An Architectural Framework for Integrated Multiagent Planning, Reacting, and Learning

Gerhard Weiß

Institut für Informatik, Technische Universität München
D-80290 München, Germany
weissg@in.tum.de

Abstract. Dyna is a single-agent architectural framework that integrates learning, planning, and reacting. Well known instantiations of Dyna are Dyna-AC and Dyna-Q. Here a multiagent extension of Dyna-Q is presented. This extension, called M-Dyna-Q, constitutes a novel coordination framework that bridges the gap between plan-based and reactive coordination in multiagent systems. The paper summarizes the key features of Dyna, describes M-Dyna-Q in detail, provides experimental results, and carefully discusses the benefits and limitations of this framework.

1 Introduction

Dyna (e.g., [31, 32] and [33, Chapter 9]) is an architectural framework that integrates learning, planning, and reacting in single agents. This integration is based on two fundamental observations that can be summarized as follows:

- "Learning is a valuable basis for both planning and reacting." Through learning an agent acquires information that enables him to plan and react more effectively and efficiently. More specifically, according to Dyna an agent plans on the basis of an incrementally learnt world model and reacts on the basis of incrementally learnt values that indicate the usefulness of his potential actions.
- "Both planning and reacting are a valuable basis for learning." An agent uses the outcomes of his planning and reacting activities for improving his world model and the estimates of their actions' usefulness. More specifically, according to Dyna planning serves as a basis for trial-and-error learning from hypothetical experience, while reacting serves as a basis for trial-and-error learning from real experience.

Figure 1 summarizes this interwining of learning, planning, and reacting. This figure is complemented by Figure 2 which overviews the general flow of control and information within a Dyna agent. Both real and hypothetical experience are used for updating the action values. Additionally, real experience (reacting) is employed for learning a world model which helps to handle hypothetical experience (planning). Two well known instantiations of the Dyna framework are Dyna-AC (Dyna plus actor-critic learning) and Dyna-Q (Dyna plus Q-learning); see e.g. [31]. An advantage of Dyna-Q over Dyna-AC is that it is simpler to realize. In particular, Dyna-AC requires two learning rules and two memory structures (evaluation function and policy), while Dyna-Q requires only

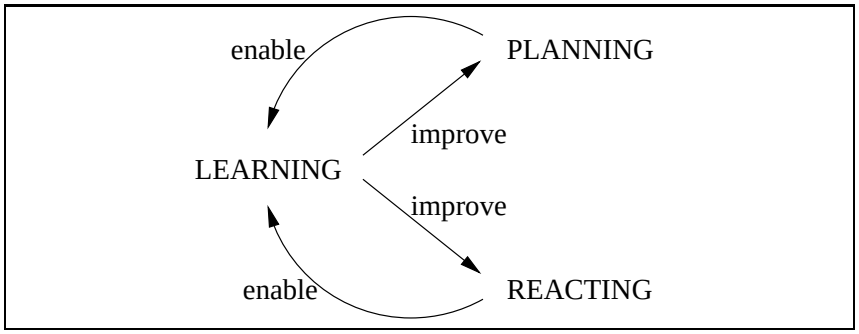

Fig. 1. Dyna's relationships between learning, planning, and reacting.

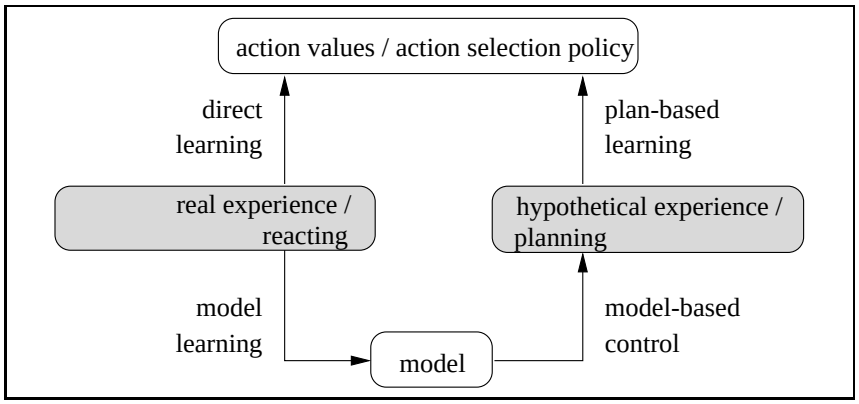

Fig. 2. Dyna's internal flow of control and information.

one learning rule and one memory structure (which is a cross between Dyna-AC's two memory structures).

