A Knowledge Representation Framework for Evolutionary Simulations with Cognitive Agents

Nausheen Saba Shahid, Dan O'Keeffe, Kostas Stathis
Department of Computer Science, Royal Holloway University of London, UK
nausheen.shahid.2017@live.rhul.ac.uk, {daniel.okeeffe, kostas.stathis}@rhul.ac.uk

***Abstract*—We propose a generic knowledge representation framework that supports evolutionary game-theoretic simulations using cognitive agents. The framework allows an experimenter to test a population of such agents via generations to study how specific population behaviours evolve over time. A generation is composed of rounds which can be further divided into encounters according to model-specific requirements. As agents in the population interact, events (caused either by agent actions or by separate environment processes) take place. These events change the environment, changes are then perceived by agents that, in turn, decide to take new actions that affect the environment. This process continues until it is time to evolve a new generation, when strategies of the fittest players are selected for the next generation to start evolving. This evolutionary loop continues until all the terminating conditions of the simulation are met. We use the framework to show how to successfully repeat existing experiments from evolutionary simulations of agent cooperation. Our results validate our framework and pave the way for EVO-COGNISIM, a simulation platform that implements the key aspects of the framework in a systematic manner.**

***Index Terms*—Game-theoretic simulation, Cognitive agents, Cognitive simulation platforms, Evolution of Cooperation.**

I. Introduction

We are interested in simulation models that involve modeling human systems to gain an understanding of how they behave over time. In particular, we are concerned with models that need to represent how certain behavioural characteristics, whether specific to individuals or groups, are so robust and resilient that survive over generations in a certain environment [1]. These types of models are called evolutionary models as they are based on evolutionary concepts of selection and reproduction [2].

To study the evolution of human behaviour many game-theoretic simulation models have been proposed, especially agent-based models of how individuals cooperate [3], [4]. Although these models provide useful insights of how evolution affects cooperation in specific application domains, their major shortcomings are that the agent modelling is often specified by low-level data structures and algorithms that, despite being efficient, do not always reflect the high-level concepts these components are meant to represent [5]. On top of this, the agent modelling being carried out is often based on mathematical methods that make their behaviour difficult to interpret, in particular by experimenters who simply act as simulation users, and were not involved in their design, specification or implementation.

There are many efforts [6], [7], where a society of agents has been modeled and evolved as an evolutionary simulation, some expressed as complex mathematical strategies such as [8]. Also, in some efforts, popular simulation platforms such as NetLogo [9], Repast [10] and AnyLogic [11] have been used to implement the simulation model. The major shortcoming of these efforts is that the agents in these approaches lack the cognitive complexity in symbolic terms to represent human behaviour, i.e., these models are not based on a set of symbols that can be combined (e.g., concatenated) in a multitude of ways, according to precise grammatical rules, and where both elementary symbols and any admissible combination of them can be assigned with meaning (i.e., each symbol can be mapped into some entity from the domain at hand). Another limitation is that these works don't discuss explicitly how they have formulated evolution. Did they perform it at the platform level or the agent level? As a result, it is difficult to repeat their experiments, especially if one is using a different programming language, or platform, as there is a lack of an explicit specification of evolution that is executable and interpretable.

This highlights the need for a more transparent and flexible simulation platform, where the evolutionary behaviour of components is interpretable and more understandable by humans. One of the contribution of our work is to be able to support experiments evolving generations of multiple entities conceptualised as populations of cognitive agents. As cognitive agent frameworks are more demanding computationally (whether they are autonomous [12] or not), another contribution of this work is to offer experimenters optimization opportunities that make the evolutionary setup scalable and, thus, a complex simulation feasible. To exemplify these contributions, we present our framework that we refer to as EVO-COGNISIM. To demonstrate the usefulness of EVO-COGNISIM, another contribution of our work is that we show how to use the framework to program an existing model for evolutionary cooperation [13], illustrating how to reproduce existing simulations in a systematic manner.

This paper is structured as follows. Section 2 presents the related work and the gaps that have motivated our effort. In Section 3 we select a model as a case study to exemplify the requirements of our framework. Section 4 introduces the reference model architecture of EVO-COGNISIM, provides the knowledge representation of its components, and identifies areas for improving its operational performance. EVO-

COGNISIM is then used in Section 5 to implement the agent-based modelling of the selected use case, whose results are faithfully reproduced and evaluated, demonstrating the usefulness of our platform. We summarise our effort in Section 6, where we also discuss directions for future work.

II. Related Work

There have been some efforts to use cognitive agents for social simulation [14]–[16] but these simulations were focused on modeling social phenomena such as simulation of escaping panic [14], simulation of survival in a group, crowd simulation of emergency response and pedestrian mobility after an earthquake from a non-evolutionary perspective. A notable one is discussed by Singh [17], where he proposed a middleware to integrate Belief-Desire-Intention (BDI) agents with the Agent-Based Modeling (ABM) Platform. These efforts, however, did not capture the evolutionary aspect of agent behaviour, and so cannot support evolutionary game-theoretic models as we do.

A more recent work with BDI agents expressed in the agent-oriented language Jason is described in [18]. This work extends the widely used JaCaMo platform [19] for simulation applications and, although important for building practical simulations because it links concepts such as events, activities and workspaces, it does not explicitly show how the simulation cycle is formulated symbolically, nor does it link the simulation cycle to the evolution of generations explicitly, as provided in this paper.

