Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

SEN

COllective INtelligence with sequences of actions

P.J. 't Hoen, S.M. Bohte

REPORT SEN-R0302 JUNE 30, 2003

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3711

COllective INtelligence with Sequences of Actions

Coordinating actions in Multi-Agent Systems

Pieter Jan 't Hoen   Sander M. Bohte
{hoen,sbohte}@cwi.nl
CWI, Centre for Mathematics and Computer Science
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
The design of a Multi-Agent System (MAS) to perform well on a collective task is non-trivial. Straightforward application of learning in a MAS can lead to sub optimal solutions as agents compete or interfere. The COllective INtelligence (COIN) framework of Wolpert et al. proposes an engineering solution for MASs where agents learn to focus on actions which support a common task. As a case study, we investigate the performance of COIN for representative token retrieval problems found to be difficult for agents using classic Reinforcement Learning (RL). We further investigate several techniques from RL (model-based learning, Q(λ)) to scale application of the COIN framework. Lastly, the COIN framework is extended to improve performance for sequences of actions.

2000 Mathematics Subject Classification: 68M14,68T05
1998 ACM Computing Classification System: I.2.6,I.2.11
Keywords and Phrases: Reinforcement Learning, cooperative Multi-Agent Systems
Note: Work carried out under project SEN4 “Evolutionary Systems and Applied Algorithmics”.

1. Introduction

As argued by Wellman [14, 15], a computational problem can be considered as a resource allocation problem. Borrowing from the insights of economics, it is however becoming increasingly clear that few concepts for resource allocation scale well with increasing complexity of the problem domain. In particular, centralized allocation planning can quickly reach a point where the design of satisfying solutions becomes complex and intractable. Conceptually, an attractive option is to devise a distributed system where different parts of the system each contribute to the solution for the problem. Embodied in a so-called distributed Multi-Agent System (MAS), the aim is thus to elicit “emergent” behavior from a collection of individual agents that each solve a part of the problem.

This emergent behavior relies implicitly on the notion that the usefulness of the system is expected to increase as the individual agents optimize their behavior. A weak point of such systems has however long been the typical bottom-up type of approach: researchers first build an intuitively reasonable system of agents and then use heuristics and tuned system parameters such that – hopefully – the desired type of behavior emerges from running the system. Only recently has there been work on more top-down type of approaches to establish the conditions for MASs such that they are most likely to exhibit good emergent behavior [1, 4, 2].

In typical problem settings, individual agent in the MAS contribute to some part of the collective through its private actions. The joint actions of all agents derive some reward from the outside world. To enable local learning, this reward has to be divided amongst the individual agents where each agent aims to increase its received reward by some form of learning. However, unless special care is taken as to how this reward is shared, there is a risk that agents in the collective work at cross-purposes. For example, agents can reach sub-optimal solutions by competing for scarce resources or by inefficient task distribution among the agents as they each only consider their own goals (e.g. a Tragedy of the Commons [3]).

The COllective INtelligence (COIN) framework, as introduced by Wolpert et al., suggests how to engineer (or *modify*) the rewards an agents receives for its actions (and to which it adapts to optimize) in *private utility functions*. Optimization of each agent's private utility here leads to increasingly effective emergent behavior of the collective, while discouraging agents from working at cross-purposes.

In particular, the work by Wolpert et al. explores the conditions sufficient for effective emergent behavior for a collective of independent agents, each employing "sufficiently powerful" Reinforcement Learning (RL) for optimizing their private utility. These conditions relate to (i) the learnability of the problem each agent faces, as obtained through each individual agent's private utility function, (ii) the relative "alignment" of the agents' private utility functions with the utility function of the collective (the *world utility*), and lastly (iii) the learnability of the problem. Whereas the latter factor depends on the considered problem, the first two in COIN are translated into conditions on how to shape the private utility functions of the agents such that the world utility is increased when the agents improve their private utility.

