Evolving effective behaviors to interact with tag-based populations

Osman Yucel, Chad Crawford & Sandip Sen [a]
[a] *The University of Tulsa*;

This is an Author's Accepted Manuscript of an article published in Connection Science vol. 27, issue 3, 2015 copyright Taylor & Francis, available online at:

http://www.tandfonline.com/doi/full/10.1080/09540091.2015.1031467

Abstract

Tags and other characteristic, externally perceptible features that are consistent among groups of animals or humans can be used by others to determine appropriate response strategies in societies. This usage of tags can be extended to artificial environments, where agents can significantly reduce cognitive effort spent on appropriate strategy choice and behavior selection by reusing strategies for interacting with new partners based on their tags. Strategy selection mechanisms developed based on this idea have successfully evolved stable cooperation in games such as the Prisoner's Dilemma game but relies upon payoff sharing and matching methods that limit the applicability of the tag framework. Our goal is to develop a general classification and behavior selection approach based on the tag framework. We propose and evaluate alternative tag matching and adaptation schemes for a new, incoming individual to select appropriate behavior against any population member of an existing, stable society. Our proposed approach allow agents to evolve both the optimal tag for the environment as well as appropriate strategies for existing agent groups. We show that these mechanisms will allow for robust selection of optimal strategies by agents entering a stable society and analyze the various environments where this approach is effective.

1 Introduction

Agents in open multi-agent environments must be able to quickly adapt to new environments and use past, relevant experience to react to new scenarios and choose effective interaction policies with new partners. As in human societies, agents in artificial societies also come with external features that may, in a large majority of cases, suggest social groups they belong to and hence present at least a coarse-level view of their behavioral characteristics, biases and preferences.

For example, businesses often employ the idea of signalling; that is, to make explicit decisions about the business's appearance (such as the area it is located in, the exterior design of the building, etc) which conveys well-understood signals to potential clients about the reliability of the business. Similarly, consider the scenario of interviewers and interviewees at a job fair. In that case, the groups of interviewers and interviewees are two interacting populations. Interviewees are looking for jobs of interest to them: to be viewed as attractive and dependable to the target pool of interviewers, interviewees can adapt their appearance, both in person and on paper, and hope to get on the short list of those interviewers to those they would prefer to interact with. In addition, it might require slightly different presentation and interaction style to impress different interviewers. The appearance and posturing of different interviewers can also suggest whether, for example, a more formal or a more casual interaction modality is suitable for the interviewee to maximize their chances of making the next level of interviews.

External features or tags of an agent can then be utilized to both reduce cognitive effort in the strategy selection and to provide a means to classify agent experience. While these generalizations are not always

socially optimal, as in social stereotypes, they do provide a key tool for cognition that allows a pragmatic management of the complexity of life [7].

Prior tag mechanisms have shown that cooperation is possible within the tags framework [9]. These mechanisms however are not particularly suited for open environments. Limitations on agent interactions and payoff distribution may incite cooperation, however they restrict the conditions under which agents can learn within the environment. Another shortcoming is that although these mechanisms allow choosing of which partners in the environment to interact with, it does not enable learning appropriate strategies to interact with each and every member. In this paper, we are interested in developing tag-based mechanisms that allow newcomers to a society to adapt both their tag (external appearance) and their matching strategy (who they want to cooperate/not cooperate against). Therefore we will analyze the interactions of agents in open environment where a group of new agents may migrate to a population of stable agents. The goal of these newcomers, then will be to quickly learn the habitual preferences and behavioral biases of the existing population so that they are able to effectively and gainfully interact with them.

To facilitate speedy learning, we assume two basic features in the environment: (a) the stable population has some latent behavioral biases that are at least indirectly reflected in their external, observable features[1], and (b) members of the newly arriving group can independently interact with the members of the stable population and that their relative success or the reward from this interaction are shared with other newcomers, who can then choose to alter their interaction strategies to align with the more successful of their peers.

In this paper, we adopt a binary interaction outcome scenario, where in each interaction, each party can choose to cooperate (C) or not ($\overline{C}$) with the opponent. We assume that each agent, both in the incoming and existing populations, have an external, visible feature set or tag and an internal hidden matching mechanism that determines whom that agent will cooperate with. Matching an agent implies cooperating with that agent, while not matching implies not cooperating. We investigate various scenarios of varying payoffs for matching and not matching, corresponding to different real world situations.

We evaluate several combinations of evolutionary and classical learning approaches to see if the incoming population can successfully develop effective interaction policies from repeated interactions. Learning of such policies is key for a successful assimilation of the incoming population into the stable one. Our results show varying degree of success with different representation scheme and dictated by the correlation between the external features and the intrinsic behavioral traits of the stable population.

In our study we also tried to create scenarios where learning agents are not best served by pure strategies like "always cooperate" or "always not cooperate", but are better served by conditional policies like "cooperate with cooperators and do not cooperate against non-cooperators". We propose a method to adjust the payoff matrix of the game such that the evolving agents are incentivized to play conditionally depending on its opponent's expected behavior rather than committing to defecting or cooperating unilaterally.

2 Related Works

Previous work on tag-based mechanisms of game-playing and cooperation have focused on developing evolutionary schemes to learn optimal actions based on tag relationships to strategies [11, 9, 10]. The proposed mechanisms are generally developed to be played in iterated stage games of Prisoner's Dilemma or the Anti-Cooperation game.

Such mechanisms invoke population models of learning, in which agents may develop identities via mimicry [10]. McDonald and Sen observe that the particular game being played by the agents has a significant effect on strategy. Agents are more likely to coordinate in scenarios where effective coordination among agents is achieved by mimicry, i.e., agents can choose the same behavior to achieve coordination and hence homogeneous agents can achieve optimal outcome. However, existent tag-based mechanisms are less effective in scenarios where agents play games in which coordinating to attain some Nash or Pareto-optimal equilibrium requires diverse strategies from each agent, i.e., heterogeneous agents outperform homogeneous agents.

[1]Note that we do not require the knowledge of that mapping or, for that matter, any guidance to the nature of the function(s) mapping from external appearances to intrinsic behavioral traits.