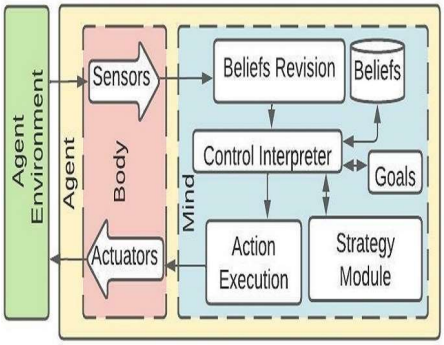


Fig. 1: The COGNISIM Agent Architecture from [20].

The work closest to our approach is COGNISIM [20]. This provides a cognitive agent model as shown in Figure 1 with an associated knowledge representation to express game-theoretic interactions in simulations. As shown in Figure 2, however, the agent environment of COGNISIM only supports tournament-based experiments for a population of agents but abstracts away from how they might evolve in different generations. To address this issue, this work non-trivially complements COGNISIM by providing the missing knowledge representation framework for evolutionary simulations, using the environment evolutionary cycle of [21] as a base.

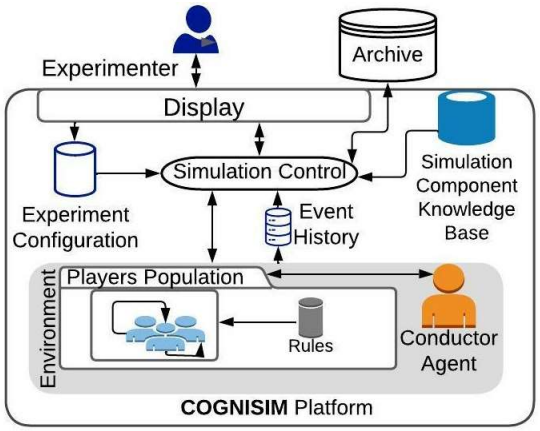


Fig. 2: The COGNISIM Reference Model from [20].

III. Evolutionary Cooperation Use Case

To understand the requirements of an evolutionary model that need to be incorporated into COGNISIM, we next select an evolutionary model from the literature to analyze as a use case. Many mathematical models have been proposed to study the evolution of cooperation among self-interested agents e.g. see [22], [23]. We will follow the model of [13], which arguably is considered the state-of-the-art in the field of gossip-mediated evolutionary cooperation. According to this model [13], we need a population of n players to interact T number of times. At each time point $T_i \in T$, players participate in g *giving* games and r *gossip* sessions. In a giving game a *donor* is selected randomly to give benefit (i.e. *cooperate*) or not (i.e. *defect*) to a randomly selected *recipient*. Similarly, in a gossip session, a randomly selected *speaker* evaluates $n-1$ players to a *listener*, who is randomly selected too (but not evaluated by the speaker). Every player maintains an *image score* of every other player; intuitively an internal representation of another player's reputation (s_{ij}) using the following equation:

$$s_{\mathrm{ij}} = C(i \leftarrow j) - D(i \leftarrow j) + \rho \sum (u \neq i)[G_{\mathrm{j}}(i \leftarrow u) - B_{\mathrm{j}}(i \leftarrow u)] \quad (1)$$

where ρ is the weight the player i gives to the gossiping information in the calculation of other player (j) 's image score. C_{ij} is the number of times j has cooperated with player i in the past, D_{ij} is the number of times player j has defected with player i in the past. G_{ij} is the number of times i has received a good gossip about player j from its trusted sources and B_{ij} is the number of time player i has received bad gossip about player j from its trusted sources. Also, every player follows a strategy with sub-strategies using the following parameters:

$$F = (\rho, k, q_G, q_B, q_R, \alpha) \quad (2)$$

where ρ is already defined, k is the discrimination threshold that a player (i) uses when selected as a donor and gives benefit

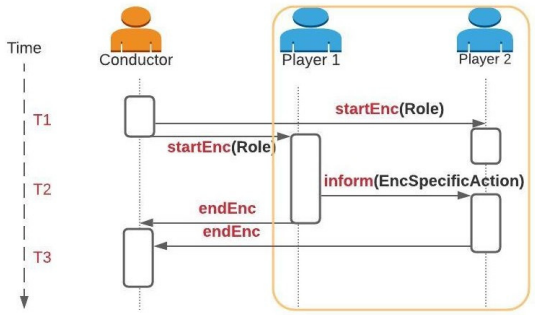


Fig. 3: Flow of an Encounter

to a recipient player (j) if the s_{ij} is greater or equal to k. q_G and q_B are the criteria for rating a player as *good/bad* respectively during a gossip story. Moreover, a gossiper can be self-advertising ($\alpha = 1$) or non self-advertising ($\alpha = 0$). q_R represents the image score criterion of a player whose gossip a player trusts.

Depending on the discrimination threshold k, players are categorized as [13]:

1) *Intrinsic cooperators* ($k \leq 0$), with two further subtypes:
   - *Stern discriminators* ($k = 0$);
   - *Generous discriminators* ($MinImageRange < k < 0$).
2) *Intrinsic defector* ($MaxImageRange \geq k > 0$).
3) *Unconditional cooperator* ($MinImageRange \geq k$).
4) *Unconditional defector* ($MaxImageRange < k$).