Wolpert et al. have derived private utility functions that perform well on the above first two conditions. The effectiveness of this top-down approach and their developed utilities are demonstrated by applying the COIN framework to a number of example problems: network routing [20], increasingly difficult versions of the Al Ferrol Bar problem [17], and Braess' paradox [11]. The COIN approach proved to be very effective for learning these problems in a distributed system. In particular, the systems exhibited excellent scaling properties. Compared to optimal solutions, it is observed that a system like COIN becomes relatively *better* as the problem is scaled up [17].

In recent work [10], the COIN framework has been applied to problems where different "single agent" RL algorithms are traditionally tested: grid-based world exploration games [10, 8]. In this problem-domain, agents move on a grid-like world where their aim is to collect tokens representing localized rewards as efficiently as possible (e.g. [8]). For a Multi-Agent System, the challenge is to find sequences of actions for each individual agent such that their *joint* sequences of actions optimize some predetermined utility of the collective. The main result of [10], in line with earlier work, is that the derived utility functions as advocated in the COIN framework significantly outperform standard solutions for using RL in collectives of agents.

We observe that in [10] the used RL algorithm, Q-learning, is the same as used in previous work on COIN. However, for learning sequences of actions with RL, there are substantially more powerful methods which we adapt for the COIN framework. We further report some modifications to these RL methods to address specific issues that arise in the COIN framework. We find that using these methods, our enhanced COIN approach yields more optimal exploration while converging more quickly.

We start from our replication efforts of the grid-world problem of [10]. We report an anomaly in that a collection of selfish, greedy agents proved to be performing similarly to the more elaborate and computationally intensive COIN approach. To find out whether this issue was isolated to the particular grid-world example chosen, we designed grid-worlds that require more coordination among the agents. We then found that in those cases COIN does provide significant improvements compared to simple, greedy agents.

This document is structured as follows. In Section 2, we describe the COIN framework. In Section 3 we report on our reproduction of [10]. In Section 4 we present problems that require more coordinated joint actions of a MAS. We also introduce an extension for COIN for sequences of actions. In Section 5, we adapt a number of more advanced RL methods for learning sequences of actions to the COIN framework, and report on the performance improvements. In Section 6 we discuss future work and conclude.

2. Background: COllective INtelligence

In this Section, we briefly outline the theory of COIN as developed by Wolpert et al. More elaborate details can be found in [21, 17, 18]. Broadly speaking, COIN defines the conditions that an agent's private utility function has to meet to increase the probability that learning to optimize this function leads to increased performance of the collective of agents. Thus, the challenge is to define a suitable private utility function for the individual agents, given the performance of the collective.

Formally, let ζ be the joint moves of all agents. A function $G(\zeta)$ provides the utility of the collective system, the *world utility*, for a given ζ. The goal is to find a ζ that maximizes $G(\zeta)$. Each individual agent η has a private utility function g_η that relates the reward obtained by the collective to the reward that the individual agent collects. Each agent will act such as to improve its own *payoff*. The challenge of designing the collective system is to find private utility functions such that when individual agents optimize their payoff, this leads to increasing world utility G, while the private function of each agent is at the same time also easily learnable (i.e. has a high *signal-to-noise* ratio, an issue usually not considered in traditional mechanism design).

Following a mathematical description of this issue, Wolpert et al. propose the **Wonderful Life Utility** (WLU) as a private utility function that is both *learnable* and *aligned* with G, and that can also be easily calculated. In a collective system consisting of multiple agents collectively collecting rewards (tokens) on a grid, as discussed in more detail in Section 3, the WLU for one agent η at time t with respect to the collective is ([10]):

$$WLT^{\vec{0}}_{\eta,t}(\zeta) = GR_t(\zeta) - (T(L_{\hat{\eta},<t+1}, \Theta) - T(L_{\hat{\eta},<t}, \Theta)) \tag{2.1}$$

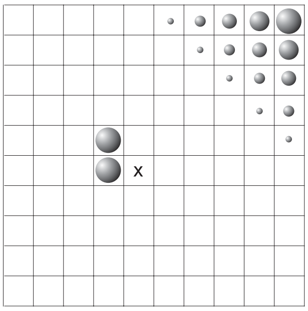
where:

