

Managing the Exploration/Exploitation Trade-Off in Reinforcement Learning

Youssef Achbany, Francois Fouss, Luh Yen, Alain Pirotte & Marco Saerens
Information Systems Research Unit (ISYS/IAG)
Université catholique de Louvain
Place des Doyens 1
B-1348 Louvain-la-Neuve, Belgium
{achbany, fouss, yen, pirotte, saerens}@isys.ucl.ac.be

June 1, 2005

Abstract

In this work, we present a model that integrates both exploration and exploitation in a common framework. First of all, we define the concept of **degree of exploration** from a state as the entropy of the probability distribution on the set of admissible actions in this state. This entropy value allows to control the degree of exploration linked to this state, and should be provided by the user. Then, we restate the exploration/exploitation problem as a global optimization problem: define the best exploration strategy that minimizes the expected cumulated cost, while maintaining fixed degrees of exploration. This formulation leads to a set of nonlinear updating rules reminiscent from the “value iteration” algorithm. Interestingly enough, when the degree of exploration is zero for all states (no exploration), these equations reduce to Bellman’s equations for finding the shortest path while, when it is maximum, a full “blind” exploration is performed. We further show that if the graph of states is directed and acyclic, the nonlinear equations can easily be solved by performing a single backward pass from the destination state. The theoretical results are confirmed by simple simulations showing that the model behaves as expected.

1. Introduction

Balancing **exploration** and **exploitation** is an important issue in reinforcement learning. Exploration aims to continually try new ways of solving the problem, while exploitation aims to capitalize on already well-established solutions. Exploration is of course especially important when the environment is changing. In this case, good solutions can deteriorate over time, or new, better, solutions can appear over time. Without exploration, the system would not be aware of these facts, and the system would inevitably deteriorate over time. One of the key features of reinforcement learning is that it explicitly considers the exploration/exploitation problem in an integrated way; this is one of its specificities [16], in contrast with, for instance, Markov decision processes [3], [4], [13].

Usually, one tackle this exploration problem by using a probabilistic approach for selecting actions (choice randomization). In other words, it is common to assign a probability distribution on the set of admissible actions so that, at some fixed interval of time, an action will be chosen stochastically with, however, a clear preference for “good solutions” [10]. This way, the system readjusts its policy periodically by exploring up-to-now sub-optimal actions. Generally, the agent chooses action u_i with a probability that is a function of the immediate cost associated with this action (usually, a softmax action selection; see [16], [10]). However, as we show in this paper, this strategy is sub-optimal because it is accounted for in an ad-hoc manner: it does not integrate the exploration issue within the optimization process. Stated in other words, the same optimization strategy is used whether or not exploration is performed.

The objective of this paper is precisely to propose a model that integrates both exploration and exploitation in a common framework. In order to simplify the analysis, we concentrate on a “*stochastic shortest-path problem*”, as defined in [4] and described later in the next section; we are currently working on extensions to “average cost per state” as well as full stochastic problems. To this end, we define the concept of **degree of exploration** from a state as the entropy [8] related to the probability distribution of the set of admissible actions on this state.

This entropy value allows to control the degree of exploration linked to this state, and is provided a priori by the user. When the entropy is zero, no exploration is performed from this state, while when the entropy is maximal, a full, blind, exploration, with equal probability of choosing any action, is performed. Then, we restate the exploration/exploitation dilemma as a global optimization problem: define the best exploration strategy (the probability distribution of choosing an action in a given state) that minimizes the expected cumulated cost from the initial state while maintaining a fixed degree of exploration. This problem leads to a set of nonlinear equations defining the optimal solution. These equations, which are similar to Bellman’s equations, can be solved by iterating them until convergence, which is proved for a particular initialization strategy. They provide the action policy (the probability distribution of choosing an action in a given state) that minimizes the average cost from the initial state to the destination, or terminal, state, for a given degree of exploration.

Interestingly enough, when the degree of exploration is zero for all states, the nonlinear equations reduce to Bellman’s equations for finding the shortest path from the initial state to the destination (solution) state. On the other hand, when the degree of exploration is maximum for all states, the nonlinear equations reduce to the linear equations allowing to compute the average cost for reaching the solution from the initial state in a Markov chain with transition probabilities equal to the inverse of the number of admissible actions for each state (full blind exploration).

The main drawback of this method is that it is computationally demanding since it relies on iterative algorithms like the value iteration algorithm. We however show that if the graph of states is directed and acyclic, the equations can easily be solved by performing a single backward pass from the destination state. One way to obtain such a directed, acyclic, graph is to use Dial’s procedure, popular in the transportation networks field [6].

Section 2 introduces the notations, the standard stochastic shortest-path problem and the way we manage exploration. Section 3 describes our procedure for solving

the stochastic shortest-path problem with exploration. The particular case where the graph of states is directed and acyclic is treated in Section 4. Some numerical examples are shown in Section 5, while Section 6 is the conclusion.

2. Statement of the problem and notations

2.1. Statement of the problem

In this work, we analyze reinforcement learning problems involving exploration. For the sake of simplicity, we will concentrate here on what is called “*stochastic shortest-path problems*”.

In a stochastic shortest-path problem, we assume that during every state transition a bounded cost, given by $C(u(s_t)|s_t) \geq 0$, is incurred where s_t is the current state at time t and $u(s_t)$ contains a control action selected from a set of admissible actions, or choices, $U(s_t)$, available in state s_t . The cost $C(u(s_t)|s_t)$ can be viewed as the cost of performing action $u(s_t)$ given that the agent is in state s_t , and is assumed to be bounded. The **control action** $u(s_t)$ at any time t is determined according to a **policy** π that maps every state k on the set of admissible actions $U(k)$ with a certain probability distribution, $P(u(k)|s = k)$, with $u(k) \in U(k)$. Thus the policy associates to each state k a probability distribution on the set of admissible actions $U(k)$: $\pi \equiv \{P(u(1)|s = 1), P(u(2)|s = 2), \dots\}$ with each $u(k) \in U(k)$. For instance, if the admissible choices in state k are $U(k) = \{u_1^k, u_2^k, u_3^k\}$, the probability distribution $P(u(k)|s = k)$ involves three values, $P(u(k) = u_1^k|s = k)$, $P(u(k) = u_2^k|s = k)$, and $P(u(k) = u_3^k|s = k)$. Under a so-called stationary policy π , the control depends only on the current state s .

