Master of Aerospace Engineering

Research Project

Adaptive Coefficient Based Action Selection Strategy for Automated Probabilistic Planning

S2 Progress report

Author: González Ayuso, Alberto

Due date of report : March 31, 2020
Actual summission date: April 2, 2020

Starting date of project: January 27, 2020. Duration: 14 Months

Tutor: Caroline Chanel

Contents

1 Goal of the project

The main objective of this research project is to endow an autonomous aerial vehicle with the capability of finding the best route between two points in a cluttered environment. This means that the agent must be able to make its own decisions in order to operate under conditions of efficiency, safety and uncertainty.

If the Markov Decision Process (MDP) is the modelling chosen to deal with a such planning problem, there are classical methods that enable to find a policy solution. Nonetheless, these methods are not suitable for an online planning since they require a budget in terms of computational time or memory which is not available in an embedded application. This is the reason why this research project focuses on online versions of the current state-of-the-art algorithms that seem to be promising strategies.

In this context, this project focuses on online planning under uncertainty. In particular, for a given class of online resolution methods such as Monte-Carlo Tree Search (MCTS). MCTS usually depends on an action selection strategy that in turn depends on empirical coefficient values. Finding a such empirical coefficient value in advance is not suitable in the online planning context (no time available to that). Thus, our aim is to study if a previous work [7] could be generalisable and then applied to online MCTS-like algorithms. Succinctly, this work suggests that an entropy-based coefficient could be able to replace these empirical values.

2 Main bibliography and State of the Art

The domain of planning problems under uncertainty became a popular topic of research in 1950s with some early works. Then, more recently, the main models and algorithms were adopted by the Artificial Intelligence community in the 1990s, generating a sharp increase in the number of papers concerning different approaches for these problems. This let us a bibliographical background made of 70 years of progress, development and knowledge. For this reason, getting up to date with the *State of the Art* is the first step to pursue the objective of the research project.

Mausam and Kolobov [1] suggest that the more general frame, which supports such kind of sequential decision making problems, is the Markov Decision Process model, or MDP for short. Therefore, MDPs cover a wide range of problems with a different level of complexity. Features such as partial or full observability, finite or infinite horizon, deterministic or probabilistic transition models, are examples of these different levels of complexity.

These authors also provide a first definition for a Finite Discrete-Time Fully Observable MDP as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{T}, \mathcal{R})$. Expanding on this definition, $\mathcal{S}$ is the finite set of all possible states for the autonomous agent. In each decision epoch (whose sequence may be finite or infinite and it is

represented by $\mathcal{D}$) the agent is able to take some actions that eventually will lead it to a new state. These actions are described in the transition model $\mathcal{T}$. To solve this problem, it is necessary to find a proper policy which tells the agent what to do in a particular state in order to reach the goal or satisfy a particular criterion ($\pi\colon \mathcal{S} \to \mathcal{A}$). In other words, the action selection must be guided and this is the reason why a reward function $\mathcal{R}$ is needed.

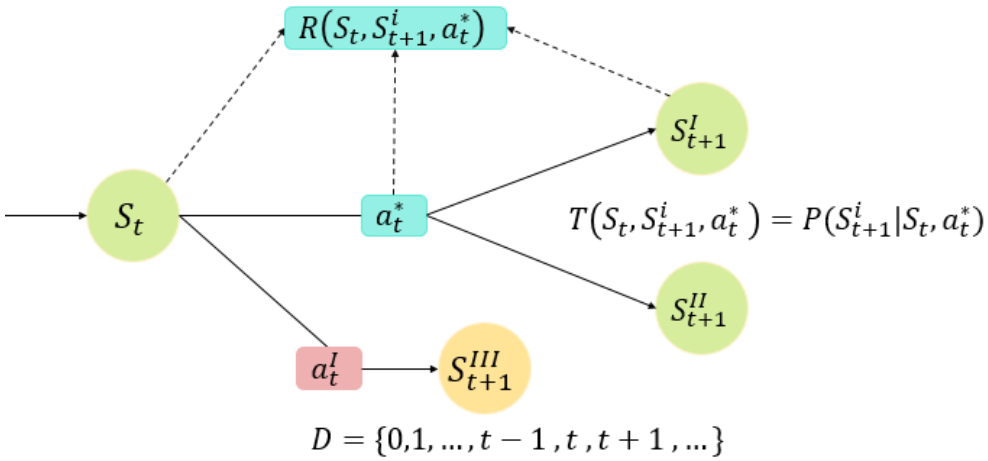

Figure 1: Finite Discrete-Time Fully Observable MDP.

