

ASAP-UCT:TO DO

Bachelor thesis

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Artificial Intelligence
<http://ai.cs.unibas.ch>

Examiner: Prof. Dr. Malte Helmert
Supervisor: Dr. Thomas Keller

Tim Steindel
tim.steindel@stud.unibas.ch
2014-050-389

02/11/2017

Acknowledgments

Calculations were performed at sciCORE (<http://scicore.unibas.ch/>) scientific computing core facility at University of Basel.

Abstract



Missing: ABstract
schreiben

Table of Contents

| | |
|-------------------------------------|-----|
| Acknowledgments | ii |
| Abstract | iii |
| 1 Introduction | 1 |
| 2 Background | 3 |
| 3 ASAP-UCT | 7 |
| 4 Evaluation | 9 |
| 5 Conclusion | 10 |
| Bibliography | 11 |
| Appendix A Appendix | 12 |
| Declaration on Scientific Integrity | 13 |

1

Introduction

The idea of robots in everyday life is a dream for quite some time in our society. Be it autonomous cars or just the robotic vacuum cleaner. However not only the hardware is problematic, but also the intelligence behind these robots. A robot vacuum cleaner for example which only cleans a room that is already clean is just as effective as one which permanently switches rooms and never cleans a room completely. So it needs to learn which are good actions in which environment. Furthermore our environment is dynamical and the robot has not the power to determine the outcomes without doubt or has not all the information. For this kind of problem there exists a planner. A planner tries to produce a set of actions that leads to a desired goal given the possible actions and an initial state.

An example for a planning under uncertainty problem is the academic problem. Let us assume we have a student who wants to graduate. To accomplish this, he needs to pass several courses. These courses can have other courses as precondition or overlap in content, furthermore the student can only take one course simultaneously. Therefore the student has several paths to reach his goal and the question now is, how to determine an optimal solution. The problem for our student is to manage the cost respectively the reward and also the probability to pass the courses. So would it be very profitable for the student to write in his first semester his state examination, but his probability to pass would be nearly zero. Hence the student needs to accumulate first costs, before he can get to the point where he gets his reward (his graduation).

For a planner this task is difficult to simulate because there is no immediate reward to gain from taking a course, therefore it is harder to know if the current path is promising. If we calculate all the possible combinations, we get $n! + 2$ states for $n \geq 1 \in \mathbb{N}$. So it is not advisable to use a brute force algorithm.

A traditional approach for an algorithm is to use abstraction. The theory behind abstraction is, that we calculate equivalence classes, so that each class in an equivalence class has the same value and thus reduces the state space.

Another approach are Dynamic Programming algorithms, here we break our problem in sub-

problem down and save the intermediate result. The main advantage is that it gives an optimal solution, but because it needs to save the intermediate results, it needs a lot of memory.

An alternative approach is to use a Monte-Carlo-Tree Search (MCTS) algorithm where we have only a part of the tree and try to find promising action and expand the tree in this direction. They have the nice advantage that they can be stopped anytime and can give a promising action back. A well-known algorithm for this is for example the UCT algorithm [1].

In this paper we combine a UCT-algorithm with abstraction of state, but furthermore we also compute the equivalence class of state-action pairs similar to the paper [2]. We firstly define the theoretical background needed like the Markov decision process, the search tree and UCT. We then introduce ASAP-UCT (Abstraction-State-Action-Pair) as another approach for an algorithm. Afterwards we compare our result with a UCT-based MDP solver [?]. We conclude the paper with a short summary.

2

Background

A problem like the academic problem example can be modeled by a Markov decision process (MDP). The useful property of a MDP is that a new state s' is only dependent of his previous state s and an action a , it is independent of the predecessor of s . Furthermore we construct the MDP, so that it is a dead-end free MDP.