The decision process is thus **randomized** because, for each state, the agent chooses the next action according to the probability distribution $P(u(k)|s = k)$. Randomization is introduced in order to guarantee a given **degree of exploration** that will be controlled by the entropy of the probability distribution; randomized choices are very common in a variety of fields; for instance game theory (called mixed strategies in this context; see for instance [12]) or decision sciences [14]. We do not authorize, however, that an agent returns to the initial state; in other words, we do eliminate all the control actions that let us return to the initial state k_0 .

Moreover, we assume here that once the action has been chosen, the next state s_{t+1} is known in a deterministic way, $s_{t+1} = f(u(s_t)|s_t)$ where f is a function. It provides the next state s_{t+1} , given that we are in state s_t and we choose action $u(s_t)$. Therefore, the stochastic nature of the problem is due to the fact that the choices are randomized.

The goal is to minimize the total cost over an infinite number of steps,

$$V_\pi(s = k_0) = E_\pi \left[\sum_{t=0}^{\infty} C(u(s_t)|s_t) \mid s_0 = k_0 \right] \quad (2.1)$$

which is the total expected cost accumulated over an infinite horizon, given the initial state k_0 . The expectation is taken on the policy, that is, on all the random variables $u(k)$ associated to the states k .

As in Bertsekas [4], we assume that there is a special cost-free **destination state** (the solution state); once the system has reached that state, it remains there at no

further cost. We also consider a problem structure such that termination is inevitable, at least under an optimal policy. Thus, the horizon is in effect finite, but its length is random and may be affected by the policy being used. The conditions for which this is true are, basically, linked to the fact that the destination state can be reached in a finite number of steps from any potential initial state; for a rigorous treatment, see [2], [4].

2.2. Computation of the total expected cost for a given policy

The essence of the problem is thus to reach the destination state $s = d$ with minimal expected cost from the initial state k_0 ; as already stated before, this problem is usually called the stochastic shortest-path problem. The deterministic shortest-path problem is obtained as a special case where, for each state, the control action is determined in an univocal way.

This problem can be represented as a Markov chain where each original state and each action is a state. The destination state is then considered as an absorbing state with no outgoing link. In this framework, the problem of computing the expected cost (2.1) from any state k is closely related to the computation of the average first-passage time in the associated Markov chain [9]. The **average first-passage time** is the average number of steps a random walker starting from the initial state k_0 will take in order to reach destination state d , and can be easily generalized in order to take the costs of the transitions into account. By first-step analysis (see for instance [17]), we show in Appendix A that, once the policy is fixed, $V_\pi(k)$ can be computed through the following equations

$$\begin{cases} V_\pi(s = k) = \sum_{i \in U(k)} P(u(k) = i | s = k) [C(u(k) = i | s = k) + V_\pi(s = k'_i)], \\ \quad \text{with } k'_i = f(u(k) = i | s = k) \text{ and } k \neq d \\ V_\pi(s = d) = 0, \text{ where } d \text{ is the destination state} \end{cases} \quad (2.2)$$

Thus, in equation (2.2), k'_i is the state resulting from the application of control action $u(k) = i$ in state k . In the sequel, we will use the following shortcuts for the different quantities of interest: $V_\pi(s = k) = V_\pi(k)$, $C(u(k) = i | s = k) = C(i | k)$, $f(u(k) = i | s = k) = f(i | k)$ and $P(u(k) = i | s = k) = P(i | k)$.

(2.2) is a system of linear equations that can be solved by iterating the equations or by inverting the so-called fundamental matrix [9]; it is analogous to Bellman's equations in Markov decision processes.

Notice that the same framework can easily handle multiple destinations problems by defining one absorbing state for each destination. If the destination states are d_1, d_2, \dots, d_m , we have

$$\begin{cases} V_\pi(k) = \sum_{i \in U(k)} P(i | k) [C(i | k) + V_\pi(k'_i)], \\ \quad \text{where } k'_i = f(i | k) \text{ and } k \neq d_1, d_2, \dots, d_m \\ V_\pi(d_j) = 0 \text{ for all destination states } d_j, j = 1, \dots, m \end{cases} \quad (2.3)$$

We now have to address two questions: (1) how do we control the randomized choices, i.e. exploration, and (2) how do we compute the optimal policy for a given degree of exploration.

2.3. Controlling exploration by fixing entropy at each state

Now that we have introduced the problem, we will explain how we manage exploration. At each state k , we define the **degree of exploration** E_k by

$$E_k = - \sum_{i \in U(k)} P(i|k) \log(P(i|k)) \quad (2.4)$$

which is simply the entropy of the probability distribution of the control actions in this state [5], [8]. This degree of exploration may vary from state to state, and is provided by the user. It measures the uncertainty about the choice at each state and is equal to zero when there is no uncertainty at all (the distribution $P(i|k)$ reduces to a Kronecker delta), or is equal to $\log(n_k)$, where n_k is the number of admissible choices at node k , in the case of maximum uncertainty, $P(i|k) = 1/n_k$ (a uniform distribution).

Furthermore, the **exploration rate** E_k^r at state k is defined as the ratio between the degree of exploration and the maximum degree for that state,

$$E_k^r = \frac{E_k}{\log(n_k)} \quad (2.5)$$

and takes its values in the interval $[0, 1]$.

Fixing the entropy at a state sets the exploration level from this state; increasing the entropy increases exploration up to the maximal value, in which case there is no more exploitation since the next action is chosen completely at random, with a uniform distribution, without taking the costs into account.

3. Optimal policy with exploration constraints

3.1. Optimal policy and expected cost

We now turn to the problem of determining the optimal policy under exploration constraints. More precisely, we will seek the policy, that is, the probability distributions of the actions within the states, $\pi = \{P(u(1)|s = 1), P(u(2)|s = 2), \dots\}$, for which the expected cost $V_\pi(k_0)$ from initial state k_0 is minimal while maintaining a given degree of exploration on the states, E_k . Before going into the details, remember that, whatever the chosen policy, the system will remain a Markov chain and consequently Equation (2.2) remains valid for any admissible policy. The problem is thus to find the transition probabilities leading to the minimal expected cost, and can be formulated as a constrained optimization problem involving a Lagrange function.