Figure 1 tries to illustrate how the elements of the tuple are linked to each other. In it, it is possible see different outputs rooted at the decision epoch t. This means that, once the agent arrives at s_t, it can move to a new state that depends on the choice between the possible actions and the transition model which gives a probability distribution given the previous state an the action. This figure also emphasises the fact that different actions produce different rewards, thus there exists an optimal action, denoted as a^*_t and given by $\pi(s_t)$, that maximises the expected reward.

Some problems accept more than one solution, thus it is important to remark the fact that researchers have made a huge effort to create algorithms that enable to compute the optimal policy, hereafter called the solution. In a more general case, optimal solution can be hard to find since only small problems can be entirely solved. In this context, near-optimal solution is a more realistic terminology.

As it has been said above, the way different policies are compared is based on the reward (or cost) functions. This means that each time the agent triggers a transition it will earn value (or pay a price) depending on the previous and following states and the action. An easy way to understand MDP solutions could be condensed into the following statement: *'A policy will be considered as optimal when it has the highest accrual value (or lowest accrual cost)'.*

Note that this is a huge simplification. Mausam and Kolobov [1] discuss rigorously the whole

process to prove the solution existence, in which they describe how utility functions are used to provide policies with a quality value. These value functions are, in turn, based on the reward functions.

Once the concept of MDP is more clear, it is possible to go further and define a particular short of problem which derives from MDPs. That is to say, the Stochastic Shortest-Path MDP (SSP) [1]. This model features an infinite number of decision epochs (infinite-horizon) and it is based of cost functions. Consequently, this model fits perfectly the subject problem of this project. However, it is important to remind that this study aims to check if the algorithm form [7] is generalisable to MDPs, not only SSP MDP.

This first theoretical part of this introductory report leaves the formulation of the problem clearly defined. Now the target is to plough through the literature in order to reach the work based on the entropy-based coefficient which is the basis of the project.

Given the MDP models, the *State of the Art* offers a large repertoire of solvers. Roughly speaking they may be divided into three main groups: fundamental algorithms, heuristic search algorithms and Monte-Carlo Tree Search based algorithms. This classification is an adaptation of the one suggested by Mausam and Kolobov [1]. This project aims to implement a version of a method that belongs the last group. The *UCT* method (as in [2],[3] and [4]), is a sample based algorithm that provides an approximation to the optimal policy. The longer *UCT* works, the closer to the optimal policy.

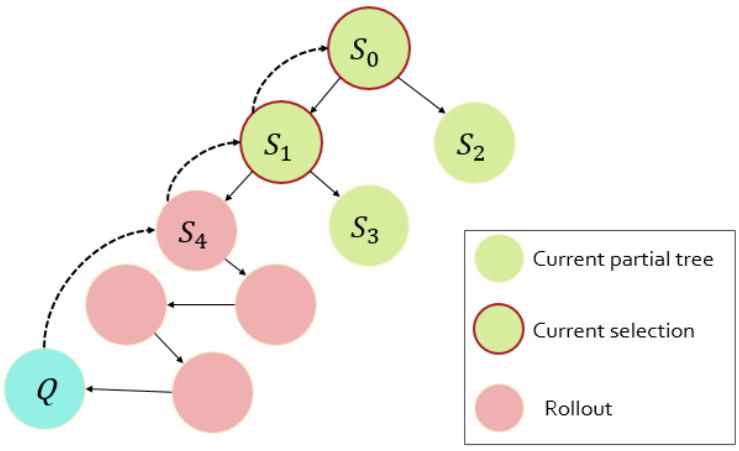

Figure 2: MCTS-UCT algorithm adapted from [2].

UCT stands for Upper Confidence Bounds (for UCB see [2]) applied to trees. This opens a new field of research as deep as the MDPs. Hence, it is necessary to get familiar with Monte Carlo Tree Search Methods. The developments of MCTS techniques are supported by other fields such as decision theory (MDPs), game theory, Monte Carlo methods and bandit-based methods. These

pillars lead to two main hypothesis for MCTS methods [2].