This paper describes work that extends the Dyna framework to multiagent settings. Based on Dyna-Q a general coordination framework called M-Dyna-Q for multiagent systems is proposed that integrates joint learning, joint planning, and joint reactivity. M-Dyna-Q bridges the gap between two contrary main approaches to multiagent coordination, namely, plan-based coordination (see e.g. [4–6, 9, 12, 15, 23, 28, 16]) and reactive coordination (see e.g. [2, 8, 10, 18, 20, 22, 25, 27] and also [7]). The basic idea behind the former approach is that the agents jointly generate hypothetical activity sequences on the basis of their world model in order to find out in advance (i.e., before acting in the real world) what actions among all possible actions are most promising. Against that, the basic idea behind the latter approach is that the agents jointly generate rapid reactions on the basis of simple stimulus-response rules that can be carried out by the agents. A unique key feature of M-Dyna-Q is that it brings together these two contrary ideas.

The paper is structured as follows. First, Section 2 introduces M-Dyna-Q in detail. Next, Section 3 describes initial experimental results on M-Dyna-Q. Finally, Section 4 discusses M-Dyna-Q and provides pointers to related work.

2 The M-Dyna-Q Framework

According to the M-Dyna-Q framework the overall multiagent activity results from the repeated execution of two major joint activities—action selection and learning—, each running either in real or hypothetical mode. In the most simplest form (which also underlies the experiments reported in the next section), the agents switch between the two modes at a fixed and predefined rate. The real mode corresponds to (fast) "reactive behavior," whereas the hypothetical mode corresponds to (slower) "plan-based behavior." During action selection, the agents jointly decide what action should be carried out next (resulting in the next real or a new hypothetical state); this decision is made on the basis of the agents' distributed value function in the case of operating in the real mode, and on the basis of the agents' joint world model in the case of operating in the hypothetical mode.[1] During learning the agents adjust both their world model and their value function if they act in the real mode, and just their world model if they act in the hypothetical mode. Below these two major activities are described in detail. Figure 3 conceptually overviews the basic working cycle of M-Dyna-Q. As this figure shows, every working cycle runs either in real mode (in this case it is called a real working cycle) or hypothetical mode (in this case it is called a hypothetical working cycle). The Figure 4 illustrates the flow of control and information within this framework.

Throughout the paper the following simple notation is used and the following elementary assumptions are made. $\mathcal{A}g = \{A_1, \ldots, A_n\}$ $(n \in \mathsf{N})$ denotes the finite set of agents available in the MAS under consideration. The environment in which the agents act can be described as a discrete state space, and the individual real and hypothetical states are denoted by $\mathcal{S}, \mathcal{T}, \mathcal{U}, \ldots$ $\mathcal{A}c_i^{poss} = \{a_i^1, \ldots, a_i^{m_i}\}$ $(m_i \in \mathsf{N})$ denotes the set of all possible actions of the agent A_i, and is called his *action potential*. Finally, $\mathcal{A}c_i^{poss}[\mathcal{S}]$ denotes the set of all actions that A_i could carry out (identifies as "executable") in the environmental state $\mathcal{S}$ $(\mathcal{A}c_i^{poss}[\mathcal{S}] \subseteq \mathcal{A}c_i^{poss})$.