In Appendix B, we derive the optimal probability distribution of control actions within a state k , which appears to be a logit distribution

$$P(i|k) = \frac{\exp[-\theta_k (C(i|k) + V^*(k'_i))]}{\sum_{j \in U(k)} \exp[-\theta_k (C(j|k) + V^*(k'_j))]}, \quad (3.1)$$

where $k'_i = f(i|k)$ is a following state and V^* is the optimal (minimum) expected cost

given by

$$\begin{cases} V^*(k) = \sum_{i \in U(k)} P(i|k) [C(i|k) + V^*(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \\ V^*(d) = 0, \text{ for the destination state } d \end{cases} \quad (3.2)$$

In Equation (3.1), θ_k must be chosen in order to satisfy

$$\sum_{i \in U(k)} P(i|k) \log(P(i|k)) = -E_k \quad (3.3)$$

for each state k , and takes its values in $[0, \infty]$. This last equation guarantees a fixed exploration level (entropy) at each state. Of course if, for some state, the number of possible control actions reduces to one (no choice), no entropy constraint is introduced.

Equation (3.1) has a nice appealing interpretation: choose preferably (with highest probability) action i leading to state k'_i of lowest expected cost, including the cost of performing the action, $C(i|k) + V^*(k'_i)$. Thus, the agent is routed preferably to the state which is nearest (in average) to the destination state.

Since Equation (3.3) has no analytical solution, θ_k must be computed numerically in terms of E_k . This is in fact quite easy since it can be shown that the function $\theta_k(E_k)$ is strictly monotonic decreasing, so that a line search algorithm (such as the bisection method, see [1]) can efficiently find the θ_k corresponding to a E_k .

3.2. Computation of the optimal policy

Equations (3.1) and (3.2) suggest an iterative procedure very similar to the well-known value iteration algorithm for the computation of both the expected cost and the policy.

1. Initialization phase:

- Initialize all the $P(i|k)$ to

$$P(i|k) \leftarrow \frac{\exp[-\theta_k C(i|k)]}{\sum_{j \in U(k)} \exp[-\theta_k C(j|k)]} \quad (3.4)$$

where θ_k is set in order to respect the prescribed degree of entropy at each state (see Equation (3.3)). Here, the probabilities only depend on the costs of the actions; this defines a Markov chain with fixed transition probabilities.

- Initialize the $V(k)$ by computing the corresponding expected cost asynchronously for each visited state k :

$$\begin{cases} V(k) \leftarrow \sum_{i \in U(k)} P(i|k) [C(i|k) + V(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \\ V(d) \leftarrow 0, \text{ for the destination state } d \end{cases} \quad (3.5)$$

until convergence. This is the standard iterative procedure for computing the expected cost until absorption in a Markov chain, with transition probabilities given by (3.4) (see [9]).

2. Computation of the policy and the expected cost under exploration constraints:

- For each visited state k :
- Update the probability distribution of the state by:

$$P(i|k) \leftarrow \frac{\exp[-\theta_k (C(i|k) + V(k'_i))]}{\sum_{j \in U(k)} \exp[-\theta_k (C(j|k) + V(k'_j))]}, \quad (3.6)$$

where $k'_i = f(i|k)$ and θ_k is set in order to respect the prescribed degree of entropy (see Equation (3.3)).

- Update the expected cost asynchronously:

$$\begin{cases} V(k) \leftarrow \sum_{i \in U(k)} P(i|k) [C(i|k) + V(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \\ V(d) \leftarrow 0, \text{ where } d \text{ is the destination state} \end{cases} \quad (3.7)$$

The convergence of these updating equations is proved in Appendix C. However, the described procedure is computationally demanding since it relies on iterative procedures like the value iteration algorithm in Markov decision processes.

Notice also that, while the initialization phase (3.4) is necessary in our convergence proof, other simpler initialization schemes could also be applied, for instance,

$$\begin{cases} P(i|k) \leftarrow \frac{\exp[-\theta_k C(i|k)]}{\sum_{j \in U(k)} \exp[-\theta_k C(j|k)]} \\ V(k) \leftarrow 0, \text{ for all states } k, \end{cases} \quad (3.8)$$

instead of (3.4)-(3.5) and then directly apply (3.6) and (3.7).

While convergence is not proved in this case, we observed that this updating rule works well in practice; in particular, we did not observe any convergence problem – it is indeed this rule that will be used in our experiments.

3.3. Some limiting cases

We will now show that when the degree of exploration is zero for all states, the nonlinear equations reduce to Bellman's equations for finding the shortest path from the initial state to the destination (solution) state.

Indeed, from Equations (3.4)-(3.7), if the parameter θ_k is very large, which corresponds to a near-zero entropy, the probability of choosing the action with the lowest value of $(C(i|k) + V(k'_i))$ dominates all the others. In other words, $P(j|k) = 1$ for the action j corresponding to the lowest average cost (including the action cost), while $P(i|k) = 0$ for the other alternatives $i \neq j$. Equations (3.7) can therefore be rewritten as

$$\begin{cases} V(k) \leftarrow \min_{i \in U(k)} [C(i|k) + V(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \\ V(d) \leftarrow 0, \text{ where } d \text{ is the destination state} \end{cases} \quad (3.9)$$

which are Bellman’s equations for finding the shortest path to the destination state.

On the other hand, when $\theta_k = 0$, the choice probabilities reduce to $P(i|k) = 1/n_k$, where n_k is the number of admissible actions in state k , and the degree of exploration is maximum for all states. In this case, the nonlinear equations reduce to the linear equations allowing to compute the average cost for reaching the destination state from the initial state in a Markov chain with transition probabilities equal to $1/n_k$. In other words, we then perform a “blind” exploration, without taking the costs into consideration.

Any intermediary setting $0 < E_k < \log(n_k)$ leads to an optimal exploration vs. exploitation strategy minimizing the expected cost, and favoring short paths to the solution.

In Appendix B, we further show that if the graph of states is directed and acyclic, the nonlinear equations can easily be solved by performing a single backward pass from the destination state.

4. Optimal policy for directed, acyclic, graphs

4.1. Definition of a directed acyclic graph (Dial’s procedure)

In the case where the states form a directed, acyclic graph, the procedure described in the previous section simplifies greatly. We will first describe a procedure, proposed by Dial [6], which allows to simplify the model in order to obtain an acyclic process. An acyclic process is a process for which there is no cycle, that is, one can never return to the same state. Dial [6] proposed the following procedure allowing to compute an acyclic process from the original one:

- First, compute the minimum cost from the initial state to any other state (by using for instance Bellman’s equations). The minimum cost to reach any state k from the initial state k_0 will be denoted as $D(k|k_0)$.
- Then, only consider actions leading to a state with a higher minimum cost from the initial state; all the other actions being labeled as “inefficient” and eliminated. Indeed, it is natural to consider that “efficient paths” should lead away from the origin. Thus, all the actions leading to a state transition $k \rightarrow k'$ for which $D(k_0|k') \leq D(k_0|k)$ are prohibited; the others are “efficient actions” and are allowed.

