Review of Economic Studies (2015) **0**, 1–46 doi:10.1093/restud/rdv046

Recursive Lexicographical Search: Finding All Markov Perfect Equilibria of Finite State Directional Dynamic Games

FEDOR ISKHAKOV
CEPAR, University New South Wales
JOHN RUST
Georgetown University
and
BERTEL SCHJERNING
University of Copenhagen

First version received January 2014*; final version accepted August* 2015 (*Eds.*)

We define a class of dynamic Markovian games, *directional dynamic games* (DDG), where directionality is represented by a strategy-independent partial order on the state space. We show that many games are DDGs, yet none of the existing algorithms are guaranteed to find *any* Markov perfect equilibrium (MPE) of these games, much less *all* of them. We propose a fast and robust generalization of backward induction we call *state recursion* that operates on a decomposition of the overall DDG into a finite number of more tractable *stage games,* which can be solved recursively. We provide conditions under which state recursion finds at least one MPE of the overall DDG and introduce a *recursive lexicographic search* (RLS) algorithm that systematically and efficiently uses state recursion to find *all* MPE of the overall game in a finite number of steps. We apply RLS to find all MPE of a dynamic model of Bertrand price competition with cost-reducing investments which we show is a DDG. We provide an *exact non-iterative algorithm* that finds all MPE of every stage game, and prove there can be only 1, 3, or 5 of them. Using the stage games as building blocks, RLS rapidly finds and enumerates all MPE of the overall game. RLS finds a unique MPE for an alternating move version of the leapfrogging game when technology improves with probability 1, but in other cases, and in any simultaneous move version of the game, it finds a huge multiplicity of MPE that explode exponentially as the number of possible cost states increases.

Key words: Dynamic games, Directional dynamic games, Markov perfect equilibrium, Subgame perfect equilibrium, Multiple equilibria, Partial orders, Directed acyclic graphs, d-subgames, Generalized stage games, State recursion, Recursive lexicographic search algorithm, Variable-base arithmetic, Successor function.

JEL Codes: D92, L11, L13

3.4. *RLS Algorithm*

Having set up the machinery and showing how it is possible to jump directly from one feasible ESS to another using the jump (successor) function $\mathcal{J}(\gamma)$ we are now ready to provide a simple description of how the RLS algorithm works.

Definition 16. (RLS Algorithm). *Consider a finite state DDG $\mathcal{G}$ with $\mathcal{T}$ stages. The RLS algorithm consists of the following operations*

---

input : DDG $\mathcal{G}$
output: Set Λ of all feasible ESS which correspond to all MPE of the DDG $\mathcal{G}$
run the DAG recursion (9) to find all $N=\sum_{\tau=1}^{\mathcal{T}} n_\tau$ substages d of the game $\mathcal{G}$
initialize ESS sequence with always feasible ESS $\gamma^0=(0,0,\ldots,0,0)$, let $\Lambda=\{\gamma^0\}$
run the **state recursion algorithm** with $\Gamma=\gamma^0$
record the number of computed MPE in every d-stage game $\mathcal{G}_\tau(d)$ in the vector $ne(\gamma^0)$
for $k=1,2,\ldots$ **do**
    compute the next ESS in the sequence $\gamma^k=\mathcal{S}(\gamma^{k-1})$ using variable bases $ne(\gamma^{k-1})$
    denote τ_0 the stage corresponding to the highest updated digit of γ^k
    If $\gamma^k=(-1,\ldots,-1)$ **return** Λ
    by Lemma 6, γ^k is a feasible ESS
    run **state recursion algorithm** with $\Gamma=\gamma^k$ *only* for stages $\tau<\tau_0$ which are dependent on the highest updated stage τ_0
    record the number of computed MPE in every d-stage game $\mathcal{G}_\tau(d)$ in the vector $ne(\gamma^k)$
    update the set of feasible ESS found by RLS by setting $\Lambda=\Lambda\cup\{\gamma^k\}$

Note that the RLS algorithm repeatedly invokes the state recursion algorithm to solve the game only *partially*, which requires a trivial adjustment of the Definition 13. This is an important feature of RLS which cuts its run time approximately in half. There is no need to resolve the τ stage games unless the changes of the ESS γ^k at the current iteration relative to the ESS γ^{k-1} from the previous iteration could have affected the MPE in earlier stages of the DDG. The arrangement of the digits of the equilibrium selection string (21) ensures that resolving the stage games corresponding to the digits γ to the left of the highest update digit is unnecessary. Another trivial extension of the state recursion algorithm has to do with saving the quantities of the d-stage game MPE in the vector $ne(\gamma^k)$.

As mentioned above, the prior knowledge of the upper bound K on the number of MPE in the d-stage games is not necessary. This information is updated over the course of running the RLS algorithm, starting with the initialization at the always feasible ESS $\gamma^0=(0,\ldots,0)$. Each time the RLS algorithm encounters a new feasible ESS γ, it updates the maximum number of MPE at state points where the solution may have changed.

Theorem 5. (Strong convergence of RLS Algorithm). *Given the DDG $\mathcal{G}$, assume there exists a solution algorithm that can find all MPE of every d-stage game $\mathcal{G}_\tau(d),\forall\tau,d$, and that the set of all MPE $\mathcal{E}(\mathcal{G}_\tau(d))$ of every d-stage game is finite. Then the RLS algorithm will return the set $\Lambda=\{\gamma^0,\ldots,\gamma^{J-1}\}$ of all feasible ESS of game $\mathcal{G}$, and thus find all MPE of DDG $\mathcal{G}$, in at most $J=|\mathcal{E}(\mathcal{G})|$ steps, where $|\mathcal{E}(\mathcal{G})|$ is the total number of MPE of the DDG $\mathcal{G}$.*