Procedia Computer Science

Volume 80, 2016, Pages 855–864

ICCS 2016. The International Conference on Computational Science

Reinforcement Learning with Multiple Shared Rewards

Douglas M. Guisi[1], Richardson Ribeiro[1], Marcelo Teixeira[1], Andre P. Borges[2] and Fabricio Enembreck[3]

[1]*Department of Informatics, Federal University of Technology of Paraná, Pato Branco, Brazil*
*{dguisi, richardsonr, marceloteixeir*a}@utfpr.edu.br

[2]*Department of Informatics, Federal University of Technology of Paraná, Ponta Grossa, Brazil*
apborges@utfpr.edu.br

[3]*Graduate Program in Computer Science, Pontifical Catholic University of Paraná, Curitiba, Brazil*
fabricio@ppgia.pucpr.br

Abstract

A major concern in multi-agent coordination is how to select algorithms that can lead agents to learn together to achieve certain goals. Much of the research on multi-agent learning relates to reinforcement learning (RL) techniques. One element of RL is the interaction model, which describes how agents should interact with each other and with the environment. Discrete, continuous and objective-oriented interaction models can improve convergence among agents. This paper proposes an approach based on the integration of multi-agent coordination models designed for reward-sharing policies. By taking the best features from each model, better agent coordination is achieved. Our experimental results show that this approach improves convergence among agents even in large state-spaces and yields better results than classical RL approaches.

Keywords: Adaptive Agents, Shared Rewards, Interaction, Learning, Coordination.

1 Introduction

The collective behavior of social groups (e.g., agents) has inspired the development of computational models for generating solutions to optimization problems. This behavior is the result of patterns of interaction between agents in a population and, rather than being merely a property of a single control system, is a consequence of the behavior of the individuals in the population. In such a system, each individual's structure is relatively simple, but complex social structures emerge from their collective behavior (Ribeiro and Enembreck, 2013).

In a multi-agent system (MAS), agents need to interact and coordinate in order to carry out tasks. Coordination between agents can help to avoid problems with redundant solutions, inconsistency of

Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2016

doi:10.1016/j.procs.2016.05.376

execution, resource waste and deadlock (Stone and Veloso, 2000), enabling learning-based coordination models to solve complex problems involving social and individual behaviors (Zhang and Lesser, 2013).

In addition to being able to learn, an agent in an MAS must be able to cooperate with other agents in the system in order to attempt to solve problems that require locally unknown knowledge or that could compromise the agent's performance. In this way, sharing agent expertise (usually in terms of *action policies*) becomes essential to converge to a global behavior that satisfies a certain specification or simply solves a particular problem.

An alternative to sharing an agent's expertise among multiple agents is to use specific computational paradigms that maximize performance based on reinforcement parameters (rewards or punishments) applied to agents as they interact with the environment. Learning based on paradigms of this kind is called reinforcement learning (RL) (Kaelbling *et al.*, 1996), (Sutton and Barto, 1998).

RL algorithms, such as *Q*-learning (Watkins and Dayan, 1992), can be used to discover the optimal action policy for a single agent when it repeatedly explores its state-space. A major concern that arises from this action-policy-discovering approach is that it tends to suffer from large state-spaces because of *state-space explosion* problems. In such cases, RL involving multiple agents has proved to be a promising strategy as it modularizes the whole problem and implements action policies locally (Ribeiro *et al.*, 2008).

The idea behind the implementation of local policies is to discover a global action plan generated by combining agents' local knowledge. When this approach leads to the best global action plan, the policy π is said to be optimal (π^*) and corresponds to the highest rewards received by the agents.

In this paper we integrate coordination models for multiple agents using RL techniques. Our approach collects "good" features from individual approaches in the literature and integrates them into a single framework that can then be used to establish optimized information-sharing strategies for multiple agents. The preliminary results allow the performance of the integrated model to be compared with the performance of each model used in the framework. In general, the convergence rate among agents was substantially better than in the other cases.

