Utile Coordination: Learning interdependencies among cooperative agents[1]

Jelle R. Kok§ jellekok@science.uva.nl
Pieter Jan 't Hoen* hoen@cwi.nl
Bram Bakker§ bram@science.uva.nl
Nikos Vlassis§ vlassis@science.uva.nl

§ Informatics Institute, Faculty of Science, University of Amsterdam, The Netherlands
* CWI, Centre for Mathematics and Computer Science, Amsterdam, The Netherlands

Abstract-

We describe Utile Coordination, an algorithm that allows a multiagent system to learn where and how to coordinate. The method starts with uncoordinated learners and maintains statistics on expected returns. Coordination dependencies are dynamically added if the statistics indicate a statistically significant benefit. This results in a compact state representation because only necessary coordination is modeled. We apply our method within the framework of coordination graphs in which value rules represent the coordination dependencies between the agents for a specific context. The algorithm is first applied on a small illustrative problem, and next on a large predator-prey problem in which two predators have to capture a single prey.

1 Introduction

A multiagent system (MAS) consists of a group of interacting autonomous agents [Stone and Veloso, 2000, Vlassis, 2003]. Modeling a problem as a MAS can have several benefits with respect to scalability, robustness and reusability. Furthermore, some problems are inherently distributed and can only be tackled with multiple agents that observe and act from different locations simultaneously.

This paper is concerned with fully cooperative MASs in which multiple agents work on a common task and must learn to optimize a global performance measure. Examples are a team of soccer playing robots or a team of robots which together must build a house. One of the key problems in such systems is *coordination*: how to ensure that the individual decisions of the agents result in jointly optimal decisions for the group.

Reinforcement learning (RL) techniques have been successfully applied in many single-agent domains to learn the behavior of an agent [Sutton and Barto, 1998] . In principle, we can treat a MAS as a 'large' single agent and apply the same techniques by modeling all possible joint actions as single actions. However, the action space scales exponentially with the number of agents, rendering this approach infeasible for all but the simplest problems. Alternatively, we can let each agent learn its policy independently of the other agents, but then the transition and reward models depend on the policy of the other learning agents, which may result in suboptimal or oscillatory behavior.

Recent work (e.g., [Guestrin et al., 2002a, Kok and Vlassis, 2004]) addresses the intermediate case, where the agents coordinate only some of their actions. These 'coordination dependencies' are context-specific. It depends on the state whether an agent can act independently or has to coordinate with some of the other agents. This results in large savings in the state-action representation and as a consequence in the learning time. However, in that work the coordination dependencies had to be specified in advance.

This paper proposes a method to learn these dependencies automatically. Our approach is to start with independent learners and maintain statistics on expected returns based on the action(s) of the other agents. If the statistics indicate that it is beneficial to coordinate, a coordination dependency is added dynamically. This method is inspired by 'Utile Distinction' methods from single-agent RL [Chapman and Kaelbling, 1991, McCallum, 1997] that augment the state space when this distinction helps the agent predict reward. Hence, our method is called the Utile Coordination algorithm.

As in [Guestrin et al., 2002b, Kok and Vlassis, 2004], we use a coordination graph to represent the context-specific coordination dependencies of the agents compactly. Such a graph can be regarded as a sparse representation of the complete state-action space and allows for factored RL updates. Our method learns how to extend the initial coordination graph and represent the necessary coordination dependencies between the agents using derived statistical measures.

The outline of this paper is as follows. In section 2 we review the class of problems and solution methods that we take into consideration. In section 3 we describe the concept of a coordination graph which is used extensively in the remainder of the paper as our representation framework. In section 4 the specific contribution of this paper, the Utile Coordination method, is explained. Experiments are presented in section 5.1 and section 5.2 which illustrate our new method on respectively a small coordination problem and a much larger predator-prey problem. In this popular multiagent problem a number of predators have to coordinate their actions to capture a prey. We show that our method outperforms the non-coordinated individual learners and learns a policy comparable to the method that learns in the complete joint-action space. We conclude in section 6 with some general conclusions and future work.

2 Collaborative multiagent MDPs

In this section we discuss several multiagent RL methods using the *collaborative multiagent MDP* (CMMDP) framework [Guestrin, 2003], which extends the single agent Markov Decision Process (MDP) framework to multiple co-

[1]Appeared in the Proceedings of the IEEE Symposium on Computational Intelligence and Games (CIG'05).

