



DAI-Labor
TU Berlin

Andreas Lommatzsch, Sahin Albayrak

Using Multi-Agent Systems for Learning optimal Policies for complex Problems

Zuerst erschienen in:

Proceedings of the
ACMSE 2007 March 23-24, 2007,
Winston-Salem, North Carolina, USA

DAI-Labor, Technische Universität Berlin
Fakultät IV für Elektrotechnik und Informatik
www.dai-labor.de

Using Multi-Agent Systems for Learning optimal Policies for complex Problems

Andreas Lommatzsch
andreas.lommatzsch@dai-labor.de
DAI-Labor, TU-Berlin, Germany

Sahin Albayrak
albayrak@dai-lab.de
DAI-Labor, TU-Berlin, Germany

March 21, 2007

Abstract

The automatic computation of an optimal solution for a complex problem is a challenging task if no additional knowledge is available. For bounded sized problems there are universally applicable algorithms (e.g. genetic algorithms, branch and bound, reinforcement learning). The disadvantage of these algorithms is their high computational complexity so that real world problems can only be solved efficiently, if the search space is reduced dramatically.

In this paper we present an approach that enables the automatic computation of the parameter dependencies of a complex problem without any additional information. The basic idea is to apply reinforcement learning and to incrementally acquire knowledge about the implicit parameters dependencies. Based on the obtained data an optimal strategy is learned. For speeding up the learning process a multi-agent architecture is applied, that supports the simultaneous analysis of alternative strategies. We prove the advantages of our approach by successfully learning a control strategy for a model helicopter.

1 Introduction

The most demanding task in computing optimal solutions for complex problems is to find adequate problem decompositions and to detect the dependencies between the derived subtasks. For solving subproblems of limited complexity, several different approved methods exist, like genetic algorithms or reinforcement learning. In this paper we show how to find an adequate decomposition for a complex problem imitating human exploration strategies. Having decomposed

the complex task, each subtask is delegated to an agent community. The agents analyze alternative tactics based on Reinforcement Learning and incrementally collect knowledge about the respective tasks.

Our approach is based on learning algorithms that provide useful results even when the training scenarios have uncommon parameter values and adequate training scenarios are not available (“no additional knowledge”). A learning algorithm that meets this requirement is Q -Learning[10] because this algorithm provides suitable learning results only by observing the input and output parameter values. Moreover Q -Learning works incrementally and improves the discovered solutions over time. The main disadvantage of Q -Learning is the high demand of computational power but distributing the computation within a network helps to cope with this problem.

For computing solutions for subtasks we use the multi-agent platform JIAC [8]. Multi-agent systems (MAS) have been developed as a promising area of research providing effective techniques for managing autonomous intelligent agents running distributed in a heterogeneous network. MAS enable to solve many complex real world problems by providing methods for coordination, communication and cooperation between intelligent autonomous agents as well as strategies for load balancing [7] and failure management. The use of JIAC simplifies the coordination between concurrently analyzed strategies and accelerates the learning process by parallelizing complex computations.

Bringing together Q -Learning and multi-agent systems we combine the advantages of a robust learning algorithm and the load balancing and coordination strategies provided by MAS. We decompose the complex problem into subtasks. Each task is delegated to a group of agents applying different learning methods. An intelligent coordination strategy between the agents ensures that at each time the most successful strategy for a subtask is selected. The best strategy is provided to the agents that learn solutions for the remaining subtasks. This guarantees that the agents (using the learned strategies for the subtasks) are able to solve the complex task together.

The main objective of this paper is to present an agent based approach for computing an optimal solution for a complex problem. The proposed strategy is demonstrated by the calculation of a control strategy for an autonomously flying helicopter. The paper is organized as follows: First, we formally describe the analyzed problem and explain the applied evaluation methodology. Section 3 describes the approach in detail. In section 4, we present the evaluation results of our method for learning an optimal steering policy for the helicopter MARVIN. Section 5 discusses the features of our method and explains the coordination between the agents. Finally, we give a conclusion and an outlook on future work.

