# Collaborative decision in multi agent learning of action models

Christophe Rodrigues[1], Henry Soldano[1,3], Gauvain Bourgne[2], and Céline Rouveirol[1]

[1] L.I.P.N, UMR-CNRS 7030, Univ. Paris-Nord, 93430 Villetaneuse, France
[2] LIP6, Université Pierre et Marie Curie, Sorbonne Universités, 75005 Paris
[3] ABI, Université Pierre et Marie Curie, Sorbonne Universités, 75005 Paris

**Abstract.** Consistency-based collaborative on line learning of relational action models has been recently presented. This framework considers a community of agents, each rationally acting following their relational action model, and assumes that the observed effect of past actions that led an agent to revise its action model can be useful to other agents of the community, potentially speeding up the on-line learning process of agents in the community. In the present article, we discuss how collaboration can be extended in this framework at the individual decision level. More precisely, we first discuss how an agent may predict the effect of some action in its current state through a voting process taking into account all the action models in the community. Second, when an agent fails to produce a plan, using its own action model for reaching a goal from its current state, we study how it can interact with the other agents of the community for selecting an appropriate action that will allow it to revise its action model and progress towards its goal.

## 1 Introduction

Adaptive behavior studies how an autonomous agent can modify its own behavior in order to adapt to a complex, changing and possibly unknown environment. Any adaptive agent needs to simultaneously learn from its experience and act to fulfill various goals. Thus, an adaptive system needs to integrate some kind of *online* learning together with action selection mechanisms. Adaptation within relational representations has been primarily addressed by Relational Reinforcement Learning (RRL) [5] by extending the classical Reinforcement Learning (RL) problem to first order representations. In the *indirect* or *model-based* Reinforcement Learning framework [17] the agent explicitly learns such an *action model*, allowing it to predict the effect of actions, and use it as an input of a symbolic planner, whose output is a plan to execute in order to reach its current goal. *Indirect* RL proved to be very efficient when handling relational – Datalog – representations [3, 8].

A recent work [15, 16] proposed a community of autonomous agents, in which each agent has both capabilities of learning on-line an action model represented as a relational rule set, and planning with this action model [14]. Each agent acts in its environment following its relational action model, and exchanges information with other agents following the general multi agent learning protocol SMILE [1, 2]. Intuitively, the

SMILE protocol is based on a ”consistency maintenance“ process: after revising its current model in order to ensure that the revised model is *consistent* with the observations it has memorized, the agent communicates its revised model to the other members of the community, and possibly receives past observations they have memorized and that contradict the revised model. After a number of such revision/criticism interactions, resulting in a *global revision*, the revised model is stated as *globally consistent* with the observations memorized by all the agents.

We present here a significant extension of this work oriented towards decision and planning at the agent level. First, we consider that, to take some decision, the agent needs to predict the effect of a given action in the current state. Rather than only considering its own action model, it will use a voting procedure taking into account the other agents' individual predictions to make the final prediction. The effect of this voting mechanism is, as expected, particularly useful in the early stages of learning, when agents models are still very diverse. Second, let us consider the case where, during a leaning episode, the agent is unable to build any plan for reaching the current goal it has been assigned from its current state. Rather than performing some action independent from the goal to reach, for instance, by randomly selecting an applicable action, or by implementing more sophisticated, individual, active strategies [8, 14], it will then ask to other agents whether they are able to build such a plan, for this pair of current and goal states, given their action model. In case of success, agents will then send back the first action of their plans, the requesting agent can then select, among all proposed first actions of these plans an action to perform, therefore keeping on orienting its actions towards the assigned goal and reducing the risk of applying illegal actions – i.e., actions without effects.

In this article, we design this new collaborative behavior and investigate during our experiments to what extent they help the agents in being successful in predicting effects and successfully accomplishing tasks. We also consider the effect of these behaviors on the learning speed in the community. The paper is organized as follows : IRALe is described in Section 2, while adaptation of iSMILE to relational action model learning is described in Section 3. Then, Section 4 details the new collaborative behaviors mentioned above. Finally, experiments are detailed and interpreted in section 5.

