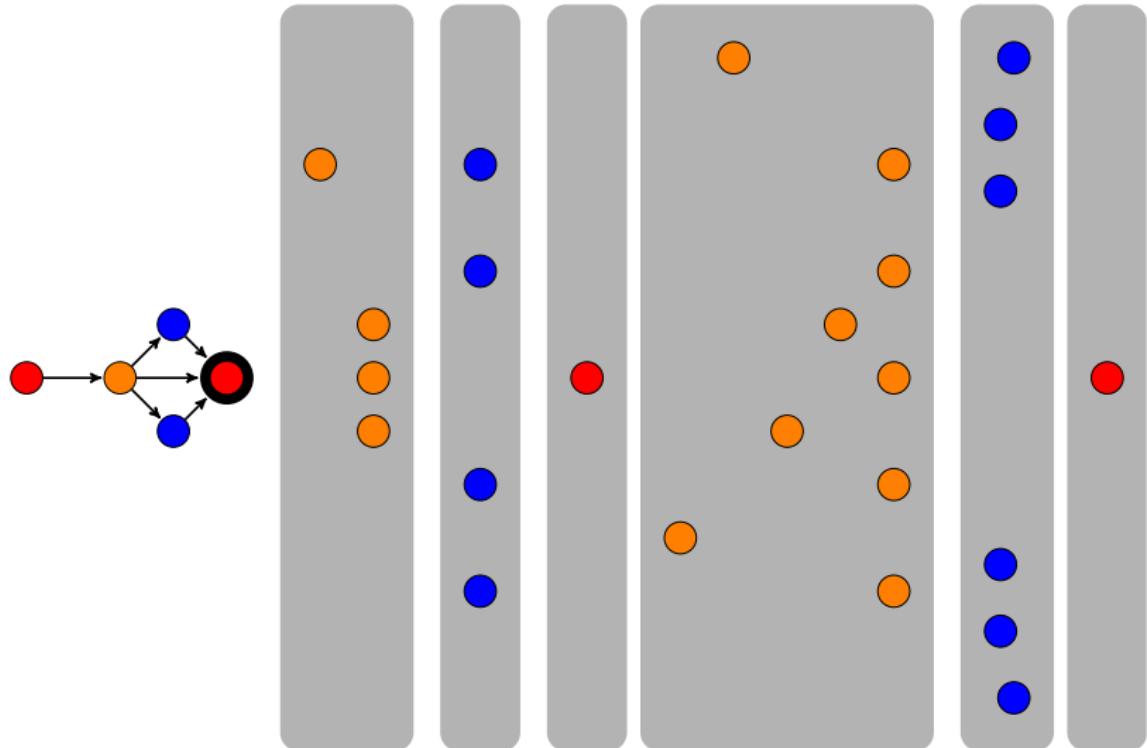
DAG recursion to partition S into stages

Remove terminal states



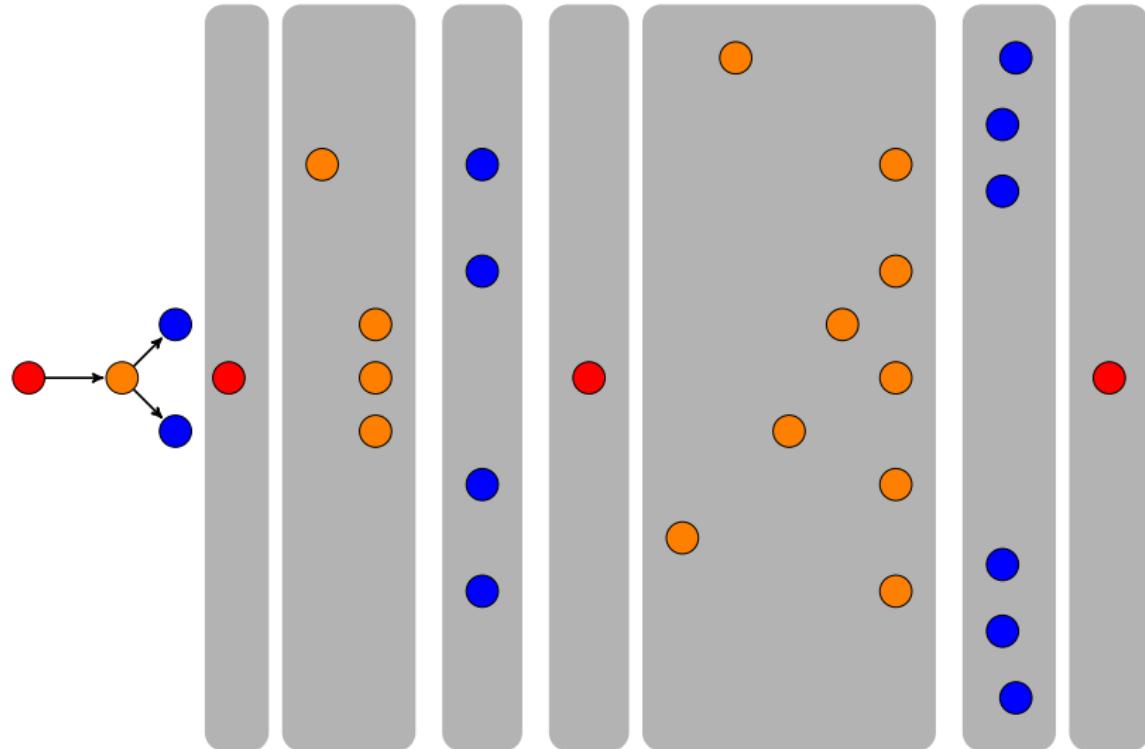
DAG recursion to partition S into stages

Identify terminal states



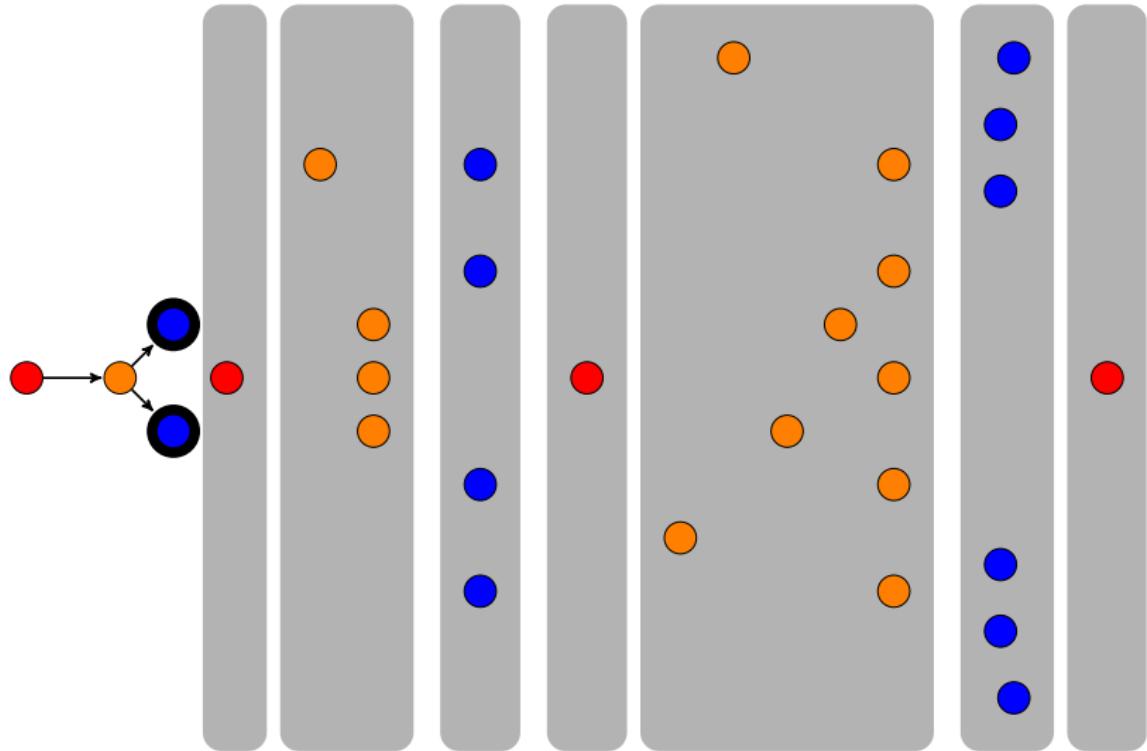
DAG recursion to partition S into stages

Remove terminal states



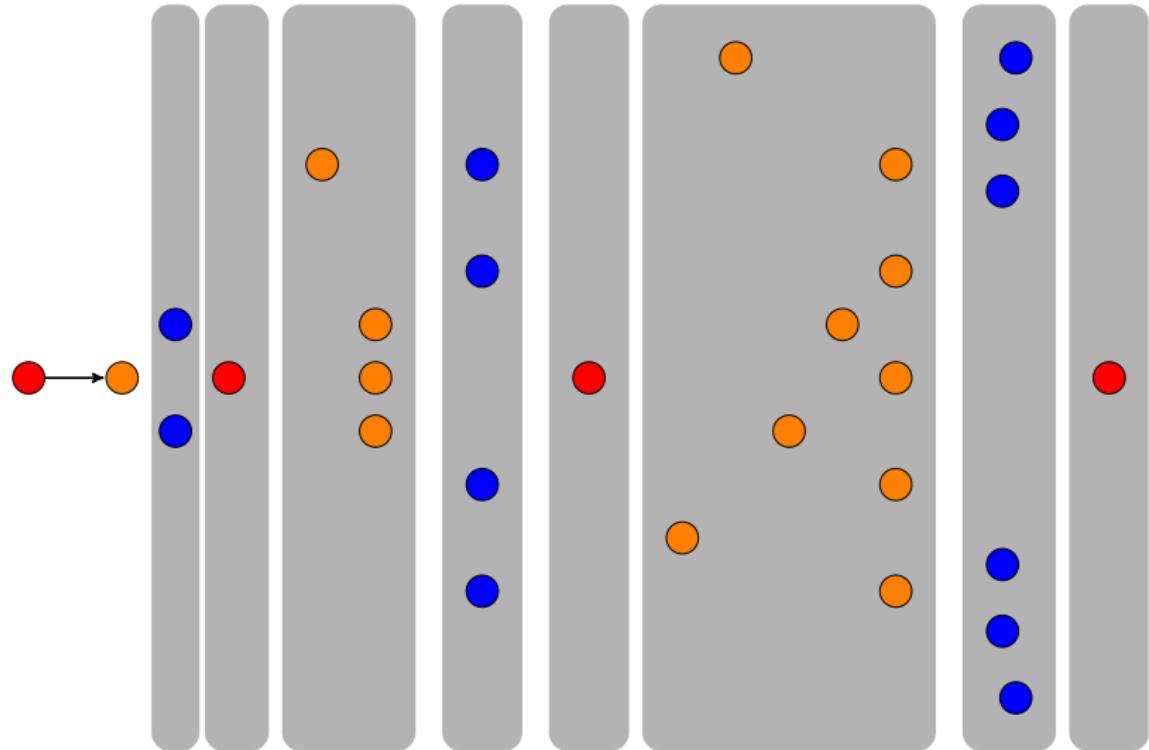
DAG recursion to partition S into stages

Identify terminal states



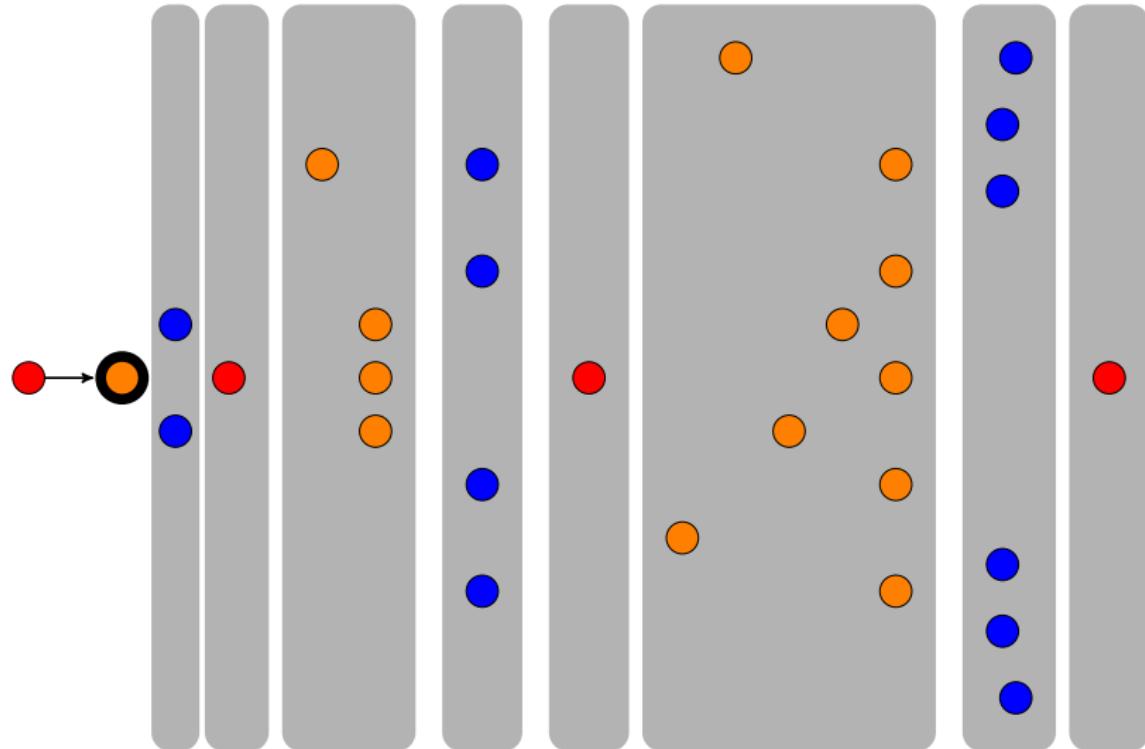
DAG recursion to partition S into stages