operating agents. Formally, a CMMDP is defined as a tuple $\langle n, S, \mathcal{A}, T, R\rangle$ where n is the number of agents, S is a finite set of world states, $\mathcal{A} = \times_{i=1}^{n}\mathcal{A}_i$ are all possible joint actions defined over the set of individual actions of agent i, $T : S \times \mathcal{A} \times S \to [0, 1]$ is the Markovian[2] transition function that describes the probability $p(s'|s, a)$ that the system will move from state s to s' after performing the joint action $a \in \mathcal{A}$, and $R_i : S \times \mathcal{A} \to \rm I\!R$ is the reward function that returns the reward $R_i(s, a)$ for agent i after the joint action a is taken in state s. A policy is defined as a mapping $\pi : S \to \mathcal{A}$. The objective is to find an optimal policy π^* that maximizes the expected discounted future cumulative reward, or expected return

$$\begin{aligned} Q^*(s, a) &= \max_{\pi} Q^{\pi}(s, a) \\ &= \max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))|\pi, s_0 = s, a_0 = a\right] \end{aligned} \tag{1}$$

for each state s. The expectation operator $E[\cdot]$ averages over reward and stochastic transitions and $\gamma \in [0, 1)$ is the discount factor. Note that the agents try to maximize global returns based on global expected reward $R(s, a) = \sum_{i=1}^{n} R_i(s, a)$ which is the sum of all individual rewards. This is in contrast with stochastic games [Shapley, 1953] where each agent tries to maximize its *own* payoff. If this framework is constrained such that each agent receives the same reward, it corresponds exactly to the MMDP (multiagent MDP) framework of [Boutilier, 1996].

Fig. 1 depicts a small example problem of a collaborative multiagent MDP (and MMDP) with two agents and seven states. In each state every agent selects an individual action from the action set $\mathcal{A}_1 = \mathcal{A}_2 = \{c, d, e\}$, and based on the resulting joint action the agents move to a new state. The next state (and the subsequent reward) depends on the *joint* action only in state s_0. When either of the agents chooses action d, they move to s_1, and after any of the possible joint actions (indicated by $(*, *)$) both agents receive a reward of 0.5 in state s_4. For joint action (e, e) the agents will eventually receive a reward of 3, while for the remaining three joint actions in s_0, the agents will receive a large negative reward of -15. It is difficult for the agents to learn to reach the state s_5 if they learn individually. We discuss this further in section 5.1.

Reinforcement learning (RL) [Sutton and Barto, 1998] can be applied to learn the optimal policy in MDPs. In this paper we consider the case where the transition and reward model are not available, but an agent observes the complete state information. We focus on Q-learning, a well-known learning method for this setting. Q-learning starts with an initial estimate $Q(s, a)$ of the expected discounted future reward for each state-action pair. When an action a is taken in state s, reward r is received and next state s' is observed,

[2]The Markov property implies that the state at time t provides a complete description of the history before time t.

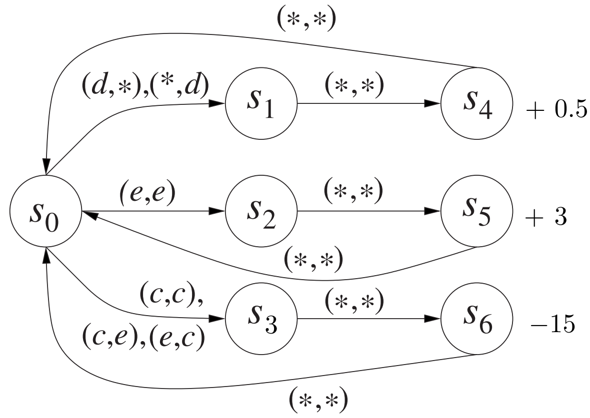

Figure 1: Simple coordination problem with seven states. Only in state s_0 does the joint action has an influence on the next state. The digits on the right represent the given reward to the agents in the corresponding state.

the corresponding Q-value is updated by

$$Q(s, a) := Q(s, a) + \alpha[R(s, a) + \gamma \max_{a'} Q(s', a') - Q(s, a)] \tag{2}$$

where $\alpha \in (0, 1)$ is an appropriate learning rate. Q-learning converges to the optimal Q-function $Q^*(s, a)$ when all state-action pairs are visited infinitely often by means of an appropriate exploration strategy. One of the most common strategies is $\epsilon-$greedy exploration in which at every step the greedy action $a^* = \arg\max_a Q(s, a)$ is selected with probability $1 - \epsilon$ and a (random) non-greedy action is selected with probability ϵ. In the above description of RL for MDPs, we assumed a tabular representation of the Q-table in which all state-action pairs are explicitly enumerated. Next, we will discuss three methods to apply RL to a CMMDP, which has multiple agents and joint actions.

At one extreme, we can represent the system as one large agent in which each joint action is modeled as a single action, and then apply single agent Q-learning. In order to apply such a *joint action MDP* (JAMDP) learner a central controller represents the complete JAMDP Q-function and informs each agent of its individual action, or all agents represent the complete Q-function separately, and execute their own individual action[3]. This approach leads to the optimal policy, but is infeasible for large problems since the joint action space, which is exponential in the number of individual actions, becomes intractable both in terms of storage, as well as in terms of exploration[4]. In the example of Fig. 1, this approach stores a Q-value for each of the nine joint actions in state s_i.

At the other extreme, we have *independent learners* (IL)

[3]The problem of determining the joint (possible exploration) action can be solved by assuming that all agents are using the same random number generator and the same seed, and that these facts are common knowledge among the agents [Vlassis, 2003].

[4]Note that function approximations techniques can also be used to deal with large state-action spaces. However, they are more often applied to large state spaces, instead of large action spaces because of the difficulty of generalizing over different actions.