2 Problem decomposition and incremental learning

To decompose a complex problem into a set of subtasks we assume that the goal state is available in a conjunctive normal form. We split the formula at the AND operators into a set of separate propositions. The idea behind this is that learning strategies for the separate conditions is much easier than solving the complex task. Each subtask is delegated to an agent that performs Q -Learning to obtain a policy for the assigned task. The use of coordination and cooperation strategies ensures that each agent considers the policies of the other agents.

Having decomposed the complex goal into a set of less complex tasks, a reasonable sequence for analyzing the subtasks must be determined. In general, there are two possible approaches for defining a learning sequence in multi-agent systems:

1. All agents compute their strategies *simultaneously*. That means, that an agent does not know the policies of the other agents. The advantage of this approach is that a change in the policy of one agent does not influence the policies of other agents. The disadvantage of this approach is that each agent must consider almost all parameters to ensure that the learned policy works successfully with any possible policy. The complexity of each subtask is often so high, that the subtasks cannot be solved within a reasonable period of time.
2. The use of an *incremental* approach enables to learn control policies that are optimized on the already known strategies. Thus an implicit coordination between the agents is established, because an agent takes into account the policies of other agents in the learning procedure. Thereby the number of parameters that have to be explicitly considered by each agent can be reduced significantly.

A disadvantage of an incremental learning approach is that it requires additional effort for computing a reasonable sequence. Moreover the change of an early learned policy often requires to re-learn all policies that where build up the adapted policy.

Weighing the advantages and disadvantages of these two approaches, we decided to apply an incremental method because of the within reached search space reduction. In many situations incremental learning is an intuitive approach that imitates human learning strategies. The analysis of sophisticated problems often makes sense only, if solutions for elementary subtasks are already known. The disadvantage of an incremental approach is that additional effort is required to learn a coordination strategy. However, this effort pays off because an effective search space reduction is the key for the successful use of reinforcement learning.

A key issue for applying incremental learning is a strategy for **measuring the performance** of an agent. Two alternative approaches were tested:

1. **Objective Functions:** We define an objective function that determines the quality of an agent's state by calculating the distance between the actual state and the goal state.
2. **Reinforcement Learning:** The Q -Learning-Algorithm computes for each state s the reward $V(s) = \max_{a \in A} Q(s, a)$ that can be expected when in each state the best action is chosen ($Q(s, a)$ describes the expected cumulative reward for performing action a in the state s). The distribution of V -values shows for how many states a successful strategy has been learned. The advantage of this approach is that if Q -Learning is applied for determining the optimal strategy, no additional effort is needed for computing the quality of a strategy. The disadvantage of this method is that the obtained results can not be easily compared with alternative learning strategies.

3 Approach steps

The following chapter describes the steps of our approach in detail. An evaluation of each step based on the analysed scenario (learning a strategy of an model helicopter) is explained in section 4.

3.1 Finding the easiest subtask

Due to the decision to apply an incremental learning approach the agents have to compute a sequence for examining the derived subtasks. To determine the problem to start with, each subtask is delegated to a group of agents. Assuming that there is one control parameter for each subtask primarily influencing the learning success, each agent of a group tries to learn a strategy for controlling one of the steering parameters. The performance of the learned strategies in each agent group is determined and the best strategy of each group is compared which the quality obtained by other agent groups. The subtask for which the most successful strategy has been computed is selected as the task to begin with.

3.2 Learning policies for the subtasks

In order to find solutions for the subtasks, we analyzed two different methods (objective functions and reinforcement Learning). In this paper we focus on learning based on reinforcement learning: Reinforcement learning allows for learning optimal control strategies by observing the behavior of a process treated as black box. In training sessions the agent explores different paths through the state space and determines for each state the best command (maximizing the received rewards, while minimizing the punishment). The performance of a

learned policy can be judged either by analysing the V -Function or by testing how fast an agent returns from a randomly selected state to a goal state.

3.3 Finding the relevant parameters for a subtask

In order to use **Q -Learning** it is required to define a state space which represents the environment model. The state space is set up from the parameters considered in the learning procedure. Based on the idea of using Q -Learning for measuring the learning success, we analyze how the learning performance depends on the state space used for Q -Learning. If an agent is able to learn a solution based on a state space formed by only a small set of parameters that works as good as a control strategy based on a huge state space, the additional parameters are irrelevant for the problem. Thus an agent should only consider the minimal state space that allows for obtaining good results.

