Inducing selfish agents towards social efficient solutions

João Schapke[1], Ana L.C. Bazzan[1]

Universidade Federal do Rio Grande do Sul, Brazil
joaoschake@gmail.com, bazzan@inf.ufrgs.br

Abstract. Many multi-agent reinforcement learning (MARL) scenarios lead towards Nash equilibria, which is known to not always be socially efficient. In this study we aim to align the social optimization objective of the system with the individual objectives of the agents by adopting a central controller which can interact with the agents. In details, our approach establishes a communication channel between reinforcement learning agents, and a controller implemented with metaheuristics. The interaction benefit the convergence of both algorithms. Further, we evaluate our method in repeated games with high price of anarchy and show that our approach is able to overcome much of the issues caused by the non-cooperative behaviour of the agents and the non-stationary effects they cause.

CCS Concepts: • **Computing methodologies** → **Multi-agent planning**; **Reinforcement learning**.

Keywords: genetic algorithm, metaheuristics, Q-learning, reinforcement learning

1. INTRODUCTION

In multi-agent scenarios it is common to have agents which interact with one another and optimize, each, its own individual utility. This characterizes a non-cooperative environment and may bring sub-optimal solutions. For example, in a scenario where agents share limited public resources the objectives of the agents may be opposed to one another, i.e., the profit of an agent has a negative impact on the profit of others. Thus, agents may develop competitive behaviour and take actions that exhaust limited resources for short-term profit. In other scenarios, objectives may not be diametrically opposed to each other, but the lack of coordination among the agents may still worsen the performance of them all. This is for instance the case of a traffic congestion scenario, in which all agents decide to take the same road, instead of spreading across all available choices in order to minimize the total travel time. These outcomes are undesirable, in most cases we wish agents to optimize a collective objective and act cooperatively.

Cooperative behaviour is not trivially solved in the reinforcement learning (RL) framework. It requires reasoning over the global and long-lasting impact of individual actions. Standard multi-agent reinforcement learning (MARL) approaches do not consider this non-Markovian effect, and thus, do not provide suitable solutions. A common approach is to model the impact of individual actions as stochasticity of the environment. This provides a simplified framework, but hinders the capability of agents to cooperate. Often, if cooperative behaviour is not strongly enticing, agents converge to Nash equilibria, which may not be aligned with socially efficient solutions.

This approach, however, has parallels to real world scenarios. Real agents in large scale scenarios also overlook the impact of their own actions on the system as a whole. Traffic congestion and public resource allocation problems are not solved by the meticulous calculation of individual agents. They are solved by coordination established by a central authority (e.g. laws, semaphores.) This leads us to

infer that many cooperation problems can be solved with RL agents by introducing a central controller that can interact with the agents and has global information on the system. In this study we adopt a framework with such intuition and show it improves the performance of RL agents in non-cooperative scenarios.

We extend previous work aimed at solving traffic assignment problems by [Bazzan 2019]. Our approach establishes a communication channel between agents and a central controller whose task is to induce the agents to follow cooperative strategies. Essentially, the controller optimizes the system's social welfare using metaheuristics (i.e., genetic algorithm.) The communication channel allows for the supervisor to make suggestion of choices to the agents, so as to prevent the agents to falling into local minima. Both approaches are optimized individually and are allowed for restricted interaction with one another.

Following, we briefly introduce some concepts that underlie our study in Section 2. We then draw on closely related work in Section 3. Our approach is detailed in Section 4. In Section 5 we empirically show that our approach leads agents to converge to solutions near the global optimum. Section 6 concludes the article.

2. BACKGROUND

2.1 Game Theory and cooperation

Game theoretics problems, or games, usually contrast two types of behaviours of agents, a defective and a cooperative behaviour. In these games, which we denote as the base-game, the cooperative behaviour provides the most socially efficient solution if all players follow it, while the defective behaviour is more promising for the individual player. In repeated games, the same base-game is repeated through many iterations with the same agents. This leads to a scenario that agents may influence one another. Defective behaviour may lead other agents to defect as well causing a poorer overall performance to all agents. Thus, agents need to cooperate with one another to maximize their own utility.