This results in an acyclic graph since all the states can be ordered by increasing $D(k_0|k)$ and only transitions from a lower $D(k_0|k)$ to a higher $D(k_0|k')$ are allowed. We consequently relabel the states by increasing D , from 0 (initial state) to n (ties are ordered arbitrarily). The label of destination state is still noted d . Of course, this procedure can be adapted to the problem at hand. For instance, instead of computing the minimum cost from the initial state, one could compute the cost to the destination state $D(d|k)$, and eliminate the actions according to this criterion. Other measures, different from the minimum cost, could be used as well.

4.2. Computation of the optimal policy

In this case, if $U(k)$ now represents the set of “efficient actions” for each state, the algorithm (3.6)-(3.7) reduces to

1. **Initialization phase:** $V(d) = 0$, for the destination state d .
2. **Computation of the policy and the expected cost under exploration constraints:**

For $k = (d - 1)$ to initial state 0, compute:

$$\begin{cases} P(i|k) = \frac{\exp[-\theta_k (C(i|k) + V(k'_i))]}{\sum_{j \in U(k)} \exp[-\theta_k (C(j|k) + V(k'_j))]}, \\ V(k) = \sum_{i \in U(k)} P(i|k) [C(i|k) + V(k'_i)], \text{ with } k \neq d \end{cases} \quad (4.1)$$

where $k'_i = f(i|k)$ and θ_k is set in order to respect the prescribed degree of entropy at each state (see Equation (3.3)).

The fact that the states have been relabeled by increasing D and the efficient actions always lead to a state with higher label guarantees that the values of the expected cost at states k'_i are known when using formula (4.1).

5. Discounted problems

In this section, instead of Equation (2.1), we consider **discounted problems**, for which there is a discount factor $0 < \alpha < 1$,

$$V_\pi(k_0) = E_\pi \left[\sum_{t=0}^{\infty} \alpha^t C(u(s_t)|s_t) | s_0 = k_0 \right] \quad (5.1)$$

The meaning of α is that future costs matter to us less than the same costs incurred at the present time. In this situation, we will see that there is no need to assume the existence of a destination state d .

Indeed, this problem can be converted to a stochastic shortest-path problem for which the analysis of previous sections holds (see [4]). Let $s = 1, \dots, n$ be the states and consider an associated stochastic shortest-path problem involving, as for the original problem, the states $s = 1, \dots, n$ plus an extra *termination, absorbing, state* t , with state transitions and costs obtained as follows: from a state k , when a control action $i \in U(k)$ is chosen with probability $P(i|k)$, a cost $C(i|k)$ is incurred and the next state is $k'_i = f(i|k) \neq t$ with a fixed probability α , or t with probability $(1 - \alpha)$. Once the agent has reached state t , he remains in this state forever with no additional cost. Here, absorbing state t plays the same role as destination state d in previous section and our previous stochastic shortest path model can easily be adapted to this framework, leading to the following optimality condition

$$V^*(k) = \alpha \sum_{i \in U(k)} P(i|k) [C(k, i) + V^*(k'_i)], \text{ with } k'_i = f(i|k), 0 < \alpha < 1 \quad (5.2)$$

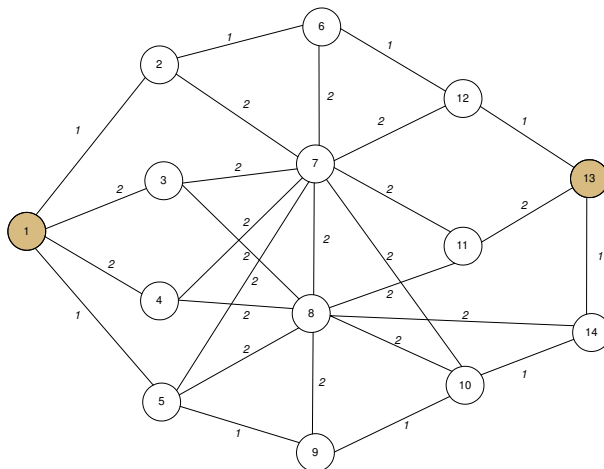


Figure 6.1: The graph used in our experiments. The initial (source) state is state 1 while the destination state is state 13. The initial costs are indicated on the edges.

where, as before, $P(i|k)$ is given by

$$P(i|k) = \frac{\exp[-\theta_k (C(i|k) + V^*(k'_i))]}{\sum_{j \in U(k)} \exp[-\theta_k (C(j|k) + V^*(k'_j))]} \quad (5.3)$$

which are the corresponding optimality conditions for discounted problems.

6. Simulation results

Our experiments were performed on a graph composed of 14 nodes connected by edges of different weights representing costs.

The algorithm described in this paper is used in each experiment. It searches an optimal path in a given graph, by going from a starting node to a destination node while fixing the exploration rate. We will investigate how the algorithm reacts when we vary the value of the entropy (exploration rate), and the impact of these values on the total expected cost. The experiments use the algorithm (3.6)-(3.7) to compute the minimal cost.

Two simple experiments were performed, testing the algorithm's capacity to find the optimal paths in two different settings: (1) a static environment (i.e., an environment where the weight of the edges does not change over time); (2) a dynamic environment.

In the first experiment, we analyze the paths used by the agents to move from one node to another in a static environment. In the second experiment, we observe how the algorithm reacts in a dynamic environment, when the weight of the edges varies over time. We repeated both experiments for four values of the entropy.

In each experiment, the matrices of expected costs and transition probabilities are updated every hundred steps according to Equations (3.6) and (3.7) while a step

represents the complete routing of a agent from its source to its destination. At the beginning of each simulation, the $V(k)$ and $P(i|k)$ are initialized according to Equation (3.8). The exploration rate is fixed to a specified value common to each state.

6.1. First experiment

6.1.1. Description

During this first experiment, we send 15,000 agents through the network shown in Figure 6.1. The initial and destination nodes are node 1 and node 13. The costs of the external edges (i.e., the edges on the paths $[1,2,6,12,13]$ and $[1,5,9,10,14,13]$) are initialized to 1, while the costs of all other edges are initialized to 2.

