UCLouvain

École polytechnique de Louvain

Randomized Markov Decision Processes with multiple inputs and outputs

Randomized Shortest Path approach

Author: **Louis Henrotte**
Supervisor: **Marco Saerens**
Readers: **Jean-Charles Delvenne, Pierre Leleux**
Academic year 2022–2023
Master [120] in Mathematical Engineering

I would like to express my sincere gratitude to Pr. Marco Saerens for his guidance, his feedback, his availability and his valuable help throughout the entire process of making this master's thesis.

I also wish to express my gratitude to Pierre Leleux for his availability, his precious suggestions and his help throughout the simulation process of this work.

Lastly, I would like to extend my heartfelt gratitude to my family for their unconditional support and understanding throughout this academic journey.

This work would not have been possible without the collective contributions of these individuals, and for that, I am grateful.

Contents

Chapter 1

Introduction

1.1 General introduction

The present work aims to integrate two extensions of the randomized shortest-paths (RSP) framework studied in [20, 25, 35] which adapts the relative entropy-regularized optimal transport problem to a graph structure. In this context, a unit flow is injected into a set of input nodes and collected from a set of output nodes with specified marginals. The goal is then to minimize the expected transportation cost (exploitation) while incorporating a paths-based relative entropy regularization term (exploration), which results in a randomized, stochastic, routing policy. The degree of exploration is monitored by a temperature parameter, that brings a trade-off between a purely random and a well-defined least expected cost strategy. Exploration is especially relevant when the environment is changing and it can be used, for example, in continual exploration where we control the temperature parameter [1].

The first extension we studied investigates a different point of view on Markov decision processes (MDP), from [26, 7], reinterpreting them as a constrained RSP problems but on a bipartite graph. In this context, it explores stochastic policies for the scenario involving a single source and target. This includes defining a set of state nodes as a first set, alongside a corresponding set of action nodes forming the other set. These nodes are linked together and constraints on the action-to-state transition probabilities are added, provided by the environment, as for regular MDPs. The goal is then to find the optimal state-to-action probabilities balancing between a least-cost and a random policy. These environment constraints imply therefore some difficulties compared to the standard RSP. Paper [7] covers, in that matter, an equivalent formulation and develops a method that is closely related to the dual linear programming formulation of the MDP [33], inspired by [29]. Fixing the action-to-state probabilities is then straightforward in this new formulation.

The second extension comes from [12], and explores another approach from [20] for solving RSP problems with specified margins. It uses another equivalent extended graph to benefit from the single source and target results. This graph consists of adding one supernode injecting a flow to the inputs and another one collecting it from the outputs as if we were therefore in a single source and target context. It brings some constraints on the newly-defined edges adjusting the specified flows, which are not expected in standard RSP and other bag-of-path models [17]. The idea to solve such problems comes initially from paper [13]. It offers a way to deal with these by considering more general capacity constraints and then solves the resulting optimisation problem. It is thus possible to get the optimal policy of a

margin-constrained RSP and more generally capacity-constrained.

This paper aims therefore at applying/combining these two approaches to be able to solve RSP problems with random environment constraints as well as fixed flow margins associated to the multiple inputs and outputs. In order to do so, we established another formulation of the problem, with a newly-defined graph incorporating supernodes among the sets of state and action nodes. We then used the results from both extensions to develop a new algorithm that allows us to find the optimal stochastic policy of a larger range of optimal transport problems. In addition, the formulation used in [7, 29] and later here has the advantage of being adapted easily to other divergence regularizations. We hence cover the case of Tsallis generalized entropy regularization [40, 4, 28] instead of the usual Shannon entropy regularization (giving rise to Kullback-Leibler or KL divergence). It's interesting as it has the nice property of providing sparse policies by setting the probability of choosing inefficient actions to zero when getting closer to optimality.

The introduced algorithm is used in an transportation context and can handle several sources, targets and probability constraints from the environment. It covers a large range of problems and we will try here to see in which specific case it could be useful compared to less general methods. We mainly did so by running simulations in a "pseudo" traffic context to which randomness is added through some simplified coherent aspects. We therefore conducted several comparison experiments with other less general methods such as standard RSP methods, or the value iteration algorithm given in the classical MDP framework [8, 36] without entropy. The goal is to underline situations where the use of our algorithm would be of a good choice. For example, if the environment is random, and since we're considering it (without being able to predict it), it should allow more effective behaviors than with methods without such consideration. If the environment is also non-stationary, the system could benefit from simple continual exploration compared to standard MDPs. Or more specifically in our traffic context, the fact a randomized policy would spread the traffic over multiple paths, therefore decreasing the risk of congestion.