Based on q_G and q_B values, the following strategies for a gossiper are provided:

- *Fair* ($q_G = q_B = 0$).
- *Biased* ($q_G = q_B = k$).
- *All Good (ALLB)* ($q_G, q_B < MinImageRange$).
- *All Bad (ALLG)* ($q_G, q_B > MaxImageRange$).

We formalize the above cooperation model as a multi-agent environment with cognitive agents of the form:

$$M =< A, R_{beh}, R_{env}, Gen, G, R, Rounds > \quad (3)$$

where $A = \{a_1, a_2, ..., a_{n+1}\}$ is a set of $n + 1$ agents consisting of one *conductor* agent and n *player* agents. $R_{beh} = \{rb_1, rb_2, ..., rb_p\}$ is a set of behavioural rules representing the agent strategies consisting of both *discrimination* and *gossiping* sub-strategies. An agent a_i should be initialised with a subset of these behavioural rules ($R^{a_i}_{beh} \subset R_{beh}$). R_{env} are all the domain specific environment-level rules that govern the simulation. It covers both generation level rules as well as evolution rules used for evolving one generation to another. Generation level rules are the same as rules in the tournament setup. Gen is the total number of generations to be evolved, and $Nc \in \{1..Gen\}$ is the current generation evolving. A generation goes through a number of $Rounds$ before it terminates.

In [13], a round allows G giving games and R gossiping sessions to take place (in a sequence) in a single time step. We can treat each giving game or gossiping session as an *encounter* with three time steps, as illustrated in Figure 3. The evolution of each generation is like a single tournament of COGNISIM, but the way the conductor agent conducts the simulation is different. For example, in this model, the conductor randomly selects two agents and requests them to start an encounter by assigning them roles. For the purpose of this discussion, we assume it is a giving encounter. Then, the donor (Player1 in Figure 3) agent will inform the recipient (Player2 in Figure 3) about its specific action. In the next cycle, it informs the conductor that the encounter has been completed. A similar interaction protocol is used for a gossiping encounter. A simulation experiment is then an environment that evolves generations of cognitive agents. In the next section, we will discuss such a new framework.

IV. EVO-COGNISIM

We discuss how to complement COGNISIM to simulate evolutionary models.

A. EVO-COGNISIM Reference Model

The component architecture of EVO-COGNISIM is shown in Figure 4. As with tournaments in COGNISIM(or other platforms [24]), the simulation management components such as *Display*, *Experiment Configuration*, *Simulation Control*, *Simulation Component Knowledge Base (SCKB)*, *Event History* of the framework interact in the same way. The main difference now is that we need an *Evolution Rules* knowledge base so the *Simulation Control* can determine which strategies survive from one generation to another. Also, the agent environment consists of generations of player agents whereas the conductor agent resides in the agent environment until the end of the simulation and is responsible for starting and conducting all the encounters in each round of a generation.

B. EVO-COGNISIM Components

Like COGNISIM, an agent of EVO-COGNISIM is a component that has a default functionality to *perceive* the environment, *revise* its internal state with the percepts it receives, *decide* which action to perform next, including actions that allow the agent communication with other agents and then *perform* the action. In EVO-COGNISIM, the internal state of agent i at time t, denoted as IS^t_i is characterized as:

$$IS^t_i := KB_i \bigcup BB^t_i \quad (4)$$

where KB_i is a static knowledge base component defined at the time the agent is created and BB^t_i is the belief base component of i that can change as time t progresses. A component like KB_i in EVO-COGNISIM is defined using logic-based rules of the form:

rule(C, H, [B1,B2,..Bn]),

where C is a unique identifier of the component, H is the head of the rule, and the list [B1,B2,...,Bn] represents

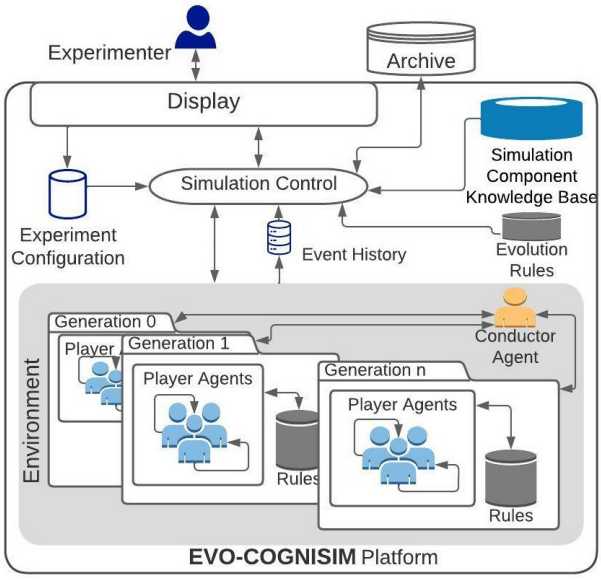


Fig. 4: EVO-COGNISIM Reference Model

the conditions in the rule's body. For example, we show below how a *stern cooperator* agent component ag1 (with discrimination threashold equal to 0) knows whether an image score (ImgScore) of another player is good or not.

```
rule(ag1, discrimination_threshold(0),[]).
rule(ag1,good_image(ImgScore),[
            discrimination_threshold(K),
            ImgScore >= K
]).
```

The first rule is a fact, i.e. the body is empty, while the second rule contains conditions that determine what is a good image score. As in COGNISIM, a vanilla meta-interpreter demo/2 is used to infer what is known by an agent and deals with additional aspects such as negation-as-failure, arithmetic and system specific operations [25]. A call to agent ag1 of the form ?- demo(ag1, good_image(5)), will allow us infer that the image score 5 is good, given what that agent knows.