The remainder of this paper is organized as follows. In Section 2, the state of the art is presented, some coordination methods for learning in multi-agent systems are described, the *Q*-learning algorithm is reviewed and interaction models are summarized. In Sections 3 and 4, details of the proposed method and the environment used to evaluate it are discussed, and numerical results illustrating the performance of the proposed method are presented. The conclusions are given in Section 5.

2 Learning in Multi-Agent Systems

Multi-agent reinforcement learning, in which multiple agents are involved in the solution of a complex task in a common environment, has been the subject of a considerable amount of research over the last decade. Unlike learning in an environment with a single agent, learning in an MAS assumes that the relevant knowledge is not locally available in a single agent, making it is necessary to coordinate the whole process (Chakraborty and Stone, 2014), (Zhang and Lesser, 2010), (Xuan and Lesser, 2002). One way for an agent to coordinate its actions is by interacting with other agents, changing and evolving their coordination model.

Coordination by interaction involves combining the efforts of a group of agents in the search for solutions to global problems (DeLoach and Valenzuela, 2007). Interaction can be considered the set of behaviors that result when a group of agents act to satisfy their goals and consider constraints imposed by resource limitations and individual skills. There is a significant body of literature on learning from interactions (Ribeiro *et al.*, 2013), (Xinhai and Lunhui, 2009) and collective or social learning (Ribeiro *et al.*, 2013) (Ribeiro and Enembreck, 2013).

In learning problems involving RL, interaction depends basically on a structure that enables communication among agents so they can share their accumulated rewards, immediately reinforcing the transition system. With this in mind, Chapelle *et al.* (2002) created an interaction model that calculates rewards based on the individual satisfaction of neighboring agents in which agents continuously emit a level of personal satisfaction during the learning process. Adopting a different approach, Saito and Kobayashi (2016) developed a learning strategy in which agents are able to remember information they have accumulated so they can reuse it later. This method has been tested on colored mazes, and reports confirm that it has a positive impact on jumpstarts and reduces the total learning cost compared with conventional *Q*-learning.

Ribeiro and Enembreck (2013) combined theories from different fields to build social structures for state-space searches based on the way interactions between states occur and reinforcements are generated. They used social measures to guide exploration and approximation processes. Their experiments showed that identifying social behavior that incorporates interaction between agents within the social structure helps to improve the coordination and optimization process and yields results that are statistically more significant.

Integrating different methods into a single improved generic coordination model is usually challenging, especially because of the wide range of problems and the amount of knowledge about the problem domain required. Furthermore, in an MAS, conflicting values for cumulative rewards can be generated, as each agent uses only local learning values (DeLoach and Valenzuela, 2007). Collective learning assumes that the relevant knowledge is acquired when rewards are shared, intensifying the relationship between agents.

2.1 Reinforcement Learning

In RL an agent is given a reward or punishment by the environment in response to its actions (Kaelbling *et al.*, 1996). This type of learning has been extensively investigated in the literature (Grzes and Hoey, 2011), (Devlin *et al.*, 2014), (Efthymiadis and Kudenko, 2015), (Tesauro, 1995), (Walsh *et al.*, 2010), (Zhang and Lesser, 2013) as part of efforts to find solutions to NP-hard problems.

We introduce briefly the Markov decision process (MDP) used to formalize the RL problem. An MDP is a tuple (S, A, $\beta^{a}_{s,s'}$, R), where *S* is a finite set of environmental states that can consist of a variable sequence of *states* = $< x_1, x_2, ..., x_y >$. An episode is a sequence of actions $a \in A$ that leads the agent from an initial-state to a goal-state. $\beta^{a}_{s,s'}$ is a function that indicates the probability that the agent arrives in state *s* when an action *a* is applied in state *s'*. Similarly, $R^{a}_{s,s'}$ is the reward received whenever the transition $\beta^{a}_{s,s'}$ occurs.