Firstly, these methods claim that the exact value of the actions may be approximated via simulations (rollouts). This simulations or samples, expand the tree exploring new states and actions. Then and secondly, the algorithm backs up the discovered values and makes the tree more accurate as it is built. Figure 2 tries to explain how UCT works. It shows how nodes and actions are selected following a particular criterion and then a rollout is launched to recover and backup information about the Q-value of this new node.

As new nodes are expanded in the tree, a well known dilemma arises. The exploration-exploitation dilemma deals with the conflict between explore apparently suboptimal paths searching long term rewards and exploit the current best policy. This is the point where UCT takes part. This algorithm suggest applying UCB1 to choose between exploit or explore. Upper Confidence Bounds derive from bandit-based methods and they help to resolve the dilemma by setting a limit to the growth of regret (expected loss due to not taking the best action) [2].

The idea is to enhance the performance of this method in order to return admissible policies even when the budget is extremely tight, which is the case of online planning. A promising idea is to work with the exploration coefficient within the UCB. It is proved that it is possible to find an optimal value for this parameter such that the resources needed to find the solution are minimised (or simply the approximation is better with a fixed budget). Nevertheless, this value varies significantly between different planning domains and computing a priori this value in each environment is simply unfeasible.

This research project targets a first research direction concerning a dynamic tuning of the exploration parameter in accordance with entropic criteria. That is to say, explore more where the uncertainty is higher, otherwise exploit [7].

3 Milestones of the project

The intermediate goals of the project can be divided into two main groups. The first one is basically a first contact with the fields related to the research project. By contrast, the following stage will require more technical work as it will be explained later.

The very initial and basic objectives have been gathered under the name of project kick-start. In figure 3 a road map of this initial stage can be observed. There, it is possible to see a break down of activities and the progress made up to this moment. It is clear that the most part of the work suggested for this stage is focused on getting used to all the new concepts, theories, models and algorithms. In fact, not only has the main bibliography been included in this road map, but also the slides of two lectures and basic grasp of coding with a toolbox.

The milestones that remain unaccomplished are focused on reading the papers that empathise

aspects of particular algorithms. There is also a bridge goal that links the bibliography with the technical work and it consists in defining and coding a problem with its solution.

The next group of milestones will follow this dynamics of coding, and they will be useful to check if all the concepts have been clearly understood and continue delving into the topic. They can be seen in figure 4.

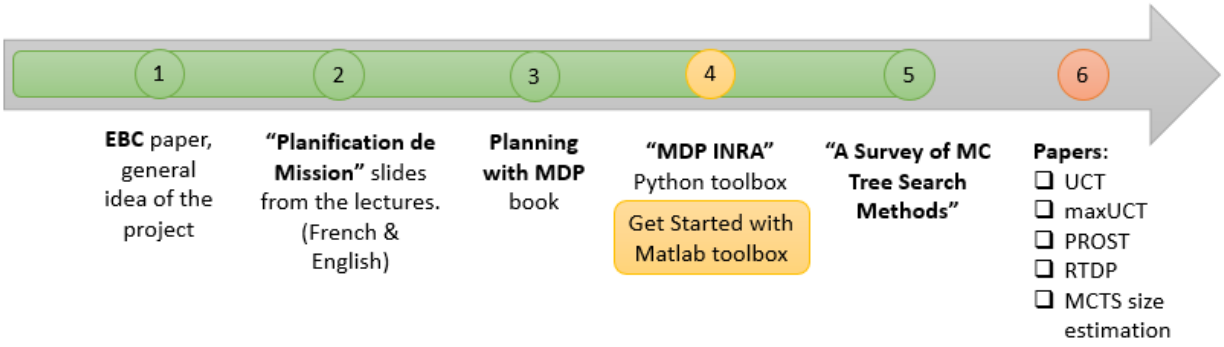

Figure 3: Project Kick-Start road map.