TABLE I: Event Calculus Ontology in COGNISIM.

Predicate	Description
happens_at(E, T)	Event E happens at time T.
holds_at(F=V, T)	Fluent F has value V at time T.
holds_for(F=V, [Ts,Te])	Fluent F continuously has value V from time Ts to time Te.
broken(F=V, [Ts,Te])	Fluent F has changed value V from time Ts to time Te.
initiates_at(E, F=V, T)	Event E initiates value V for fluent F at time T.
terminates_at(E, F=V, T)	Event E terminates value V for fluent F at time T.

In COGNISIM organising an agent's knowledge base as a component described by rules is important, but we still need to address how to represent belief bases as components that change over time. For this we use the *Event Calculus (EC)* that relies on three basic constructs to represent change: *events*, *fluents* (state variables whose values change) and *time points* with a linear time model. Many versions of the EC exist [26], here we use the version described in [27] but with *multi-valued fluents* [28]. Events happen instantaneously and are represented in terms of the time point when they happen. An event initiates/terminates the value of a fluent in the state of interest indexed by a time point. Table I summarises the ontology of the domain-independent axioms we employ to represent state changes in the environment or the agents.

We define the general axioms of the calculus as in [27], but for caching we adopt the update mechanism described in [29]. We can then use these techniques to query evolving components by adding to the meta-interpreter a bridge rule:

```
demo(C, F=V@T) ← holds_at(C:F=V, T).
```

The bridge rule interprets time-dependent conclusions of the form F=V@T in a component C as component indexed fluents of the form C:F=V at time T using the EC axioms. We can now treat the agent beliefs as an evolving and time-dependent component. For example, to calculate the image score of another agent X at time T, an agent ag1 uses the rule:

```
rule(ag1, image(X,ImgScore,T), [
                    cooperation(X)=C@T,
                    defection(X)=D@T,
                    goodgossip(X)=G@T,
                    badgossip(X)=B@T,
                    gossipweight(P),
                    ImgScore is (C-D) + P*(G-B)
])
```

where beliefs such as (cooperation(X)=C and defection(X)=D are maintained as fluents in the Event Calculus level, while conditions such as gossipweight(P) are maintained as facts. Now, to compute with the rule we need to call the meta-interpreter to answer a query of the form:

```
?-demo(ag1, image(ag2,Img,3)).
```

Such a query will answer the question of *what is the image score of player* ag2 *from the point of view of* ag1 *at time* 3? and will instantiate the computed image score in variable Img.

C. Teleo-reactive Behaviours

In EVO-COGNISIM, each strategy a player uses to achieve a goal is represented as a teleo-reactive program [30]:

$$
\begin{aligned}
G\colon \{ & \\
& C_{11}, C_{12}, ..., C_{1k} \rightarrow A_1, \\
& C_{21}, C_{22}, ..., C_{2l} \rightarrow A_2, \\
& ....., \\
& C_{n1}, C_{n2}, ...C_{nm} \rightarrow A_n \\
\} &
\end{aligned}
$$

where G is the goal that an agent is pursuing and $A_i(i \in \{1...n\})$ is either an atomic action or a sub-goal SG of G. A condition of the form $C_{ij}(i \in \{1...n\} \wedge j \in \{k, l, .., m\})$ are conditions that need to hold at a given time t to make the agent select a particular Ai. The ordering of condition-action rules above is important as the conditions of the i^{th} rule implicitly imply the negation of all the conditions of the $i-1$ rules. For atomic actions, teleo-reactive rules are translated in the KB_{Ag} as selection rules of the form:

rule(Ag, select(E, G, A, T), [C_1,C_2,..., C_m]).

E represents the domain specific encounter (information required to distinguish different interactions by the implementation), G represents the goal that agent Ag is pursuing, and A is the action that is to be selected at time T. Similarly, {C_1,C_2..., C_m} represent the conditions that must be evaluated to true for the action to be selected. An example of a selection rule when the goal is to act as a stern cooperator is given below:

```
rule(ag1, select(E, be_stern, cooperate, T),[
                    rec_of(E)=R,
                    image(R,Img,T),
                    good_image(Img)
]).
```

Such a rule states that if the agent ag1 has the be_stern goal in an encounter E at T, then it should cooperate with a recipient R if its image Img is good according to ag1's knowledge. A similar rule structure can deal with an action that is sub-goal SG, following the format below:

rule(Ag, select(E, G, A, T),[C_1,C_2,..., select(E, SG, A, T)]).

SG now represents a whole sub-behaviour that agent Ag has to pursue at time T, and return action A assuming that SG and G are not conflicting. An example of such a rule is shown below:

```
rule(ag1, select(E, be_stern_fair, A, T),[
                    type_of(E)=giv,
                    my_role(E)=donor,
                    select(E, be_stern, A, T)
]).
```

The rule above is required for *stern cooperation* and *fair gossip* and assumes a top level goal called be_stern_fair for the behaviour of the agent to be triggered during the decision-making process using demo/2 to interpret teleo-reactive rules.