- ζ is the joint moves of all the agents.
- L is the location matrix of the agents over time. And
  - L_η is the location of agent η for all the time steps.
  - $L_{\eta,t}$ is the location of agent η at time step t.
  - $L_{\eta,<t}$ are the locations of agent η at earlier time steps.
  - $L_{\hat{\eta}}$ are the location of agents other than η.
- $T(L, \theta)$ returns the value of the tokens received from the location matrix.[1]
  - θ is the location of the initial tokens.
  - $T(L, \theta) = \sum_{x,y} \theta_{x,y} min(1, L_{x,y})$, i.e. the tokens picked up are the visited tokens, but no more than once.
- $GR_t(\zeta) = T(L_{<t+1}, \theta) - T(L_{<t}, \theta)$, i.e. the value of all the tokens picked up at time step t.

Hence $WLT^{\vec{0}}_{\eta,t}(\zeta)$ for agent η at time step t is equal to the value of all the tokens picked up by all the agents for that step minus the value of the tokens picked up by the other agents $\hat{\eta}$ at time step t. If agent η picks up a token τ at time step t, which is not picked up by the other agents, then η receives a reward of $T(\tau)$. If this token is however picked up by any of the other agents at time step t, then the first term GR_t of Equation 2.1 is unchanged while the second term drops with the value of τ. Agent η then receives a penalty $-T(\tau)$ for competing for a token targetted by one of the other agents $\hat{\eta}$.

Compared to the WLU function, other payoff functions have been considered in the literature for distributed Multi-Agent Systems: the Team Game utility function (TG), where the world-utility is equally divided over all participating agents, or the Selfish Utility (SU), where each agent only considers the reward that it itself collects through its actions. These two common alternatives are extreme examples. The Team Game utility can suffer from poor learnability, as for larger collectives it

[1] [10] uses V instead of T, which we use to not confuse the issue with the V as the valuation function in Q learning.



```
for(int x=0; x < n; x=x+1)
for(int y=0; y < n; y=y+1) {
      tokens[x][y] = 1.2*(x+y-n)/(1.0*n);
      if(tokens[x][y] < 0.4)
         tokens[x][y] = 0; }
 tokens[n/2][(n/2)-1] = 1.0;
 tokens[(n/2)+1][(n/2)-1] = 1.0;
```

(a) The tokens

(b) The algorithm

Figure 1: The original problem of [10]

becomes very difficult for each agent to discern what contribution is made (low signal-to-noise ration), and the SU suffers from – potentially – poor alignment with the world-utility. The superiority of the use of private WLU functions with local reinforcement learning has been shown for a variety of problems [19, 17, 11]. In the terminology of the COIN framework, the WLU is *factored* and *aligned* with the world utility. The Aristocratic Utility (AU) is not treated in this work as [10] observes comparable performance for the AU and WLU while the former is significantly more difficult to implement.

We compute the WLU for an agent η by first letting all agents except η, i.e. $\hat{\eta}$, make their moves and only then moving η. For N moves, the agents $\hat{\eta}$ generate grids $grid_0$ to $grid_{N-1}$ where $grid_t$ documents the tokens picked up by agents $\hat{\eta}$ at time step t and the rewards which can be experienced by η for its moves. The agents $\hat{\eta}$ start from grid $grid_0$ filled with the initial tokens and with all grids $grid_{t>0}$ initially empty. At time t, agents $\hat{\eta}$ pick up tokens from $grid_t$ at their current locations. A penalty (that is: the negative of the value of the token) is then substituted for this token in $grid_t$. The grid $grid_{t+1}$ is then filled with (a copy of) the remaining tokens of $grid_t$ prior to the moves of agents $\hat{\eta}$ at timestep $t+1$. Agent η then starts at the modified grid $grid_0$ after the moves of the agents $\hat{\eta}$ are completed. Note that a token picked up by η at timestep t is removed from $grid_{t>}$; a token can only be picked up once.