Joint Action Selection. According to M-Dyna-Q each agent A_i maintains state-specific estimates of the usefulness of his actions for goal attainment. More specifically, an agent A_i maintains, for every state $\mathcal{S}$ and each of his actions a_i^j, a quantity $Q_i^j(\mathcal{S})$ that expresses his estimate of a_i^j's state-specific usefulness with respect to goal attainment. Based on these estimates, action selection works as follows. If the agents operate in the "real mode", then they analyze the current real state $\mathcal{S}$, and each agent A_i identifies and announces the set $\mathcal{A}c_i^{poss}[\mathcal{S}]$ of actions it could carry out immediately (assuming the availability of a standard blackboard communication structure and a time-out announcement mechanism). The action to be carried out is then selected among all announced

[1] In accordance with common usage in the field of reinforcement learning, here "model" refers to a mapping from state-action pairs to state-reward pairs, and "value function" refers to a mapping from state-action pairs to values expressing the estimated usefulness of carrying out actions in given states.

1. **Joint Action Selection:**
   - the agents choose among $mode = real$ and $mode = hypo(thetical)$
   - if $mode = real$: $state =$ current real state
   - if $mode = hypo$: $state =$ any previously observed real state
   - each agent announces the actions it could carry out in $state$
   - if $mode = real$: an action is selected from all actions announced in $state$ with probability proportional to their estimated usefulness, the selected action is carried out (resulting in a new real state)
   - if $mode = hypo$: an action is randomly choosen from all actions previously carried out in $state$ (resulting in a hypothetical successor state)
2. **Joint Learning:**
   - if $mode = real$: the agent that carried out the selected action adjusts this action's estimated usefulness dependent on the usefulness of the actions applicable in the new real state, and the agents update their world model
   - if $mode = hypo$: the agent that could carry out the selected hypothetical action adjusts this action's estimated usefulness dependent on the usefulness of the actions applicable in the hypothetical successor state

Goto 1

Fig. 3. The basic working cycle of M-Dyna-Q.

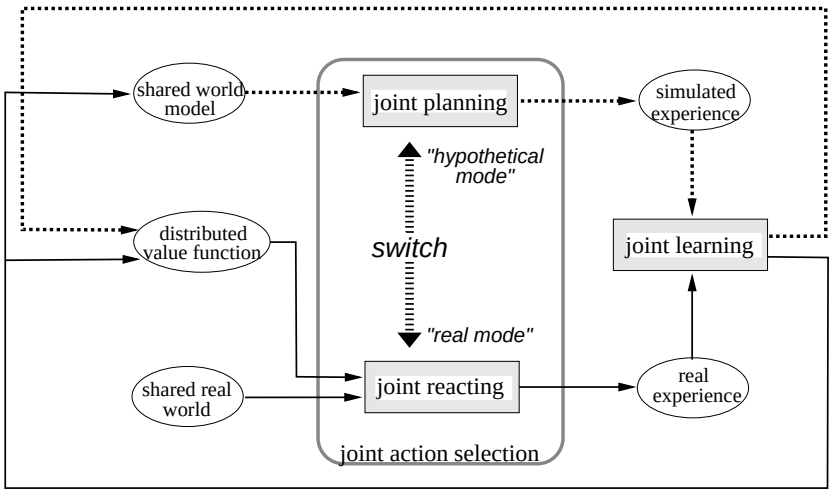

Fig. 4. M-Dyna-Q's flow of control and information in the real (solid lines) and hypothetical (dotted lines) mode.

actions dependent on the agents' action selection policy. A standard policy (which was also used in the experiments reported below) is that the probability of selecting an an-

nounced action a_i^j is proportional to the estimated usefulness of all actions announced in $\mathcal{S}$, i.e.,

$$\frac{e^{Q_i^j(\mathcal{S})}}{e^{\sum_{a_i^j} Q_i^j(\mathcal{S})}} \tag{1}$$

where the sum ranges over all currently announced actions (i.e., over $\bigcup_{i=1}^{n} \mathcal{A}c_i^{poss}[\mathcal{S}]$). If the agents operate in the "hypothetical mode," then they *(i)* randomly choose an environmental state $\mathcal{S}$ from those real states which they already encountered in the past and *(ii)* select an action a_i^j from those already carried out in this state according to Equation (1). This means that in the hypothetical mode the agents simulate real activity and do as if $\mathcal{S}$ is the current real state and a_i^j had been selected for execution. Because the agents only choose hypothetical states that they already know from experience, they avoid to be forced to make speculative activity decisions under unknown hypothetical environmental circumstances. Note that the agents do single-step planning when operating in the hypothetical mode. This "planning in very small steps" has been adopted from the single-agent Dyna-Q framework with the intention to enable the agents to redirect their course of activity without unnecessarily wasted computation and communication whenever necessary.