Further in Section 4 we introduce a repeated game with a routing problem as the base-game. Our problem serves as a benchmark of a difficult large-scale cooperation problem. In many games cooperation and coordination are intertwined as cooperation requires coordination, which is the case for the problem proposed. Although this is not a strict rule, [Leibo et al. 2017] introduced a sequential social dilemma which the non-cooperative strategy required more coordination.

Two useful concepts are those of the Nash equilibrium and the price of anarchy which we describe next.

2.1.1 *Nash equilibrium.* A Nash equilibrium is defined as a solution of a non-cooperative game involving two or more players, in which each player has nothing to gain by changing his own strategy. Formally, for the set of possible strategies S_i for player i, a solution S^* is a Nash equilibrium if:

$$\forall i, x_i \in S_i, x_i \neq x_i^* : f_i(x_i^*, x_{-i}^*) \geq f_i(x_i, x_{-i}^*)$$

x_{-i}^* corresponds to the strategies for every player besides player i, and f_i denotes a mapping from the chosen strategies of all players to the reward received by player i: $f_i : X \rightarrow R_i$.

2.1.2 *Price of Anarchy.* Is a measure of the degradation introduced by uncoordinated selfish behaviour inside a system. It is formally defined as the ratio between the worst equilibrium of the individual solution and the optimal centralized solution:

$$\frac{\max f_{s \in S}(s)}{\min f_{s \in E}(s)}$$

S denotes the set of all solutions and E is the set of possible Nash equilibria. Hence, a game that has high price of anarchy is an instance where the Nash Equilibrium provides a socially inefficient solution.

2.2 Reinforcement Learning

In order to model selfish agents we use the standard reinforcement learning framework. We define a Markov Decision Process (MDP); which consists of a reward function $r : (s, a) \to \mathbb{R}$ which maps the action taken by an agent in a particular state to a reward, a transition function $\rho : (s_t, a_t) \to s_{t+1}$, mapping a state and action pair into a new state, and a state distribution function ρ_0, which gives the initial state. Each agent i in the MDP receives an observation o_i of the state s at each time-step, the agent aims to find a policy π that maps his current observation into an action that leads to the most reward.

In multi-agent reinforcement learning (MARL), we can model each agent individually and also avoid explicitly modeling the behaviour of other agents by assuming their change of behaviour as stochasticity produced by the environment. This provides a simple scenario that we can solve using common RL algorithms, such as Q-Learning. In turn, this also leads to an non-stationary environment into which the agent has to adapt. Additionally, in these scenarios the Markov property does not hold, nor the convergence guarantees for most reinforcement learning algorithms.

Throughout this study we use Q-Learning [Watkins and Dayan 1992] with an epsilon greedy policy for the aims of modeling selfish agents. The algorithm finds an optimal policy for each agent by estimating the Q value of each possible state-action pair in the game. The update rule for the agent's Q values is given by:

$$Q(s, a) = Q(s, a) + \alpha[R(s, a) + \gamma \max Q'(s', a') - Q(s, a)] \tag{1}$$

Parameters α and γ denote the learning rate and the discount factor, respectively.

2.3 Metaheuristics

Our central controller, or supervisor, is modelled with the genetic algorithm [Goldberg 2006], a metaheuristic algorithm inspired by biological evolution. The intuition behind the genetic algorithm is to create a random population of solutions and iteratively improve it by random mutation and crossover. In details, at each iteration of the genetic algorithm we randomly mutate solutions within the population by a probability m, then we choose solutions of the population with probability c and merge them together to create a novel solution. These selections are stochastically chosen and give a higher propensity to the most fit solutions in the population. Additionally, we also choose to keep the e most fit individuals of the current population into the next, denominated as the elite. The fitness of a solution is defined by a fitness function given to the algorithm, in our study the fitness function measures the social efficiency of the solution, i.e., the average reward of all agents. Therefore the supervisor contrasts the agents and optimizes the global performance of the system.