An RL agent must learn a policy Q: $S \rightarrow A$ that maximizes its expected cumulative reward, where $Q(s, a)$ is the probability of selecting action *a* in state *s'*. The optimal policy must satisfy Bellman's equation for each state $s \in S$:

$$Q(s, a) = R(s, a) + \gamma \sum \beta(s, a, s') \times \max Q(s', a) \tag{1}$$

where β is the weight of the values of future rewards and $Q(s, a)$ is the expected cumulative reward given that action *a* is executed in state *s* (Sutton and Barto, 1998). To reach an optimal policy, an agent that uses an RL algorithm must iteratively explore the state space ($S \times A$), updating the cumulative rewards and storing these in a table *Q*. The *Q*-learning algorithm proposed by (Watkins and Dayan, 1992) converges to an optimal policy by applying the following update rule (Equations 2 and 3) after a time step *t*:

$$V = \gamma \max Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \tag{2}$$

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha [R(s_t, a_t) + V] \tag{3}$$

where V is the utility value of performing an action a in state s, and $\alpha \in [0, 1]$ is the learning rate. In dynamic environments it is desirable to use strategies because a satisfactory policy may no longer be appropriate after a change in the environment. When a strategy, such as ε-greedy, is used, the agent selects an action with the greatest Q value with probability $1 - \varepsilon$. In previous experiments with the Q-learning algorithm (Ribeiro *et al.* 2008), we found that the agent was not able to converge in dynamic environments during training (see Section 3) so we used an important property of the Q-learning algorithm, namely, that actions can be chosen using a random exploration strategy determined by ε. The state transition is given by Equation 4:

$$\pi(s) = \begin{cases} argmaxQ(s,a), & if\ q > \varepsilon \\ a_{random}\ , & \text{otherwise} \end{cases} \quad (4)$$

where q is a random value with uniform probability in [0, 1], ε ($0 \leq \varepsilon \leq 1$) is a parameter that defines the exploration trade-off (the greater the value of ε, the smaller the probability of a random choice) and a_{random} is a random action selected from the possible actions in state s. Here, the ε-greedy exploration strategies were able to estimate better rewards and come up with new action policies.

In this paper, Q-learning is used to generate and evaluate partial and global action policies. By applying Q-learning, a policy can be found for each agent. However, if similar agents interact in the same environment, each agent has its own MDP, and optimal global behavior cannot be determined by local analysis. Thus, in an environment involving several agents, the goal is to select the actions of each MDP at time t so that the total of the expected rewards for all agents is maximized.

2.1.1 Reinforcement Learning with Multiple Shared Rewards

In RL algorithms with shared rewards, one agent's actions can produce a policy that has an effect on all individuals and eliminate their idiosyncratic behavior. Rewards are shared by agents through a partial action policy (Q_i). Usually, such policies contain partial information (learning values) about the environment but communicate with a central structure to share rewards in an integrated way in order to maximize the sum of the partial rewards obtained during the learning process. When policies $\pi_1,\ldots,$ π_x are integrated, a new policy π^+ can be generated, where π^+ denotes the best rewards acquired by agents during the learning process.

Ribeiro *et al.* (2008) showed how agents exchange information during learning. When the supervisor agent receives rewards from other agents, the following process occurs: on reaching goal state g by a lower-cost path, agent$_i$ uses a model to share the rewards with other agents. The reward of a partial policy π_1 can be used to upgrade the overall policy π^+, further influencing how other agents update their knowledge and interact with the environment.

Ribeiro *et al.* (2008) also described the function that shares these rewards. This sharing can be accomplished in three ways, all of which involve internal sharing using Q-learning. The best rewards from each agent are sent to π^+, forming a new policy with the best rewards acquired by agent$_i$, which can be socialized with other agents. A policy is considered optimal when the agent is able to find the goal-state with the lowest possible cost, i.e., a cost similar to that provided by the supervisor agent (A* algorithm).