Teleo-reactive rules in EVO-COGNISIM are similar to the BDI model [31], except they do not rely on planning from first-principles or explicit intention management, as teleo-reactive programs are plan libraries where the intentions are implicit as part of the rule ordering and goal/sub-goal execution. As we will discuss in a while, this type of basic cognition is flexible and robust enough to represent quite complex behaviours.

D. Environment Evolution

In the evolutionary context, the predicates of the core *Simulation Control* are now as shown next.

```
evolve_all_for(gens(Nc,N),[Te,Te])←
                    Nc>N,
                    finalize_at(gens(Nc,N), Te).
evolve_all_for(gens(Nc,N),[Ts,Te])←
                    Nc≤N,
                    initialize_at(gen(Nc,Ti, Tj), Ts),
                    evolve_one_for(gen(Nc),[Ti, Tj]),
                    evolve_all_for(gens(Nc+1,N), [Tj,Te]).

evolve_one_for(gen(Nc),[Ts, Te]) ←
                    Ts>Te,
                    finalize_at(gen(Nc),Te).
evolve_one_for(gen(Nc),[Ts, Te])←
                    Ts≤Te
                    consume_at(gen(Nc), Es, Ts),
                    display_at(gen(Nc), Es, Ts),
                    optimize_at(gen(Nc), Ts),
                    evolve_one_for(gen(Nc),[Ts+1, Te]).
```

The first clause of evolve_all_for/2 halts the simulation when the number of the current generation Nc has exceeded the number of generations specified at the start of an experiment N and returns the simulation end time Te, as well as finalizing/saving the results. The second clause takes the number of generations N for the experiment, the current generation Nc, and the starting time Ts, initializes Nc and determines its starting and ending times (Ti and Tj respectively), evolves it, and proceeds to the next generation Nc+1 until all N generations are simulated. The evolution of a single generation Nc is dealt within evolve_one_for/2, which takes the starting time Ts and ending time Te, consumes all the events Es that take place in the agent environment, displays the transition to the next state to all agents, including the user. If necessary, a state is optimised and simulation moves to the next cycle, until the end time Te is reached. The evolution of a generation stops if Te is reached, where the generation is finalized.

1) Initialising a Generation: As seen in the previous section, an experiment needs to set up a population of agents, which will need to be initialised and associated with a generation number, Nc. We specify this in EVO-COGNISIM using initialize_at/2 which is similar to the initialization of a tournament. In the first generation, we initialise all the *player* agents and the *conductor*, and we set each player's P fitness to zero; a variable stating how fit a player agent is. A generation will then evolve until it reaches termination, at which point it will need to evolve into a new one using the notion of *cultural reproduction* [32] [13], where the strategies of the fittest players get more chances of survival to transfer into the next generation. To support this, we need a second definition of initialize_at/2 for initializing subsequent generations.

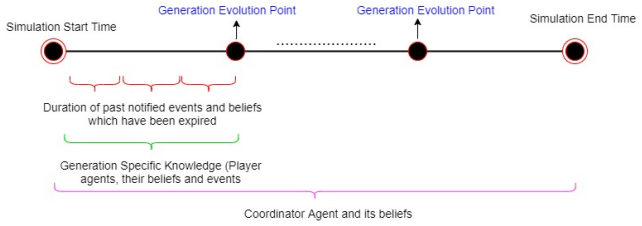


Fig. 5: Time span of generations in memory

```
initialize_at(gen(Nc,Ti,Tj),Ts)←
        not start_time(Ts),
        calculate_duration(Nc,Ts,Ti,Tj),
        update_at(initially(generation(Nc)=true),Ts),
        players_of(Nc-1,Players),
        selection_of_fittest(Players, Parents),
        strategies_of(Parents, Strategies),
        new_players(Strategies, NewPlayers),
        forall(member(P, NewPlayers),
              (update_at(player_of(Nc,P),Ts),
               initialize_at(player(P), Ts),
               update_at(initially(fitness(P)=0),Ts))),
        clear_at(player_agents(Nc-1), Ts).
```

Those players whose Strategies survive to the next generation are selected as Parents. This selection is performed in selection_of_fittest/2. Then the Strategies of the Parents players are used to create NewPlayers using new_players/2. These new players are then associated with the next generation, their knowledge bases get initialized and their fitness is set to zero. As a final step in this process, we remove the previous generation from the working memory using clear_at/2.

E. Performance Optimization

As evolutionary simulations generate a large number of events, to improve the simulation efficiency, the concept of *forgetting* in COGNISIM is re-specified in EVO-COGNISIM to forget all events and agent knowledge when a generation ends. Figure 5 shows the life span of different types of knowledge in the working memory of the EVO-COGNISIM. Within a generation forgetting is the same as COGNISIM. The relevant predicate such as optimize_at/2 [20] takes the generation number Nc as well with time T.

```
clear_at(players(Nc),T)←
    findall(A,(role_of(A,player), generation_of(A,Nc)), As),
    forall(member(P,As),clear_at(player(P),T)).
```

The major difference now is that at the time of evolution of a generation from one to another, first a new generation is initialized, then the previous generation is forgotten using clear_at/2, a predicate that retracts knowledge from the memory (see section IV-D1). If it has been called with players(Nc) as the first parameter, it searches for all the player agents of generation Nc and retracts them one by one.