Joint Learning. Learning is realized by the agents through adjusting the estimates of their actions' usefulness. Suppose that a_i^j has been selected in the real or hypothetical state $\mathcal{S}$ and $\mathcal{T}$ is the resulting successor state. All agents that could carry out actions in $\mathcal{T}$ inform the agent A_i about these actions' estimated usefulness. A_i determines the maximum

$$maxQ_k^l(\mathcal{T}) \quad =_{\text{def}} \quad \max\{Q_k^l(\mathcal{T}) : a_k^l \text{ is executable in } \mathcal{T}\} \tag{2}$$

of these estimates and adjusts his estimate $Q_i^j(\mathcal{S})$ according to

$$Q_i^j(\mathcal{S}) \quad = \quad Q_i^j(\mathcal{S}) + \alpha \cdot [R + \beta \cdot maxQ_k^l(\mathcal{T}) - Q_i^j(\mathcal{S})] \tag{3}$$

where R is the external reward (if any) and α and β are small constants called learning rates. ($maxQ_k^l(\mathcal{T})$ defines, so to say, the global value of the state $\mathcal{T}$.) This adjustment rule can be viewed as a straightforward multiagent realization of standard single-agent Q-learning [34, 35] in which the individual Q-values and thus the value function is distributed over and maintained by several agents. Whereas the adjustment rule is applied in both the real and the hypothetical mode, the world model is updated by the agents only if they act in the real mode; this is reasonable because the most reliable way to improve a world model obviously is to observe the effects of real actions.

3 Experimental Results

We made initial experiments with several synthetic state-action spaces that allow to efficiently obtain indicative results. This paper describes the results for the state-action spaces shown in the Tables 1 (SAS1) and 2 (SAS2). SAS1 consists of 18 world states $(0, \ldots, 17)$ and 32 actions that can be carried out by 5 agents. Most of the actions can

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_1^1	0	1
a_1^1	6	11
a_1^1	7	10
a_1^2	4	8
a_1^3	8	9
a_1^3	10	14
a_1^3	11	17
a_1^3	12	16
a_2^1	0	3
a_2^1	3	7
a_2^2	5	9
a_2^2	14	9

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_2^3	1	7
a_2^3	15	10
a_2^4	6	9
a_2^4	10	16
a_2^4	4	7
a_2^4	12	17
a_3^1	6	12
a_4^1	2	5
a_4^1	7	8
a_4^1	0	4
a_4^2	2	6
a_4^2	9	15

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_5^1	3	5
a_5^1	5	10
a_5^2	3	8
a_5^3	5	8
a_5^3	0	2
a_5^3	2	7
a_5^3	8	10
a_5^4	12	13
a_5^4	4	6
a_5^5	1	5
a_5^5	9	14
a_5^5	11	15

start state: 0 goal state: 17 reward: 1000

Table 1. State-action space SAS1.

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_1^1	0	1
a_1^1	1	10
a_1^1	6	15
a_1^2	11	20
a_1^2	2	6
a_1^2	7	11
a_1^3	12	16
a_1^3	0	21
a_1^4	17	1
a_1^4	3	7
a_1^4	8	12
a_1^4	13	17
a_1^4	18	2
a_1^4	0	21
a_1^5	4	8
a_1^5	9	13

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_2^1	14	18
a_2^1	0	2
a_2^2	19	3
a_2^2	5	9
a_2^2	10	14
a_2^2	15	19
a_3^1	1	6
a_3^1	0	21
a_3^1	2	7
a_3^2	6	11
a_3^2	7	12
a_3^2	11	16
a_3^3	12	17
a_3^3	21	6
a_3^3	3	8
a_3^3	0	3