The interaction models for cooperative RL presented in (Ribeiro *et al.*, 2008) are summarized below (i, ii, iii). The cooperative RL algorithm, the other algorithms and the elements formalizing the RL models are detailed in (Ribeiro *et al.*, 2008).

i) Discrete Model: Agents share learning in a predefined cycle of interactions c. Cooperation in this model occurs as follows: the agent accumulates rewards it obtained as a result of its actions during the learning cycle. At the end of the cycle, the agent sends the values of π_i to π^+. An agent shares its reward if and only if it improves the efficiency of the other agents in the same state.

ii) Continuous Model: Agents cooperate at every transaction $T^a_{s,s'}$. Cooperation in the continuous model occurs as follows: if $s \neq g$, then every action performed by the agent generates a reward value,

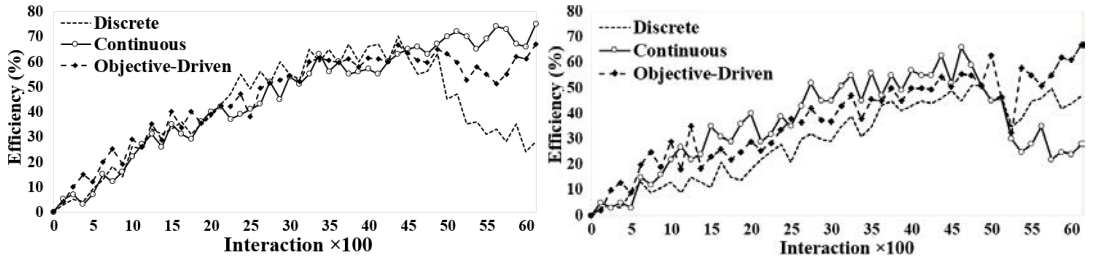
which is the sum of the accumulated reinforcements for all players in action a in state s. The goal is to accumulate the greatest rewards in π_i so that these can be shared at each interaction.

iii) Objective-driven Model: Unlike in the discrete model, cooperation in the objective-driven model occurs when the agent reaches the goal state, i.e., $s = g$. In this case, the agent interacts and accumulates reward values. This is necessary because in this model the agent shares his rewards only when the goal state is reached. When the agent reaches the goal state, the reward value is sent to π^+. If the reward value for the state improves the overall efficiency, then the agents share the reward. This shows that even when unsatisfactory rewards are shared (because of a lack of interaction), the agent is able to adapt his behavior without adversely affecting global convergence.

3 Integrated Interaction Model

In RL based on shared rewards, it is common to discover intermediate action policies that do not help to achieve a certain goal. In fact, knowledge exchange among agents may lead to intermediate action plans that do not immediately help agents converge. As each agent constantly updates its own learning, all agents must be aware of all updates taking place and of each agent's rewards.

Using the previously presented approaches for agent coordination based on shared rewards, there is no guarantee that the action plans will converge. While policies with initially mistaken states and values are improved by rewards shared by other intermediate policies, improving the π^+ function, the opposite is also possible, i.e., initially interesting policies with high rewards may become less attractive during execution of a given policy.



A. **400** states; **5** agents B. **400** states; **10** agents

Figure 1. Coordination models (Ribeiro *et al.*, 2008).

In order to overcome this inconvenience (local maximum), we integrated interaction models for multi-agent coordination, combining features of the discrete, continuous and objective-driven models previously presented. By analyzing the results obtained using these models, we found that the behavior of π^+ changes as a function of the number of interactions of the algorithms, the number of episodes involved in the problem and the cardinality of the set of agents used. This can be seen in Figure 1, which shows the results of coordination models in a 400-state environment with 5 and 10 agents.

The proposed model captures the best features from each individual interaction model at every interaction in the coordination process. New action policies are discovered without delaying the learning process, reducing the probability of conflicts between actions from different policies.

Technically, the model can be summarized as follows: at every interaction in the *Q*-learning process, the agent's performance is obtained from each interaction model. This is used to build a *learning table*, here referred to as *IM*-π^+ (Integrated Model). When the model update condition is reached, the agent starts its learning process using the best performance calculated from the learning tables in the interaction models. The learning is then transferred to *IM*-π^+, which will therefore always contain the best rewards from the discrete, continuous and objective-driven models.

