Recursive Lexicographical Search: Finding all Markov Perfect Equilibria in Directional Dynamic Games

Fedor Iskhakov, University of New South Wales
John Rust, Georgetown University
Bertel Schjerning, University of Copenhagen

ZICE2014
Febrary 5th, 2014

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?
- 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

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 $\mathcal{G}$
- 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
- 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 $\mathcal{G}$.
- 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: 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 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
- 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 $\mathcal{G}$
- 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.
- 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.
- 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

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

Components of Markovian (stochastic) game $\mathcal{G}$

1. n players who take actions at times $t \in \{1, 2, \ldots, T\}$ where T may be ∞,
2. A *finite state space* S and and *action space* A
3. *state-specific constraint sets* $A_i(s)$ representing the set of feasible actions of player i in state s
4. Von Neumann-Morgenstern utility functions $u_i(s_t, a_t) \leftrightarrow$ the payoff to player i in state s_t under actions a_t
5. Markovian state transition probability $p(s'|s, a)$, where $a = (a_1, \ldots, a_n)$ is the vector of actions chosen by players
6. Player-specific discount factors $(\beta_1, \ldots, \beta_n)$, $\beta_i \in [0, 1)$
7. Common knowledge of state s and all the objects above
8. Private information ϵ_i

Behavior strategies

- σ – feasible set of Markovian *behavior* strategies of the players in game $\mathcal{G}$, i.e. n-tuple of mappings $\sigma = (\sigma_1, \ldots, \sigma_n)$ where $\sigma_i : S \to \mathcal{P}(A)$
- $\mathcal{P}(A)$ – set of all probability distributions on the set A.
- Feasibility requires that $\text{support}(\sigma_i(s)) \subseteq A_i(s)$ for each $s \in S$
- A pure strategy is a special case where $\sigma_i(s)$ places a unit mass on a single action $a \in A_i(s)$
- $\Sigma(\mathcal{G})$ – set of all feasible Markovian strategies of the game $\mathcal{G}$

Definition of equilibium

Definition (MPE)

A *Markov perfect equilibrium* of the stochastic game $\mathcal{G}$ is a pair of

- feasible strategy n-tuple σ^*, and
- n-tuple of *value functions* $V(s) = (V_1(s), \ldots, V_n(s))$ where $V_i : S \to R$, such that

1. the system of Bellman equations of the problem is satisfied (with the expectations taken probability distributions induced by opponents' strategies in σ^*), and
2. the strategies constitute mutual best responses of the players, and assign positive probabilities only to the actions in the set of maximizers of the Bellman equation.

Bellman equations

$$V_i(s) = \max_{a \in A_i(s)} \Big[E\left\{ u_i(s,(a,\sigma^*_{-i}(s))) \right\} + \beta_i E\left\{ \sum_{s' \in S} V_i(s') p(s'|s,(a,\sigma^*_{-i}(s))) \right\} \Big],$$

where the expectation is taken over the probability distributions given by the opponents' strategies σ^*_j, $j \neq i$.

- If the maximizer over $a \in A_i(s)$ is unique, then $\sigma^*_i(s)$ is a unit mass on this optimal action.
- If there are multiple $a \in A_i(s)$ that attain the maximum, then $\sigma^*_i(s)$ is a probability distribution whose support is a subset of the set of $a \in A_i(s)$ that attain the maximum.

Bellman equations encode subgame perfection

- In definition of MPE the notion of "subgame perfectness" is reflected by the restriction implicit in the "Principle of optimality" of dynamic programming.
  For each player's strategy σ_i^*, the following holds

Definition (Principle of Optimality/One Shot Deviation Principle)

"whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the fir st decision" (Bellman, 1957).

- Thus the equilibrium is subgame perfect

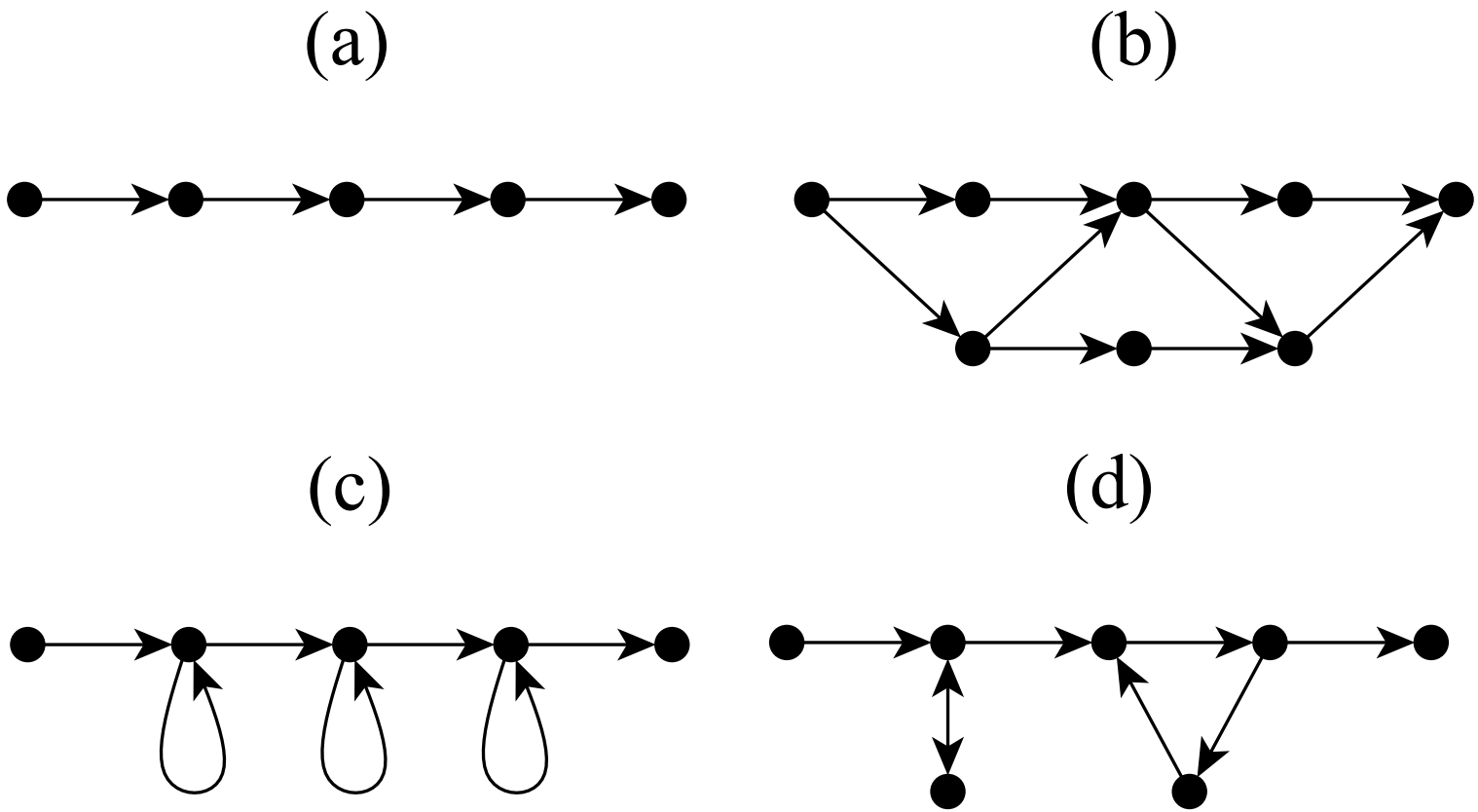
Directional components in the state space

- Suppose S can be written as $S = D \times X$ so $s = (d, x)$ where we refer to d as the *directional component of* s and x as the *non-directional component of* s
- In the leapfrogging example, $s = (c_1, c_2, c, m)$, so we have $d = (c_1, c_2, c)$ and $x = m$.
- If $\mathcal{G}$ is a finite horizon game, then $s = (t, x)$, where t denotes time. Then clearly $d = t$ is the directional component of the state space.

How to formalize the notion of directionality?

Markovian Games as Directed Graphs (DGs)

Games (a) to (c) are *directional.* What about game (d)?



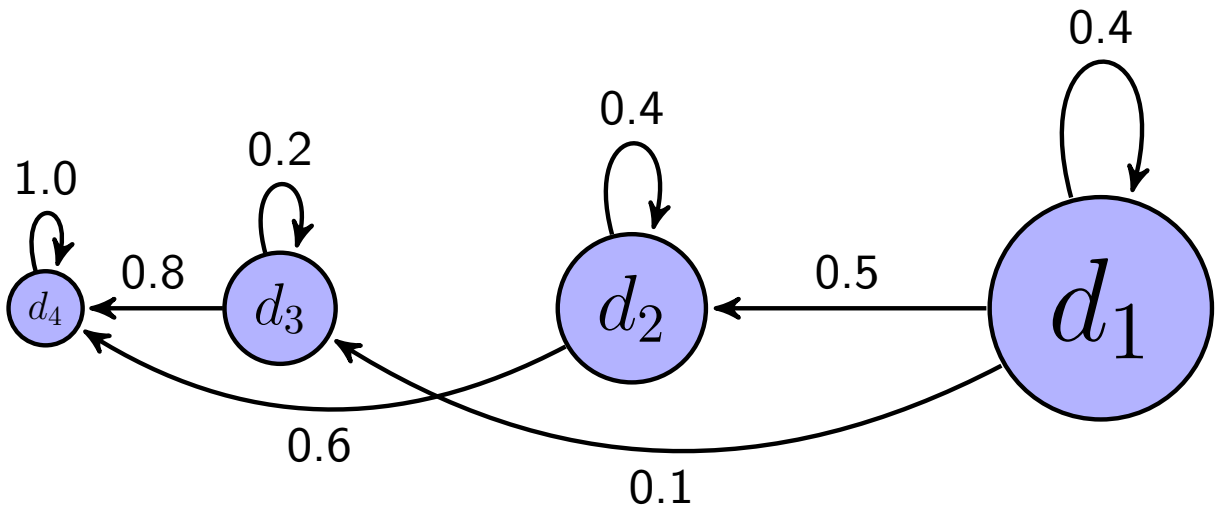
Strategy-specific partial order over D

- Let σ be a feasible Markovian strategy in the game, and let $\rho(d'|(d,x),\sigma(d,x))$ be the *conditional hitting probability* of the state $d' \in D$ starting from the state $s = (d,x)$.
- **Definition** $d' \succ_\sigma d$ *iff*
  $\exists x \in X \quad \rho(d'|(d,x),\sigma(d,x)) > 0$ and
  $\forall x' \in X \quad \rho(d|(d',x'),\rho(d',x')) = 0.$
- **Lemma** $\succ_\sigma$ is a *strict partial order of* D, i.e. it is irreflexive, asymmetric, and transitive.

The No-Loop (Anti-Cycling) Condition

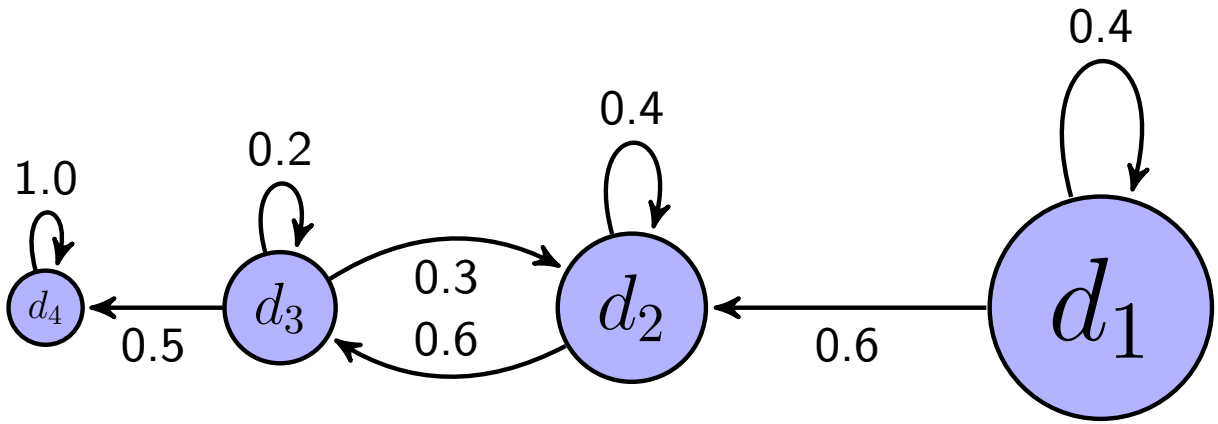
- The partial order $\succ_\sigma$ defines a clear notion of *directionality* in the strategy-induced law of motion of the game $\mathcal{G}$. Because $\succ_\sigma$ is transitive and asymmetric, there can be no loops (cycles) in any subset of *comparable* elements of D
- There are two ways (d', d) can be non-comparable w.r.t $\succ_\sigma$:
  a) there may be no communication between d' and d,
  b) there may be a *loop* (cycle) between d' and d.
- **Definition: No Loop Condition** $d' \not\succ_\sigma d$ and $d' \neq d$ *iff*
  $\forall x \in X \ \rho(d'|(d,x),\sigma(d,x)) = 0$ *and*
  $\forall x' \in X \ \rho(d|(d',x'),\sigma(d',x')) = 0$.

Bargaining over a stochastically shrinking pie



Notice that d_2 and d_3 are not comparable under the induced partial order $\succ_\sigma$. However the no-loop condition is satisified.

Bargaining over a shrinking/growing pie



This game induces the same partial order on D, $\succ_\sigma$, but the no-loop condition fails due to the loop (cycle) between d_2 and d_3.

Consistency in Induced Partial Orders and DDGs

- **Definition** If σ and σ' are two feasible, Markovian strategies of $\mathcal{G}$, we say the induced partial orders of D are *consistent* if

$$d' \succ_\sigma d \Longrightarrow d \not\succ_{\sigma'} d'$$

Definition (DDG)

A *Dynamic Directional Game* (DDG) is a finite state Markovian game $\mathcal{G}$ that satisfies the following two conditions:

1. Every feasible, Markovian strategy σ satisifies the No-Loop Condition
2. Every pair of feasible, Markovian strategies σ and σ' induce consistent partial orderings, $\succ_\sigma$ and $\succ_{\sigma'}$, respectively.

Defining a *Strategy-Independent* Partial Order for a DDG

- **Definition** Let $\succ_\sigma$ and $\succ_{\sigma'}$ be two strategy-induced partial orders of D. We say that $\succ_{\sigma'}$ is a *refinement* of $\succ_\sigma$ *iff* $\forall d, d' \in D$ we have $d' \succ_\sigma d \Longrightarrow d' \succ_{\sigma'} d$
- **Definition** Let $\{\succ_\sigma | \sigma \in \Sigma(\mathcal{G})\}$ be a set of partial orders of D induced by the set of all feasible Markovian strategies in the game $\mathcal{G}$, $\Sigma(\mathcal{G})$. Then let $\succ_\mathcal{G}$ be the *join* (or coarsest common refinement) of the the set of partial orders $\{\succ_\sigma | \sigma \in \Sigma(\mathcal{G})\}$.
- **Theorem** The join partial order $\succ_\mathcal{G}$ exists, is strategy-independent, and equals the transitive closure of the union of the partial orders in the set $\{\succ_\sigma | \sigma \in \Sigma(\mathcal{G})\}$.

DDGs and DAGs

- **Definition** A *directed acyclic graph* (DAG) is a directed graph with no cycles connecting any two vertices of the graph.
- **Lemma** If $\mathcal{G}$ is a finite state DDG, then the directed graph induced by $\succ_{\mathcal{G}}$, $\mathcal{D}(\mathcal{G})$, is a DAG.
- We use $\mathcal{D}(\mathcal{G})$ to partition D into $\mathcal{T}$ elements that are *totally ordered* and form the indices we need to do state recursion to find a MPE of $\mathcal{G}$, $D = \{D_1, D_2, \ldots, D_{\mathcal{T}}\}$.
- The τ index is a *generalization of the notion of a time index used in standard backward induction arguments.* We say that τ indexes the *stage* of the DDG $\mathcal{G}$.

DAG-recursion: identifying the stages of a finite state DDG

- **Definition** The *terminal nodes* of a DAG $\mathcal{D}(\mathcal{G})$ are the set of all vertices $d \in D$ that have no descendants. We let $\mathcal{N}(\mathcal{D}(\mathcal{G}))$ denote the set of terminal nodes of $\mathcal{D}(\mathcal{G})$.
- **Definition** The *non-terminal sub-DAG* of a DAG $\mathcal{D}(\mathcal{G})$ is the DAG $\mathcal{D}_1(\mathcal{G})$ given by $\mathcal{D}_1(\mathcal{G}) = \mathcal{D}(\mathcal{G}) - \mathcal{N}(\mathcal{D}(\mathcal{G}))$. Thus, $\mathcal{D}_1(\mathcal{G}))$ is the sub-DAG of $\mathcal{D}(\mathcal{G})$ the results when you remove its terminal nodes $\mathcal{N}(\mathcal{D}(\mathcal{G}))$.
- **Definition** (DAG-Recursion) Define a sequence of sub-DAGs of $\mathcal{D}(\mathcal{G})$, $\{\mathcal{D}_\tau(\mathcal{G})\}$ recursively by

$$\mathcal{D}_{\tau+1}(\mathcal{G}) = \mathcal{D}_\tau(\mathcal{G}) - \mathcal{N}(\mathcal{D}_\tau(\mathcal{G})).$$

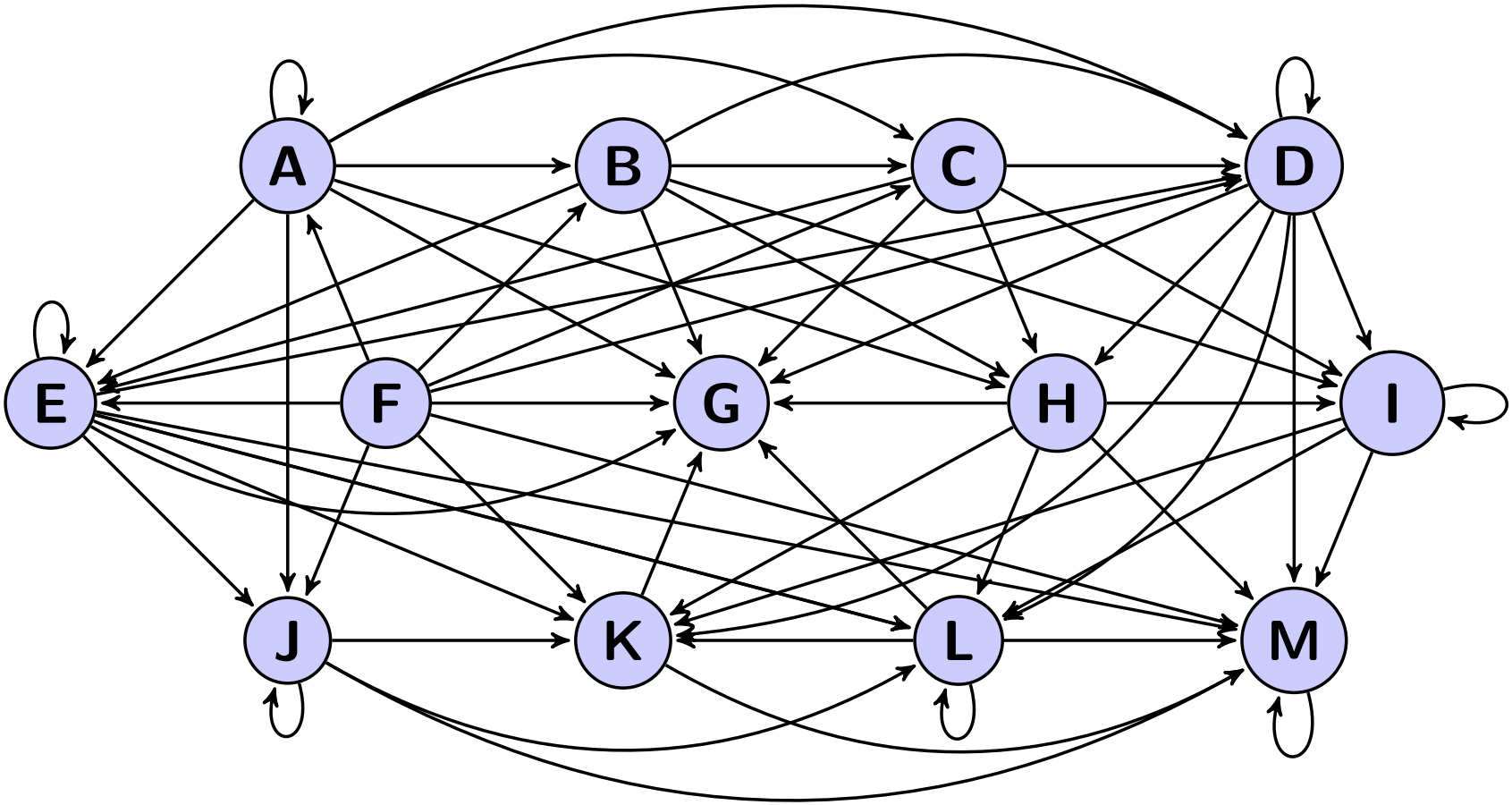
DAG-recursion terminates with the empty set

- **Definition** The *stages* of the finite state DDG $\mathcal{G}$ are given by

$$\{S_1, S_2, \ldots, S_\tau, S_{\tau+1}, \ldots, S_\mathcal{T}\}$$

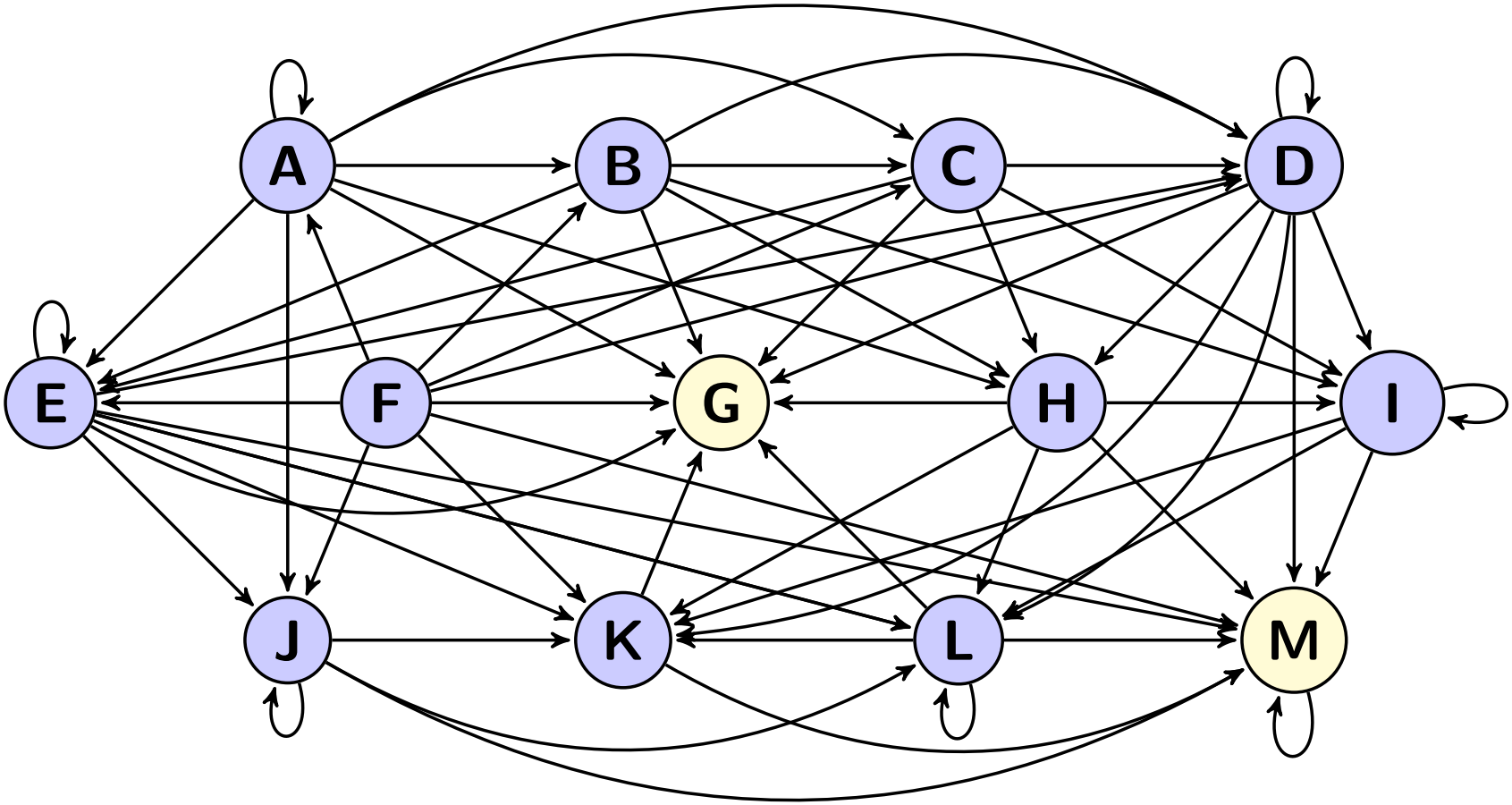
where $S_\tau = (D_\tau \times X)$ where $\{D_1, \ldots, D_\mathcal{T}\}$ is the partition of D defined by the DAG-recursion.

Finding the Stages of a Dynamic Directional Game



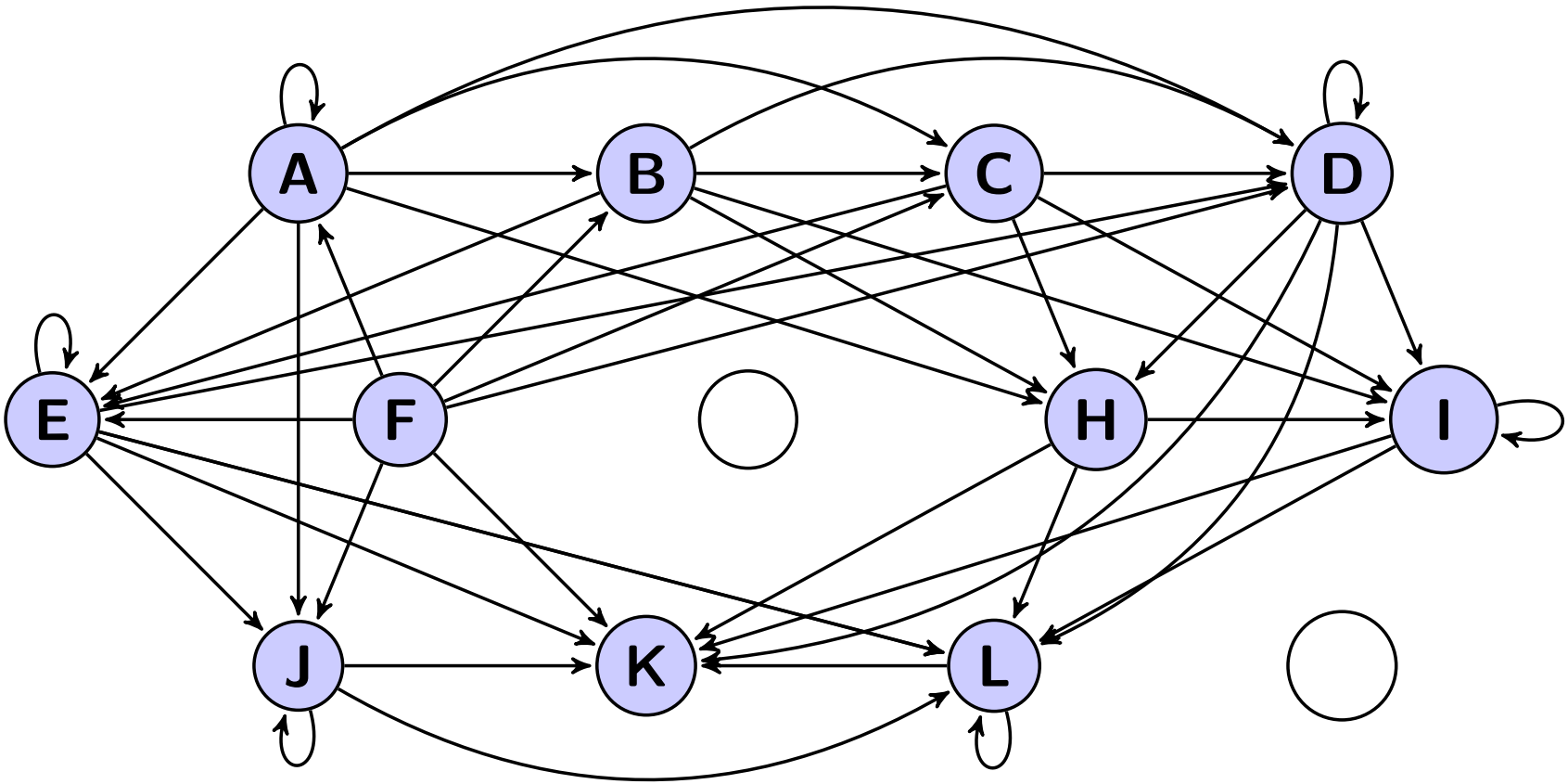
A DDG has a graph that is a DAG. Is this graph a DAG?