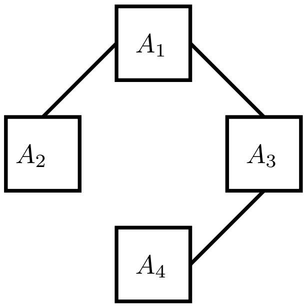

Figure 2: An example coordination graph for a 4-agent problem. Each node represents an edge, while the edges define the coordination dependencies.

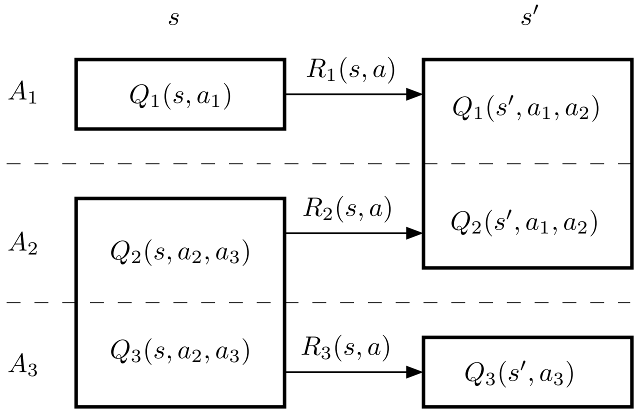

Figure 3: Example representation of the Q components of three agents for a transition from state s to state s'. In state s agent 2 and 3 have to coordinate their actions, while in state s' agent 1 and 2 have to coordinate their actions.

[Claus and Boutilier, 1998] who ignore the actions and rewards of the other agents and learn their policies independently. This results in a large reduction in the state-action representation. However, the standard convergence proof for single agent Q-learning does not hold in this case, since the transition model for each agent depends on the unknown policy of the other learning agents. This can result in oscillatory behavior or convergence to a suboptimal policy. As we will see in section 5.1, independent learners converge to the suboptimal policy (d, d) for state s_0 in the example problem of Fig. 1, since the penalty for incorrect coordination has a large negative influence on the individual Q-values for actions c and e.

The next section is devoted to an intermediate approach, introduced in [Kok and Vlassis, 2004], in which the agents only coordinate their actions in certain predefined states. In section 4 we extend this method to learn the states in which coordination is needed.

3 Coordination Graphs

In this section, we will describe context-specific coordination graphs (CGs) [Guestrin et al., 2002b] which can be used to specify the coordination dependencies for subsets of agents. In a CG each node represents an agent, while an edge defines an action dependency between two agents. For example, the graph in Fig. 2 shows a CG for a 4-agent problem in which agent A_3 and A_4 have to coordinate, and A_1 has to coordinate with both A_2 and A_3. Since only connected agents have to coordinate their actions, the global coordination problem is decomposed into a number of local problems. The dependencies between the agents are specified using *value rules* of the form $\langle \rho; c : v \rangle$, where c is defined as the context (defined over possible state-action combinations) and the payoff $\rho(c) = v$ is a (local) contribution to the global payoff. This is a much richer representation than the IL or JAMDP variants, since it allows us to represent all possible dependencies between the agents in a context-specific manner. In 'coordinated' states, where actions of the agents depend on each other, the rules are based on joint actions, while for 'uncoordinated' states they are based on the individual actions of an agent. In our running example of Fig. 1, value rules for all possible joint actions, e.g., $\langle \rho; s_0 \wedge a_1 = e \wedge a_2 = e : v \rangle$ are needed in state s_0, while value rules based on individual actions, e.g., $\langle \rho; s_1 \wedge a_1 = a : v \rangle$, are a rich enough representation for all other states. The rules in a CG can be regarded as a sparse representation of the complete state-action space since they are defined over subsets of all state and action variables.

In order to compute the joint action with maximum total payoff, the agents first condition on the context and eliminate all rules that are inconsistent with the current state. Then a *variable elimination* algorithm is applied in which each agent first solves a local maximization problem (which depends only on its neighbors in the graph) and then communicates the resulting conditional strategy to one of its neighbors. After this, the communicating agent is eliminated from the graph. This procedure continues until only one agent remains, which then determines its contribution to the optimal joint action based on the conditional strategies of all agents. Thereafter, a pass in the reverse order is performed in which all eliminated agents fix their strategies based on the selected actions of their neighbors. After completion of the algorithm, the selected joint action corresponds to the optimal joint action that maximizes the sum of the payoff of the applicable value rules for the current state. Although the elimination order does not have an effect on the outcome of the algorithm, it does have an effect on the needed computation time. We refer to [Guestrin et al., 2002b] for details.

We applied coordination graphs successfully in our RoboCup simulation team by manually specifying both the coordination dependencies and the associated payoffs using value rules [Kok et al., 2004]. This resulted in the world champion title in the RoboCup-2003 soccer simulation league, illustrating that such a representation can capture very complex and effective policies.