The goal of this simulation is to observe the paths followed by the routed agents, in function of various values for the entropy. The updating rule used here is (3.6)-(3.7), while initialization is given by (3.8).

We simulate the agents transfer by assigning the same value of the exploration rate at each node. Each node corresponds to a state and the action corresponds to the choice of the next edge to follow. We repeat the experiment four times with, for each of them, a fixed value of the entropy corresponding to a percentage of the maximum entropy (i.e., exploration rates of 0%, 30%, 60% and 90%).

6.1.2. Results

The following graphs, displayed in Table 6.1, show the behaviour of the algorithm when varying the values of the exploration rate. For each tested value, a graph (with the same topology as described above) is used to represent the results: the more an edge is used in the routing, the more its grey level and width are important.

Graph (a) shows the results when sending the 15,000 agents by using an exploration rate fixed at 0%. When exploration rate is zero, the algorithm finds the shortest path from node 1 to node 13 (path $[1,2,6,12,13]$), and no exploration is carried out: $[1,2,6,12,13]$ is the only path used for sending the agents.

Graph (b) shows the followed paths for an entropy of 30% of the maximum value for each node. The path $[1,2,6,12,13]$, corresponding to the shortest path, is still the most used, while other edges not belonging to the shortest path are now also investigated. Moreover, we observe that the second shortest path, i.e. path $[1,5,9,10,14,13]$ also conveys a significant traffic here. Notice however that, despite the rise of exploration, the path mostly used by the algorithm remains the shortest path. Exploitation thus remains a priority while exploration is nevertheless present.

Graph (c) shows the paths used for an exploration rate of 60%. As for the previous experiment, the algorithm uses more and more edges that are not part of the shortest path, but still prefers the shortest path.

Graph (d) shows the result with an exploration rate of 90%. With such a value, the exploitation of the shortest path is no more the priority; exploration is now clearly favored over exploitation.

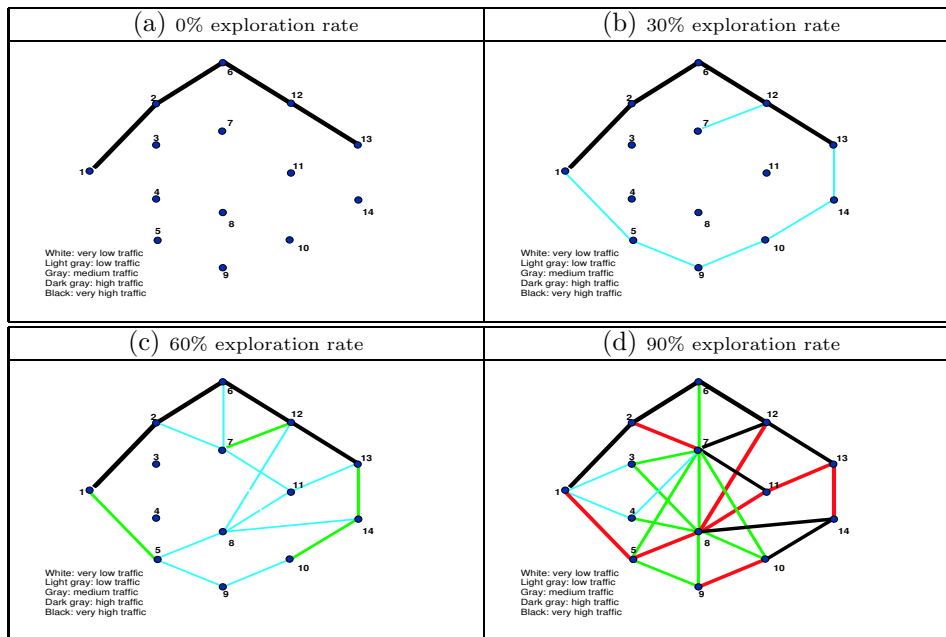


Table 6.1: Traffic through the edges for the four different exploration rates. We clearly observe that the exploration progressively increases with the exploration rate.

6.2. Second experiment

6.2.1. Description

In this second experiment, we reiterate the previous simulations on the same graph with various values of the exploration rate. The difference lies in the dynamic aspect of the environment used in this experiment. Indeed, the goal of this second experiment is to analyze the behaviour of the algorithm when a change in the environment occurs (i.e., a change in the cost of some edges).

Except the change in the environment, the context remains the same as in the first experiment, namely to convey 15,000 agents from a source node (1) to a destination node (13) by using the suggested algorithm. Thus, at each time step, an agent is sent from node 1 to reach node 13. At the beginning, the costs of the external edges (i.e., the edges on the paths $[1,2,6,12,13]$ and $[1,5,9,10,14,13]$) are initialized to 3, while the cost of all the others (internal edges) are initialized to 6. In this configuration, the shortest path from node 1 to node 13 is the path $[1,2,6,12,13]$ with a total cost of 12.

After having sent 7,500 agents (i.e., the middle of the simulation), the cost of all the internal edges was set to 1, all other things being equal. This change creates new shortest paths, all of them with a total cost of 4, passing through internal nodes $[3, 4, 7, 8, 11]$.

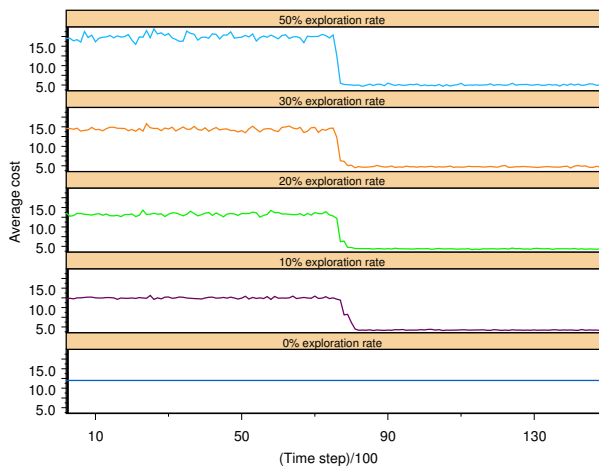


Figure 6.2: Average cost needed to reach destination node (13) from the initial node (1) in terms of time step. We observe that when no exploration is performed (exploration rate of 0%), the system misses a path that becomes (at time step 7500) much cheaper (in terms of cost) than the optimal one up to now.

6.2.2. Results

Figure 6.2 contains five graphs representing the total costs of the transfer of all agents from their source to their destination, averaged on the first 7,500 agents. We display the results for five levels of exploration rate: 0%, 10%, 20%, 30% and 50%.

