

Plan, replan and plan to replan

Algorithms for robust courses of action under strategic uncertainty

Maciej Łatek and Seyed M. Mussavi Rizi
 Department of Computational Social Science
 George Mason University
 4400 University Drive, Fairfax, VA 22030
 mlatek, smussavi@gmu.edu

January 6, 2010

Abstract In this paper we present an efficient computational implementation of non-myopic n -th order rationality using multi-agent recursive simulation in which simulated decision makers use simulation to inform their own decision making. An agent is n -th order rational if it determines its best response assuming that other agents are $(n - 1)$ -th order rational with zeroth-order agents behaving according to a specified, non-strategic rule. We describe how to combine these two techniques with a replanning heuristic to create a decision rule, called REplanning N -th order RAtionality (RENORA), allow an agent to strategize for more than one move forward in a tractable manner. Our approach addresses (a) randomness of the environment, (b) strategic uncertainty arising when an opponent has more than one equally good courses of action to choose from and (c) failures in plan execution caused by either the environment or the opponent interference. To demonstrate the properties of RENORA, we introduce a model of a dynamic environment that encompasses both competition and cooperation between two agents, trace the relative performance of agents as a function of RENORA parametrization, and outline in detail the steps RENORA agents go through as they reason about the environment and other agents.

Keywords Recursive Agent-based Models, Multiagent Learning and Decision-making, Cognitive Architectures, Robust Replanning

1 Introduction

The departure point for this paper is n -th order rational agents. An agent is first-order rational if it calculates the best response to his beliefs about the strategies of zeroth-order agents and the state of the world. An agent is n -th order rational if it determines its best response assuming that the other agents are $(n - 1)$ -th order rational. n -th level rationality models have few degrees of freedom, often only the rationality levels of all strategic agents that can be calibrated from data. If this is accomplished, such models can perform *descriptive* and *normative* roles of a decision framework that guides agents on their courses of action (COA) in a multiagent setting. Combined with easy sensitivity analysis of results and an efficient multiagent formulation that can be solved even for complex environments, n -th order rationality is a convenient heuristic for reasoning in multiagent environments.

In this paper, we first integrate myopic n -th order rational with multiagent models. We then show how to extend such a framework to make them robust and tractable for non-myopic agents with long planning horizons. Finally, we introduce a

dynamic multiagent environment and use it to outline in detail the steps that endogeneously replanning n -th order rational agents go through as they reason about the environment and other agents. Finally, we perform sensitivity analysis of the extended n -th order rationality formulation.

2 Multiagent Recursive Simulation

Assume a model of reality Ψ either as a multiagent model that describes strategic interactions among K agents or as a statistical model that simply predicts some macro variables. Before we show how to introduce planning agents into Ψ , let us describe what questions it answers. Ψ

What can happen Defines the space of feasible COA for each agent and all possible sequences of interactions among agents.

What has happened Contains a library of historical trajectories of interactions among agents called historical behaviors library (HBL). If no actual information is

available, HBL is either empty or filled with hypothetical expert-designed interaction scenarios.

How agents value the world Codes every agent’s payoffs for any trajectory of interactions among agents based on the agent’s implicit or explicit preferences or utility function.

Latek et al. (2009) show that Ψ (a) can be decomposed into a state of the world C_t and agents’ current COA $\mathbf{p}_t = (p_t^1 \ p_t^2 \ \dots \ p_t^K)$ where p_t^i stands for agent i COA at time t and (b) maps C_t and \mathbf{p}_t into a realization of the future state C_{t+1} and current agents’ payoffs $\mathbf{r}_t = (r_t^1 \ r_t^2 \ \dots \ r_t^K)$ where r_t^i is the current payoff for agent i : $\langle \mathbf{r}_t \ C_{t+1} \rangle = \Psi(\mathbf{p}_t, C_t)$.

Agent i can use heuristics or statistical procedures to compute the probability distribution of payoffs for each COA it picks; then pick a COA that is in some sense “suitable”. Alternatively, it can clone Ψ , simulate the world forward; derive the probability distribution of payoffs for each available COA by simulation and pick a suitable COA. When applied to multiagent models this recursive approach to decisionmaking amounts to having simulated decisionmakers use simulation to choose a COA (Gilmer, 2003). Note that agents perceive Ψ with varying degrees of accuracy and have different computational capabilities to clone Ψ . So agents need not produce a clone of Ψ that is isomorphic to Ψ itself; however, in this paper we assume they do. We call this technology multiagent recursive simulation (MARS).