Recursion on Directed Acyclic Graphs (DAG-recursion)



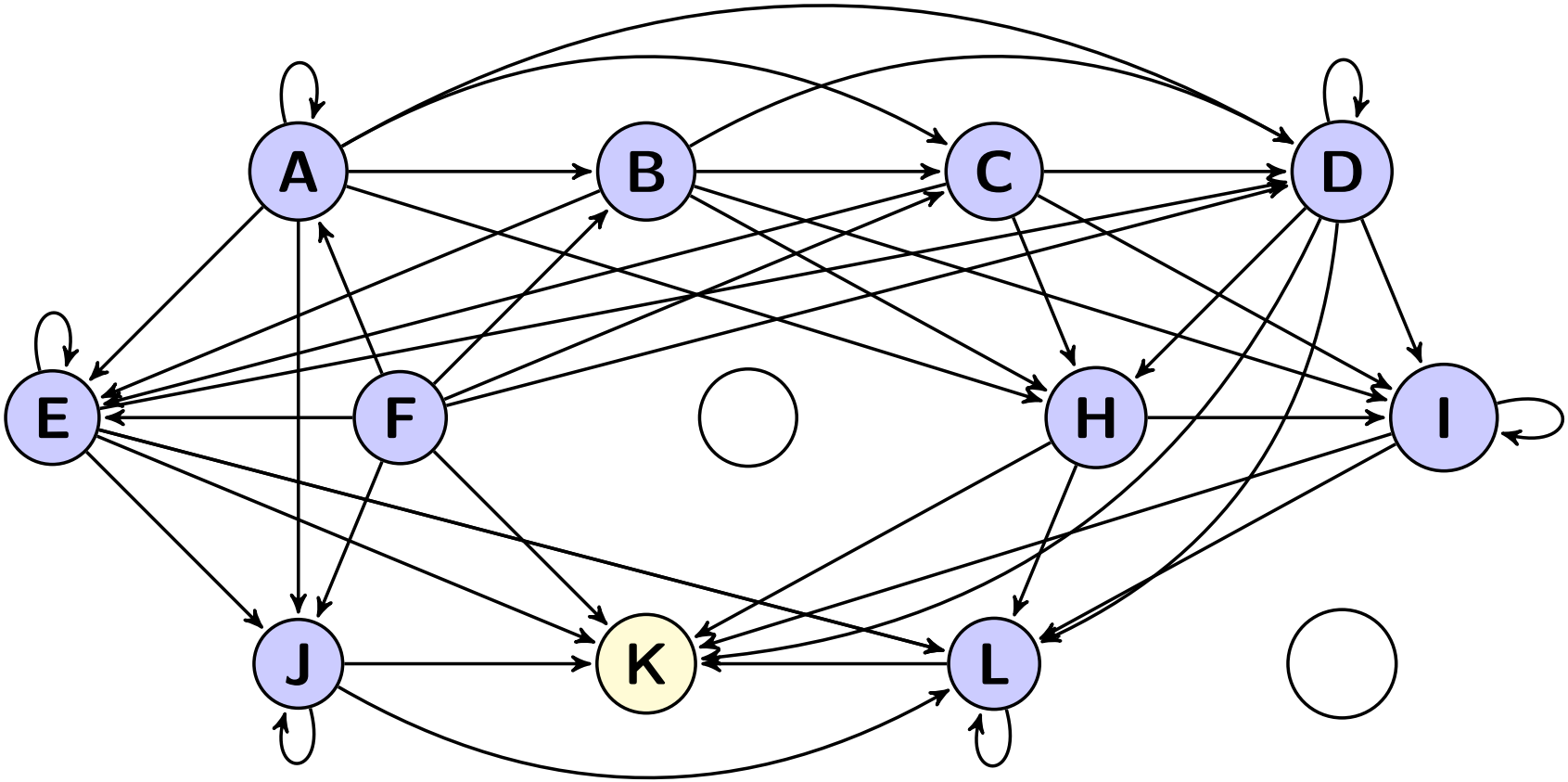
Identify the terminal nodes of the DAG, $S_1 = \{G, M\}$

DAG-recursion: eliminate terminal nodes



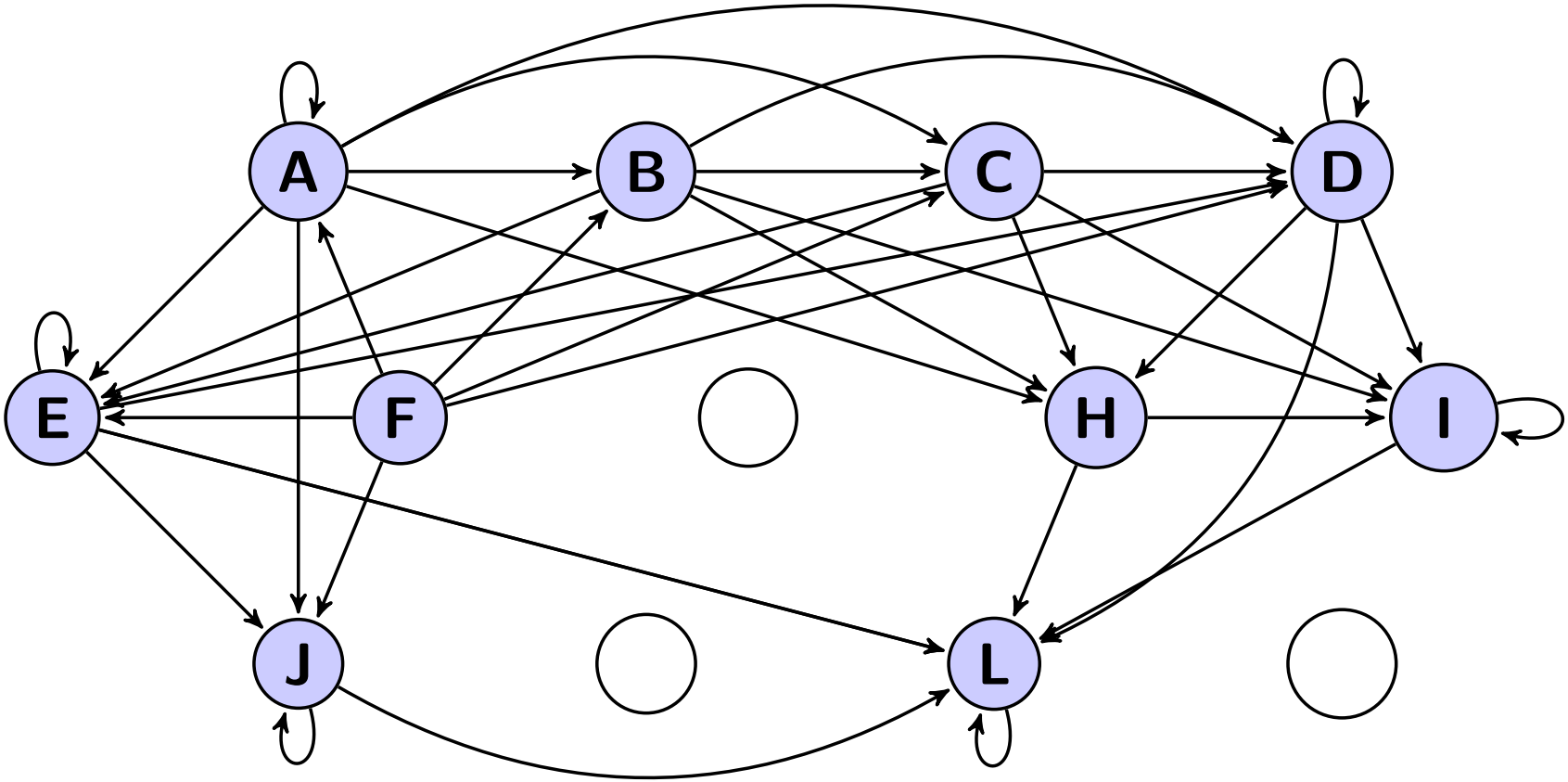
Now eliminate the terminal nodes to obtain the first "sub-DAG"

DAG-recursion: identify the terminal nodes of the sub-DAG



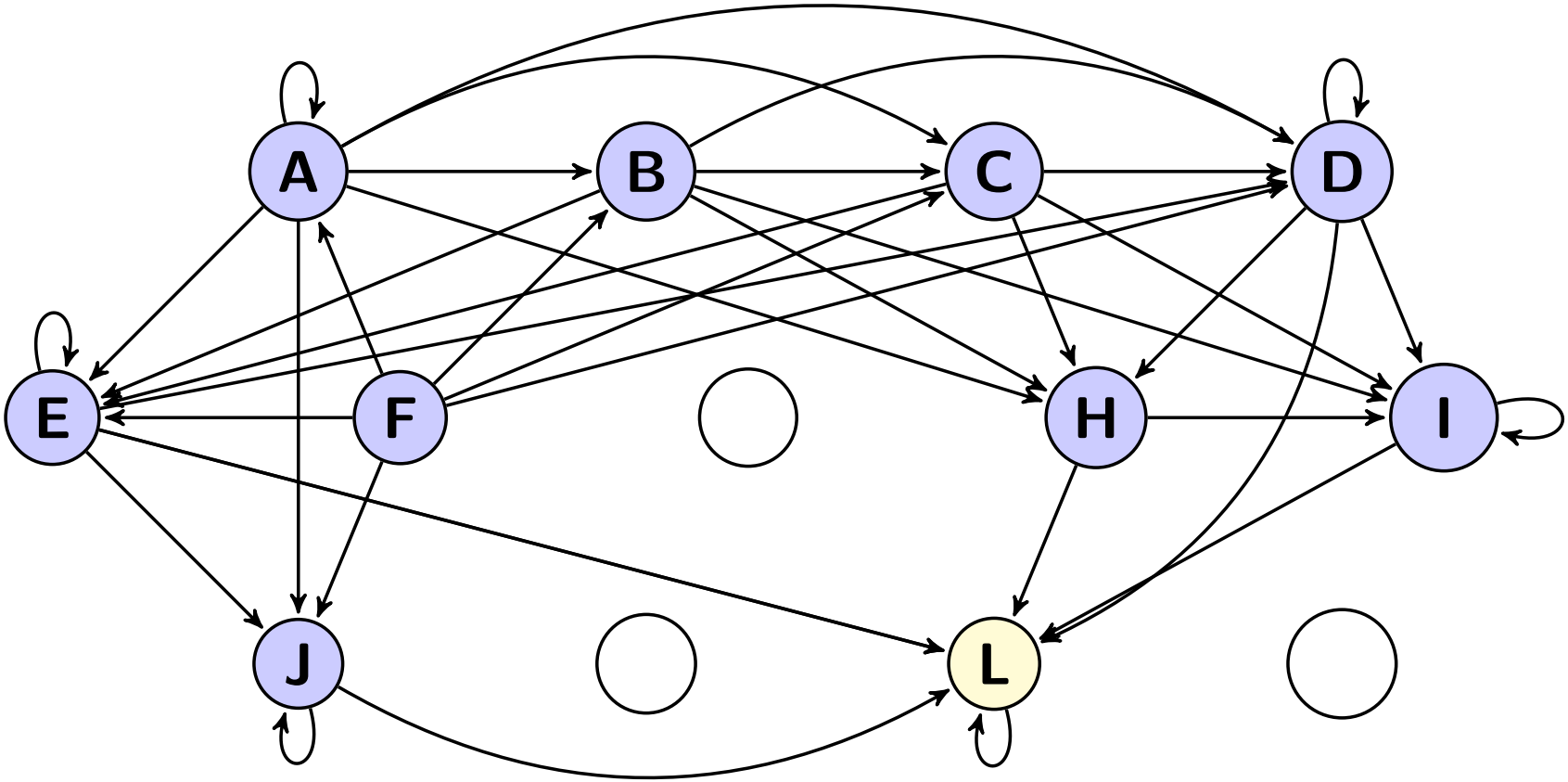
Now identify the terminal nodes of the 1st sub-DAG, $S_2 = \{K\}$

DAG-recursion: eliminate terminal nodes of sub-DAG 1



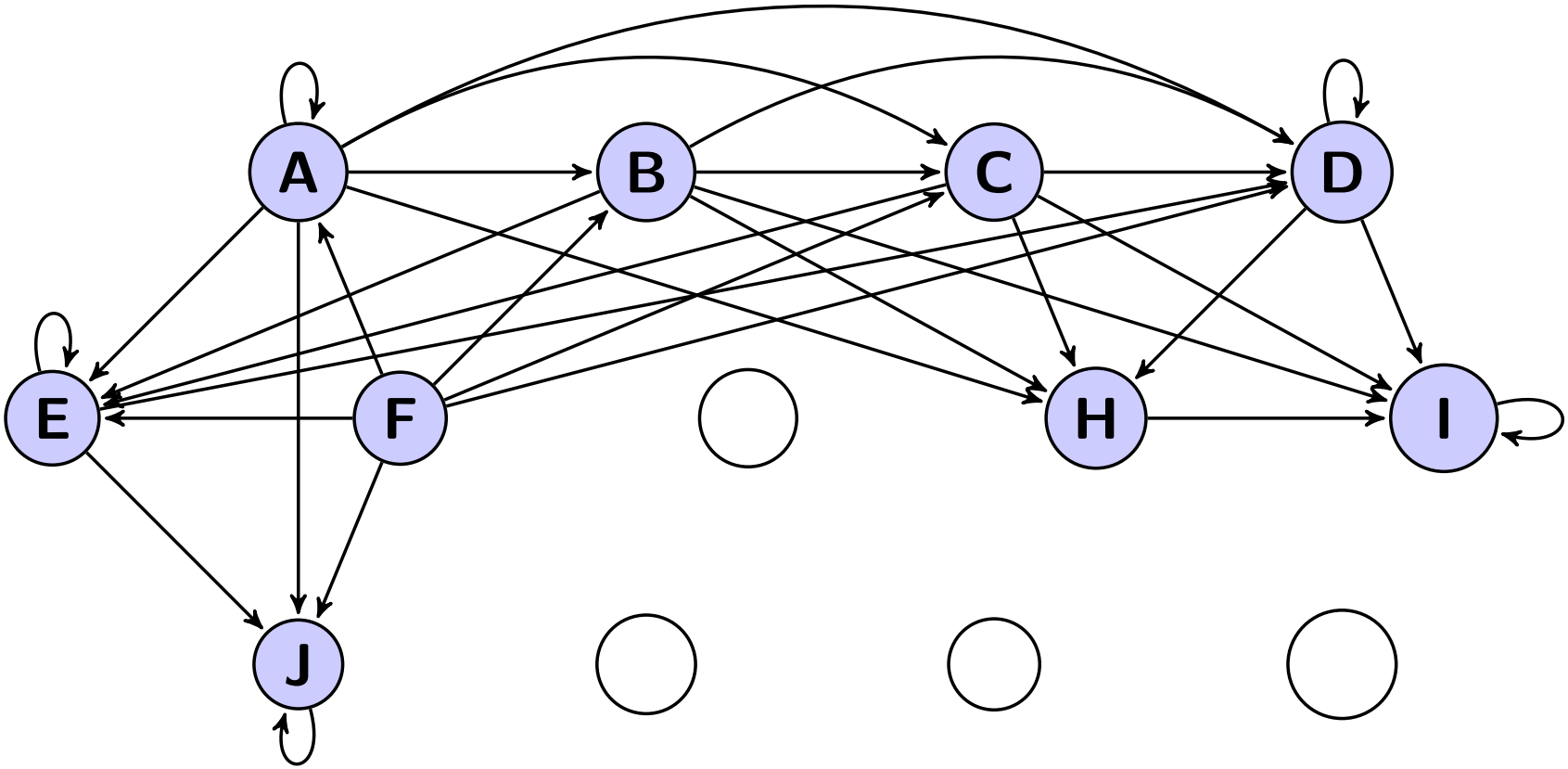
We now have identified stages $S_1 = \{G, M\}$, $S_2 = \{K\}$

DAG-Recursion: identify terminal nodes of sub-DAG 2



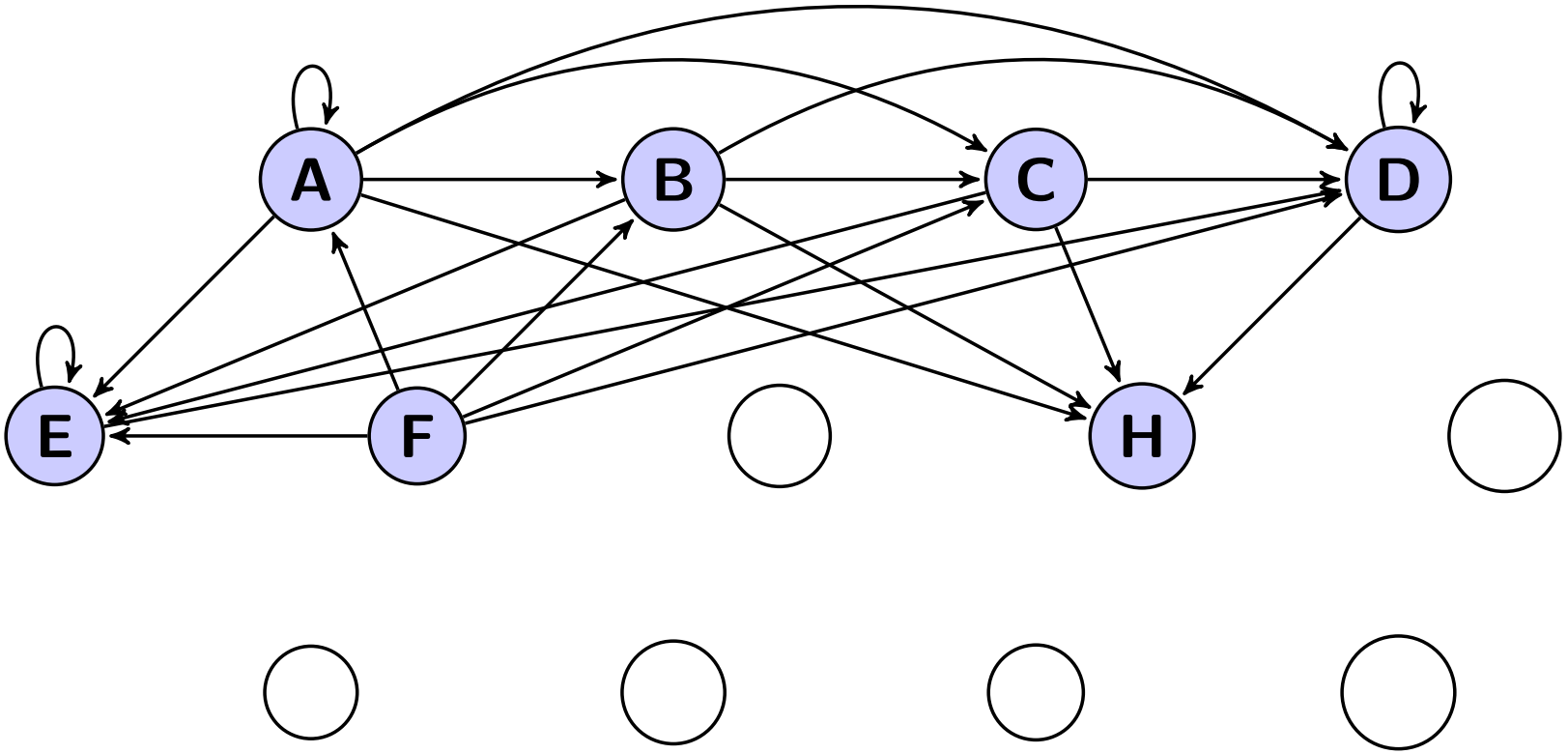
Now identify the terminal nodes of the 2nd sub-DAG, $S_3 = \{L\}$

DAG-recursion: eliminate I,J nodes to get sub-DAG 2



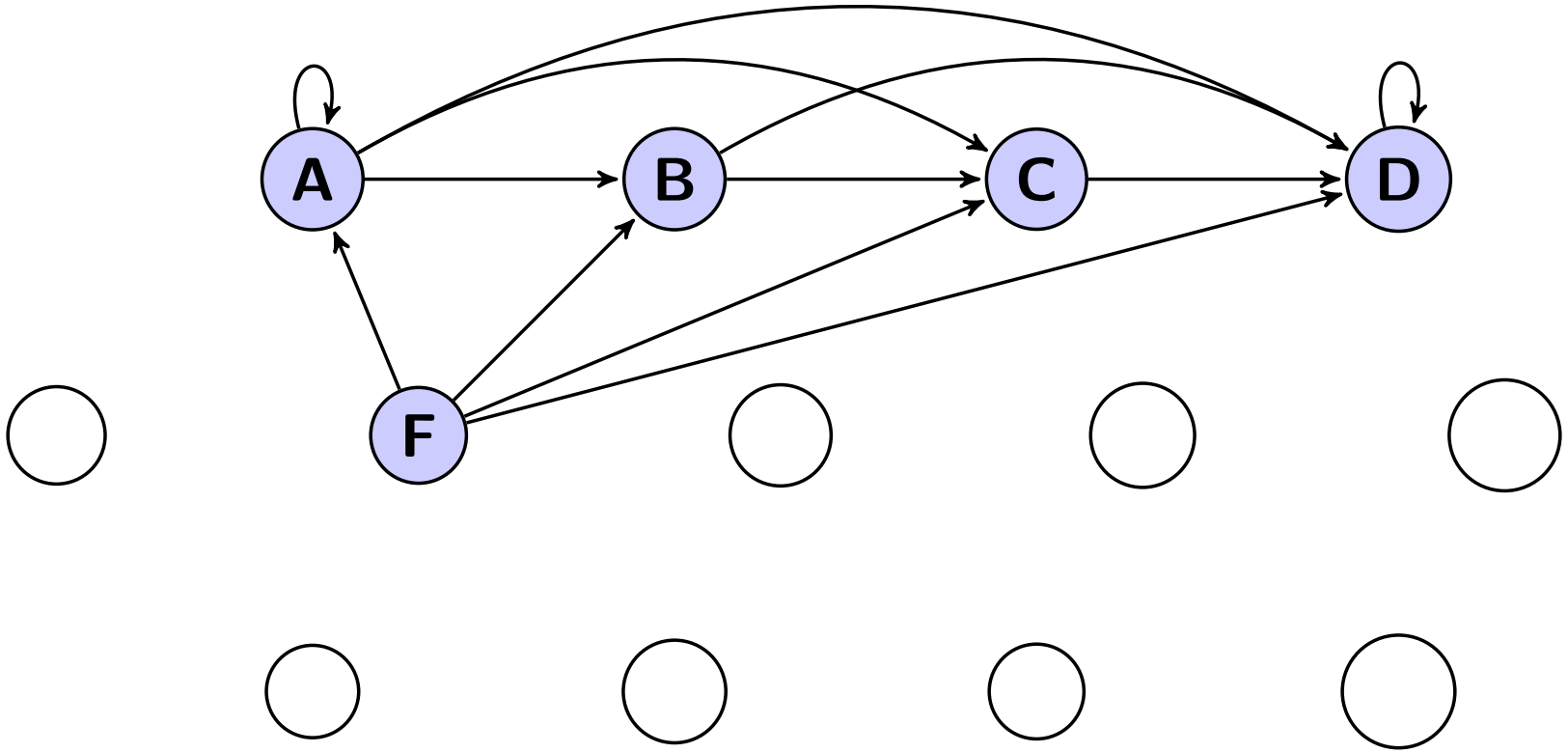
We now have identified stages $S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$

DAG-Recursion: eliminate E,H nodes to get sub-DAG 3



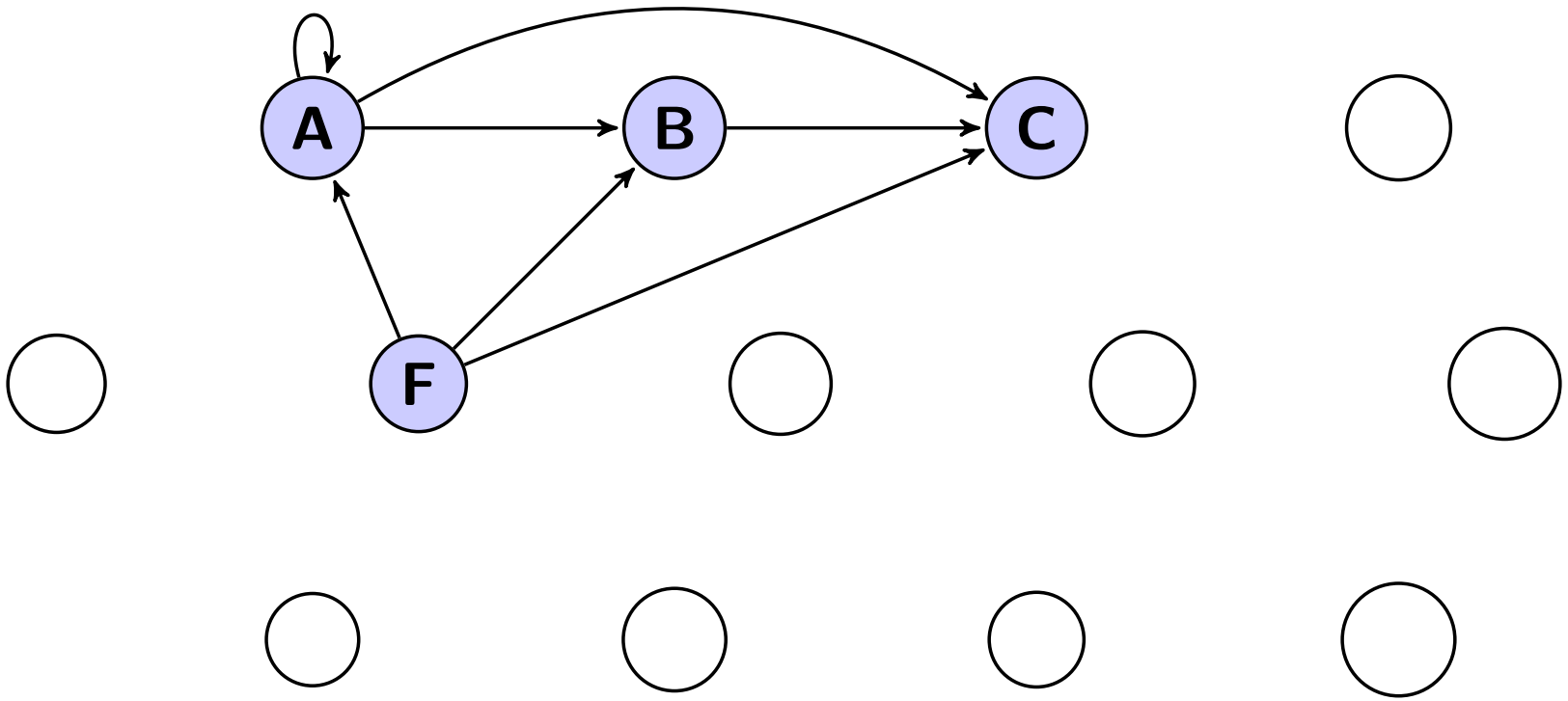
$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$

DAG-Recursion: eliminate D-node to get sub-DAG 4



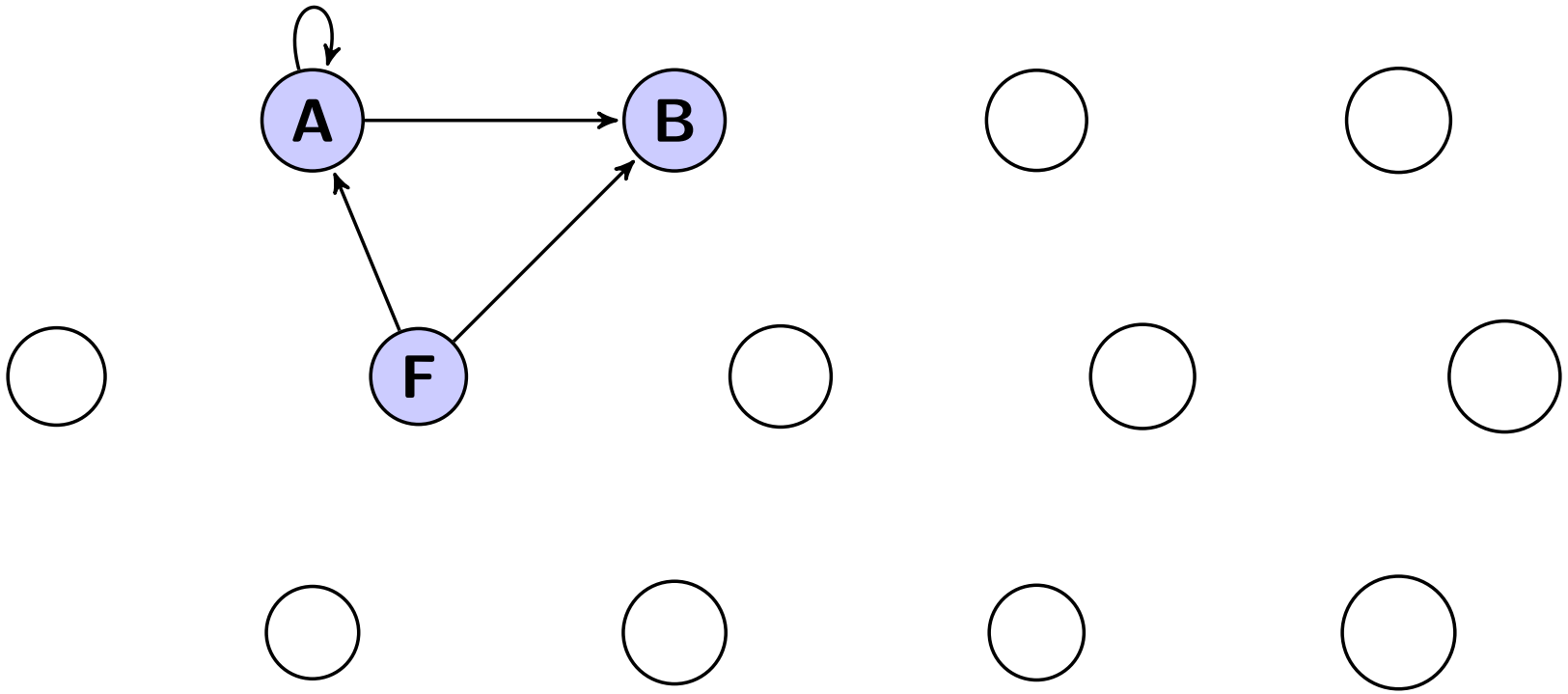
$$S_1 = \{G, M\},\ S_2 = \{K\},\ S_3 = \{L\},\ S_4 = \{I, J\},\ S_5 = \{E, H\}$$

DAG-Recursion: eliminate C-node to get sub-DAG 5



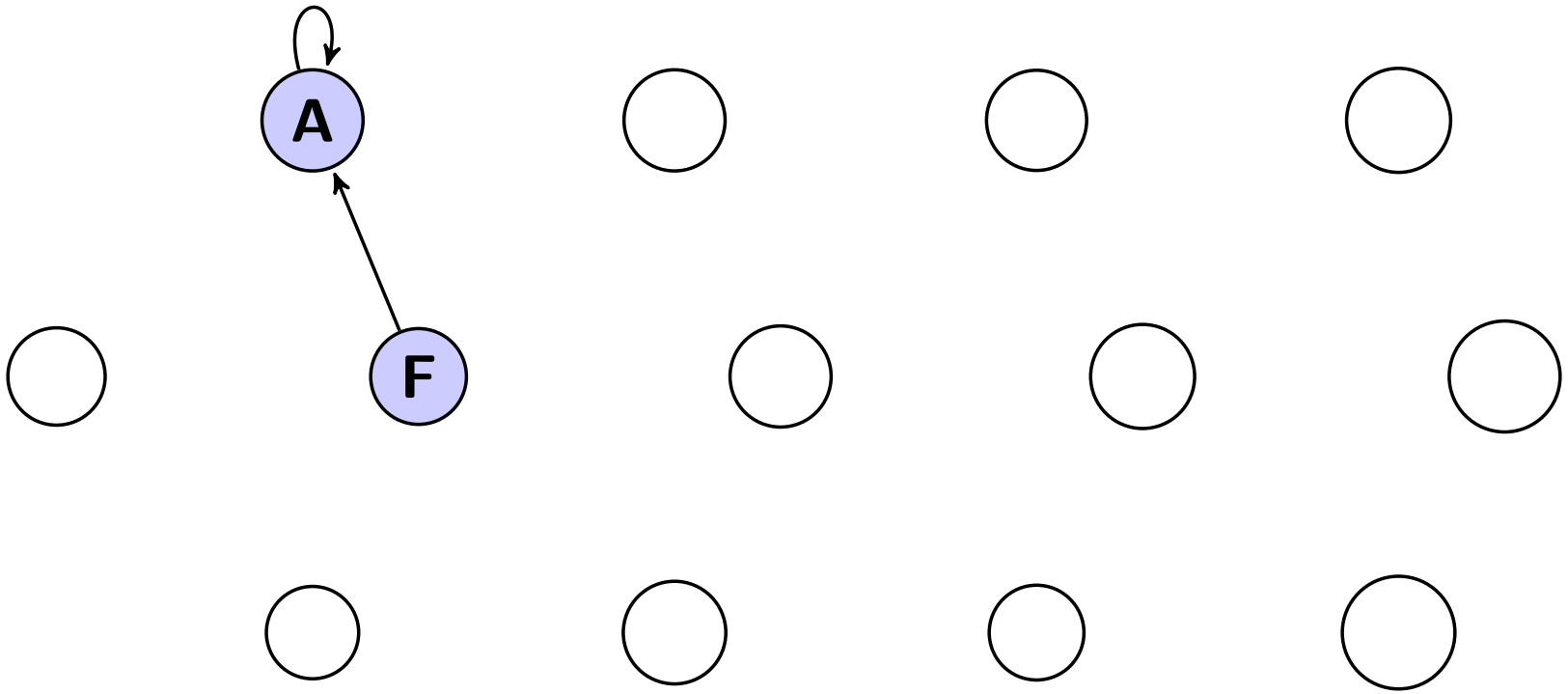
$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$, $S_5 = \{E, H\}$, $S_6 = \{D\}$