The population, denoted by $\mathcal{POP}$, has $p = |\mathcal{POP}|$ individuals. This value must be carefully set so the set of individuals within it is diverse, but the value should not be too high as it would mean increased computation at each generation.

3. RELATED WORK

[Tan 1993] explored cooperation in multi-agent scenarios through interaction among agents themselves during play and with an expert player that shares his previous experiences. His results showed that Q-Learning agents could coordinate and learn from each other. More recently, [Leibo et al. 2017] and [Pérolat et al. 2017] studied cooperation among agents in complex non-cooperative sequential social

dilemmas. The benchmarks used by both consisted of graphical multi-player games and they used a deep learning variant of Q-Learning, Deep Q-Learning [Mnih et al. 2016], to learn directly from pixels. These previous analysis showed the capability of Q-Learning algorithms to learn cooperative strategies given enough incentive.

Recent work by [Foerster et al. 2016] also establishes communication between reinforcement learning agents. Similarly to our approach they also develop a decentralized and centralized scheme. They established a centralized training procedure and a decentralized execution. During execution, agents may communicate with each other, while in the training procedure agents are optimized by a central controller. In contrast, our approach allows agents to periodically communicate with a central controller, while they are trained individually from one another or from the controller.

[Verbeeck et al. 2007] developed a similar approach to ours, as they solved repeated games through an hybrid approach composed by the interaction of agents to a supervisor. Their work focused on inducing agents to reach egalitarian solutions, while we aim for the agents to reach the globally optimal solution.

As an alternative to centralized approaches, [Hughes et al. 2018] developed an approach that fosters cooperation in the MARL framework at the agent level through inequity aversion. In this approach, the authors add penalty terms to the reward function of agents based upon the model proposed by [Fehr and Schmidt 1999], which penalizes agents for non egalitarian solutions.

4. APPROACH

Our approach is composed by two interacting entities, the agents and the supervisor. Given an MDP constituting a multi-agent environment we independently model each agent using Q-Learning. Every agent $\mathbf{i}$ has its own associated reward function $r_i : (s, a) \rightarrow \mathbb{R}$. Agents choose an action within a finite set of alternatives given by the environment. In turn the supervisor has its fitness function given by the accumulated rewards of all agents $\sum_i r_i$. An individual in the supervisor's population, represents a vector of actions with the size equals to the number of agents. Throughout this study our MDP's are instances of single-step repeated games and therefore have a single state (as proposed in [Claus and Boutilier 1998].) This means that the update rule in Equation 1 directly maps to the value of the action itself.

The supervisor can influence the agents by providing them a suggestion, that is, it can periodically decide which action is taken by the agents. The suggestion corresponds to the best solution in the supervisor's current population. While agents can interact back to the supervisor, at every game, they communicate the actions they took and the reward they obtained from the environment to the supervisor. This interaction is beneficial for both algorithms; The supervisor gets additional information due to the exploratory behaviour of the agents' epsilon greedy policy. It presents agents to social efficient solutions to follow and learn from. This is useful for the agents as otherwise they would not have the coordination to experience these solutions. Further, we show that this simple interaction prevents agents to converge to Nash equilibria.

We provide a high level overview of the training procedure in Algorithm 1. Our procedure requires the parameters both to Q-learning and the genetic algorithm as presented in Section 2. Additionally, a parameter $\delta \in \mathbb{N}$ for the rate of suggestion from supervisor to the agents and an objective function r, representing the MDP or the single-step repeated game being played, must be passed. The procedure is initialized by generating the initial population of solutions of the supervisor and the Q tables of the agents. At every iteration the agents decide which action to take, while at every δ iterations the agents receive a suggestion of which actions to take from the supervisor. The agents receive a reward from the environment and are updated using the Bellman equation 1. Their actions and rewards are also included in the supervisor's population in place of its current worst performing solution. The supervisor's population is updated at every $|\mathcal{POP}|$ steps, that keeps the number of environment

Algorithm 1: Training procedure using Q-learning and the genetic algorithm for the players and the supervisor, respectively.