Such models provide effective mechanisms which consistently produce social rationality and/or Nash equilibria [5]. The Principle of Social Rationality, as proposed by Hogg and Jennings [6], is: *If a socially rational agent can perform an action whose joint benefit is greater than its joint loss, then it may select that action.* The goal of Hales' evolutionary model is to produce emergent dynamics which result in a system of socially rational agents. We are interested in producing socially rational agents, but stubborn agents in our system that refuse to cooperate force the learning agents in the system to not cooperate with others as well. As a result, the problem is one of classifying stubborn agents based on their outward attributes and cooperating with those that maximizes the number of cooperations, followed by non-cooperations.

Previous work on tag mechanisms also focuses on changing tag representation over the matching representation [4]. Our model is interested only in tags composed of bit strings, and we suggest four novel improvements to the matching mechanism that increase performance in the cooperation game.

Stereotyping in trust systems shares some very similar qualities with tag-based mechanisms, and similar techniques have been used to construct such stereotypes [1, 12]. These approaches use a machine learning classifier to separate agents into groups, with the intent of making generalizations about groups to determine the trustworthiness of agents inside such groups. Similar applications of classification in multi-agent frameworks focus on subjects such as text classification [3]. We are interested in developing similar classification schemes to map visible feature attributes into strategies.

Unlike the aforementioned papers, our approach uses an evolutionary scheme to generate a classifier. While previous papers have gathered information based on single interactions, we are interested in population-level interactions which is more realistic in social situations where interactions are numerous and short. The evolutionary mechanism we introduce includes sharing of tag information among the evolving population as well as strategic information. Finally, our agents play a different simultaneous game in which cooperation from both players is encouraged, and otherwise the preferred action is to not cooperate. This game models many realistic situations, in which agents are motivated to cooperate with others but because of preconceived negative biases, there are those who will not cooperate with certain groups. In this case, players are motivated to not attempt cooperation with these agents as well, as some sort of negative retribution. In some ways, players in this game are learning to be socially rational, but in contrast to previous works there exists a group of irrational agents (known as "static agents") which act irrationally in all respects.

3 Simulation Setup

In the current framework, agents have two properties: a tag and a matching mechanism. Tags represent the external identity of the agent; it is independent of the agent's behavior, but other agents can only see the tag of that agent. A tag is a bit string of length TL; an agent a has a tag T_a and $T_a(i)$ represents the value of ith tag bit. The use of tags in a population of agents has been shown to induce coordination [11].

Matching mechanisms are functions which gets the tag of the agent they are playing against, and return the decision to cooperate or not. Details on the matching mechanism and specific implementations are listed in section 4.

The population is partitioned in two: a static population and a learning population. Agents in the static population have fixed matching mechanisms and fixed tags. Those agents in the learning population may modify both their tag and matching mechanism. The goal of these modifications is to maximize their payoff in repeated games with random interactions with the static population.The static agents play C or $\overline{C}$ according to their matching mechanism and the opponent agent's tag. The goal of evolving agents is to separately and simultaneously adjust their tag, to maximize cooperation with the static population, and develop a strategy that effectively discriminates cooperators from non-cooperators. In more details, the learning task can be stated as follows: given the collection of past observations of tags and behaviors of agents from the static population, choose a tag and matching mechanism that will match more tags of those opponents whose matching mechanism match this agent's tag while avoiding matching tags of those agents whose matching mechanism does not match this agent's tag. Another way of characterizing the desired behavior is that an ideal tag-tag matching pair is one that will maximize cooperating with those agents who cooperate with us [2] and vice versa.

[2]In our work, an agent "cooperates" with an opponent if its tag matching mechanism matches the tag of the opponent.

Note that the level of cooperation obtainable by the incoming population depends on the nature of the static population. In particular, it is possible to learn to elicit more cooperation if static population tag matching schemes are more match-friendly, i.e., match larger percentage of the tag space.

On another note, the effectiveness of the tag-based approach critically depends on the correlation between external appearance and internal behaviors. Newcomers will face a more difficult coordination learning problem if external appearances do not correlate with agent behaviors, i.e., when agents of the static population with similar tags behave very differently. We run experiments with different grouping schemes (discussed in sections 6.1 and 6.2) to simulate both relatively less and more challenging coordination learning scenarios. Agents play a modified version of the coordination game where one mutual strategy is favored over the other; the exact payoffs are discussed in section 5.1 in further detail.

4 Matching Methods

Matching mechanisms are deterministic functions on tags that determine an agent's strategy; that is, matching mechanism $M : T \rightarrow \{C, \overline{C}\}$ maps from the tag space (all possible bitstrings of length TL) to some strategy in the available set of two strategies. The matching methods presented consider the only "visible" features of their opponents – their tags – and do not consider any other information, e.g., their individual identity, past performance, etc. In other words, agents have only information on themselves and the tag they are interacting with when choosing a strategy in the game. Agents in the simulation cannot observe the inner workings of matching mechanism of their opponents, i.e., it is the hidden strategy of the opponent. The learning population will also use the matching mechanisms discussed below, but can modify hidden, internal information to adjust its behavior.

In this study four different matching mechanisms are used:

Ternary Matching Strings: Agents use a ternary matching string MS composed of values in $\{0, 1, *\}$, where $*$ corresponds to a don't care, and of length equal to the length of the agent tags. For an agent a with a ternary matching string to cooperate with agent b,

$$M_{MS}(a, b) : \forall_{i \in \{1,2,\ldots,TS\}}(MS_a(i) = T_b(i)) \vee (MS_a(i) = *),$$

where $MS_a(i)$ corresponds to value of the ith position of a's matching string and $T_b(i)$ is the value of the ith position of the tag of agent b. So, a match occurs if, for every position, either the matching string contains the same value as the other agent's tag or contains a don't care symbol. Such a matching approach makes sense when agents decide on which tag positions or features are important, and judge others based on whether they meet the criteria on these salient features.

Hamming Distance: Agents decide to cooperate based on the similarity of their own tags with that of others. In Hamming matches, the total number of bit differences between the tags of two agents a and b must be at least H_{min} but no more than H_{max}. In other words, agents should prefer those similar to them, but prefer prefer their partners to be at least somewhat different:

$$M_H(a, b) : H_{min} \leq |\{i \in \mathbb{Z} : TS_a(i) \neq TS_b(i)\}| \leq H_{max}.$$

So matching is based only on the tags of the two interacting agents.

Decision Tree: In this matching mechanism, an agent uses a decision tree to classify the opponent's tag and decide whether to cooperate or not. Nodes in the tree correspond to tests on tag features and results in one of two outcomes. Decision trees can compute arbitrary functions on boolean features and hence the range of behaviors that can be represented is greater than that can be represented with the matching mechanisms using Hamming distance or ternary strings.