## 2 Relational Action Learning

In this section, we discuss the relational action model revision algorithm IRALe [14] that each agent uses to revise its relational action model. IRALe learns a STRIPS-like action model as a set of rules from state/action/effect triples. Several rules can be associated to each action, where each rule completely describes the effect of the action in a given context. In this way, the model allows to represent conditional effects. IRALe only memorizes *counter-examples*, namely examples that have provoked a prediction error (the observed effect is not the predicted one) at some point during the model construction. IRALe learns deterministic rules, i.e. once the preconditions of some rule are satisfied, the rule always predicts the same effect. The algorithm is primarily intended to learn in the *realizable* case, i.e., when there exists an exact action model, but has been proven to be accurate when learning in the presence of some amount of noise [13]

**Related work** Learning planning operators has been studied intensively, including the problem of learning the effects of actions, in the context of Relational Reinforcement Learning (RLL). The first model that integrated an incremental action model and policy learning is MARLIE [3]. Learning relational action rules has also been studied in the context of inductive logical programming by Otero et al. [11]. In both cases, the model predicts the value of each possible effect literal (positive or negative) separately. Let us also mention the work of Xu et al. [18] and Mourao et al. [10], that both learn black box models in batch mode. Mourao and colleagues propose an additional step for extracting rules after a black box model has been learned. [19, 20] learn models as sets of rules from plan traces, but they do not incrementally revise this model.

Other works [8, 12] address stochastic effects. Learning is then performed from scratch and needs prior memorization of the whole set of observations.

### 2.1 States, actions, examples and rules

Examples are described as conjunctions of ground literals. Following a STRIPS-like notation, state literals that are not affected by the action are not described in the effect part. The examples are denoted by $x.s/x.a/x.e.add, x.e.del$, with $x.s$ a conjunction of literals, $x.a$ a literal of action and, regarding the effect part, $x.e.add$ a conjunction of positive literals and $x.e.del$ a conjunction of negated literals. Some examples may have an empty effect list (i.e., $x.del = x.add = \emptyset$), accounting for illegal action applications in specific contexts [14].

*Example 1.* Figure 1 displays an example $e$ of the action *move* in a blocks world: $onTable(a), onTable(b), on(c, a)/move(c, b)/on(c, b), \neg on(c, a)$. □

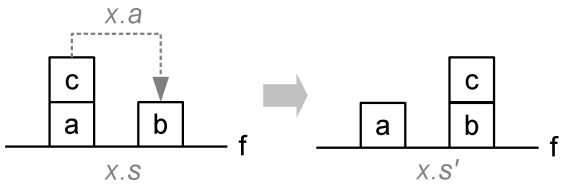



**Fig. 1.** Example of a $move$ action in a simple blocks world

IRALe builds an action model, made of a set of rules $T$ according to a set of observed examples $O$ that have been memorized during the agent history. Each rule $r$ is composed by a precondition $r.p$, an action $r.a$ and an effect $r.e$, and is denoted as $r.p/r.a/r.e$. $r.p$ is a conjunction of positive literals which have to be satisfied to apply the rule, $r.a$ a is a literal defining the performed action, $r.e$ is composed of two sets of literals: $r.e.add$ is the set of literals getting true when the rule is applied, and $r.e.del$ is the set of literals getting false when the rule is applied. According to a rule $r$, an action $r.a$ has no other effects but those described by $r.e$. A default rule is implicitly added to

$T$: for any action $a$, whenever no rule for $a$ applies, the action is predicted to have no effect, i.e. $e.del = e.add = \emptyset$. Note that in an action rule, preconditions and effect may contain (existential) variables that do not appear in the action literal.