3 n -th Order Rational Agents

Agents in any Ψ pick COA that achieve a goal, for example maximizing the stream of expected payoffs for the planning horizon of h periods forward. If a Ψ contains strategic agents whose payoffs depend on the choices of other agents, such agents must have access to plausible mechanisms to compute optimum COA. n -th order rationality is one such mechanism. An n -th order rational agent (NORA) assumes that other agents in Ψ are $(n - 1)$ -th order rational and best responds to them. A zeroth-order rational agent acts according to a non-strategic heuristic such as randomly drawing a COA from the HBL or continuing the current COA. A first-order rational agent assumes that all other agents in Ψ are zeroth-order rational and best responds to them. A second-order rational agent assumes that all other agents in Ψ are first-order rational and best responds to them. NORA have inconsistent beliefs about the level of rationality each has. For example, observe that if the assumption of a second-order rational agent about other agents in Ψ is correct; they must assume that the second-order rational agent is zeroth-order rational agent.

NORA offer the following advantages in a multiagent settings: (a) Models can be heterogeneous in NORA level of ra-

tionality. (b) NORA do not require any learning phase to satisfy Hannan consistency: they converge to the best response to the other agents’ COA at every stage. (c) NORA can be efficiently implemented even for complex Ψ by using MARS. In the next section we introduce a structural design that introduces uses MARS to solve planning and replanning for NORA.

4 Robust Planning with MARS-NORA

4.1 Myopic Planning

To describe the algorithm that introduces myopic NORA into a Ψ , we denote the level of rationality for an NORA with $d = 0, 1, 2, \dots$. We label the NORA corresponding to level of rationality d as A_d and a set of its possible COA as ℓ_d :

- $d = 0$ A zeroth-order rational agent A_0 chooses COA in ℓ_0 ;
- $d = 1$ A first-order rational agent A_1 chooses COA in ℓ_1

and so forth. Now we can show how myopic NORA use MARS to plan COA.

ℓ_0 contains **non-strategic** COA that are not conditioned on A_0 expectations of what other agents will do. Without assuming that other agents optimize, A_0 arrives at ℓ_0 by using non-strategic heuristics like expert advice, drawing a COA from a probability distribution over the COA space or sampling the HBL for a COA. Example 1 shows possible choices for ℓ_0 used by an A_0 stock trader.

A trader holds a stock that has lost 15% value. He can sell the stock, hold it, or buy more:

1. If the industry stock value has shrunk less than 15%, sell. Else, hold.
2. With probability 0.1, sell or buy more. Else hold.
3. If in the last year the stock has not rebounded 90% of times within 2 weeks of a 15% devaluation, sell. Else hold.

Example 1: Rule-driven ℓ_0 for an A_0 stock trader.

Recall that an A_1 agent forms ℓ_1 by best responding to ℓ_0 adopted by another agent in Ψ whom it assumes to be A_0 . If A_1 assumption is true, the other agent does not assign a level of rationality to the A_1 agent. So A_1 finds a strategy that **on average** performs best when the A_0 agent adopts any COA in its ℓ_0 , integrating out the stochasticity of the Ψ . A_1 can sample its opponent ℓ_0 uniformly or according to the opponent empirical frequency of adopting each COA. Algorithm 1 shows this process.

Input: Set ℓ_0 for A_0 ; COA space for A_1 ; number of samples K

Output: Set ℓ_1 of optimal COA for A_1

```

foreach COA  $a_1$  available in  $A_1$  do
    foreach  $a_0 \in \ell_0$  do
        foreach  $i \leq K$  do
             $s = \text{cloned } \Psi$ ;
            query  $s(a_0, a_1) = A_1$  payoff;
        end
    end
    Compute average  $\bar{s}(a_1)$  over  $K$  samples;
end
Eliminate all dominated COA, arriving at  $\ell_1$ ;
Return  $\ell_1$ ; choose a single COA for  $A_1$  from  $\ell_1$ .
    
```

Algorithm 1: NORA($A_1, 1$)

Best response formation for A_2 follows a similar vein: An A_2 agent best responds to another agent who it assumes to be A_1 . Therefore, A_2 assumes that the other agent *assumes* that the A_2 agent is indeed A_0 . A_2 finds a strategy that **on average** performs best when the A_1 agent adopts **any** of its ℓ_1 COA. In order to accomplish this, the A_2 agent first computes a set of ℓ_1 for A_1 ; then it best responds to the ℓ_1 it has computed. Algorithm 2 shows this process.