Definition 1. A Markov decision process (MDP) is defined as a 6-tuple $\langle S, A, T, R, s_0, H \rangle$ where:

- S is a finite set of states
- A is a finite set of actions
- $T: S \times A \times S \rightarrow [0, 1]$ is a probability distribution that gives the probability $T(s, a, s')$ that applying action a in s leads to s'
- $R: S \times A \rightarrow \mathbb{R}$ is the reward function
- $s_0 \in S$ is the initial state
- $H \in \mathbb{N}$ is the finite horizon

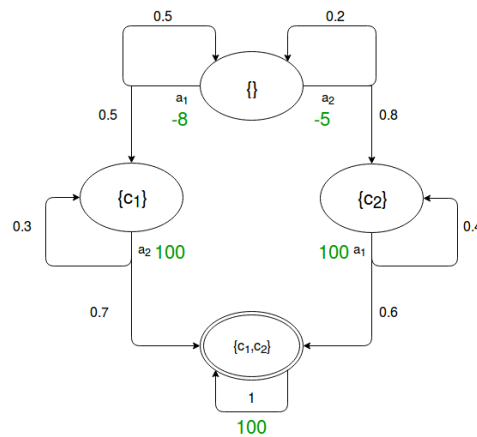


Figure 2.1: an MDP for an academic problem with two courses. Black labels give transition probability, green are labels reward, the action a_i is to take course c_i

The constructed graph is a sufficient model to illustrate the problem and to find a solution, but because most algorithm work better with a tree, we flatten the graph and break cycle open if necessarily. Before we define the tree, we introduce two types of nodes: *decision node* and *chance node*.

Definition 2. Given an MDP $\langle S, A, T, R, s_0, H \rangle$ a decision node is a 2-tuple $\langle s, h \rangle$ where

- $s \in S$ is a state
- $h \in [0, H]$ is the stepstogo to reach a leaf

and a chance node is a 3-Tuple $\langle s, a, h \rangle$ where

- $s \in S$ is a state
- $a \in A$ is an action
- $h \in [1, H]$ is the stepstogo to reach a leaf

These nodes are parts of the tree, which is as a 7-tuple $\langle D, C, E_c, E_d, n_0, L_c, L_d \rangle$ where:

- D is a finite set of decision nodes
- C is a finite set of chance nodes
- $E_c \subseteq D \times C$ are the edges which connect a decision node with a chance node
- $E_d \subseteq C \times D$ are the edges which connect a chance node with decision node
- $n_0 \in D$ is the root node
- $L_c \subseteq C \times A$ is the action of the following chance node
- $L_d \in [0, 1]$ is the probability to get to the following decision node

With the given MDP we can now recursively build our tree with the following properties:

- the root node $\langle n_0, H \rangle$
- For each decision node $d = \langle s, h \rangle \in D$ and each action $a \in A$, there is a chance node $c \in C$ with $c = \langle s, a, h \rangle$, an edge $\langle d, c \rangle$ in E and $L_c(\langle d, c \rangle) = a$
- For each $\langle s, a, h \rangle$ in C where $T(s, a, s') > 0$ and $h > 0$ there is a $d = \langle s, h - 1 \rangle$

The tree has also some invariant properties. So is the root node n_0 a decision node, also decision nodes and chance nodes alternate. Finally all leaves of this tree are decision nodes.

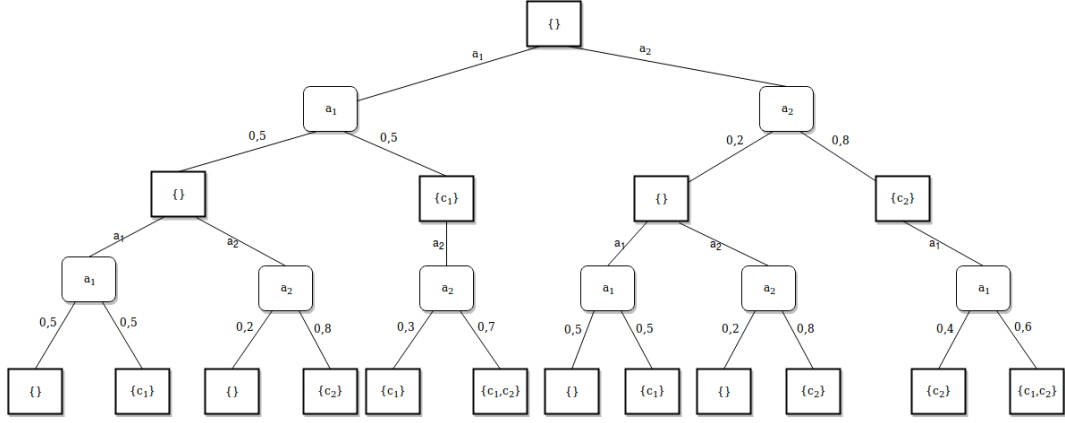


Figure 2.2: the corresponding tree to the MDP 2.1, *chance nodes* have rounded edges, a_i is the action to take course i, for easier understandability we remove the state notation in *chance nodes* and we omit redundant actions