Remove terminal states



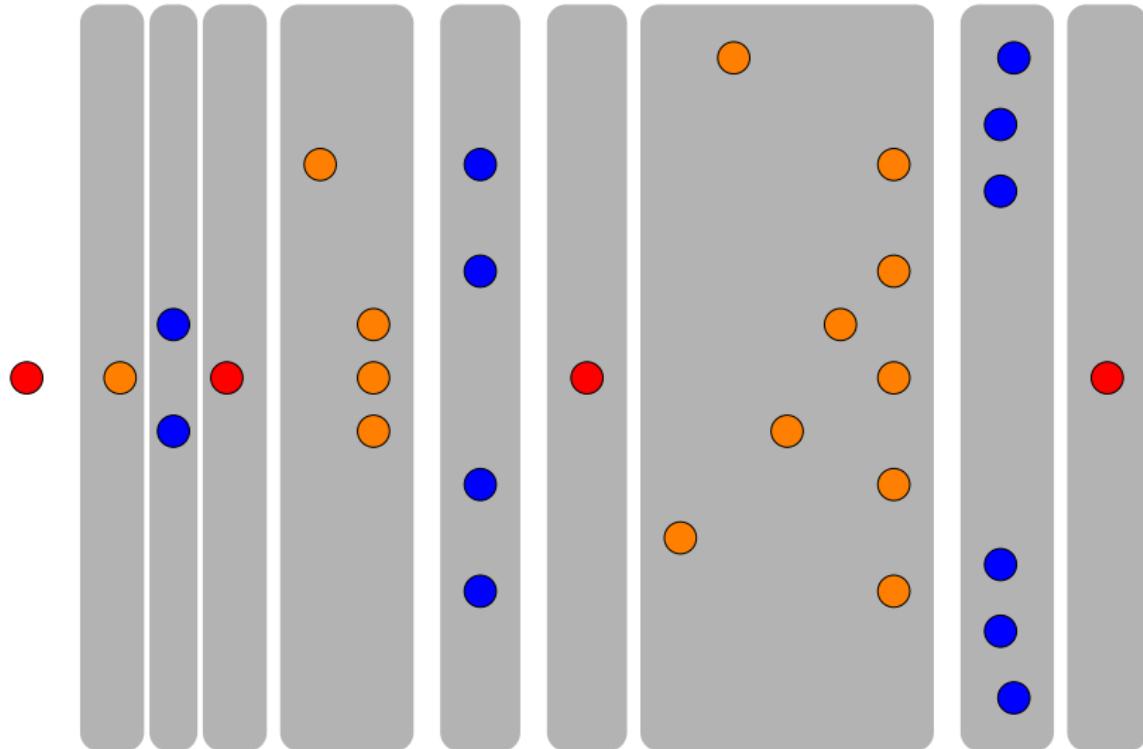
Resulting partition of S into stages

Identify terminal states



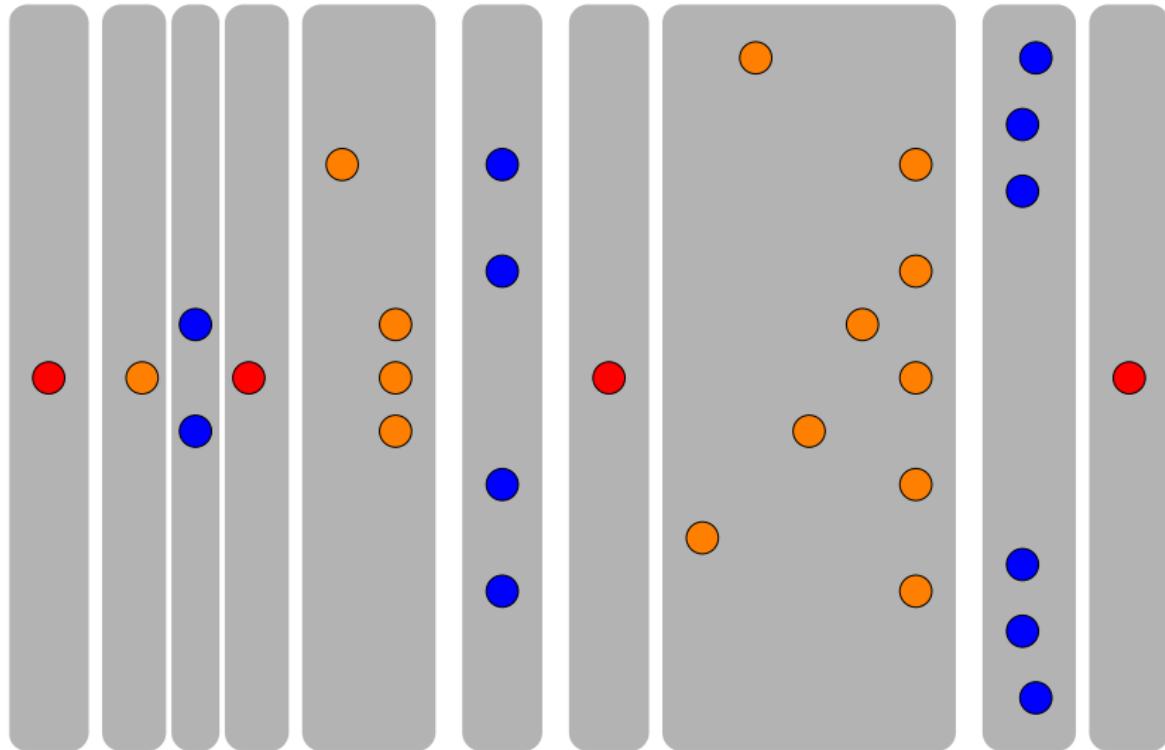
Resulting partition of S into stages

Remove terminal states



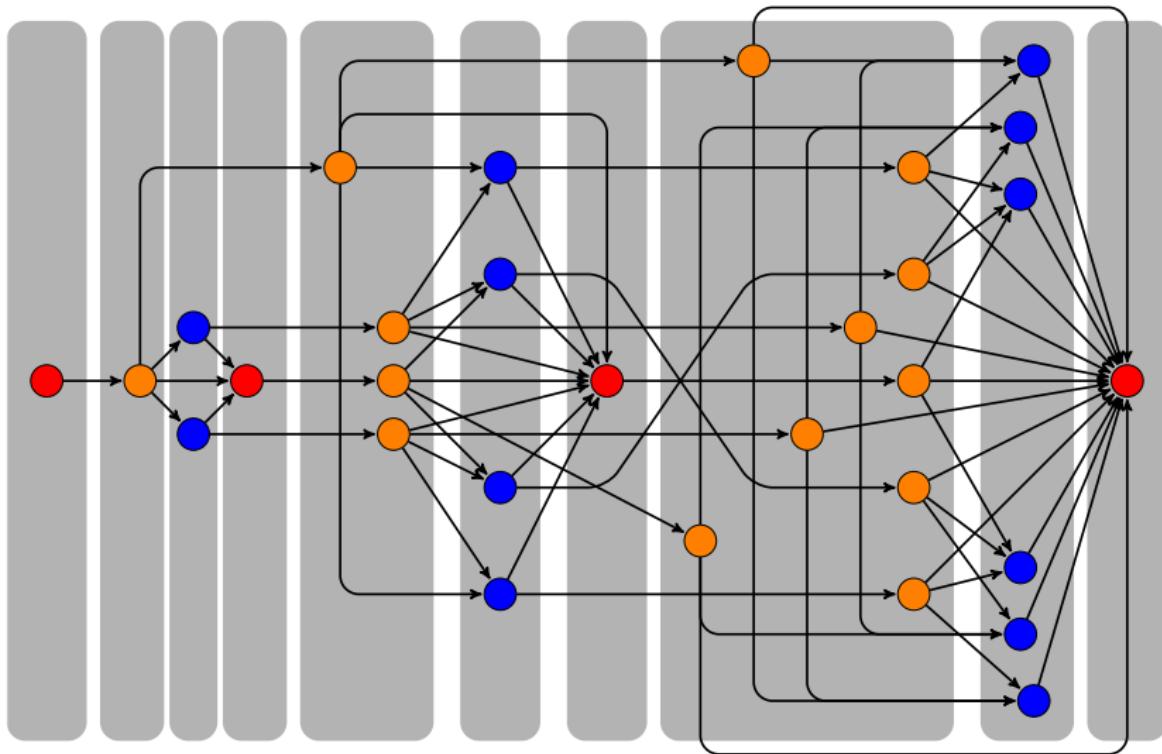
Resulting partition of S into stages

The stages of the game are totally ordered



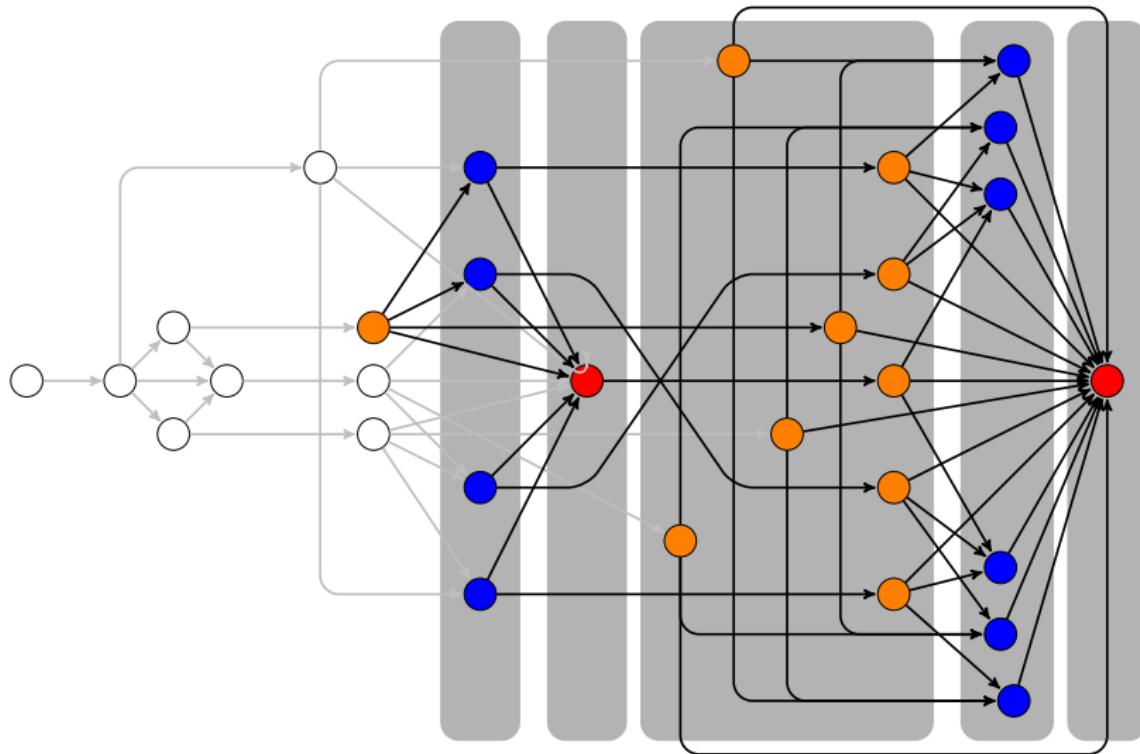
Subgames of DDG and continuation strategies

Subgames of DDG follow the order of stages



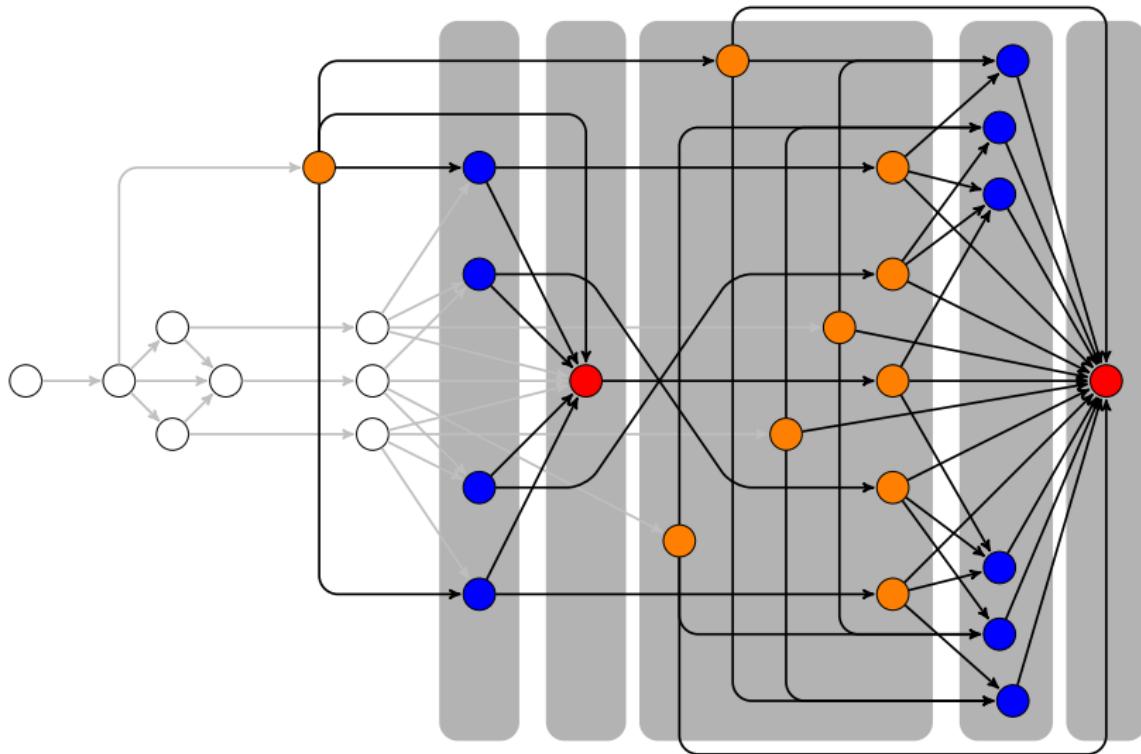
Subgames of DDG and continuation strategies

Solutions depends on later stages



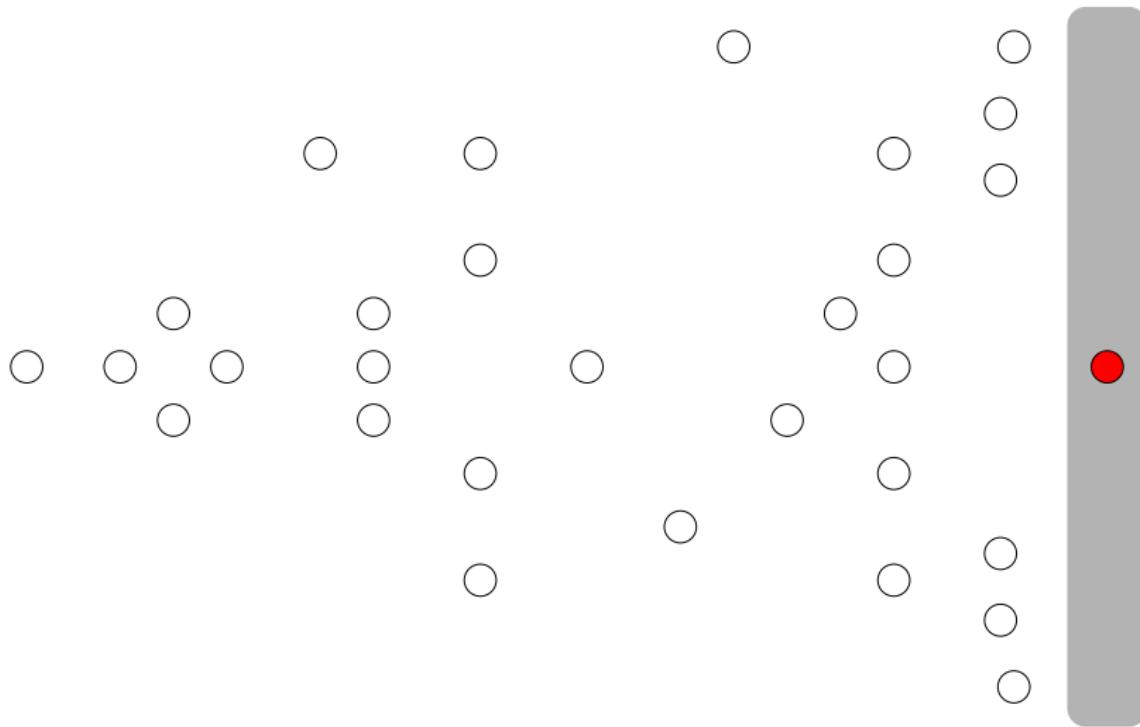
Subgames of DDG and continuation strategies

Within a given stage dependence is the same



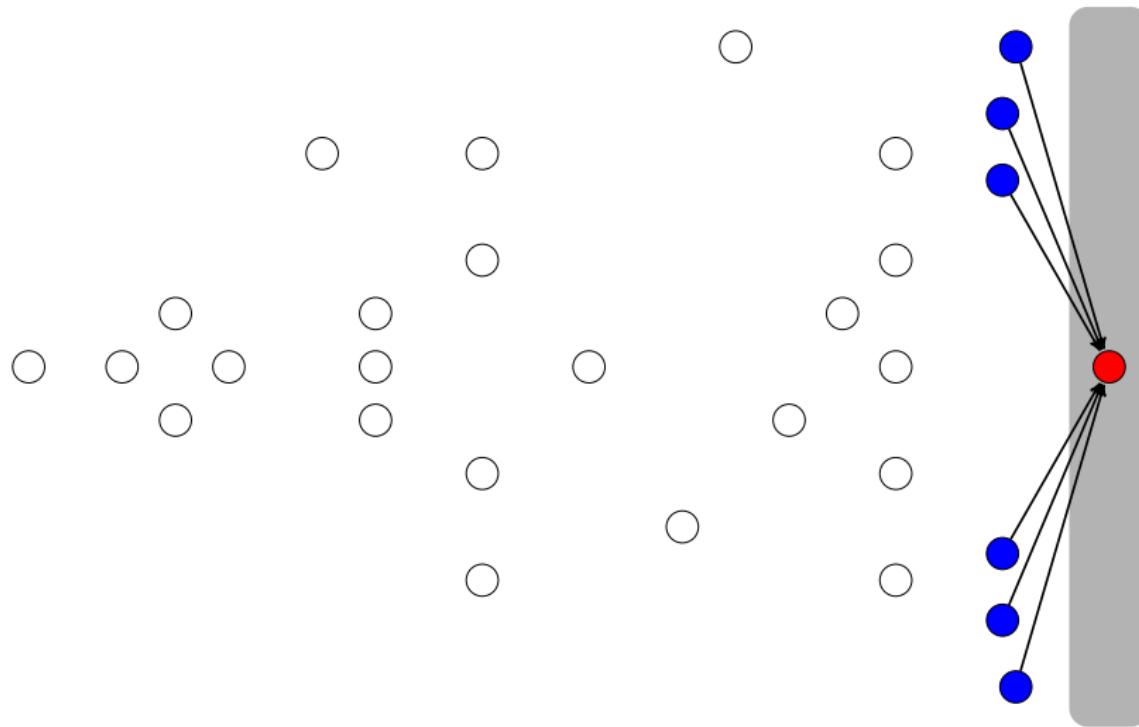
State recursion algorithm

Backward induction on stages of DDG



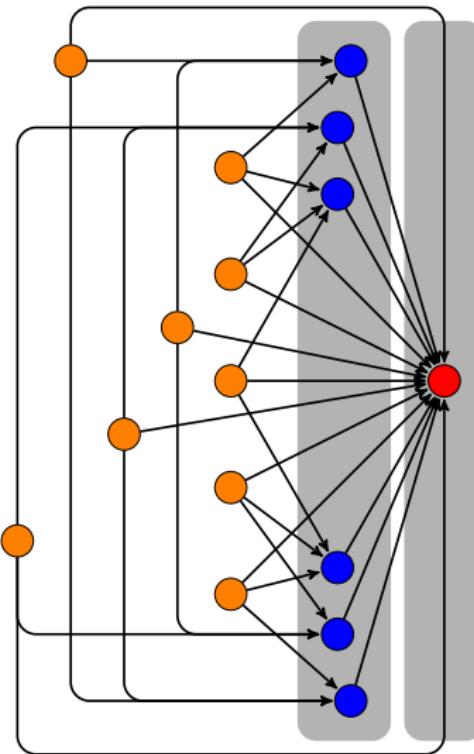
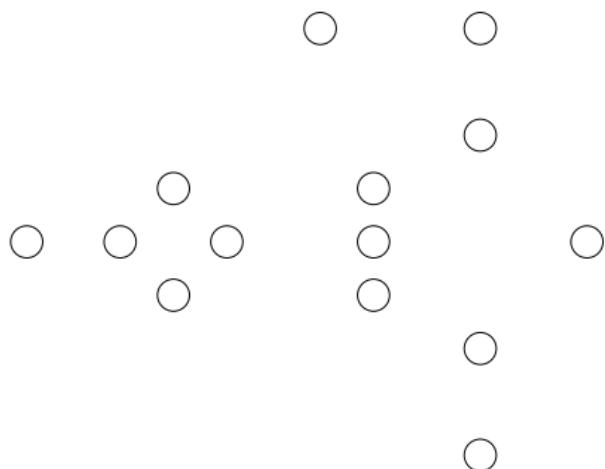
State recursion algorithm