Input: α, ϵ, γ: Q-learning parameters,
Input: m, c, e: Genetic algorithm parameters,
Input: δ interaction factor from supervisor to players
Input: r objective function
Result: S^*, Q^*: The optimized solution and the Q table of each player.
Initialize Q_i of each player i
Initialize $\mathcal{POP}$, supervisor's initial population
for $step \in \{1, ..., K\}$ **do**
  if $step$ % $\delta == 0$ **then**
    actions ← supervisor.best_solution
  else
    actions ← agents.act()
  end
  rewards ← r(actions)
  agents.update(actions, rewards)
  `// actions replace the worst solution in the supervisor's population and reorder it.`
  supervisor.worst_solution ← actions
  supervisor.rank_pofpulation()
  `// This makes the agents and the supervisor get updated in a equivalent step rate.`
  if $0 \equiv step \pmod{|\mathcal{POP}|}$ **then**
    supervisor.run_generation()
  end
end

interactions of both algorithms at the same pace.

We introduce an instance of a routing problem in the form of a repeated game, displayed at Figure 1. The figure contains a graph with two types of nodes, start nodes and destination nodes. All edges in the graph have associated cost functions, which gives the cost of travelling through a route. The cost functions are parameterized by the total flow of agents in the edge, i.e., the total number of agents going through that route. Destination nodes have an associated value, which the player obtains for reaching the node. The individual objective of each agent is, therefore, given by the value of the destination node it reaches minus the cost of the arcs used by it. Agents must only decide which destination to reach in each iteration of the game.

In the example we adjusted the cost functions and the values for the case that 100 players start at each start position so as to obtain a high price of anarchy for the system. Further we restrict the destinations allowed for each agents to those two hops away from the agent's initial position. The socially optimal solution (i.e., the one that is optimal for the system as a whole) of the problem is given when each agent travels to the destination to the right of its start position. On the other hand, in the Nash equilibrium, agents in the start nodes (S1, S2, S3) travel to D2, and agents in (S4, S5, S6) to D5. The mean reward of the agents obtained in the optimal solution is 60, and in the Nash equilibrium is 26.66.

Noticeably, the paths leading to the most valuable destinations have a cost that is proportional to the flow of agents, while paths to the least valuable destinations are inversely proportional. This leads to the case where if most agents travel to the most valuable destination not only the route towards it becomes more costly but so as every other route. Therefore, agents should distribute themselves over the destination choices so as to keep costs of the paths low. In our experiments Q-Learning fails to do so, the agents take on the most beneficial destinations and converge to the system's Nash equilibrium that, we remark, is not socially optimal.

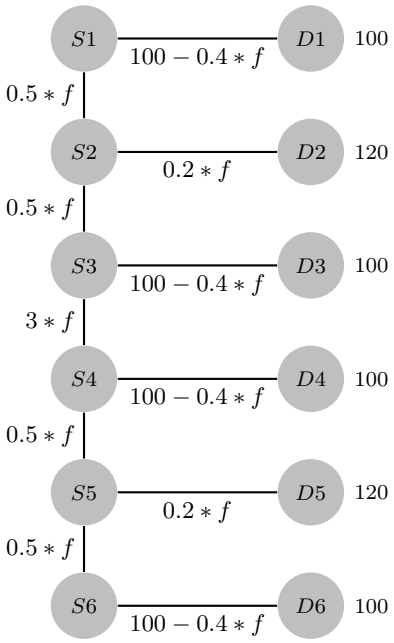

Fig. 1. **Benchmark scenario.** The graph represents a routing game. Agents start into the start nodes (S_i) and must reach a destination node (D_j). Edges have an associated cost of travel, which is linear to the flow (f) of agents travelling through it. Destination nodes have an associated value that is received by an agent upon reaching it.