There are two greedy approaches to determine the subset of relevant parameters:

1. We start with the set of all parameters and incrementally remove the parameter that does not reduce the performance of the obtained strategies.
2. We start with an empty parameter set and extend this set by the parameter that leads to the best learning improvement. The search space is expanded until the quality of the control strategy cannot be improved further.

Due to the fact that the required time for Q -Learning grows exponentially with the number of considered parameters, we decided to incrementally expand an empty parameter set.

3.4 Finding interdependencies and learning optimized policies

Having in mind the strategies for finding the relevant parameters, we define a procedure for solving the subtasks. The approach steps are as follows:

1. First of all, the agents must figure out the subtask to start with. To find the “simplest” subtask, each of the subtasks is delegated to a group of at least one agent. Each group of agents tries to compute solutions for the assigned task; each agent of the group learns a strategy for one control parameter. Thus, that in the case of n subtasks (agent communities) and m control parameters $n \times m$ strategies have to be analyzed. The obtained strategies are compared based on their performance. The most successful policy shows the “simplest” subtask and the parameter for that the steering policy must be learned. The testing of $n \times m$ different strategies requires some computational power. Fortunately the agents learn in parallel, providing a remarkable speedup.

2. The subtask, for which an agent learned the most successful strategy, is analyzed more precisely to compute all the relevant parameters for this task (see section 3.3).
3. In the learning procedure we have to distinguish between the parameters defining the considered state space and the parameters that are classified as irrelevant parameters: The values of the relevant parameters are chosen so that all state-action-combinations are analyzed sufficiently often. The remaining parameters are initialized randomly¹.
4. The performance of the learned $n \times m$ (where n is the number of subtasks and m is the number of control parameters) policies is measured and summarized in a matrix. From the matrix we anticipate for which parameter a strategy has to be learned. Knowing the subtask and the parameter for which a policy should be learned, we then compute the parameters that have to be considered for an optimal policy.
5. In order to figure out the relevant parameters for an *optimal* policy, we incrementally determine the parameters that must be considered. This is done based on *Q*-Learning (section 3.3).
6. Having successfully learned a strategy for the first subtask, we start the next learning cycle by computing the subtask that should be analyzed next. In contrast to the previous cycle one more control parameter is steered based on a learned strategy (“incremental learning”).
7. This procedure is continued until for each subtask the relevant parameters are computed.

3.5 Optimizing control policies

Having learned a basic steering strategy for all subtasks and knowing the relevant parameters, we optimize the learned policies. Therefore we globally analyze the parameter dependencies to derive a learning sequence optimized for the approach of incremental learning. Further improvements can often be achieved by using a fine grained state space for *Q*-Learning and by interpolating state parameter values (e.g. based on neural networks).

4 Case study and Evaluation

In this section we describe how our approach can be used for solving a concrete problem. We present the result obtained by learning a control strategy for a model helicopter. Helicopters represent a challenging control problem with

¹Because almost all parameters are defined randomly, only a limited learning success can be expected. If the goal state is reached even under these uncertain conditions, the analyzed parameter set contains the relevant parameters.

high-dimensional complex dynamics, and are widely regarded as difficult to control [3][5]. The helicopter “MARVIN” we use was originally developed for the participation in the International Aerial Robotics Competition (IARC) as an autonomously flying helicopter that fulfills predefined tasks [4]. To avoid mechanical problems, we decided to use a MARVIN simulator that enables an efficient and safe testing of dangerous flight situation (a requirement to perform Q -Learning). The simulator describes the state space of the helicopter by means of 13 parameters:

- place (d_{forward} , d_{sideward} , d_{altitude})
- velocity (v_{forward} , v_{sideward} , v_{climb})
- angle (ϕ_x , ϕ_y , ϕ_{yaw})
- angular velocity (ω_x , ω_y , ω_{yaw}) and
- velocity of main rotor (ω_{mr})

To control the helicopter there are five steering parameters:

- **STH** enables to steer the main rotor velocity
- **SPC** enables to steer the operating altitude
- **SPH** enables to steer the velocity of heck rotor,
- **SPX** enables to control the sideward vertical pitch
- **SPY** enables to steer the forward vertical pitch.