There are two benchmark agents trained for comparison to the learning population's performance:

Optimal Learning : Chooses the best possible tag and matching scheme to maximize the agent's payoff. The matching scheme is determined by finding the best strategy to play against each tag in the static population. "Best strategy" is defined as that which maximizes payoff, assuming that static

M_i :	0	1	*	1
$M_{i'}$:	0	1	0	1
T_j :	0	1	1	1

Figure 1: An example of a ternary matching string and tag interaction. Player i will cooperate with j, but i' does not cooperate with j due to a mismatch in values at index 3.

T_i :	0	1	0	1
T_j :	1	0	1	0
T_k :	1	1	0	1

Figure 2: The Hamming distance between T_i and T_j is 4, and between T_i and T_k is 1. For cooperation, agents need to be similar, but have some differences.

agents are equally likely to interact with a learning agent. The best tag is found by enumerating over all possible tags and best matching schemes, and choosing that with the best performance. This method is enumerative with complexity $O(2^{TS})$.Note that the optimal agents are capable of perfectly discriminating between the static agents only if those with identical tags have identical behaviors, which is not the case in general. The results of the optimal learners show us what is the best that could be done against the current static population. So it should be viewed as a quality measure of the static population instead of a competitor agent type to our evolving agents[3]. These agents would need to memorize how to behave against any tag, which would require a memory of history while our agents only need to know their evolved matching mechanism, and try playing against all agents to find the best tag and then play again to decide the necessary behavior against them ,which is extremely time consuming compared to an evolutionary scheme. We are trying to enable the agents to get the best outcome possible without having the complete information of the static population, which would make optimal learning impossible.

Best Ternary String : Agents with ternary matching strings will underperform the optimal learning agent as the representative power of ternary strings are lower than the optimal learning agents. The best ternary string for a given tag is found by enumerating over all possible ternary strings with complexity $O(3^{TS})$. Finding the best possible tag given each corresponding best ternary match requires enumerating over all tags which is $O(2^{TS})$.; overall, we multiply these numbers to enumerate both tag and ternary match together, which is $O(6^{TS})$ time complexity. Similar to the optimal learner, the results of best ternary are the best that could be done against the given static population using ternary matching strings. These results are the best results that our agents could achieve. Nonetheless, finding the exact best pair of tag and matching strings would require the complete knowledge of the static population. The evolving agents, however, interact with only a limited sampling of the population.

We have performed a series of experiments with different population configurations using different matching mechanisms. In a given configuration, all static agents use the same matching mechanism, which can be different from the matching mechanism used by all members of the incoming population. All but the `Optimal Learning/Best Ternary String` matching methods have been used with the static population members (this is because the learning classifier will change the existing population from a static to a dynamic one). Similarly, all but the `Hamming Distance` and `Decision tree` matching method has been used with the incoming population members. This is because, as stated, the Hamming distance function is a fixed function and does not contain a learning opportunity (though there is a possibility for learning H_{min} and H_{max}, which we have not explored in this paper). The massive state space of the decision tree also makes it hard for the learning population to develop any significant matching scheme.

[3]The optimal learning approach is a brute force approach which makes it infeasible for large populations and large tag lengths

5 Evolving the new population

The members of the incoming population need to adapt their behavior and external appearance, their tags, so as to realize maximum utility from interaction with all the members of the static population. We use an evolutionary framework, where each newcomer interacts with each existing agent. Their individual experiences can be used, if desired, to adapt their personal strategy from personal interaction histories with the static population, e.g., what is done when the Optimal Learning or Best Ternary match is used, Unlike these optimal learning methods, the evolutionary framework allows for collaboration between agents in the population to learn the behavior of the static population. As this is an evolutionary mechanism, the agents do not keep track of all the interactions they have with each static agent. The only information they have is the utility they receive at the end of a generation. They use this information to decide to keep their current behavior or adopt some other agent's behavior.

In the following we present the evolutionary process used to learn the tags and strategies of the incoming population.

Algorithm 1 Evolutionary Algorithm

```
for g generations do
    for each agent, A_E, in evolving population,P_E do
        for each agent, A_S, in static population, P_S do
            if A_E.matches(A_S.tag) then
                if A_S cooperates then
                    A_E gets P_CC;
                else
                    A_E gets P_C̄C;
            else
                if A_S doesnotcooperate then
                    A_E gets P_CC̄;
                else
                    A_E gets P_C̄C̄;
        calculate the fitness, average payoff, of A_E;
    Create new empty Population P_N
    while P_N is not full do
        Choose agents a, b ∈ P_E ∋ a ≠ b
        if rand > P_Crossover then
            c ← offspring created with uniform bit crossover
        else
            c ← a copy of a
        Mutate c with probability P_Mutation;
        Add c to P_N
    Add the best Agent in P_E to P_N;
    P_E = P_N
```

For every generation, the population is able to reproduce both asexually and sexually; new agents will be either copied from the previous generation, or a new agent which is produced via crossover of two agents from the previous generation. Since the evolutionary learning population uses only ternary matching, the crossover mechanism used is simply uniform.

The fitness of an agent in the evolving population is calculated by the cumulative payoff of games played against the entire static population. The evolutionary algorithm selects 2 agents from the population to decide one parent.The decision is made by selecting the agent with the highest fitness value of the two with the probability of $P_{Selection}$.This means the better agent will be selected with $P_{Selection}$ and the worse one still has the chance to be selected with the probability of $1 - P_{Selection}$. When two parents are selected, with probability $P_{Crossover}$ two agents tag and match are used to generate a new offspring using uniform

crossover. This means with $1 - P_{Crossover}$ probability, the crossover will not occur and the first parent is directly transferred to the next generation.

After selection, every agent will be subject to a mutation stage given a parameter mutation probability μ.

The mutation operation used in the evolutionary process is dependent on the type of matching mechanism used in the experiment. The Hamming distance method is not used in the evolutionary population, so no mutation operator is used for that matching mechanism. For ternary Matching String (MS) mechanism the mutation function traverses each bit of the MS and replaces it with one of 0,1 or * with the probability of μ. Mutation is not applied to the matching function generated by the intelligent classifier.

When the new population is, almost, ready the best agent from the previous population is copied directly to the next generation to achieve elitism.