Input: Set $\ell_1 = \text{NORA}(A_1, 1)$; COA space for A_2 ; number of samples K

Output: Set ℓ_2 of optimal COA for A_2

```

foreach COA  $a_2$  available to  $A_2$  do
    foreach  $a_1 \in \ell_1$  do
        foreach  $i \leq K$  do
             $s = \text{cloned } \Psi$ ;
            Calculate  $s(a_1, a_2) = A_2$  payoff;
        end
    end
    Compute  $\bar{s}(a_2)$ ;
end
    
```

end
Eliminate all dominated A_2 COA, arriving at ℓ_2 ;
Return ℓ_2 ; choose a single COA for A_2 from ℓ_2 .

Algorithm 2: NORA($A_2, 2$)

4.2 Non-myopic Planning

Algorithms 1 and 2 use MARS to solve the myopic planning problem for NORA. How can A_d derive optimum COA if (a) it wishes to plan for more than one step; (b) takes random lengths of time to execute a COA or aborts COA execution mid course, and (c) interacts asynchronously with other NORA. To address these issues, we introduce the notion of planning horizon h . While no classic solution to problems (b) and (c) exists, the classic method of addressing (a), finding the optimum of $h \times$ number of COA, leads to exponential explosion. The

following algorithm called RENORA solves (a), (b) and (c) simultaneously:

Input: COA space for A_d ; ℓ_{d-1} ; d ; h ; number of samples K

Output: Set ℓ_d of optimal COA for A_d

```

foreach COA  $a_d$  available to  $A_d$  do
     $s = \text{cloned } \Psi$ ;
    Assign initial COA to all agents  $\in s$ ;
    foreach  $a_{d-1} \in \ell_{d-1}$  do
        while  $s.\text{time}() < h$  do
            if  $a_d$  is not executing then
                RENORA( $A_d, d, h - s.\text{time}()$ )
            end
        end
        Accumulate  $A_d$  payoff  $\ += s(a_{d-1}, a_d)$ ;
    end
    Compute  $\bar{s}(a_d)$ ;
end
    
```

end
Eliminate dominated COA, return ℓ_d .

Algorithm 3: RENORA(A_d, d, h)

5 Experiments

5.1 Environment

To demonstrate the properties of RENORA, we use a multi-agent environment we call PushGame, a two-player stochastic game with 5 states A to E shown in Figure 1. Formally, a general-sum, two player, stochastic-game M on states $S = \{1, \dots, N\}$, and actions $A = \{a_1, \dots, a_k\}$ consists of:

- **Stage Games:** Each state $s \in S$ is associated with a two-player, fixed-sum game in strategic form, where the action set of each player is A . We use R^i to denote the payoff matrix associated with stage-game i .
- **Probabilistic Transition Function:** $P_M(s, t, a, a')$ is the probability of a transition from state s to state t given that the first agent plays a and the second agent plays a' .

In PushGame, each agent has to choose one of the two actions at each state: agent 1 has actions U and D and agent 2 actions L and R . A 2×2 matrix associated with each state codes payoffs p_1 for agent 1 and p_2 for agent 2 depending on the state, the agent and its opponent actions. Additionally, certain combinations of agent actions may cause states to change. For example, if agent 1 plays D and agent 2 plays L in state A , both agents receive payoff 0, but the state will change to B . States are grouped into three categories. State A does not favor any agent and requires coordination between agents to ensure payoff 1. If one of the agents deviates in order to secure a payoff higher than 1, it may break the symmetry of the game. States B and C favor agent 1 who receives a constant

payoff of 2 at the expense of agent 2 who receives either 0 or -1 . States D and E favor player 2.

At each asymmetric state, the stronger agent is predictable: agent 1 in states B and C always plays U ; agent 2 in states D and E always plays R . Suppose in state A agent 1 deviates and forces transition to state B . The weaker agent 2 has two choices: it can either avoid payoff -1 and coordinate with the stronger agent 1 to receive 0 or accept the punishment of -1 in order to return to the symmetric state A . Return to symmetry requires the weaker agent to accept a short-term loss in the hope of long-term gain. This deterministic setup for PushGame allows us to test the influences of agent rationality levels and planning horizons without the obfuscating effect of inherent randomness in the environment or strategic uncertainty.