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_3^3	9	14
a_4^1	0	4
a_5^1	4	9
a_5^1	5	10
a_5^1	8	13
a_5^1	10	15
a_5^1	13	18
a_5^1	0	21
a_5^1	14	19
a_6^1	15	20
a_6^1	1	7
a_6^2	6	12
a_6^3	11	17
a_6^3	21	9
a_6^3	2	8
a_6^3	7	13

a_i^j	$\mathcal{S}$	$\mathcal{T}$
a_6^3	12	18
a_6^4	17	3
a_7^1	3	9
a_7^1	8	14
a_7^2	13	19
a_7^2	0	5
a_7^2	18	4
a_7^2	14	20
a_7^2	4	10
a_7^2	9	15
a_7^3	0	21
a_7^3	19	5
a_7^4	5	6
a_7^4	21	5
a_7^4	10	11
a_7^4	15	16

start state: 0 goal state 1: 16 goal state 2: 20
reward in state 16: 500 reward in state 20: 1000

Table 2. State-action space SAS2.

be carried out in different states (e.g., action a_1^1 in the states 0, 6, and 7), and different actions can be carried out in the same state (e.g., the actions a_2^2, a_5^1, and a_5^3 can be carried out in state 5). The learning task is to find a sequence of at most 5 actions that transform the start state 0 into the goal state 17. (An example of a solution sequence of length $L = 4$ is $< a_5^3, a_4^2, a_3^1, a_2^4 >$.) Reward is provided only if the goal state is reached. SAS2 is designed analogously, except that there are two goal states with different reward levels.

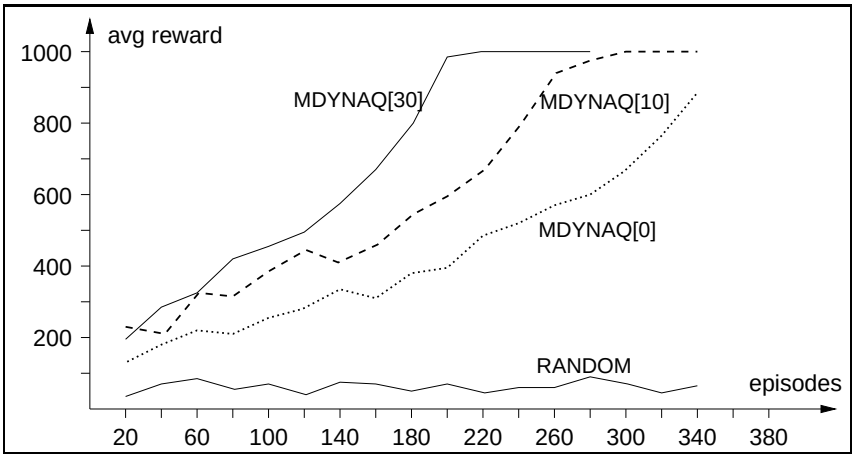

Fig. 5. Experimental results for SAS1.

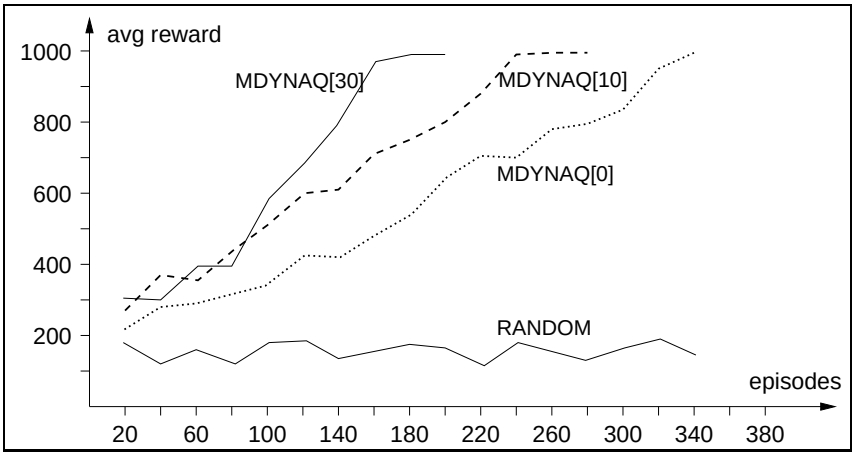

Fig. 6. Experimental results for SAS2.