Backward induction on stages of DDG



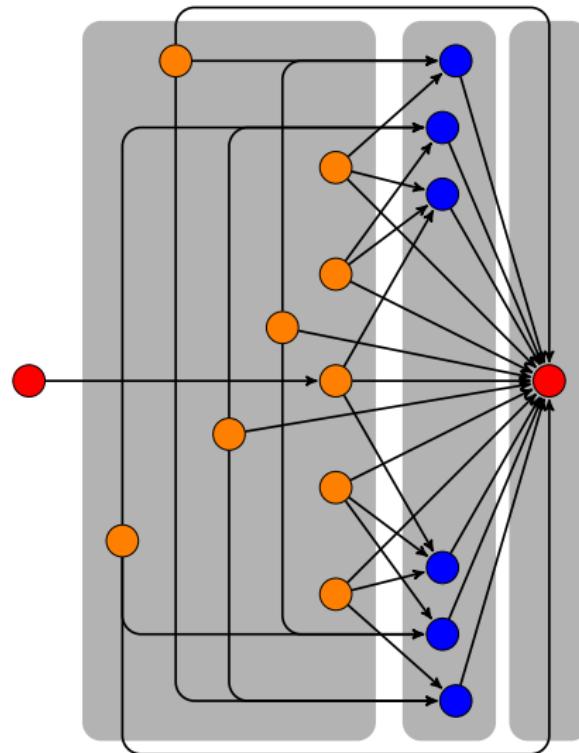
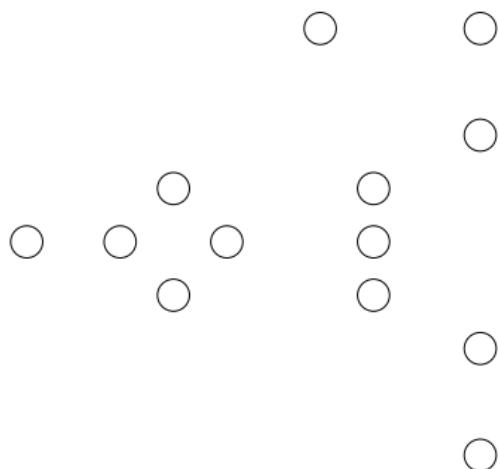
State recursion algorithm

Backward induction on stages of DDG



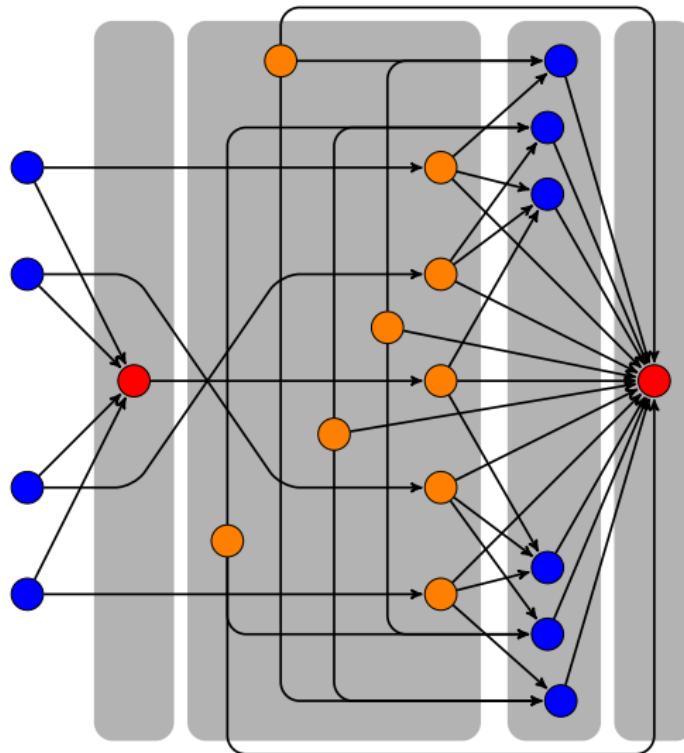
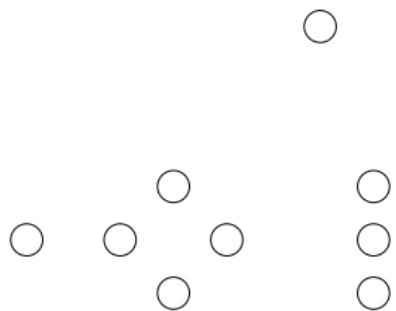
State recursion algorithm

Backward induction on stages of DDG



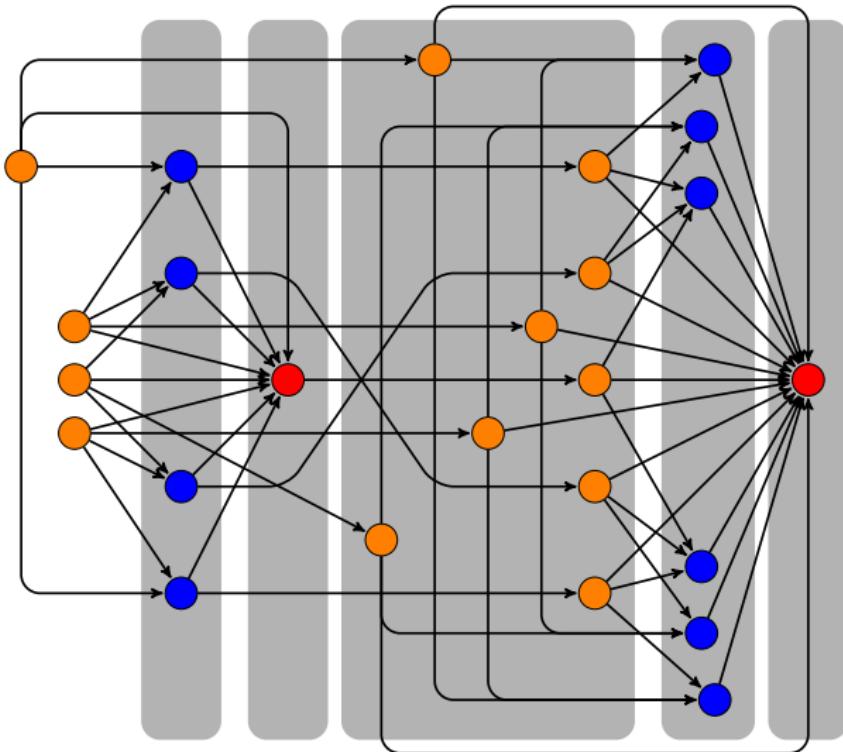
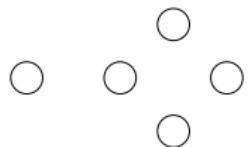
State recursion algorithm