5.1 Payoff Matrix Adjustments

In most of the cooperation problems, pure strategies, which are very easy for the agents to come up with, create good enough utilities so the agents become unwilling to move from that strategy and search for better strategies. Drężewski shows that these strategies would cause the agents to get stuck at a local maximum instead of searching for a global one [2]. In most real life scenarios, people rarely choose to use pure strategies (interact with or ignore everybody), and instead adopt selective strategies.

Research on Commodity Theory has shown that the scarcity of some item is inversely proportional to its value [8]. Since cooperation and non-cooperation are some commodity, the same principle may impact the development of stereotypes. For example, an interviewee may attempt interviews with more interviewers if hiring is low; however, when everyone is hiring, the interviewee would desire to not interview with those that may not need them. To that end, the payoff matrix is adjusted to take into account the cost of heterogenous outcomes (where the learning and static player choose different strategies) given the likelihood of cooperation for a static population: if the static population is likely to cooperate, then the cost of cooperating with non-cooperators increases; conversely, if the static population is not likely to cooperate, the cost of not cooperating with cooperators increases. By calculating an 'unstabilized point', we want to achieve a society which will be willing to move away from pure strategies. That way the agents will be more likely to find the mixed strategies, which will achieve better utilities.

As mentioned above, each interaction between a newcomer and a current member of the static population corresponds to a stage game. The payoff matrix of the game incentivizes the evolving strategies and matching mechanisms of the incoming population. We now outline our design of the payoff matrix that incentivizes the emergence of conditional matching strategies, which respond to the matching behavior of the opponents, rather than "always cooperate" or "always defect" matching behavior. For any random tag and a matching mechanism, the probability of matching is affected by the matching mechanism used and the parameters of the system. Δ signifies the difference in payoff of pure cooperation (that is, given a static population, the learning agent chooses to cooperate in all cases) and pure non-cooperation. We then estimate the value of Δ for a given system of static agents:

$$E[\Delta] = k(P_{CC} - P_{\overline{C}C}) + (1-k)(P_{C\overline{C}} - P_{\overline{C}\overline{C}}) \tag{1}$$

where k is the probability that a member of a static population will cooperate given a random tag. k is affected by the matching mechanism used by a static agent: the tag matching mechanism, for example, has a very low k since one string yields very few cooperation across the set of possible tags.

There are three possible configurations of Δ which will radically effect the dynamics of the evolutionary learning system:

1. When $\Delta >> 0$, (is significantly greater than 0) a random learning population will be rewarded for the few agents they cooperate with; as such, learning populations will converge towards a total cooperate strategy;

2. When $\Delta << 0$ (is significantly less than 0) the learning population is rewarded for not cooperating, and will instead converge towards total non-cooperation;

	C	$\overline{C}$
C	4	α
$\overline{C}$	$P_{CC} - \frac{1-k}{k}(P_{\overline{CC}} - \alpha)$	2

Table 1: Payoff matrix for the evolving population (row player) against the static agent population (column player) for $\alpha = 1$ and probability k.

3. The special case of $\Delta \approx 0$ represents when the payoff matrix does not incentivize the learning population to adopt either total cooperation or defection; then any naive mixed strategy that ignores tags would yield approximately the same payoff; agents then may be led to correctly classify the static population.

To increase payoff, learning agents must cooperate with cooperative static agents and correspondingly not cooperate with non-cooperative agents. Since the trivial solutions for $\Delta >> 0$ and $\Delta << 0$ may be easily developed, we further examine the case of $\Delta = 0$, and the payoff matrix in Table 1 is the solution to Equation 1, by appropriately choosing the payoff values for cooperation and non-cooperation.

Finally, we calculate k for every matching mechanism used for the static population. Consider k_{TS}, the probability of matching a random tag given a random ternary matching string:

$$k_{TS} = P_m^{TL}, \tag{2}$$

where P_m is the probability of matching one bit. For a ternary string this value is $\frac{2}{3}$ since, the possible match values $\{0, 1, *\}$ are equally probable and for any bit in $\{0, 1\}$ will match one item in $\{0, 1\}$ and will definitely match *.

The corresponding probability for decision tree matching, k_{DT} is

$$k_{DT} = 0.5. \tag{3}$$

Consider the decision tree D in the set of all possible decision trees for a given domain of tags. Then, a complement decision tree D' may be constructed by switching the binary classification of every leaf node on the tree. So D and D' have opposite classifications, and this transformation is unique for D and D'. Since every decision tree has such a complement, any decision tree that may cooperate with a given tag will have another tree that does not cooperate. Therefore, the probability of cooperation with a randomly generated decision tree for any tag is 0.5.

Finally, we calculate the probability of matching for Hamming distance, k_{HD}:

$$k_{HD} = \sum_{i=H_{min}}^{H_{max}} P_m^{TL-i} * (1 - P_m)^i * \binom{TL}{i} \tag{4}$$

When the agents in the static population uses Hamming distance as their matching mechanism, the probability of a random generated tag matching their mechanism can be calculated as in Equation 4. P_m is the probability of matching a bit as it was in the definition of k_{TS}. H_{min} is the minimum Hamming Distance at which the agent will cooperate, and H_{max} is the corresponding maximum.

6 Generating Realistic Static Population Structures

The main idea behind using stereotypes for behavior adjustment on static population, is that there exists a relationship between external features and behaviors. Even if the behavior is not a direct function of the external features, it is expected that the agents with similar external features should behave similarly.

For such an assumption to be valid, the agent populations in the static society in our experiment needed to exhibit some underlying regularity. One way of creating such regularity was to create groups of individuals from several stereotypical individuals chosen as seeds at the start of the population generation process. We decided to implement two separate grouping approaches for the static population generation: uniform and scale free grouping.

6.1 Uniform Grouping

Uniform grouping approach creates a number of agents to represent selected number of groups(N_{Groups}). This approach assumes there are approximately equal sized groups in static populations. Then the static population is filled with agents which are copies of a randomly selected group. The probability of an agent to belong to any group is equally likely in this approach. The algorithm to generate static population with uniform grouping is presented in Algorithm 2.

Algorithm 2 Generating Static Population Using Uniform Groups

Create N_{Groups} and representing agents A_n
while P_{static} *is not full* **do**
    Pick one of the representing A_n
    Make A_{new} a copy of A_n
    Mutate A_{new} with $P_{mutation}$ probability
    Add A_{new} to P_{static}

6.2 Scale-Free Grouping

In most of the real-life populations, agents are more likely to join larger groups thus making those groups even larger. Scale-free grouping approach creates a number of agents to represent selected number of groups(N_{Groups}). N_{Groups} agents are randomly created and added to the static population (P_{static}). The algorithm for generating static population with scale-free grouping is presented in Algorithm 3.