Because of various interdependencies even simple flight operations require much experience with the helicopter’s behavior. E.g. a higher **SPC** value raises the air drag so that more energy (a higher **STH**-value) is needed to assure a constant main rotor velocity. Because of the complex interdependencies learning an optimal steering policy for a helicopter is a good example for analyzing advanced learning strategies [5] [2].

In our evaluation we analyzed the following task: The helicopter should learn to fly directly to a predefined position and stay there until a new goal position is defined. Thereby the helicopter should control the main rotator so that it runs with a predefined velocity, and align the rear into a predefined direction. Based on these abilities the helicopter is able to fly complex air routes by adapting the goal position dynamically.

4.1 Task decomposition

For decomposing the complex task, we describe the goal state with a formula (written in a conjunctive normal from). In the evaluated scenario we get a formula consisting of five propositions:

- The main rotor runs with the recommended velocity ($\omega_{\text{mr}} = 1200$) **AND**

- The distance and the velocity forward to the goal point are zero ($\delta_{\text{forward}} = 0, v_{\text{forward}} = 0$) **AND**
- The distance and the velocity sideward to the goal point are zero ($\delta_{\text{sideward}} = 0, v_{\text{sideward}} = 0$) **AND**
- The helicopter has reached the defined flying altitude and climb rate is zero ($d_{\text{altitude}} = 0, v_{\text{altitude}} = 0$) **AND**
- The predefined yaw angle is reached ($\phi_{\text{yaw}} = 0, \omega_{\text{yaw}} = 0$).

From each of these five conditions one subtask is derived.

4.2 Computing the “easiest” subtask

For figuring out the easiest subtask we use 5 agent groups each with 5 agents. The agents of each group perform Q -Learning to compute a policy for one subtask whereas each agent controls one of the five control parameters (STH, SPX, SPY, SPH, SPC).

| Task | Controlled Parameter | Success Rate |
|--|----------------------|--------------|
| Steer velocity of main rotor to 1200 min^{-1} | STH | 0.11% |
| | SPX | 0.00% |
| | SPY | 0.00% |
| | SPH | 0.01% |
| | SPC | 0.06% |
| Steer helicopter forward to goal position ($d_{\text{forward}} = 0$) | STH | 0.01% |
| | SPX | 0.01% |
| | SPY | 0.01% |
| | SPH | 0.01% |
| | SPC | 0.02% |
| Steer helicopter sideward to goal position ($d_{\text{sideward}} = 0$) | STH | 0.01% |
| | SPX | 0.01% |
| | SPY | 0.01% |
| | SPH | 0.01% |
| | SPC | 0.01% |
| Steer yaw angle (cockpit orientation) | STH | 0.03% |
| | SPX | 0.03% |
| | SPY | 0.03% |
| | SPH | 0.03% |
| | SPC | 0.02% |
| Steer altitude | STH | 0.00% |
| | SPX | 0.00% |
| | SPY | 0.00% |
| | SPH | 0.00% |
| | SPC | 0.00% |

Table 1: The Table shows the success rates for all of the learned strategies. According to the measured data, the agent controlling the parameter STH in order to steer the main rotor velocity has learned the most successful strategy. Thus controlling the main rotor speed is the “easiest” task.

The learning parameters were set as follows: Minimal number of updates for each state-action-pair: 20000; number of discretization intervals for the controlled parameter: 20; number of discretization intervals for the remaining parameters: 1. The obtained results are presented in Table 1. The success rates show, that steering the velocity of the main rotor by controlling the parameter `STH` is the easiest subtask.

Having figured out the subtask to start with, we determine the parameters that must be considered. In the analyzed example we delegated the task to a group of 17 agents where each agent considers one of the parameters. The computed “Value-Functions” are depicted in Figure 1. Since the learned Value-Function, computed while parameter ω_{mr} was considered, shows the maximal reachable value, the agent that controls the parameter `STH` must take into account the parameter ω_{mr} . In the next steps it has been detected that additionally only the parameter `SPC` (“Collective Pitch”) is relevant.