Backward induction on stages of DDG



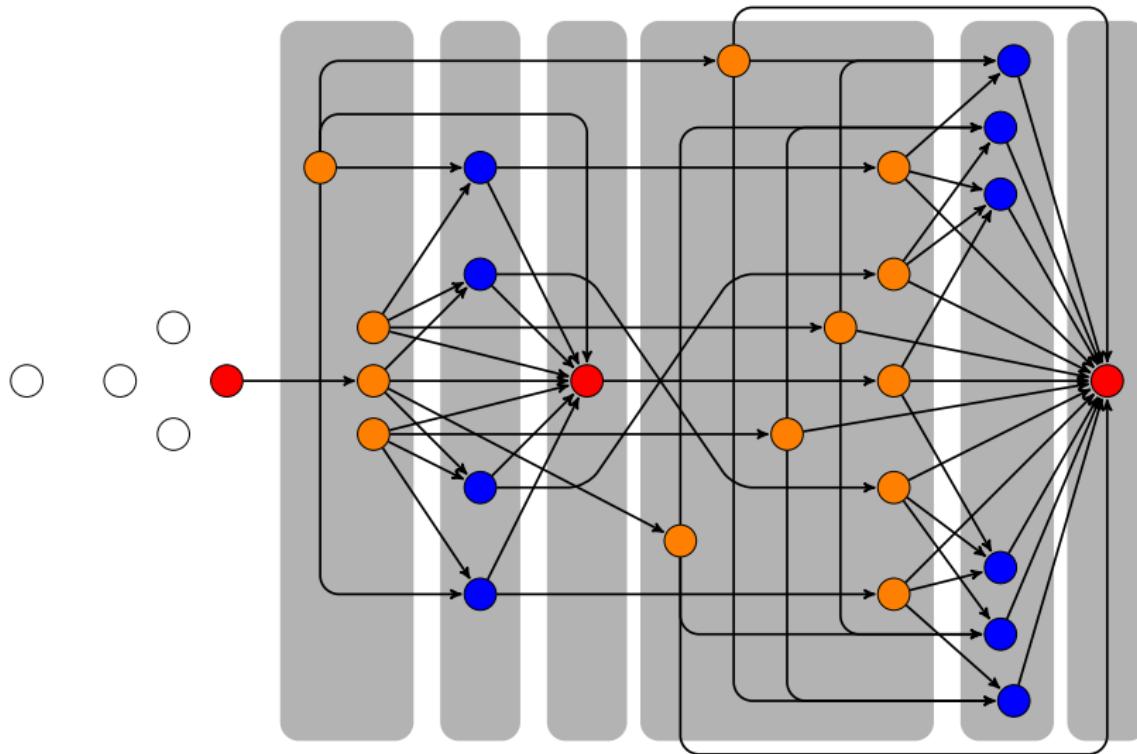
State recursion algorithm

Backward induction on stages of DDG



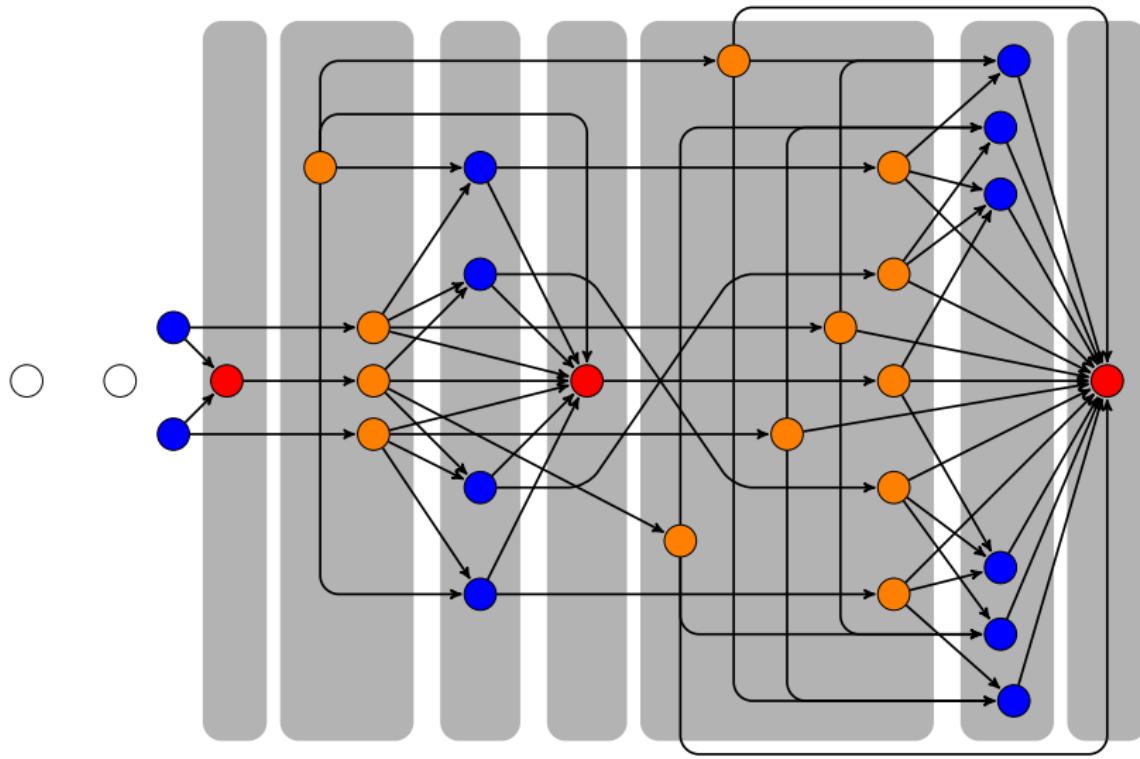
State recursion algorithm

Backward induction on stages of DDG



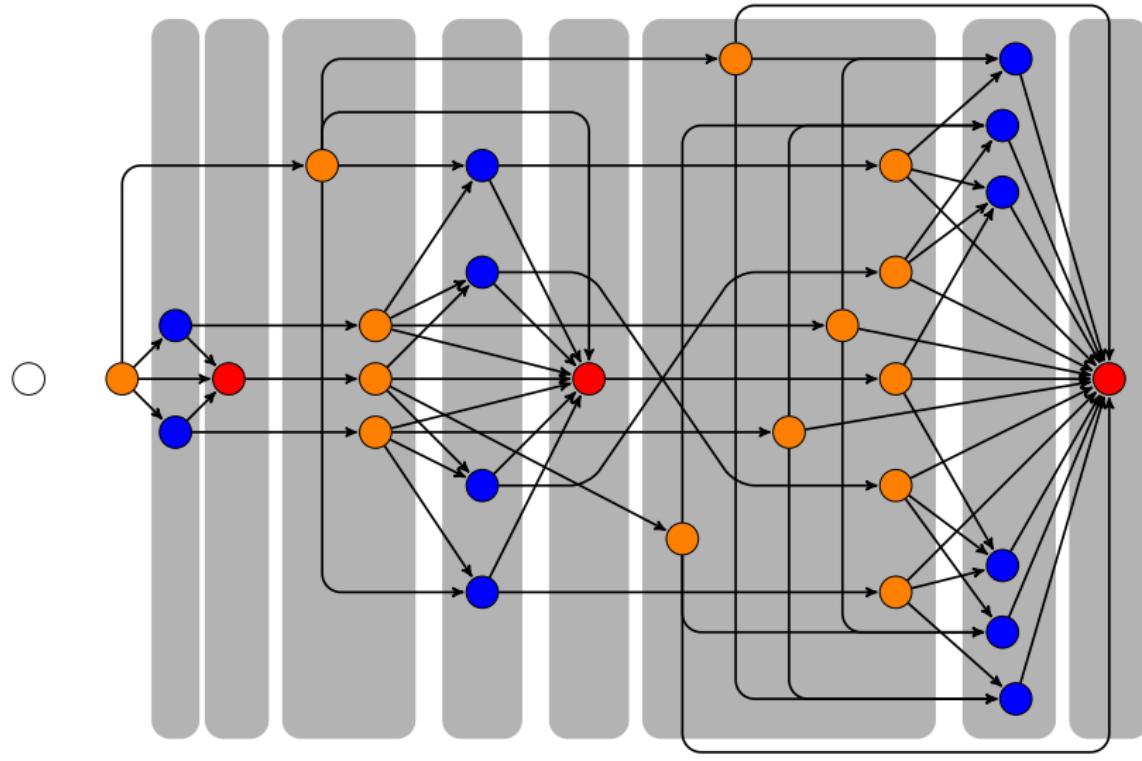
State recursion algorithm

Backward induction on stages of DDG



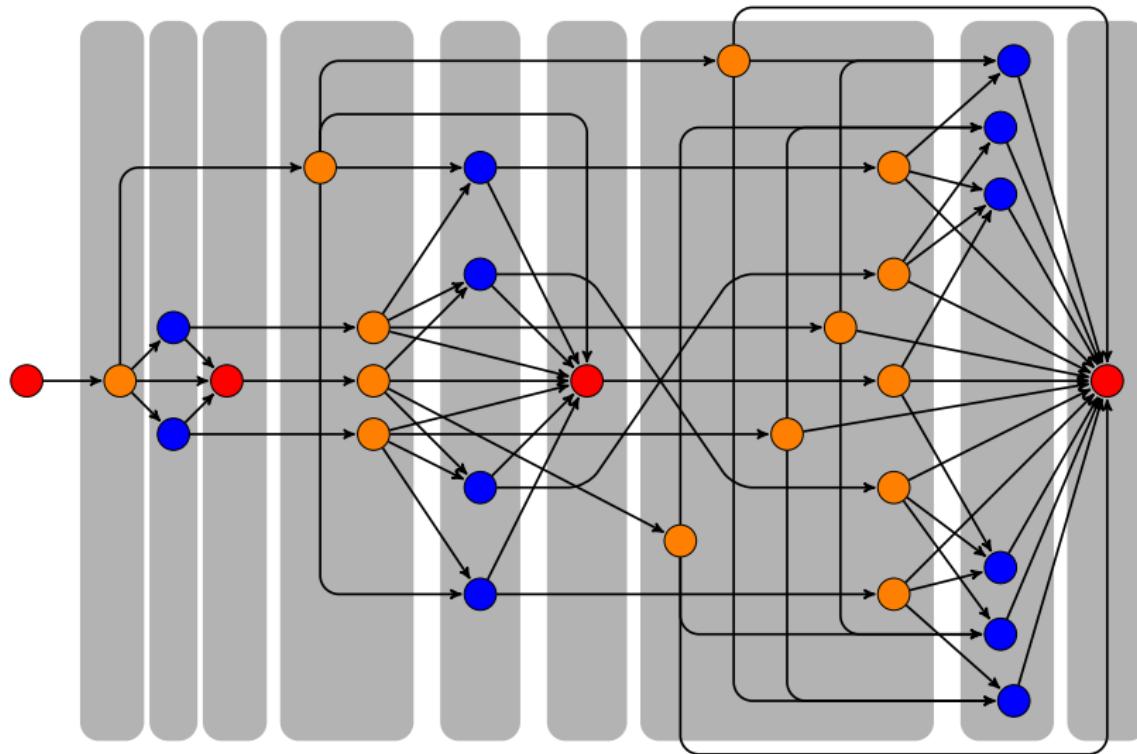
State recursion algorithm

Backward induction on stages of DDG



State recursion algorithm

Backward induction on stages of DDG

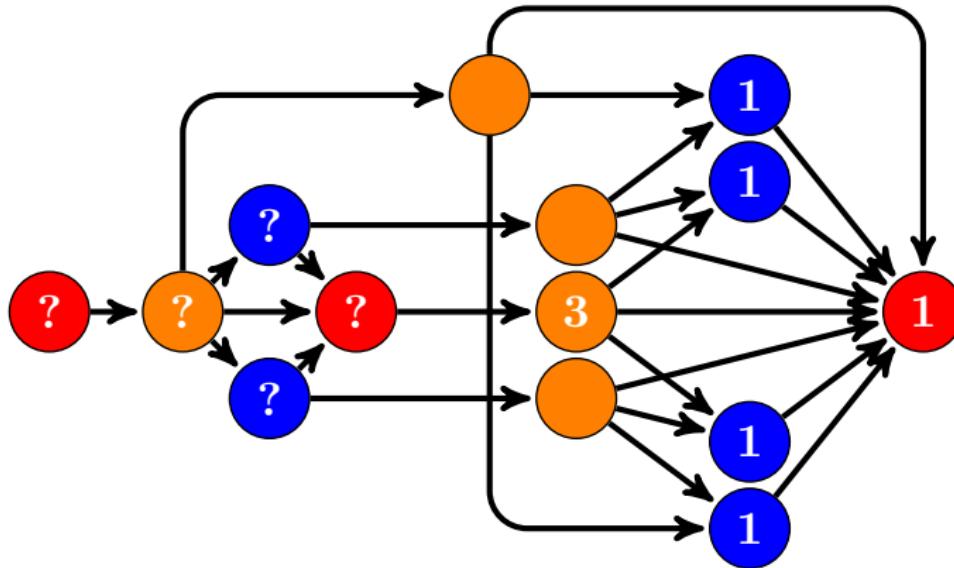


State Recursion versus Backward Induction

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

Multiplicity of equilibria

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**

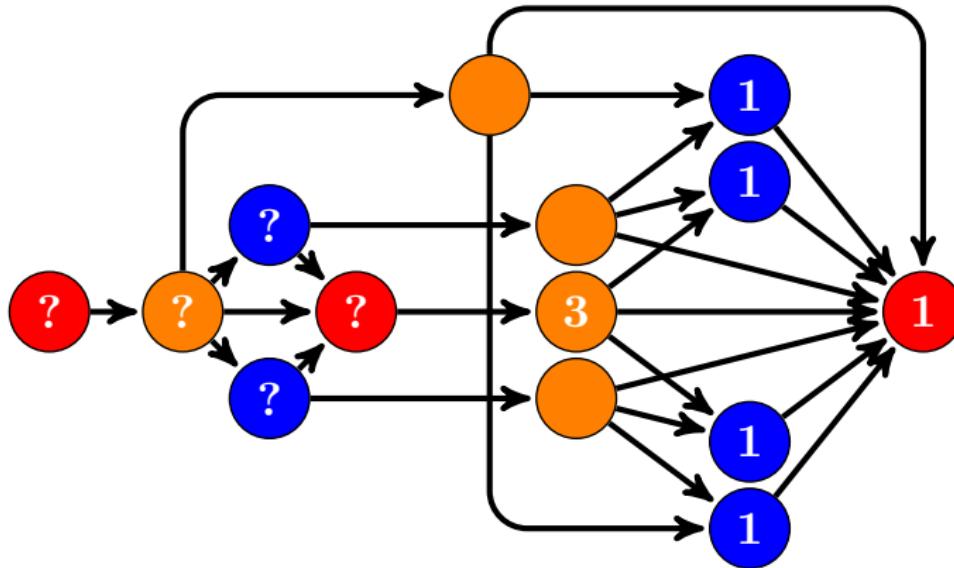
Challenge:

- 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 equilibria

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 $0, 1, \dots, K - 1$

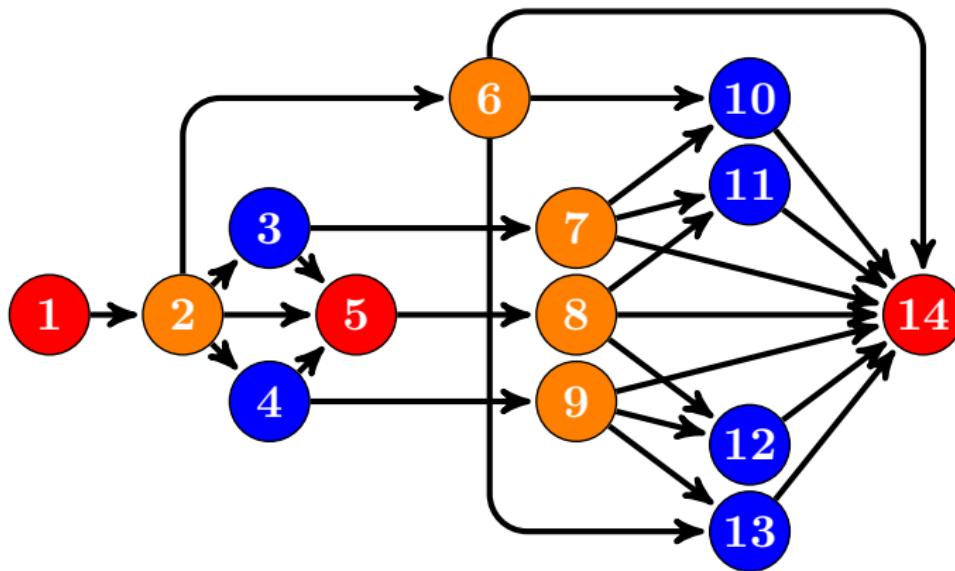
Dependence preserving property:

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)

corner														
edges							interior							
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
c	0	0	0	0	0	0	0	0	0	1	1	1	1	2
c1	0	0	0	2	1	2	2	1	1	1	1	2	2	2
c2	0	2	1	0	0	2	1	2	1	1	2	1	2	2

Digits of the ESR string mapped to the state space

Dependent preserving property



All possible ESR in lexicographic order

ESR string	c	e	e	e	e	i	i	i	i	c	e	e	i	c
	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	0	0	1
0	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	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	0	1	2
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	0	0	2	1
0	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	0	1	0	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	2	0
2	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	2

4,782,969

Recalculation of fesibility condition for new ESR

Avoid recalculation of subgames

ESR string	c	e	e	e	e	i	i	i	i	c	e	e	i	c
	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Nr of eqb	1	1	1	1	1	3	3	3	3	1	1	1	3	1
	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
	1	1	1	1	1	3	3	3	3	1	1	1	3	*

No changes in the solution of the game including the number of stage equilibria

Might have changed

Jumping over blocks of infeasibles ESRs

ESR string	c	e	e	e	e	i	i	i	i	c	e	e	i	c	Iteration:
	14	13	12	11	10	9	8	7	6	5	4	3	2	1	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	1	1	1	1	1	3	3	3	3	1	1	1	1	3	1a
	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	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2
	1	1	1	1	1	3	3	3	3	1	1	1	1	3	1a
	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	0	0	2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
	1	1	1	1	1	3	3	3	3	1	1	1	1	3	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	2	1a
	0	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	0	0	3b
	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	3c
	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	0	0	3d
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	4
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	...

Variable base arithmetics

- Replace the base- K ($\text{mod}(K)$) 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 = $(0, \dots, 0)$
- ② Run State Recursion using the current ESR
- ③ Save the number of equilibria in every stage game as $ne(\text{ESR})$
- ④ Add 1 to the ESR in bases $ne(\text{ESR})$ 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

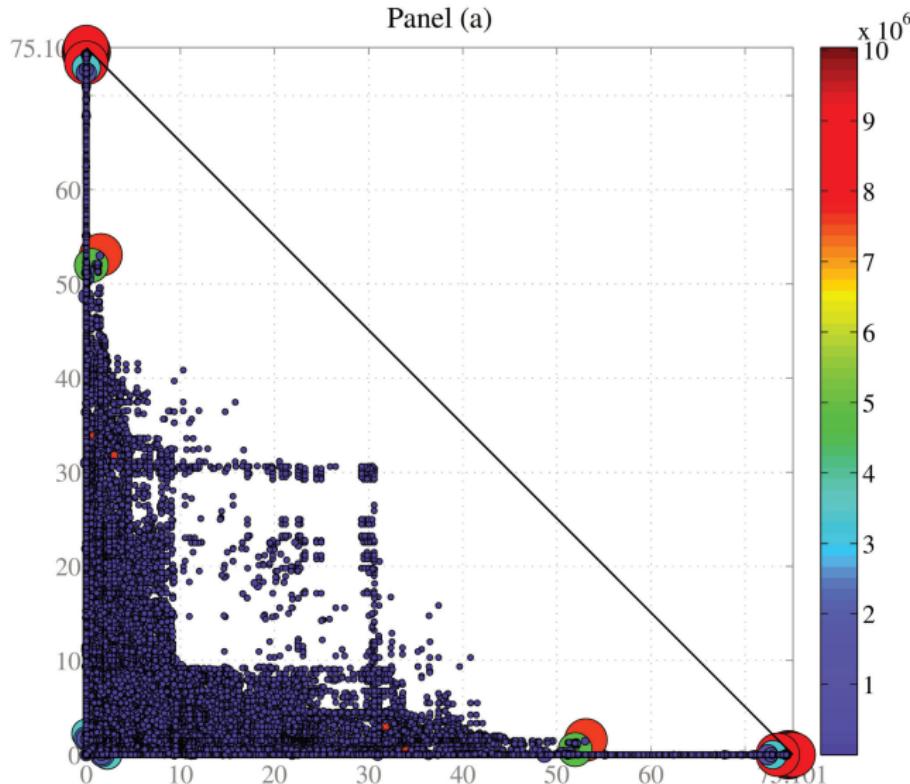
$K = 3$

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.

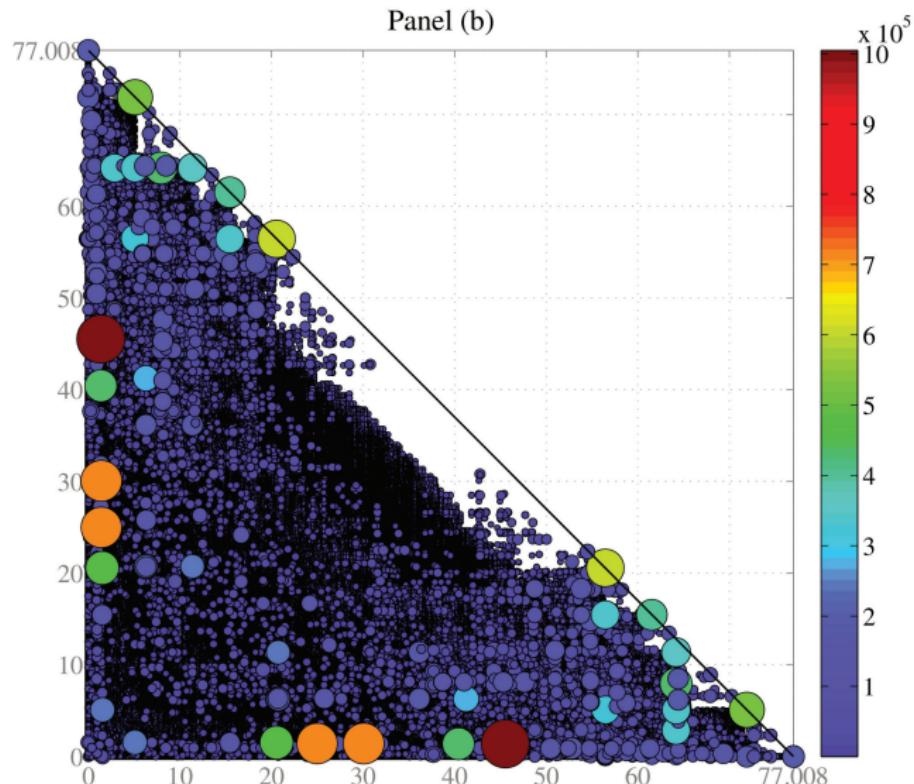
Road map of the talk

- 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)

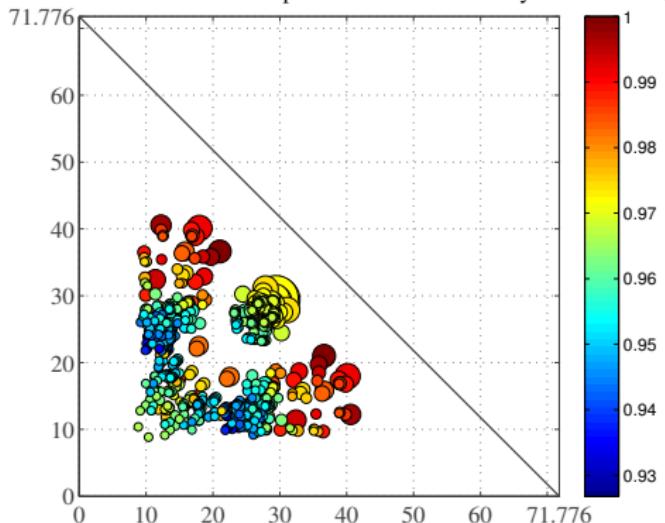


Pay-offs (stochastic tech progress)

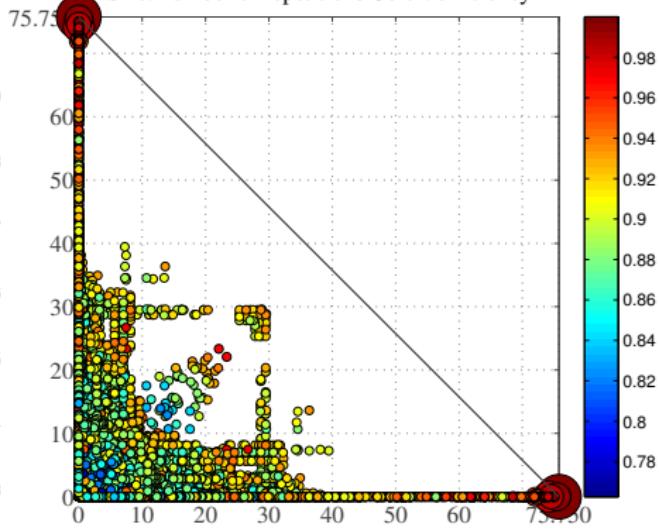


Pay-offs: alternating vs simultaneous move games

Panel (a): Non-monotonic tech. progress
17826 equilibria, 792 distinct pay-off points
Size: number of repetitions Color: efficiency

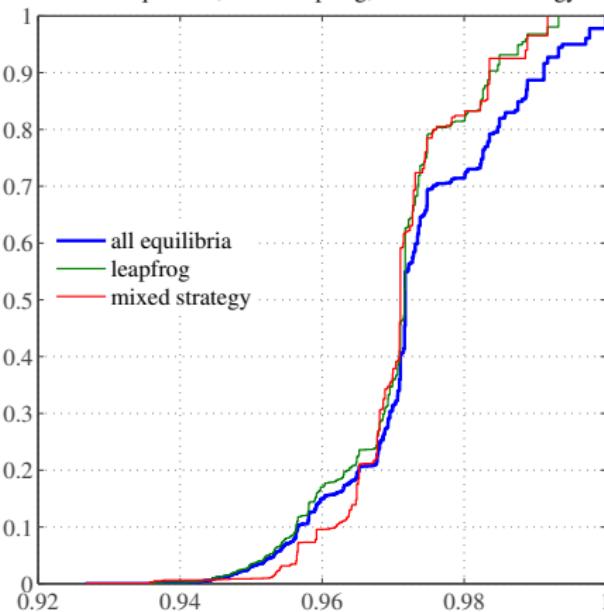


Panel (b): Simultaneous move
28528484 equilibria, 16510 distinct pay-off points
Size: number of repetitions Color: efficiency