Algorithm 3 Generating Static Population Using Scale-Free Groups

Create N_{Groups} agents and add them to P_{static}
while P_{static} *is not full* **do**
    Pick one of the existing agents(A_n) in P_{static}
    Make A_{new} a copy of A_n
    Mutate A_{new} with $P_{mutation}$ probability
    Add A_{new} to P_{static}

The distribution of the agents into the groups, using these two grouping approaches, can be seen in Figure 3. The major distinction between uniform and scale-free grouping is that scale-free grouping has a larger variance in the tags and matching mechanisms. This is because the agents in uniform grouping will base their identity of off an ideal leader of each group. On the other hand, agents in the scale free approach may copy their identity from any member of the group, which may lead the resulting agent to be more dissimilar than the ideal group leader. A learning scheme in the scale-free approach will need to determine when it is appropriate to attempt to cooperate with the largest group, or try to coordinate with multiple smaller groups.

7 Experimental Results

We now present results of our experiments with different population configurations, using different combinations of matching mechanisms.

Unless otherwise specified, simulations are run until convergence (identical tags and matching strings in the learning population) or until the system exceeds 10000 generations (in which the system likely never converges). Our learning population has 100 incoming or new learning agents, to be tested against a population of 1000 static agents. Mutation rates are different for the tag and matching mechanism. The tag mutation rate is at $\frac{1}{16}$, and ternary match mutation is $\frac{1}{1000TS|N_L|}$. The match mutation rate needs to be much lower than tag mutation rate since small adjustments in the ternary match string can have drastic effects on the

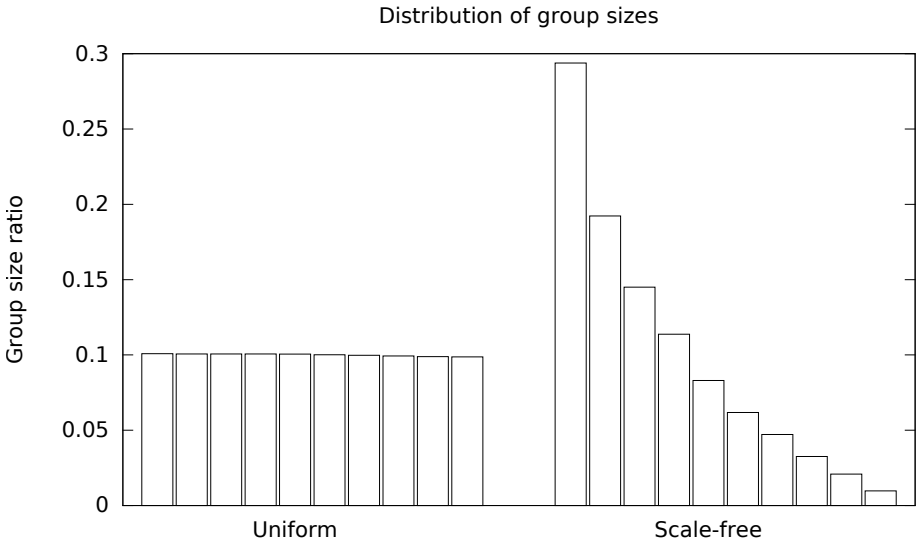

Figure 3: The ratio of 1000 agents in groups when they are distributed into 10 groups.

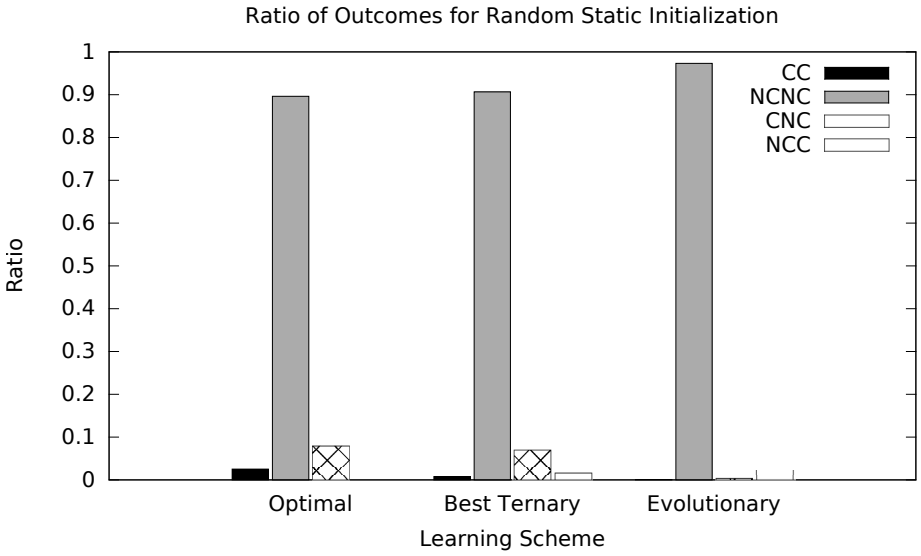

Figure 4: Ratio of each game outcome for the random grouping scheme with 100 learning agents against 1000 static agents with ternary matching strings, tag size of 8, and averaged over 4 trials. Solid bars represent desired outcomes ($CC$and $\overline{CC}$). The labels on the figure represent strategies played by the learning agent and the static agent respectively (NC is $\overline{C}$).

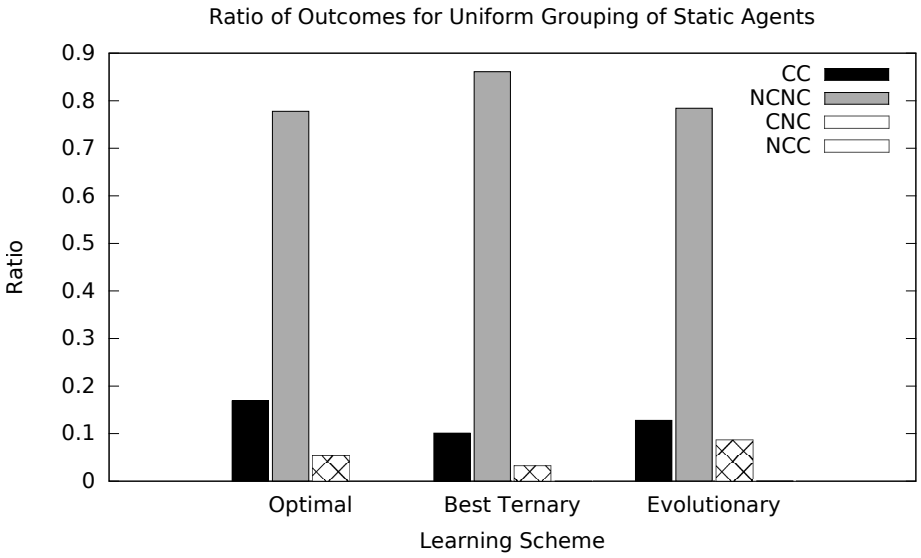

Figure 5: Ratio of each game outcome for the uniform grouping scheme with 100 learning agents against 1000 static agents with ternary matching strings, 10 groups, tag size of 8, and averaged over 4 trials.