5. EXPERIMENTS AND ANALYSIS

Our experiments intend to measure the effectiveness of the communication between players and the supervisor. We ran each experiment 30 times and report the mean and the standard deviation of the runs. Unless otherwise noted, the parameters for the Q-learning algorithm were set to an initial epsilon $\epsilon_0 = 1$, learning rate $\alpha = 0.6$, epsilon decay $\gamma = 0.99$. The parameters for the genetic algorithm were set to mutation rate $m = 0.5$, crossover rate $c = 0.5$, population size $p = 50$, elite size $e = 5$. The parameter δ for communication between supervisor and agents was set to $\delta = 10$.

Proposed Environment.

The results of the problem proposed in Section 4 are displayed in Figure 2. We show the performance of the players and the supervisor in the case which both establish communication with one another and in the case which both are optimized independent of one another. The performance of the QL agents without any aid to develop coordination showed the ailments formulated in Section 4, the players converge to the Nash equilibrium, which gives short term benefit. Our approach converged to a solution closer the global optimum, showing that a supervisor can successfully induce greedy individuals to coordinate. The supervisor also benefits from the additional exploration provided by the epsilon-greedy policy of the Q-Learning agents.

Interestingly this is a hard case for Q-learning even if substantionally decrease the difficulty of the problem. If we condition the initial Q-table of the agents so as they bias closer routes, i.e., by setting values of choices to the inverse of the geodesic distance from the start position, and we set the initial epsilon e_0 to a low value, agents will already start in a solution very near to the global optimal one. In this situation one might expect that agents will be able to maintain their performance, or at least perform better than the agents solving the standard problem. This is not the case, as shown in Figure 3 agents end up converging to the same solution sub-optimal solution as in the previous

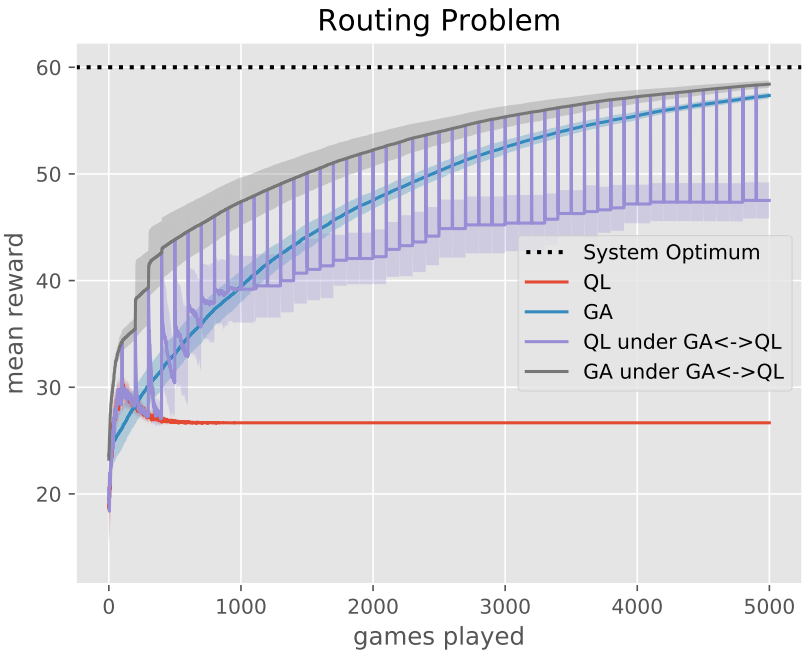

Fig. 2. **Comparison of algorithms in the proposed routing problem of Figure 1.** Here we show the mean rewards of the agents with the choices made by the supervisor and by the Q-Learning agents under our proposed algorithm, and we show the performance of both algorithms without any interaction among them as well.

experiment. In the results of the figure we set initial epsilon $e_0 = 0.05$ and initialize the Q-tables as aforementioned. Although extreme scenarios as this one are quite rare this experiment illustrates the tendency of Q-learning to converge to Nash equilibria.