DAG-Recursion: eliminate B-node to get sub-DAG 6



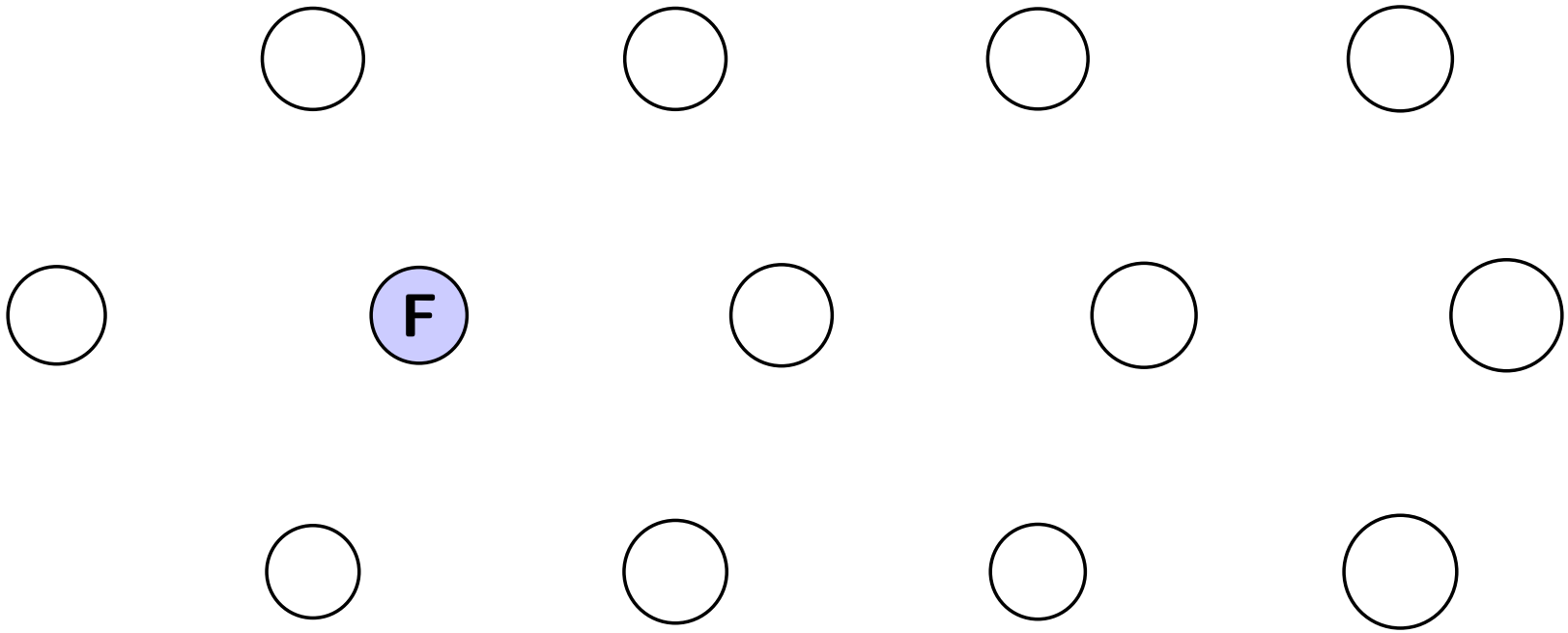
$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$, $S_5 = \{E, H\}$,
$S_6 = \{D\}$, $S_7 = \{C\}$

DAG-Recursion: eliminate A-node to get sub-DAG 7



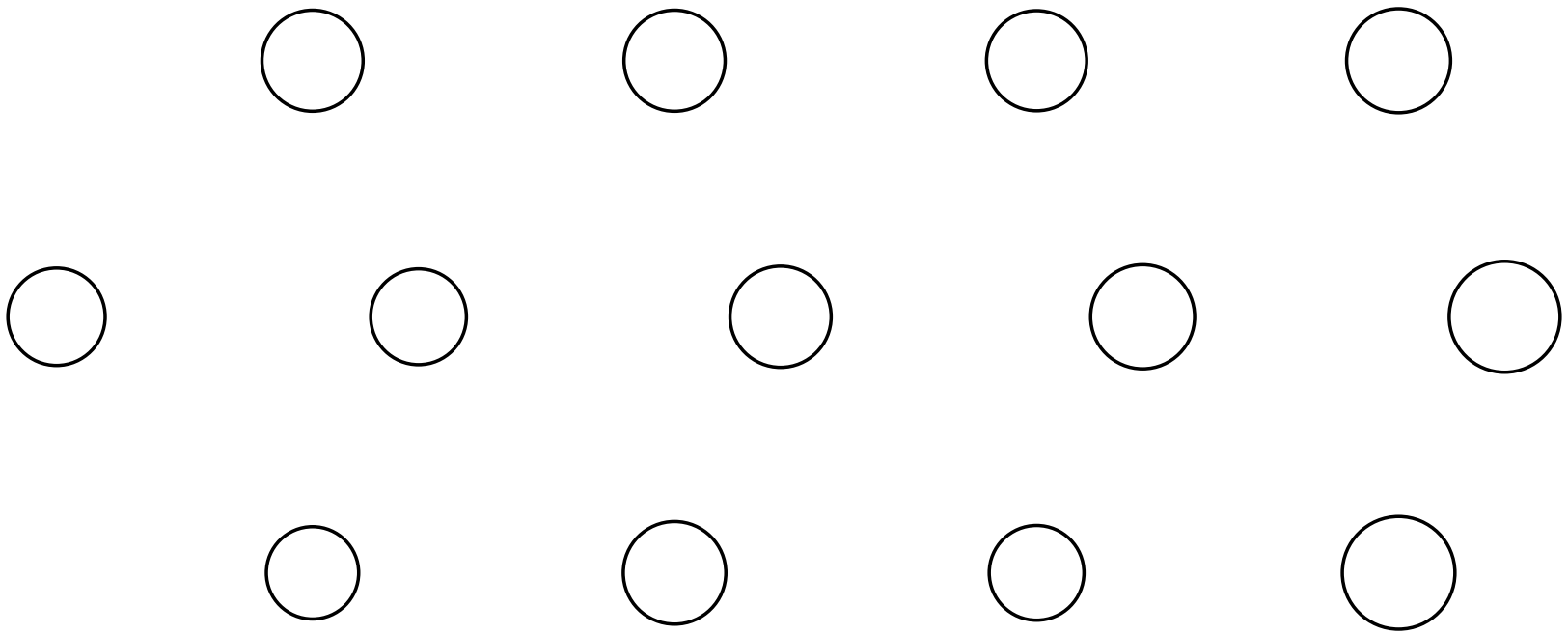
$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$, $S_5 = \{E, H\}$,
$S_6 = \{D\}$, $S_7 = \{C\}$, $S_8 = \{B\}$

DAG-Recursion: eliminate F-node to get sub-DAG 8



$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$, $S_5 = \{E, H\}$,
$S_6 = \{D\}$, $S_7 = \{C\}$, $S_8 = \{B\}$, $S_9 = \{A\}$

DAG-Recursion: we're done! No nodes left to eliminate



$S_1 = \{G, M\}$, $S_2 = \{K\}$, $S_3 = \{L\}$, $S_4 = \{I, J\}$, $S_5 = \{E, H\}$,
$S_6 = \{D\}$, $S_7 = \{C\}$, $S_8 = \{B\}$, $S_9 = \{A\}$, $S_{10} = \{F\}$

Subgames and stage games of DDG

- Given the partition $\{D_1, \ldots, D_{\mathcal{T}}\}$ of the directional component D
- Let $\Omega_\tau = \cup_{t=\tau}^{\mathcal{T}} S_t$ where $S_t = D_t \times X$
- $\mathcal{G}_\tau$ – **Stage τ subgame** of $\mathcal{G}$
  DDG with state space Ω_τ and other elements of the game (number of players, time horizon, utility functions, discount factors, action sets and laws of motion) be properly restricted for this state space versions of the element of the original game $\mathcal{G}$
- $\mathcal{G}_\tau(d)$ – **d-subgame** of $\mathcal{G}$
  Similarly defined subgame on state space
  $\Omega_\tau(d) = \{d \times X\} \cup \left(\cup_{t=\tau+1}^{\mathcal{T}} S_t\right)$

State recursion over subgames of DDGs

- State recursion involves finding a MPE of the overall game $\mathcal{G}$ inductively, starting by finding MPEs at all points in the *endgame*, i.e. the stage $\mathcal{T}$ subgame $\mathcal{G}_{\mathcal{T}}$, and proceeding by backward induction over the stages of the game, from stage $\mathcal{T}-1$ to stage $\mathcal{T}-2$ until the initial stage 1 is reached and solved.
- When stage 1 is solved in this backward induction procedure, effectively the whole $\mathcal{G}$ is also solved, as follows from the following lemma.

Lemma

If $\mathcal{G}$ is a finite state DDG, and $\mathcal{G}_1$ is its stage 1 subgame, then $\mathcal{G} = \mathcal{G}_1$.

Multiplicity of Equilibrium and Equilibrium Selection Rules

- $\mathcal{E}(\mathcal{G})$ – the set of all MPE of $\mathcal{G}$.
- Multiple MPEs in some of the d-subgames $\mathcal{G}_\tau(d) \rightarrow$ the equilibria in the d'-subgames at the earlier stages $\tau' < \tau$ will be *dependent* on which of the MPEs of the d-subgames is played at stage τ
- $\Rightarrow$ Solution computed with backward induction is *dependent* on the *equilibrium selection rule* (ESR) that selects one of the equilibria at every d-subgame of $\mathcal{G}$, and thus induces (or selects) a particular MPE in the whole game.
- $e(\mathcal{G}) \in \mathcal{E}(\mathcal{G})$ – a particular selected MPE from the set of all MPE of $\mathcal{G}$.

Definition of Equilibrium Selection Rule

Definition (Equilibrium selection rule)

Let Γ denote a *deterministic* rule for selecting one of the MPE from every d-subgame $\mathcal{G}_\tau(d)$, i.e.

$$e(\mathcal{G}_\tau(d)) = \Gamma\left(\mathcal{E}(\mathcal{G}_\tau(d))\right) \quad \forall d \in D. \tag{1}$$

- By selecting an equilibrium in every d-subgame, ESR Γ also induces (or selects) an equilibrium in every subgame $\mathcal{G}_\tau$, $e(\mathcal{G}_\tau) = \Gamma\left(\mathcal{E}(\mathcal{G}_\tau)\right)$.
- Define the projections $e_\sigma(\mathcal{G}) = \sigma^*$ and $e_V(\mathcal{G}) = V$ that pick each of these objects from a given equilibrium.

Stage τ Continuation strategies

Definition (Stage τ continuation strategy)

A *stage τ continuation strategy* $\sigma_\tau(s|S_\tau, e_\sigma(\mathcal{G}_{\tau+1}))$ is any feasible Markovian strategy for points $s \in S_\tau$ that reverts to a MPE strategy $e_\sigma(\mathcal{G}_{\tau+1})$ in the stage $\tau+1$ subgame $\mathcal{G}_{\tau+1}$. That is,

$$\sigma_\tau(s|S_\tau, e_\sigma(\mathcal{G}_{\tau+1})) = \begin{cases} \sigma(s) & \text{if } s \in S_\tau, \\ e_\sigma(\mathcal{G}_{\tau+1}) & \text{otherwise.} \end{cases} \tag{2}$$

Generalized Stage Games

Definition (Stage game)

Let $\mathcal{G}$ be a finite state DDG, and consider a particular stage of the game $\tau \in \{1, \ldots, \mathcal{T}\}$ and $d \in D_\tau$. A *d-stage game*, $\mathcal{SG}_\tau(d)$, is a d-subgame $\mathcal{G}_\tau(d)$ where the set of feasible strategies is restricted to continuation strategies, i.e. if $\Sigma(\mathcal{SG}_\tau(d))$ is the set of feasible, Markovian strategies of the stage game and $\Sigma(\mathcal{G}_\tau(d))$ is the set of feasible Markovian strategies of the d-subgame $\mathcal{G}_\tau(d)$, then we have $\sigma \in \Sigma(\mathcal{SG}_\tau(d))$ *iff*

$$\sigma(s) = \sigma_\tau(s|(d \times X), e_\sigma(\mathcal{G}_{\tau+1})), \quad s \in (d \times X) \cup \Omega_{\tau+1}. \tag{3}$$

State recursion decomposes a big game into smaller ones

- It follows that $\Sigma(\mathcal{SG}_\tau(d)) \subset \Sigma(\mathcal{G}_\tau(d))$, i.e. the set of feasible Markovian strategies in a d-stage game $\mathcal{SG}_\tau(d)$ is a subset of the set of feasible Markovian strategies in the d-subgame $\mathcal{G}_\tau(d)$.
- Similarly the set of feasible Markovian strategies in the stage game $\mathcal{G}_\tau$ is a subset of the feasible Markovian strategies in the stage τ subgame $\mathcal{G}_\tau$.
- By restricting strategies in this way, we reduce the problem of finding MPE strategies of a stage game $\mathcal{SG}_\tau(d)$ to the much smaller, more tractable problem of computing a MPE on the reduced state space $(d \times X)$ instead of the much larger state space $\Omega_\tau(d)$.

State recursion produces a subgame perfect equilibrium

Theorem (Subgame perfection)

Let $\mathcal{E}(\mathcal{SG}_\tau(d))$ be the set of all MPE of the stage game $\mathcal{SG}_\tau(d)$ and let $\mathcal{E}(\mathcal{G}_\tau(d))$ be the set of all MPE of the d-subgame $\mathcal{G}_\tau(d)$. Then we have

$$\mathcal{E}(\mathcal{SG}_\tau(d)) = \mathcal{E}(\mathcal{G}_\tau(d)) \tag{4}$$

i.e. there is no loss in generality from computing all MPE of every d-subgame $\mathcal{G}_\tau(d)$ by restricting the search for equilibria to finding all MPE of the corresponding stage game $\mathcal{SG}_\tau(d)$ using only continuation strategies.

State recursion algorithm

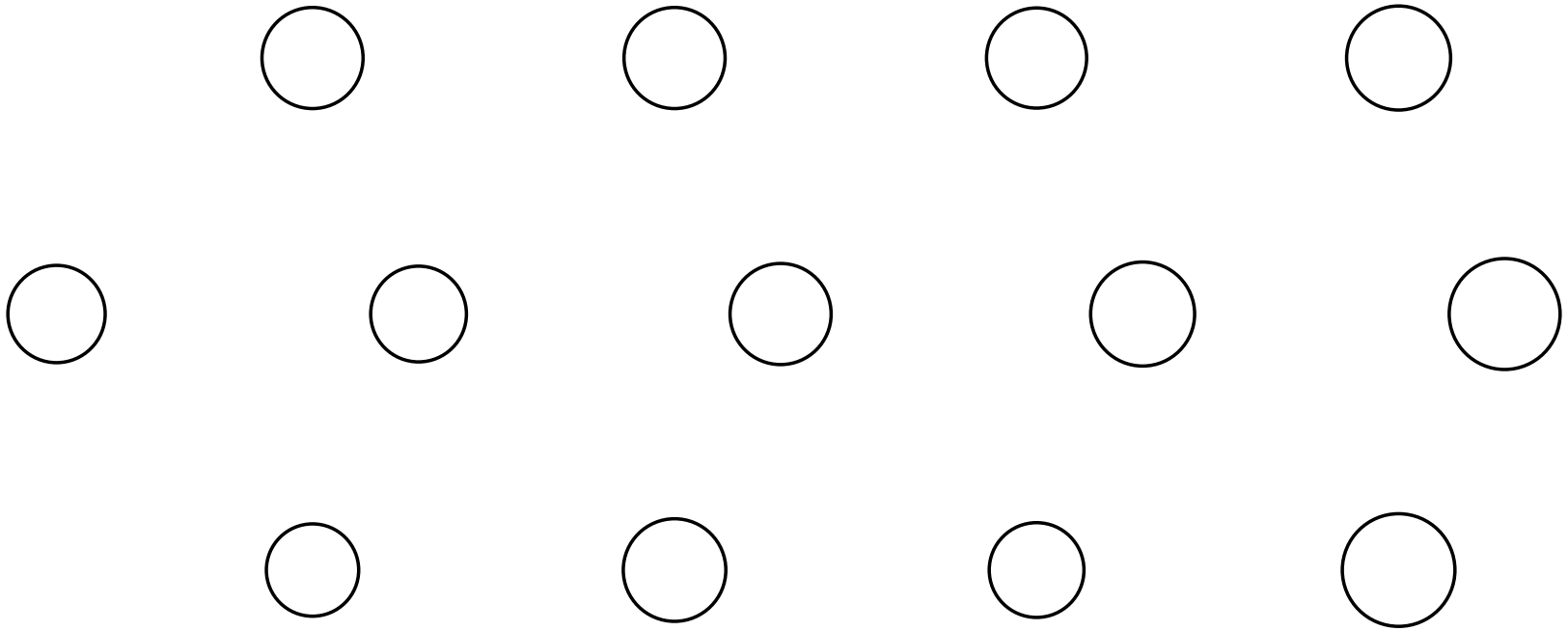
- For $\tau = \mathcal{T}, \mathcal{T}-1, \ldots, 1$ do
  - For $i = 1, \ldots, n_\tau$ do
    1. compute $\mathcal{E}(\mathcal{SG}_\tau(d_{i,\tau}))$
    2. using an equilibrium selection rule Γ, select a particular MPE from $\mathcal{E}(\mathcal{SG}_\tau(d_{i,\tau}))$, $e(\mathcal{SG}_\tau(d_{i,\tau})) = \Gamma(\mathcal{E}(\mathcal{SG}_\tau(d_{i,\tau})))$
    3. $e(\mathcal{SG}_\tau(d_{i,\tau}))$ is a MPE of the d-subgame $\mathcal{G}_\tau(d_{i,\tau})$
  - union of the MPEs for each d-stage game $\{e(\mathcal{SG}_\tau(d_{i,\tau}) | i = 1, \ldots, n_\tau\}$ is a MPE for the stage game at stage τ $e(\mathcal{G}_\tau)$
- $\mathcal{E}(\mathcal{G}_1) = \mathcal{E}(\mathcal{G})$

Convergence of the state recursion algorithm

Theorem (Convergence of State Recursion)

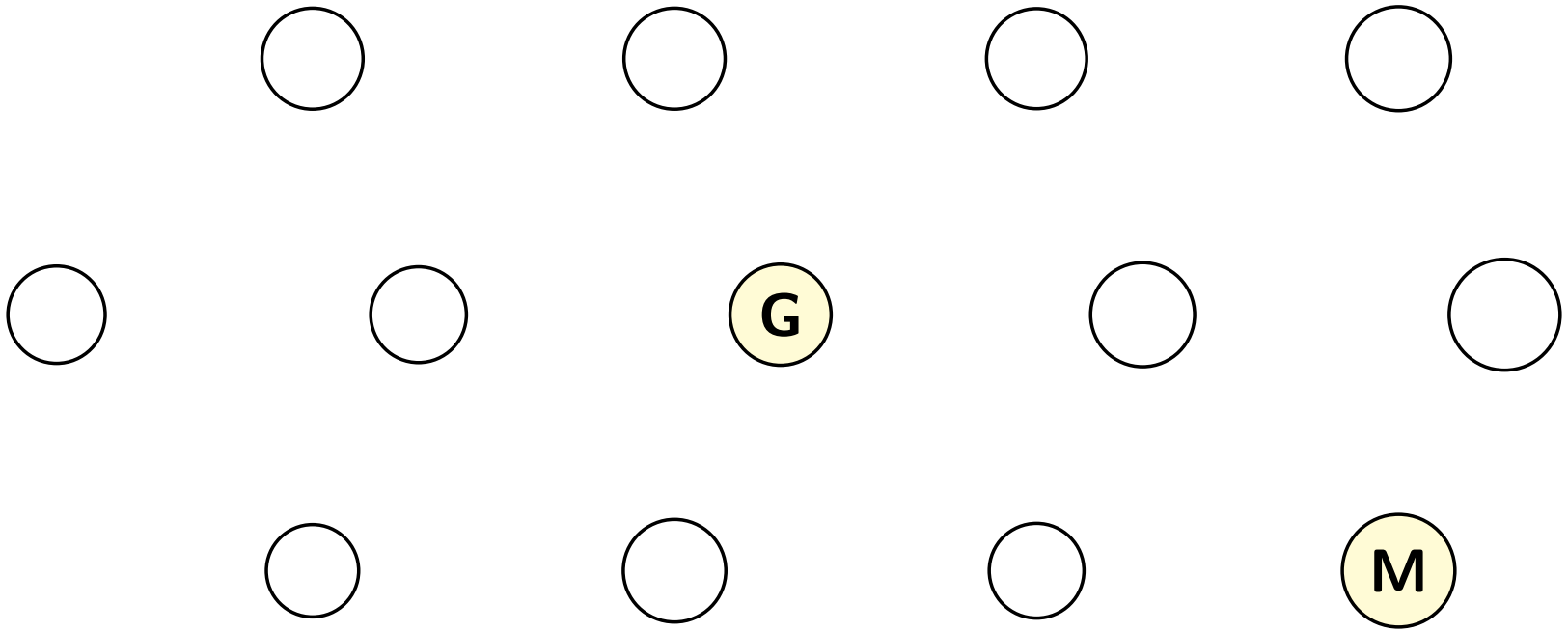
Let $\mathcal{G}$ be a finite state DDG.
Given an equilibrium selection rule Γ the state recursion algorithm computes an MPE of $\mathcal{G}$.

State Recursion: DAG recursion in reverse



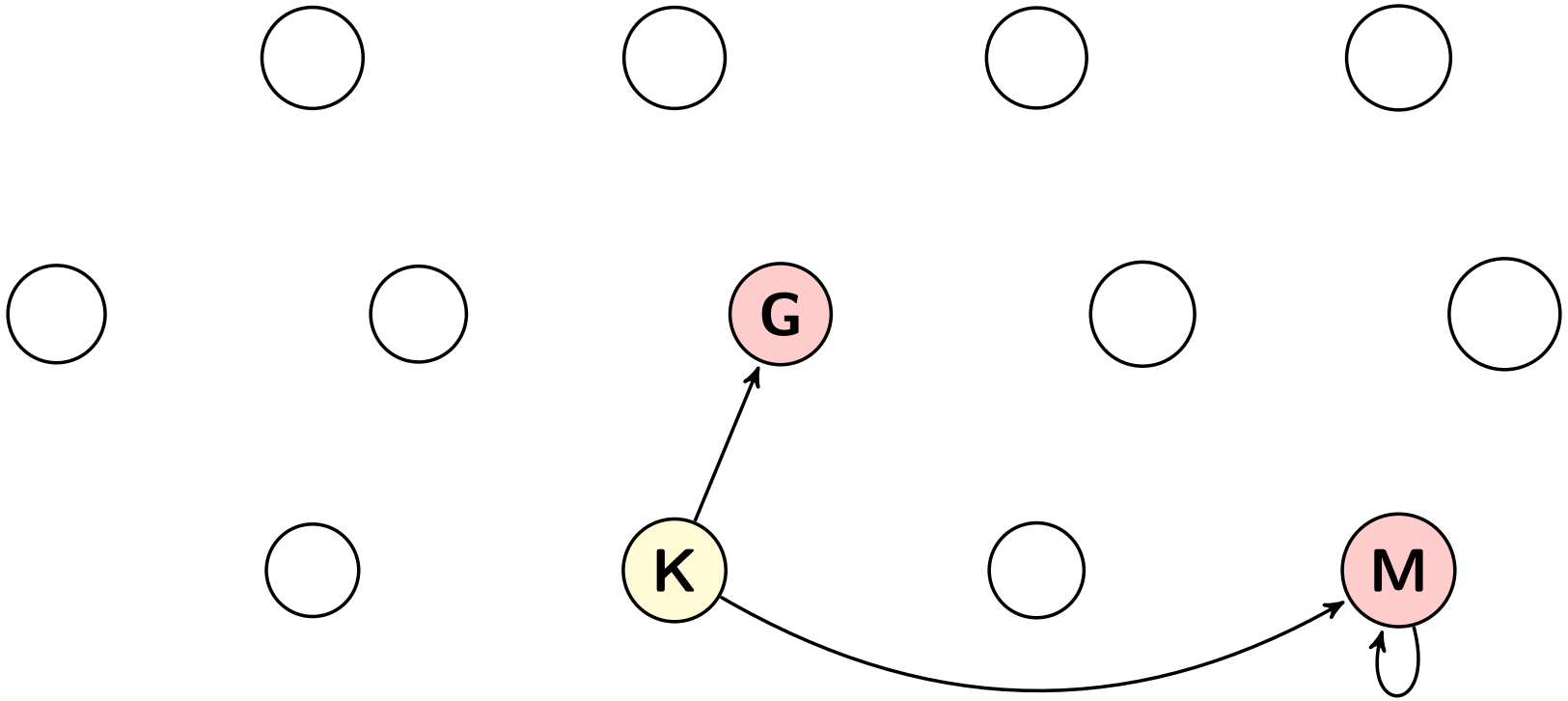
To get ready to do state recursion, which is a backward induction in the DAG, reverse index the stages of the game
$S_{10} = \{G, M\}$, $S_9 = \{K\}$, $S_8 = \{L\}$, $S_7 = \{I, J\}$, $S_6 = \{E, H\}$, $S_5 = \{D\}$, $S_4 = \{C\}$, $S_3 = \{B\}$, $S_2 = \{A\}$, $S_1 = \{F\}$

State Recursion step $\mathcal{T} = 10$: the "end game"



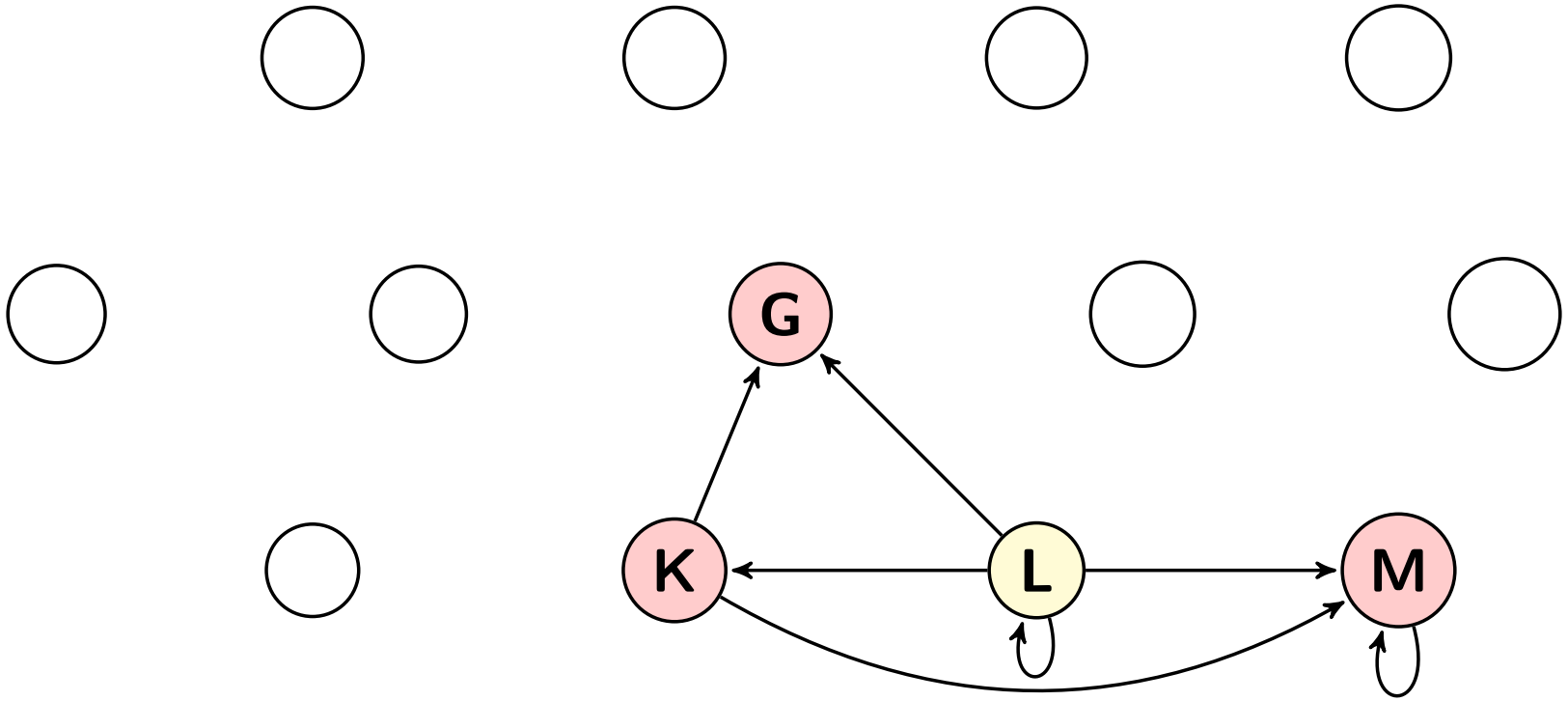
Find all equilibria in the *end game states* $S_{10} = \{G, M\}$

State Recursion $\mathcal{T} - 1 = 9$: Now find all MPE in $S_9 = \{K\}$



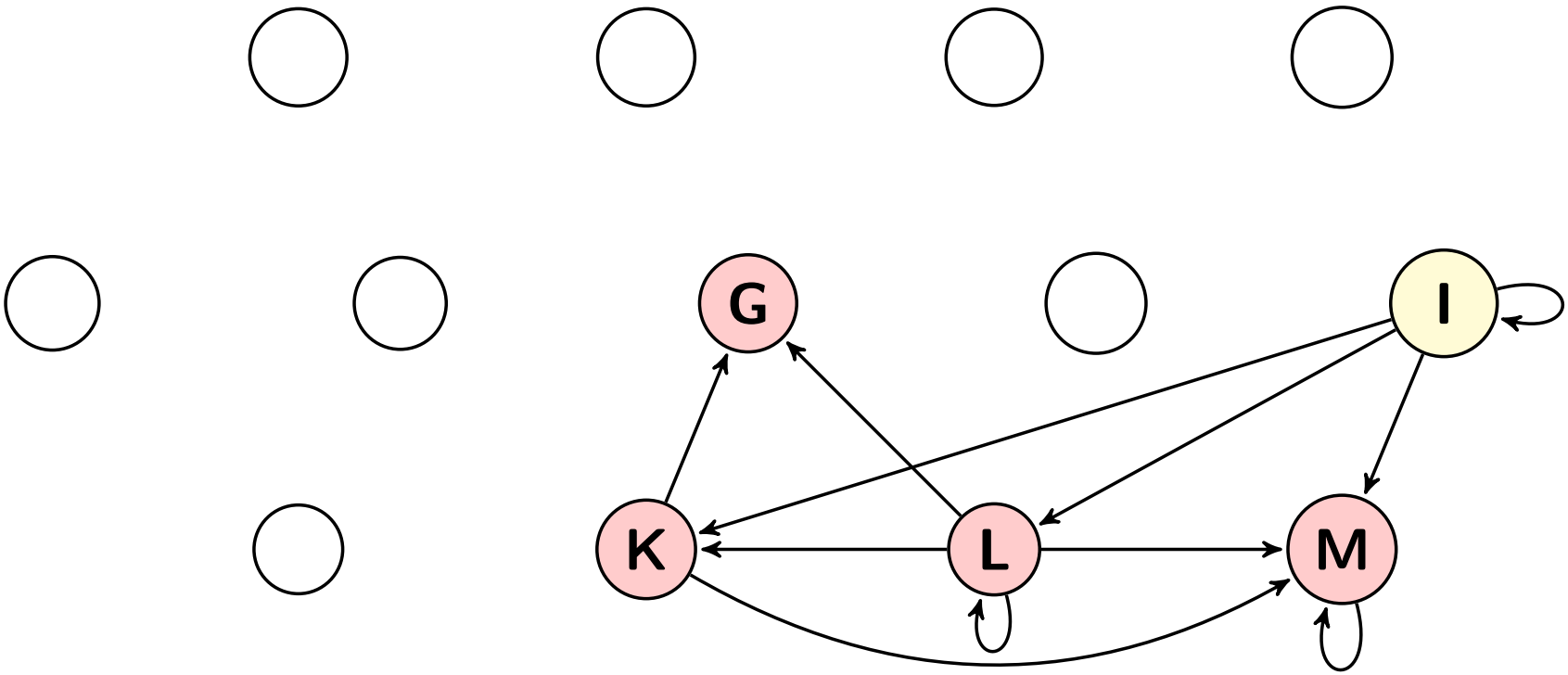
Find all MPE in the *generalized stage game* $S_9 = \{K\}$