4.3 Controlling the helicopter based on the learned policies

As the result of the learning procedure we know for every subtask both the relevant command parameter and the parameters that must be considered. Knowing all these dependencies, the learned strategies have been optimized by using a fine-grained discretisation and by smoothing based on neural networks. The evaluation of the learned policy shows that the helicopter successfully learned to fly. An example for steering the helicopter is shown in Figure 2. At first the helicopter climbs to the desired altitude. Then it flies a left turn and stops at the goal position.

5 Agent coordination

The evaluation of our approach shows that an implicit coordination is an adequate strategy for solving complex problem. The use of multi-agent systems allows for modeling the learning procedure as an evolutionary process: In order to figure out the problem to start with several agent communities test alternative strategies trying to outperform the other communities. Within each community the agents compete with each other as well as applying different strategies for the same subtask. Having finished one learning cycle the performance of all agents is compared and the best agent is determined. The learned strategy can now be adopted by other agents and further be improved (e.g. by taking into account more parameters). So the agent communities incrementally acquire knowledge whereby in each learning step only the most successful agents for each subtask can pass their knowledge to other agents. This procedure continues until the overall performance is good enough to solve the complex problem.

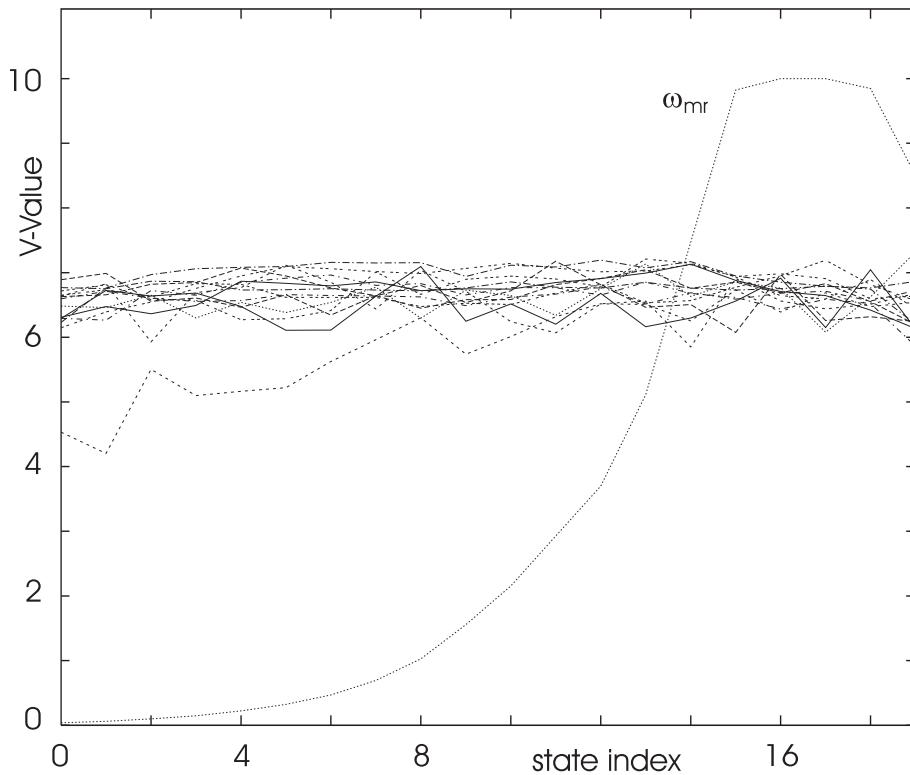


Figure 1: The Figure shows the V -Function for the learned strategies in order to control the main rotor velocity based on parameter STH . For each of these learned policies exactly one parameter is considered. All graphs except one look similar and are almost constant. The V -Function learned while the main rotor velocity has been considered reaches the maximum value. Thus the parameter ω_{mr} must be considered while learning to steer main rotor velocity.

By using incremental learning difficulties can arise, when there are two-way dependencies between agents. In this case a learning sequence might not be determined a priori. In practice, this problem can be solved by building a model for the behavior of one of the involved parameters. In most cases, a sensible assumption is that the parameter values in the actual step are similar to those in the preceding step.