V. Experiment Evaluation

We evaluate EVO-COGNISIM for evolutionary simulations by performing multiple experiments. We validated our approach by reproducing key experimental results from [13].

A. Experiments on the Gossip Model

1) First experimental setup:: A generation of 660 player agents is initialised, where every 5 players share the same strategic values for discrimination_threshold (a value chosen from -5.0 to 5.5 at intervals of 0.5) and gossip_weight (a value chosen from [1,1/2,1/4,1/8,1/1024,0]). Rounds, N and G are set to 660, 10 and 5 respectively, whereas R is varied from 0 to 25 at intervals of 5 for different experiments. As a result the λ $(= G/R)$ value can vary from 0 to 5. We set trust_criteria=everyone and gossip_type=fair with self_advertise=no for experiments related to first and third hypotheses. We also set gossip_type=fair with 264 self advertisers (self_advertise=no) for all the experiments related to the second and third hypotheses. 20 trials were performed for each experiment. The benefit and cost of cooperation are set to 10 and 1 respectively.

Hypothesis 1: If a population consists of all types of discriminators who gossip fairly and no one self-advertises, then intrinsic cooperators perform well on getting more gossiping opportunities. Figure 6 shows the proportion of all trials where cooperation emerged. As it can be seen, the society cooperated more (higher value/shown in blue) in trails with more gossiping encounters (higher λ) or where the value of T is higher (higher generation).

Hypothesis 2: If a population consists of all types of discriminators who gossip fairly and some players falsely self-advertise, then intrinsic cooperators perform well at the intermediate level of gossiping. Figure 7 shows that most of the trials, where a population had a higher level of gossiping, did not evolve into cooperation (no higher values/not blue) because false gossip has proliferated too much in the population, whereas cooperation (higher value/blue) emerged when there is an intermediate level of gossip (λ is 2 or 3) in the society.

Hypothesis 3: When all possible strategic values are uniformly distributed at the first generation, stern discriminators are more likely to evolve than more generous discriminators. All trials in the previous two experiments (of Figure 6 and 7) highlight that if a population evolved into a cooperative society, stern agents were almost always present at the end, whereas there is no specific generous cooperation strategy found that survives till the end.

2) Second experimental setup:: A bi-strategic generation[1] of 100 players is initialized with two strategies; always defect (discrimination_threshold=7) or stern cooperation discrimination_threshold=0. Different proportions of each

[1]Only two strategies are used by all players in the generation.

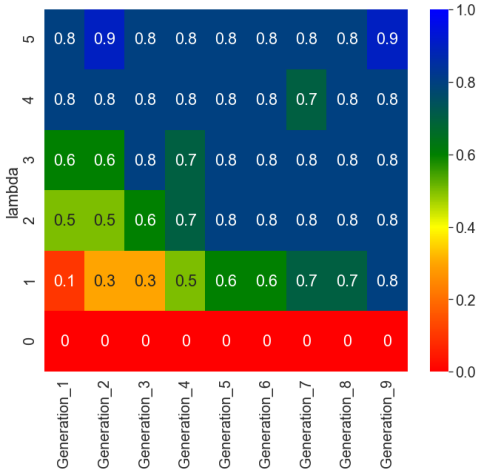


Fig. 6: Cooperation in Fair Gossiping Society with no self-advertisers

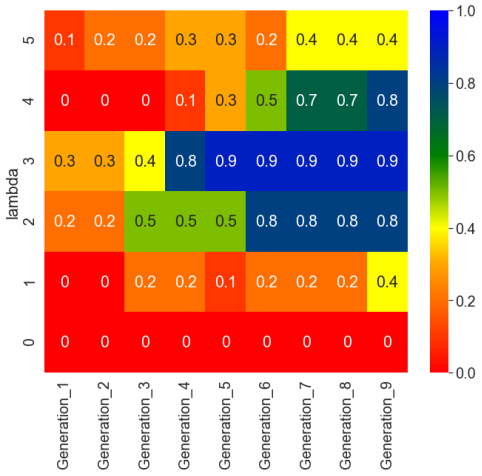


Fig. 7: Cooperation in Fair Gossiping Society with some self-advertisers

strategy are set for different trials. We set gossip_weight=1 and 0 for stern cooperators and defectors respectively. Round values vary from [50,100,150,200], whereas R is varied from 0 to 10 at intervals of 1 for different trials. We also set trust_criteria=everyone and gossip_type=fair, with self_advertise=no for stern cooperators and nogossip for defectors. N , G, benefit and cost are set as before.

Hypothesis 4: In a bi-strategic population of non-gossiping defectors and fair gossiping stern discriminators, less number of stern discriminators are required to evolve cooperation if more time or gossip opportunity is available. Different proportions of stern discriminator are tried with defectors to find out the expected minimum number of stern discriminators required for the fixation of this strategy and as Figure 8 shows, a lesser number is needed if T is high or more gossip encounters are available.