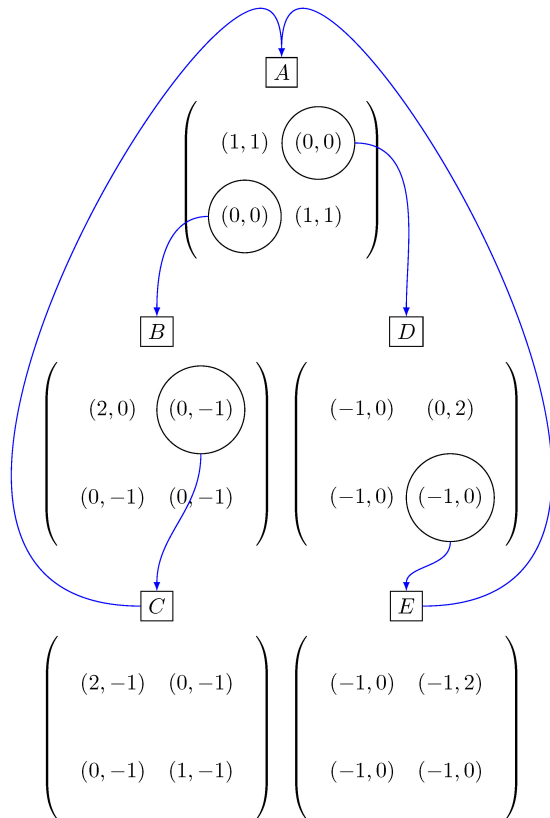


Figure 1: PushGame: A 5-state stochastic game used as a testbed to demonstrate the properties of RENORA. Possible transitions among states are denoted with \rightarrow and happen with probability 1 if agents play a proper combination of actions. If no transition is drawn, the state does not change from iteration to iteration.

5.2 Simulation traces

Figure demonstrates the mechanics of simulation cloning and replanning. We lay out the trace of a single call to

RENORA($A_1, 3, 3$) on Figure 2(a). Six outgoing paths appear on each reoptimization node: 3 of which are blue, corresponding to simulations cloned by agent 1; 3 are red, corresponding to simulations cloned by agent 2. Each bundle of 3 same colored paths corresponds to a single call to RENORA with one subtree shorter than the remaining two. The shorter subtree corresponds to the first instruction of RENORA where an A_d is figuring out the initial step by its A_{d-1} opponent. The remaining two subtrees evaluate the fitness of each of the two available actions available in each state of PushGame. Assuming that each agent reoptimizes after completing an action, every call to RENORA leads to other calls to RENORA with smaller d , shorter h or both. In the process of solving the replanning problem each agent uses cloned simulations to optimize over its COA and to predict the steps its opponents would take and the evolution of the states of PushGame. Repeated interactions between the two agents generate traces shown on Figure 2(b).

5.3 Influence of d and h

In order to assess the influence of d and h on the performance of a PushGame agent, we performed a simple parameter sweep outlined in Table 1, the results of which are summarized in Table where absolute and relative performance of agent 1 is averaged out and presented as a function of $h_1 - h_2$ and $d_1 - d_2$. Additionally, we enumerate the frequency with which cooperative state A is visited. We divide $(h_1 - h_2) \times (d_1 - d_2)$ into three regions:

$|h_1 - h_2| \geq 3 \wedge |d_1 - d_2| \geq 3$ One agents has a very short planning horizon and a low rationality level whereas the other has a long planning horizon and high rationality level. Cooperation is sustained and the more rational agent ensures fast return to state A . If agent 1 is the rational agent, it makes sure that the return to symmetry happens through a branch of PushGame that favors him;

$h_1 - h_2 \leq -2 \wedge d_1 - d_2 \geq 3$ Agent 1 has a higher level of rationality, but a much more shorter planning horizon than agent 2. Agent 1 is unable to make short-time tradeoffs and gets locked in an asymmetric branch that does not favor him. His absolute and relative performance is minimized;

$(h_1 - h_2) + (d_1 - d_2) \approx 0$ Both agents have similar cognitive capacities, cooperate often maximizing their absolute payoffs. If agent 1 has a higher planning horizon, it may also maximize its relative payoff.