3. Learning Joint Sequences of Actions

In this Section we present our findings when reproducing [10], with additional details from [6]. Agents in [10] jointly have to learn to explore a grid and efficiently retrieve the available tokens. The agents in one step move either up, down, right, or left. The order of movement for each of the agents is identical in each turn. The tokens on the grid are as defined in Algorithm 1(b) for a $n \times n$ grid where we consider the case from [10] where $n = 10$. A cluster of tokens increasing in value towards the edge of the grid is formed in a corner of the grid. Two extra tokens of value 1 are then added close to the center of the grid. This gives the grid of Figure 1(a) where **x** marks the initial location of the agents at $[n/2, n/2]$.

Each individual agent uses Q-learning [8] as its RL algorithm. A learner's input space consists of the location of the agent in the grid and the action space consists the four directions in which the agent can move. The policy π of an agent in [10] is stochastic according to a softmax function; in the policy, a random action a_i is chosen for state s and constant c (set at 50) with normalized chance in $[0, 1]$ of $\frac{c^{Q(s,a_i)}}{\sum_j c^{Q(s,a_j)}}$. The discount factor γ is set to 0.95. The learning rate α_t at time step t for a state s is taken as $\alpha_t = \frac{1}{1+0.0002*visits(s)}$, where $visits(s)$ is the number of times the state s has been visited during a learning step ([6]). The decreasing value of α serves to induce agents to initially explore, and

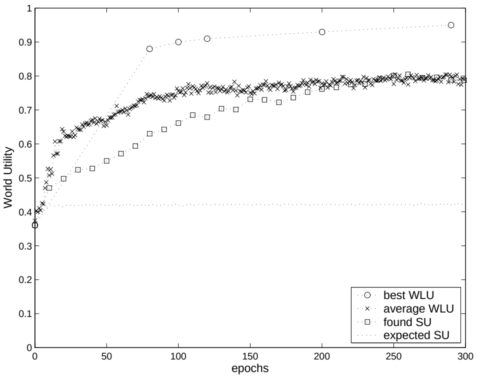


Figure 2: Reproduced results

then gradually fine-tune their behavior to maximize their utility as α drops.

In Figure 2 we present the learning curves for the problem setting [10]. With the dynamic α of [10] we reproduced the low utility for the SU of ≈ 0.4. However, we found that for a fixed α of 0.1 using the SU, the MAS achieved a fitness of ≈ 0.8 in 300 epochs. For a small α, the selfish agents are able to focus gradually on collecting the tokens even though they directly compete. The large concentration of tokens in the corner of the grid acts as an attractor for these agents. In the further experiments we present results for a default value of α of 0.1 due to generally poor results for a dynamic α. [2] In Figure 2, we also show the results which correspond to findings in [10] for the WLU in a typical best case. Poorer solutions were however found when averaging over 100 runs.

Our finding suggests that the problem considered in [10] is not as well suited to present the additional distributed coordination capabilities of the COIN framework as was claimed. We therefore study more general token schemes where coordination of the joint actions is a prerequisite for high performance. In the further experiments we also each step randomize the order of movement for the agents to better simulate a realistic MAS.

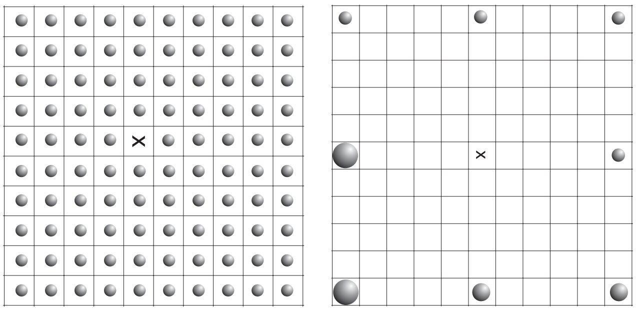
4. Coordinated Grid-World Problems

Given the good performance of the SU in Section 3, we designed a number of token-retrieval problems that in particular play to the weaknesses of selfish agents or have a low signal-to-noise ratio for learning. For both problems, a reasonable utility is achieved for a wide selection of parameters using the selfish utility or the team game, but the challenge is in achieving near to optimal performance. We present two interesting examples in Figures 3(a) and 3(b), where **x** marks the start locations of the agents.