State Recursion step $\mathcal{T} - 2 = 8$: Find all MPE in $S_8 = \{L\}$



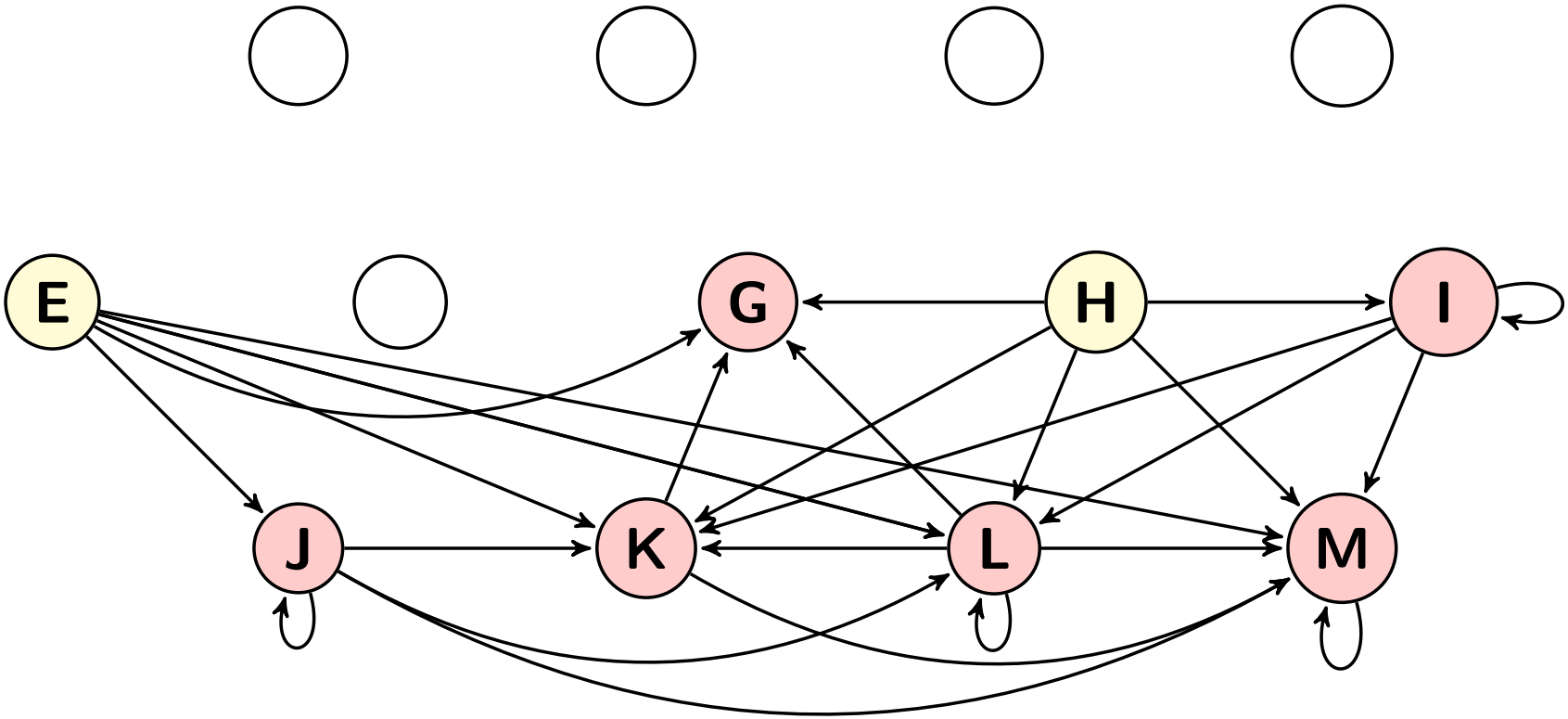
Find all MPE in the generalized stage game $S_8 = \{L\}$

State Recursion step $\tau = 7$: Find all MPE in $S_7 = \{I, J\}$



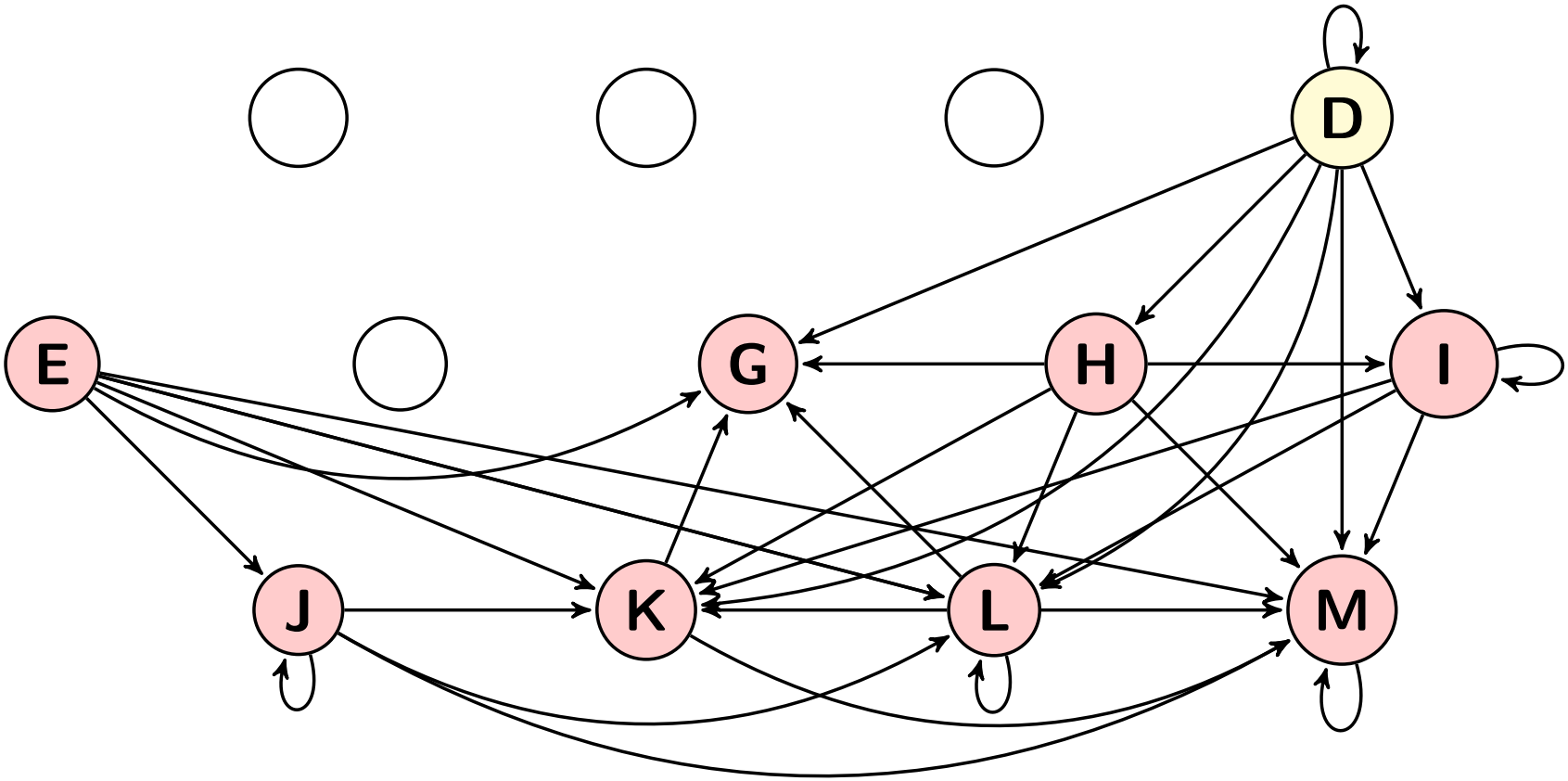
Find all MPE in the generalized stage game $S_7 = \{I, J\}$

State Recursion step $\tau = 6$: Find all MPE in $S_6 = \{E, H\}$



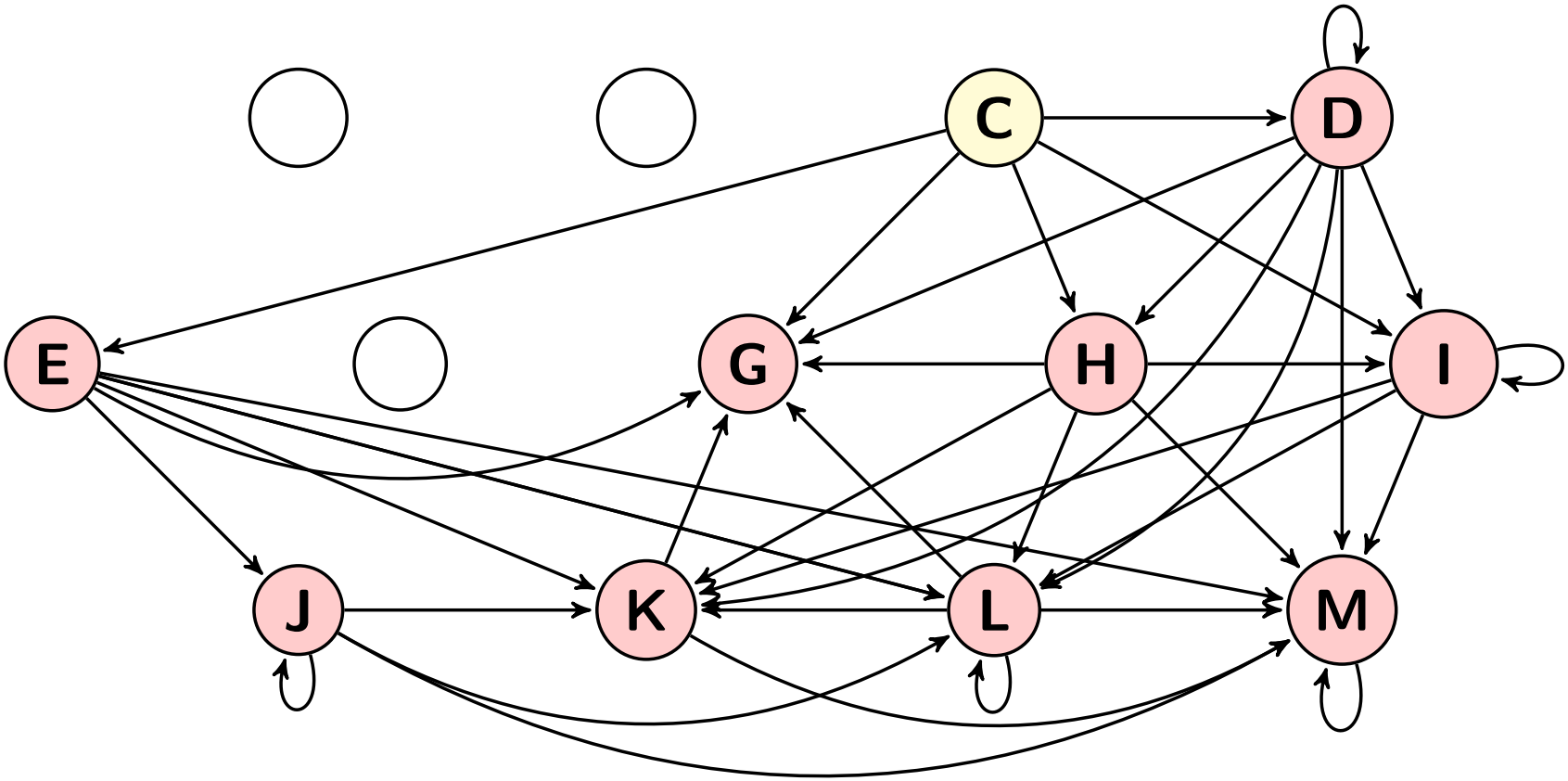
Find all MPE in the generalized stage game $S_6 = \{E, H\}$

State Recursion step $\tau = 5$: Find all MPE in $S_5 = \{D\}$



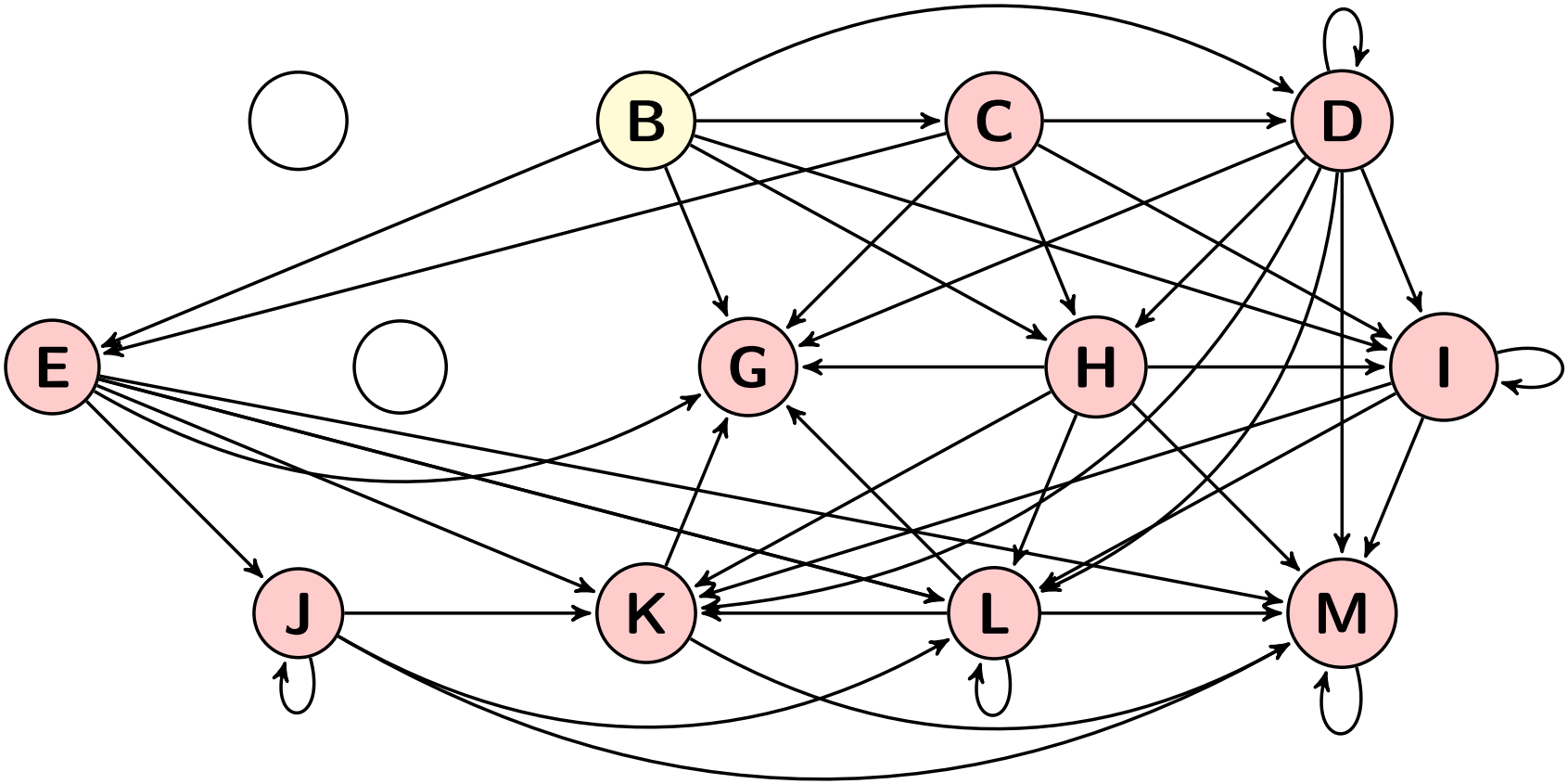
Find all MPE in the generalized stage game $S_5 = \{D\}$

State Recursion step $\tau = 4$: Find all MPE in $S_4 = \{C\}$



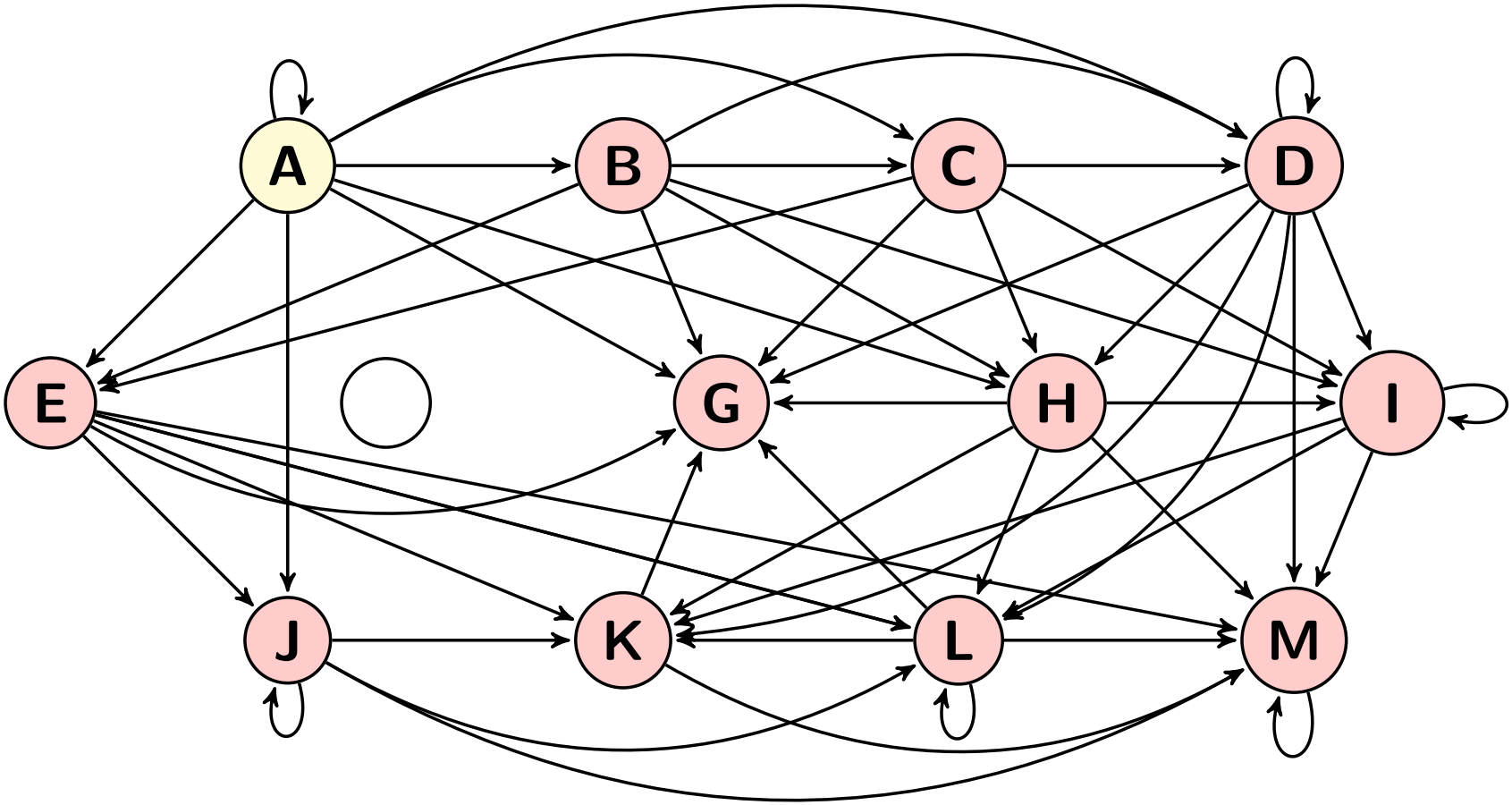
Find all MPE in the generalized stage game $S_4 = \{C\}$

State Recursion step $\tau = 3$: Find all MPE in $S_3 = \{B\}$



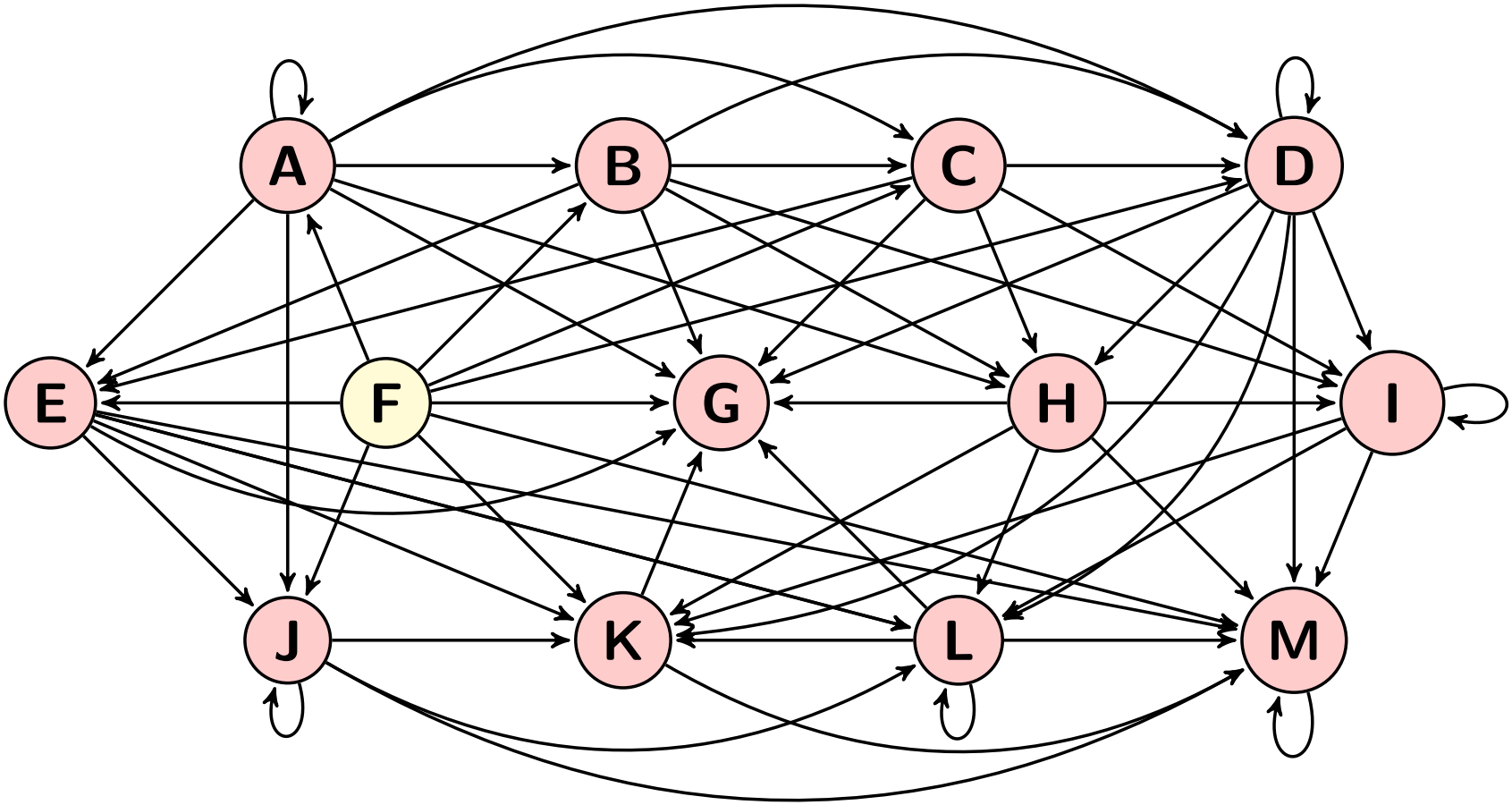
Find all MPE in the generalized stage game $S_3 = \{B\}$

State Recursion step $\tau = 2$: Find all MPE in $S_2 = \{A\}$



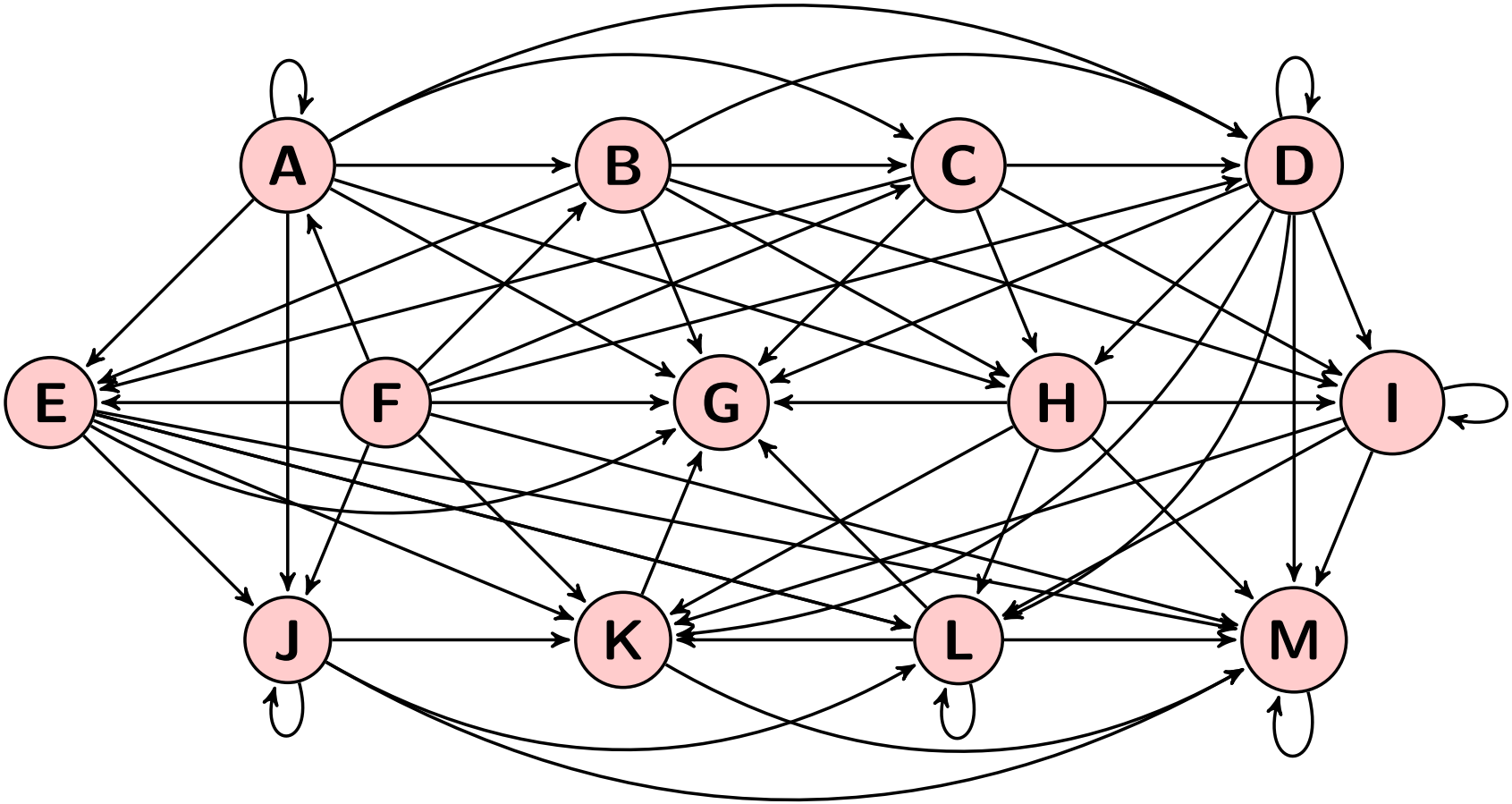
Find all MPE in the generalized stage game $S_2 = \{A\}$

State Recursion step $\tau = 1$: Find all MPE in $S_1 = \{F\}$



Find all MPE in the generalized stage game $S_1 = \{F\}$

We're done: state recursion has found a MPE of $\mathcal{G}$



It found *one* MPE of the DDG $\mathcal{G}$ given ESR Γ

Finding all MPE

Approach:

1. Formalize the notion of ESR Γ on a computer
2. Make a loop over all ESR and solve the model for each Γ

Problems:

- 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

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

Properties of RLS algorithm:

- Complete: Computes all MPE equilibria of the game
- Fast: time spent of search of feasible ESRs is negligible in comparison to time spent on solving the game
  - Efficiently skip infeasible ESRs
  - Re-use results of previously computed subgames

Assumptions of RLS

1. There is a method to find *all* MPE in every d-stage game (i.e. equilibria within the class of continuation strategies)
2. The number of equilibria in every d-stage game is finite
3. $\Rightarrow$ RLS finds *all* MPE of the DDG $\mathcal{G}$.

- RLS also works if this algorithm can only find *some* of the equilibria of d-stage games.
- In the latter case RLS is not guaranteed to find *all* MPE of $\mathcal{G}$, but it can still find, potentially, *very many* MPE of $\mathcal{G}$.

Efficiency of RLS

1. **[Decompose]** State recursion decomposes a large, complex game into a sequence of much simpler component stage games
2. **[Reuse]** RLS minimizes the computational cost when if traverses the set of all MPE
3. **[Jump]** RLS is caplable to jump from one feasible ESR to next in one step

Equilibrium Selection Strings (ESS)

ESS formalizes the ESR Γ

- K – the least upper bound on the number of possible equilibria in any stage game of $\mathcal{G}$.
  Implementation does not require the prior knowledge of K
- N – the total number of d-substages of the DDG $\mathcal{G}$.
  The state recursion algorithm must loop over all N of these substages to find a MPE in the stage games that correspond to each of these N d-stages to construct a MPE of $\mathcal{G}$.
- Equilibria in every d-stage games are indexed $\{0, 1, \ldots, K-1\}$

Equilbrium Selection Strings (ESS), continued

Definition (Equilibrium Selection Strings)

An *equilibrium selection string* (ESS), denoted by γ, is a vector in Z_+^N whose components are integers expressed in base K arithmetic, i.e. each coordinate (or "digit") of γ takes values in the set $\{0, 1, \ldots, K-1\}$.

$$\gamma = (\gamma_T, \gamma_{T-1}, \ldots, \gamma_1),$$

$$\gamma_\tau = (\gamma_{1,\tau}, \ldots, \gamma_{n_\tau,\tau})$$

$\gamma^0 = (0, \ldots, 0)$ – the initial ESS that consists of N zeros, always feasible assuming at least one equilibrium exists in every d-stage game

Ordering of stages/digits in ESSs

- Elements of γ are ordered in a particular way
- γ_τ are ordered from right to left corresponding to stages of $\mathcal{G}$ from τ to $\mathcal{T}$
- Higher digits of ESS correspond to later stages of $\mathcal{G}$
- $\Rightarrow$ when digit j is changed in the loop over ESS, only stages to the right $(\tau < j)$ have to be resolved

Enumerating all possible Equilibrium Selection Strings

- K^N possible equilibrium strings for the DDG $\mathcal{G}$
  this is an upper bound on the number of possible MPE of $\mathcal{G}$

- The loop starts from $\gamma^0 = (0, \ldots, 0)$
- One step in the loop is mod(K) addition $+1$
- Second ESR is $\gamma^1 = (0, \ldots, 0, 1) = 1$ in base-K representation

Does γ^1 correspond to an MPE of $\mathcal{G}$?

- $\gamma_1 = (0, 0, \ldots, 0, 1)$ may or may not correspond to a MPE of $\mathcal{G}$ because there may be only a *single* MPE at the d_{1,n_1}-stage game of $\mathcal{G}$.
- If there is only a single MPE in this substage $\rightarrow$ the equilibrium string γ_1 is *infeasible* because the only feasible equilirbrium index for the first d-stage is 0

Definition (Feasible Equilibrium Selection String)

An equilibrium string γ is *feasible* if all of its digits index a MPE that exists at each of the corresponding d-stage games of $\mathcal{G}$.

Tracking the number of MPE at each substage

Definition ($ne(\gamma)$ string)

Let the $N \times 1$ vector $ne(\gamma)$ be the maximum number of MPE at each stage game of $\mathcal{G}$ under the ESR implied by the equilibrium string γ. Define $ne(\gamma)$ using the same format as the equilibrium string, so that the digits of the equilibrium string γ are in one to one correspondence with the elements of the vector $ne(\gamma)$ as follows:

$$ne(\gamma) = \left(ne_{\mathcal{T}}, ne_{\mathcal{T}-1}\left(\gamma_{>\mathcal{T}-1}\right), \ldots, ne_1\left(\gamma_{>1}\right)\right),$$

where $\gamma_{>\tau} = \left(\gamma_{\tau+1}, \ldots, \gamma_{\mathcal{T}}\right)$ is a $\mathcal{T} - \tau \times 1$ vector listing the equilibrium selection sub-string for stages of $\mathcal{G}$ higher than τ.