Algorithm 1 shows how the discrete, continuous and objective-driven interaction models are integrated. For every learning iteration, the agent's performance is compared with the interaction models used. When a given model returns a superior performance, this learning is transferred to $IM\text{-}\pi^+$, which represents the current action policy of the integrated model.

In the next section, a simulation environment for assessing the efficiency of the proposed model is presented.

Algorithm 1: Model-integration algorithm

```
(s,a): model state/action;
Q_M_D: discrete model learning table;
Q_M_C: continuous model learning table;
Q_M_O: objective model learning table;
1  for each instance of (s,a) do
2     IM-Π+ ← max(Q_M_D, Q_M_C, Q_M_O)
3  end for
4  return IM-Π+
```

4 Experimental Results

To evaluate the proposed approach we used a simulation environment composed of a state-space representing a traffic structure through which agents (drivers) try to find a route in an empirical scenario with a grid-world structure. The structure has an initial state s_{init}, an objective state g and a set of actions $A = \{\uparrow$ (*forward*), $\rightarrow$ (*right*), $\downarrow$ (*back*), $\leftarrow$ (*left*)$\}$. A state s is a pair (X,Y), which defines the position on the X and Y axes, respectively. A status function st: $S \rightarrow ST$ maps traffic situations (rewards) to states, such that $ST = \{$*–0.2* (*free route*)*; –0.3* (*low congestion*)*; –0.4* (*high congestion or unknown*)*; –0.5* (*very high congestion*)*; –1.0* (*blocked*)*; 1.0* (*g*)$\}$.

Agents simulate routes that are available for drivers, and the global goal is to produce an action policy (a combination that maps states and actions) that can determine the best route connecting s_{init} to g. The global action policy is defined by determining step by step which action $a \in A$ should be performed at each state $s \in S$. After an agent's move (transition/interaction) from a state s to a state s', it knows whether or not the action was positive, as it recognizes the set of rewards shared by the other models. The reward for a given transition $T^a_{s,s'}$ is denoted by $st(s')$.

The results described in this section allow the performance of the proposed model and those of the discrete, continuous and objective-driven interaction models to be compared. The parameters used in the integrated model are the same as those used for the individual models: $y = 0.95$, $\alpha = 0.3$ and $\varepsilon = 0.9$.

To evaluate the performance of the model, we used different arbitrarily generated scenarios (Figure 2) in an attempt to reproduce situations that resembled real-world situations as closely as possible. The agents are randomly positioned in the state-space. When an agent reaches the goal state, it is randomly positioned in another state s until the stopping criterion (a maximum number of interactions) is satisfied.