1.2 Related works

As we already discussed, this work is based on the ideas of 2 articles in particular [12, 7]. The first one covers a way of solving randomized shortest-path problems with multiple sources and targets and capacity constraints. [12] is a follow-up of [20] investigating the entropy-regularized optimal transport on a graph problem and computing quantities over full paths by adopting a bag-of-path formalism [17]. [18] already presented a natural generalization of this model with multiple sources and targets but following a local framework. They also proposed a source–target coupling technique offering an optimal assignment to the problem. The second introduces a way to see Markov Decision Process as a standard RSP to which are brought some constraints on action-to-state probabilities. [7] introduces a new structure and an associated algorithm, which has been adapted directly from [29]. They adopted an edge flow formalism and proposed a regularization based on Tsallis r-divergence instead of Kullback-Leibner divergence. They aimed at finding the regularized routing policies on weighted directed graphs.

More generally, the authors of [5] also proposed an algorithm for solving the multiple sources and targets entropy-regularized optimal transport problem within directed graphs. They considered storage capacity on the nodes while [12] only took into account constraints on the edges. They did so by using a designed procedure based Dykstra's iterative projection algorithm where they obtain a policy formed of edge flows (instead of a Markov chain in our case). Moreover, the extension of the RSP problem on a bipartite graph was investigated and extended in the machine learning field by [14], who showed that using Sinkhorn's matrix scaling algorithm is more efficient than standard linear programming solvers.

Concerning entropy-regularized MDPs, numerous papers were born building upon the primary works of [39, 23]. Many algorithms were then developed, including additional regularization terms based on Shannon entropy (Kullback-Leibler divergence) to the transition rewards or costs. (see articles such as [11], [32] or [38, 37] for examples and references). In addition, instead of using the Kullback-Leibler divergence for computing randomized policies, some recent papers cover the extension to the Tsallis r-entropy or divergence. As discussed in [29], it is well-known for its sparse properties when approaching the optimum, which can be explained by its close link to L1/L2 regularization [21]. Interestingly, the authors of [27] used this entropy regularization (for $r = 2$) in MDPs and managed to show that solving the problem and get the policy can be done via a sparse value iteration algorithm.

1.3 Contributions and organization of the paper

The main contributions of this work are :

- The applications of the RSP formalism to MDP problems is broadened by considering the case of multiple inputs and ouputs.
- The introduction of a new algorithm is proposed in order to compute optimal randomized policies for this extended case both for Kullback-Leibler divergence and Tsallis r-divergence.
- An experimental comparison between our new algorithm and other methods from the RSP context through several traffic simulations on a local road network of Louvain-la-Neuve. It underlines its benefits and drawbacks.
- A small study on the effect of using an adaptive learning rate method (ADAM) on our gradient descent algorithm.

The next chapter presents the notation and the needed background around the two extensions we are studying. It includes another MDP formulation using the RSP framework and an algorithm used in the extended case of multiple sources and targets. Chapter 3 presents the theoretical structure and approach used in the elaboration of our new algorithm. It follows with the simulation part, chapter 4, which introduces a simple illustrative randomized board game example. It then covers how our algorithm can be used to solve more complex comparison considering the pseudo-traffic example. It includes also a study related to the efficiency of the algorithm. Chapter 5 finishes with the conclusions of the paper.

Chapter 2

Background

2.1 The standard Randomized Shortest-Paths (RSP) formalism

2.1.1 Notation

This section starts with a review of the randomized shortest-path relevant notations, based on full paths, together with the description of the associated problem. This section is issued from [20], inspired by the transportation science models presented in [2, 3, 15]. For reference, [19, 18] also show another derivation but based on local edge flows.

We assume a weighted, strongly connected graph $G = (\mathcal{V}, \mathcal{E})$, with a set of n nodes $\mathcal{V} = \{1, 2, ..., n\}$ and a set of m edges $\mathcal{E} = (i, j)$. On these edges are defined non-negative weights, noted a_{ij}, representing local connectivity between the different nodes and constituting the adjacency matrix $\mathbf{A}$. We consider that in the case there is no affinity between 2 nodes, the value is zero. From this matrix, we can naturally define a reference transition probabilities matrix denoted as $\mathbf{P}_{\text{ref}}$ associated with the random walk behaviour on our graph G. Element-wise, we have:

$$[\mathbf{P}]_{ij} = p_{ij} = \frac{a_{ij}}{\sum_{j' \in \mathcal{S}ucc(i)} a_{ij'}} = \frac{a_{ij}}{\sum_{j'=1}^{n} a_{ij'}} \tag{2.1}$$

where $\mathcal{S}ucc(i)$ is the set of nodes achievable from node i. Moreover, we also assume non-negative costs c_{ij} associated with each edge (i, j), translating the drawback of going through this edge. Both sets of parameters a_{ij} or c_{ij} can be independently defined from the other, or following some relationship as $c_{ij} = 1/a_{ij}$. These elements form in turn the cost matrix $\mathbf{C}$. Note that when there is no edge between 2 nodes, we consider its cost to be infinite.

2.1.2 Problem statement

We can continue with a short description of the associated problem. We discuss here the case of a graph G with a single source and target since we will cover the more general case later in Section 2.2. Otherwise, [20] contains more details about the margin-constrained RSP.