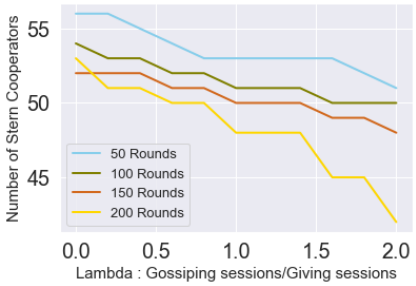
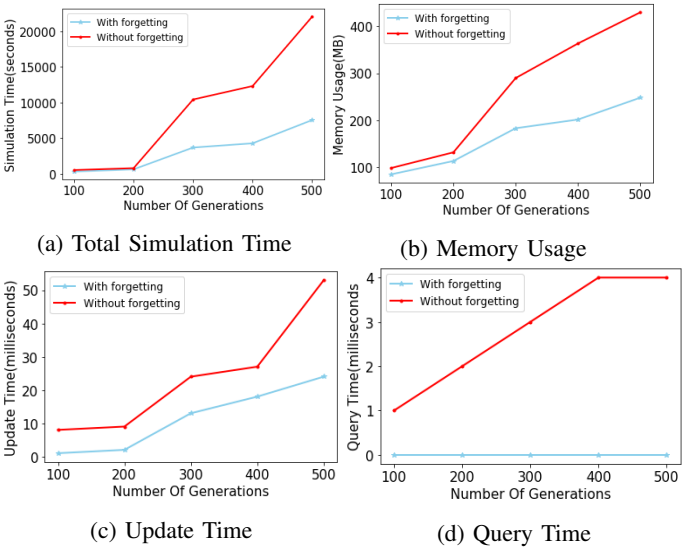


Fig. 8: Expected stern cooperators in a generation

B. *Performance Evaluation of* EVO-COGNISIM

The performance of the EVO-COGNISIM platform has been evaluated on an Intel(R) Xeon(R) CPU E5-2683 v4 with 2.10GHz and 98GB of RAM by varying the number of generations from 100 to 500. Other parameters such as Rounds G and R are set to 660, 5 and 5 respectively and a bi-strategic population of 20 agents is used. Figure 9a shows that the execution time of the experiment increases linearly with the number of generations. Also, the forgetting mechanism makes the simulation run efficiently whereas without forgetting, the performance deteriorates after 40 rounds. Similarly, as shown in Figure 9b the RAM usage also grows rapidly without the forgetting mechanism after approximately 40 rounds, so we therefore set the time for optimisation experimentally (within a generation) as the time equivalent to 40 rounds. Figures 9c and 9d show the update and query times with and without the forgetting mechanism. After 300 generations, circa 11880000 events have been processed. The update time is approximately 13ms and the query time is negligible (less than a millisecond). The simulation run time is 10436.785s in total and it uses 289.9MB RAM. Table II shows the summary of performance evaluation results.



(a) Total Simulation Time

(b) Memory Usage

(c) Update Time

(d) Query Time

Fig. 9: Performance Evaluation of EVO-COGNISIM.

TABLE II: Performance Evaluation of the EVO-COGNISIM platform. First column(×) in every criteria represents values without forgetting and the second column(✓) shows values recorded with forgetting.

N (20 players)	Total Time(sec)		Update Time(ms)		Query Time(ms)		RAM(MB)	
	×	✓	×	✓	×	✓	×	✓
100	540.23	330.4	8	1	1	0	84.5	98
200	802.889	634.45	9	2	2	0	113	131.5
300	10436.785	3694.843	24	13	3	0	182.9	289.9
400	12324.287	4289.561	27	18	4	0	201.45	363.6
500	22052.02	7548.842	53	24	4	0	248	430

VI. Conclusion and Future Work

We have proposed a systematic knowledge representation framework for evolutionary simulations using cognitive agents. The framework non-trivially extends previous work and allows an experimenter to test a population of such agents via generations to study how specific population behaviours evolve over time. The proposed framework is implemented in EVO-COGNISIM, a proof-of-concept prototype of our approach. To validate EVO-COGNISIM, we have reproduced successfully the results of existing simulation experiments based on a state-of-the-art model from the literature.

One open question that arises as part of developing EVO-COGNISIM is the following: Now that we have an explicit representation for developing evolutionary simulations, what is it that we need to provide, after an evolutionary simulation has ended, so that an experimenter can inspect post-hoc the behaviour of agents that produced the results? We endeavour to answer this question in our future work.

Acknowledgements

The first author expresses gratitude for the support received by Royal Holloway, University of London & the Higher Education Commission (HEC), Pakistan. The third author was funded by Leverhulme Trust, Research Grant LIP-2022-001.

References

[1] G. Kendall, X. Yao, and S. Y. Chong, *The Iterated Prisoners' Dilemma: 20 years on*. World Scientific, 2007, vol. 4.

[2] J. M. Smith, *Evolution and the Theory of Games*. Cambridge Univ. Press, 1982.

[3] M. A. Nowak and K. Sigmund, "Evolution of indirect reciprocity," *Nature*, vol. 437, no. 7063, pp. 1291–1298, 2005.

[4] H. Ohtsuki and Y. Iwasa, "The leading eight: social norms that can maintain cooperation by indirect reciprocity," *Journal of theoretical biology*, vol. 239, no. 4, pp. 435–444, 2006.

[5] V. A. Knight, O. Campbell, M. Harper, K. M. Langner, J. Campbell, T. Campbell, A. Carney, M. Chorley, C. Davidson-Pilon, K. Glass *et al.*, "An open framework for the reproducible study of the Iterated Prisoner's Dilemma," *Journal of Open Research Software*, vol. 4, no. 1, 2016.