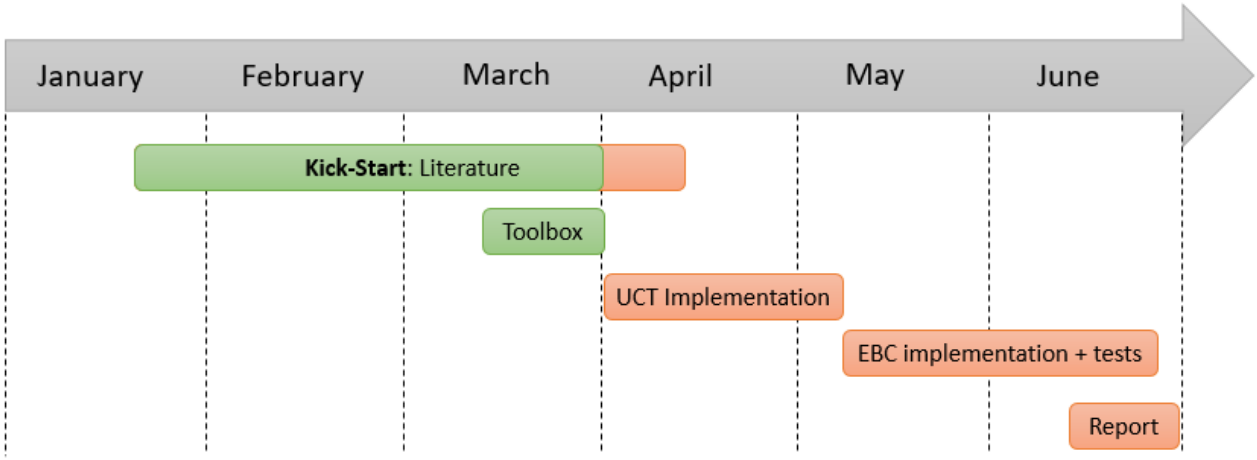

Figure 4: Rough diagram of the organisation of the project.

4 INRA MDP Toolbox and first problems

4.1 Description of the work

The goal of this task is to get familiar with some fundamental algorithms such as Value Iteration and Policy Iteration and also work with problem definitions and typical trouble shooting.

To progress faster, one can take advantage of a given toolbox [8] that can be found here. It was initially created by the Applied Mathematics and Computer Science Unit of INRA Toulouse (France) and it is really useful to solve the problems quickly, check if the problems are well formulated and validate our own algorithms. However, the description of the problems will be totally personal.

4.2 Technical progress

The huge repertoire of functions and examples provided by the toolbox have been explored and tested. Once the operation of the toolbox is clear, the next step consists in describing a problem of our own.

The chosen problem was similar to the Maze's problem from the lectures [9]. In this problem, the agent must travel through a grid map in order to find the goal. Nevertheless, it must take into account some obstacles which have been modelled as dead-ends. Mausan and Kolobov [1] describe dead-ends as states such that there is no policy which enables the agent to reach the goal form them. In the modelling context, this means that if the agent falls into such kind of state, all the actions will always take it to the same state with probability 1. Figure 5 shows the configuration of the grid map and the basic idea behind the transition model.

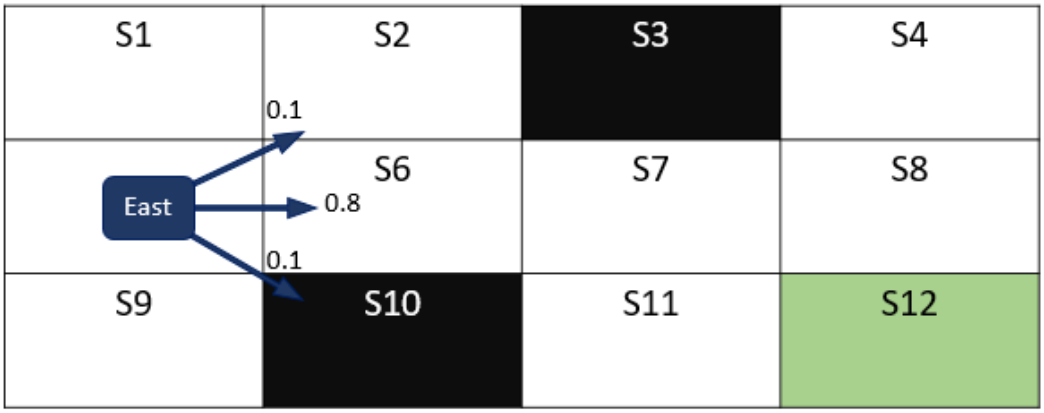

Figure 5: Problem definition.