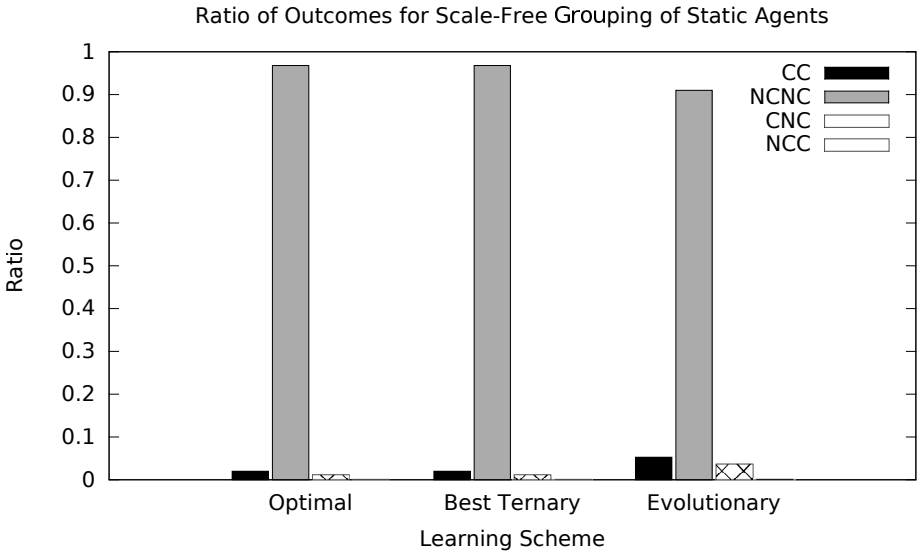

Figure 6: Ratio of each game outcome for the scale-free grouping scheme with 100 learning agents against 1000 static agents with ternary matching strings, 10 groups, tag size of 8, and averaged over 4 trials.

entire population (especially when tag size is larger). The tag length of all agents is 8, and for Hamming matches the parameters have the value $H_{min} = 2$ and $H_{max} = 4$. The selection parameter for tournament selection is set to $P_{Selection} = 0.8$ and the crossover probability is 0.8.

During the simulation, each agent in the evolutionary learning population interacts with a random sample of 10% of the static population. Their overall performance is determined by how the resulting agents interact with the entire static population.

Results in Figure 5 and Figure 6 indicate that even with few groups and smaller tags, the optimal strategy cannot yield total coordination. This is especially true for the scale-free group case, in which only 2% coordinated cooperation could be achieved.

The evolutionary learning mechanism performs comparably with the others in group-related scenarios. In many cases it has been able to find the best group to cooperate with, and can correctly adjust its tag and matching mechanism to cooperate with that group in return. One challenge faced by the evolutionary mechanism is that the large majority of tag and match combinations result in always defecting, and so the agents need to focus their search on cooperative combinations. To encourage this outcome, learning agents are initialized with matching strings composed entirely of *: that is, these agents begin by cooperating with every agent in the other population. In this way, evolving agents are encouraged to find a good tag (that is, a tag that yields high cooperation) before becoming more discriminatory.

Surprisingly, the optimal learning agent for random (in Figure 4) and scale-free (in Figure 6) populations perform similarly; both yield low degrees of coordinated cooperation. The fact that groups have more diversity may make them more similar to random populations when the average intragroup distance is large enough; however, one notable difference is that the best ternary and best evolutionary agents have higher degrees of coordinated cooperation. Ternary strings are made to identify ideal agents to cooperate with; as such, the challenge of a ternary string is simply to find a subset of similar tags that cooperate with the given agent's tag. The payoff for the outcome $\overline{C}C$is also incredibly low for static ternary populations, so agents with ternary matching strings desire to expand this subset as far as possible to cover all potential cooperators.

7.1 Random Population Configurations for Static Ternary Matching

When agents are grouped, some inferences can be made about what type of behavior can be expected when observing a certain tag. In the random population configuration, this relationship does not exist. Results for evolving a Ternary population against a random static population is shown in the first row of Table 2. The best learning agent in this scenario could barely find any mutual cooperation; this shows the necessity for grouping configurations of the population.

7.2 Alternative Matching Mechanisms for the Static Population

Table 2 shows the performance of the evolutionary mechanism against each matching mechanism used for the static population grouped using the Uniform scheme.

When we tried our evolving agents against 3 types of static agents we get the results seen in Table 2. The first thing about the results which attract attention is the agents were getting perfect results against the static populations using hamming distance. While this looks quite impressive, any random agent was already expected to get 60% cooperation. This quality makes it a trivial problem to solve when we created the static population by grouping them.

Against the decision tree population, the evolving agents were not very successful and they could not really improve the prior probability of 50% getting cooperation. Since the decision tree matching scheme is not similarity-based, small changes in tag composition may lead to significant, and difficult to learn, changes in the behavior of a Decision tree matching mechanism. In essence, the learning problem is significantly more difficult in this case as decision trees can represent a much larger complexity class of functions compared to ternary matching, which for example cannot represent functions with disjunctions.

Ternary string-based matching was the most discriminatory; equation 2 indicates that a random agent can expect to cooperate with only 4% of the population. This pattern was accounted for in the adjusted payoff matrix. So the learning populations were unwilling to cooperate, and only did cooperate with a small

number of static agents. Ternary strings are more discriminatory, so by their nature learning ternary strings will have high rates of non-cooperation.