Figures 5 and 6 show the results for SAS1 and SAS2, respectively. Learning proceeds by the repeated execution of episodes, where an episode is defined as any sequence of at most 5 real working cycles that transform the start state into the goal state (successful sequence) or into a non-goal state (unsuccessful sequence). The parameter setting was as follows: $\alpha = 0.2$, $\beta = 0.9$, and $R = 1000$ (SAS1 in state 17), $R = 500/1000$ (SAS2 in states 16/20). The initial Q-values were all zero. Each figure shows four curves: RANDOM (random walk of maximum length 5 through the state-action space); MDYNAQ[30] (30 hypothetical working cycles after each real working cycle), MDYNAQ[10] (10 hypothetical experiences after each real experience), and MYDNA[0] (no hypothetical experiences at all). Each data point shows the mean reward obtained in the previous 20 episodes, averaged over 5 independent runs. The main

observations are as follows. First, MDYNAQ clearly performed better than uninformed RANDOM search which achieved an average reward level of 64 in the case of SAS1 and 156 in the case of SAS2. This indicates the general performance capacity of M-Dyna-Q. Second, MDYNA always reached the maximum reward level, even if there are different reward levels as it is the case with SAS2. This indicates the robustness of M-Dyna-Q against local performance maxima. Third, MDYNAQ[30] performed better than MDYNAQ[10] which performed better than MDYNAQ[0]. On the average, MDYNAQ[30] (MDYNAQ[10], MDYNAQ[0]) reached the maximum reward level after about 190 (300, 400) episodes in the case of SAS1 and after about 160 (240, 340) cycles in the case of SAS2. This indicates how learning based on hypothetical experiences contributes to M-Dyna-Q's overall performance.

4 Discussion

This paper described a multiagent extension of a single-agent architecture known as Dyna-Q. This extension, M-Dyna-Q, constitutes a coordination framework that combines the ideas of two contrary main approaches to coordination in multiagent systems, namely, plan-based and reactive coordination. Each of these two approaches has its specific advantages and disadvantages:

- An advantage of plan-based coordination is that the probability of carrying out unsuccessful activity sequences, which additionally may be expensive and perhaps even irreversible, is kept low. A disadvantage of this approach is that it is limited by the accuracy of the world model used by the agents. Another disadvantage is that it tends to be rather time-consuming and that the computation and communication costs for coordinating planning activities of multiple agents can grow enormously with the length of the planning sequences. Both disadvantages are directly correlated with the dynamics of the world in which the agents act and with the number of agents involved in the planning process—the more dynamic the world is and/or the more agents are involved, the more challenging it is to cope with these disadvantages.
- An advantage of reactive coordination is that it enables agents to rapidly respond to environmental changes without requiring to equip the agents a priori with complex and often difficult-to-obtain knowledge about their environment. A disadvantage of this approach is that concerted interaction and overall coherence can hardly be achieved through simply applying stimulus-response rules, especially in environments in which there are inherent dependencies among the agents' potential actions. Another disadvantage is that it can lead rather poor performance in environments in which it is hard (costly, time-consuming, and so forth) to undo the effects of actions.

Obviously, M-Dyna-Q aims at merging plan-based and reactive coordination such that there advantages are retained while their disadvantages are avoided.

M-Dyna-Q explicitly integrates joint planning, joint reacting, and joint learning. It is this integration that makes the M-Dyna-Q unique and different from a number of related

approaches to multiagent activity coordination, including approaches that rely on either pure planning or pure reaction (see the references provided in Section 1), approaches that rely on a combination of planning and learning (e.g., [29, 30, 38]), and approaches that rely on a combination of reacting and learning (e.g., [3, 17, 19, 21, 26, 24, 36, 37]). M-Dyna-Q can be considered as a generalization of these approaches, and as such it aims at offering "maximum coordination flexibility." This is not to say that M-Dyna-Q is the best choice for every application. However, due to its potential flexibility this framework seems to be a very promising candidate especially in environments whose dynamics and coordination requirements are not know in advance.