### 2.2 Rule covering and contradiction

Matching operations between rules and examples relies on subsumption under Object Identity, denoted as OI-subsumption [6], which is an intuitive partial order relation when learning action rules for planning [12].

Rule matching definition relies on the definitions of pre-matching and post-matching functions. Pre-matching checks whether a given rule may apply to predict the effect of a given action in a given state, and post-matching checks which rule(s) of the action model may explain the effect observed in the example. The question of whether the action model contradicts or is consistent with an example is addressed through the following definitions. Given an example and a rule pre-matching the example, *covering* checks whether the effect part of the example is accurately explained/predicted by the rule, while rule contradiction appears whenever the rule incorrectly predicts the outcomes of the action. The model $T$ needs to be revised whenever the current action model fails to predict the observed effect of some action in the current state. The model includes a *default rule* that predicts the empty effect whenever no other rule pre matches a given state/action pair. In case of failure, the state/action/effect example $x_u$ is said to *contradict* the model, is stated as a counter-example and memorized in $O$. A model $T$ is said *complete* with respect to $O$ whenever no example in $O$ makes $T$ incomplete, and $T$ is said *coherent* with respect to $O$ whenever no example in $O$ makes $T$ incoherent. In both cases of contradiction, the model $T$ needs to be updated to $T'$ in order to preserve coherence and completeness *w.r.t.* $x_u$ and the other past counter-examples in $O$.

### 2.3 Online revision of the action model

The interactions between the agent and the environment produce examples, and when an example contradicts the model, the latter has to be revised by modifying or adding one or several rules. When such a new counter-example $x_u$ is encountered, two kinds of modifications may have to be performed, either generalization or specialization (see [14] for details). We focus here on the generalization process, that takes place in order to preserve completeness of the model, whenever no rule of $T$ pre-matches $x_u$. The rules $r$ of $T$ which are candidates for generalization are such that $r$, up to the generalization of some constants into variables, post-matches $x_u$. Preconditions are then generalized with $x_u$ using least general generalization under OI subsumption. If such a generalization does not contradict any example in $O$ (preserving coherence), $r$ is replaced by the new minimally generalized rule. If no consistent generalization exists, $x_u$ becomes a rule and is added as such to $T$. Finally, $x_u$, as a counter-example, is stored in $O$.

Note that only counter-examples are memorized in $O_i$, i.e., observations that contradicted the current model at some time point. This is sufficient to ensure that learning converges, in the realizable case.

# 3 Action Model Learning in a community of agents

## 3.1 Individualistic collaborative learning

In this section, we present a framework for collective incremental action model learning relying on the SMILE framework [1, 2].

A community of agents, or $n$-MAS, is a set of agents $a_1$, ..., $a_n$. Each agent $a_i$ has a model, here a set of action rules $T_i$, that will be revised during the learning process, together with a set of counter-examples $O_i$. The set of all counter-examples stored in the MAS is denoted by $O$ (for all agents $j \in \{1, .., n\}, E = \cup_{j\in\{1,..,n\}} O_j$). The consistency properties are defined as follows.

**Definition 1**
*- An IRALe agent $a_i$ is* a-consistent *iff $T_i$ is consistent with respect to $O_i$ (see section 2.2), i.e., the agent model $T_i$ correctly predicts observed effects $x.e$ for all counter-examples in $O_i$.*
*- An IRALe agent $a_i$ is* mas-consistent *iff $T_i$ is consistent with respect to $O$, i.e., to all counter-examples stored by agents of the $n$-MAS.*

The global revision $M_s$ is triggered by an agent $a_i$ upon direct observation of a *contradictory* observation $x$, denoted as an *internal counter-example*. This counter-example breaks a-consistency, enforcing revision of $T_i$ into $T_i'$ and is stored in $O_i$. An interaction $I(a_i, a_j)$ between the *learner* agent $a_i$ and an agent $a_j$, acting as a *critic*, is as follows:

1. Agent $a_i$ sends the revision $T_i'$ to $a_j$;
2. Agent $a_j$ checks the revision $T_i'$. If $T_i'$ is a-consistent with respect to its set of counter-examples $O_j$, $a_j$ sends a notification of *acceptance* of $T_i'$ to $a_i$. Otherwise, $a_j$ sends a counter-example $x' \in O_j$, denoted as an *external counter-example* for $a_i$, such that $x'$ contradicts $T_i'$. Then, $x'$ is stored in $O_i$.