Characterization of Feasible ESSs

- We use the notation $ne_\tau(\gamma_{>\tau})$ to emphasize that the number of MPE at stage τ depends only on the equilibria selected at higher stages of $\mathcal{G}$.
- Notice that in the endgame $\mathcal{T}$ there are no further stages of the game, so the maximum number of MPE in this stage, $n_\mathcal{T}$ does not depend on any substring of the equilibrium string γ.

Lemma

The ESS γ is feasible if and only if

$$\gamma_{i,\tau} < ne_{i,\tau}(\gamma_{>\tau}), \quad \tau = 1, \ldots, \mathcal{T}, \quad i = 1, \ldots, n_\tau$$

The Jump Function

$\mathcal{J}(\gamma)$ is the "smallest" ESS *after* γ that is also a feasible ESS.

Definition (Jump function)

Let $\mathcal{J} : Z_+^N \to Z_+^N \cup [\text{STOP}]$ be defined by

$$\mathcal{J}(\gamma) = \begin{cases} \textit{argmin}_{\gamma'}\{\gamma' : \gamma' > \gamma \text{ and } \gamma' \text{ is feasible}\} \\ [\text{STOP}] \text{ if } \nexists \gamma' : \gamma' > \gamma \text{ and } \gamma' \text{ is feasible} \end{cases}$$

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		

ESS γ in variable base arithmetics

- Bases: $ne(\gamma) = \left(ne_T, ne_{T-1}\left(\gamma_{>T-1}\right), \ldots, ne_1\left(\gamma_{>1}\right)\right)$
- Successor function $\mathcal{S} : Z_+^N \to Z^N : \; S(\gamma) = \gamma + 1$
- Defined correctly in variable bases:

$$\gamma{+}1 = \begin{cases} (\quad \ldots \quad, \gamma^{(1)} + 1) & \text{if } \gamma^{(1)} + 1 < ne^{(1)}, \\ (\ldots, \gamma^{(j-k)+1}, 0, \ldots, 0) & \text{otherwise,} \end{cases}$$

where $k : \; \gamma^{(j-k)+1} < ne^{(j-k)}$

- In the latter case dependent bases change to $\left(ne^{(1)}, \ldots, ne^{(j-k)}, \tilde{ne}^{(j-k+1)}, \ldots, \tilde{ne}^{(N)}\right)$
- But $\gamma + 1$ is still a well-defined number in this new bases because all digits with new bases are zeros

Relation between the Successor and Jump Function

- $\mathcal{S}'$ – augmented with $[STOP]$ signal when more than N digits are needed for the successor result

Theorem

Assume that ESS are expressed in variable bases as above.
If γ is a feasible ESS, then $\mathcal{J}(\gamma) = \mathcal{S}'(\gamma)$.

- In other words, when the bases for the equilibrium selection string are chosen in the right way, the function that returns next feasible ESS is just a successor function
- The number of steps RLS takes is the same as number of MPE in the model!

RLS Algorithm

1. Set $\gamma^0 = (0, \ldots, 0)$, $k = 0$
2. Solve for an MPE given the ESS γ^k with State Recursion
3. Fix the bases for the ESS at computed $ne(\gamma)$
4. Apply a successor function to find next feasible ESS

$$\gamma^{k+1} = S'(\gamma^k)$$

5. Stopping rule: $\gamma^{k+1} \stackrel{?}{=} [\mathrm{STOP}]$
6. Return to step 2

Main result of the RLS Algorithm

Theorem (Decomposition theorem)

Assume there exists an algorithm that can find all MPE of every stage game of the DDG $\mathcal{G}$, and that the number of these equilibria is finite in every stage game.
Then the RLS algorithm finds all MPE of DDG $\mathcal{G}$ in at most $|\mathcal{E}(\mathcal{G})|$ steps, which is the total number of MPE of the DDG $\mathcal{G}$.

Example: The leapfrogging model

Basic Setup

- Discrete time, infinite horizon ($t = 1, 2, 3, \ldots$)
- Two firms, homogenous/differentiated goods, no entry or exit
- Each firm maximizes expected discounted profits
- Bertrand competition: set product prices (simultaneously)

Investment decision: Whether to invest in state of the art production technology

- Pay investment cost of $K(c)$ to obtain marginal cost $c \geq 0$
- Time to build: state of the art technology is operational after a one period lag
- State of the art costs follows exogenous Markov process and only improves

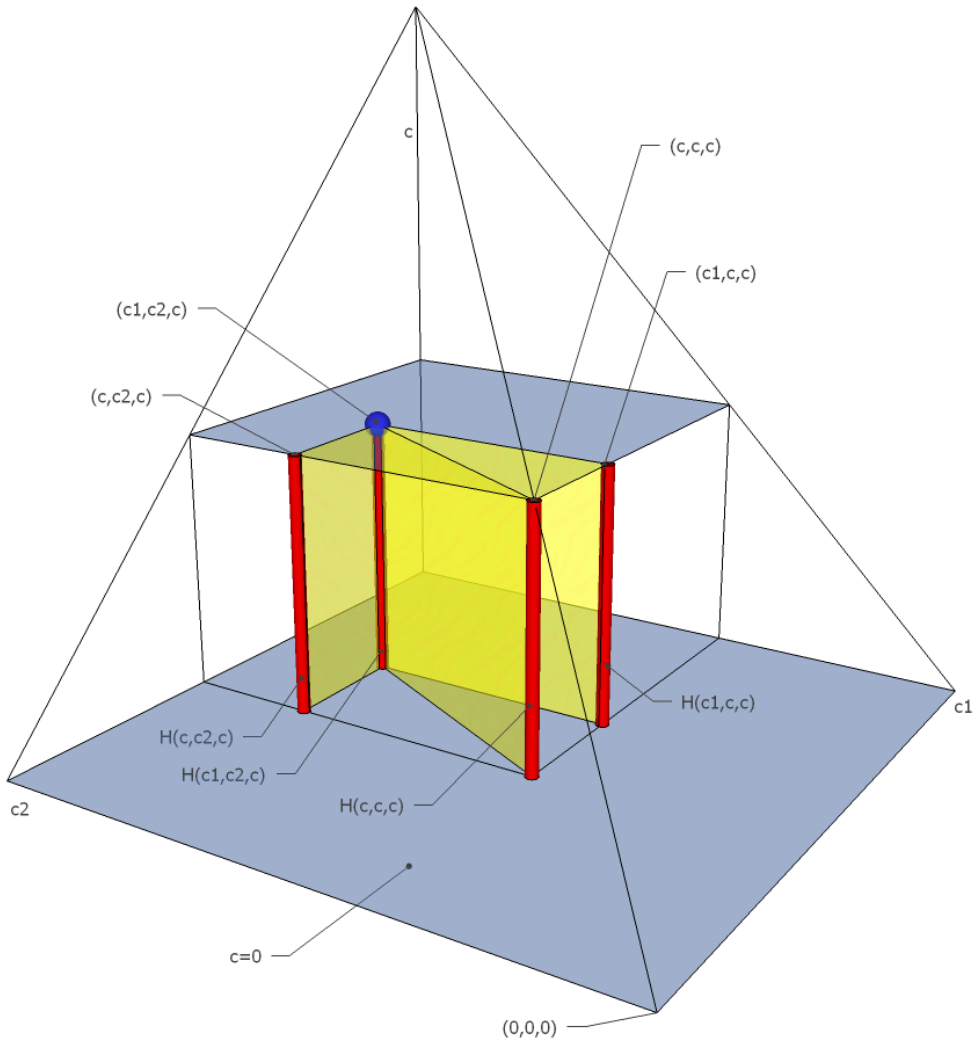
Example: The leapfrogging model - cont.

Timing of cost reducing investment decisions

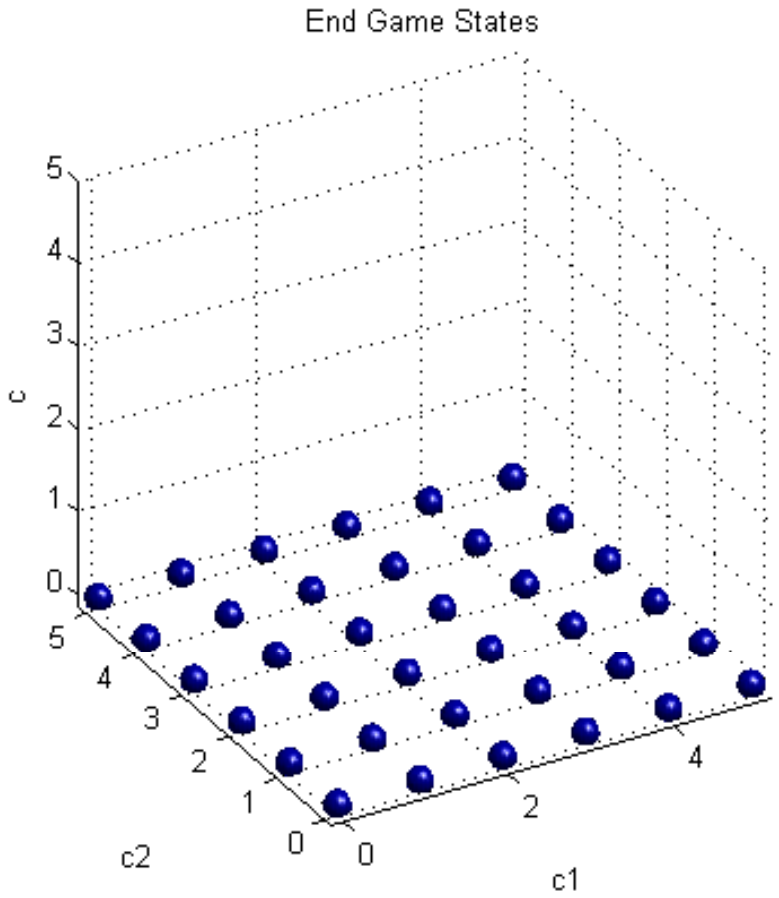
1. *Simultaneous moves*:
   - Fully directional, $d = (c_1, c_2, c)$
2. *Alternating moves*:
   - The right to move, m, follows a Markov process (deterministic alternation as a special case).
   - m is clearly a non-directional,
   - but game still have a directional component, $d = (c_1, c_2, c)$

State space of the game: a "quarter pyramid"

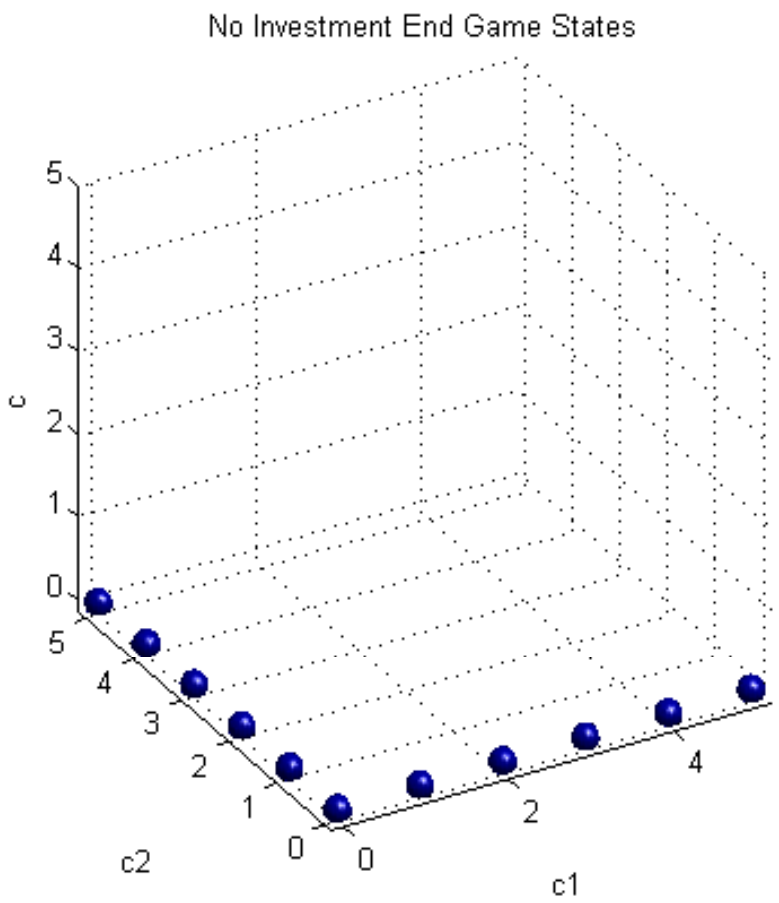
$S = \{(c_1, c_2, c) | c_1 \geq c,\ c_2 \geq c, c \in [0, \overline{c}]\}$



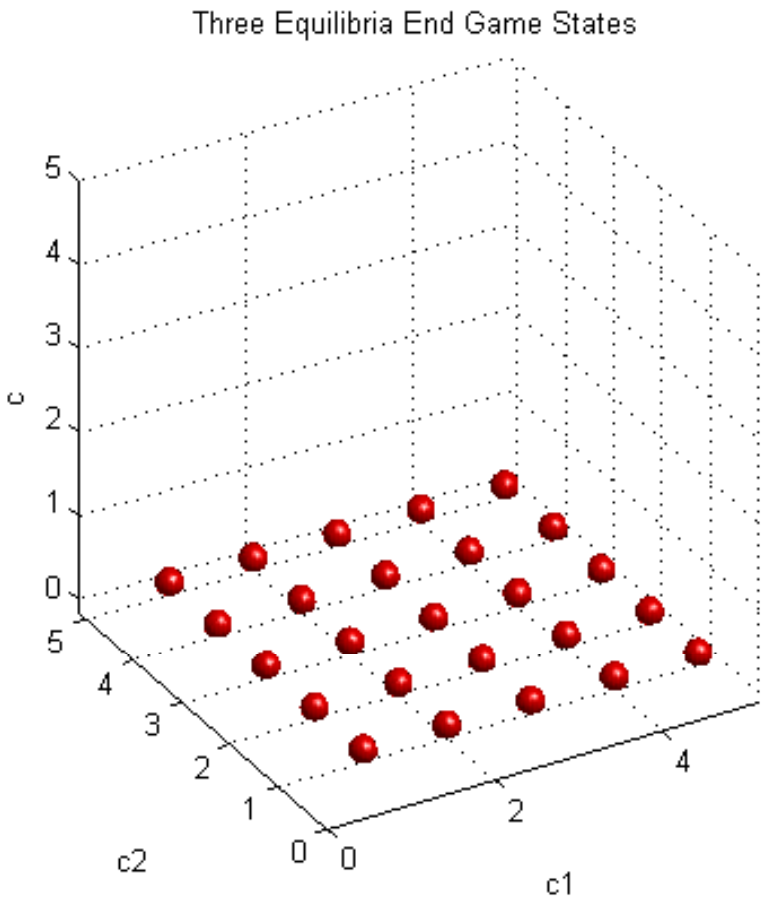
End Games States



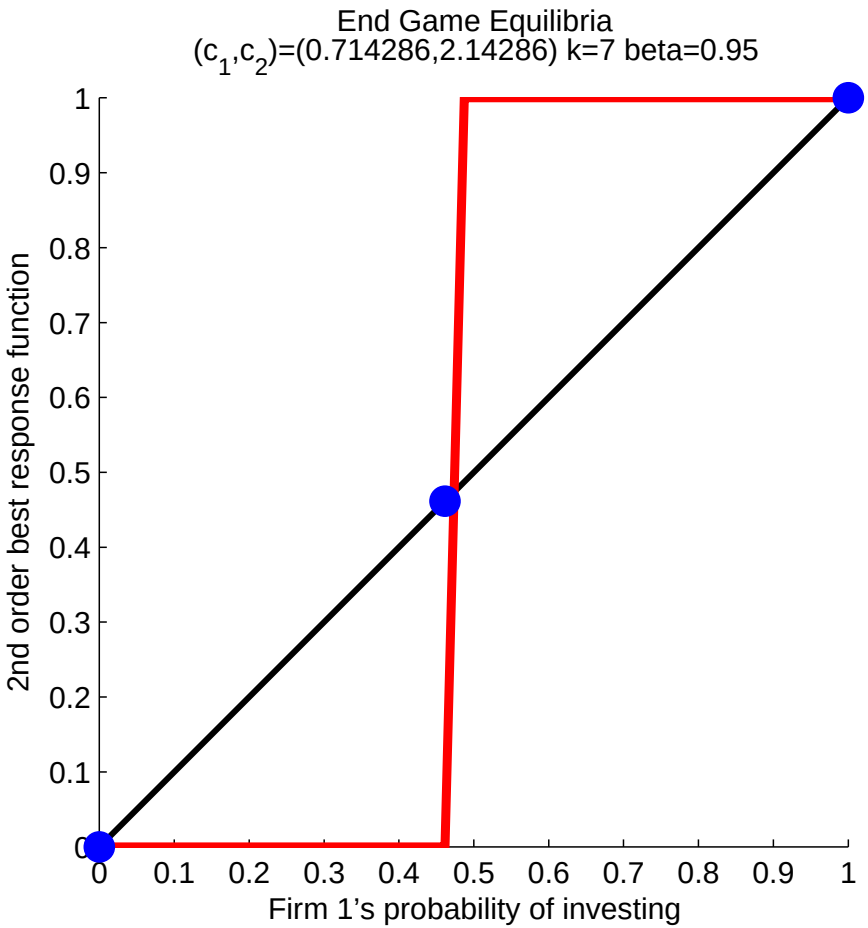
No Investment End Games States



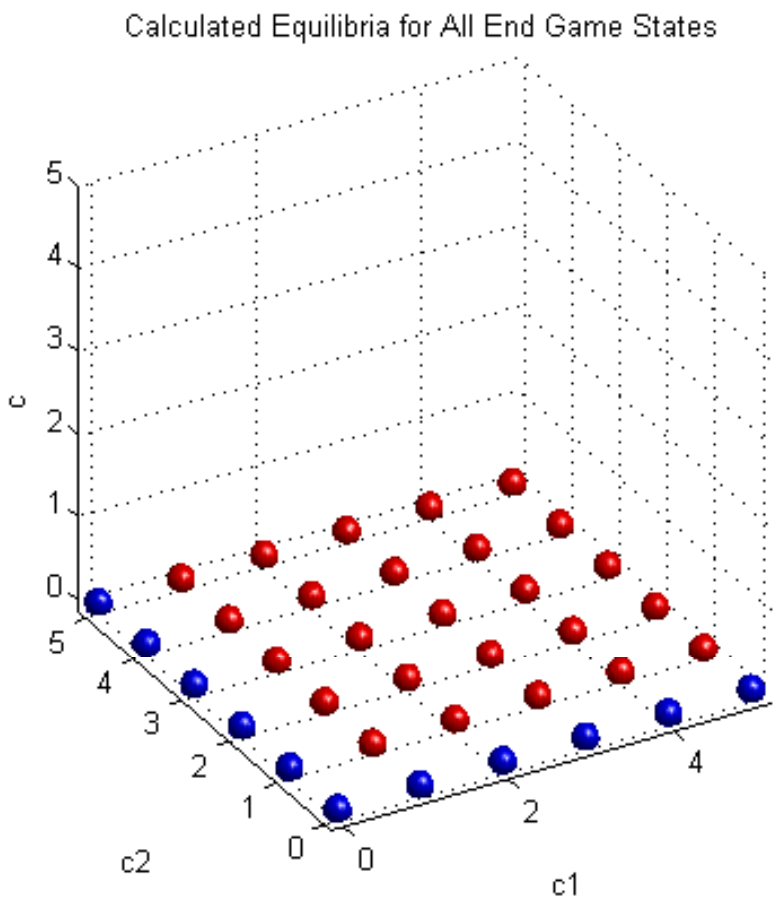
Multiple Equilibria End Games States



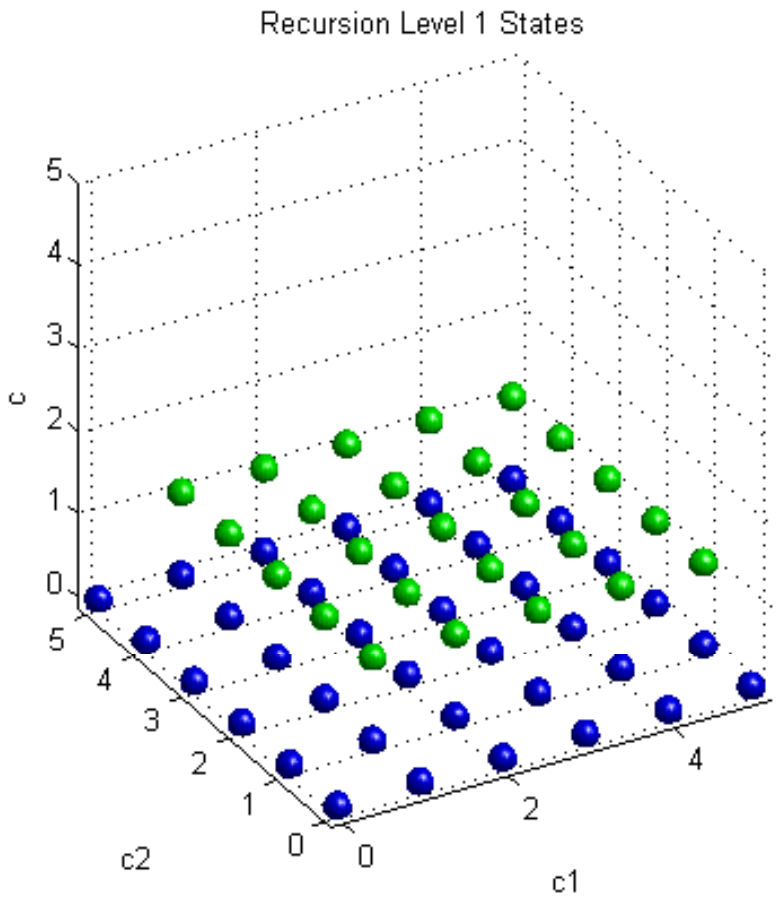
Second order best response function, $\eta = 0$



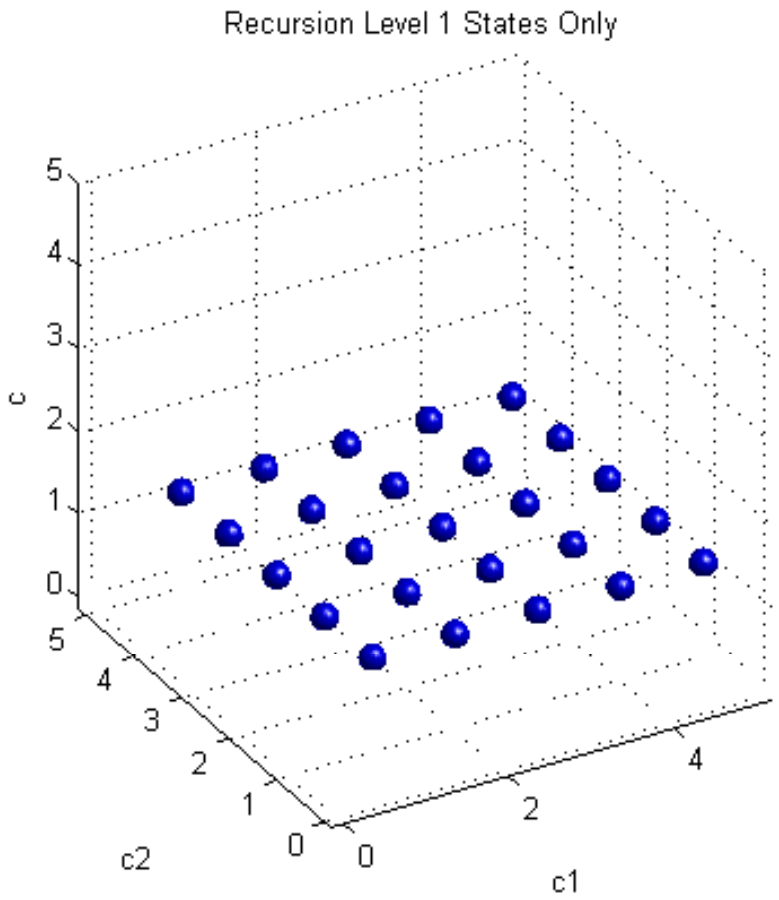
Calculated Equilibria for End Games States



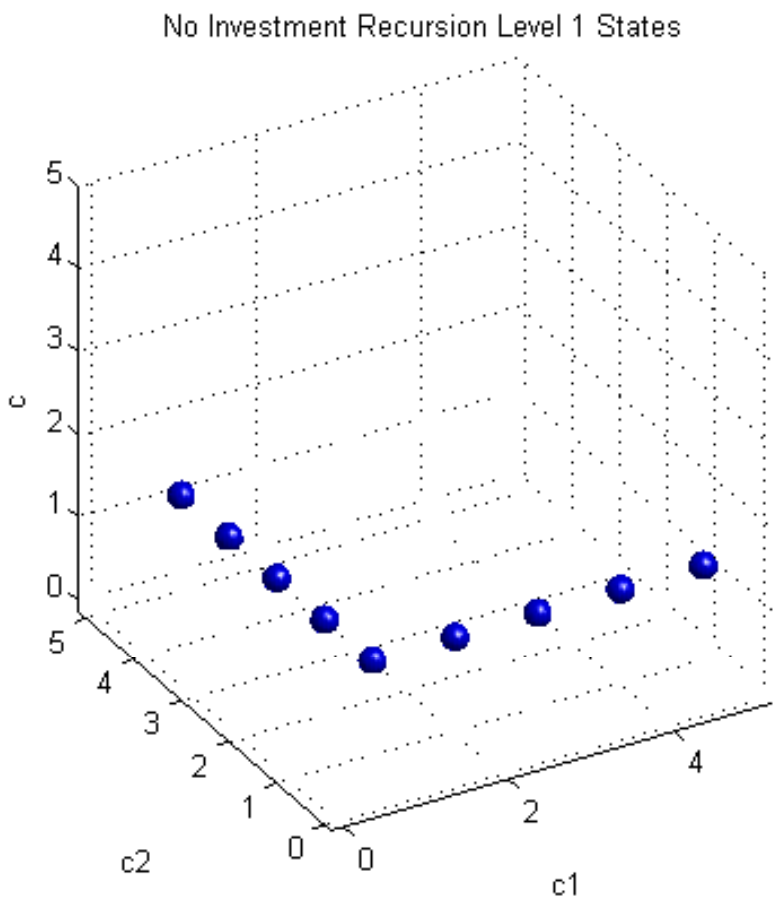
Recursion Level 1 States



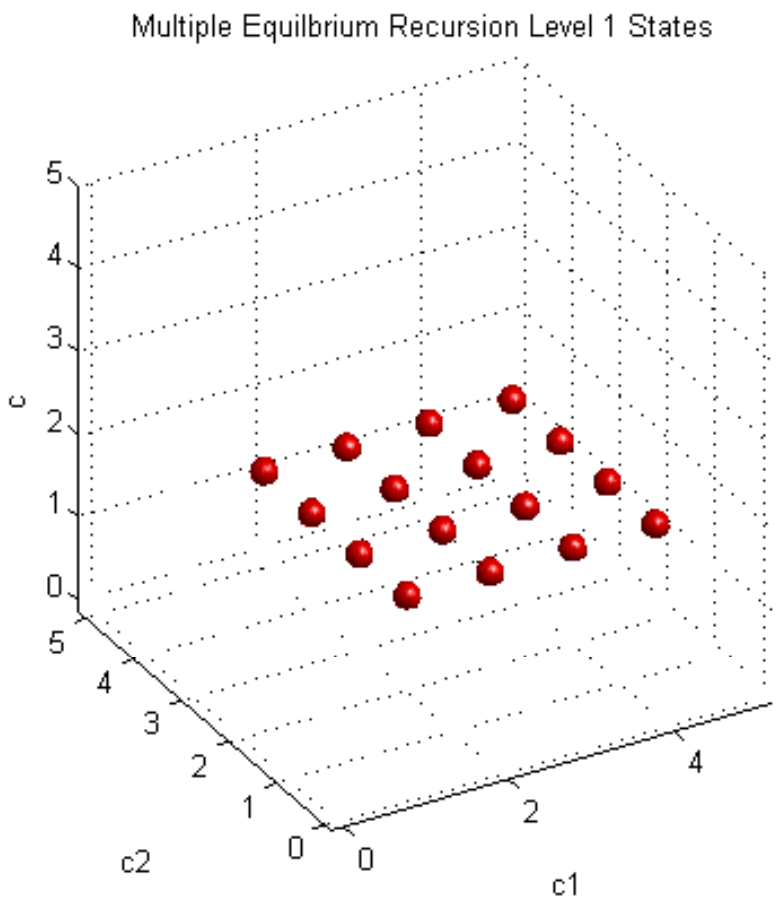
Recursion Level 1 States, isolated



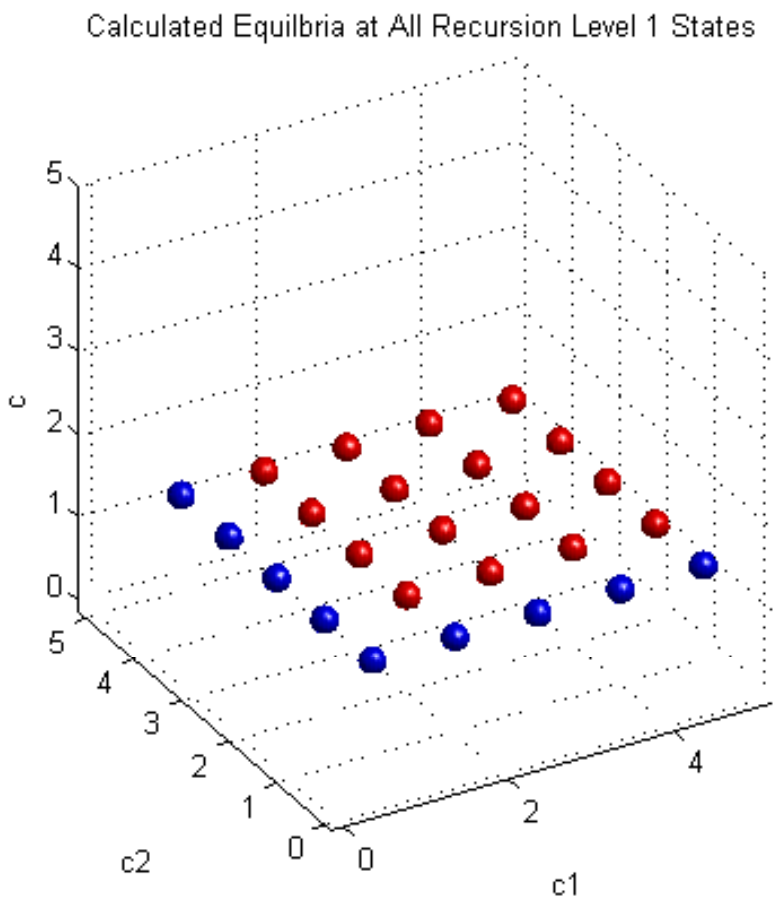
No Investment Recursion Level 1 States



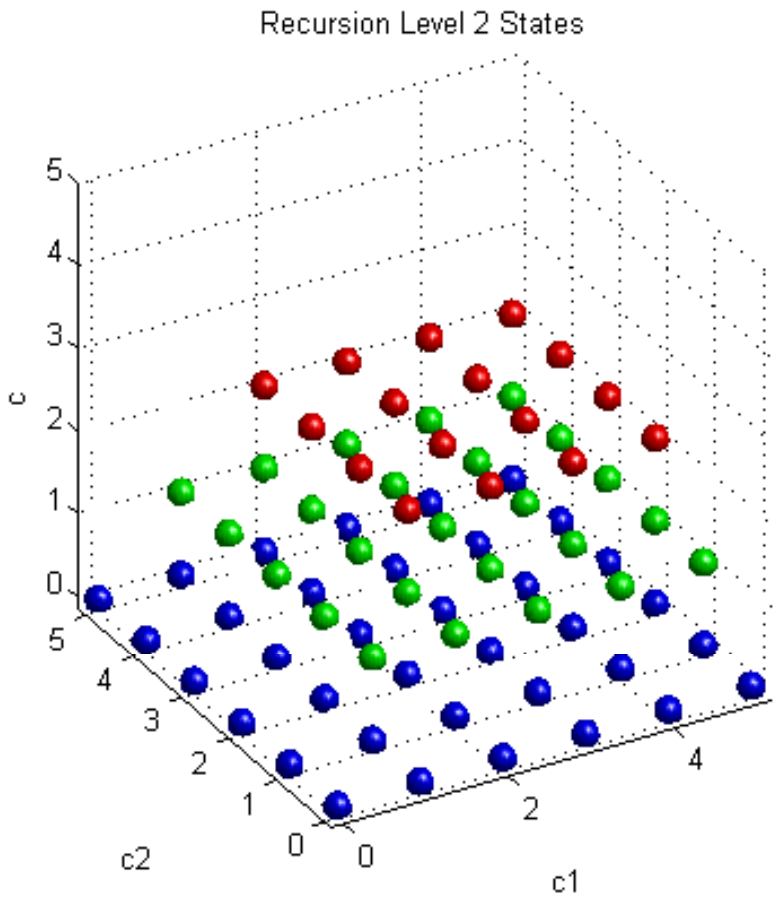
Multiple Equilibria Recursion Level 1 States



