Recursive Agent-Based Simulation

Jonathan THALER

***Abstract*—In this paper we ask what influence recursive simulations has on the dynamics of an Agent-Based Simulation. We investigate the famous Schelling Segregation and endow our agents with the ability to project their actions into the future by recursively running simulations. Based on the outcome of the recursions they are then able to determine whether their move increases their utility in the future or not.**

We investigate the dynamics when using recursive simulations and compare it to the dynamics of the original model and our optimizing movement-strategy where agents are not projecting into the future. We hypothesize that in the case of a deterministic future this approach allows the agents to increase their utility as a group but we hypothesize that this is not the case when the future is non-deterministic as the power to predict is simply lost in this case. Further we hypothesize that a deterministic future endogenously turns into a non-deterministic one if the fraction of anticipating agents reaches a given threshold, rendering the predictive power in effect useless. This implies that MetaABS is only useful for optimization if not the whole system is trying to anticipate. We put strong emphasis on the point that MetaABS is *not* another optimization technique but that this research was investigated to raise and address interesting philosophical questions and implications: determinism, computational complexity, simulation argument,... TODO

TODO: important difference to game theory: the initiating agent makes no assumptions of the others behaviours, it lets them simulate, thus leaving them being autonomous

TODO: what if we let 2 agents play the prisoners dilemma recursively? which strategy does emerge? does tit-for-tat? problem: we need some strategy for the agents because we let them simulate the others, it is NOT a game-theoretic approach where an agent systematically plays through the opponents options. But is there a way to let tit-for-tat emerge? e.g. do random-bidding and then see what the other agent does and bid the payoff which is the highest in total? it makes a difference if the agent optimizes for best payoff alone or best payoff combined!

The main contribution of this paper is the introduction of recursive agent-based simulation, a completely new method in ABS, which we termed MetaABS.

I. INTRODUCTION

The 'meaning' of MetaABS is not really clear: how can it be interpreted? It is not so much about the dynamics but more on the philosophical questions it raises.

But also we wanted to check if the same happens as in the recursive simulation paper [1]: deterministic vs. non-deterministic AND one-agent recursion or all-agents recursion

We implemented our Meta-ABS in Haskell using the functional reactive programming paradigm following the Yampa library. We believe that pure functional programming is especially suited to implement Meta-ABS due to its lack of implicit side-effects and copying of data. [1]

[1]Code available under https://github.com/thalerjonathan/phd/tree/master/coding/papers/metaABS

II. BACKGROUND

A. Schelling Segregation

We follow in our implementation the original paper of Schelling as in [3] where we focus on the *Area Distribution* section. One assumes a discrete 2-dimensional lattice-world with NxM fields. Each field is either occupied by an agent of a given color (e.g. Red or Green) or is free. Each field has 8 neighbours, which denotes a Moore-Neighbourhood. In Schellings original work the lattice-world is limited at its borders but we assume a torus world which is wrapped around in both the x- and y-dimensions resulting in 8 neighbours also for fields at the border. The occupation density was set by Schelling to be about 70%-75% which he identifies as being a setting which allows the agents to move around freely without making the lattice-world too sparse. The agents make their move sequentially one after another. In each move an agent calculates the number of neighbours which are of equal color. If the number satisfies the agents needs about the neighbourhood then the agent is regarded as being 'happy' and will stay on this field. On the other hand the agent moves to the nearest unoccupied field which neighbourhood satisfies its needs.

TODO: add pictures

1) Anticipating behaviour: Schelling explicitly mentions in [3] that nobody anticipates moves of others. This is what we introduce using the recursive simulation. Agents select an unoccupied place as in the original model and move to it if it satisfies them in the future as well. The interpretation for that behaviour is: agents heard about an uprising place in town which they want to be part of, thus anticipating their moving there.

2) Non-Deterministic Extensions: TODO: for the non-deterministic future we need to extend this model with non-deterministic behaviour e.g. pick random position.

B. Related Research

TODO: [2] mention kirman complex economics where he investigates the model more in depth

III. META ABS

Informally, Meta-ABS can be understood as giving the agents the ability to project the outcome of their actions into the future. They are able to halt time and 'play through' an arbitrary number of actions, compare their outcome and then to resume time and continue with a specifically chosen action e.g. the best performing or the one in which they haven't died. More precisely, what we want is to give an agent the ability to run the simulation recursively a number of times where the

this number is not determined initially but can depend on the outcome of the recursive simulation.