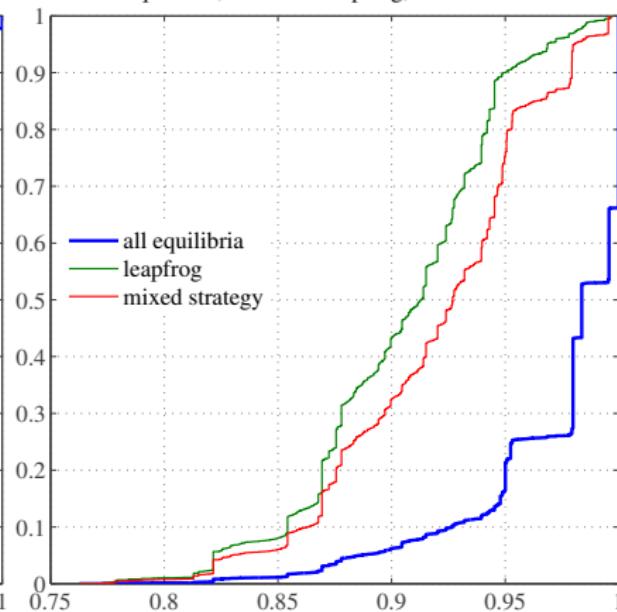


Efficiency: alternating vs simultaneous move games

Panel (c): Non-monotonic tech. progress
8913 equilibria, 7817 leapfrog, 2752 mixed strategy

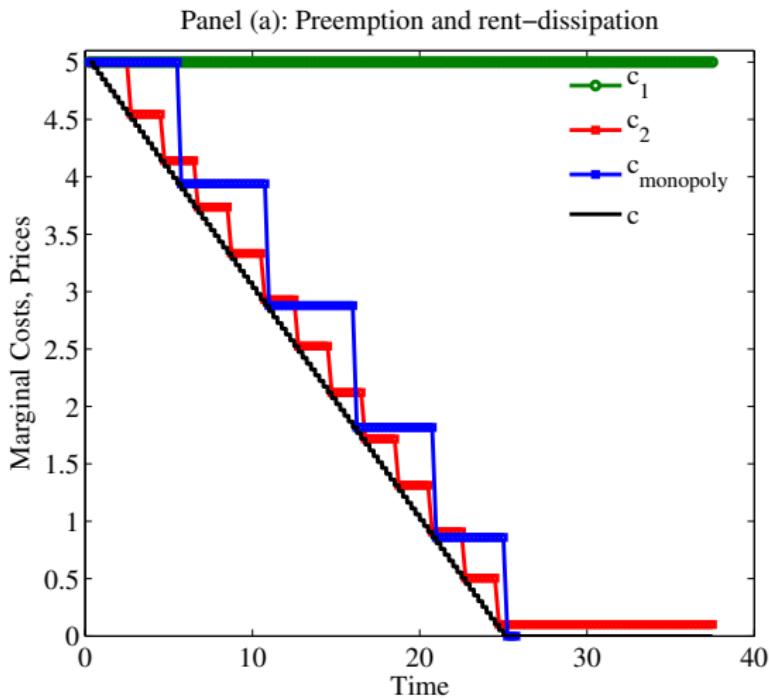


Panel (d): Simultaneous move
14264242 equilibria, 2040238 leapfrog, 2730910 mixed strategy



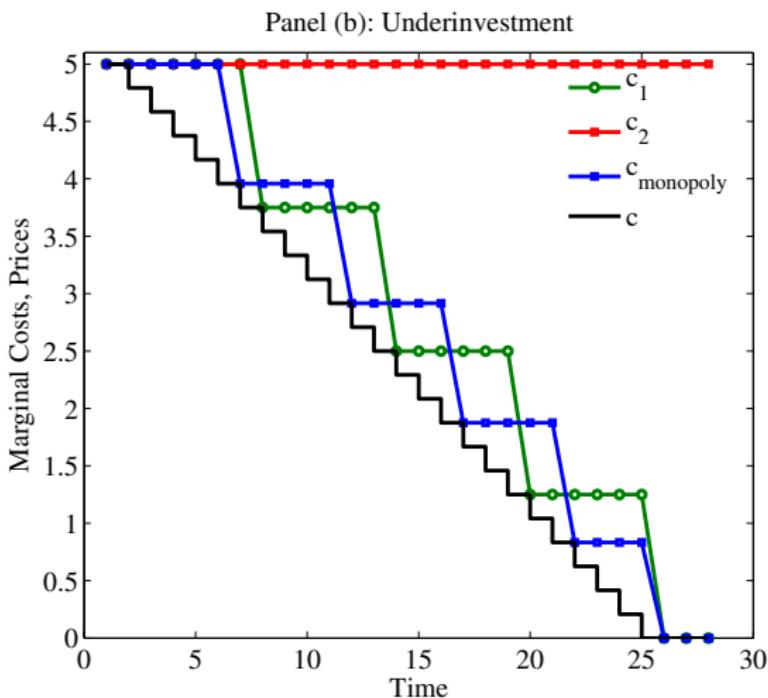
Full preemption and rent dissipation

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



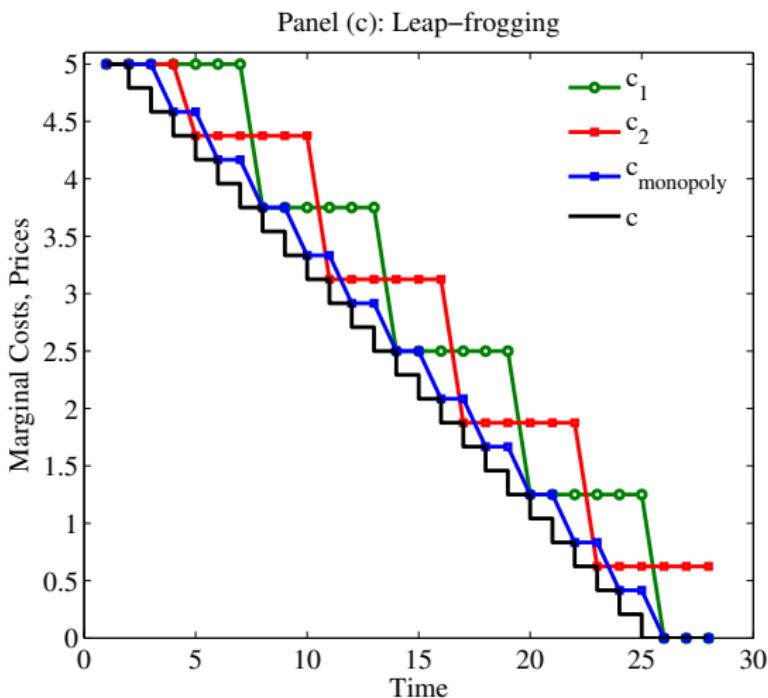
Underinvestment

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



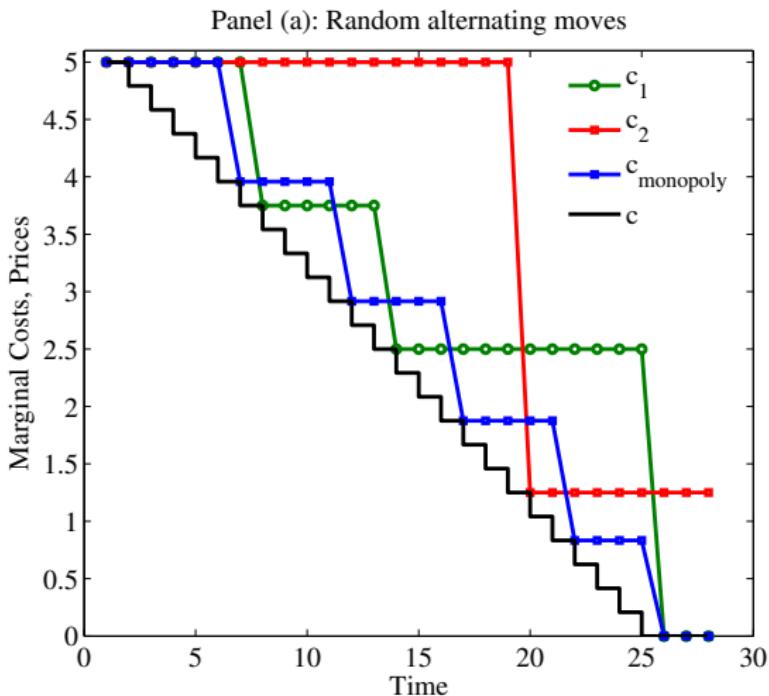
Leap-frogging

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



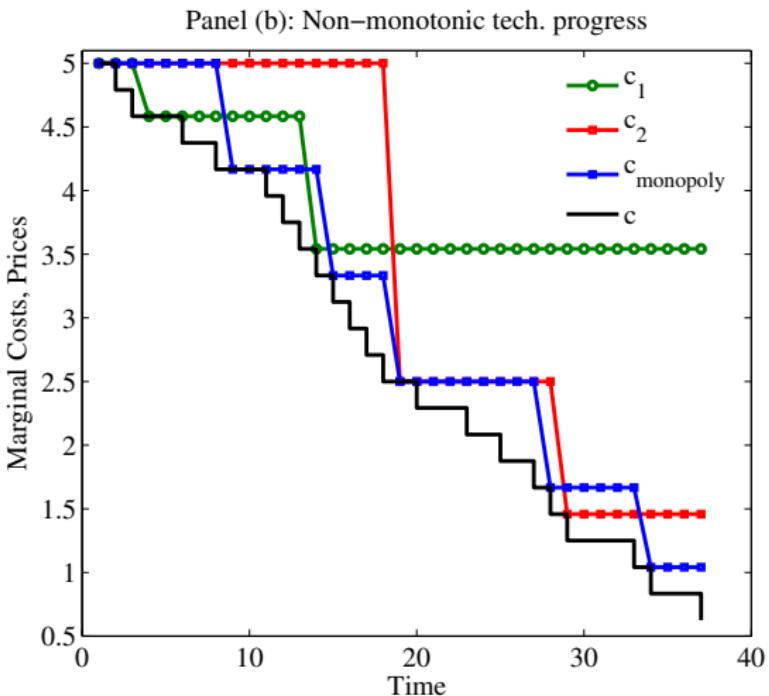
Random alternation → Leapfrogging

Riordan and Salant's result is not robust



Random onestep technology → Leapfrogging

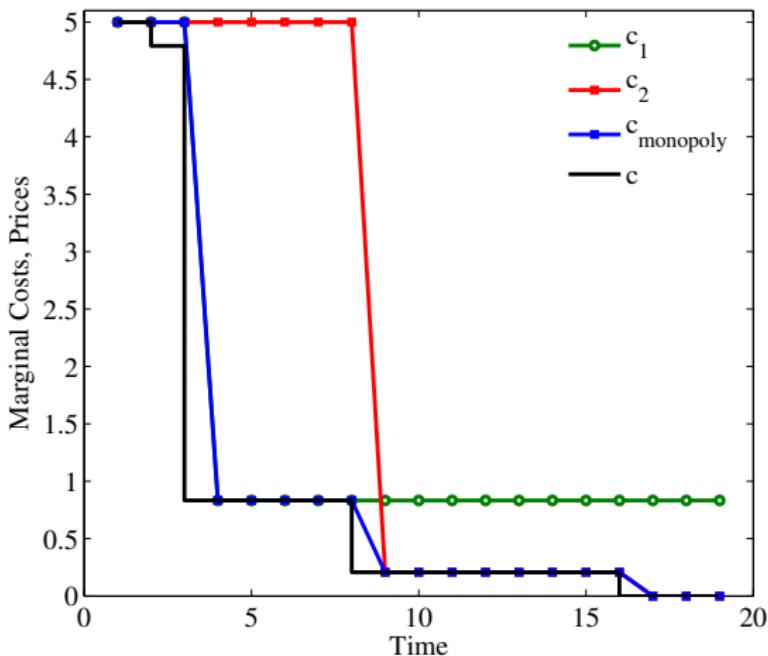
Riordan and Salant's result is not robust



Random multistep technology → Leapfrogging

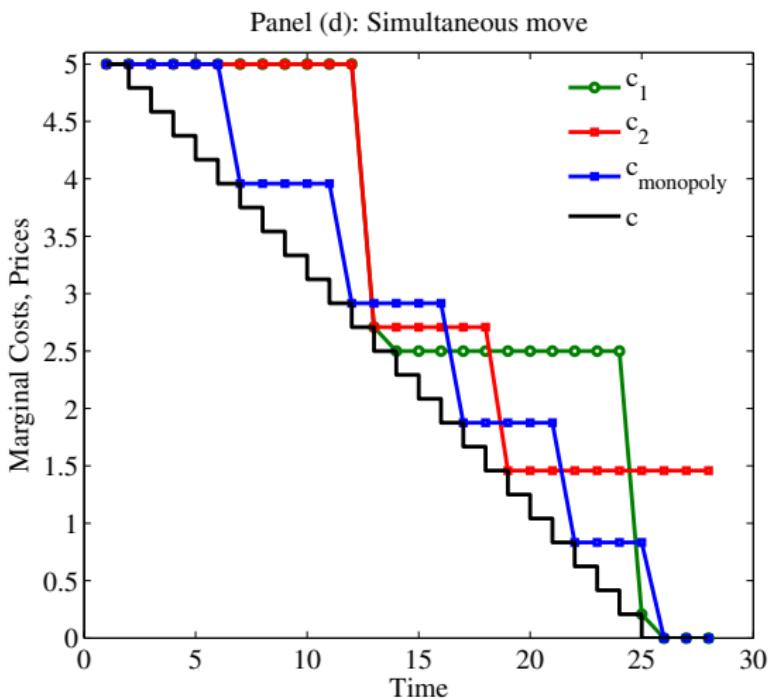
Riordan and Salant's result is not robust

Panel (c): Non-monotonic multistep tech. progress



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

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

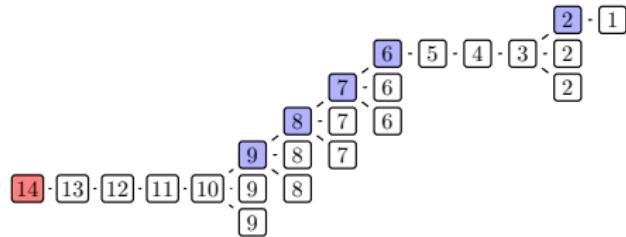
② Inner loop

Maximization of the likelihood function w.r.t. equilibrium selection

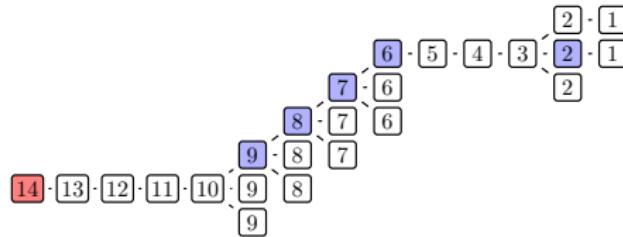
$$\mathcal{L}(Z, \theta) = \max_{(\mathbf{P}^\ell(\theta), \mathbf{V}^\ell(\theta)) \in SOL(\Psi, \theta)} \frac{1}{M} \sum_{i=1}^N \sum_{m=1}^M \sum_{t=1}^T \log P_i^\ell(\bar{a}_i{}^{mt} | \bar{\mathbf{x}}^{mt}; \theta)$$