The first graph (on the bottom) shows the total cost for an exploration rate of 0%. We observe that, despite the environment change (which leads to the creation of shorter paths with a total cost of 4), the cost remains equal to 12 throughout simulation. This is due to the fact that, once the algorithm has discovered the shortest path, it does not explore anymore.

The four remaining graphs show the results for exploration rate of 10%, 20%, 30% and 50%. Each of these settings is able to find the new optimal path after the change in environment –the total cost is updated from 12 to 4. This is due to the fact that the algorithm carries out exploration and exploitation in parallel. This also explains the slight rise in total cost when increasing the value of the entropy.

Figure 6.3 shows the evolution of the average cost, the maximal cost as well as the minimal cost (computed on the first 7,500 agents) in terms of the exploration rate. Increasing the entropy induces a growth in the average total costs, which is completely foreseeable since the raise in this entropy causes an enhancement in exploration, therefore producing a reduction in the exploitation and increasing the average total costs. We also notice that the difference between the minimum and the maximum total cost grows with the increase in entropy but remains quite weak.

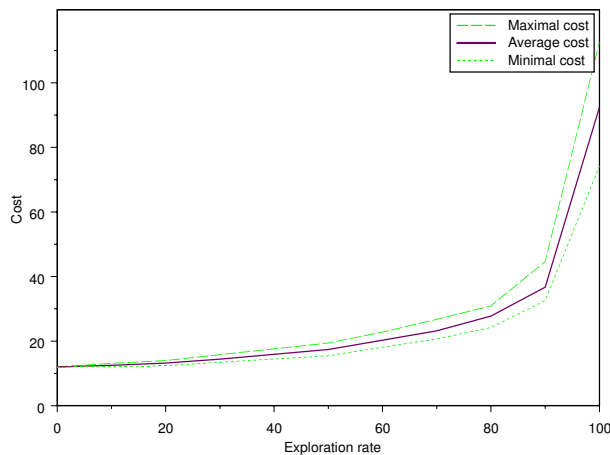


Figure 6.3: Average, maximal and minimal costs needed to reach destination node (13) from the initial node (1) in terms of exploration rate.

7. Conclusions

In this work, we presented a model integrating both exploration and exploitation in a common framework. The exploration rate is controlled by the entropy of the choice probability distribution defined on the states of the system. When no exploration is performed (zero entropy on each node), the model reduces to the common value iteration algorithm computing the minimum cost policy. On the other hand, when full exploration is performed (maximum entropy on each node), the model reduces to a “blind” exploration, without considering the costs.

The main drawback of the present approach is that it is computationally demanding since it relies on iterative procedures like Bellman’s algorithm.

Further work will aim to investigate full “stochastic shortest-path” problems, as well as alternative cost formulation, such as the “average cost per step”. Moreover, since the optimal average cost is obtained by taking the minimum among all the potential policies, it can be shown that it defines a distance measure between the states of the process. Further work will investigate the properties of this distance, which generalizes the Euclidean commute time distance between nodes of a graph, as introduced and investigated in [15], [7].

References

- [1] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: Theory and algorithms*. John Wiley and Sons, 1993.
- [2] D. P. Bertsekas. *Neuro-dynamic programming*. Athena Scientific, 1996.
- [3] D. P. Bertsekas. *Network optimization: continuous and discrete models*. Athena Scientific, 1998.

- [4] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2000.
- [5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. John Wiley and Sons, 1991.
- [6] R. Dial. A probabilistic multipath assignment model that obviates path enumeration. *Transportation Research*, 5:83–111, 1971.
- [7] F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Markov-chain computation of similarities between nodes of a graph, with application to collaborative filtering. *Submitted for publication*, <http://www.isys.ucl.ac.be/staff/francois/Articles/Saerens2005a.pdf>, 2005.
- [8] J. N. Kapur and H. K. Kesavan. *Entropy optimization principles with applications*. Academic Press, 1992.
- [9] J. G. Kemeny and J. L. Snell. *Finite Markov Chains*. Springer-Verlag, 1976.
- [10] T. M. Mitchell. *Machine learning*. McGraw-Hill Compagnies, 1997.
- [11] J. Norris. *Markov Chains*. Cambridge University Press, 1997.
- [12] M. J. Osborne. *An introduction to Game Theory*. Oxford University Press, 2004.
- [13] M. Puterman. *Markov decision processes: discrete stochastic programming*. John Wiley and Sons, 1994.
- [14] H. Raiffa. *Decision analysis*. Addison-Wesley, 1970.
- [15] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. *Proceedings of the 15th European Conference on Machine Learning (ECML 2004). Lecture Notes in Artificial Intelligence, Vol. 3201, Springer-Verlag, Berlin*, pages 371–383, 2004.
- [16] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.
- [17] H. M. Taylor and S. Karlin. *An Introduction to Stochastic Modeling, 3th Ed.* Academic Press, 1998.

APPENDIX: PROOF OF THE MAIN RESULTS

A. Appendix: Computation of the expected cost for a fixed policy

The goal is to compute the expectation of the cumulated cost over an infinite number of steps, under the policy π :

$$V_\pi(k_0) = E_\pi \left[\sum_{t=0}^{\infty} C(u(s_t)|s_t) \mid s_0 = k_0 \right], \quad (\text{A.1})$$

which thus represents the expected cost accumulated over an infinite horizon, when starting from some initial state k_0 , before reaching the destination state d . The expectation is taken on the policy, that is, on all the random variables $u(k)$ associated to the states k .

We now derive the recurrence relation for computing $V_\pi(s)$ by first-step analysis (for a rigorous proof, see, for instance, [9]). The cumulated cost, $C_\pi(k_0)$, incurred during one particular walk (one realization of the random process), starting from state k_0 and entering for the first time destination state d , is

$$C_\pi(k_0) = \sum_{t=0}^{T_{k_0}} C(u(s_t)|s_t) \quad (\text{A.2})$$

where T_{k_0} is the time (number of steps) taken to reach destination node. Now, without changing the problem, we can assume that the destination state d is an absorbing state, so that $\text{P}(s_{t+1} = k | s_t = d) = \delta(d, k)$, where δ is the delta of Kronecker – the process stops once state k has been reached. If we further assume $C(u|d) = 0$, $C_\pi(k_0)$ in Equation (A.2) can be rewritten as

$$C_\pi(k_0) = \sum_{t=0}^{\infty} C(u(s_t)|s_t)$$

since once we have reached state k , we remain in k forever and $C(d|d) = 0$. Furthermore, we immediately observe that $C_\pi(d) = 0$.