A. Functional description

In this section we will give a formal description of how Meta-ABS works. Although we look at it from a programming-language agnostic way, we follow a functional description (rather than object-oriented) both in our pseudo-code and description because we think it allows for a much clearer formalization.

We assume an Agent-Behaviour Function with an argument of type AgentIn and a return value of type AgentOut.

TODO: for haskell-code use https://www.andres-loeh.de/lhs2tex/

```
agentBeh :: AgentIn -> AgentOut
```

We will keep the AgentIn and AgentOut types opaque and provide a few functions to work on them. We assume some domain-specific function which can create a new domain-specific solution by taking an AgentIn and returning an AgentOut. This can be used to iteratively update the agent

```
createNewSolution :: AgentIn -> AgentOut

agentBeh :: AgentIn -> AgentOut
agentBeh ain = createNewSolution ain
```

If an agent wants to initiate recursive simulations it calls the function recursive

```
recursive :: AgentOut -> AgentOut

agentBeh :: AgentIn -> AgentOut
agentBeh ain = recursive createNewSolution ain
```

This will tell the simulation system to initiate a recursive simulation. The system will then call the agent-behaviour function again but provides information that the agent is running recursively through the AgentIn type. To check this it is possible to query it by using the isRecursive function:

```
isRecursive :: AgentIn -> Bool

agentBeh :: AgentIn -> AgentOut
agentBeh ain
        | isRecursive ain = handleRecursion ain
        | otherwise = recursive createNewSolution ain

handleRecursion :: AgentIn -> AgentOut
```

The AgentIn type provides an additional function to get a list of AgentOut of all the past recursions. invariant: when in recursion and returning an aout and staying in recursion the aout returned will show up in the next recursive calls list at the top

```
recursiveAgentOuts :: AgentIn -> [AgentOut]
```

Further we need a function to stop the recursion if we are happy with one of the AgentOut:

```
unrecursive :: AgentOut -> AgentOut
```

Putting all together we can continue the calculation of recursions until the agent is satisfied with one AgentOut.

```
handleRecursion :: AgentIn -> AgentOut
handleRecursion ain
        | satisfies recursiveAgentOuts ain = unre
        | otherwise = recursive createNewSolution

satisfies :: [AgentOut] -> Bool
pickBest :: [AgentOut] -> AgentOut
```

We need to establish a very important invariant of time. We can query the simulation-time using the function

```
simTime :: AgentIn -> Real
```

The following invariant must hold: when calling simTime during recursion it must return constantly the same time as when starting the recursion. TODO: more formal

Now when following this approach and running more than one agent we will end up in an infinite regress as every agent will run the other agents which will lead them to initiate a recursion on their own. We need a mechanism to prevent agents which are recursively simulated by the initiating agent to run recursive simulations by their own. TODO: we need to define the terms precisely (initiating agent, recursively simulated agent)

TODO: how do we restrict recursion to only the recurring agent? either all or none or a list of agentids? the problem is that this is model dependent

So far we assumed (TODO: make that clear in the up description) that each recursive simulation is executed for 1 step and thus returning only one AgentOut. What if we want to run a recursive simulation for an arbitrary number of time-steps? We need again the mechanism to prevent 'other' agents from initiating recursive simulations but that is not enough, we need some mechanism of preventing the initiating agent of recursively initiating simulations in every time-step of the recursive simulation - ideally we should leave the choice to the initiating agent but it needs to know that it is in an evolving recursive simulation. within it it can initiate an additional recursive simulation or not - thus an agent can spawn an arbitrary number of recursive simulations within recursive simulations within recursive simulations...

First we need to establish a little bit of terminology to be able to unambiguously discuss the formal approach of Meta-ABS TODO: recursion-depth: TODO: recursion-replications TODO: recursion-length

B. Deterministic vs. Non-Deterministic future

The model as described in Background section is completely deterministic once it is running because it makes no use of a random-number generator and there are no other sources

of non-determinism - the next move of an agent is always completely predictable. If we introduce randomness through a random-number generator into our model then the future becomes non-deterministic *if the state of random-number generator when running recursive simulations is different from when the simulation is run non-recursively.*

TODO: What if the agents are shuffled every time before being traversed sequentially? The deterministic iteration is of importance here!