6. CONCLUSION

In this study we adopted a simple approach that establishes a communication channel between a central controller and local agents. We empirically showed the effectiveness of the approach in inducing local agents to more socially efficient solutions. We also proposed a generic routing problem that serves as a benchmark to measure coordination of multi-agent algorithms in large-scale scenarios.

REFERENCES

Bazzan, A. L. C. Aligning individual and collective welfare in complex socio-technical systems by combining metaheuristics and reinforcement learning. *Eng. Appl. of AI* vol. 79, pp. 23–33, 2019.

Claus, C. and Boutilier, C. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. AAAI '98/IAAI '98. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 746–752, 1998.

Fehr, E. and Schmidt, K. M. A theory of fairness, competition, and cooperation. *The Quarterly Journal of Economics* 114 (3): 817–868, 1999.

Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., pp. 2137–2145, 2016.

Goldberg, D. E. *Genetic algorithms*. Pearson Education India, 2006.

Hughes, E., Leibo, J. Z., Phillips, M., Tuyls, K., Dueñez Guzman, E., García Castañeda, A., Dunning, I., Zhu, T., McKee, K., Koster, R., Roff, H., and Graepel, T. Inequity aversion improves cooperation in

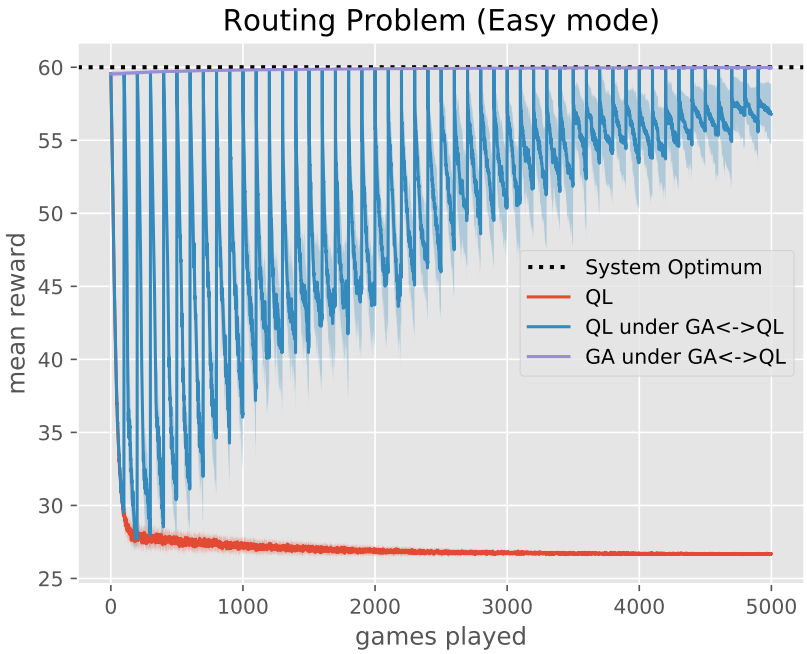

Fig. 3. **Comparison of algorithms in the proposed routing problem of Figure 1.** In this setup the initialization of the Q-tables is biased to start in at an optimal solution.

intertemporal social dilemmas. In *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.). Curran Associates, Inc., pp. 3326–3336, 2018.

Leibo, J. Z., Zambaldi, V., Lanctot, M., Marecki, J., and Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. AAMAS '17. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, pp. 464–473, 2017.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. *CoRR* vol. abs/1602.01783, 2016.

Pérolat, J., Leibo, J. Z., Zambaldi, V., Beattie, C., Tuyls, K., and Graepel, T. A multi-agent reinforcement learning model of common-pool resource appropriation. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., pp. 3643–3652, 2017.

Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)*. Morgan Kaufmann, Amherst, MA, USA, pp. 330–337, 1993.

Verbeeck, K., Nowé, A., Parent, J., and Tuyls, K. Exploring selfish reinforcement learning in repeated games with stochastic rewards. *Autonomous Agents and Multi-Agent Systems* 14 (3): 239–269, Apr, 2007.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning* 8 (3): 279–292, 1992.