[6] A. L. Bazzan, R. H. Bordini, and J. A. Campbell, "Evolution of agents with moral sentiments in an Iterated Prisoner's Dilemma exercise," in *Game theory and decision theory in agent-based systems*. Springer, 2002, pp. 43–64.

[7] C. Power, "A spatial agent-based model of N-person Prisoner's Dilemma cooperation in a socio-geographic community," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 1, p. 8, 2009.

[8] T. A. Han, L. M. Pereira, F. C. Santos, and T. Lenaerts, "Why is it so hard to say sorry? evolution of apology with commitments in the iterated prisoner's dilemma," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, 2013, pp. 177–183.

[9] S. Tisue and U. Wilensky, "Netlogo: Design and implementation of a multi-agent modeling environment," in *Proceedings of Agent*, vol. 2004, 2004, pp. 7–9.

[10] N. Collier, "Repast: An extensible framework for agent simulation," *The University of Chicago's Social Science Research*, vol. 36, p. 2003, 2003.

[11] A. Borshchev, S. Brailsford, L. Churilov, and B. Dangerfield, "Multi-method modelling: Anylogic," *Discrete-event simulation and system dynamics for management decision making*, pp. 248–279, 2014.

[12] M. Witkowski and K. Stathis, "A dialectic architecture for computational autonomy," in *Agents and Computational Autonomy*, M. Nickles, M. Rovatsos, and G. Weiss, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 261–273.

[13] M. Seki and M. Nakamaru, "A model for gossip-mediated evolution of altruism with various types of false information by speakers and assessment by listeners," *Journal of Theoretical Biology*, vol. 407, pp. 90–105, 2016.

[14] C. Adam and B. Gaudou, "BDI agents in social simulations: A survey," *The Knowledge Engineering Review*, vol. 31, no. 3, pp. 207–238, 2016.

[15] M. Grinberg and E. Todorov, "Cognitive agent based simulation platform for modeling large-scale multi-level social interactions with experimental games," *Information Content and Processing*, vol. 3, 2016.

[16] T. Deutsch, T. Zia, R. Lang, and H. Zeilinger, "A simulation platform for cognitive agents," in *2008 6th IEEE International Conference on Industrial Informatics*. IEEE, 2008, pp. 1086–1091.

[17] D. Singh, L. Padgham, and B. Logan, "Integrating BDI agents with agent-based simulation platforms," *Autonomous Agents and Multi-Agent Systems*, vol. 30, no. 6, pp. 1050–1071, 2016.

[18] A. Ricci, A. Croatti, R. H. Bordini, J. F. Hübner, and O. Boissier, "Exploiting simulation for MAS development and execution—the JaCaMo-Sim approach," in *EMAS'20*. Springer, 2020, pp. 42–60.

[19] O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with JaCaMo," *Science of Computer Programming*, vol. 78, no. 6, pp. 747–761, 2013.

[20] N. S. Shahid, D. O'Keeffe, and K. Stathis, "Game-theoretic simulations with cognitive agents," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 1300–1305.

[21] K. Stathis, "A game-based architecture for developing interactive components in computational logic," *Journal of Functional and Logic Programming*, vol. 5, no. 2000, p. 58, 2000.

[22] Y. Gong, S. Liu, and Y. Bai, "Reputation-based co-evolutionary model promotes cooperation in Prisoner's Dilemma game," *Physics Letters A*, vol. 384, no. 11, p. 126233, 2020.

[23] M. A. Nowak, "Five rules for the evolution of cooperation," *Science*, vol. 314, no. 5805, pp. 1560–1563, 2006.

[24] J. Hopkins, Ö. Kafalı, and K. Stathis, "Open game tournaments in STARLITE," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 1927–1928.

[25] A. Brogi and F. Turini, "Metalogic for knowledge representation," in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991*, J. F. Allen, R. Fikes, and E. Sandewall, Eds. Morgan Kaufmann, 1991, pp. 61–69.

[26] E. T. Mueller, "Event calculus," *Foundations of Artificial Intelligence*, vol. 3, pp. 671–708, 2008.

[27] L. Chittaro and A. Montanari1, "Efficient temporal reasoning in the cached event calculus," *Comput. Intell.*, vol. 12(3), pp. 359–382, 1996.

[28] A. Artikis, M. J. Sergot, and J. V. Pitt, "Specifying norm-governed computational societies," *ACM Trans. Comput. Log.*, vol. 10, no. 1, pp. 1:1–1:42, 2009.

[29] O. Kafali, A. Romero, and K. Stathis, "Agent-oriented activity recognition in the event calculus: An application for diabetic patients," *Computational Intelligence*, vol. 33, no. 4, pp. 899–925, 2017.

[30] N. Nilsson, "Teleo-reactive programs for agent control," *Journal of artificial intelligence research*, vol. 1, pp. 139–158, 1993.

[31] A. S. Rao, M. P. Georgeff *et al.*, "BDI agents: from theory to practice." in *ICMAS*, vol. 95, 1995, pp. 312–319.

[32] D. G. Rand and M. A. Nowak, "Human cooperation," *Trends in cognitive sciences*, vol. 17, no. 8, pp. 413–425, 2013.