In the problem of Figure 3(a), the whole grid (of size 10×10) is filled with tokens of value 1, except for the initial location of the agents. Each agent is allowed 10 moves. Thus, the agents have to disperse in order to maximize the number of visited nodes on the grid. The world utility is defined as the maximum number of tokens the agents can collectively pick up in their moves. Such a grid is representative of a situation where agents have little prior information of the world and have to devise a strategy for maximum exploration. This problem is difficult to solve due to the low signal-to-noise ratio with uniform reward for all visited locations. We increase the action-space by allowing the agents to move diagonally in order for them to better be able to disperse from their clustered initial position.

For the problem of Figure 3(b), differently valued tokens are placed on the edge of a 11×11 grid and agents take five steps. Diagonal moves are also possible. The agents are hence able to pick up all

[2] This generally poor performance can be caused by the slow drop in the value of α with the number of visits to a state resulting in a high α for initial epochs which will cause a system to react strongly to rewards (α drops to 0.2, 0.1 and to 0.05 for respectively $20,000$, $45,000$ and $95,000$ visits). Thus, a dynamic α allows for easy propagation of reward over regions with little feedback in reinforcement signal, but can also lead to strong fluctuations in behavior not suitable for difficult scenarios where delicate tuning of behavior is required.



(a) Completely filled. (b) Tokens on the edge

Figure 3: The two new grids

tokens if they cooperate perfectly by each focusing on a distinct token. This problem is representative of a complex set of tasks which must all be completed by one of the agents, but the different tasks have varying priorities. A solution is difficult to learn as agents may focus exclusively on the high priority tokens and neglect to collect the cheap tokens needed for high performance.

In the first coordination problem with eight agents, selfish agents using SU achieve a utility of close to 0.8 (Figure 4(a)). With a dynamic (high) α of Section 3, a high fitness is quickly reached but a higher eventual fitness is reached using a fixed α of 0.1. An absolute higher fitness was not found for a wide study of parameters. This problem is hence relatively simple for the SU to solve partially, but the challenge is in achieving (near to) full utility. The Team Game Utility (TG) achieved a comparable performance, but likewise suffers from a low signal-to-noise-ratio and is not able to improve performance beyond the presented level.

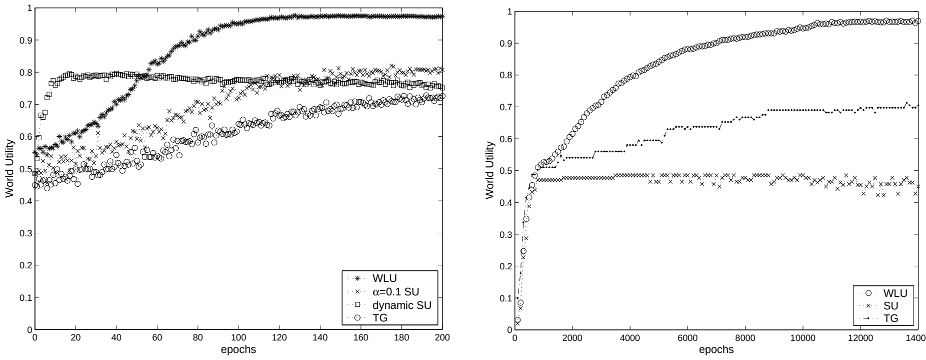
Agents using the WLU showed slightly better performance with a final utility of 0.82. This low result was found to be caused by loops in the paths of the individual agents promoted by the COIN framework. Due to the softmax function for choice of action of Section 3, an agent is averse to actions with negative Q values. An agent η hence quickly learns to avoid penalties imposed by other agents $\hat{\eta}$. Agent η then tends to find a good partial path and revisit it as a 0 immediate reward[3] for an action is superior to receiving a penalty.