The chosen transition model is stochastic. This means that the outcome of a action is uncertain to some extent. The probability distribution changes depending on the state and the action. For instance, the outcome of an action that drives the agent towards a wall leaves the agent at the same position.

The problem has been formulated as an Stochastic Shortest Path MDP. Thus, the dead-ends have not been taken into account rigorously (see [1] MDPs with dead-ends). It would have been necessary to define a finite-penalty SSP MDP but the idea was not to make it too complex. The description of the problem can be seen below.

$$\mathcal{S} = \{s_1, \ldots, s_{12}\} \tag{1}$$

$$\mathcal{A} = \{s_1, \ldots, s_5\} = \{North, South, East, West, Stay\} \tag{2}$$

$$\mathcal{T} \colon \mathcal{S} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1] \tag{3}$$

$$\mathcal{T}(s, s', a) = P(s'|s, a) \tag{4}$$

$$\mathcal{C} \colon \mathcal{S} \times \mathcal{S} \times \mathcal{A} \longrightarrow [-\infty, 0] \tag{5}$$

$$\mathcal{G} = \{s_{12}\} \tag{6}$$

It is important to remark the fact that the cost function has been defined as negative rewards in order to use the toolbox whose settings are focused on maximising the value function. It is also noticeable that the cost function has been stored as five matrix that belong to $\mathbb{R}^{12\times12}$. However, the toolbox uses only a matrix such that $C(s, a) \in \mathbb{R}^{12\times5}$. This is not a major issue and the proof will be shown below.

Regarding the strategy to solve the problem, Mausam and Kolobov [1] explain that VI algorithm is based on a set of equations called Bellman equations. They are used to express the optimality principle for SSP MDPs (γ discounted) by giving the optimal value function in terms of the optimal Q-value for a pair sate-action provided the state is not the goal.

$$V^*(s) = \max_{a \in \mathcal{A}} \mathcal{Q}^*(s, a) \tag{7}$$

$$\mathcal{Q}^*(s, a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s, s', a) \cdot (\mathcal{C}(s, s', a) + \gamma V^*(s')) \tag{8}$$

The bellman's operators in both cases (two different cost matrix) seem to be slightly different but they are, in essence, the same. The following equations will show how to link the bellman operators used to compute the back up in the Value Iteration algorithm.

$$V_n(s) \leftarrow \max_{a \in \mathcal{A}} \mathcal{Q}(s,a) \tag{9}$$

$$V_n(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot (\mathcal{C}(s,s',a) + \gamma V_{n-1}(s')) \tag{10}$$

From the last expression, one can define the Q-value as:

$$\mathcal{Q}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot (\mathcal{C}(s,s',a) + \gamma V(s')) \tag{11}$$

Solving the product and splitting the sum, one can reach:

$$\mathcal{Q}(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot \mathcal{C}(s,s',a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot V(s') \tag{12}$$

Where the first term, for a fixed a and s is basically the sum of the products between probabilities and costs. The toolbox gathers these two matrix as PR (or $\mathcal{TC}$ in this notation) that is equivalent to $C(s,a)$. Equation (13) is a sentence of mdp_computePR.m, which is the function in charge of computing this matrix.

$$>> PR(:,a) = sum(P(:,:,a) .* R(:,:,a), 2) \tag{13}$$

So, consequently:

$$\mathcal{Q}(s,a) = \mathcal{C}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot V(s') \tag{14}$$

And finally:

$$V_n(s) \leftarrow \max_{a \in \mathcal{A}} \ \mathcal{C}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot V_{n-1}(s') \tag{15}$$

$$\pi_n(s) \leftarrow arg\max_{a \in \mathcal{A}} \ \mathcal{C}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s,s',a) \cdot V_{n-1}(s') \tag{16}$$

It is important to remember that the costs are negative in this problem, this is why there is a maximisation problem. In addition, the VI algorithm has been coded again in accordance to

the first definition in order to compare and validate the results of the toolbox. The following pseudo-code summarises the Matlab code.