Table presents the projection of a 4-dimensional parameter space into 2 dimensions; therefore, it should be interpreted with caution. Nevertheless, it proves that the RENORA algorithm allows an agent to make strategic decisions in a dynamic environment.

Parameter	Scenario value	Meaning
h	$\{1, \dots, 5\}$	Planning horizon. Each agent has its own h and d .
d	$\{0, \dots, 4\}$	Level of rationality. For $d = 0$, ℓ_0 is assumed to be uniform randomization over actionspace regardless of planning horizon.
numSamples	1	Number of samples taken to control for the randomness of the environment. PushGame is deterministic.
forwardLookingSamples	1	Number of samples taken to control for strategic uncertainty.
backwardLookingSamples	0	Number of historical COA that agents include in ℓ_0 .
maxT	50	Maximal time for an individual simulation run.
numRep	20	Number of repetitions per combination of h and d .

Table 1: Simulation parameters used in experiments.

6 Summary

In this paper, we introduced a context-independent multi-agent implementation of n -th order rationality for replanning agents with arbitrary planning horizons and demonstrated its functionality on a test cases. We presented algorithms that enable us to introduce n -th order rational agents into any multi-agent model and demonstrated that n -th order rational agents are model-consistent. We also showed how an n -th order rationality model deviates systematically from equilibrium predictions as agents are engaged in a multi-tiered game of out-guessing each others' responses to the current state of world.

References

Gilmer, J. (2003). The use of recursive simulation to support decisionmaking. In S. Chick, P. J. Sanchez, D. Ferrin, & D. Morrice (Eds.) *Proceedings of the 2003 Winter Simulation Conference*.

URL <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:THE+USE+OF+RECURSIVE+SIMULATION+TO+SUPPORT+DECISIONMAKING#0>

Latek, M., Axtell, R., & Kaminski, B. (2009). Bounded ratio-

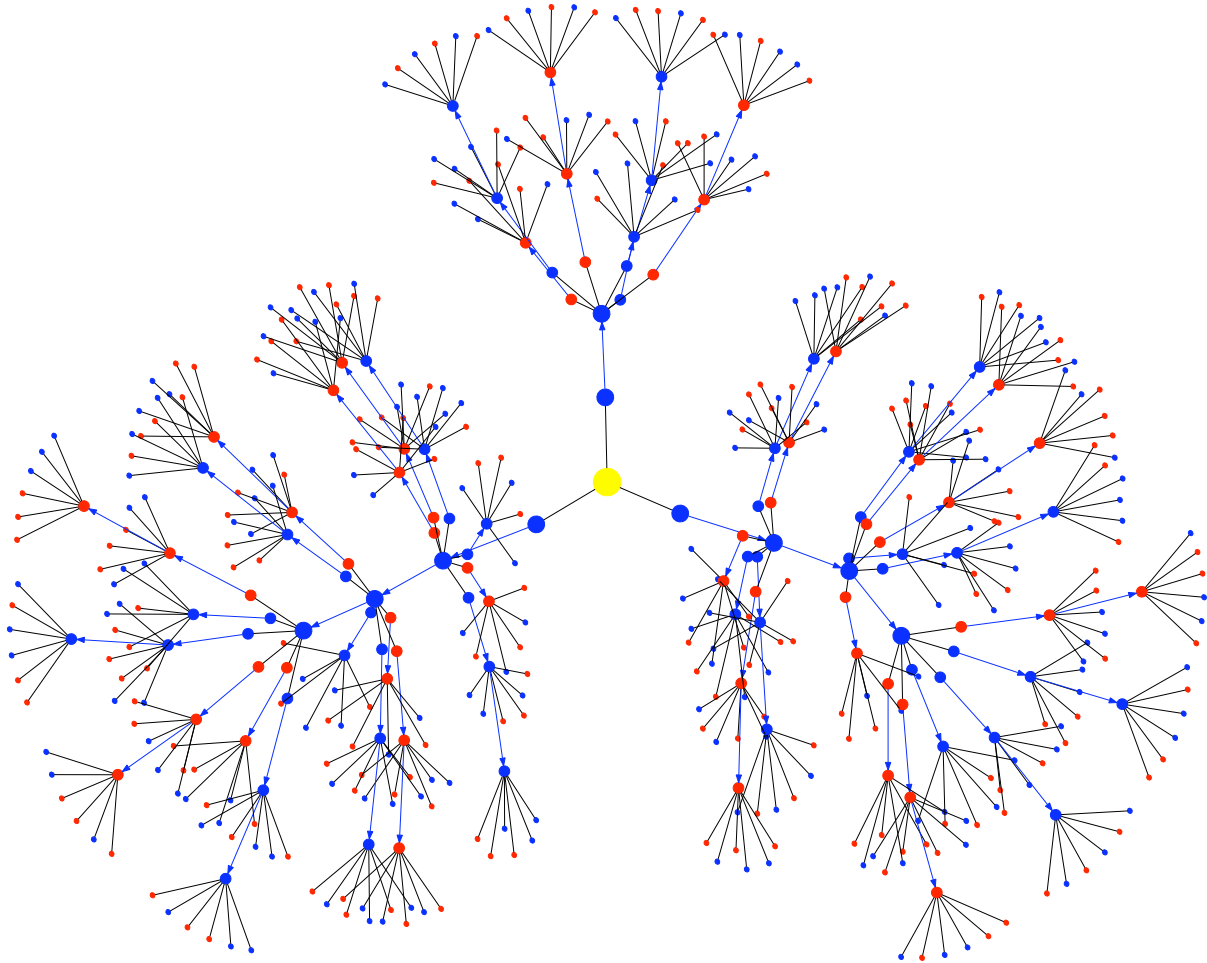
nality via recursion. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, (p. 457-464).