Let us now compute the quantity of interest, $E[C_\pi(k_0)|s_0 = k_0]$, for $k_0 \neq d$:

$$V_\pi(k_0) = E[C_\pi(k_0)|s_0 = k_0] = E_\pi \left[\sum_{t=0}^{\infty} C(u(s_t)|s_t) | s_0 = k_0 \right] \quad (\text{A.3})$$

$$= \sum_{i_0, i_1, \dots} \text{P}(u(s_0) = i_0, u(s_1) = i_1, \dots | s_0 = k_0) \left[\sum_{t=0}^{\infty} C(u(s_t)|s_t) \right] \quad (\text{A.4})$$

$$= \sum_{i_0, i_1, \dots} \text{P}(u(s_0) = i_0, u(s_1) = i_1, \dots | s_0 = k_0) \times \left[C(u(s_0)|s_0) + \sum_{t=1}^{\infty} C(u(s_t)|s_t) \right] \quad (\text{A.5})$$

$$= \sum_{i_0} \text{P}(u(s_0) = i_0 | s_0 = k_0) \times \left\{ \sum_{i_1, i_2, \dots} \text{P}(u(s_1) = i_1, u(s_2) = i_2, \dots | u(s_0) = i_0, s_0 = k_0) \times \left[C(u(s_0)|s_0) + \sum_{t=1}^{\infty} C(u(s_t)|s_t) \right] \right\} \quad (\text{A.6})$$

$$= \sum_{i_0} \text{P}(u(s_0) = i_0 | s_0 = k_0) \left\{ \sum_{i_1, i_2, \dots} \text{P}(u(s_1) = i_1, u(s_2) = i_2, \dots | s_1 = k_1) \times \left[C(u(s_0)|s_0) + \sum_{t=1}^{\infty} C(u(s_t)|s_t) \right] \right\} \quad (\text{A.7})$$

$$= \sum_{i_0} \mathbb{P}(u(s_0) = i_0 | s_0 = k_0) [C(u(s_0) | s_0) + V_\pi(k_1)] \quad (\text{A.8})$$

where we used the Markov property (the future behaviour only depends on the least state) and $k_1 = f(u(k_0) = i_0 | s_0 = k_0)$.

Therefore, since $V_\pi(d) = 0$, we can compute the expected cost at state k in terms of the expected cost at each state that can be reached from state k :

$$\begin{cases} V_\pi(k) = \sum_{i \in U(k)} \mathbb{P}(i|k) [C(i|k) + V_\pi(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \\ V_\pi(d) = 0 \end{cases} \quad (\text{A.9})$$

These equations are very similar to the relations allowing to compute the average first-passage times in Markov chains [11]. Their meaning is quite obvious: in order to compute the total expected cost from state k to state d , one has to go to any reachable state k'_i and proceed from there.

B. Appendix: Determination of the optimal policy

We now turn to the problem of finding the optimal policy under entropy constraints. We therefore consider an agent starting from state k_0 and trying to reach the destination state by choosing an action according to a policy $\pi = \{\mathbb{P}(u(1)|s = 1), \mathbb{P}(u(2)|s = 2), \dots\}$. We already know (2.2) that the expected cost from any state, $V_\pi(k)$, is provided by the following equations

$$\begin{cases} V_\pi(s = k) = \sum_{i \in U(k)} \mathbb{P}(u(k) = i | s = k) [C(u(k) = i | s = k) + V_\pi(s = k'_i)], \\ \text{with } k'_i = f(u(k) = i | s = k) \text{ and } k \neq d \\ V_\pi(s = d) = 0, \text{ where } d \text{ is the destination state} \end{cases} \quad (\text{B.1})$$

The goal here is to determine the policy that minimizes this expected cost, when starting from state k_0 and under entropy constraints (2.4). We therefore introduce the following Lagrange function, taking all the constraints into account,

$$L = V(k_0) + \sum_{k \neq d} \lambda_k \left(V(k) - \sum_{i \in U(k)} \mathbb{P}(i|k) [C(i|k) + V(k'_i)] \right) \quad (\text{B.2})$$

$$+ \lambda_d (V(d) - 0) + \sum_{k \neq d} \mu_k \left(\sum_{i \in U(k)} \mathbb{P}(i|k) - 1 \right) \quad (\text{B.3})$$

$$+ \sum_{k \neq d} \eta_k \left(\sum_{i \in U(k)} \mathbb{P}(i|k) \ln \mathbb{P}(i|k) + E_k \right) \quad (\text{B.4})$$

with $k'_i = f(i|k)$. Differentiating this Lagrange function in terms of the choice probabilities, $\partial L / \partial \mathbb{P}(j|l)$, and equating to zero gives

$$-\lambda_l (C(j|l) + V(k'_j)) + \mu_l + \eta_l (\ln \mathbb{P}(j|l) + 1) = 0 \quad (\text{B.5})$$

with $k'_j = f(j|l)$.

By multiplying Equation (B.5) by $P(j|l)$ and summing over $j \in U(l)$, we easily obtain

$$-\lambda_l V(l) + \mu_l + \eta_l(E_l + 1) = 0 \quad (\text{B.6})$$

from which we deduce $\mu_l = \lambda_l V(l) - \eta_l(E_l + 1)$. Replacing μ_l by its expression in (B.5) provides

$$-\lambda_l(C(j|l) + V(k'_j) - V(l)) + \eta_l(\ln P(j|l) - E_l) = 0 \quad (\text{B.7})$$

Defining $\theta_l = -\lambda_l/\eta_l$ and extracting $\ln P(j|l)$ from (B.7) gives

$$\ln P(j|l) = -\theta_l(C(j|l) + V(k'_j) - V(l)) + E_l \quad (\text{B.8})$$

or, equivalently,

$$P(j|l) = \exp[-\theta_l(C(j|l) + V(k'_j))] \exp[-\theta_l V(l) + E_l] \quad (\text{B.9})$$

Summing this last equation over $j \in U(l)$, and remembering that k'_j depends on j ($k'_j = f(j|l)$), allows us to compute the second factor in the right-hand side of Equation (B.9):

$$\exp[-\theta_l V(l) + E_l] = \left[\sum_{j \in U(l)} \exp[-\theta_l(C(j|l) + V(k'_j))] \right]^{-1} \quad (\text{B.10})$$

By replacing (B.10) in Equation (B.9), we finally obtain

$$P(j|l) = \frac{\exp[-\theta_l(C(j|l) + V(k'_j))]}{\sum_{j \in U(l)} \exp[-\theta_l(C(j|l) + V(k'_j))]} \quad (\text{B.11})$$

Finally, expressing the fact that each probability distribution has a given entropy,

$$-\sum_{i \in U(k)} P(i|l) \ln P(i|l) = E_l,$$

allows us to compute the value of θ_l in terms of E_l . The form is the optimal policy; it can be shown that it is indeed a minimum.