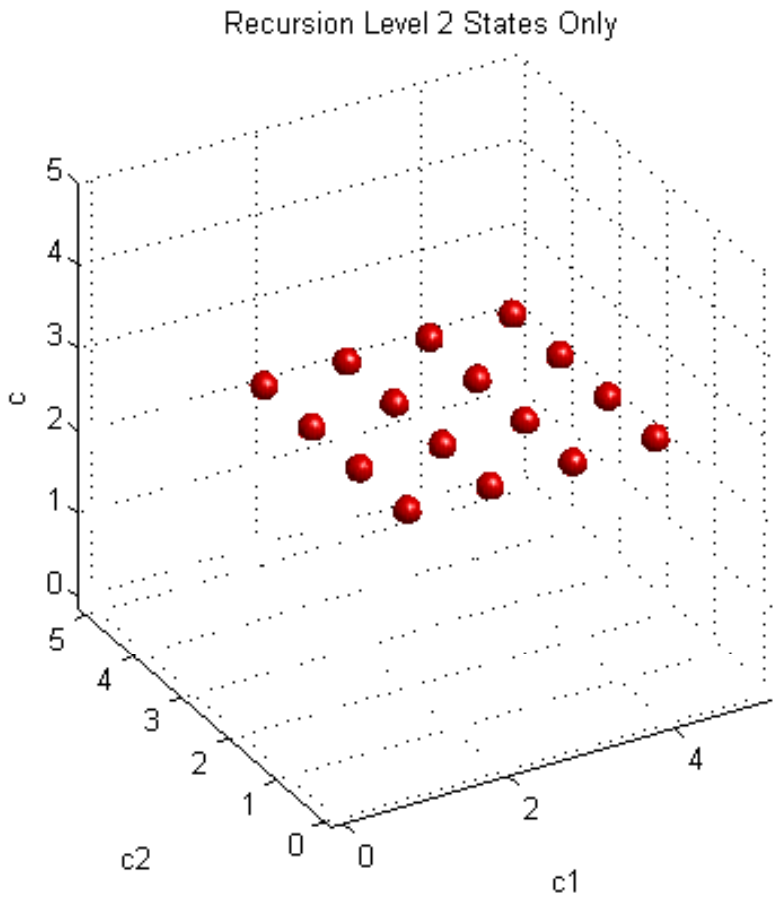
Calculated Equilibria Recursion Level 1 States



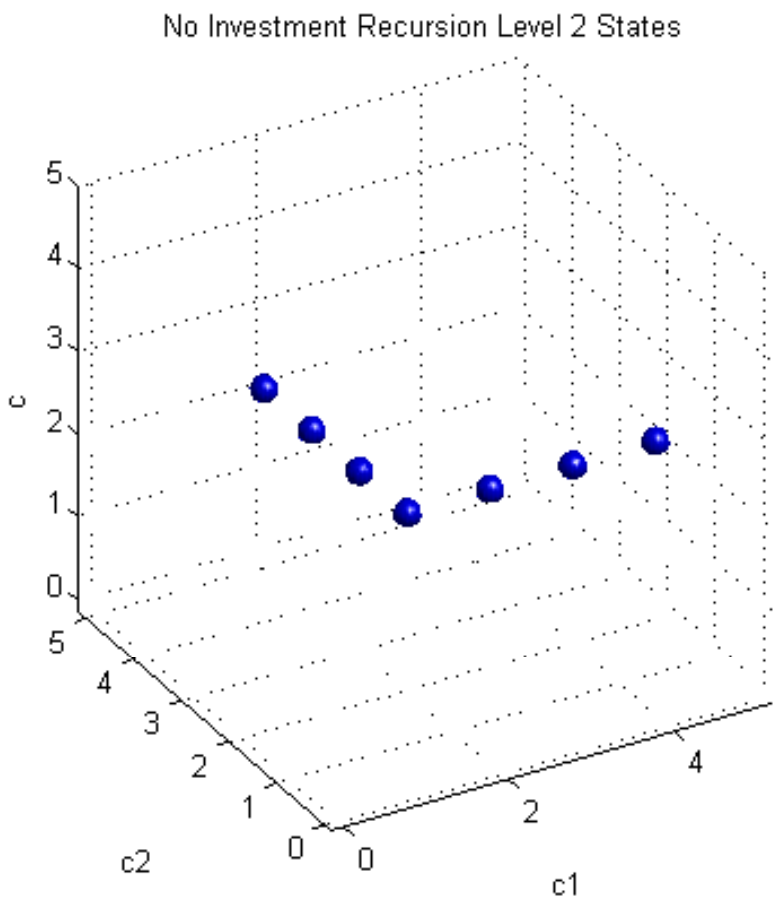
Recursion Level 2 States



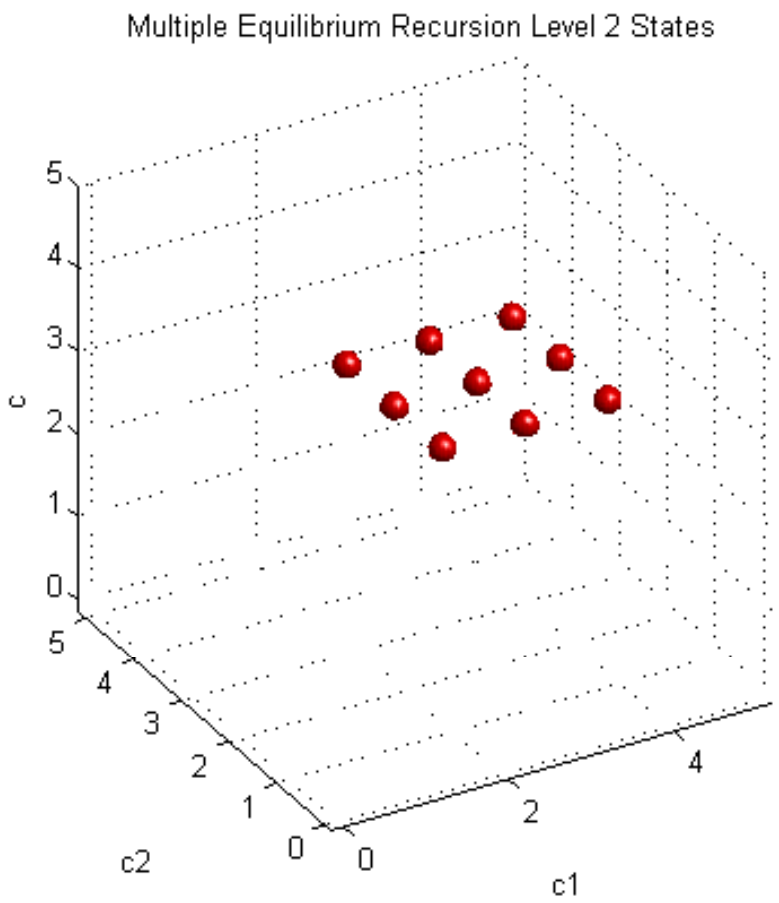
Recursion Level 2 States, isolated



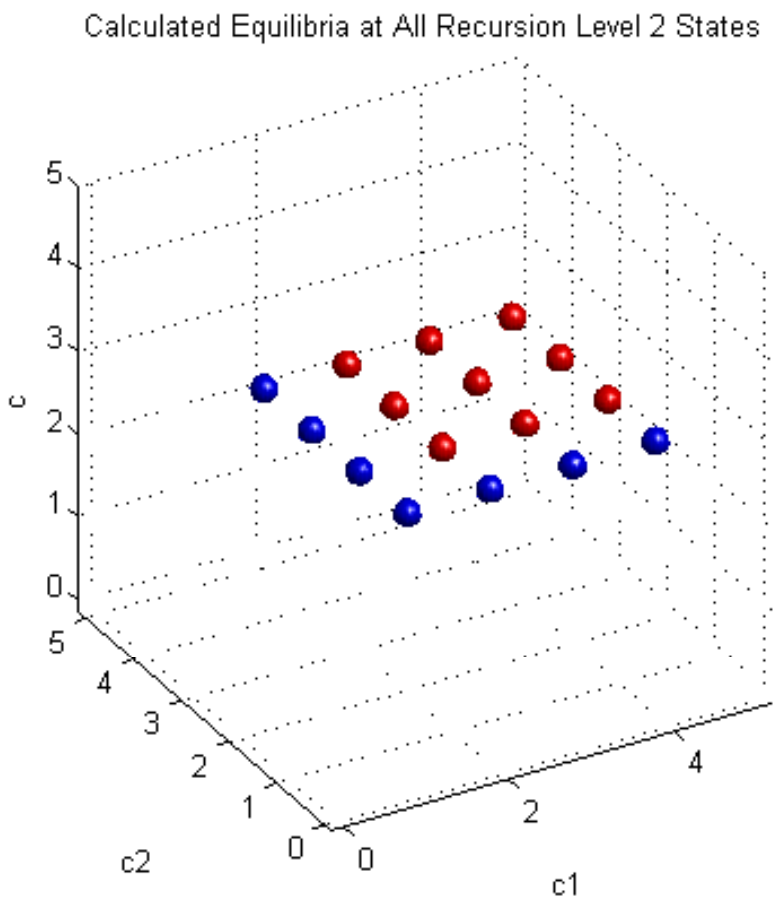
No Investment Recursion Level 2 States



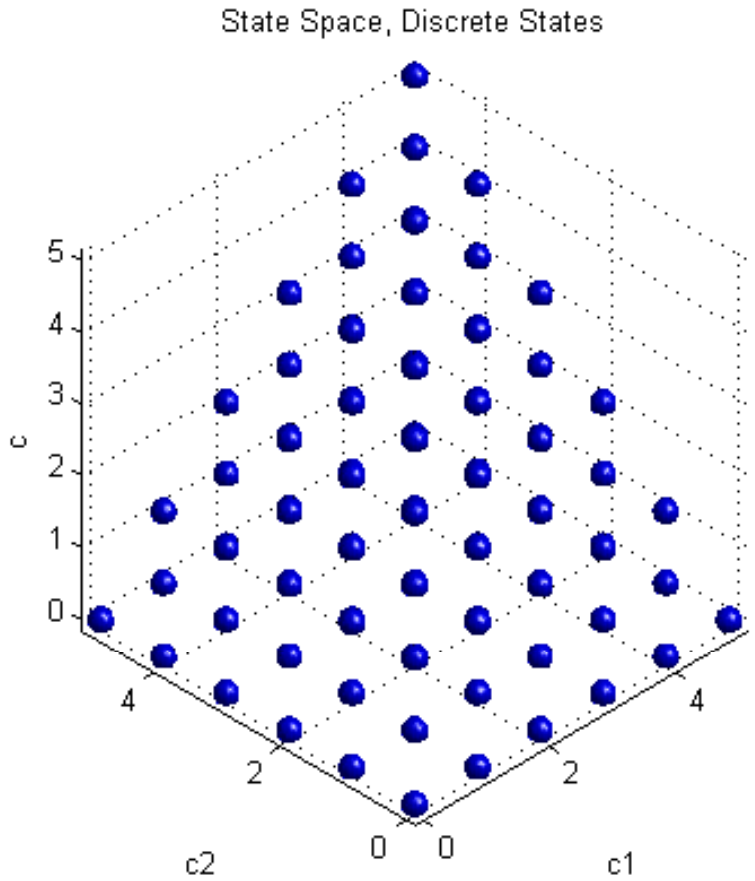
Multiple Equilibria Recursion Level 2 States



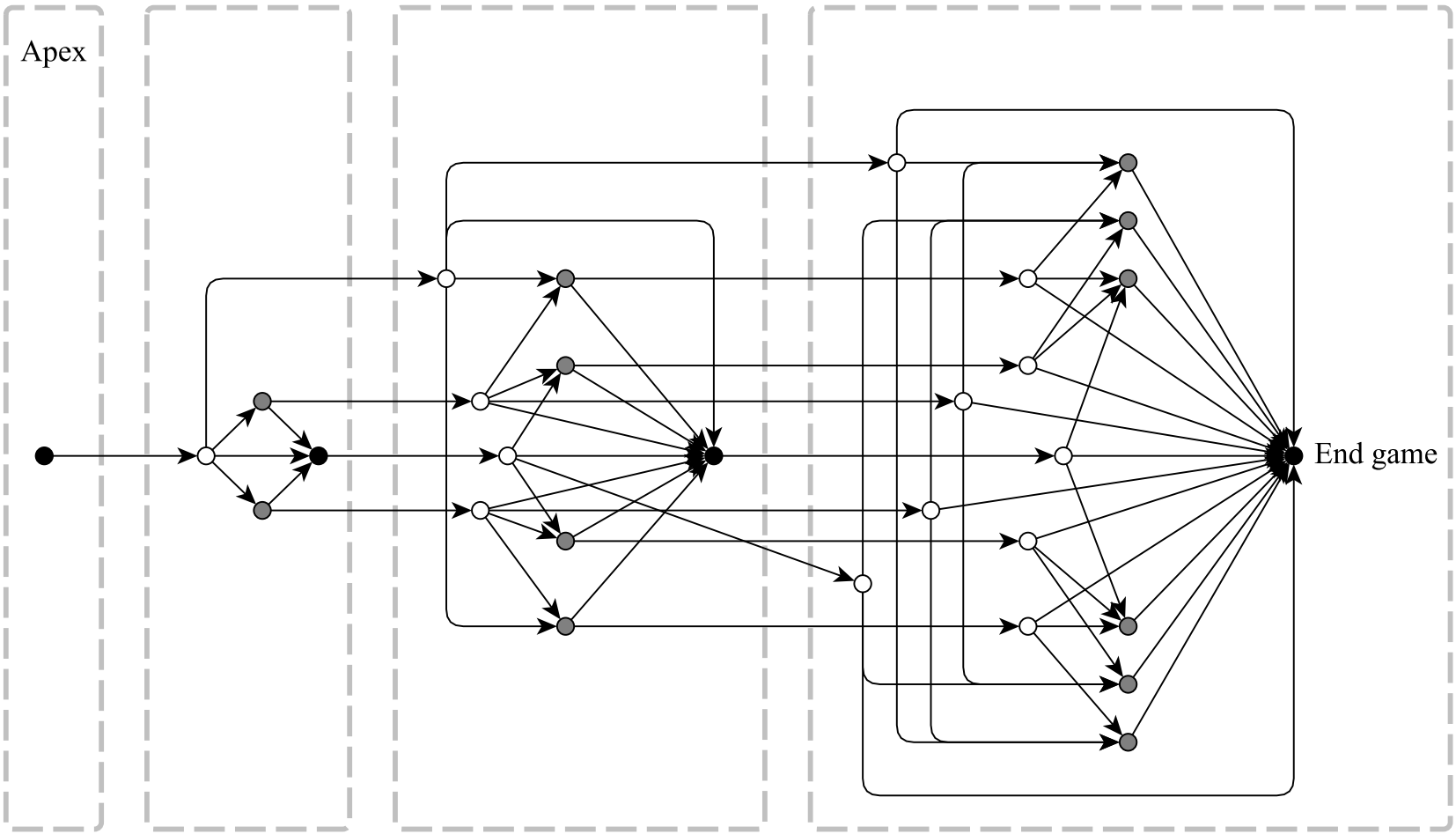
Calculated Equilibria Recursion Level 2 States



Continue recursion to calculate equilibria in all states



Partial ordering of states



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

Properties of RLS algorithm:

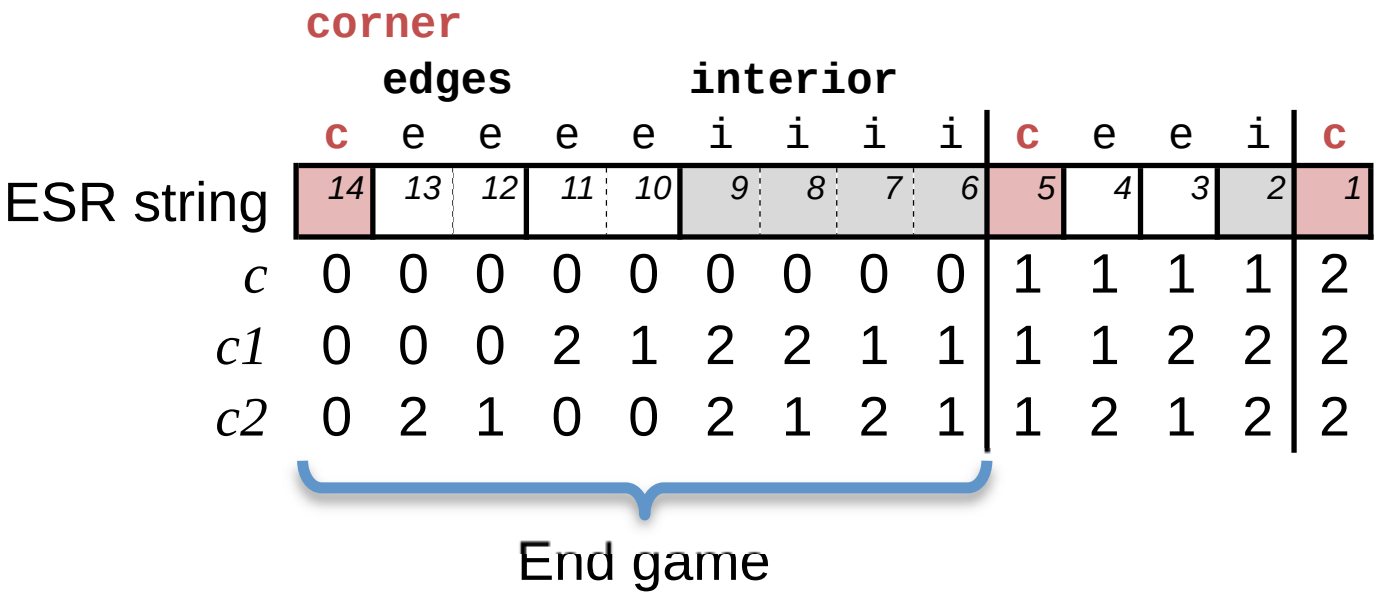
- Complete: Computes all MPE equilibria of the game
- Fast: time spent of search of feasible ESRs is negligible in comparison to time spent on solving the game
  - Efficiently skip infeasible ESRs
  - Re-use results of previously computed subgames

Represent ESR as *equilibrium string* of digits

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

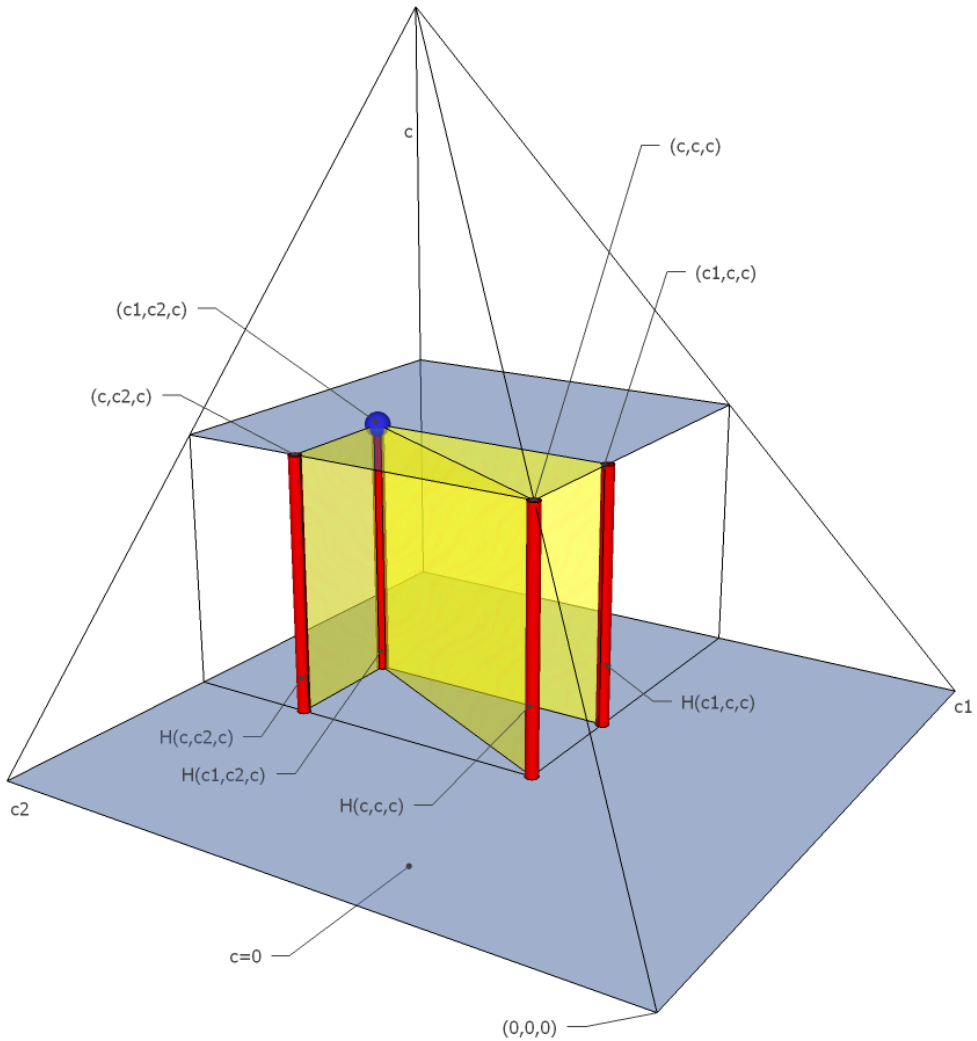
Dependence preserving property:

Any point of the state space may depend on points to the left and not the points to the right



State space of the game: a "quarter pyramid"

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



Particular ESRs examples

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*
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

End game

Examples:															
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	First equilibrium always
		0	0	2	2		2	0			0	2			High cost to invest
		2	2	0	0		0	2			2	0			Low cost to invest
	1					1			1	1			1	1	Mixed when equal

All possible ESRs

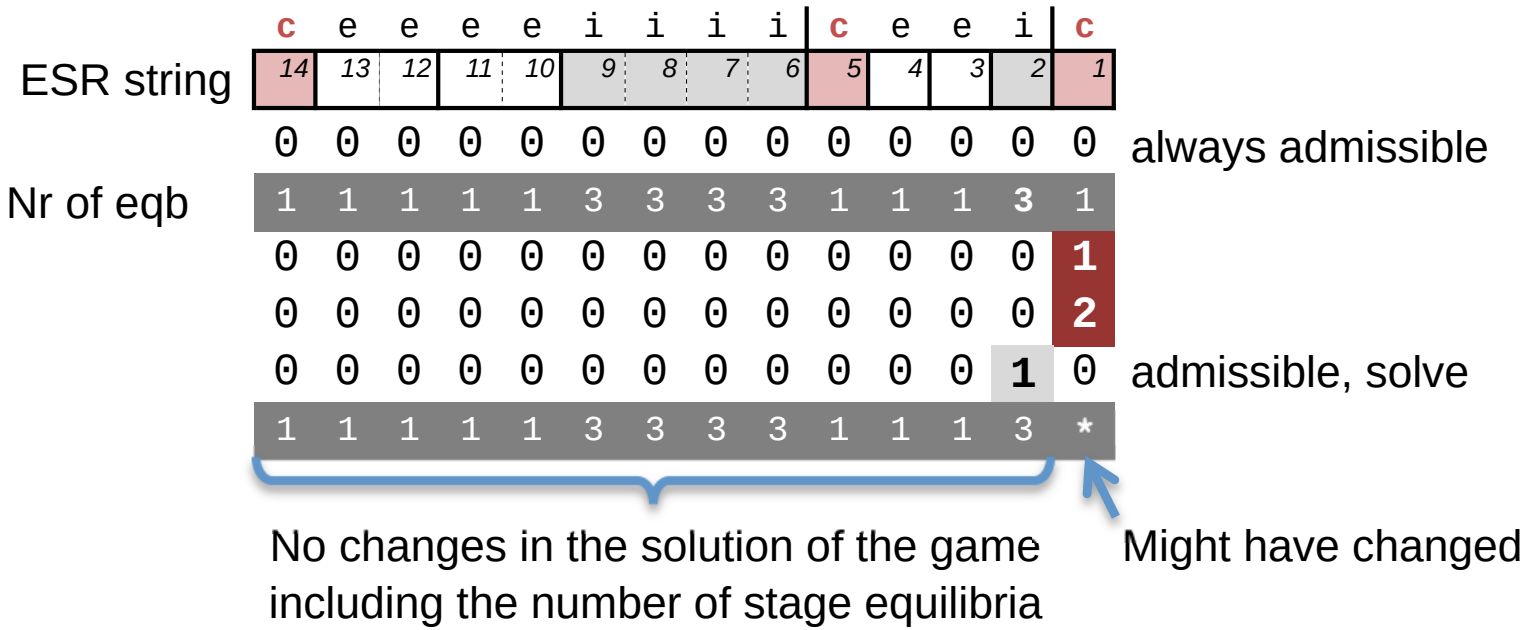
Lexicographic order

	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

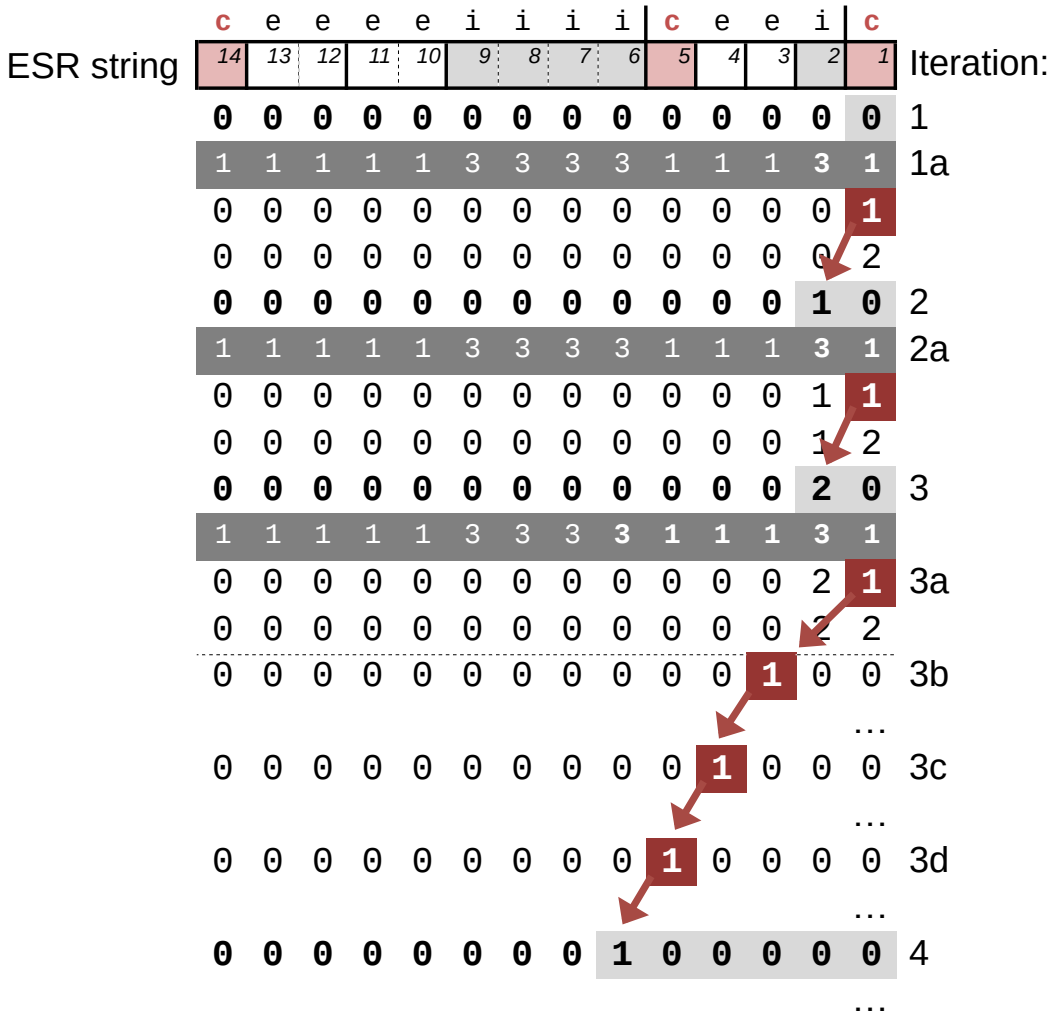
Recalculation of fesibility condition for new ESR

Avoid recalculation of subgames



Jumping over blocks of infeasibles ESRs

Using block structure of lexicographic ordering



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 for the rest of Talk

Results and Simulations

1. Resolution to the Bertrand investment paradox
2. Sufficient conditions for uniqueness of equilibria
3. Characterization of the set of equilibrium payoffs
4. Efficiency of equilibria
5. Leap-frogging or preemption and rent-dissipation
6. Danger of imposing symmetry
7. Limitations of homotopy parameter methods

Resolutions to Bertrand Investment Paradox

Earlier work:

- Fudenberg et al. (1983 RIE), Reinganum (1985 QJE), Fudenberg and Tirole (1985 ReStud), ...
- Riordan and Salant (1994 JIE): Preemption and rent dissipation (unique equilibrium)

We show:

1. Many types of endog. coordination is possible in equilibrium
   - Leapfrogging (alternating investments)
   - Preemption (investment by cost leader)
   - Duplicative (simultaneous investments)
2. The equilibria are generally inefficient due to over-investment
   - Duplicative or excessively frequent investments

Resolution to the Bertrand investment paradox

Theorem (Solution to Bertrand investment paradox)

If investment is socially optimal at a state point $(c_1, c_2, c) \in S$, *then*

- *no investment by both firms cannot be an MPE outcome in the subgame starting from* (c_1, c_2, c) *in either the simultaneous or alternating move versions of the dynamic game.*

Multiplicity of equilibria

Theorem (Sufficient conditions for uniqueness)

In the dynamic Bertrand investment and pricing game a sufficient condition for the MPE to be unique is that

1. *firms move in alternating fashion (i.e. $m \neq 0$), and,*
2. *for each $c > 0$ in the support of π we have $\pi(c|c) = 0$.*

1. Corollary: If firms move simultaneously, equilibrium is generally *not unique*.
2. Corollary: If technological change is stochastic, equilibrium is generally *not unique*.