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

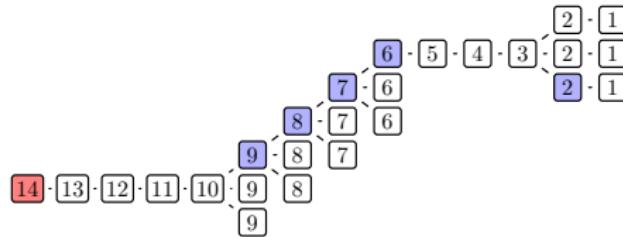
RLS Tree Traversal, step 1



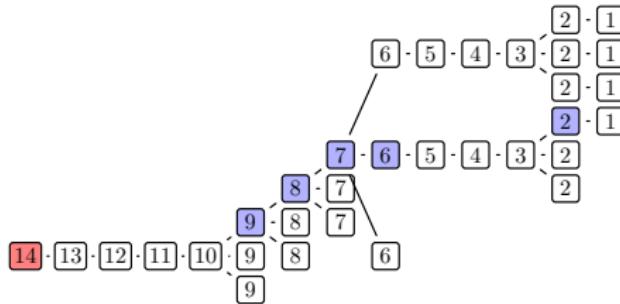
RLS Tree Traversal, step 2



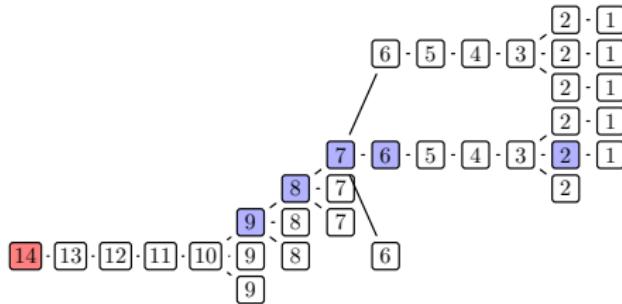
RLS Tree Traversal, step 3



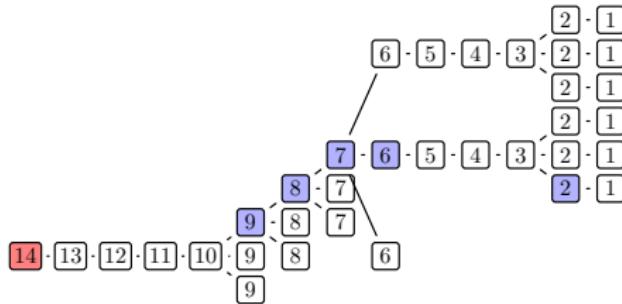
RLS Tree Traversal, step 4



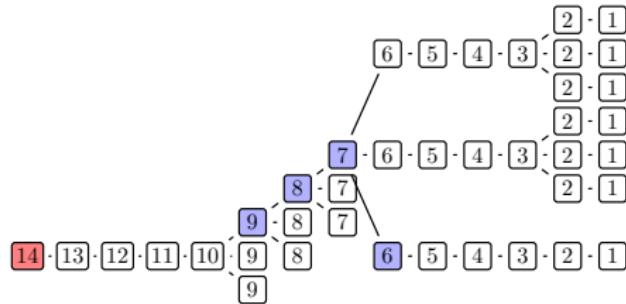
RLS Tree Traversal, step 5



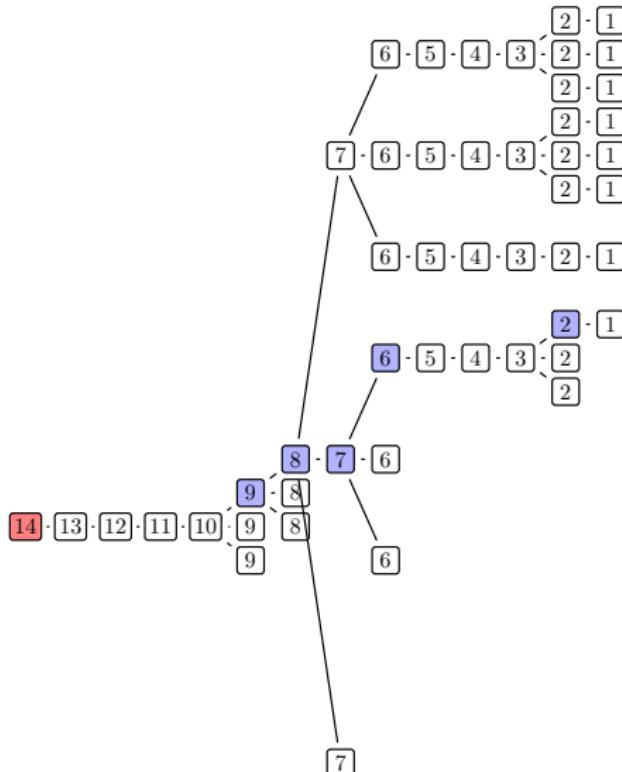
RLS Tree Traversal, step 6



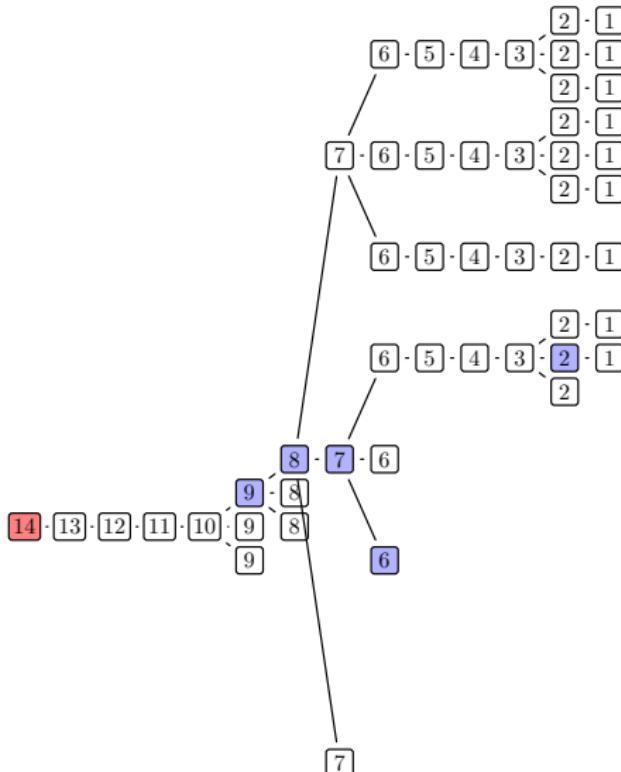
RLS Tree Traversal, step 7



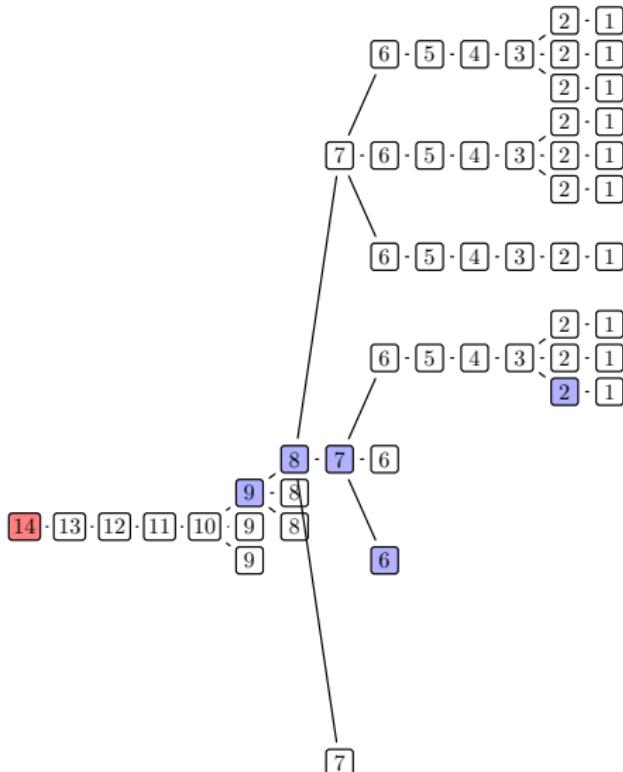
RLS Tree Traversal, step 8



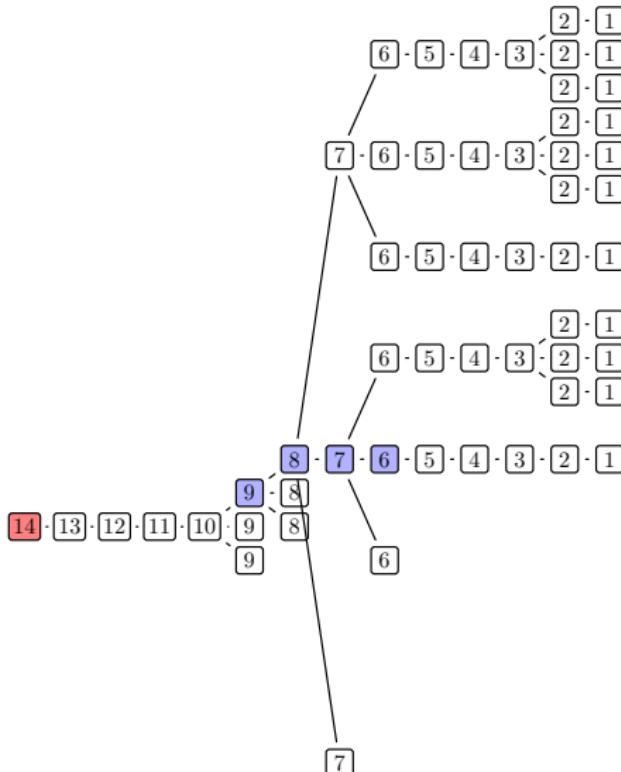
RLS Tree Traversal, step 9



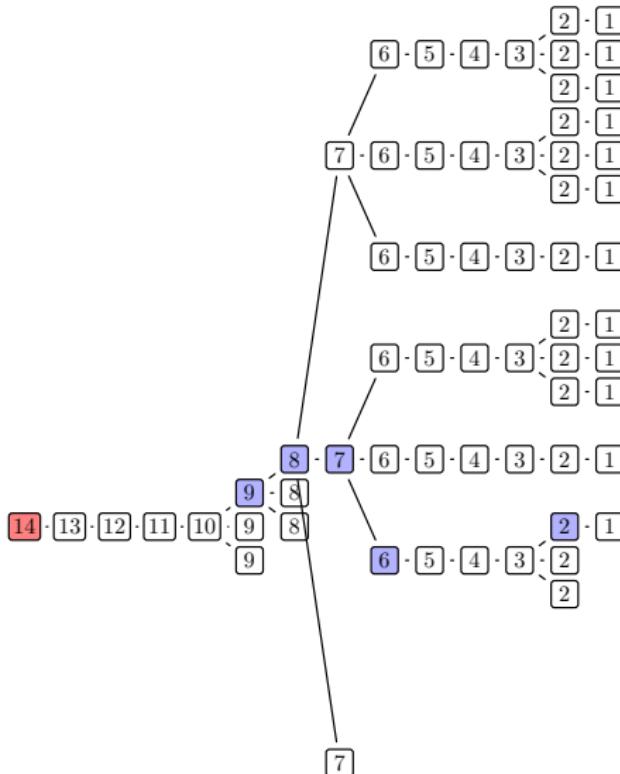
RLS Tree Traversal, step 10



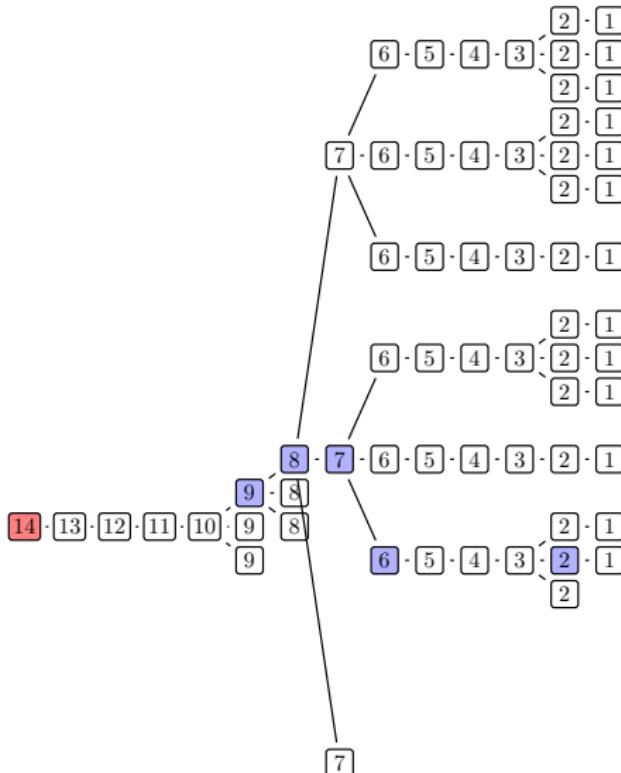
RLS Tree Traversal, step 11



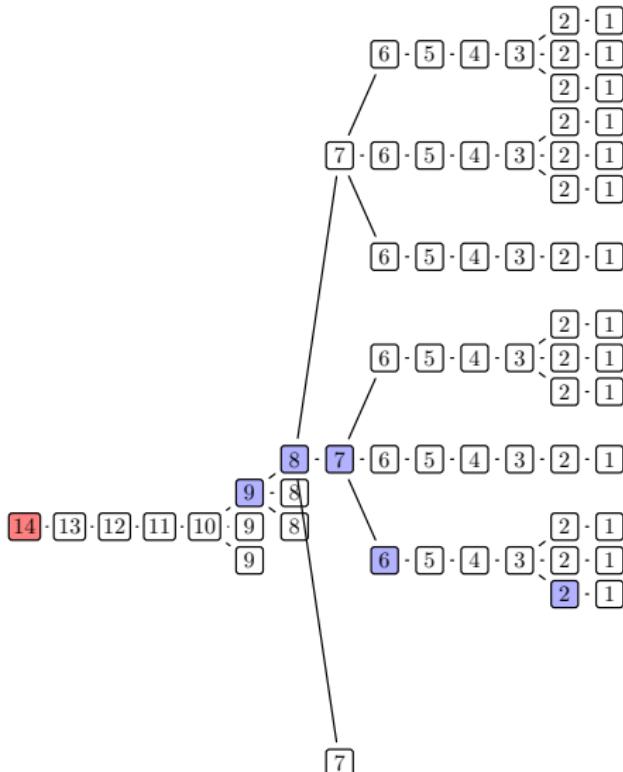
RLS Tree Traversal, step 12



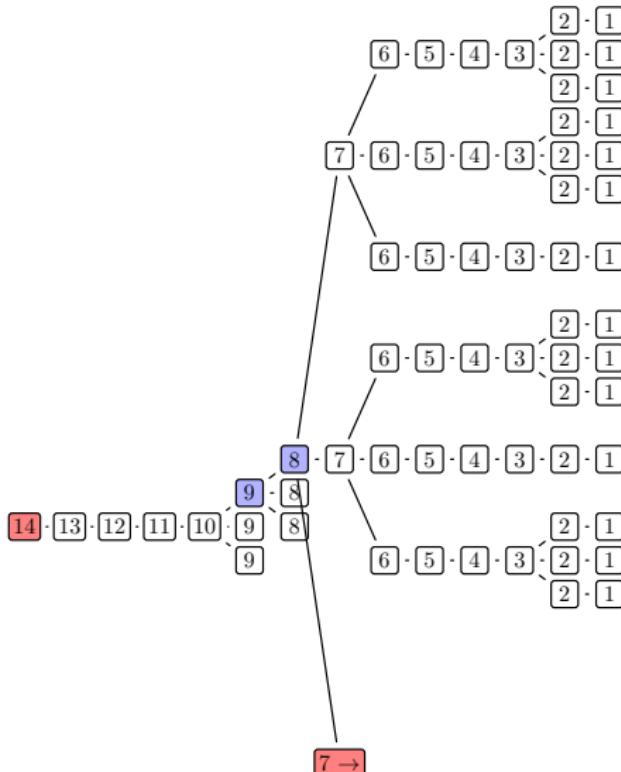
RLS Tree Traversal, step 13



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

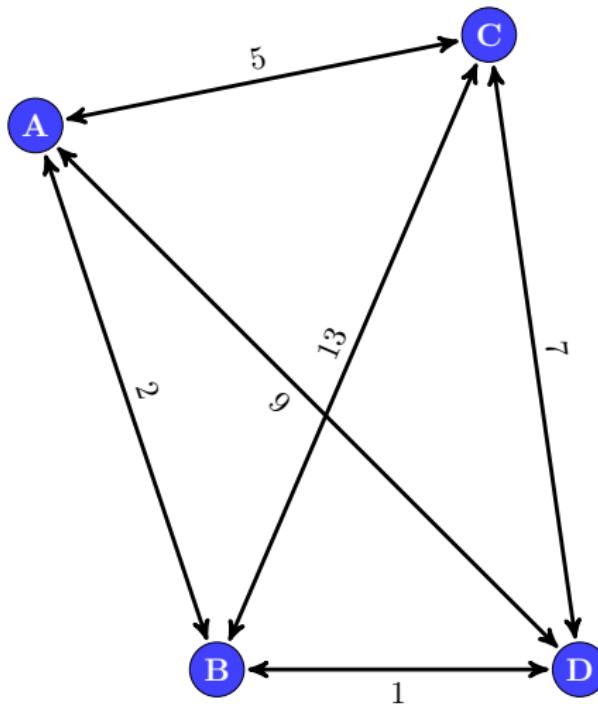
Definition (Classic TSP)

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?