Hamming distance was the least discriminatory of all matching mechanisms used in the static population. The evolutionary mechanism was able to cooperate with the entire static population; this is because the grouping scheme made their behavior incredibly simple to learn. Unlike other matching mechanisms, it is the tag that determines the behavior of an agent using the Hamming distance mechanism; as such, agents with identical tags would share identical behaviors. Realistic matching mechanisms, on the other hand, take into account both internal and external information when making judgments of an individual.

Static Match	CC	$\overline{CC}$	$C\overline{C}$	$\overline{C}C$	Avg. Fitness
Ternary	0.48	91.15	5.93	2.45	0.07
Hamming	100.00	0.00	0.00	0.00	4.00
Decision	38.15	16.28	30.87	14.7	1.69

Table 2: The results for running the evolutionary mechanism and introducing the evolving population, who use ternary strings for matching, against a static population of who use varying mechanisms.

7.3 Group numbers and the Optimal Strategy

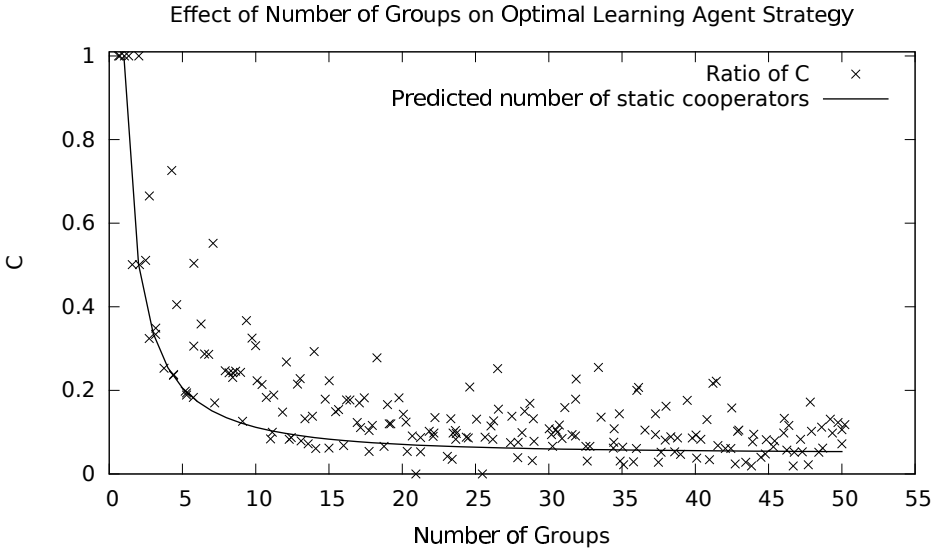

Figure 7: Empirical data of the rate of the best learning agent playing C in a simulated game against 1000 uniformly grouped static agents with $TS = 8$. The line is the estimated proportion of static agents that may cooperate with a given agent, assuming that the agent already cooperates with one group.

The number of groups in the static population has a significant effect on the optimal strategy that an agent can have. Figure 7 shows the rate of the best optimal learning agent playing C against the static population with uniform grouping. There is clearly an exponential decreasing trend as the number of groups increase; on average, the best learning agent can only cooperate with half of the static population when there are two groups. By the time there are even 20 or more groups, the best learning agent is playing 20% to 0% C against the static population.

The reason why the number of groups have such a strong impact on the optimal strategy is because groups have a low likelihood of overlap with whom they cooperate with. For a limited case, suppose a static population has g groups where each group has agents share identical tag and ternary matching mechanism, and suppose that the optimal learning agent can always achieve total mutual outcomes. Clearly, the best strategy can match at least one group by changing its tag and matching mechanism accordingly. However, matching any other group depends on how many groups are willing to cooperate with the learning

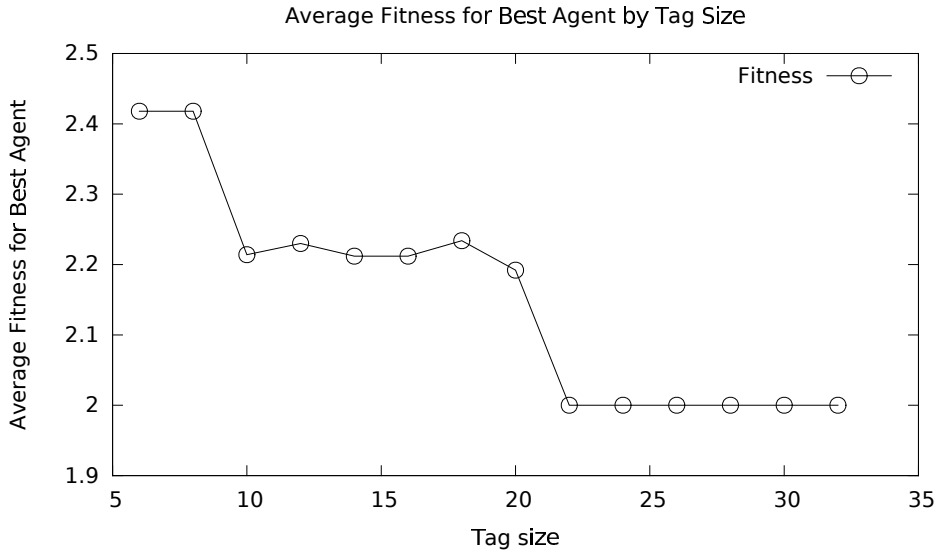

Figure 8: The fitness achieved by best agent with different tag lengths.

agent's chosen tag. This likelihood is the parameter, k, representing the likelihood of a matching mechanism matching a random tag. Then, the estimated number of groups that an agent can cooperate with would be determined from a binomial distribution:

$$E[C] = \frac{1}{g} + \sum_{i=2}^{g} \binom{g}{i} k^i (1-k)^{g-i}.$$

For $TS \to \infty$, $k \to 0$ and the estimated number of groups that the best agent can cooperate with approaches $\frac{1}{g}$ (since the sum term is eliminated). In the experimental data, the best agent tends to cooperate with more than the proportional estimated number of cooperators since it will always choose to cooperate with groups that "overlap" with others in their matching string (that is, multiple groups matching similar tags) over those that do not.

7.4 Tag Length Effect

The length of the tag determines the size of the search space for our experiments. The longer tags means a broader space so it becomes harder and harder for our agents to find the optimal (or in some cases even near optimal) tag and match strings. We tried every even number as the tag length between 6 and 32 (both included) to see how that effects our evolutionary population.

In Figure 8 it is shown that the agents are able to find better strategies for shorter tags. When the tag gets longer than 22 bits the agents usually get stuck in pure not cooperate strategy (see Figure 9).