An iteration is then composed of a local revision performed by the *learner* agent $a_i$, followed by a sequence of interactions $I(a_i, a_j)$. If an external counter-example $x'$ is transmitted to $a_i$, this triggers a new iteration, starting with a new revision of the learner to restore its a-consistency. When all critics have sent a notification of acceptance of a proposed revision $T_i$, $a_i$ is mas-consistent [1, 2].

We consider here the case where an agent never modifies its own current hypothesis but for internal or external counter-examples. Such agents are denoted as *individualistic*. When the learner agent $a_i$ restores its mas-consistency, its new action model $T_i'$, is now consistent with the new counter-example set $O_\cup\{x\}$ stored in the community. However, the other agents $a_j$ for $j \neq i$, i.e. the critics, are not guaranteed to be consistent with the new counter-example $x$. This leads to define the following *delayed mas consistency* property:

**Definition 2** *Let $O^t = \cup O_j^t$ be the information stored in the MAS at time $t$*

- *An agent $a_i$ is mas-consistent$^t$ iff it is mas-consistent at time $t$.*

- *A $n$-agents community $a_1, \ldots, a_n$ is said to be* delayed consistent *if each agent is consistent$^t$ with $t = min\{t_1, \ldots, t_n\}$ where each $t_i$ is the time of the last revision of agent $a_i$.*
- *A revision mechanism is* delayed mas-consistent *iff an agent applying at some time $t$ this global revision mechanism maintains the delayed consistency of the MAS.*

The following property is then the basis of individualistic learning following the iSMILE protocol [1].

**Proposition 1.** *$M_s$ is* delayed mas-consistent*.*

At a given time $t$, a $n$-MAS $a_1, \ldots, a_n$ is *delayed consistent*. The set of examples the MAS is consistent with is $O^{t_m}$ where $t_m = min(t_i)$ and $t_i$ is the time of last revision of agent each $a_i$ immediately preceding $t$. Note that counter-examples that have been handled during the interval $t - t_m$ have only been seen by subsets of agents of the MAS. We will denote as the *desynchronization effect* the decrease in average accuracy, with respect to a single agent whose memory would be $O^t$, resulting from the delay between the various revision times $t_i$. This desynchronization effect increases as the number of agents increases. A way to reduce this effect by increasing communications has been previously investigated [1].

### 3.2 The agent behavior model

We consider here a community of individualistic agents acting in their own environment. The behavior of an agent $i$ is as follows: at a given moment, the agent has its own current action model $T_i$ and corresponding counter-examples memory $O_i$. It is also provided with some goal it has to reach, as for instance stacking block $b$ on top of block $c$. The agent tries to build a plan, using some planning mechanism. If it succeeds in building a plan, this means that its current action model predicts some effect $\hat{e}$ of the first action $a$ of the plan in the current state $s$. It will then perform this action, observing the effect $e$. If $e = \hat{e}$, this means that the new current state $s'$ is as intended in the plan execution and the agent will apply the next action of the plan. Otherwise, this prediction error defines a new counter-example $x$ with $x.s = s, x.a = a, x.e = e$, the current action model is revised locally and the new model is transmitted to the other agents, therefore triggering the iSMILE $M_s$ global revision process.