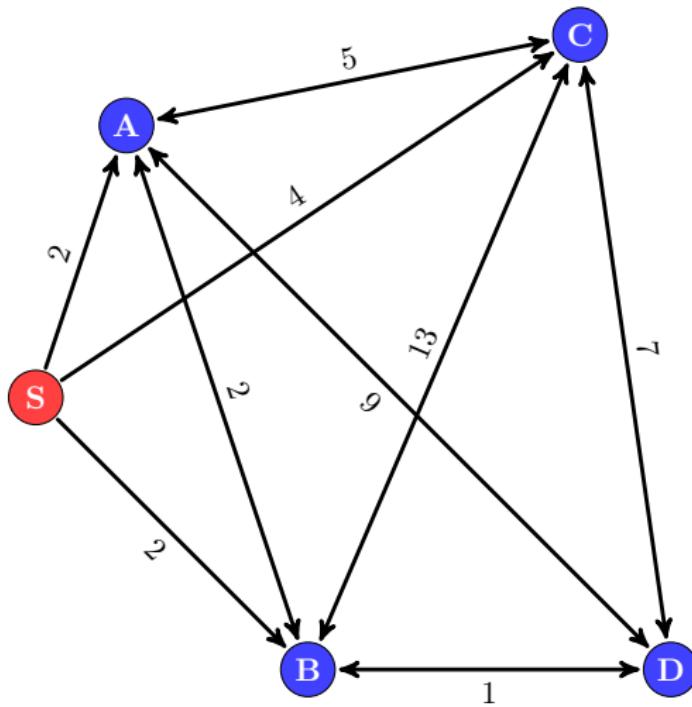


Papadimitriou, 1977 *Theoretical Computer Science*

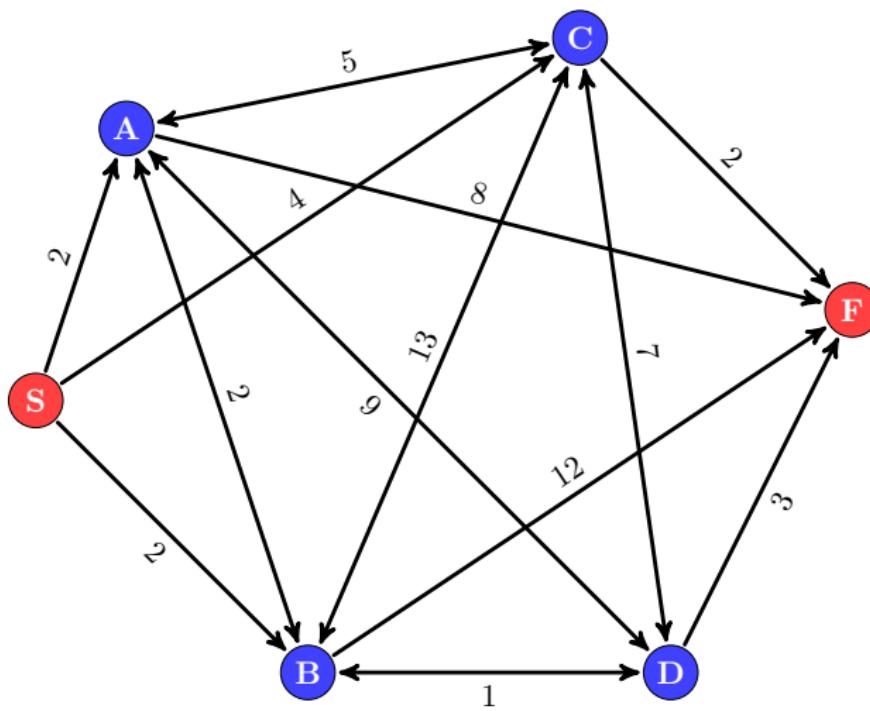
TSP: four cities to visit



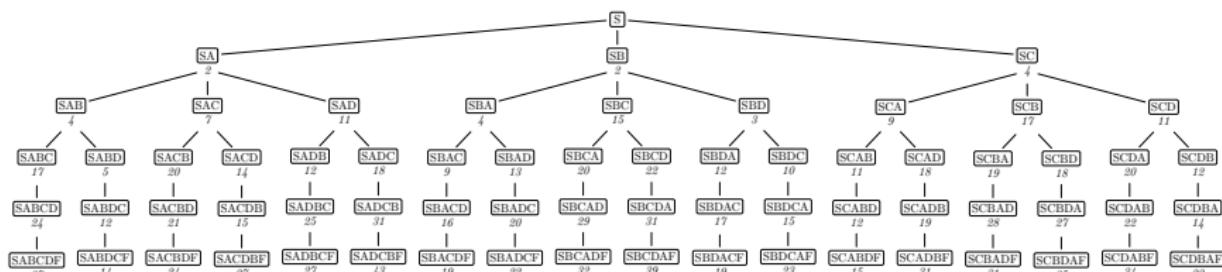
TSP: can start at three cities



TSP: what is the shortest path?

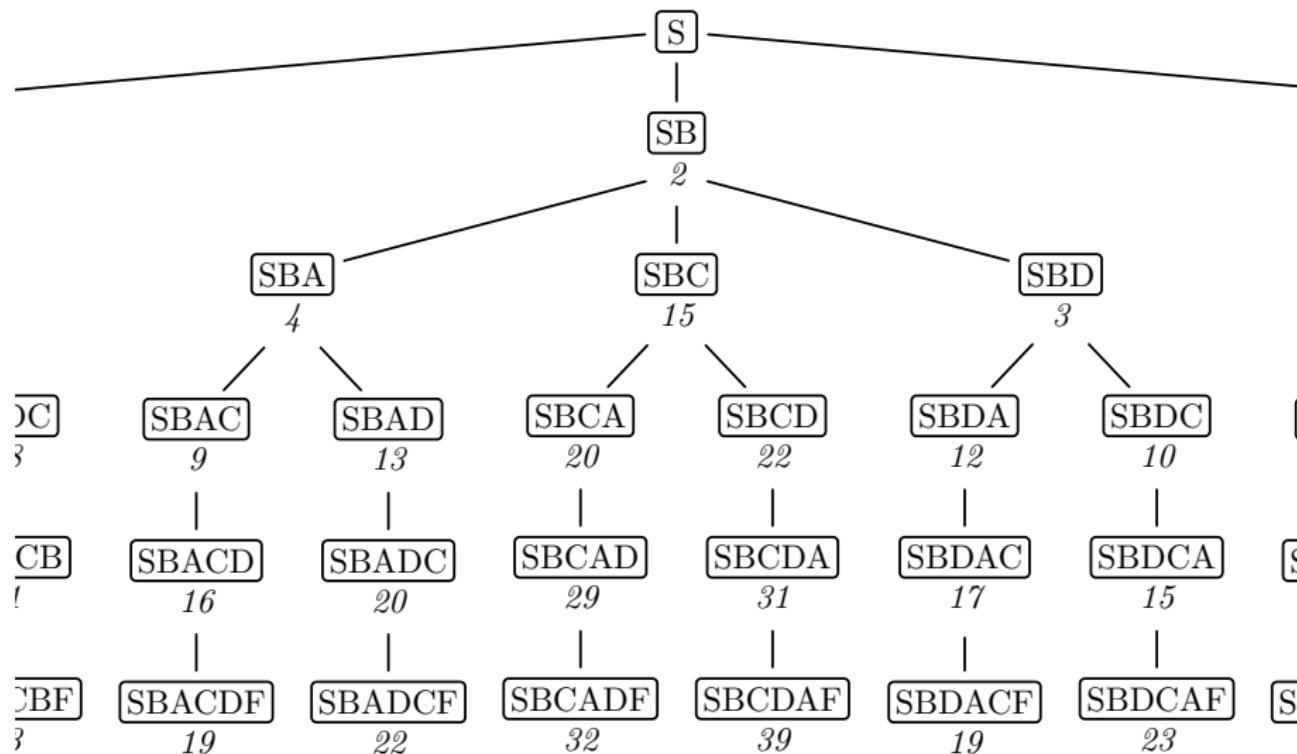


TSP: full tree of choices

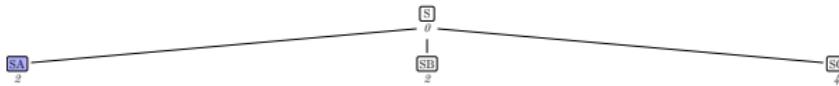


- 67 nodes
- 6 layers
- 18 nodes on the bottom layer \leftrightarrow 18 paths to choose from