URL <http://portal.acm.org/citation.cfm?id=1558013.1558076>

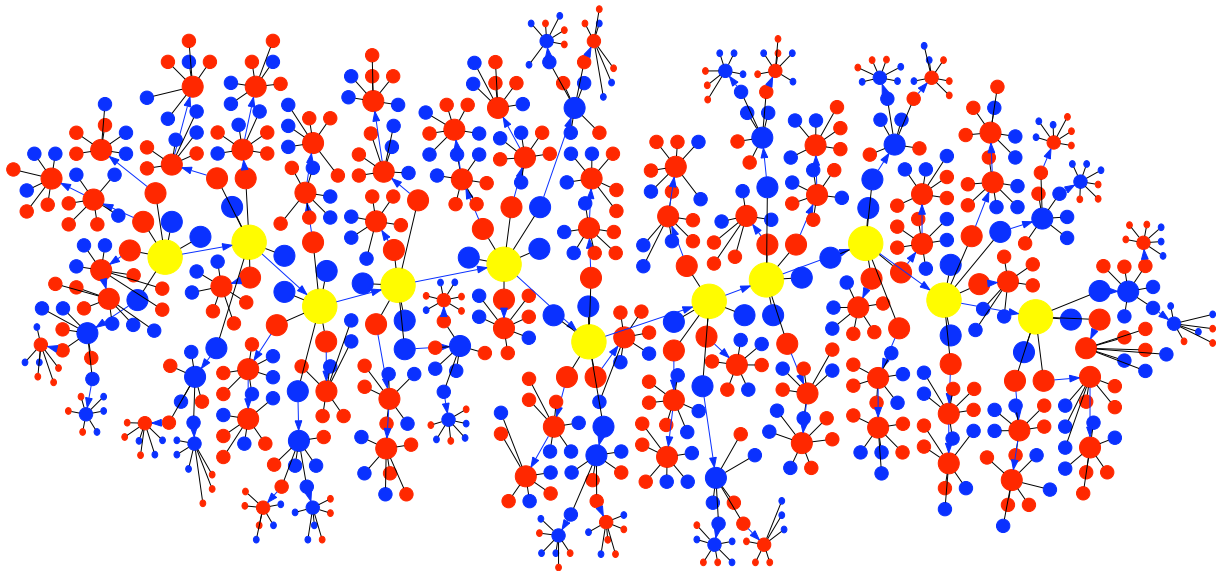
7 Biographies

Maciej M. Latek is a doctoral candidate in Computational Social Science at George Mason University. He holds a graduate degree in Quantitative Modeling from the Warsaw School of Economics. An operations researcher and data miner, Mr. Latek has worked with strategic interaction environment using a number of modeling approaches such as game theory, multi-agent modeling and data mining.