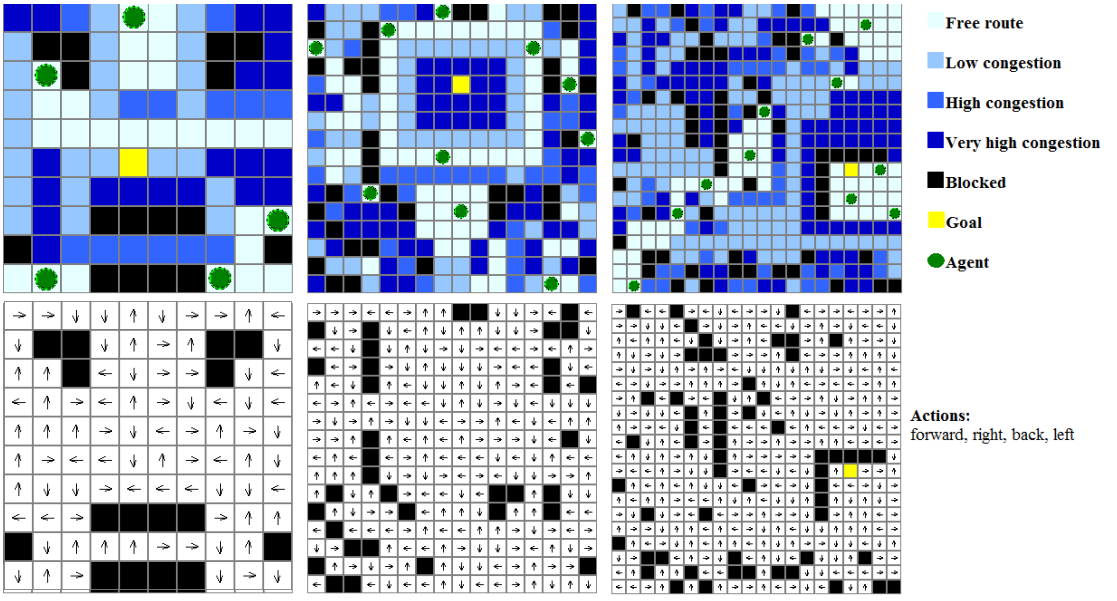


Figure 2. Simulated scenarios[1] (100, 256, 400 states). The agents have a visual field depth of 1 in the grid world.

Learning with the algorithm was repeated twenty times for each scenario since it was found that doing experiments in one environment alone using the same inputs could lead to a variation in the results computed by the algorithm as agent actions are probabilistic and the values generated during learning are stochastic variables. The action policy determined by an agent can therefore vary from one experiment to another. The efficiency reported in this section is the mean of all the experiments in each scenario. Twenty replications were sufficient to evaluate the algorithm's efficiency as the quality of the policies did not change significantly (see Table 1).

State-space	Number of agents		
	3	5	10
100	± 2.3%	± 3.1%	± 4.4%
256	± 3.2%	± 3.7%	± 4.7%
400	± 4.1%	± 4.3%	± 5.3%

Table 1. Variation in the results computed by the algorithm (standard deviation).

Although the problem simulated here could be deemed somewhat simplistic, it should be remembered that a total of *s* states can generate a large solution space, in which the number of possible policies is $|A||^{s}|$. The results shown below provide a comparison of the proposed model and the other methods using 3, 5 and 10 agents in an environment composed of 100, 256 and 400 states. The efficiency of the models (the *Y* axis in the graphs) is based on the number of correct hits by an agent during each interaction. A correct hit occurs when the agent finds the goal-state with its costs optimized using the evaluation methodology for RL algorithms introduced in (Ribeiro *et al.*, 2006).

[1] Download simulator: http://paginapessoal.utfpr.edu.br/marceloteixeira/downloads-1/list-of-available-downloads/QLearning.zip.

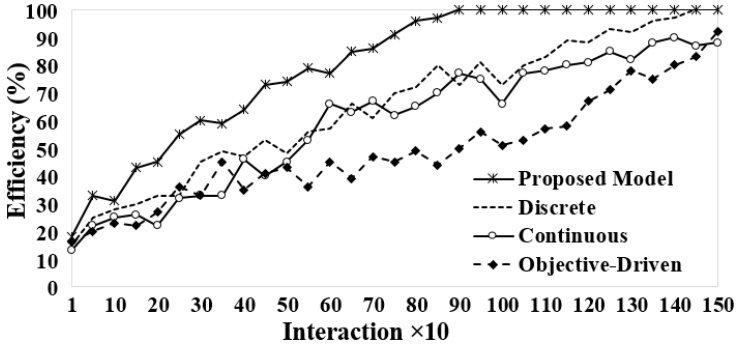


Figure 3. 100 states and **10** agents.

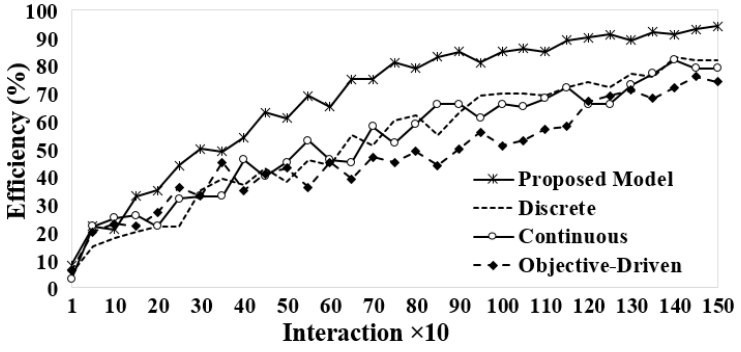


Figure 4. 256 states and **10** agents.