M-Dyna-Q, in its current form, is limited as follows. First, and most important, M-Dyna-Q requires the agents to strictly synchronize their action selection and learning activities. In particular, it requires the agents to sequentialize their actions such that only one action per working cycle is executed. Obviously, this restricts multiagent systems in the parallel capabilities they might have in a given application domain, and further research is necessary to identify and analyze methods for weakening this limitation. A possible solution may be to take *sets* of compatible actions rather than individual actions as the agents' basic activity units, as done in our previous work (e.g. [37]). Second, M-Dyna-Q realizes just a very simple form of planning consisting of one-step lookahead activities of the individual agents. Although this makes sense in a variety of situations (especially in unknown and highly complex environments), in general it is desirable and necessary that the agents possess more advanced planning capabilities. Improvement is necessary both w.r.t. a more flexible handling of the planning depth and a more efficient exploitation of the planning results. Several sophisticated distributed planning mechanisms have been described in the literature (e.g., see [1]), and to explore the use of these mechanisms within M-Dyna-Q is another interesting issue for future research. Third, M-Dyna-Q assumes that the agents can maintain and use a joint world model without remarkable efforts. This is not necessarily the case in application domains in which the agents are not aware of all the effects of their actions or in which they sense different parts of their environment. This limitation requires an extension of M-Dyna-Q toward distributed modeling and diagnosis, as investigated in e.g. [11, 13, 14]. And fourth, switching between real mode (reactivity) and hypothetical mode (planning) is done in a very simple way, and in view of more complex environments there is a need for a more sophisticated switch control. For instance, switching may occur in dependence on the overall performance and may be itself subject to learning. "Optimal switching" constitutes a research theme that is not only of relevance to M-Dyna-Q, but to any multiagent as well as single-agent approach that aims at bringing together reactivity and planning.

To summarize, M-Dyna-Q offers a novel perspective of multiagent coordination based on a unified view of concerted learning, planning, and reacting. What makes M-Dyna-Q additionally interesting is that it has been directly derived from a single-agent architectural framework. We think that these features and the encouraging intitial experimental results clearly justify to say that it is worth to further explore M-Dyna-Q along the research directions outlined above.

Acknowledgments. The research reported in this paper has been supported by Deutsche Forschungsgemeinschaft DFG (German National Science Foundation) under contract We1718/6-3.

References

1. AI Magazine. Special Issue on Distributed Continual Planning, Vol. 20, No. 4, Winter 1999.
2. T. Bouron, J. Ferber, and F. Samuel. A multi-agent testbed for heterogeneous agents. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I. 2*, pages 195–214. North-Holland, Amsterdam et al., 1991.
3. R.H. Crites and A.G. Barto. Elevator group control using multiple reinforcement learning agents. *Machine Learning*, 33(2/3):235–262, 1998.
4. K.S. Decker and V.R. Lesser. Generalized partial global planning. *International Journal of Intelligent Cooperative Information Systems*, 1(2):319–346, 1992.
5. E.H. Durfee, P.G. Kenny, and K.C. Kluge. Integrated premission planning and execution for unmanned ground vehicles. In *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 348–354, 1997.
6. E.H. Durfee and V.R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-21(5):1167–1183, 1991.
7. J. Ferber. Reactive distributed artificial intelligence: Principles and applications. In G.M.P. O'Hare and N.R. Jennings, editors, *Foundations of Distributed Artificial Intelligence*, pages 287–314. Wiley, New York et al., 1996.
8. J. Ferber and E. Jacopin. The framework of ECO-problem sloving. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I. 2*, pages 181–194. North-Holland, Amsterdam et al., 1991.
9. M. Georgeff. Communication and interaction in multi-agent planning. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 125–129, 1983.
10. D. Goldberg and M.J. Matarić. Coordinating mobile robot group behavior using a model of interaction dynamics. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 100–107, 1999.
11. B. Horling, V. Lesser, R. Vincent, A. Bazzan, and P. Xuan. Diagnosis as an integral part of multi-agent adaptability. Technical Report 99-03, Computer Science Department, University of Massachussetts at Amherst, 1999.
12. M.J. Huber and E.H. Durfee. An initial assessment of plan-recognition-based coordination for multi-agent systems. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 126–133, 1996.
13. E. Hudlická and V.R. Lesser. Modeling and diagnosing problem-solving system behavior. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-17(3):407–419, 1987.
14. E. Hudlická, V.R. Lesser, A. Rewari, and P. Xuan. Design of a distributed diagnosis system. Technical Report 86-63, Computer Science Department, University of Massachussetts at Amherst, 1986.
15. F. Kabanza. Synchronizing multiagent plans using temporal logic. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 217–224, 1995.
16. F. von Martial. *Coordinating plans of autonomous agents*. Lecture Notes in Artificial in Artificial Intelligence, Vol. 610. Springer-Verlag, Berlin et al., 1992.
17. M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

