<u>Planning and Learning in Dynamical Systems (046194)</u> Homework 1, due 18/4/2019

Question 1 (Counting)

Given an integer number X, we would like to count in how many ways it can be represented as a sum of N natural numbers (with relevance to order) marked by $\Psi_N(X)$. For example, $\Psi_2(3) = 2$ since: 3 = 1 + 2 = 2 + 1.

- a. Find the following: $\Psi_1(2), \Psi_2(4), \Psi_3(2), \Psi_N(N)$ and $\Psi_1(X)$.
- b. Find a recursive equation for $\Psi_{N}(X)$.
- c. Write a code in for finding $\Psi_N(X)$ using dynamic programming.
 - 1. What is the time and memory complexity of your algorithm?
 - 2. Find $\Psi_{12}(800)$.
- d. Now assume each natural number i is associated with some cost c_i . For a given X, N we are interested in finding the lowest cost combination of natural numbers $\left\{x_i\right\}_{i=1}^N$ satisfying $\sum_{i=1}^N x_i = X$.
 - 1. Formulate the problem as a finite horizon decision problem: Define the state space, the action space and the cumulative cost function.
 - 2. Bound the complexity of finding the best combination.

Question 2 (Language model)

In the city of Bokoboko the locals use a language with only 3 letters ('B','K','O'). After careful inspection of this language, researchers have reached two conclusions:

- I. Every word starts with the letter 'B'.
- II. Every consecutive letter is distributed only according to the previous letter as follows:

$$P(l_{t+1} | l_t) = \begin{bmatrix} B & 0.1 & 0.325 & 0.25 & 0.325 \\ K & 0.4 & 0 & 0.4 & 0.2 \\ 0.2 & 0.2 & 0.2 & 0.4 \\ - & 1 & 0 & 0 & 0 \end{bmatrix}$$

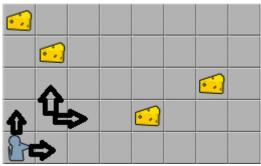
Where '-' represents the end of a word. For example, the probability of the word 'bko' is given by $0.325 \cdot 0.4 \cdot 0.4 = 0.052$.

- a. Find the probability of the following words: 'Bob', 'Koko', 'B', 'Bokk', 'Boooo'.
- b. We wish to find the most probable word in the language of length K.
 - 1. Formulate the problem as a finite horizon decision problem: Define the state space, the action space and the **multiplicative** cost function.
 - 2. Bound the complexity of finding the best combination.
 - 3. Find a reduction from the given problem to an analogous problem with **additive** cost function instead.

- 4. Explain when each approach (multiplicative vs. additive) is preferable. Hint: Think of the consequence of applying the reduction on the memory representation of a number in a standard operating system.
- 5. Write a code which finds the most probable word of a given size using dynamic programming. What is the most probable word of size 5?

Question 3 (Path planning)

Moses the mouse starts his journey at the south west room in a $M \times N$ rectangular apartment with $M \cdot N$ rooms of size 1×1 , some of which contain cheese. After his rare head injury in the mid-scroll war, Moses can only travel north or east. An illustration of Moses's life for M = 5, N = 8 is given below:



Being a mouse and all, Moses wants to gather as much cheese as possible until he reaches the north-east room of the apartment.

- a. Formulate the problem as a finite horizon decision problem: Define the state space, the action space and the cumulative cost function.
- b. What is the horizon of the problem?
- c. How many possible trajectories are there? How does the number of trajectories behaves as a function of N when M = 2? How does it behave as a function of N when M = N?
- d. Aharon, Moses's long lost war-buddy woke up confused next to Moses and decided to join him in his quest (needless to say, both mice suffer the same rare head injury).
 - 1. Explain what will happen if both mice ignore each other's existence and act 'optimal' with respect to the original problem.
 - 2. Assume both mice decided to coordinate their efforts and split the loot. How many states and actions are there now?
 - 3. Now their entire rarely-head-injured division has joined the journey. Assume there's a total of K mice, how many states and actions are there now?

Question 4 (MinMax dynamic programming)

In this problem we consider an adversarial version of finite horizon dynamic programming, which is suitable for solving 2-player games. In this setting, at time $k \in \{0,1,2,\ldots,N-1\}$ the system is at state $s_k \in S_k$, the agent chooses an action $a_k \in A_k(s_k)$ according to the agent policy $\pi_k^a(s_k)$, and subsequently the opponent chooses an action b_k from the set of allowable opponent actions $B_k(s_k,a_k)$, according to the opponent policy $\pi_k^b(s_k,a_k)$.

The system then transitions to a new state according to:

$$s_{k+1} = f_k(s_k, a_k, b_k), \quad k = 0, 1, 2, ..., N-1.$$

The instantaneous reward is denoted by $r(s_k, a_k, b_k)$, and, for an N-stage path $h_N = (s_0, a_0, b_0, \dots, s_{N-1}, a_{N-1}, b_{N-1}, s_N)$ the cumulative reward is

$$R_N(h_N) = \sum_{k=0}^{N-1} r_k(s_k, a_k, b_k) + r_N(s_N).$$

Given s_0 , the agent's goal is to find a policy π_a^* that maximizes the *worst-case* cumulative reward:

$$\pi_a^* \in \arg\max_{\pi_a} \min_{\pi_b} R_N(h_N)$$

- a. Formulate a dynamic programming algorithm for this problem. Explain what the value function represents in this case.
- b. What is the computational complexity of your algorithm?
- c. Could this approach be used to solve the game of tic-tac-toe? Explain what are the states, actions and rewards for this game.
- d. Could this approach be used to solve the game of chess? Explain.