With higher tag lengths getting cooperation from any agent becomes really unlikely for the learning population. When this is combined with our payoff adjustment method, the cost of not cooperating with a cooperator becomes really high, because they are scarse. This pushes the agents to try to get non-cooperation from all the static population and make sure that they do not defect a cooperator.

Some ways of improving the performance of the learning population has been listed earlier on: by adjusting the learning matching mechanism to be initially all cooperate, learning agents are encouraged to search the state space by tag before focusing on the matching mechanism. In addition, by reducing the mutation rate of matches, the evolving population is given even more time to search the tag space without deciding to defect from the entire population (which becomes more likely as tag size increases).

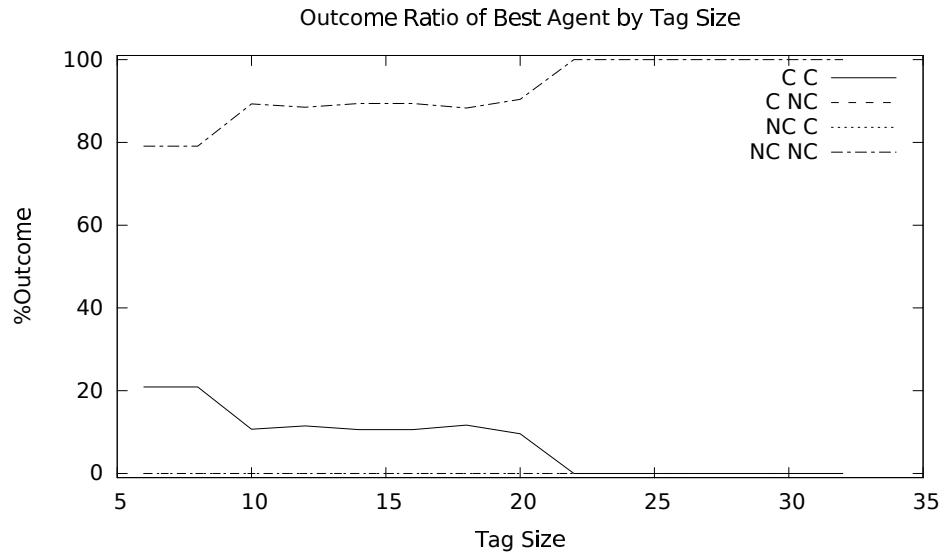

Figure 9: The distribution of outcomes with different tag lengths.

8 Concluding Remarks

Stereotyping is a core mechanism for identification and learning. Extracting hidden strategies from agents using judgments of external features is essential to reusing learnt behavior in relevant context and with appropriate company and hence to functioning effectively in a stable society. We investigate processes simulating the assimilation of agents into new cultures and synthesis of personal stereotyping mechanisms using an evolutionary approach. A set of matching mechanisms, which may classify agents into "cooperate" and "no cooperate" categories, are introduced to reproduce stereotype development. Agents cooperate in an evolutionary framework to learn inside an established society of agents.

Simulations consist of two populations: the incoming learning population and unchanging static population. Interacting agents from different populations play a modified version of the Coordination game; the goal is for both players to play identical, or mutual strategies. The optimal agents presented are a sort of benchmark of our evolutionary mechanism for low tag sizes. Their performance in the trials given show that the evolutionary population can, in many scenarios, perform as well as the optimal. In addition, the fact that the performance of the best ternary string is also near the optimal in scenarios where the evolutionary mechanism is not shows that this matching mechanism is effective.

While the evolutionary mechanism in our results shows effective adaptation to a wide variety of populations, modifications to the process may benefit the system further. Currently, the learning process is simultaneous over tag and match mechanism evolution: Separating these processes may produce mutual outcomes with more consistency at the end of the simulation by ensuring more effective convergence. While Ternary matching has proven to be an effective mechanism to discriminate for the learning population, it has been shown to have poor performance against complex mechanisms such as decision trees. Alternative learning matching schemes may be able to perform better in such scenarios.

In addition, while extending the tag length of the agents was explored in this paper, the conclusions suggest that subtleties in behavior exist for smaller tag sizes. Adjustments may make this application more realistic as Complex matching behavior may encourage more subtleties to appear as tags grow larger.

References

[1] C. Burnett, T. J. Norman, and K. Sycara. Bootstrapping trust evaluations through stereotypes. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, AAMAS '10, pages 241–248, Richland, SC, 2010. International Foundation for Autonomous Agents and Multiagent Systems.

[2] R. Drężewski. A model of co-evolution in multi-agent system. In *Multi-Agent Systems and Applications III*, pages 314–323. Springer, 2003.

[3] Y. Fu, W. Ke, and J. Mostafa. Automated text classification using a multi-agent framework. In *Proceedings of the 5th ACM/IEEE-CS joint conference on Digital libraries*, pages 157–158. ACM, 2005.

[4] D. Hales. Change your tags fast!–a necessary condition for cooperation? In *Multi-agent and multi-agent-based simulation*, pages 89–98. Springer, 2005.

[5] D. Hales and B. Edmonds. Evolving social rationality for mas using tags. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 497–503. ACM, 2003.

[6] L. M. Hogg and N. R. Jennings. Socially rational agents. In *Proc. AAAI Fall symposium on Socially Intelligent Agents*, pages 8–10, 1997.

[7] C. M. Judd and B. Park. Definition and assessment of accuracy in social stereotypes. *Psychological review*, 100(1):109, 1993.

[8] M. Lynn. Scarcity effects on value: A quantitative review of the commodity theory literature. *Psychology & Marketing*, 8(1):43–57, 1991.

[9] M. Matlock and S. Sen. Effective tag mechanisms for evolving coordination. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 251:1–251:8, New York, NY, USA, 2007. ACM.

[10] A. McDonald and S. Sen. The success and failure of tag-mediated evolution of cooperation. In *Learning and Adaption in Multi-Agent Systems*, pages 155–164. Springer, 2006.

[11] R. L. Riolo. The effects and evolution of tag-mediated selection of partners in populations playing the iterated prisoner's dilemma.". In *Proc. of Inter. Conf. Genetic Algorithms (ICGA-97), Thomas B ack (ed.)*, pages 378–385, 1997.

[12] A. R. Wagner. Using stereotypes to understand one's interactive partner. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1445–1446. International Foundation for Autonomous Agents and Multiagent Systems, 2010.