C. Computational complexity

the computation power grows exponentially with the number of recursion: give a formula depending on number of agents, recursion depth, independent moves of an agent and number of time-steps

D. Philosophical implications

1) Omega Point: the limit case: tiplers omega point and paper about god and the simulation argument accelerating turing machine: finishes after 1 time steps what would be the outcome in a zeno machine/accelerated turing machine if we don't restrict the recursions of 'others' and self?

https://en.m.wikipedia.org/wiki/Omega_Point https://en.m.wikipedia.org/wiki/Zeno_machine https://en.m.wikipedia.org/wiki/Hypercomputation

2) Multidimensional Computation: we are spanning up 3 dimensions: recursion-depth, replications, and time-steps

3) Emergent Non-Determinism: the prediction may work for a single agent but what if more and more agents predict their future? within the prediction no recursion is run so no 2nd level anticipation. hypothesis: increasing the ratio of predicting agents will decrease the effectiveness of the predictions because the future becomes then in effect non-deterministic =¿ non-determinism as emerging property? is there a limit e.g. up until which ratio does the average utility of the predicting agents increase?

the agent who is initiating the recursion can be seen as 'knowing' that it is running inside a simulation, but the other agents are not able to distinguish between them running on the base level of the simulation or on a recursive level

4) Perfect Information: The main problem of our approach is that, depending on ones view-point, it is violating the principles of locality of information and limit of computing power. To recursively run the simulation the agent which initiates the recursion is feeding in all the states of the other agents and calculates the outcome of potentially multiple of its own steps, each potentially multiple recursion-layers deep and each recursion-layer multiple time-steps long. Both requires that each agent has perfect information about the complete simulation *and* can compute these 3-dimensional recursions, which scale exponentially. In the social sciences where agents are often designed to have only very local information and perform low-cost computations it is very difficult or impossible to motivate the usage of recursive simulations - it simply does not match the assumptions of the real world, the social sciences want to model. In general simulations, with no direct link to the real world, where it is much more commonly accepted to assume perfect information and potentially infinite amount of computing power this approach is easily motivated by a constructive argument: it is possible to build, thus we build it. What we are ultimately interested in is the influence on the dynamics. Note that we identified the future-optimization technique as being locally. This is still the case despite of using global information for recurring the simulation - the reason for this is that we are talking about two different contexts here.

IV. Results

In this section we report and discuss the results of our experiments with Meta-ABS. In Table I we give the configuration of the model which is the same for all experiments. For the future move-optimization additional parameters are set, which are mentioned in the respective section.

What we are interested in are the following dynamics

1) Global happiness (Yes / No) over time
2) Global similarity over time
3) Global change of similarity between steps

A. Non-optimizing

Agents move to the nearest free spot. This is included because it is needed for the hypothesis for anticipating in non-deterministic future Hypothesis: very slow convergence, can't solve high density with high similarity requirements.

B. Optimizing

Agents move to the nearest free spot which satisfies their requirements

Hypothesis: should have much faster convergence than non-optimizing, areas act as attractors

C. Anticipating

TODO: how many agents are predicting? fraction between 0.0 and 1.0 TODO: compare the performance of the predicting-agents to the non-predicting ones

Prediction-Ratio hypothesis: increasing the ratio of predicting agents will decrease the effectiveness of the predictions because the future becomes then in effect non-deterministic

Deterministic future Hypothesis: faster convergence than optimizing

Non-Deterministic future Hypothesis: complete breakdown, falling back to dynamics of Non-optimizing

V. Conclusion and further research

So far we only looked at recursive simulation in a simulation with a strictly sequential update-strategy where agents are updated in sequence after each other as defined in TODO: cite my Art-Of-Iteration Paper. We leave the question of how Meta-ABS would apply to the parallel update-strategy and whether it is reasonable to extend it to that strategy or not for further research.

TABLE I: Model Configuration

Dimensions	50 x 50
World-type	Torus
Density	0.75
Similarity required	0.8
Agent-distribution	50% Red, 50% Green
Local-movement distance	5
Find-free-place retries	4

REFERENCES

[1] GILMER, JR., J. B., AND SULLIVAN, F. J. Recursive Simulation to Aid Models of Decision Making. In *Proceedings of the 32Nd Conference on Winter Simulation* (San Diego, CA, USA, 2000), WSC '00, Society for Computer Simulation International, pp. 958–963.

[2] KIRMAN, A. *Complex Economics: Individual and Collective Rationality*. Routledge, London ; New York, NY, July 2010.

[3] SCHELLING, T. Dynamic models of segregation. *Journal of Mathematical Sociology 1* (1971).