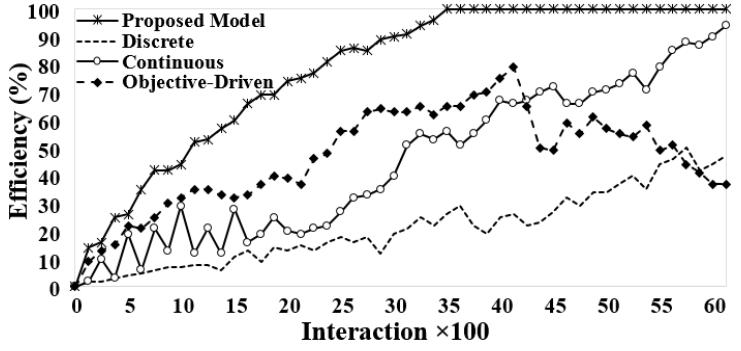


Figure 5. 400 states and **10** agents.

Figures 3 to 5 show that the integrated model was more efficient than the other models individually. In general, it was superior in any interaction phase and substantially reduced the number of interactions needed to find an appropriate action policy. Thus, with the proposed model, the agent's learning is smoother and converges more rapidly to a satisfactory action policy.

For experiments in environments with 100 states (see Table), the number of interactions was reduced on average by 22.8% when 3 agents were used; by 26.7% with 5 agents; and by 35.1% with 10 agents. For environments with 256 states, the corresponding figures were 21.1%, 21.9% and 32.7%, while for environments with 400 states, they were 17.4%, 22.9% and 29.1%. Table 2 shows the im-

provement in efficiency achieved with the integrated model using the most efficient of the three individual models (the continuous model) as reference.

State-space	Number of agents		
	3	5	10
100	18.5%	27.1%	36.9%
256	19.4%	24.9%	32.6%
400	15.2%	25.3%	34.2%

Table 2. Overall comparative analysis.

State-space	Reduction in number of interactions according to the number of agents			
	Reference*	3 *vs* Reference	5 *vs* Reference	10 *vs* Reference
100	1000	22.8%	26.7%	35.1%
256	1500	21.1%	21.9%	32.7%
400	5000	17.4%	22.9%	29.1%

Table 3. Comparison of number of interactions according to the number of agents.

* The reference value corresponds to the result when the continuous interaction model is used.

The overall improvement achieved with the integrated model when the different numbers of agents and state-spaces tested were taken into account was of the order of 27.8%.

5 Conclusions

This paper proposes an integrated model that improves coordination in multi-agent systems by combining features from existing interaction models. The method accelerates convergence of agents' action policies by the order of 27.8% and overcomes important drawbacks observed in earlier approaches (Ribeiro *et al.*, 2008).

The proposed approach aims to improve the way agents share information with each other. In order to transmit and receive information, agents share a cooperative, coordinated interaction model that generally leads to improved action policies. The kernel for establishing optimized information-sharing strategies for multiple agents is therefore the interaction model.

The performance gains for agents that cooperate using the proposed integrated model stem from the fact that *IM*-π^+ is generated from learning values discovered in a collaborative way. This interaction between agents can result in more efficient policies, leading the agents to optimal solutions. Another benefit is the substantial reduction in the number of interactions needed to generate action plans. Nevertheless, it should be noted that the number of messages exchanged between agents increases exponentially with the number of agents and the size of the state-space. This is an important issue and will be investigated in future studies.

Acknowledgment

This research was supported by the Araucária Foundation and the National Council for Scientific and Technological Development (CNPq) under grant numbers 378/2014 and 484859/2013-7, respectively. This research also was supported by the Program for Research Support of UTFPR, - DIRPPG (Diretorate of Research and Post-Graduation) Campus Ponta Grossa and PROPPG.