Multiplicity of equilibria

Theorem (Number of equilibria in simultaneous move game)

If investment is socially optimal, and the support of the Markov process $\{c_t\}$ for the state of the art marginal costs is the full interval $[0, c_0]$ (i.e. continuous state version),

- *the simultaneous move Bertrand investment and pricing game has a continuum of MPE.*

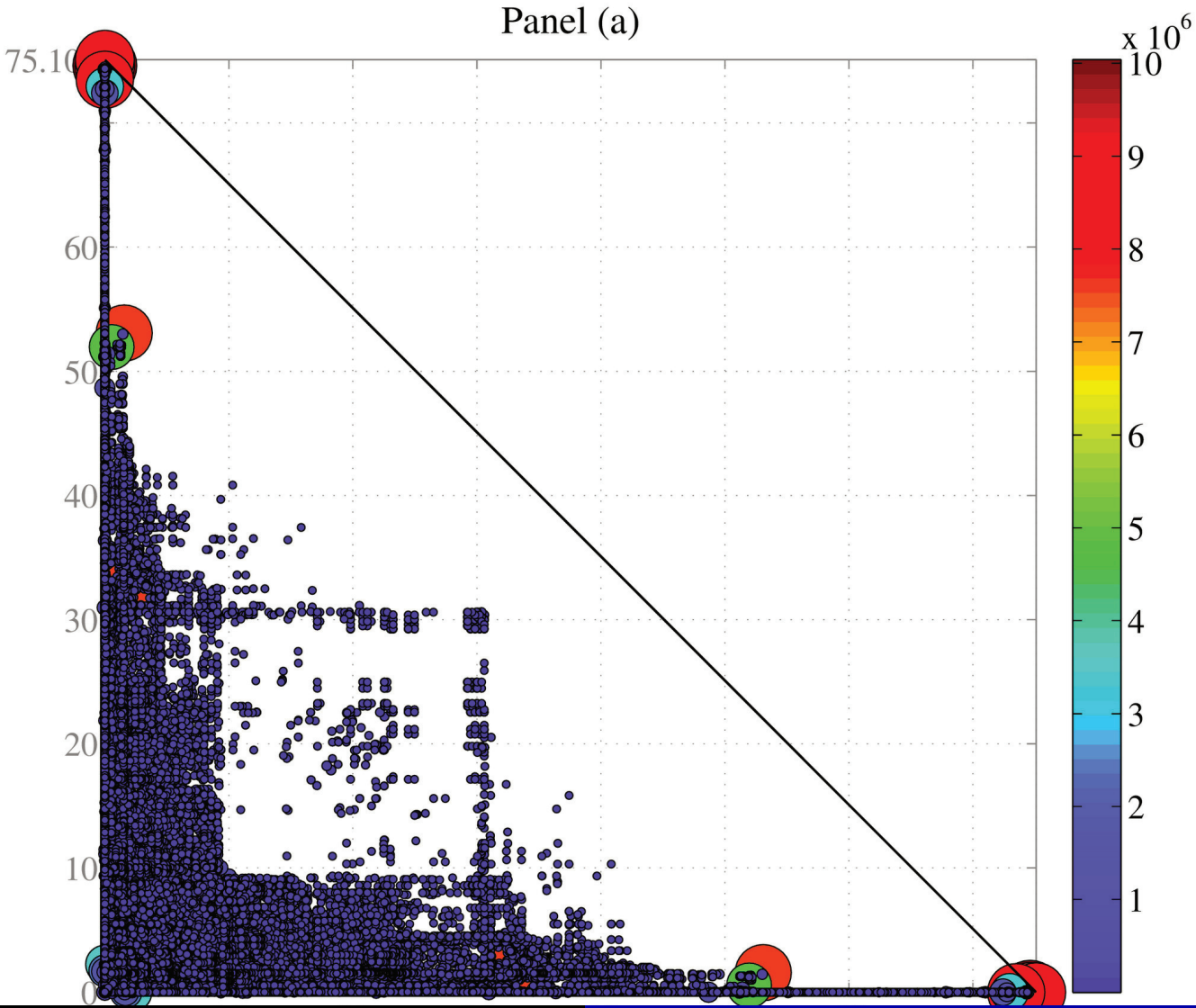
Pay-offs in the simultaneous move game

Theorem (Triangular payoffs in the simultaneous move game)

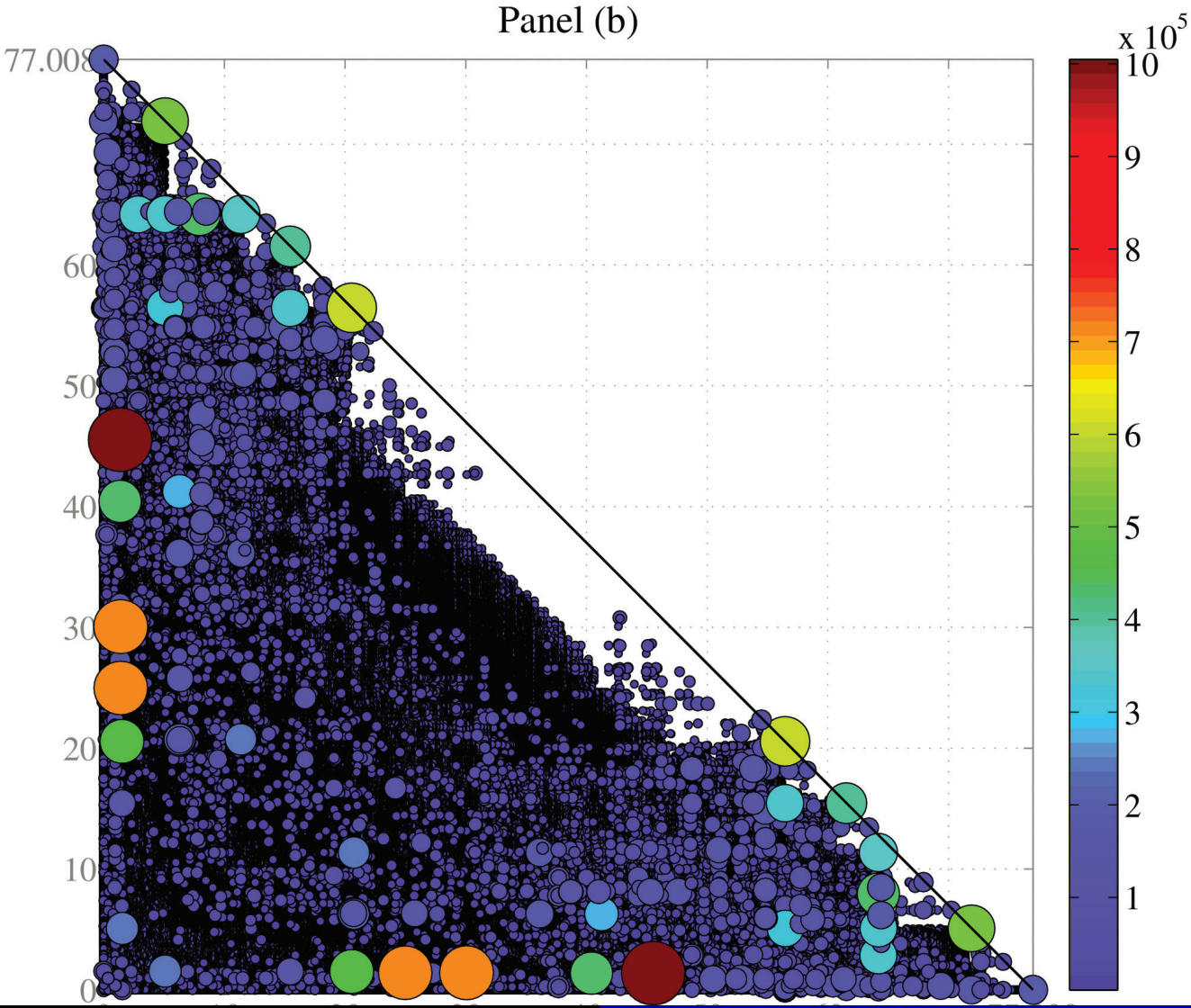
Suppose that the $\{c_t\}$ process has finite support, that there are no idiosyncratic shocks to investment (i.e. $\eta = 0$) and that firms move simultaneously

- *The (convex hull of the) set of the expected discounted equilibrium payoffs at the apex state $(c_0, c_0, c_0) \in S$ is a triangle*
- *The vertices of this triangle are at the points $(0, 0)$, $(0, V_M)$ and $(V_M, 0)$ where $V_M = v_{N,i}(c_0, c_0, c_0)$ is the expected discounted payoff to firm i in the monopoly equilibrium where firm i is the monopolist investor.*

Pay-offs (deterministic tech progress, simultaneous moves)



Pay-offs (stochastic tech progress, simultaneous moves)

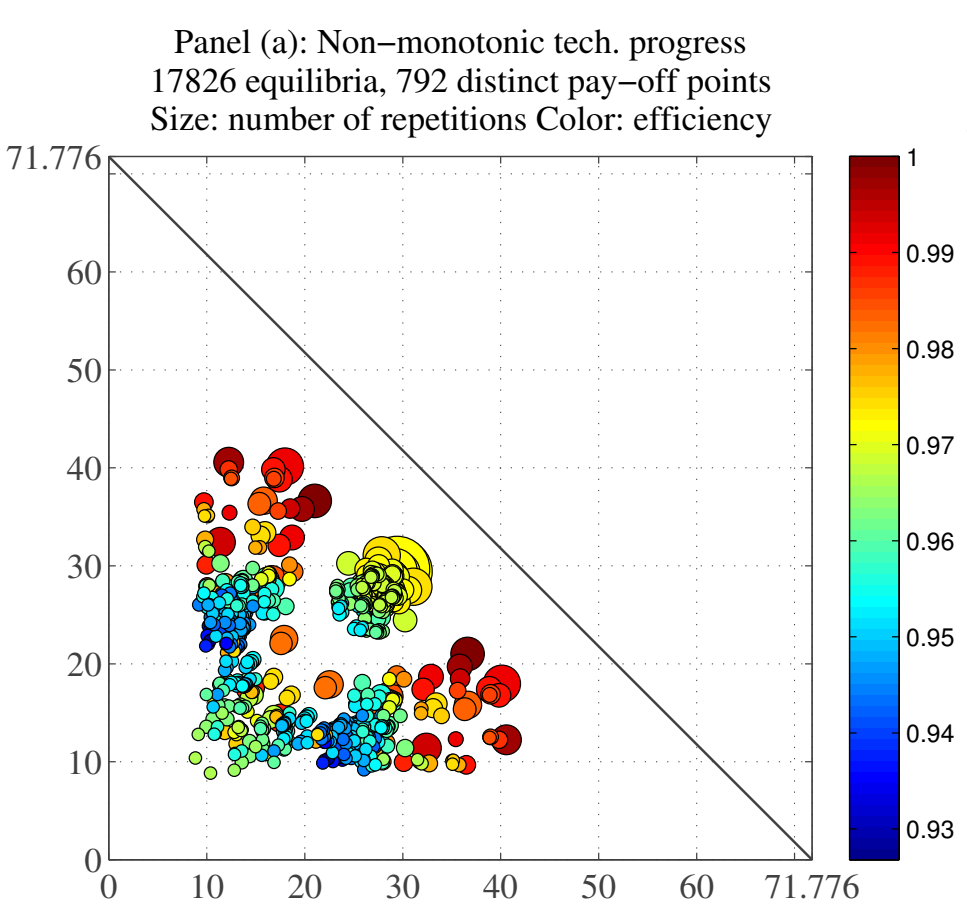


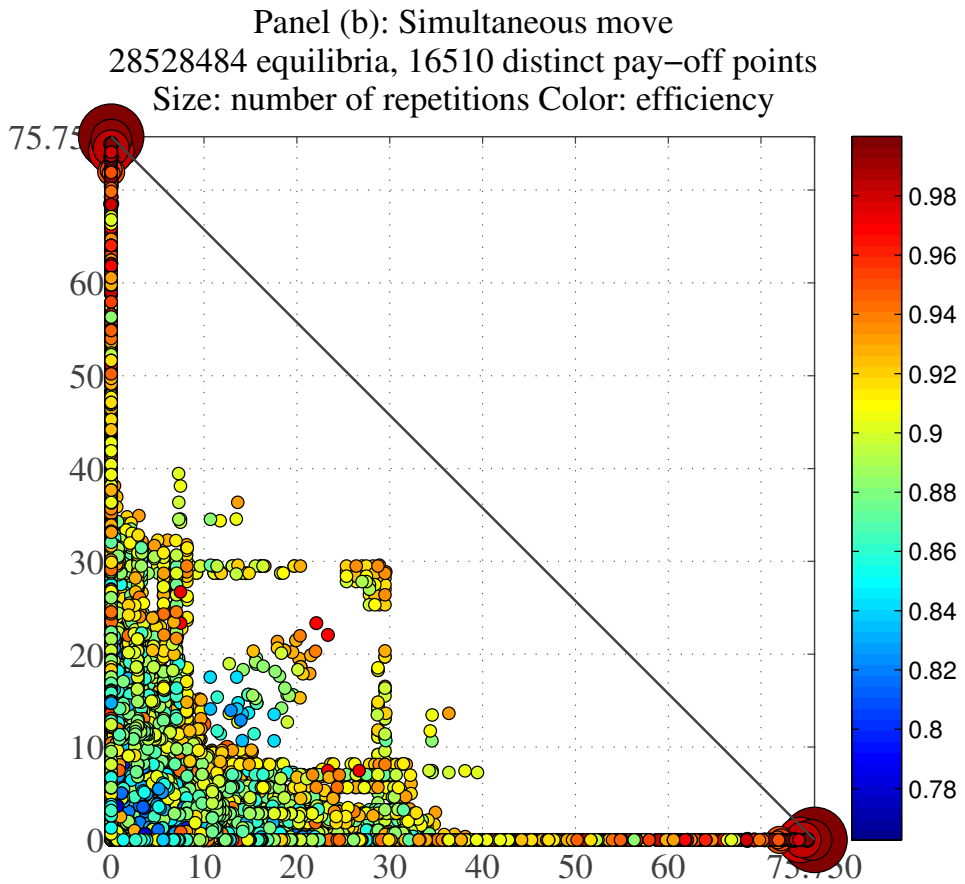
Pay-offs in the alternating move game

Theorem (Equilibrium payoffs in the alternating move game)

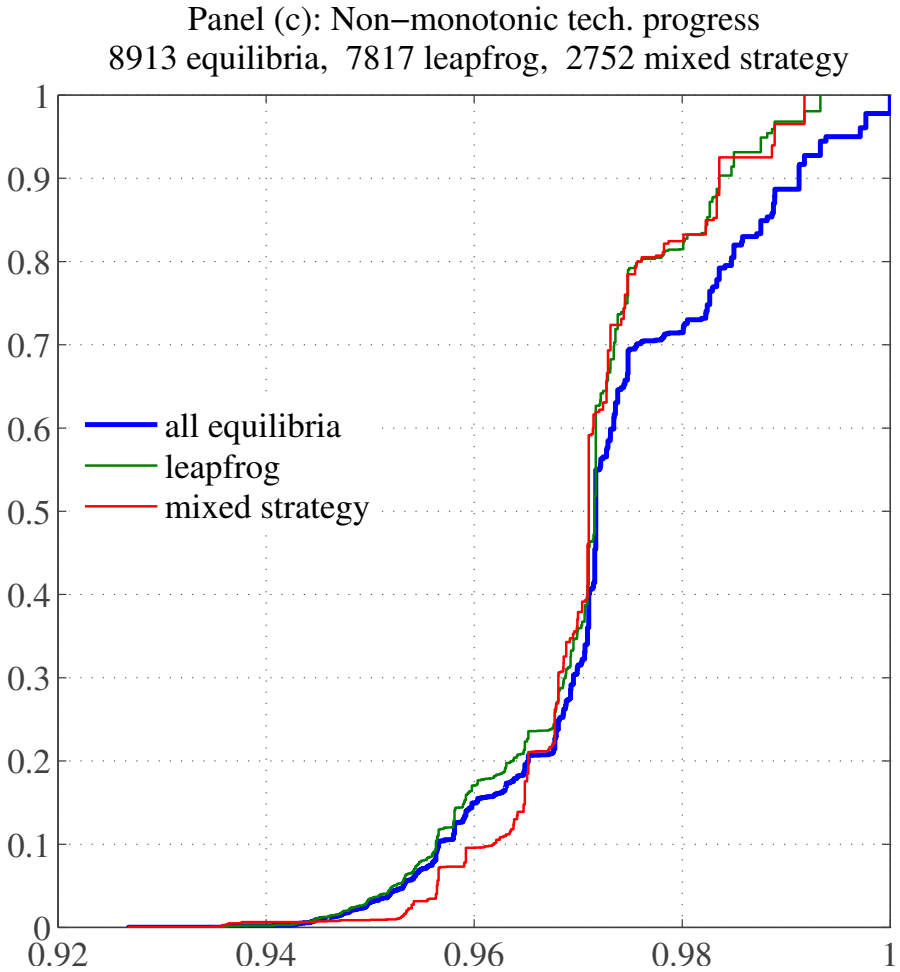
The (convex hull of the) set of expected discounted equilibrium payoffs at the apex state $(c_0, c_0, c_0) \in S$ *of the alternating game is a strict subset of the triangle with the vertices* $(0, 0)$, $(0, V_M)$ *and* $(V_M, 0)$

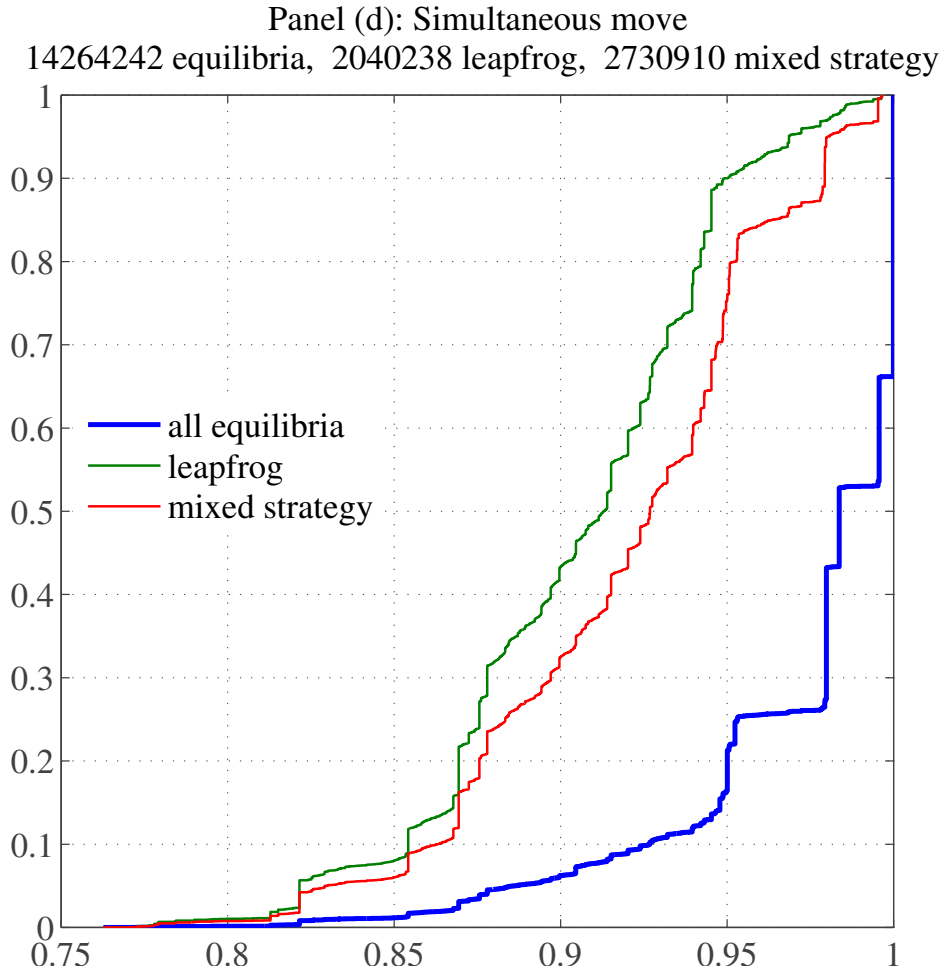
Pay-offs: alternating vs simultaneous move games





Efficiency: alternating vs simultaneous move games





Efficiency of equilibria

Simultaneous move game

Theorem (Inefficiency of mixed strategy equilibria)

A necessary condition for efficiency in the dynamic Bertrand investment and pricing game is that along MPE path only pure strategy stage equilibria are played.

Riordan and Salant: Full Preemption

Theorem (**Riordan and Salant, 1994**)

The continuous time investment game where

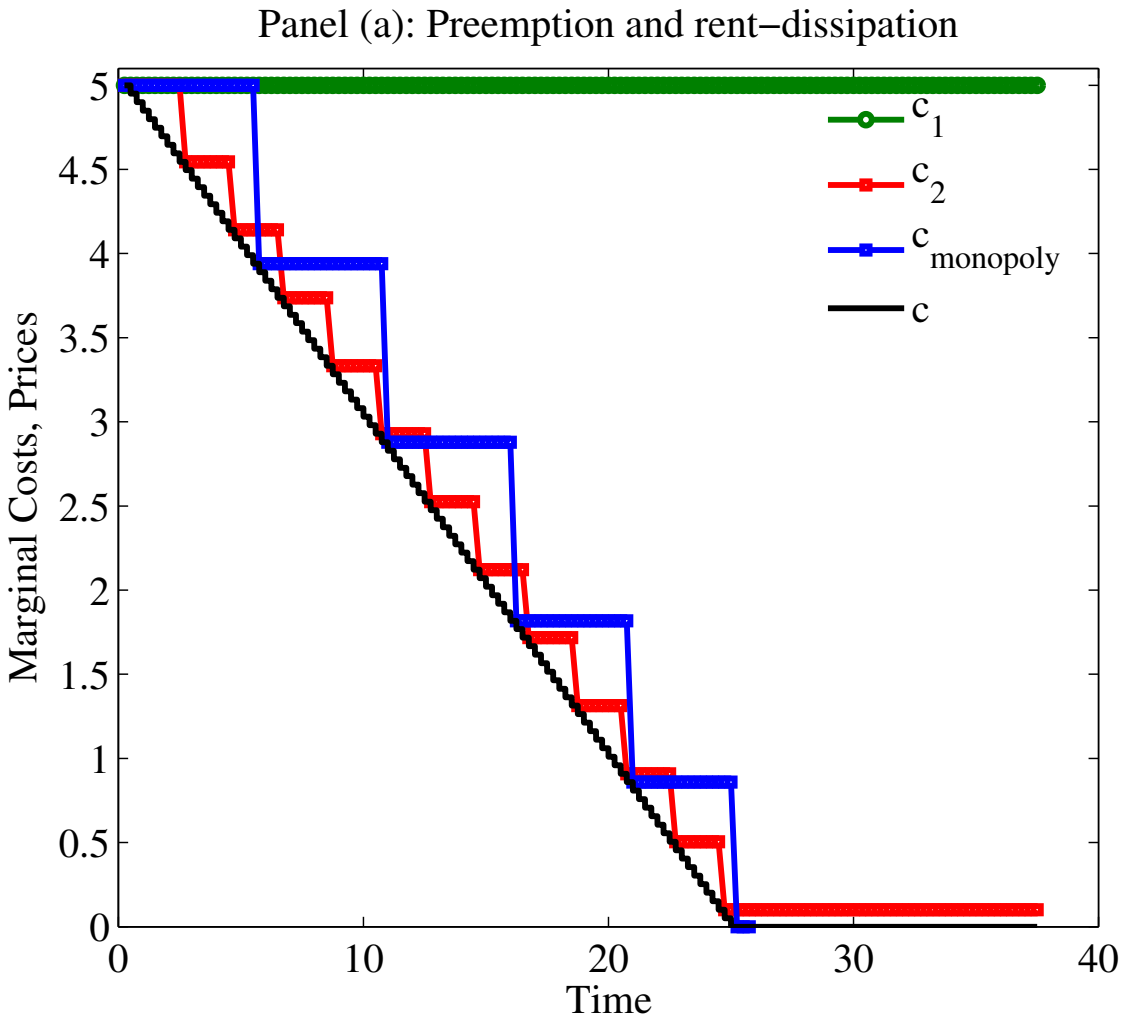
1. *right to move alternates deterministically.*
2. $K(c) = K$ *and is not prohibitively high.*
3. *technological progress is deterministic:* $c(t)$ *is a continuous, decreasing function*

has a unique MPE with

- *preemptive investments: by only one firm and no investment in equilibrium by its opponent.*
- *rent dissipation: discounted payoffs of both firms in equilibrium is* 0*, so the entire surplus is wasted on excessively frequent investments by the preempting firm.*

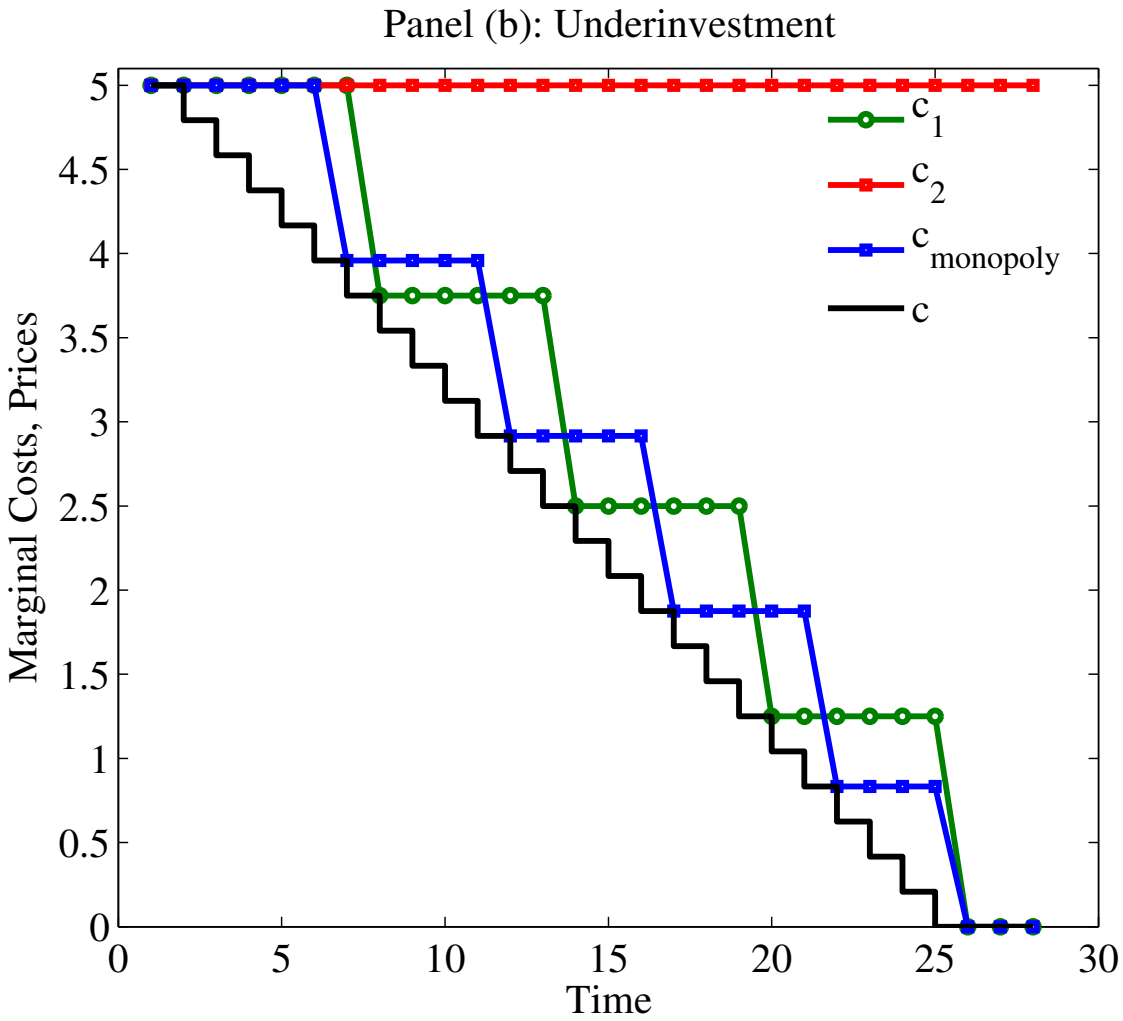
Riordan and Salant: Full preemption and rent dissipation