To alleviate this problem, we give an agent η a penalty for revisiting a state s which has been visited at an earlier time step in the same epoch. When agent η visits grid $grid_t$ during computation of the WLU as defined in Section 2, then instead of a token τ picked up being removed from grids $grid_{t>}$, a penalty $(-T(\tau))$ is set on these grids $grid_{t>}$.[4] With this approach, an agent learns not to revisit an earlier part of its travels as this also results in a penalty. For the grid of Section 3(a), in Figure 4(a), the improved high performance with use of a penalty for revisiting states is given. In the rest of the paper, the agents using the WLU likewise pay a penalty for revisiting an earlier state as otherwise the performance of the COIN framework was found to be significantly lower.

In the second problem of Figure 3(b) agents using the Selfish Utility function are attracted to the high token values, making the collective perform poorly. As can be seen in Figure 4(b), a maximum fitness of close to 0.5 is temporarily achieved as the agents explore in search of good tokens. With increasing competition for the high value tokens, the positive reinforcement signal for these targets decreases and the agents become indecisive. For the TG, a maximum fitness (even after $50,000$ epochs) of ≈ 0.7 is slowly reached as the agents are unable to effectively target a token due to the low

[3] All tokens on this earlier part of the path have been retrieved.

[4] The penalty for revisiting a state may have to increase as the grid becomes more crowded and the agent needs an incentive to explore beyond an earlier successful route and across the negative penalties deposited by neighboring agents.



(a) A full grid

(b) A sparse grid

Figure 4: Two new problems

signal-to-noise ratio. However, when using the WLU with a penalty for revisiting states, the agents are able to learn to pick up all the tokens (Figure 4(b)). A fitness of 0.5 is quickly reached and after an agent has chosen or won a token, the WLU issues sufficiently consistent penalties to convince the competing agents to look elsewhere on the grid.

Summarizing, in this section we have shown how the extended COIN framework outperforms the SU and TG for two illustrative problems. Agents in the COIN approach through the WLU are able to solve a general distribution problem where they overcome the low signal-to-noise ratio which limits the performance of the SU and TG. The agents using the WLU are also able to solve a difficult collaborative task where high priorities for a selection of task attracts naive learners at the expense of other tasks.

5. Scaling COIN

The RL algorithm used in [10] is plain Q-learning. Effectively, the update of the value-function after a move only considers the immediate reward and a valuation of the next state. It is well known that for agents optimizing grid-like world-exploration problems or learning sequences of actions, more effective RL algorithms have been developed (that take into account the expected future rewards of a sequence of moves). Wolpert et al. based their COIN framework on the presumption of individual agents using "sufficiently powerful" RL algorithms. In this Section, we explore several possibilities for using more powerful RL algorithms.

5.1 Enhanced RL Techniques

Watkins Q(λ): First, we applied Watkins' Q(λ) Learning [12, 8] in a COIN setting. Q(λ) learning has been reported to substantially improve on the results for learning for single agent applications. Through the use of eligibility traces [12, 8], a single agent can more efficiently propagate its experienced reinforcement signals through its Q-values. In the COIN framework, the propagation of a penalty produced by an agent η is expected to also more efficiently be propagated using Q(λ) over the paths of agents that interfere with the activities of η.

Within the WLU framework, we can devise an alternative to Q(λ) in the form of **Temporal Propagation of Penalties.** Temporal Penalty (TP) propagation works as follows: in the WLU, a penalty is incurred by the learner if it picks up a token at the same time step as one of the baseline agents would. Recall that the reward at time step t for agent η in Section 2 is defined as $WLT^{\vec{0}}_{\eta,t}(\zeta)$. The reward (or penalty) for agent η is determined in an interwoven fashion with the moves made by the other agents in the *same* time step. A consequence of this definition is that an agent η is not penalized if it picks up a token at time t which one of the other agents $\hat{\eta}$ is *planning* to pick up at a later time step $t_n > t$. We can however *temporally propagate* a penalty for snatching any token by η from the other agents $\hat{\eta}$: let $S(L, \Theta)$ be the set of tokens picked up during movement. Then for $S_{\hat{\eta}} = S(L_{\hat{\eta}}, \Theta)$, our modified reward for agent η at time step t of $TPWLT^{\vec{0}}_{\eta,t}(\zeta)$ is defined as:

$$GR_{\hat{\eta},t}(\zeta) + T(S(L_{\eta,t}, \Theta) \setminus S_{\hat{\eta}}) - T(S(L_{\eta,t}, \Theta) \cap S_{\hat{\eta}}). \tag{5.1}$$

The above modified utility function induces an agent η to consider all future actions of the other agents $\hat{\eta}$, and not just those that coincide for specific time steps. This can support a stronger cooperation between the agents.