In [Kok and Vlassis, 2004] coordinated behavior is learned using the concept of CGs and variations of Q-learning. We will refer to this method as 'Sparse Cooperative Q-learning'. In that work a predefined set of value rules is specified that captures the coordination dependencies of the system. At each time step the global Q-value equals the sum of the local Q-values of all n agents. The local Q-value, $Q_i(s,a)$ of an agent i depends on the payoff of the value rules in which agent i is involved and that is consistent with the given state-action pair (s,a):

$$Q_i(s,a) = \sum_j \frac{\rho_j^i(s,a)}{n_j}, \tag{3}$$

where each payoff is divided proportionally over the n_j involved agents. Such a representation of $Q_i(s,a)$ can be regarded as a linear expansion into a set of basis functions ρ_j^i, each of them peaked on a specific state-action context which may potentially involve many agents. In the sparse cooperative Q-learning method, the 'weights' of these basis functions (the values of the rules) are updated as follows:

$$\rho_j(s,a) := \rho_j(s,a) + \alpha \sum_{i=1}^{n_j} [R_i(s,a) + \gamma Q_i(s', a^*) - Q_i(s,a)]. \tag{4}$$

Note that each rule is updated based on their local contribution for the global optimal joint action. In order to compute this joint action $a^* = \arg\max_a Q(s,a)$ that maximizes the sum of the (local) payoffs for state s, the variable elimination algorithm is applied. From this, the agents can determine their contribution $Q_i(s', a^*)$ to the total payoff. A rule is updated by adding the individual reward and individual expected future reward of each agent involved in the rule, similar to Eq. (2). Effectively, each agent learns to coordinate with its neighbors, in a context-specific manner.

As an example, assume we have the following set of value rules[5]:

$$\begin{array}{ll}
\langle \rho_1 \,;\, a_1 \wedge s & : v_1 \rangle \\
\langle \rho_2 \,;\, \overline{a}_1 \wedge a_2 \wedge s' & : v_2 \rangle \\
\langle \rho_3 \,;\, a_1 \wedge \overline{a}_2 \wedge s' & : v_3 \rangle \\
\langle \rho_4 \,;\, a_1 \wedge \overline{a}_2 \wedge s & : v_4 \rangle \\
\langle \rho_5 \,;\, a_2 \wedge a_3 \wedge s & : v_5 \rangle \\
\langle \rho_6 \,;\, \overline{a}_3 \wedge s' & : v_6 \rangle
\end{array}$$

Furthermore, assume that $a = \{a_1, a_2, a_3\}$ is the performed joint action in state s and $a^* = \{a_1, \overline{a}_2, \overline{a}_3\}$ is the optimal joint action found with the variable elimination algorithm in state s'. After conditioning on the context, the rules ρ_1 and ρ_5 apply in state s, whereas the rules ρ_3 and ρ_6 apply in state s'. This is graphically depicted in Fig. 3. Next, we use Eq. (4) to update the value rules ρ_1 and ρ_5 in state s as follows:

$$\begin{aligned}
\rho_1(s,a) &= v_1 + \alpha[R_1(s,a) + \gamma \frac{v_3}{2} - \frac{v_1}{1}] \\
\rho_5(s,a) &= v_5 \;+\; \alpha[R_2(s,a) + \gamma \frac{v_3}{2} - \frac{v_5}{2} + \\
&\qquad R_3(s,a) + \gamma \frac{v_6}{1} - \frac{v_5}{2}].
\end{aligned}$$

Note that in order to update ρ_5 we have used the (discounted) Q-values of $Q_2(s', a^*) = v_3/2$ and $Q_3(s', a^*) = v_6/1$. Furthermore, the component Q_2 in state s' is based on a coordinated action of agent A_2 with agent A_1 (rule ρ_3), whereas in state s agent A_2 has to coordinate with agent A_3 (rule ρ_5).

In the above description, we assume that the coordination dependencies among the agents are specified beforehand. In the next section we describe how these dependencies can be *learned* automatically by starting with an initial set of (individual) rules based on individual actions and dynamically adding rules for those states where coordination is found to be necessary.

4 Utile Coordination

Previous work using coordination graphs assumes a known CG topology. In this section we describe our method to learn the coordination dependencies among the agents automatically. Our approach builds on the ideas of Chapman & Kaelbling's [Chapman and Kaelbling, 1991] and McCallum's [McCallum, 1997] adaptive resolution RL methods for the single agent case. These methods construct a partitioning of an agent's state space based on finding so-called 'Utile Distinctions' [McCallum, 1997] in the state representation. These are detected through statistics of the expected returns maintained for hypothesized distinctions. For every state, this method stores the future discounted reward received after leaving this state and relates it to an incoming transition (the previous state). When a state is Markovian with respect to return, the return values on all incoming transitions should be similar. However, if the statistics indicate that the returns are significantly different, the state is split to help the agent predict the future reward better. This approach allows a single agent to build an appropriate representation of the state space.

In our Utile Coordination algorithm, we take a similar approach. The main difference is that, instead of keeping statistics on the expected return based on incoming transitions, we keep statistics based on the performed actions of the other agents. The general idea is as follows. The algorithm starts with independent uncoordinated learners[6], but over time learns, based on acquired statistics, where the independent learners need to coordinate. If the statistics indicate there is a benefit in coordinating the actions of independent agents in a particular state, that state becomes a coordinated state. In the CG framework, this means new coordinated value rules are added to the coordinated graph.