If planning fails, we study in this paper two strategies. In the first standard strategy, a random action is selected and performed. Illegal actions, i.e., actions that do not produce any observable effect in the current state, are not filtered out at that step. If the selected action produces some non empty effect, the state changes and planning is attempted again until a new plan can be tentatively executed. A second more informed, *community-aided*, strategy is described hereunder.

## 4 Community-aided effects prediction and action selection

Until now, the role of the community to which an agent belongs, when the agent's on line learns its action model, has been focussed on directly increasing the accuracy of the

agent's model by sending to the agent, on an utility basis, observations that contradicts the agent's current model [15, 16]. In this section, we investigate how an agent could benefit from the other agents action models to improve its own decisions and call this a *community-aided* decision process. This is possible because we consider the iSMILE individualistic collaborative scheme, in which each agent learns its own model and has its own example memory, thus resulting in a diversity of actions models in the community. This corresponds to a form of ensemble learning: several hypotheses are produced and together exploited on new, unlabelled, situations [4].

We consider two simple ways for an agent to benefit from the other agents' models. The first one is a voting procedure, taking into account other agent's action models and allowing the querying agent to perform more accurate predictions. Note that this does not directly affect its learning trajectory. The second way has a direct impact on the agent's learning trajectory. In the agent behavior model as described above, the learning trajectory of the agent depends on a goal to achieve and in its current state it will try to make a plan for that. We propose hereunder the following agent behavior: whenever an agent fails to plan, it will ask other agents to propose some action to perform in its own current state to make it closer to its goal.

In both cases, we will consider as an *informed* agent, with respect to such a query, an agent whose model allows answering to the query. In the first case, such an informed agent has a rule in its own model that predicts a non empty effect for the (current state,action) pair sent by the querying agent. In the second case, an informed agent succeeds in building a plan, according to its own model, for the (current state,goal) pair sent by the querying agent. In what follows, only informed agents' answers to a query are considered by the querying agent. This is effective because the IRALe action models are biased towards either accurately predicting non empty action effects or, by default, predicting empty effects.

### 4.1 Voting to predict action effects to make better decisions

To benefit from other agents' models, the agent may send to all other agents a (current state, action) pair and ask for their predictions, regarding the action's effect, according to their own models. The agent may then perform a *community-aided effect prediction* using a Majority Voting process: when an agent asks the effect of a given action in a given state, the only agents (including itself) considered as informed and allowed to vote are those having some rule prematching the state for this action (i.e., the default rule predicting the empty effect for this action does not apply). The effect predicted by the majority of the voting agents is stated as being predicted by the querying agent.

### 4.2 Selecting an action to get closer to the current goal when single agent planning fails

In case the agent is unable to build a plan for a given (current state,goal) pair, it sends this pair to all other agents and it collects plans from those agents able to build one. The agent will then perform the most frequent action initiating the collected plans. This *community aided action selection*, can be seen as a kind of *active learning*: in case an agent does not know which action to perform to get closer to the goal, it will apply an

action that most *informed* agents advise. It is highly probable that this action does have a non empty effect and that applying this action will provide an informative positive counter-example to the agent. Clearly, applying such an action modifies its learning trajectory. If advices from other agents are reliable, the agent will learn useful rules for solving its goal and will successfully achieve more goals. However will this allow the agent to converge faster (in terms of number of actions to perform) towards an accurate action model? We propose hereunder experiments to study this community aided decision process.

## 5 Experiments

### 5.1 Domains

We are interested here in studying the impacts of community-aided effect prediction and of community-aided action selection in two domains with different characteristics.

The first one is a variant of the blocks world domain in which color predicates for blocks are introduced[4]. This domain requires learning several rules for capturing the impact of blocks color on the effect of the action *move*. In the *colored-blocks* world, when the agent performs the action *move*$(a, b)$, $a$ is actually moved on top of $b$ only if $a$ and $b$ have the same color. Otherwise, $a$ is not moved and its color shifts to the same color as $b$. We run experiments for the 7 blocks with 2 colors domain (7b2c). This domain has been studied already in [15, 16] in terms of average accuracy of the agents' models, average number of plans obtained and number of messages exchanged.