Additionally, convergence of learning for an agent η can possibly be enhanced through multiple epochs of learning relative to the other agents $\hat{\eta}$ not only once, as defined in Section 2, but multiple times within one epoch: **Model Based Planning**. The potential expensive calculation of $WLT^{\vec{0}}_{\eta,t}(\zeta)$ of Section 2 for each agent η is used several times by η to learn how to optimally behave relative to the other agents. Agent η traverses (copies of) the grids $grid_0$ to $grid_t$ not once, but n times during learning in one epoch. In [13], a similar model-based approach is used where agents learn according to a generalized DYNA-Q [8] architecture by interleaving of planning according to a learned world model and acting based on this model.

5.2 Results

As a case study for these more powerful RL algorithms within a COIN framework, we used the joint coordinated exploration problem of Figure 3(b). The tokens on the grid are however placed a distance of one from the edge, making the problem more illustrative by giving more opportunity for (discounted) feedback from the received rewards (as there are more successful paths for an agent to pick up a token from its initial position).

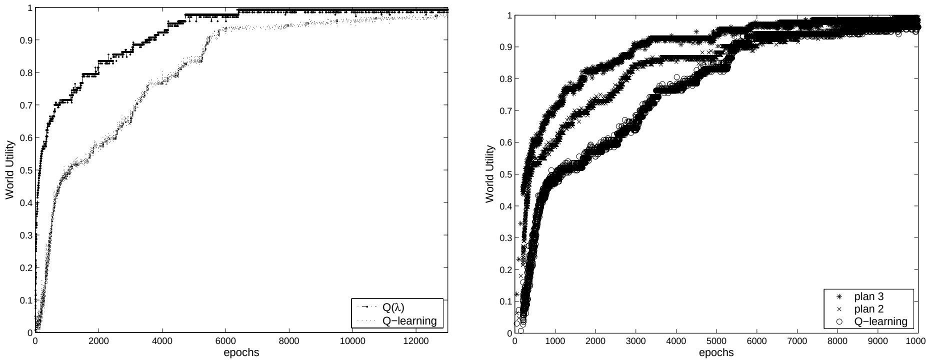
As shown in Figure 5(a), Q(λ) aids the WLU in finding a solution to the problem. Convergence speed is improved significantly and full utility is reached. We found near identical performance gain relative to Q(λ) for **Temporal Propagation of Penalties.** This similar gain is hypothesized to have roots in similar propagation of rewards. For Q(λ), discounted penalties are carried over to bordering states in a traveled path, whereas for the temporal penalty propagation, penalties are carried over from states that will be visited near in the future.

Figure 5(b) shows the improved convergence when using model-based planning. Convergence is speeded up considerably as the number of learning iterations for one agent η relative to the other agents is increased from one (standard) to two to three. For larger problems, for example a larger grid, added iterations did further increase converge properties of the system. Preliminary results for more complex scenarios indicate that differentiating the frequency of model-based learning in accordance with the strength of the absolute experienced reward of η can further significantly speed up convergence (a strong reward is an indication of whether the agent is on the right track or should do something entirely else). By proportioning the learning of η relative to the experienced reward, η can benefit better from learning from the experienced rewards relative to the actions of $\hat{\eta}$.

6. Discussion and Conclusion

In this paper we studied the COllective INtelligence (COIN) framework of Wolpert et. al. We reproduced [10] as a case study where agents use standard Reinforcement Learning (RL) techniques to collectively pick up tokens on a grid. We observed that for more complex problems the COIN framework is able to solve difficult MAS problems where fine-grained coordination between the agents is required, in contrast to multi-agent systems that use less advanced decentralized coordination.