Statistics of the expected return are maintained to determine the possible benefit of coordination for each state. That is, in each state s where coordination between two (or more) agents in a set I is considered, a sample of the 'combined return', $\hat{Q}_I(s, a_I)$, is maintained after a joint action a is performed. The combined return is an approximation of the expected return that can be obtained by the involved agents in I and equals the sum of their received individual reward and their individual contribution $Q_i(s', a^*)$ to the

[5] Action a_1 corresponds to $a_1 = true$ and action $\overline{a}_1$ to $a_1 = false$.

[6] Note that it is also possible to start with an initial CG incorporating coordination dependencies that are based on prior domain-specific knowledge.

maximal global Q-value of the next state as in Eq. (4):

$$\hat{Q}_I(s, a_I) = \sum_{i \in I} \hat{Q}_i(s, a) = \sum_{i \in I} [R_i(s, a) + \gamma Q_i(s', a^*)]. \tag{5}$$

These samples are stored with the performed action a_I. For each of these joint actions, the expected combined return can be estimated by computing the mean, $\bar{Q}_I(s, a_I)$, of the last M samples[7]. These statistics are never used to change the agent's state-action values, but are stored to perform a statistical test at the end of an m-length trial to measure whether the largest expected combined return for a state s, $\max_{a_I} \bar{Q}_I(s, a_I)$ (with variance $\sigma^2_{\max}$), differs significantly from the expected combined return $\bar{Q}_I(s, a^*_I)$ (with variance σ^2_*). The latter is the return obtained when performing the greedy joint action a^*_I in the state s (and thus corresponds to the actually learned policy). This greedy joint action can be found using the variable elimination algorithm. Note that initially, when all states are uncoordinated, a^*_I corresponds to the vector of individually greedy actions: $a^*_i = \arg\max_{a_i} Q_i(s, a_i)$.

We use the t-test [Stevens, 1990] as the statistical test to compare the two values:

$$t = \frac{\max_{a_I} \bar{Q}_I(s, a_I) - \bar{Q}_I(s, a^*_I)}{\sqrt{[(2/M)((M-1)\sigma^2_{\max} + (M-1)\sigma^2_*)/(2M-2)]}} \tag{6}$$

with $(2M - 2)$ degrees of freedom. From this value the level of significance, p, is computed indicating the probability of rejecting the null hypothesis (the two groups are equal) when it is true.[8]

An additional statistical *effect size measure* d determines whether the observed difference is not only statistically significant, but also sufficiently large. In this paper d is similar to standard effect size measures [Stevens, 1990], and it relates the difference in means to the observed maximum and minimum reward available in the task:

$$d = \frac{\max_{a_I} \bar{Q}_I(s, a_I) - \bar{Q}_I(s, a^*_I)}{r_{max} - r_{min}}. \tag{7}$$

If there is a statistically significant difference ($p < P$) with sufficient effect size ($d > D$), there is a significant benefit of coordinating the agents' actions in this state: apparently the current CG leads to significantly lower returns than the possible returns when the actions are coordinated. This can occur in the situation where one specific joint action will produce a high return but all other joint actions will get a substantially lower return (see the example at the end of section 2). Since the agents select their actions individually they will only occasionally gather the high return. However, when the stored statistics (based on joint actions) are compared with the current policy, this produces a statistical difference indicating that it is beneficial to change this state into a coordinated state. In our CG framework, the value rules based on individual actions are replaced by value rules based on joint actions for this particular state. The value of each new rule $\rho(s, a_I)$ is initialized with the learned value $\bar{Q}_I(s, a_I)$.

As coordination rules are added, the samples of $\hat{Q}_I(s, a_I)$ may correspond to *joint* actions of two agents that now coordinate in a particular state, such that now coordination between three or more agents can be learned. Alternatively, 3D, 4D, etc., tables can be constructed for three or more independent learners to test for coordination benefits when coordination between only 2 agents is not beneficial. In any case, the statistical test always looks at only two estimates of expected combined return: $\max_{a_I} \bar{Q}_I(s, a_I)$ and $\bar{Q}_I(s, a^*_I)$.

In very large state-action spaces, memory and computation limitations will make it infeasible to maintain these statistics for each state. In fact, those are the most interesting cases, because there learning sparse coordination is most useful, as opposed to the full coordination done by JAMDP learners. It then makes sense to use a heuristic 'initial filter' which detects potential states where coordination might be beneficial. The full statistics on combined returns are then only maintained for the potential interesting states detected by the initial filter. In this way, large savings in computation and memory can be obtained while still being able to learn the required coordination. One useful heuristic for filtering may be to compare $Q_i(s, a)$ to the *mode* of the last M samples of $\hat{Q}_i(s, a)$ stored in a small histogram. If they are sufficiently different, this indicates multi-modality of expected returns for this agent i, which may mean a potentially dependence on other agents' actions. In this paper, the emphasis is on showing the validity of the Utile Coordination algorithm and its learning efficiency compared to independent learners and JAMDP learners. Therefore, in the experiments reported below no heuristic initial filter is used and statistics are stored for every state.