Algorithm 1: Value Iteration

Result: ϵ-optimal Policy π and V
initialisation;
$policy \leftarrow zeros(nStates, 1)$;
$V \leftarrow V_0$;
$n \leftarrow 0$;
$Threshold \leftarrow 2\epsilon\gamma\left(1-\gamma\right)^{-1}$
while $n < maxIter$ **do**
  $n \leftarrow n + 1$;
  for $s \in \mathcal{S}$ **do**
    $V_n(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, s', a\right) \cdot \left(\mathcal{C}\left(s, s', a\right) + \gamma V_{n-1}(s')\right)$;
    $\pi(s) \leftarrow arg \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathcal{T}\left(s, s', a\right) \cdot \left(\mathcal{C}\left(s, s', a\right) + \gamma V_{n-1}(s')\right)$;
    $Res(s) \leftarrow \|V_n(s) - V_{n-1}(s)\|$
  end
  if $max\left(Res\right) < Threshold$ **then**
    epsilon-optimal policy found;
    return
  end
end

Coming back to the definition of the problem, the values inside these matrix have been defined as follows: $C(s, s', a) = -0.05$ for all the s' that are reachable from s taking the action a provided s' is neither a goal nor a dead-end. The exceptions are solved with a higher cost for dead-ends (one hundred times the nominal cost) and zero cost for the goal. Nonetheless, for some reasons this last value must have been replaced by one in order to encourage the agent to find the goal even when the actions are too risky.

After running both approaches to the VI algorithm with a discount factor of 0.95 and an ϵ of 0.01, one can obtain the following results.

East	**South**	**North**	**South**
North	**West**	**East**	**South**
North	**North**	**East**	**North**

Table 1: Optimal policy with zero cost in s_g

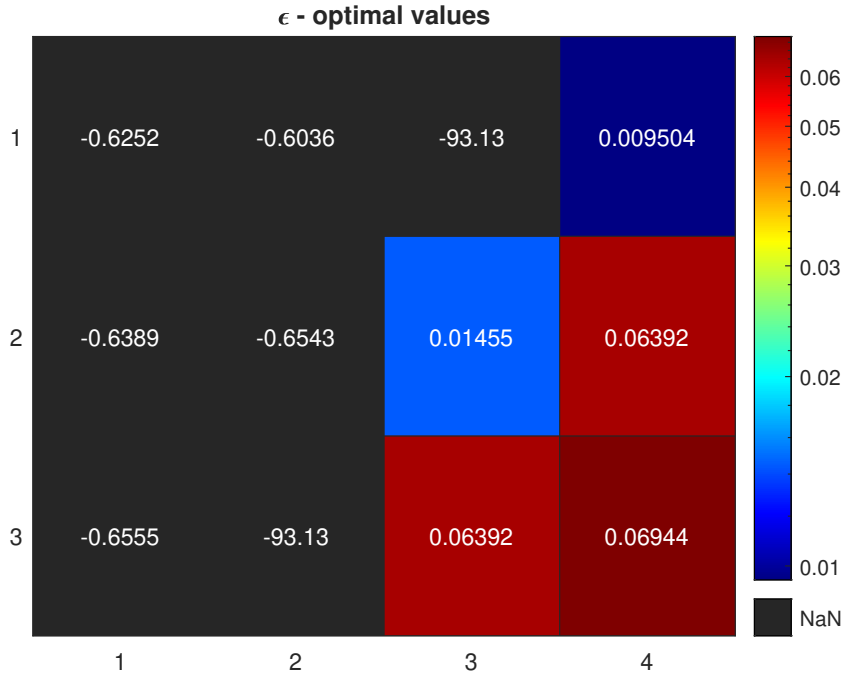

Figure 6: Optimal values with zero cost in s_g.

It is clear that the results are not good enough. Imagine that the initial state is s_1. Following the optimal policy it is impossible for the agent to reach the goal. A new dead-end has been created because of the configuration of the problem. This is due to the fact that moving in circles between s_1, s_2, s_6 and s_5 is still cheaper than taking the risk of going East in s_6. Narrow spaces between dead-ends endanger the quality of the solution.

A way to solve this problem is based on making the goal more attractive for the agent. Setting a reward for the actions reaching the goal, one can encourage going East in s_6.