Notice that differentiating the Lagrange function in terms of the $V(k)$ provides dual equations allowing to compute the Lagrange multipliers.

C. Appendix: Convergence of the iterative updating rule

Before proving the convergence of the iterative updating rule (3.6)-(3.7), let us prove a preliminary lemma that will be useful later.

Indeed, we will first show that minimizing the linear form $\sum_i p_i x_i$ in terms of p_i , subject to entropy and “sum-to-one” constraints leads to the following functional form:

$$p_i = \frac{\exp[-\theta x_i]}{\sum_j \exp[-\theta x_j]} \quad (\text{C.1})$$

where θ is chosen in order to satisfy $-\sum_i p_i \ln p_i = E$ (entropy constraint). This probability distribution has exactly the same form as (3.6), in the proposed algorithm. This lemma shows that using this distribution (C.1) automatically reduces the linear form $\sum_i p_i x_i$.

To this end, let us introduce the Lagrange function

$$L = \sum_i p_i x_i + \lambda (\sum_i p_i \ln p_i + E) + \mu (\sum_i p_i - 1) \quad (\text{C.2})$$

and compute the differential of L in term of p_j , $\partial L / \partial p_j = 0$,

$$x_j + \lambda \ln p_j + \lambda + \mu = 0 \quad (\text{C.3})$$

By multiplying Equation (C.3) by p_j and summing the result, we obtain

$$\sum_j p_j (x_j + \lambda \ln p_j + \lambda + \mu) = \sum_j p_j x_j - \lambda E + \lambda + \mu = 0 \quad (\text{C.4})$$

Thus we have $(\lambda + \mu) = -\sum_j p_j x_j + \lambda E$. By substituting $(\lambda + \mu)$ by its value in (C.3), we find

$$\lambda \ln p_i = -x_i + \sum_j p_j x_j - \lambda E \quad (\text{C.5})$$

Now, by defining $\theta = \lambda^{-1}$, Equation (C.5) can be rewritten as

$$p_i = \exp[-\theta x_i] \exp[\theta \sum_j p_j x_j - E] \quad (\text{C.6})$$

Summing Equation (C.6) over i and using the fact that $\sum_i p_i = 1$ allows us to compute the value of the second factor of Equation (C.6),

$$\exp[\theta \sum_j p_j x_j - E] = \left[\sum_j \exp[-\theta x_j] \right]^{-1} \quad (\text{C.7})$$

Thus, we obtain the expected result from (C.6)-(C.7):

$$p_i = \frac{\exp[-\theta x_i]}{\sum_j \exp[-\theta x_j]} \quad (\text{C.8})$$

where θ is chosen in order to satisfy $-\sum_i p_i \ln p_i = E$ (entropy constraint). This proves our preliminary lemma.

Now, we are ready to prove the convergence of (3.6)-(3.7). We will prove this fact by induction. First, we show that if the $V(k)$ are decreasing (less than or equal to) up to the current time step, then the next update will also decrease $V(k)$. Then, since the first update necessarily decreases $V(k)$, the following updates will also decrease the value of $V(k)$. Now, since $V(k)$ is decreasing at each update and cannot be negative, this sequence must converge.

Thus, we first show that if all the $V(k)$ are decreasing up to the current time step t , then the next update will also decrease $V(k)$. Suppose the agent does visit state k at time step t , so that an update of the expected cost is computed. If we denote by $V_t(k)$ the value of the expected cost for state k and time step t , the update of $V(k)$ is given by

$$\begin{cases} P_t(u(k) = i | s = k) = \frac{\exp[-\theta_k (C(i|k) + V_t(k'_i))]}{\sum_{j \in U(k)} \exp[-\theta_k (C(j|k) + V_t(k'_j))]} \\ V_t(k) = \sum_{i \in U(k)} P_t(i|k) [C(i|k) + V_t(k'_i)], \text{ with } k'_i = f(i|k) \text{ and } k \neq d \end{cases} \quad (\text{C.9})$$

Now, denote by $\tau < t$ the time of the last visit of the agent at state k , and consequently the last update of $V(k)$ before time t . We easily find

$$V_t(k) = \sum_{i \in U(k)} P_t(i|k) [C(i|k) + V_t(k'_i)] \quad (\text{C.10})$$

$$\leq \sum_{i \in U(k)} P_\tau(i|k) [C(i|k) + V_t(k'_i)] \quad (\text{C.11})$$

$$\leq \sum_{i \in U(k)} P_\tau(i|k) [C(i|k) + V_\tau(k'_i)] = V_\tau(k) \quad (\text{C.12})$$

The passage from Equation (C.10) to (C.11) is due to the preliminary lemma, while the passage from (C.11) to (C.12) results from the assumption that all the $V(k)$ are decreasing up to the current time step t .

Now, at the beginning of the exploration process ($t = 1$), the agent starts at node k_0 and updates $V_1(k_0)$ according to

$$V_1(k) = \sum_{i \in U(k)} P_1(i|k) [C(i|k) + V_1(k'_i)] \quad (\text{C.13})$$

$$\leq \sum_{i \in U(k)} P_0(i|k) [C(i|k) + V_0(k'_i)] \quad (\text{C.14})$$

since for every node $k'_i \neq k_0$, $V_1(k'_i) = V_0(k'_i)$ (no update has yet occurred). We thus are in the conditions of the assumption: all the $V(k)$ are decreasing up to the current time step $t = 1$. Consequently, the next update will certainly decrease $V(k)$, as well as all the subsequent updates.

Thus, if we define the decrease of $V(k)$ as $\Delta V_t(k) = V_{t-1}(k) - V_t(k) \geq 0$, since $V(k)$ cannot become negative, we have $\sum_{t=1}^{\infty} \Delta V_t(k) \leq V_0(k)$ so that $\Delta V_t(k) \rightarrow 0$ as $t \rightarrow \infty$.