Seyed M. Mussavi Rizi is a doctoral candidate in Computational Social Science at George Mason University, developing multiagent models of conflict and political economy, protocols for model verification and validation and HSCB data requirements of large-scale agent-based models. He holds graduate degrees in economics, specializing in econometrics, from Tufts University, and in International Relations, specializing in international security, from the Fletcher School.



(a) A sample invocation of $\text{RENORA}(A_1, 3, 3)$ for agent 1. The small subtree in the middle corresponds to solutions of $\text{RENORA}(A_2, 2, 3)$ and $\text{RENORA}(A_1, 2, 3)$ used by agent 1 to obtain predictions of ℓ_2 .



(b) 10 iterations of PushGame with two $\text{RENORA}(2, 2)$ agents.

Figure 2: Mechanics of RENORA. Legend: ● the top-level universe, ● observations of cloned simulations, — cloning process, → observations of the same universe at different times. Blue instances are simulation cloned by agent 1, red by agent 2.

Difference of p1-p2		h1-h2								
		-4	-3	-2	-1	0	1	2	3	4
d1-d2	-4	0.02	-0.12	0.07	0.37	0.15	0.19	0.68	-0.02	0.30
	-3	0.15	-0.44	0.03	0.34	0.15	0.33	0.41	0.57	0.32
	-2	-0.35	-0.44	-0.38	0.14	0.07	0.25	0.33	0.62	0.14
	-1	-0.47	-0.54	-0.20	-0.04	0.09	0.30	0.50	0.49	0.50
	0	-0.54	-0.53	-0.43	-0.24	0.08	0.25	0.43	0.50	0.60
	1	-0.66	-0.51	-0.45	-0.33	-0.09	-0.02	-0.02	0.23	0.42
	2	-0.70	-0.50	-0.42	-0.27	-0.01	0.08	0.15	0.49	0.57
	3	-0.18	-0.40	-0.19	-0.26	-0.15	-0.15	0.15	0.37	-0.53
	4	0.03	-0.83	-0.27	-0.27	-0.06	0.07	-0.09	0.06	0.26

Absolute payoff p1		h1-h2								
		-4	-3	-2	-1	0	1	2	3	4
d1-d2	-4	0.97	0.66	0.69	0.85	0.44	0.60	0.72	0.05	0.31
	-3	1.05	0.73	0.90	0.87	0.79	0.79	0.69	0.70	0.30
	-2	0.28	0.50	0.52	0.77	0.61	0.73	0.75	0.72	0.30
	-1	0.46	0.45	0.73	0.83	0.88	0.94	1.00	0.81	0.72
	0	0.26	0.30	0.48	0.58	0.77	0.87	0.85	0.89	0.77
	1	0.15	0.42	0.53	0.66	0.76	0.87	0.85	0.86	0.80
	2	0.01	0.19	0.34	0.52	0.69	0.83	0.82	1.02	0.91
	3	0.19	0.16	0.35	0.41	0.55	0.64	0.91	1.14	0.57
	4	0.01	0.00	0.11	0.31	0.44	0.42	0.66	0.85	0.60

Frequency of state A		h1-h2								
		-4	-3	-2	-1	0	1	2	3	4
d1-d2	-4	0.98	0.85	0.72	0.61	0.47	0.57	0.31	0.35	0.36
	-3	0.77	0.70	0.66	0.56	0.52	0.48	0.35	0.34	0.35
	-2	0.59	0.61	0.66	0.59	0.57	0.54	0.52	0.37	0.37
	-1	0.55	0.56	0.51	0.62	0.58	0.53	0.55	0.42	0.47
	0	0.56	0.53	0.58	0.57	0.67	0.58	0.54	0.59	0.49
	1	0.41	0.46	0.54	0.56	0.56	0.69	0.54	0.49	0.49
	2	0.36	0.43	0.43	0.55	0.60	0.59	0.78	0.67	0.58
	3	0.36	0.36	0.45	0.52	0.51	0.64	0.57	0.60	0.60
	4	0.35	0.29	0.34	0.42	0.54	0.50	0.73	0.89	0.73

Figure 3: The first two tables show averages of absolute and relative payoffs of agent A_1 as a function of differences $d_1 - d_2$ and $h_1 - h_2$. The last table enumerates the frequency with which the cooperative state A is visited.