5 Experiments

In this section, we apply the Utile Coordination algorithm to two problems: the example of section 2 and to the much larger predator-prey domain.

5.1 Small Coordination problem

In this section, we apply our algorithm to the simple intuitive problem depicted in Fig. 1 and compare it to the two Q-learning methods mentioned in section 2, the JAMDP learners and the Independent Learners (ILs). The latter only keep Q-values for their individual actions and therefore $42\,(= 2 \cdot 7 \cdot 3)$ Q-values are stored in total. The JAMDP learners model the joint action for every state resulting in $63\,(= 7 \cdot 3^2)$ Q-values. Just as with the ILs, our Utile Coordination approach starts with value rules based on individual actions; but it checks, after $m = 1000$ steps, for every state whether the action of the other agent should be

[7] In our experiments, we used $M = 10$.

[8] Other statistical tests that compare two groups are possible. In particular, nonparametric tests may be used, because assumptions of normality and homogeneity of variance may be violated. However, the t-test is fairly robust to such violations when group sizes are equal (as in our case) [Stevens, 1990].

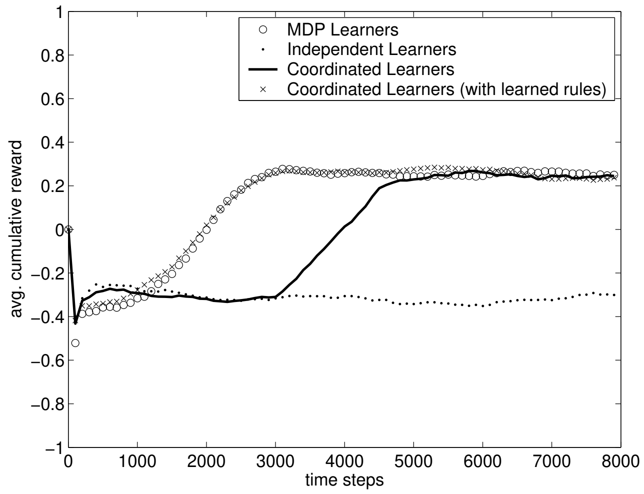

Figure 4: Running average of total cumulative reward of the previous 1500 time steps (including exploration) for the different Q-learners in the problem from section 2. Results are averaged over 30 runs.

incorporated. We use an ϵ-greedy exploration step[9] of 0.3, a learning rate $\alpha = 0.25$, and a discount factor $\gamma = 0.9$. For the parameters in our Utile Coordination approach we use a significance level $P = 0.05$ and an effect size $D = 0.01$.

Fig. 4 shows the running average of the cumulative reward (including exploration) for the three different Q-learning approaches. The independent learners do not converge to the optimal policy since the actions resulting in a low reward have a large negative impact on the Q-values corresponding to the individual actions of the optimal joint action. The JAMDP learners do not have this problem, since they model each joint action and quickly learn to converge to the optimal policy. Since our Utile Coordination approach starts with individual value rules, the learning curve resembles that of the independent learners in the beginning. However, after the third trial (3000 time steps), appropriate coordination is added for state s_0 in all 30 runs, and thereafter the system converges to the optimal policy. Fig. 4 also shows that simulation runs that start with the learned coordination dependencies found with the Utile Coordination approach produce identical results as the JAMDP learners. Although the learned representation of the Utile Coordination approach uses a sparser representation, both methods quickly converge to the optimal policy.

5.2 Predator-prey problem

In this section, we apply our Utile Coordination algorithm to the predator-prey problem. We concentrate on a problem where two predators have to coordinate their actions in order to capture a prey in a 10×10 toroidal grid. Each agent can either move to one of its adjacent cells or remain on its current position. In total this yields $242,550$ (joint) state-action pairs. All agents are initialized at random positions at the beginning of an episode. An episode ends when the prey is captured. This occurs when both predators are located in cells adjacent to the prey and only one of the two agents moves to the location of the prey and the other remains on its current position. Fig. 5 shows an example grid in which the predators will capture the prey when either the predator north of the prey, or the prey east of the prey will move to the prey position and the other predator will remain on its current position. A predator is penalized and placed on a random position on the field when it moves to the prey position without coordinating with the other predator, or moves to the same cell as the other predator. The predators thus have to coordinate their actions in all states in which they are close to each other or when they are close to the prey. In all other states, the agents can act individually. The prey behavior is fixed: it remains on its current position with a probability of 0.2 and otherwise moves moves to one of its free adjacent cells with uniform probability.

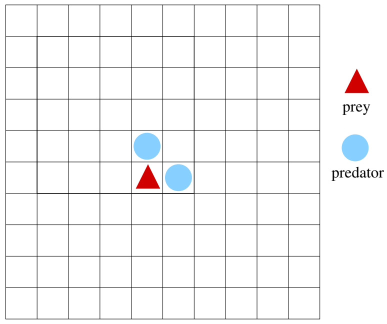

Figure 5: Graphical representation of a 10×10 grid with two agents and one prey. This situation shows a possible capture position for two predators. The prey is only captured when one of the two agents moves to the prey position and the other remains on its current position.