18. M.J. Matarić. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, 1995.
19. N. Ono and Y. Fukuta. Learning coordinated behavior in a continuous environment. In G. Weiß, editor, *Distributed Artificial Intelligence Meets Machine Learning*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1221, pages 73–81. Springer-Verlag, Berlin et al., 1997.
20. L.E. Parker. On the design of behavior-based multi-robot teams. *Advanced Robotics*, 10(6):547–578, 1996.
21. L.E. Parker. L-alliance: Task-oriented multi-robot learning insystems. *Advanced Robotics*, 11(4):305–322, 1997.
22. C. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.
23. A.E.F. Seghrouchni and S. Haddad. A recursive model for distributed planning. In *Proceedings of the 2nd International Conference on Multi-Agent Systems (ICMAS-96)*, pages 307–314, 1996.
24. S. Sen and M. Sekaran. Multiagent coordination with learning classifier systems. In G. Weiß and S. Sen, editors, *Adaption and Learning in Multiagent Systems*, Lecture Notes in Artificial in Artificial Intelligence, Vol. 1042, pages 218–233. Springer-Verlag, Berlin et al., 1996.
25. L. Steels. Cooperation between distributed agents through self-organization. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I.*, pages 175–196. North-Holland, Amsterdam et al., 1990.
26. P. Stone and M. Veloso. Collaborative and adversarial learning: A case study in robotic soccer. In S. Sen, editor, *Adaptation, Coevolution and Learning in Multiagent Systems. Papers from the 1996 AAAI Symposium*, Technical Report SS-96-01, pages 88–92. AAAI Press, Menlo Park, CA, 1996.
27. T. Sueyoshi and M. Tokoro. Dynamic modeling of agents for coordination. In Y. Demazeau and J.-P. Müller, editors, *Decentralized A.I. 2*, pages 161–176. North-Holland, Amsterdam et al., 1991.
28. T. Sugawara. Reusing past plans in distributed planning. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*, pages 360–367, 1995.
29. T. Sugawara and V. Lesser. On-line learning of coordination plans. In *Working Papers of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.
30. T. Sugawara and V. Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 33(2/3):129–153, 1998.
31. R.S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2:160–163, 1991.
32. R.S. Sutton. Planning by incremental dynamic programming. In *Proceedings of the Eigth International Workshop on Machine Learning*, pages 353–357, 1991.
33. R.S. Sutton and A.G. Barto. *Reinforcement Learning. An Introduction*. MIT Press/A Bradford Book, Cambridge, MA, 1998.
34. C.J.C.H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge University, 1989.
35. C.J.C.H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
36. G. Weiß. Action selection and learning in multi-agent environments. In *From Animals to Animats 2 – Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 502–510, 1993.
37. G. Weiß. Learning to coordinate actions in multi-agent systems. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 311–316, 1993.
38. G. Weiß. Achieving coordination through combining joint planning and joint learning. Technical Report FKI-232-99, Institut für Informatik, Technische Universität München, 1999.