The second domain is the Rover domain from the International Planning Competition[5], it is characterized by a large number actions but a single rule per action and preconditions with many literals for each action. This domain has been used previously to investigate action learning [10], but in a different experimental protocol: in [10], examples are generated independently and randomly with 50% legal and 50% illegal actions, while our agents follow episodes and revise their models on-line, therefore following a state/action/effect. . . /action/effect learning trajectory. In the Rover domain, an agent represents a base monitoring a team of $r$ rovers equipped with $c$ cameras. These rovers navigate on some area of a planet surface, divided in $w$ way-points, and the team has to perform $o$ objectives regarding science gathering operations. The results of the experiments are communicated to the base. A particular rover domain in our experiments is described as the tuple $(r, w, o, c)$ and is denoted as Rover-rwoc. Main features of the two domains, i.e. maximal arities of actions, number of preconditions and effect predicates, total number of actions and rules in the target model are reported hereunder:

| Domain | Actions | | Prec./effects | | #rules |
|---|---|---|---|---|---|
| | #act. | arity | #pred. | arity | |
| *7b2c* | 1 | 2 | 4 | 2 | 7 |
| *Rover* | 9 | 6 | 27 | 3 | 12 |

[4] A problem generator for the colored blocks world problem is available at `http://lipn.univ-paris13.fr/~rodrigues/colam`.

[5] `http://ipc.icaps-conference.org/`

### 5.2 Experimental protocol

Experiments each consist of $N$ runs and are performed for communities of 1 and 20 agents. For each agent, a run is divided into episodes of at most 50 actions each. The agent starts the first episode with an empty model[6]and the current model at the end of an episode is the starting model at the beginning of the next episode. During an episode, the agent explores its environment, starting from a random state, and tries to reach a random goal, both provided by some external controller. The random goal is reachable from the random initial state given the perfect action model. Collaborative learning follows the iSMILE protocol and exploration is performed according to the agent behavior described in section 3.2. Each agent uses FF [7] as a planner. For that purpose, the goal, domain and action model are translated into an equivalent PDDL [9] planning task. The FF planner is then allowed a short time (2s) to find a plan, otherwise planning is stated to have failed. Previously, this protocol has been used to experiment collaborative online learning with IRALE agents within a iSMILE community [15, 16].

We consider two scenarios, namely the Base scenario, in which an agent acts randomly when it fails forming a plan, and the community-aided action selection scenario (section 4). In both scenarios, we measure the standard accuracy together with the majority voting accuracy, both averaged on all agents. These accuracies are compared with the standard accuracy of a single agent and reported versus the *total number of actions performed in the community*, whatever is the number of agents (from 1 to 20). This means that we are interested here in the efficiency of learning with respect to the ressources available to the community and that the single agent here has a clear advantage, as it does not have to cope with the desynchronization effect (see Section 3.1). These accuracies are estimated using a separate test set of examples according to the distribution of effects as met during a trajectory. In both scenarios we also measure the average proportion of goals for which successful plans have been executed versus the *total number of actions performed in the community*. Of course, when there are many agents, each agent meets it efficiency objectives much sooner – in terms of *number of actions per agent* – than an isolated agent.

*Community-aided action effect prediction* We display hereunder in both the 7b2c domain (Figure2(a)) and the Rover domain (Figure 2(b)) the agent average predictive accuracy, in the standard and community-aided effect prediction scenarios, versus the total number of actions performed within the 20-agents community. We also report the corresponding single agent curve. Clearly the single agent curve is better than the 20-agents community curve, thus revealing the desynchronization effect mentioned above, but the informed agents voting procedure is effective in eliminating this effect in both domains. We also observe that the community-aided accuracies are slightly better than the single agent accuracy in both domains, demonstrating therefore some ensemble learning like benefit.