Just as with the small coordination problem, we will apply our method and compare it with the two other Q-learning approaches. Each predator i receives an (individual) reward $R_i = 37.5$ when it helps to capture the prey,

[9]Note that for fair comparison of the results, independent learners explore jointly in all experiments. That is, with probability ϵ both agents select a random action, which is more conservative than each agent independently choosing a random action with probability ϵ.

a reward of -25.0 when it moves to the prey without support, a reward of -10 when it collides with another predator, and a reward of -0.5 in all other situations to motivate the predators to capture the prey as quickly as possible. We use an ϵ-greedy exploration step of 0.3, a learning rate $\alpha = 0.25$, and a discount factor $\gamma = 0.9$. Again, we use a significance level $P = 0.05$ and and an effect size $D = 0.01$ for the Utile Coordination approach. Statistical tests to determine coordination are performed after every $m = 20,000$ episodes.

Fig. 6 shows the capture times for the learned policy during the first $400,000$ episodes for the different methods (running average of the capture times of the last 300 episodes is shown) and includes the exploration steps taken by the agents. The results are averaged over 10 runs. The IL approach does not converge to a stable policy but keeps oscillating; the Q-values for the individual actions for capturing the prey are decreased substantially when an action is performed that results in an illegal movement to the prey. The JAMDP learners model these dependencies explicitly in every state which results in convergence to the optimal policy. Our Utile Coordination approach initially does not take these dependencies into account and follows the curve of the independent learners. However, after the end of the first trial (episode $20,000$), the agents add coordinated value rules for the states in which the gathered statistics indicate that coordination is beneficial, and immediately the capture times decrease as is visible in Fig. 6. Thereafter, the average capture times keep decreasing slowly as more fine-grained coordination dependencies are added and the agents learn in the updated coordination graph structure. At the end, while new value rules are added, the found policy is similar to the policy found by the JAMDP Learners.

Table 1 shows the final capture times and the number of Q-values needed to represent the state-action space for each method. For the Utile Coordination approach on average $457.90\,(\pm 53.4)$ out of 9702 states were found to be statistically significant and added as coordinated states. This is in contrast with the $1,248$ manually specified states in [Kok and Vlassis, 2004], where coordinated rules were added for all states in which the predators where within two cells of each other or both within two cells of the prey. This difference is caused by the fact that for many states where a collision is possible and the agents have to coordinate their actions, the agents are able to learn how to avoid the collision independently and no specific coordination rule is needed.

When the learned coordination dependencies of the Utile Coordination approach are used to learn the policy of the agents, the learning curve is similar to that of the JAMDP learners. However, the latter needs a larger representation to store the Q-values. In this experiment, this does not result in a negative influence on the learning curve because of the relative small joint action-size. However, for lager problems with more agents (and more agents dependencies), this will be more severe.

Both the Utile Coordination approach and the approach based on the learned rules converge to a slightly higher capture time than that of the JAMDP Learners, indicating that coordinating in some states, not statistically significant for the Utile Coordination approach, has a very small positive influence on the final result.

6 Conclusion and future work

This paper introduced the Utile Coordination algorithm, which starts with independent, non-coordinating agents and learns automatically where and how to coordinate. The method is based on maintaining statistics on expected returns for hypothesized coordinated states, and a statistical test that determines whether the expected return increases when actions are explicitly coordinated, compared to when they are not. We implemented this method within the framework of coordination graphs, because of its attractive properties of representing compactly and efficiently the agents' state-action space, values, RL updates, and context-specific coordination dependencies. In this context, the method can be understood as testing, for a given CG, the standard CG assumption that the overall return $Q(s, a)$ is simply the sum of the individual components $Q_i(s, a)$. If this assumption is violated, the algorithm adds appropriate value rules to make the resulting CG adhere to the assumption.

There are many avenues for future work. As described before, maintaining the complete statistics for all states is not computationally feasible for large CMMDPs. Heuristic initial filters should be investigated in detail, such that the Utile Coordination algorithm can be applied to such large problems. In particular, tasks with many interacting agents should be investigated, as these are the tasks where the problem of maintaining full statistics is most obvious, and where at the same time the advantage of Utile Coordination over JAMDP learners, in terms of space and learning time, will be more pronounced.

Heuristic initial filters are not the only way to deal with large state spaces. An equally important, orthogonal possibility is a variation of the Utile Coordination algorithm based on more sophisticated, e.g., factorial or relational, state representations. This should combine well with coordination graphs, because they were explicitly designed for such state representations. An individual agent would then be able to represent only its own individual view of the environment state. Furthermore, it could for instance learn to coordinate with another agent 'when the other agent is near', rather than having to represent explicitly all environment states when it is near the other agent and learn to coordinate separately for all those states.