East	**South**	**North**	**South**
North	**East**	**East**	**South**
North	**North**	**East**	**North**

Table 2: Optimal policy with reward in s_g

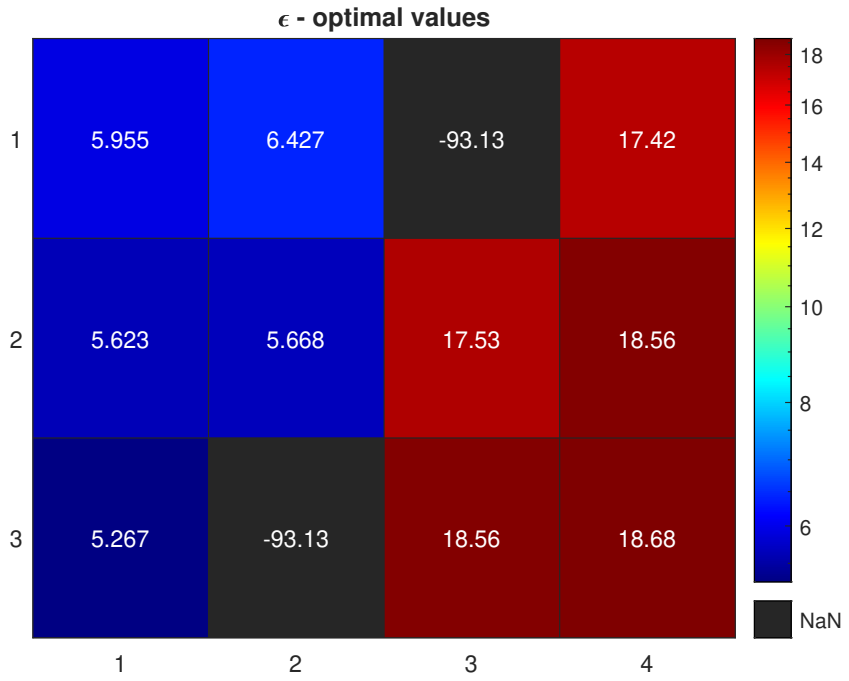

Figure 7: Optimal values with reward in s_g.

4.3 Plan versus achievements

Up to this moment, all the technical progress is based on the Matlab version of the Toolbox. The initial plan was to start directly with the Python version. However, since all this work was oriented to my own understanding this is not a major change. Future work will be implemented on Python.

References

[1] Kolobov, A. and Mausam. *Planning with Markov decision processes: An AI perspective*, Vol. 6, Morgan & Claypool Publishers, 2012.

[2] Cameron B. Browne, Edward Powley, et al. *A Survey of Monte Carlo Tree Search Methods* IEEE Transaction on computational intelligence and AI in games, Vol. 4, No. 1, March 2012.

[3] Levente Kocsis and Csaba Szepesvari. *Bandit based Monte-Carlo Planning. Machine Learning*: ECML, 2006.

[4] Keller, T., and Helmert, M., *Trial-based Heuristic Tree Search for Finite Horizon MDPs* , ICAPS 2013 - *Proceedings of the 23rd International Conference on Automated Planning and Scheduling*, 2013.

[5] Keller, T., and Eyerich, P., *PROST: Probabilistic Planning Based on UCT.* , ICAPS 2012 - *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, 2012.

[6] Bonet, B., and Geffner, H., *Labeled RTDP: Improving the Convergence of Real-Time Dynamic Programming* , ICAPS 2003 - *Proceedings of the 3rd International Conference on Automated Planning and Scheduling*, 2003.

[7] Ana Raquel Carmo, Jean-Alexis Delamer, Yoko Watanabe, Rodrigo Ventura, and Caroline P. C. Chanel. *Entropy-based adaptive exploit-explore coefficient for Monte-Carlo path planning.* In proceedings of the Conference on Prestigious Applications of Intelligent Systems, collocated with the European Conference on Artificial Intelligence, 2020.

[8] Chades I., Chapron G., Cros MJ., Garcia F., Sabbadin R. (2014). *MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems.* Ecography 37:916-920.

[9] Caroline P. Carvalho CHANEL. *Planification de Mission.* DCAS, ISAE-SUPAERO, Toulouse.