We enhanced the COIN formalism to avoid pathological situations due to the nature of the WL utility-function. In particular we discounted actions looping back to earlier action sequences to promote a unique path traveled by an agent. This enhancement resulted in near optimal fitness for difficult



(a) Q(λ).

(b) Model-based learning.

Figure 5: Extended RL applications

token retrieval actions. Furthermore, we investigated the use of more powerful RL techniques within the (enhanced) COIN framework. We explored use of Watkins Q(λ) learning, model based learning, and the extended use of penalties in COIN over sequences of actions. All three extensions led to improved performance for the problems investigated and demonstrate methods for further improving the performance of the COIN framework in larger, more complex applications.

As future work we consider boot-strapping techniques for single agent RL to the COIN framework. RL in general can significantly benefit from directed exploration ([5, 9] and [16]). Sub-goal detection as in [7] can also greatly speed up the learning of complex tasks. For example, in [7] an agent learns to focus in learning on critical points in the task which form bottlenecks for good overall performance. An open question is how the above work can be integrated in the (extended) COIN Framework for task with bottlenecks occurring due to dynamic interactions in a MAS.

References

1. A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete-Event Systems journal*, 2003. to appear.
2. C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proceedings of the ICML-2002 The Nineteenth International Conference on Machine Learning*, 2002.
3. G. Hardin. The tragedy of the commons. *Science*, 162:1243–1248, 1968.
4. M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proc. 17th International Conf. on Machine Learning*, pages 535–542. Morgan Kaufmann, San Francisco, CA, 2000.
5. T. Mitchell. *Machine Learning*. McGraw-Hill, 1997.
6. Personal communication with A. Agogino.
7. shai Menache, S. Mannor, and N. Shimkin. Q-cut - dynamic discovery of sub-goals in Reinforcement Learning. In T. Elomaa, H. Mannila, and H. Toivonen, editors, *Machine Learning: ECML 2002, 13th European Conference on Machine Learning*, volume 2430 of *LectureNotes in Computer Science*, pages 295–306. Springer, 2002.
8. R. Sutton and A. Barto. *Reinforcement learning: An introduction*. MIT-press, Cambridge, MA, 1998.
9. S. B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1992.
10. K. Tumer, A. Agogino, and D. Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Autonomous Agents & Multiagent Systems*, pages 378–385, part 1. ACM press, 2002.
11. K. Tumer and D. Wolpert. COllective INtelligence and Braess' paradox. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 104–109, Austin, Aug. 2000.
12. Watkins and Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
13. G. Weiss. A multiagent framework for planning, reacting, and learning. Technical Report FKI-233-99, Institut für Informatik, Technische Universität München, 1999.
14. M. P. Wellman. The economic approach to artificial intelligence. *ACM Computing Surveys*, 28(4es):14–15, 1996.
15. M. P. Wellman. Market-oriented programming: Some early lessons. In S. Clearwater, editor, *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, River Edge, New Jersey, 1996.
16. M. Wiering. *Explorations in Efficient Reinforcement Learning*. PhD thesis, University of Ams-

terdam, 1999.

17. D. Wolpert and K. Tumer. An introduction to COllective INtelligence. Technical Report NASA-ARC-IC-99-63, NASA Ames Research Center, 1999. A shorter version of this paper is to appear in: Jeffrey M. Bradshaw, editor, Handbook of Agent Technology, AAAI Press/MIT Press, 1999.
18. D. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 2001. in press.
19. D. H. Wolpert, K. Tumer, , and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems-11*, pages 952–958, Denver, Dec. 1998.
20. D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Advances in Neural Information Processing Systems-11*, pages 952–958, Denver, 1998.
21. D. H. Wolpert, K. R. Wheeler, and K. Tumer. General principles of learning-based multi-agent systems. In O. Etzioni, J. P. Müller, and J. M. Bradshaw, editors, *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99)*, pages 77–83, New York, May 1–5 1999. ACM Press.

Acknowledgement We thank Stefan Blom for letting us use the STW cluster at CWI.