With these definition we can now represent our planing problem, but it would be time-consuming and resource intensive to generate and evaluate the whole tree for each problem. Therefore we use a Monte-Carlo Tree Search algorithm like UCT to evaluate action which are promising and expand the tree in these direction. For this we need to update our definition of decision and chance nodes. A decision node is now a 4-tuple $\langle s, \hat{V}_t, N_t, h \rangle$ and a chance node a 5-tuple $\langle s, a, \hat{Q}_t, N_t, h \rangle$, where

- $\hat{V}_t \in \mathbb{R}$ is the state-value-estimate after t trials
- $\hat{Q}_t \in \mathbb{R}$ is the action-value-estimate based on t trials
- $N_t \in \mathbb{N}$ is the number of visit of this node at trial t

We further add a trial dependency to our tree, so the tree is now $T^t = \langle D, C, E_c^t, E_d^t, n_0, L_c^t, L_d^t \rangle$, also we change the property of our tree so that leaves are chance nodes and not decision nodes anymore. With these changes we can now use the UCT algorithm to construct our tree efficient.

Upper Confidence Bounds to Tree (UCT)

The idea behind a UCT algorithm is to iteratively add nodes to a tree. This step-wise expansion happens in trials t. Now it is the question how to get from our current tree T^t to the next T^{t+1} . The UCT achieve this transit in three phases: a *selection*-, an *expansion*- and a *backup*-phase.

The selection phase consists of an action and an outcome selection. The action selection is used when we are at a decision node d ,here we select the successor node

$$c = \underset{\langle d, c \rangle \in E_c}{\operatorname{argmax}} (Q_t(c) + V_t(d) \sqrt{\frac{N_t(d)}{N_t(c)}})$$

where we have two main component, either the action-value-estimate \hat{Q}_t dominates the function then we exploit the promising candidate or else we explore a less well examined

candidate.

For a chance node we use the outcome selection. Through either its probability or a probability biased by solved siblings, we determine in which state we will end.

When the selection phase reach a decision node, which was not previous been visited and therefore has no children , the expansion phase begins. We now expand all the children of the node and then initialize the reward with a heuristic function. After the expansion phase we have two options. Either we continue the selection phase or we initialize the backup phase.

In the backup phase the tree is been updated. Beginning in the previous expanded node we backtrack our path to the root node. On the way we update the action-value-estimate for chance nodes and the state-value-estimate for decision nodes.

With this framework we can now introduce ASAP (Abstraction of State-Action Pairs) in an UCT environment.

3

ASAP-UCT

The basic idea of ASAP is to make abstraction. In previous algorithm only state pairs were compared for abstraction [3], here we go one steps further and also compare state-action pairs. To save the abstraction we use equivalence classes. For this we introduce two equivalence relation $\sim_d \subseteq D \times D$ and $\sim_c \subseteq C \times C$.

Definition 3. For two decision nodes $d_1 = \langle s_1, h_1 \rangle$ and $d_2 = \langle s_2, h_2 \rangle$ the relation $d_1 \sim_d d_2$ is true if and only if $h_1 = h_2$ and for all $\langle d_1, c_1 \rangle \in E_c^t$ there is a $\langle d_2, c_2 \rangle \in E_c^t$ and vice versa. We denote X as the set of equivalence classes under the relation \sim_d and use $\bar{d} \rightarrow X$ to map each state to the corresponding equivalence class.

Definition 4. For two chance nodes $c_1 = \langle s_1, a_1, h_1 \rangle$ and $c_2 = \langle s_2, a_1, h_2 \rangle$ the relation $c_1 \sim_c c_2$ is true if and only if $h_1 = h_2$ and for all $\langle c_1, d_1 \rangle \in E_d^t$ there is a $\langle c_2, d_2 \rangle \in E_d^t$ where $L_{d_1} = L_{d_2}$ and vice versa.

We denote Y as the set of equivalence classes under relation \sim_c and use $\bar{c} \rightarrow Y$ to map each state to the corresponding equivalence class.

We align each equivalence class a Q-value-mean, this is for X the mean of state-value-estimates and for Y the mean of action-value-estimates.