References

Chakraborty, D. and Stone, P. (2014). *Multiagent learning in the presence of memory-bounded agents*. Autonomous Agents and Multi-Agent Systems, 28(2):182–213.

Chapelle, J., Simonin, O., and Ferber, J. (2002). *How situated agents can learn to cooperate by monitoring their neighbors' satisfaction*. ECAI, 2:68–78.

DeLoach, S. and Valenzuela, J. (2007). *An agent environment interaction model*. In Padgham, L. and Zambonelli, F., editors, Agent-Oriented Software Engineering VII, volume 4405 of Lecture Notes in Computer Science, pages 1–18. Springer Berlin Heidelberg.

Devlin, S., Yliniemi, L., Kudenko, D., and Tumer, K. (2014). *Potential-based difference rewards for multiagent reinforcement learning*. In Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS '14, pages. 165–172, Richland, SC.

Efthymiadis, K. and Kudenko, D. (2015). *Knowledge revision for reinforcement learning with abstract mdps*. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS '15, pages 763–770, Richland, SC.

Grzes, M. and Hoey, J. (2011). *Efficient planning in rmax*. In The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '11, pages 963–970, Richland.

Kaelbling, L. P., Littman, M. L., and Moore, A. P. (1996). *Reinforcement learning: A survey*. Journal of Artificial Intelligence Research, 4:237–285.

Ribeiro, R., Borges, A.P., and Enembreck, F. (2008). *Interaction models for multiagent reinforcement learning*. In Computational Intelligence for Modelling Control Automation, 2008 International Conference on, pages 464–469.

Ribeiro, R. and Enembreck, F. (2013). *A sociologically inspired heuristic for optimization algorithms: A case study on ant systems*. Expert Systems with Applications, 40(5):1814 – 1826.

Ribeiro, R., Enembreck, F., and Koerich, A. (2006). *A hybrid learning strategy for discovery of policies of action*. In Sichman, J., Coelho, H., and Rezende, S., editors, Advances in Artificial Intelligence - IBERAMIASBIA 2006, volume 4140 of Lecture Notes in Computer Science, pages 268–277.

Ribeiro, R., Ronszcka, A. F., Barbosa, M. A. C. and Enembreck, F. (2013). *Updating strategies of policies for coordinating agent swarm in dynamic environments*. In Hammoudi, S., Maciaszek, L. A., Cordeiro, J., and Dietz, J. L. G., editors, ICEIS (1), pages 345–356.

Saito, M. and Kobayashi, I. (2016). *A study on efficient transfer learning for reinforcement learning using sparse coding*. Automation and Control Engineering, 4(4):324 – 330.

Stone, P. and Veloso, M. (2000). *Multiagent systems: A survey from a machine learning perspective*. Autonomous Robots, 8(3):345–383.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning : An Introduction*. MIT Press.

Tesauro, G. (1995). *Temporal difference learning and tdgammon*. Commun. ACM, 38(3):58–68.

Walsh, T. J., Goschin, S., and Littman, M. L. (2010). *Integrating sample-based planning and model-based reinforcement learning*. In Fox, M. and Poole, D., editors, AAAI. AAAI Press.

Watkins, C. J. C. H. and Dayan, P. (1992). *Q-learning*. Machine Learning, 8(3):272–292.

Xinhai, X. and Lunhui, X. (2009). *Traffic signal control agent interaction model based on game theory and reinforcement learning*. In Computer Science Technology and Applications, 2009. IFCSTA'09. International Forum on, volume 1, pages 164–168.

Xuan, P. and Lesser, V. (2002). *Multi-Agent Policies: From Centralized Ones to Decentralized Ones*. Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems, Part 3:1098–1105.

Zhang, C. and Lesser, V. (2010). *Multi-Agent Learning with Policy Prediction*. In Proceedings of the24th AAAI Conference on Artificial Intelligence, pages 927–934, Atlanta.

Zhang, C. and Lesser, V. (2013). *Coordinating Multi-Agent Reinforcement Learning with Limited Communication*. In Ito, J. and Gini, S., editors, Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, pages 1101–1108.