6 Conclusion and future work

The use of multi-agent systems creates new ways to solve complex problems. The decomposition of a problem into a set of bounded task enables to use learn-

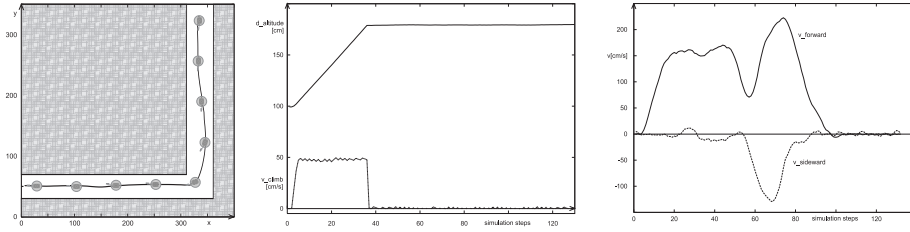


Figure 2: The Figure shows a helicopter flight controlled the learned strategies. The left part depicts the topview: The precomputed route is shown as a white “street”. The effective helicopter route is presented a as a line on that 9 times the helicopter rear alignment is show). The other diagrams show the forward, sideward and vertical speed as well as the operating altitude. The agents (steering the five control parameters) cooperate perfectly, so that complex tasks like adapting yaw angle while flying a sharp turn are managed well.

ing algorithms, which require very little knowledge about the problem. Thus complex problems can be successfully solved without knowing good heuristics or search policies. The drawback of the decomposition is that additional effort is needed for communication and cooperation between the agents. However, multi-agent platforms help to cope with the coordination problem, because multi-agent platforms provide functions for this purpose. Moreover agent platforms support the distributed problem solving, thereby achieving an improvement in scalability and parallel computation.

In contrast to alternative approaches[6][2] the proposed procedure enables to learn the parameter dependencies, so that no additional knowledge about the complex steering task is required. We show that the parameter dependencies can be successfully figured by using Q -Learning. The problem of a long-lasting learning procedure is solved by using a multi-agent framework that supports the intelligent distribution of learning subtask over several agents (hosted on different computers in a network).

The next step to optimize our approach will be an analysis of strategies to find optimal representations for learned knowledge. Another area of research is the dynamic improvement and adaptation of solving policies. In this context important questions are the determination of suitable agents for a certain task, the improvement of reliability and the consideration of time limits.

References

- [1] M. Iwen and A. D. Mali. Interaction graphs for planning problem decomposition. In *AAMAS '02: Proceedings of the first international joint*

- conference on Autonomous agents and multiagent systems*, pages 984–985, New York, NY, USA, 2002. ACM Press.
- [2] E. N. Johnson and S. Kannan. Adaptive flight control for an autonomous unmanned helicopter. In *gnc*, number AIAA-2002-4439, Monterey, CA, aug 2002.
 - [3] J. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge University Press, Cambridge, MA, 2000.
 - [4] M. Musial, G. Hommel, U. W. Brandenburg, E. Berg, M. Christmann, C. Fleischer, C. Reinicke, V. Remuß, S. Rönneke, and A. Wege. MARVIN – Technische Universität Berlin’s Flying Robot Competing at the IARC’99.
 - [5] A. Y. Ng, A. Coates, and e. a. Mark Diel. Inverted autonomous helicopter flight via reinforcement learning. In *International Symposium on Experimental Robotics*, 2004.
 - [6] A. Y. Ng, H. J. Kim, M. I. Jordan, and S. Sastry. Autonomous helicopter flight via reinforcement learning. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.
 - [7] S. K. Sahin Albayrak and J. Stender. Advanced grid management software for seamless services. In *Multiagent and Grid Systems - Smart Grid Technologies and Market Models*, pages 263 – 270, The Printing House, Inc., 2005. IOS Press.
 - [8] R. Sessler. Building agents for service provisioning out of components. In J. P. Müller, E. Andre, S. Sen, and C. Frasson, editors, *Proceedings of the 5. International Conference on Autonomous Agents*, pages 218–219, Montreal, Canada, May 2001. ACM Press.
 - [9] R. S. Sutton and A. G. Barto. *Reinforcement Learning – An Introduction*. MIT Press, 1998.
 - [10] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3/4), May 1992. Special Issue on Reinforcement Learning.