*Community-aided action selection* Figure 3, we report the average ratio of goals achieved by an agent per action performed by the agent versus the total number of actions performed within the 20-agents community as a task oriented measure of learning success.

[6] Except in the Rover domain where communication rules are assumed to be known by the agent.


(a) Effect of voting,7b2c domain (b) Effect of voting, Rover domain

**Fig. 2.** Average agent accuracy in a $n$-agents community with and without community-aided predictions versus total number of actions in the community

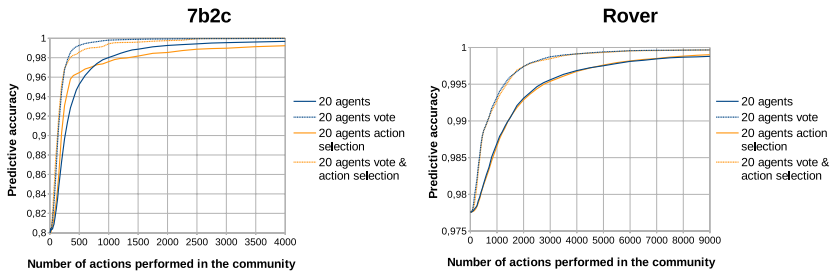
In both the 7b2c (Figure 3(a)) and Rover (Figure 3(b)) domains, we compare the results obtained without and with aided-community action selection. We also report the corresponding single agent curve. In both domains, the community aided action selection strategy demonstrates a positive effect on the ratio of goals achieved per action performed. In the Rover domain, the effect is stronger, most probably because building a plan requires in this domain rules for several actions, and benefitting from the suggestions of the most accurate agents results in more success in achieving goals. Note that, especially in the Rover domain, the community is more efficient in achieving its tasks than the single agent.


(a) Community aided action selection, 7b2c (b) Community aided action selection, Rover

**Fig. 3.** Ratio of number of achieved goals per action in a $n$-agents community with and without community-aided action selection versus total number of actions in the community

*Predictive accuracy in the Community-aided action selection scenario* Community-aided action selection modifies the learning trajectory, i.e. results in a different explo-

ration of states than the basic action selection strategy. The latter strategy can then be considered as an active learning strategy. Figure 4 compares, in both 7b2c (Figure 4(a)) and Rover (Figure 4(b)) domains, the predictive accuracies resulting from the standard learning trajectory with those resulting from the learning trajectory resulting from the community aided action selection strategy. In the 7b2c domain, the community-aided action selection brings a clear benefit at the beginning of the learning curve, and after that, has a negative effect on predictive accuracy. This is observed both when measuring the standard single agent accuracy and when measuring the voting accuracy. To a lesser extent, we observe the same behavior in the Rover domain. This means that focussing on exploitation of models within the community results first in an efficient active exploration, allowing fast learning of actions with non empty effects. However, at some point, some action rules which are not necessary for planning may fail to be learned, therefore resulting in a slower convergence towards the target model. The effect is stronger in the 7b2c case where only one (relational) action is involved, and where missing some unfrequent action rule has no negative effect on planning. Conversely, in the Rover domain, where many actions have to be combined when planning, goal-directed community-aided exploration has to be exhaustive to enhance the ratio of goals achieved by agents, and results in less overfitting.




(a) Effect of voting,7b2c domain (b) Effect of voting, Rover domain

**Fig. 4.** Average agent accuracy in a $n$-agents community with and without community-aided action selection versus total number of actions in the community. The dotted curves represent the accuracies when informed agents vote to predict action effects, in both the standard trajectory and the community aided action selection trajectory.