Confirm the result with high K and small dt



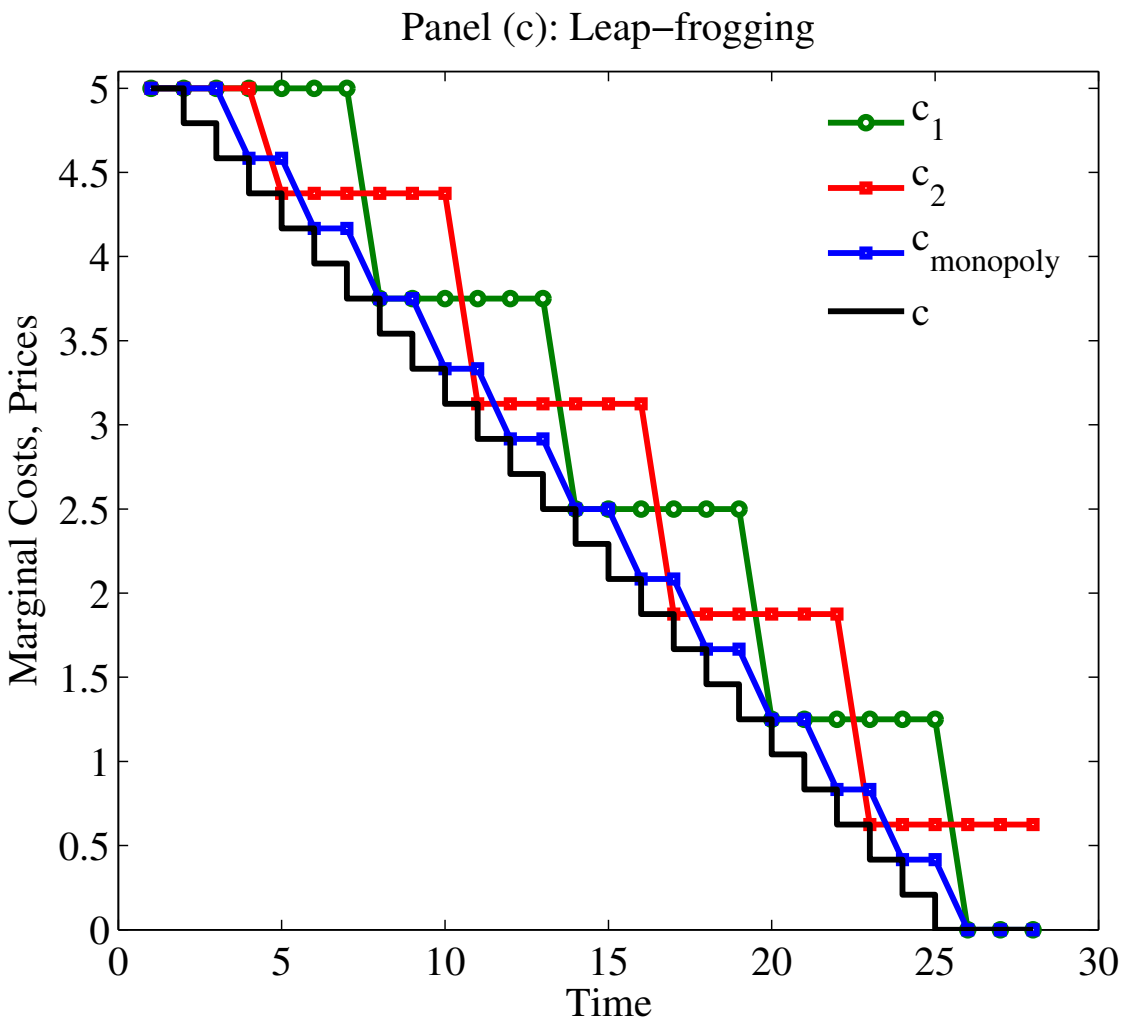
Underinvestment

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



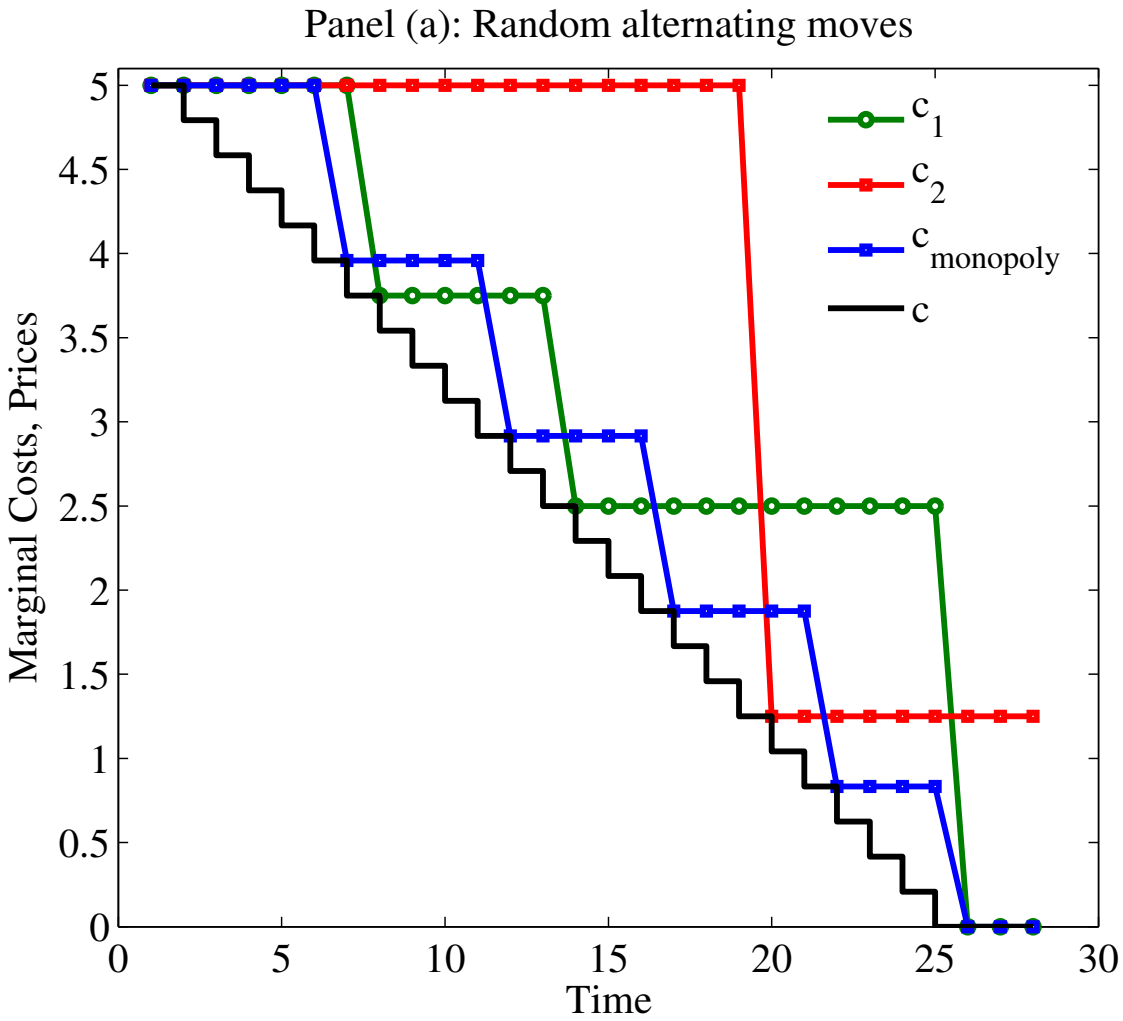
Leap-frogging

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



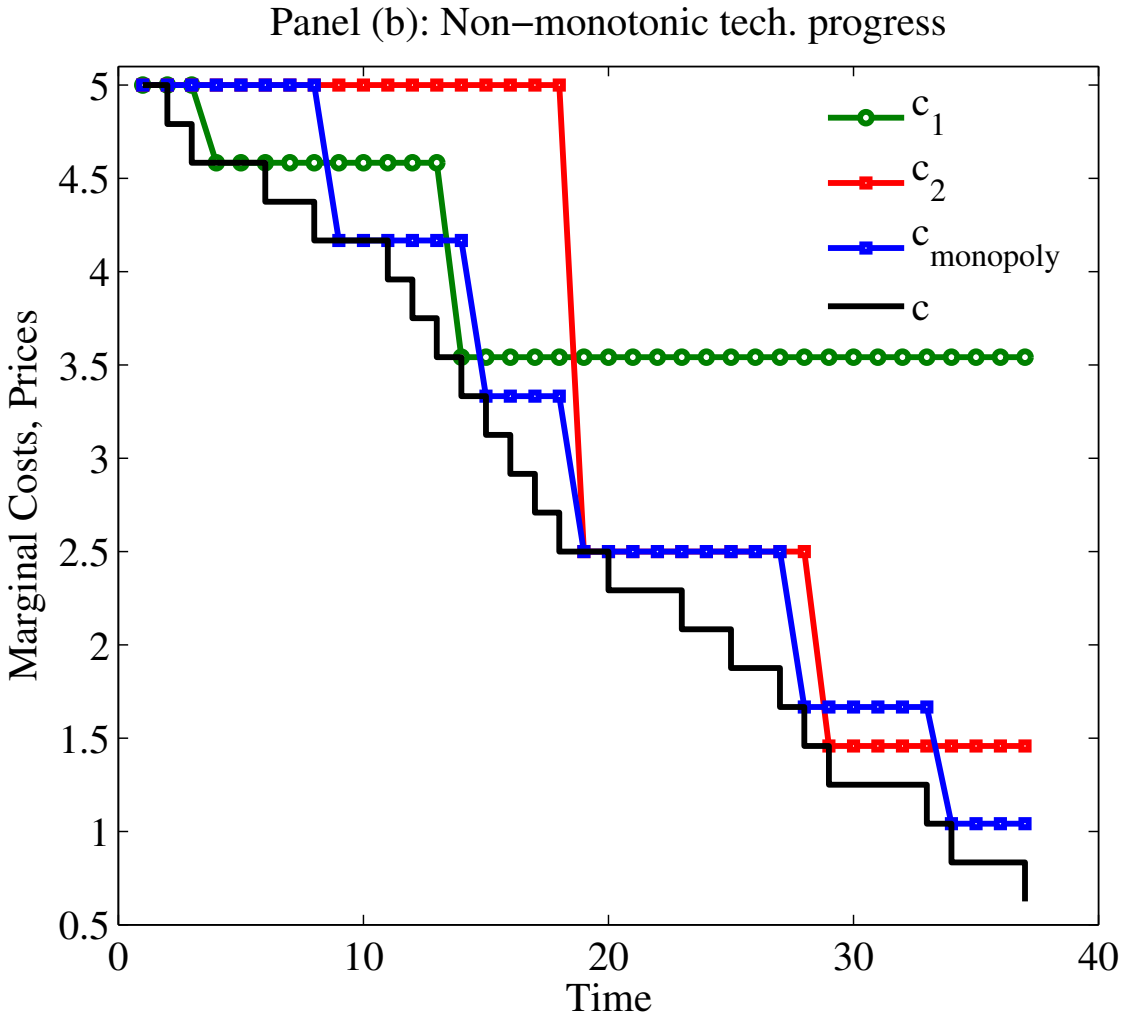
Random alternation $\rightarrow$ Leapfrogging

Riordan and Salant's result is not robust



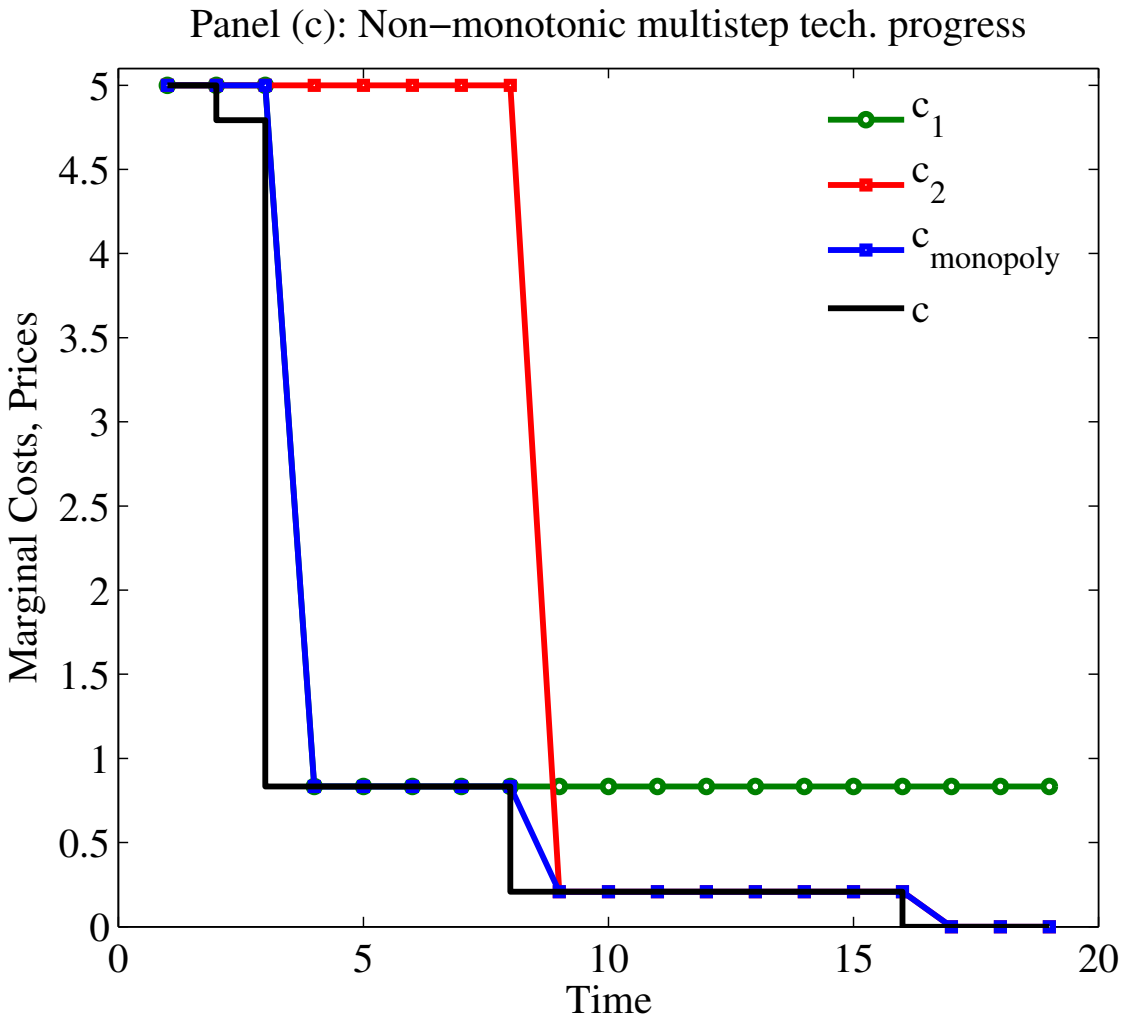
Random onestep technology → Leapfrogging

Riordan and Salant's result is not robust



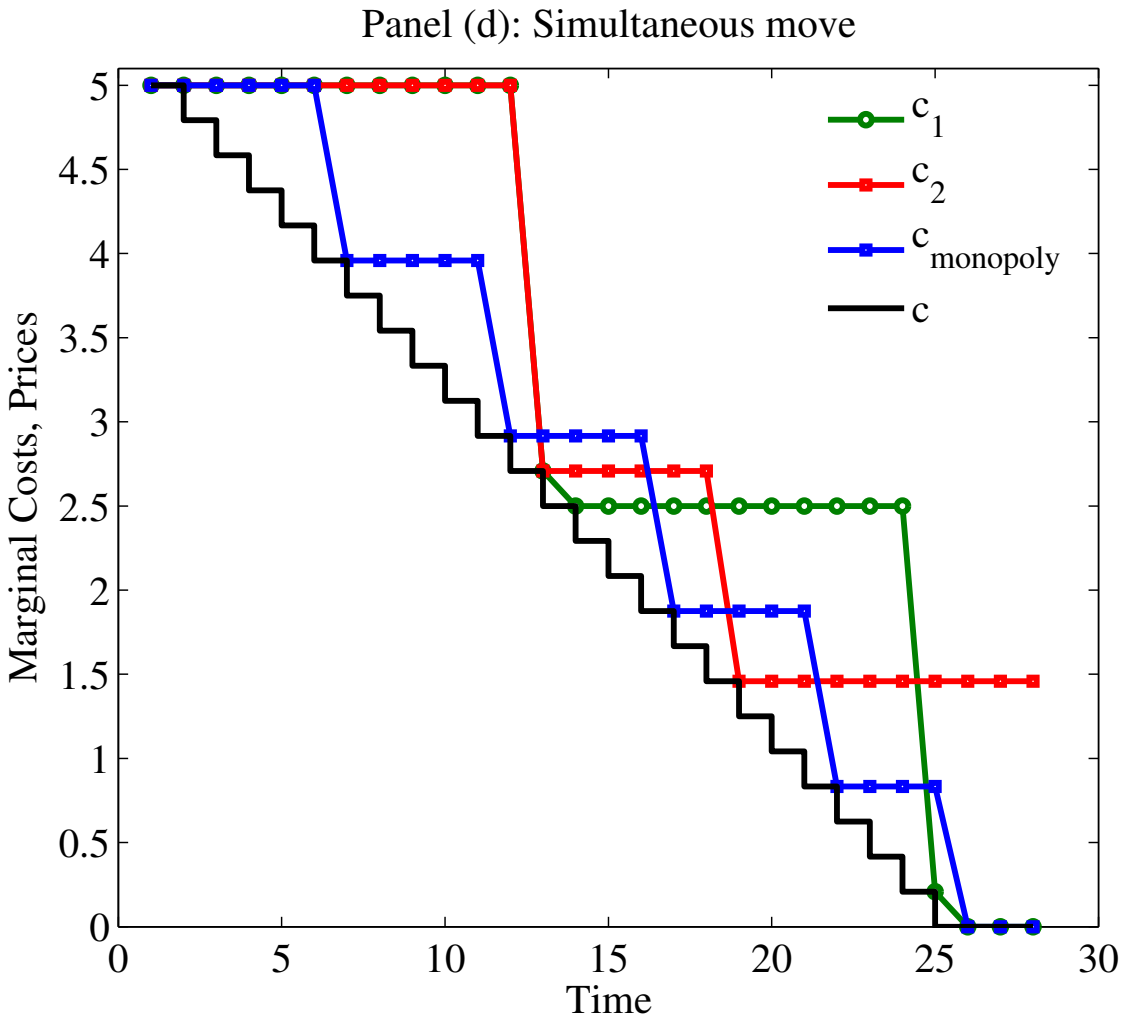
Random multistep technology $\rightarrow$ Leapfrogging

Riordan and Salant's result is not robust

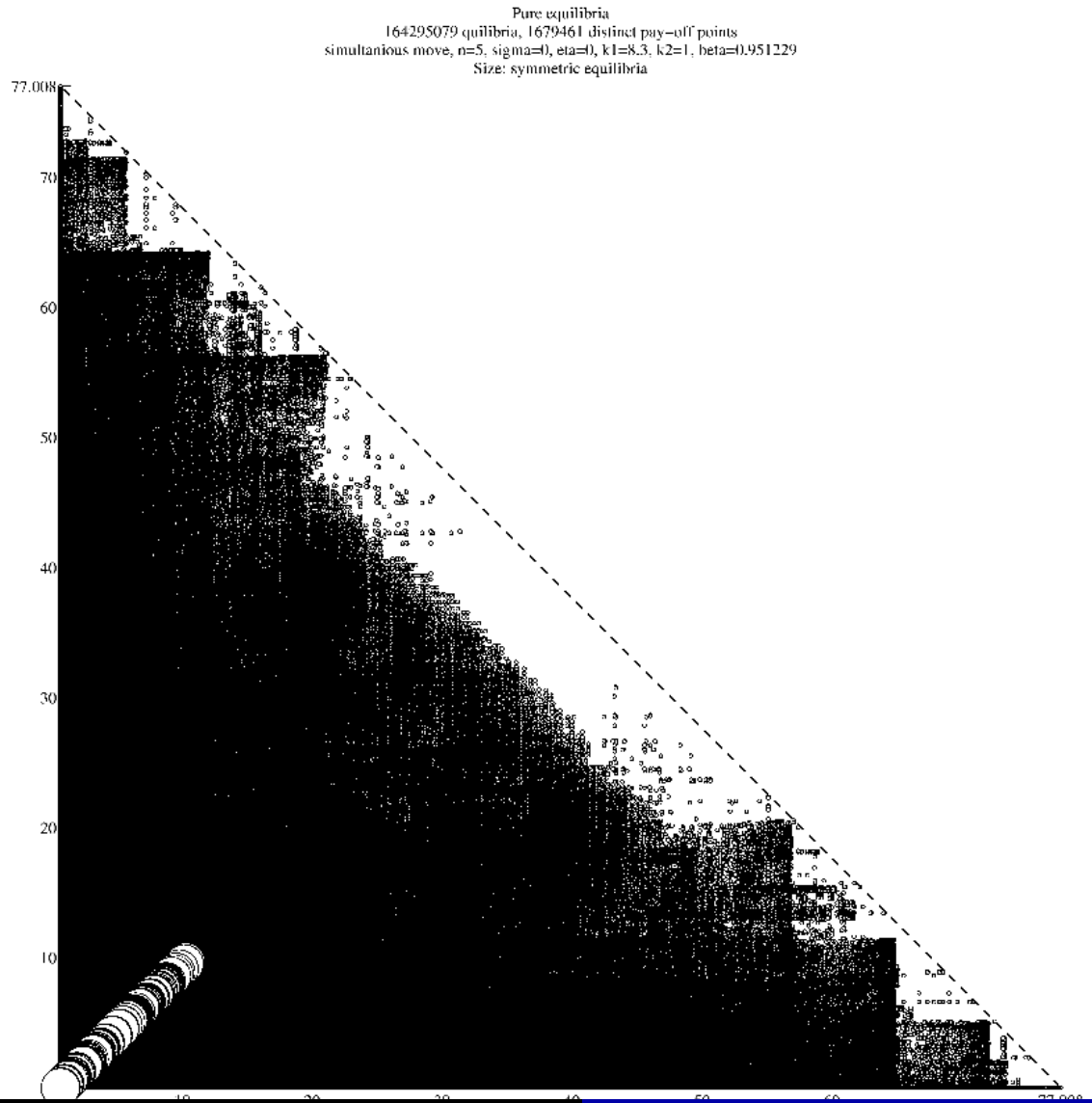


Simultaneous moves: Leapfrogging

Riordan and Salant's conjecture is wrong



Symmetric equilibria: $V_1(c_1, c_2, c) = V_2(c_2, c_1, c)$



Limitations of homotopy approach

Homotopy parameter: η

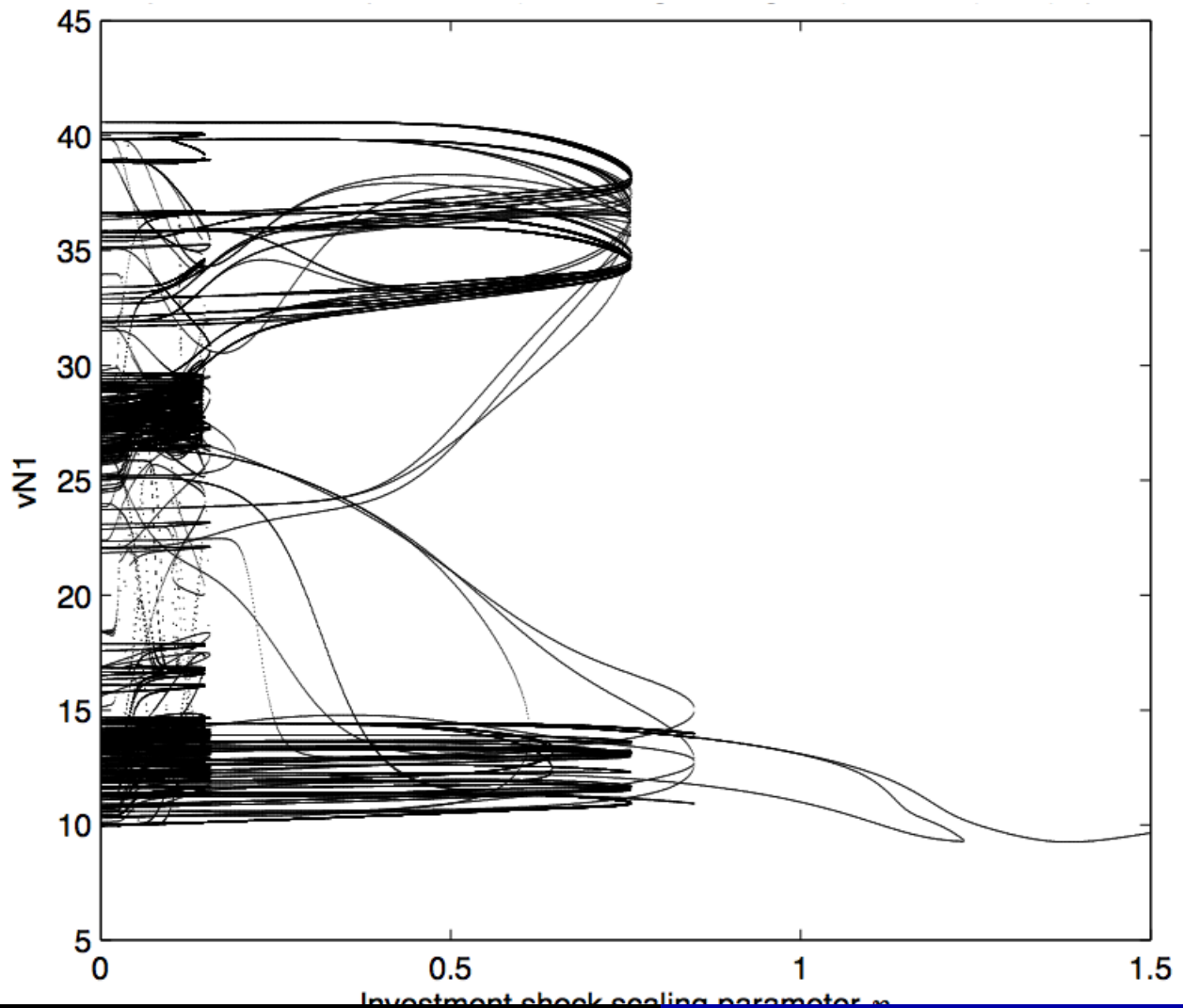
- In each period each firm incurs additive random costs/benefit from not investing and investing
- η is a scaling parameter that index variance of idiosyncratic shocks to investment
- High $\eta \rightarrow$ unique equilibrium $\eta \rightarrow 0 \rightarrow$ multiple equilibria

Problems:

- Multiplicity of equilibria $\rightarrow$ too many bifurcations along the path
- Equilibrium correspondence is not lower hemi-continuous

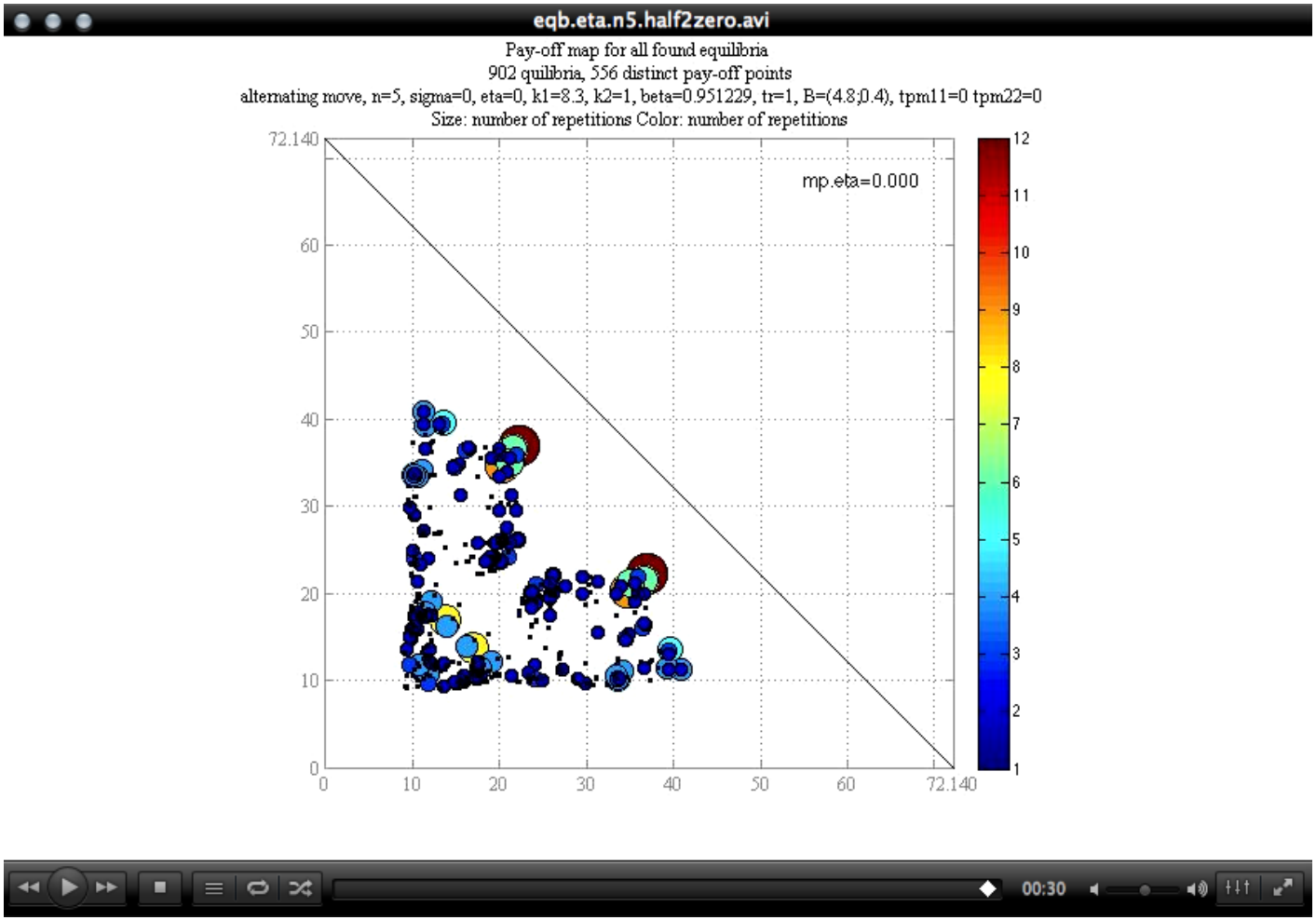
Limits of the homotopy approach

Equilibrium correspondance, alternating move game: $V_{N,1}(c_0, c_0, c_0)$ vs. η



Limits of the homotopy approach

Video: Set of equilibrium outcomes as variance of shocks decreases to zero



Theoretical conclusions

- Endogenous coordination (e.g. leapfrogging) is possible in equilibrium of dynamic Bertrand investment game
- Leapfrogging gives new interpretation of the price wars
- Numerous MPE equilibria and "Folk theorem"-like result
- Most equilibria are inefficient due to over-investment

Methodological conclusions

- When equilibrium is not unique the computation algorithm inadvertently acts as an *equilibrium selection mechanism*
- State recursion algorithm is preferred to time iterations
- Imposing symmetry restriction on equilibria knocks out most equilibria in the model
- Plethora of Markov perfect equilibria leads to new paradox: How can firms, without any explicit communication, coordinate on a single equilibrium in these games when there is generally such a vast multiplicity of possible equilibria.

Econometrica referee reports

- We submitted the paper to *Econometrica* and perhaps not surprisingly it was rejected
- Two of the referees were obviously theorists who dimissed the theoretical contribution but said they could not evaluate the computational contribution
- Referee 1 appeared to be familiar with both theory and computation
- First line in referee 1's report ... *What's wrong?*

R1's main points

- "The problem with the paper is that it is generally not possible to find all solutions to a system of nonlinear equations"
- "The only exception that I am aware of is if the system happens to be polynomial, so that so-called all-solutions homotopies can be applied (see, e.g., Sommese & Wampler 2005)."
- "In general, however, there is simply no mathematical basis for finding all solutions to a system of nonlinear equations and therefore all MPE in every d-stage game. Hence, the premise of Theorem 5, namely that there exists an algorithm that can find all MPE of every stage game, does not hold."
- "The conclusion of Theorem 5 that the RLS algorithm finds all MPE of a dynamic stochastic game is therefore irrelevant."

R1's main points, continued

- "To say it even more bluntly, the authors are assuming their main result: if you assume that there is a surefire way to find all solutions to a subset of nonlinear equations, then you may as well assume that there is a surefire way to find all solutions to the full set of nonlinear equations and thus a surefire way to compute all MPEs of the dynamic stochastic game."
- "This is not to say that the RLS algorithm cannot find all MPE for *some* dynamic stochastic games. In particular, the game may have sufficient directionality in the sense that the directional component d of the state space is large relative to the remaining component x, so that each stage game is characterized by few equations or equations that have a special structure."

R1's main points, continued

- "In this case it may be possible to derive closed-form expressions for all solutions or somehow otherwise exploit the special structure of the equations to find them all. A case in point is the investment game in Section 4 of the paper in which solving a stage game reduces to solving a second-order polynomial per the derivations on p. 61"
- "Of course, there is no reason to think that in general the stage games of a directional dynamic stochastic game will be characterized by equations that are tractable enough to allow us to obtain all solutions. Again, the math is simply not there for finding all solutions to a system of nonlinear equations, no matter how small or large that system is."

R1's main points, continued

- "What's new? "That said, the paper does add to the existing literature by formally and rigorously defining directionality and relating it to a directed acyclical graph"
- "there is a fair bit of cleverness in how the RLS algorithm enumerates MPEs in an efficient way by reusing previously computed bits and pieces of the overall game."
- "the authors cite Fudenberg & Levine (1983, JET) to claim that the RLS algorithm can find all MPEs of the T-period truncation of the infinite-horizon game and that this approximates the MPEs of the infinite horizon game as T gets large, see p. 3 and the unproven(!) Theorem 6."
- "there is no reason to think that the sequence of MPEs of the finite-horizon games converges as T gets large (though given compactness it naturally has a convergent subsequence); instead, it may cycle."

R1's main points, continued

- "What else" "I don't understand why the authors have divorced the application from the method. In the previous drafts of the paper that I have seen, the two were combined. This made sense because for the investment game that is now used as an example in Section 4 the RLS algorithm actually does find all MPEs, at least if $\eta = 0$."
- "The result was a really cool paper that combined a bunch of interesting economics with a clever algorithm. This paper I would have happily recommended for publication in *Econometrica.* In my view, the authors have erred by spinning off the method and trying to portray it as more generally applicable than it really is."

Comments from leading game theorists

- **Question:** *did game theorists already define the notion of a "directional dynamic game" and know about "recursive lexicographical search" and are we just reinventing the wheel here?*
- **Steven Morris, Princeton University** "Looks very interesting and Im sure no one scooped you. Benoit and Krishna certainly seems relevant. "
- **Eddie Dekel, Northwestern University** "As best as I could tell and know this hasn't been done."
- **Ariel Rubinstein, Tel Aviv University** "no, i dont think u re-invent the wheel . . . teh entire approach sounds intersting"
- **Andrew McLennan, Queensland** " The strategy of breaking down a big problem into smaller problems that are tractable seems like a good modelling apprach in various settings."

Comments from leading game theorists, continued

- **David Levine, Washington University** "I looked at it to try to see if it is new and useful. I think probably so. If I understand correctly the idea is that there are groups of states that are absorbing regardless of how players play. There is some study of games of this sort (for example exit games where there is a choice that ends the games) but as far as I know nothing so systematic as what you have done."
- **Robert Wilson, Stanford University** " I read the introduction, and to my knowledge this is a materially new approach, and I like the insights that come from assuming directionality for some component of the state space. Computing ALL MPE is unusual, but very welcome when it produces a narrow set of equilibria, and the efficiency of your RLS procedure is a contribution too."

A final quote, from Nelson Mandela

- "Do not judge me by my successes, judge me by how many times I got knocked down and got back up again."