Finally, we note that it should be possible to use the same statistical tests to allow *pruning* of coordination rules if they turn out to be of little use. Also, a user may inject some coordination rules into the algorithm based on a priori knowledge, and the system can subsequently learn additional rules or prune superfluous user-inserted rules. In this way, a priori knowledge and learning can be combined fruitfully.

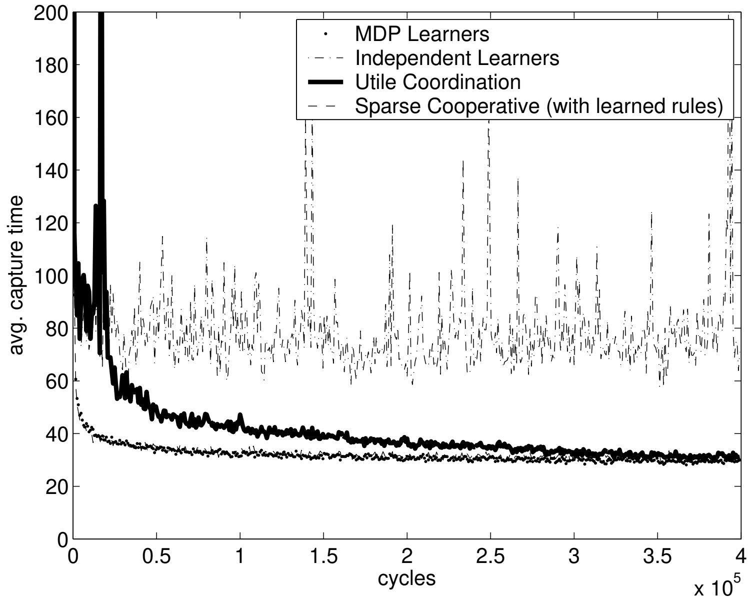

Figure 6: Running average of the capture times (over the last 300 episodes) for the learned policy of the four different methods during the first $400,000$ episodes. Results are averaged over 10 runs. Note that the learning curve of the JAMDP and the curve based on the learned representation (bottom curve) overlap and are almost similar.

Table 1: Average capture time after learning (averaged over the last 1,000 episodes) and the number of state-action pairs for the different methods.

Method	**avg. time**	**#Q-values**
Independent learners	68.77	97,020
JAMDP Learners	29.71	242,550
Utile Coordination	30.57	105,868
Sparse Cooperative (using learned rules)	30.93	105,868

Bibliography

[Boutilier, 1996] Boutilier, C. (1996). Planning, learning and coordination in multiagent decision processes. In *Proc. Conf. on Theoretical Aspects of Rationality and Knowledge*.

[Chalkiadakis and Boutilier, 2003] Chalkiadakis, G. and Boutilier, C. (2003). Coordination in multiagent reinforcement learning: A bayesian approach. In *Proc. of the 2nd Int. Joint Conf. on Autonomous agents and multiagent systems*, pages 709–716, Melbourne, Australia. ACM Press.

[Chapman and Kaelbling, 1991] Chapman, D. and Kaelbling, L. P. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In Mylopoulos, J. and Reiter., R., editors, *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 726–731, San Mateo, Ca. Morgan Kaufmann.

[Claus and Boutilier, 1998] Claus, C. and Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. In *Proc. 15th Nation. Conf. on Artificial Intelligence*, Madison, WI.

[Guestrin, 2003] Guestrin, C. (2003). *Planning Under Uncertainty in Complex Structured Environments*. PhD thesis, Computer Science Department, Stanford University.

[Guestrin et al., 2002a] Guestrin, C., Lagoudakis, M., and Parr, R. (2002a). Coordinated reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*.

[Guestrin et al., 2002b] Guestrin, C., Venkataraman, S., and Koller, D. (2002b). Context-specific multiagent coordination and planning with factored MDPs. In *Proc. 8th Nation. Conf. on Artificial Intelligence*, Edmonton, Canada.

[Kok et al., 2004] Kok, J. R., Spaan, M. T. J., and Vlassis, N. (2004). Noncommunicative multi-robot coordination in dynamic environments. *Robotics and Autonomous Systems*. In press.

[Kok and Vlassis, 2004] Kok, J. R. and Vlassis, N. (2004). Sparse Cooperative Q-learning. In Greiner, R. and Schuurmans, D., editors, *Proc. of the 21st Int. Conf. on Machine Learning*, pages 481–488, Banff, Canada. ACM.

[McCallum, 1997] McCallum, R. A. (1997). *Reinforcement Learning with Selective Perception and Hidden State*. PhD thesis, University of Rochester, Computer Science Department.

[Shapley, 1953] Shapley, L. (1953). Stochastic games. *Proceedings of the National Academy of Sciences*, 39:1095–1100.

[Stevens, 1990] Stevens, J. P. (1990). *Intermediate statistics: A modern approach*. Lawrence Erlbaum.

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

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

[Vlassis, 2003] Vlassis, N. (2003). A concise introduction to multiagent systems and distributed AI. Informatics Institute, University of Amsterdam. http://www.science.uva.nl/˜vlassis/cimasdai.