## 6 Conclusion

We have investigated the impact of collaboration at decision time in a community of agents who revise their relational action model, when communications are limited to the exchanges necessary to ensure learning convergence for each agent in the community, while allowing agents to benefit, when revising their model, from all observations memorized in the community. In such a collaboration, only informed agents, i.e. agents

whose action model is relevant to the decision to be taken, are considered, both regarding predictive accuracy and goal oriented action selection. The experiments show that, first, allowing informed agents to vote does enhance average predictive accuracy of agents, therefore balancing the effect of desynchronisation within a community. Second, allowing informed agents to propose actions to perform when an agent fails to plan significantly increases in the number of tasks successfully achieved during the community history, although it may result in a slower convergence towards a perfect model.

## References


1. G. Bourgne, D. Bouthinon, A. El Fallah Seghrouchni, and H. Soldano. Collaborative concept learning: non individualistic vs individualistic agents. In *Proc. ICTAI*, pages 549–556, 2009.
2. G. Bourgne, A. El Fallah-Seghrouchni, and H. Soldano. Smile: Sound multi-agent incremental learning. In *Proc. AAMAS*, page 38, 2007.
3. T. Croonenborghs, J. Ramon, H. Blockeel, and M. Bruynooghe. Online learning and exploiting relational models in reinforcement learning. In *Proc. IJCAI*, pages 726–731, 2007.
4. Thomas G. Dietterich. Ensemble methods in machine learning. *LNCS*, 1857:1–15, 2000.
5. S. Dzeroski, L. De Raedt, and K. Driessens. Relational reinforcement learning. *Machine Learning*, 43:7–52, 2001.
6. F. Esposito, S. Ferilli, N. Fanizzi, T. M. A. Basile, and N.Di Mauro. Incremental learning and concept drift in inthelex. *Intell. Data Anal.*, 8(3):213–237, 2004.
7. J. Hoffmann. FF: The fast-forward planning system. *The AI Magazine*, 2001.
8. T. Lang, M. Toussaint, and K. Kersting. Exploration in relational domains for model–based reinforcement learning. *JMLR*, 13:3725–2768, 2012.
9. D. McDermott. The 1998 AI planning systems competition. *AI Magazine*, 21(2):35–55, 2000.
10. K. Mourão, L. S. Zettlemoyer, R. P. A. Petrick, and M.Steedman. Learning strips operators from noisy and incomplete observations. In *Proc. UAI*, pages 614–623, 2012.
11. R. P. Otero. Induction of the indirect effects of actions by monotonic methods. In *Proc. ILP 2005*, pages 279–294, 2005.
12. H. M. Pasula, L. S. Zettlemoyer, and L. Kaelbling. Learning symbolic models of stochastic domains. *JAIR*, 29:309–352, 2007.
13. C. Rodrigues, P. Gérard, C. Rouveirol, and H. Soldano. Incremental learning of relational action rules. In *Proc. ICMLA*, pages 451–458. IEEE Press, 2010.
14. C. Rodrigues, P. Gérard, C. Rouveirol, and H. Soldano. Active learning of relational action models. In *Proc. ILP 2011*, volume 7207 of *LNCS*, pages 302–316. Springer, 2012.
15. C. Rodrigues, H. Soldano, G. Bourgne, and C. Rouveirol. A consistency based approach on action model learning in a community of agents. In *Proc. AAMAS*, pages 1557–1558, 2014.
16. Christophe Rodrigues, Henry Soldano, Gauvain Bourgne, and Céline Rouveirol. Multi agent learning of relational action models. In *Proc. ECAI*, pages 1087–1088, 2014.
17. R. S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2:160–163, 1991.
18. J. Z. Xu and J. E. Laird. Instance-based online learning of deterministic relational action models. In *Proc. AAAI*, 2010.
19. Q. Yang, K. Wu, and Y Jiang. Learning action models from plan examples using weighted max-sat. *Artificial Intelligence*, 171(2-3):107 – 143, 2007.
20. H. H. Zhuo, T. A. Nguyen, and S. Kambhampati. Refining incomplete planning domain models through plan traces. In *Proc. IJCAI*, 2013.