TSP: full tree of choices



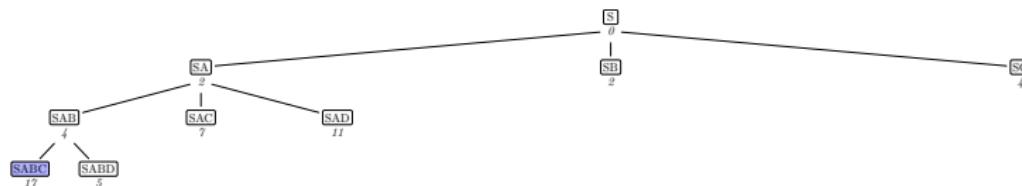
BnB step 1



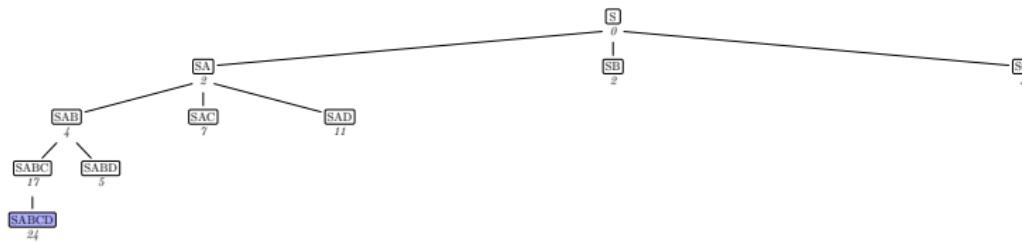
BnB step 2



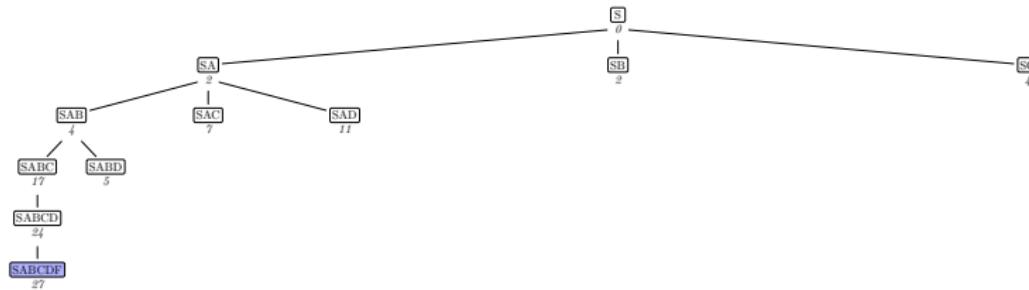
BnB step 3



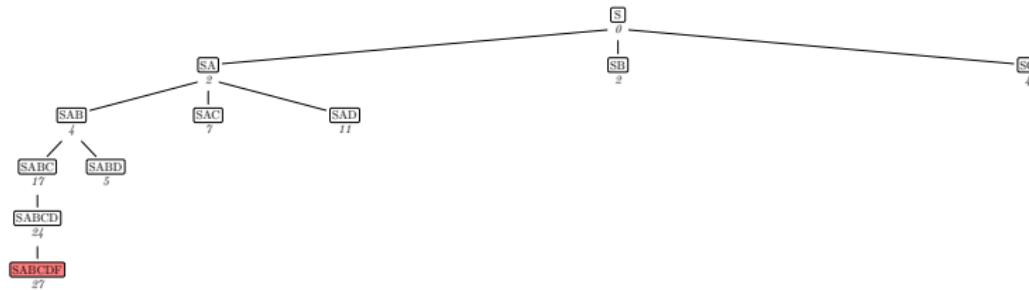
BnB step 4



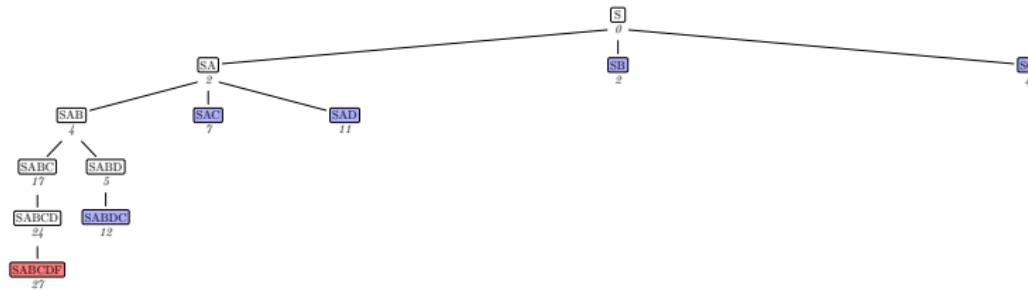
BnB step 5



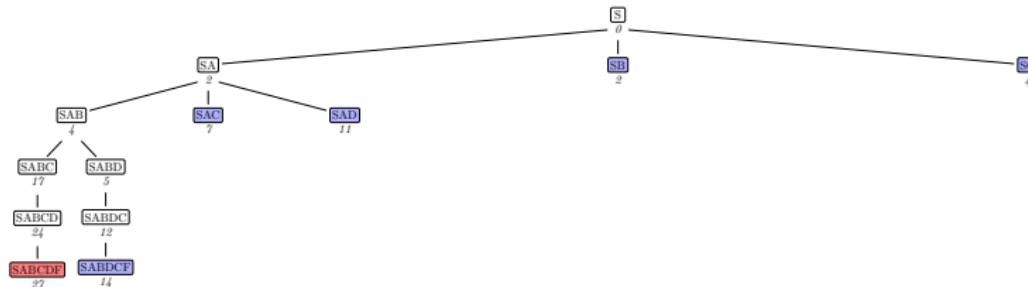
BnB step 6



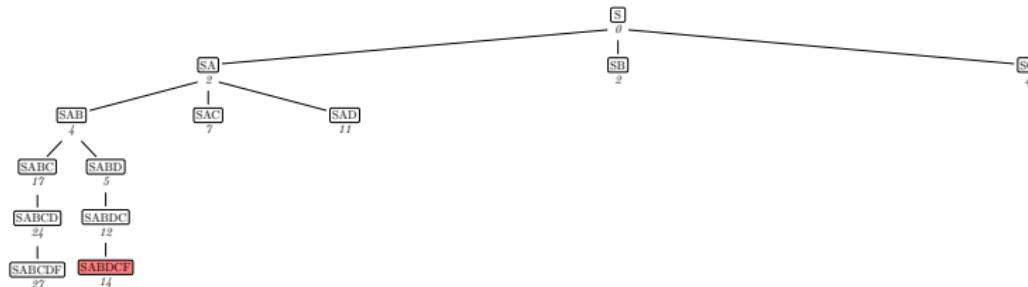
BnB step 7



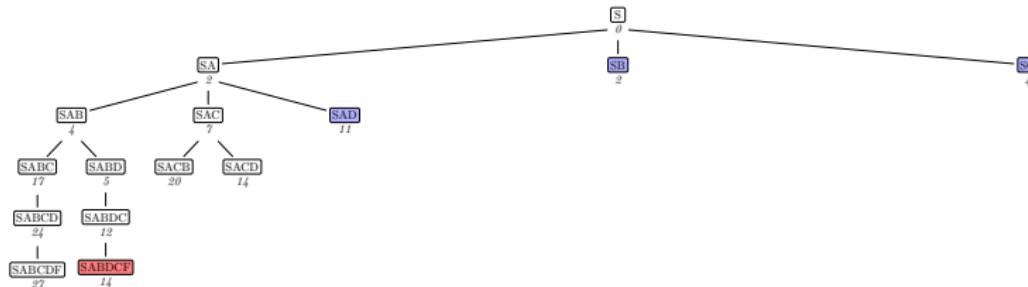
BnB step 8



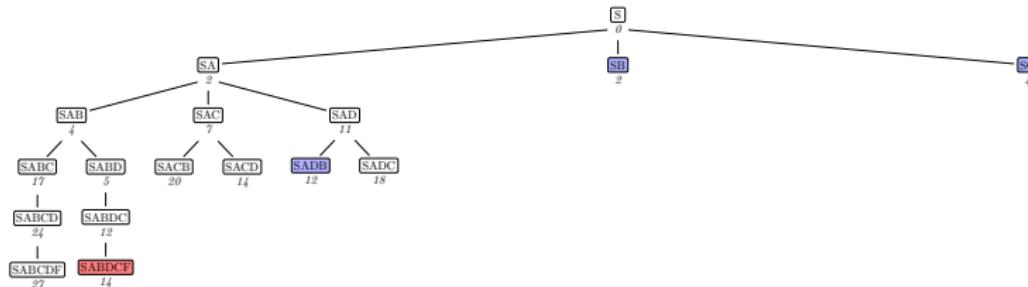
BnB step 9



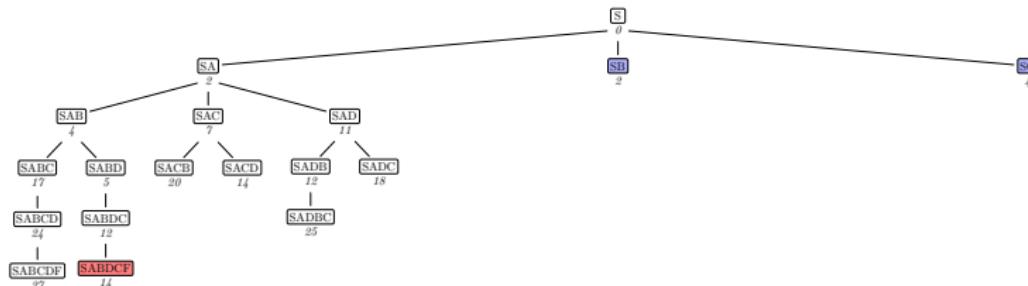
BnB step 10



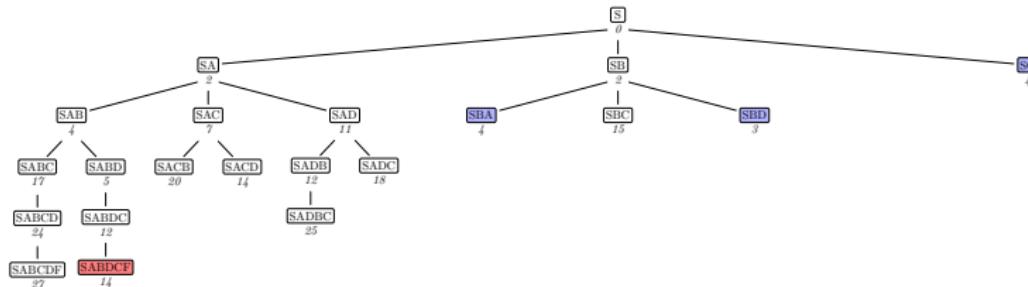
BnB step 11



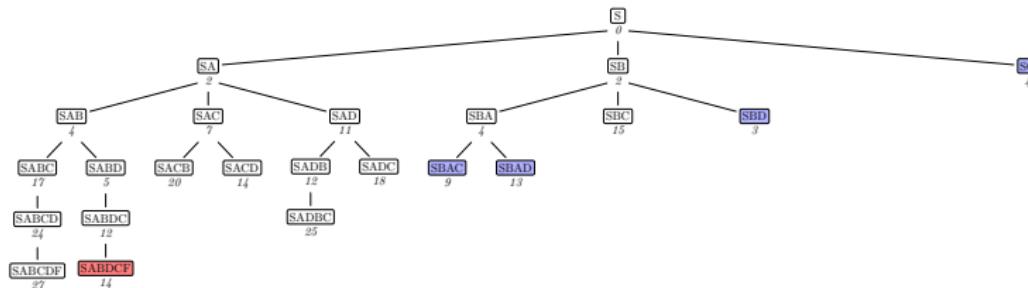
BnB step 12



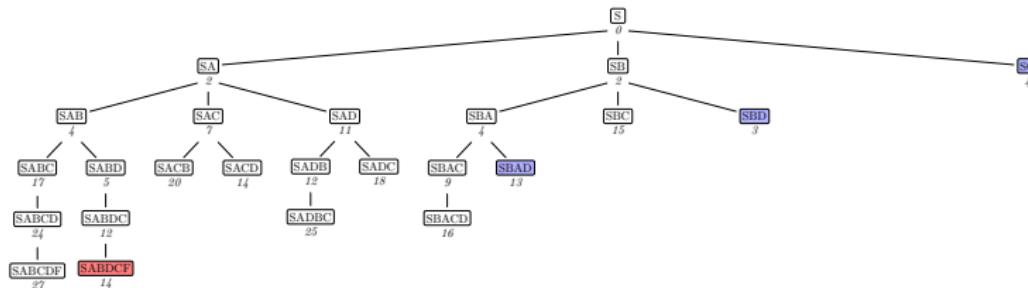
BnB step 13



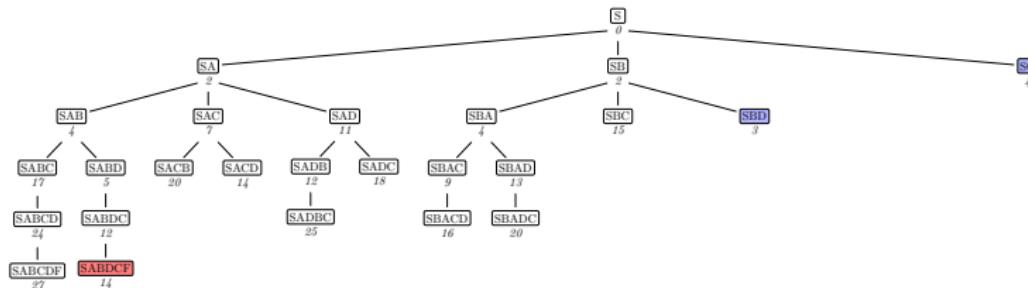
BnB step 14



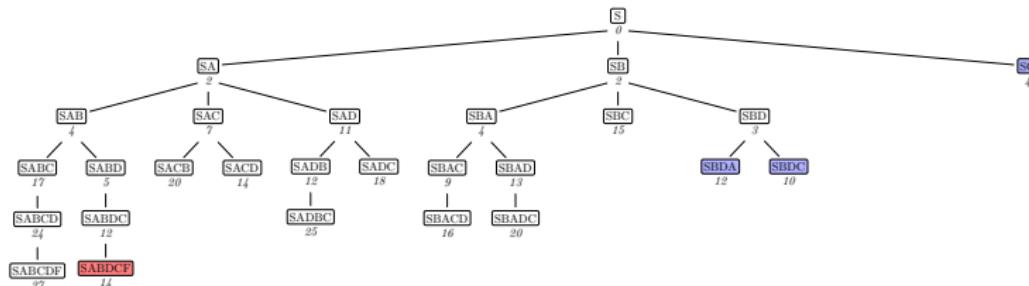
BnB step 15



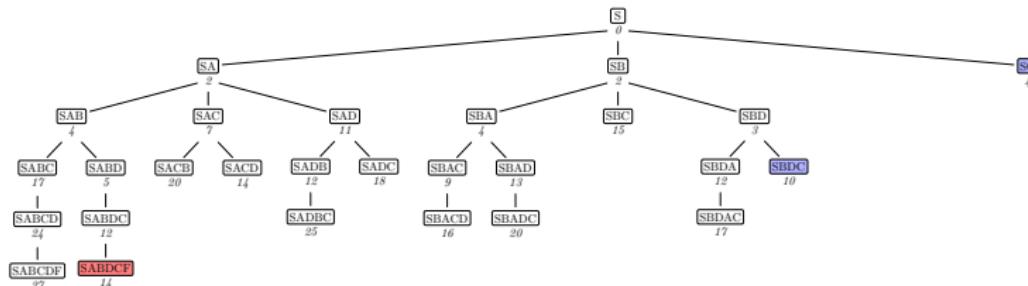
BnB step 16



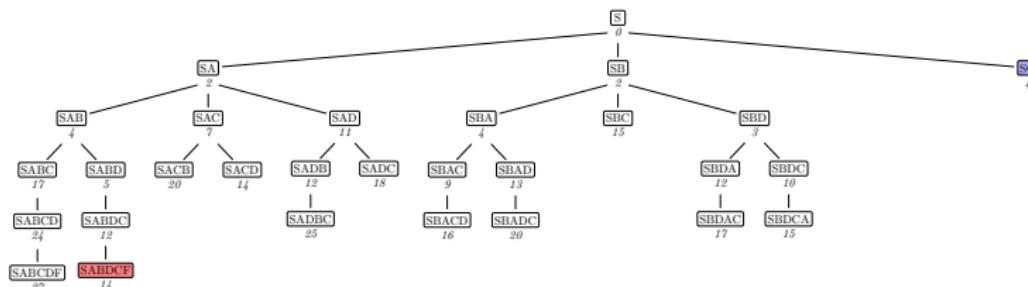
BnB step 17



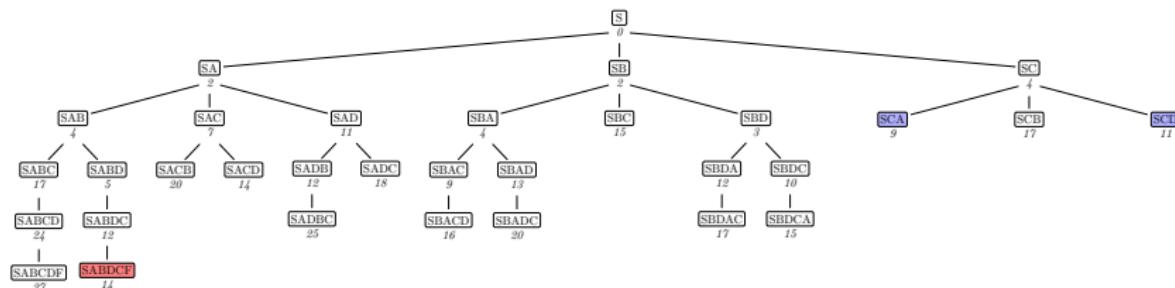
BnB step 18



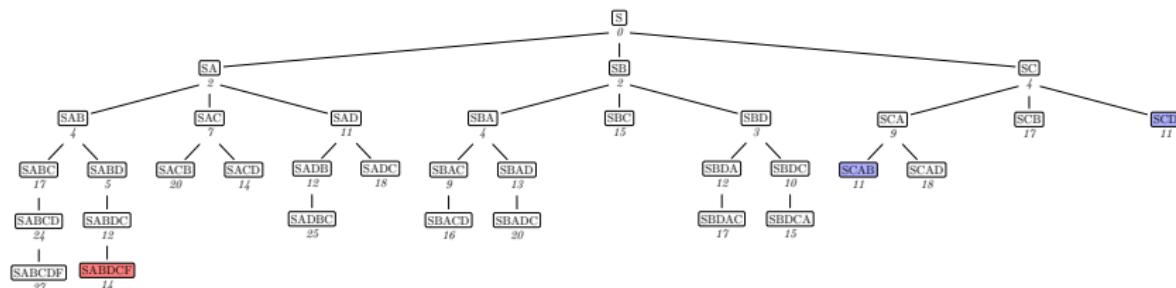
BnB step 19



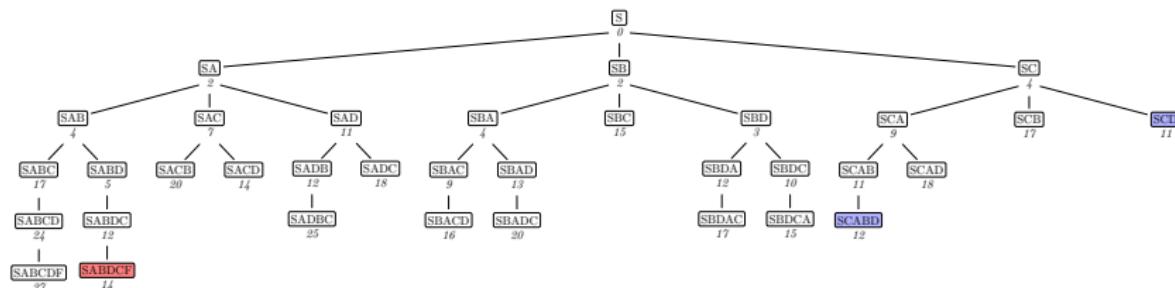
BnB step 20



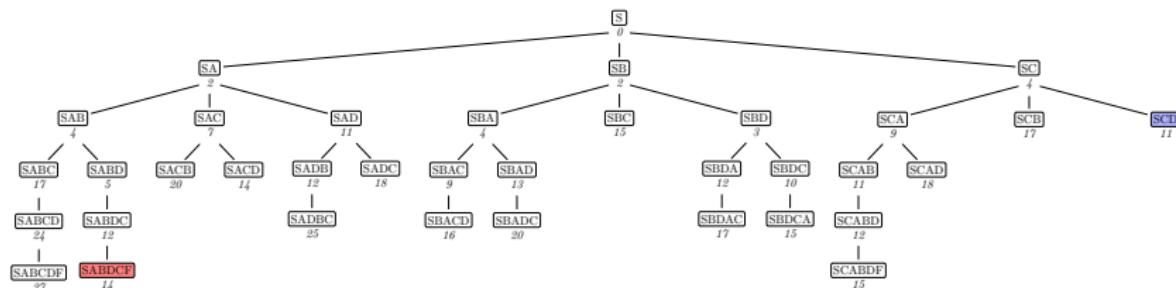
BnB step 21



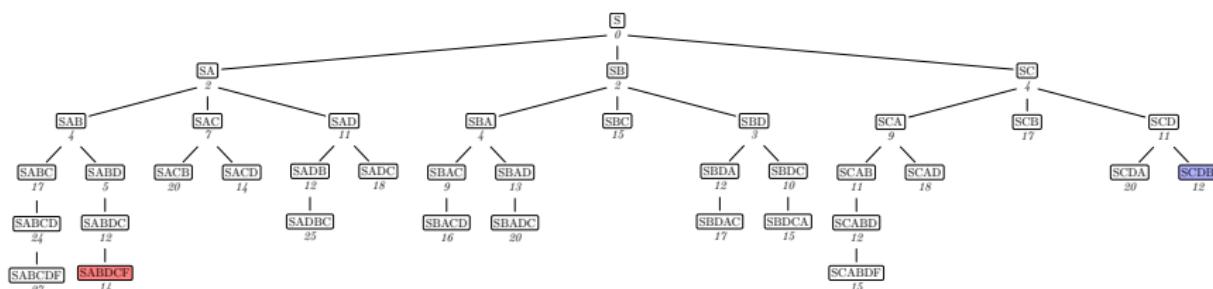
BnB step 22



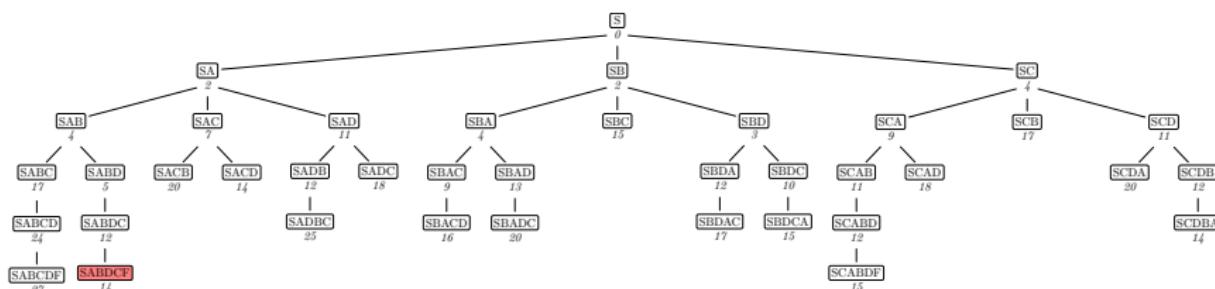
BnB step 23



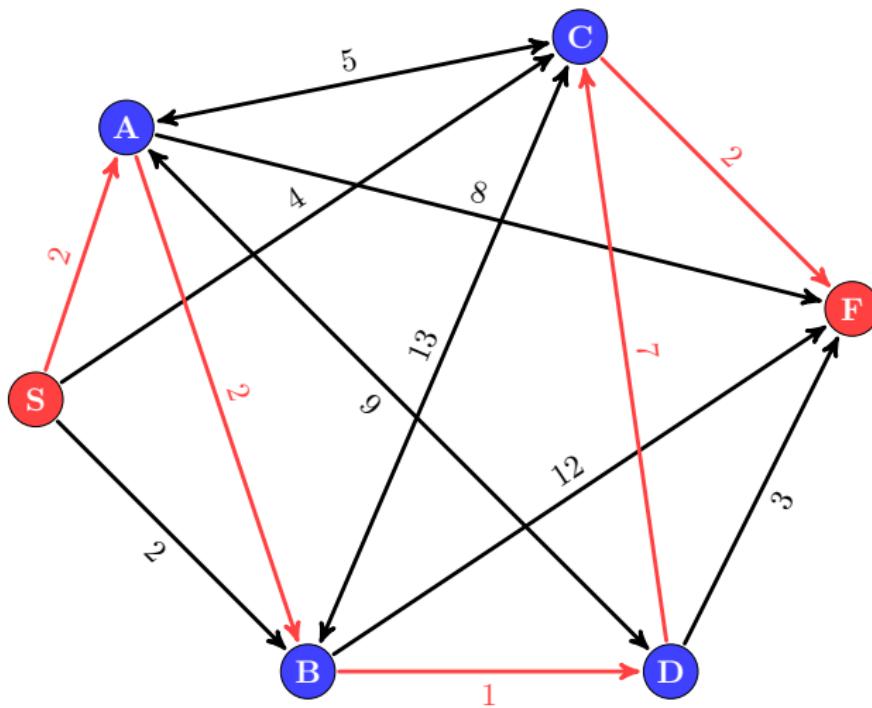
BnB step 24



BnB step 25



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

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

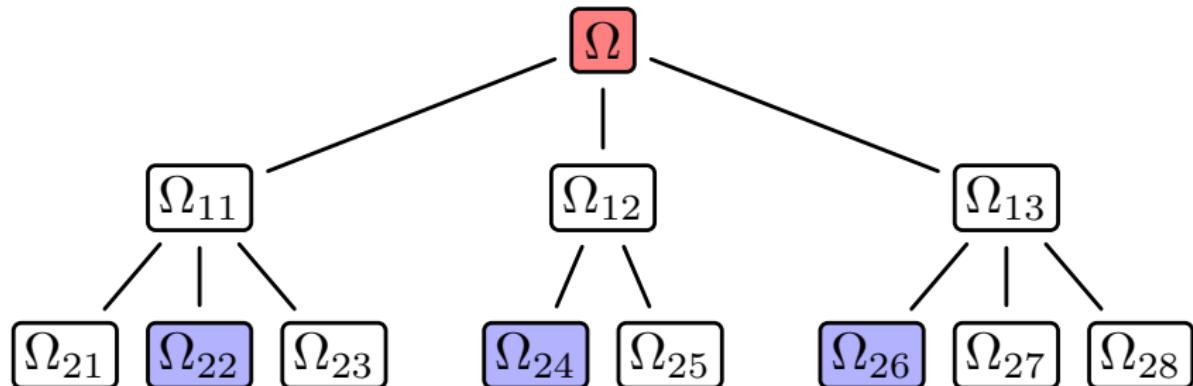
$\{\mathcal{P}_j(\Omega)\}_{j=1,\dots,J}$ forms a sequence of J gradually refined partitions

$$k_1 \leq \dots \leq k_j \leq \dots \leq k_J$$

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

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

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

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**