We expand our tree with the previous declared UCT for a given time-interval τ . After τ we calculate the equivalence classes. We recursively go from the lowest steps-to-go to the root node. The equivalence classes are generated only inside the same level, after following abstraction definition:

Definition 5. *Decision Node Abstraction*

Two decision nodes d_1 and d_2 are in the same abstraction state if and only if $d_1 \sim_d d_2$ is true.

Definition 6. *Chance Node Abstraction*

Two chance nodes c_1 and c_2 are in the same abstraction state if and only if $c_1 \sim_c c_2$.

:

After we reach the root node each node of the expanded tree has now an equivalence-class. Now the phase of generating the equivalence-classes is finished and the UCT algorithm goes again to the selection phase. The next time the UCT goes in the Back-propagation phase, the algorithm use, if the node is in a equivalence class, the Q-value mean of its equivalence class or the value-estimate respectively the action-value-estimate of the node if the node is not yet in a equivalence class. The equivalence classes are newly generated after each interval, there is no reusing of the values.

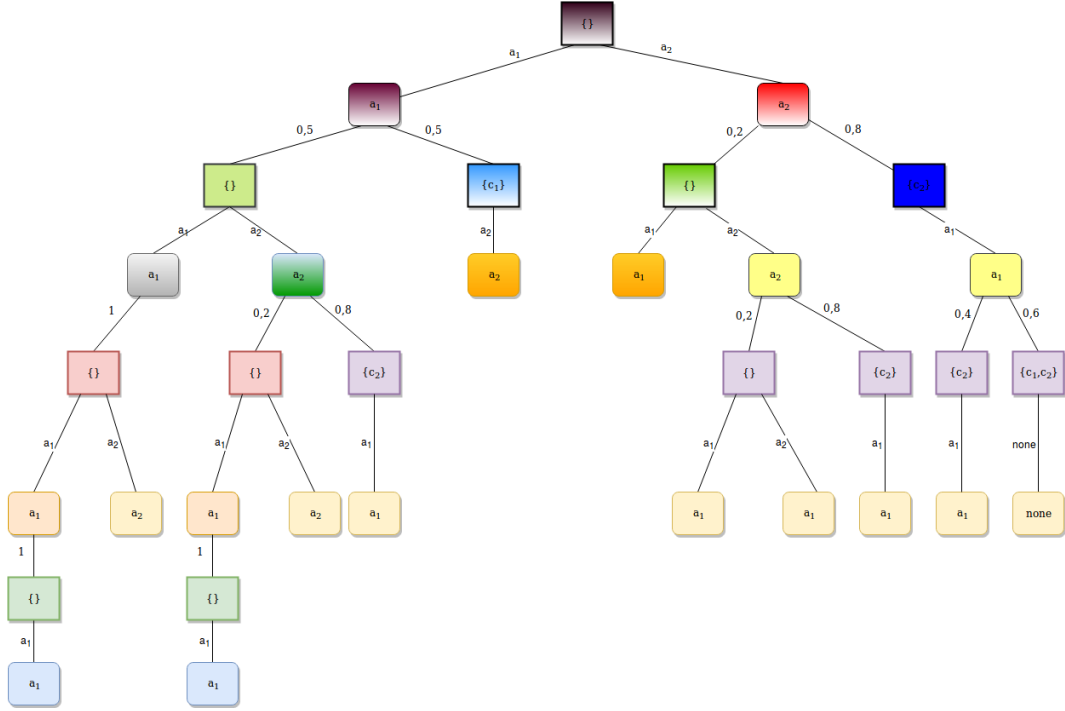


Figure 3.1: after calculating the equivalence classes, each color simulate a different equivalence class

This projection gives us the advantage that we do not have a second abstract tree and therefore have no outdated tree

But because we do not abstract between level, we still have the same depth.

4

Evaluation

5

Conclusion

Bibliography

- [1] Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. *Machine Learning: ECML* (2006).
- [2] Ankit Anand, M., Aditya Grover and Singla, P. ASAP-UCT: Abstraction of State-Action Pairs in UCT. *ar* (2015).
- [3] Robert Givan, M. G., Thomas Dean. Equivalence notions and model minimization in Markov decision process. *Artificial Intelligence*, 147:163–223 (2003).



Appendix

Declaration on Scientific Integrity

Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor

Tim Steindel

Matriculation number — Matrikelnummer

2014-050-389

Title of work — Titel der Arbeit

ASAP-UCT:TO DO

Type of work — Typ der Arbeit

Bachelor thesis

Declaration — Erklärung

I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, 02/11/2017

Signature — Unterschrift