The standard RSP is therefore a problem of finding the best policy on a graph from a source node to a target node, between a deterministic shortest-path and a random walk behavior. More intuitively, the final policy guides the random walker to the target state following some biased trade-off between total exploration and an optimal least-cost path.

As we said earlier, this derivation is based on full paths. The set of trajectories going from the input node 1 to the output node n, can be denoted in this matter as $\mathcal{P}_{1n}$. Which can be considered as a bag-of-path [17] with each particular path $\wp \in \mathcal{P}_{1n}$ having the same extremes 1 and n. The costs, as introduced below, are computed as the sum of local costs c_{ij} along the edges of a particular path. The total cost is denoted as $\tilde{c}(\wp)$ in order to have a notation different from the same local quantity. Following the same reasoning, the random walk behavior, translated by Eq. (2.1.1), is adapted as the product of the reference probabilities along the path, expressed as $\tilde{\pi}(\wp)$.

The goal is now to find the optimal distribution P over the set of paths $\mathcal{P}_{1n}$. We do so by minimizing the relative free energy potential [22]:

$$\left| \begin{array}{ll} \underset{\{\mathrm{P}(\wp)\}_{\wp\in\mathcal{P}_{1n}}}{\text{minimize}} & \phi(\mathrm{P}) = \sum_{\wp\in\mathcal{P}_{1n}} \mathrm{P}(\wp)\tilde{c}(\wp) + T \sum_{\wp\in\mathcal{P}_{1n}} \mathrm{P}(\wp) \log\left(\dfrac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) \\ \text{subject to} & \sum_{\wp\in\mathcal{P}_{1n}} \mathrm{P}(\wp) = 1 \end{array} \right. \tag{2.2}$$

The first term corresponds to the expected transport cost of going from node 1 to node n, which is the exploitation part. The second term is the relative entropy [31, 38], also called the Kullback–Leibler divergence[1] between the path probability distribution we are looking for and the pure random walk strategy. That is, the exploration part. We also have the temperature parameter T, adjusting the contributions of each terms, from which we can define $\theta = 1/T$ the inverse temperature parameter. This parameter T thus monitors the amount of randomness of the policy by interpolating between an optimal and a purely random behavior.

It can be found that the solution of Problem (2.2) leads to a Gibbs–Boltzmann distribution on the set of paths (more details in paper such as [22, 25, 17]):

$$\mathrm{P}^*(\wp) = \frac{\tilde{\pi}(\wp)\exp[-\theta\tilde{c}(\wp)]}{\sum_{\wp'\in\mathcal{P}_{1n}} \tilde{\pi}(\wp')\exp[-\theta\tilde{c}(\wp')]} \tag{2.3}$$

which is the optimal randomized policy that provides the probability of choosing any path $\wp$ in the set of all paths from the input node 1 to the output node n. Following [35, 29], it can also further be shown that this path-level probability distribution is equivalent to the one generated by a Markov chain with some biased transition probabilities p_{ij} at the local, edge, level. Biased, meaning that the random walker would be more attracted to the target node as the temperature T is decreasing, favouring shorter paths.

Note that this property will be useful later when considering the Markov Decision Processes.

[1] We should normally add non-negativity constraints on the policy but this is not necessary as it's automatically non-negative when using KL divergence regularization.

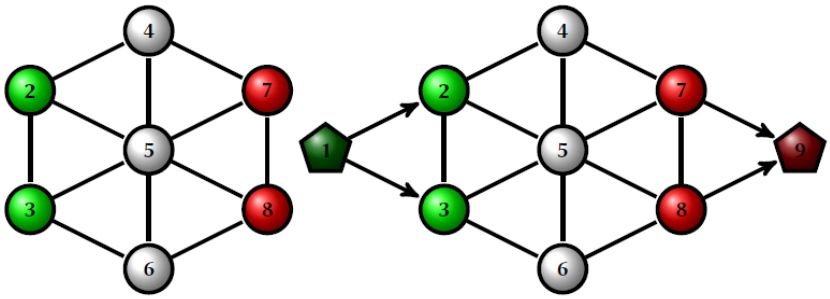

Figure 2.1: The image on the left shows an example of a graph G with two input nodes numbered as $\mathcal{I}n = 2, 3$ in green, and two output nodes, $\mathcal{O}ut = 7, 8$ in red. On the right, we have the extended graph G_{ext}, where we added 1 source supernode 1 in dark green, connected to all the input nodes, and one target supernode $n_{\text{ext}} = 9$ in dark red, connected to all the output nodes. Figures are taken from [12].

2.2 RSP with multiple inputs and outputs

Given this Randomized Shortest-Paths context, we now describe a way of solving this type of problems with more general flow constraints. That means, finding the best policy (minimizing the expected cost) for carrying the input flow from multiple source nodes to multiple target nodes, given a level of exploration given by the entropy regularization term. We give here an alternative method that allows to compute it by defining an extended graph G_{ext} with two new supernodes attached to G (see Figure 2.1). It is further developed in paper [12].

2.2.1 Notation

Before doing so, we remind of the notations used in the context of RSP with constraints on the margins, used already in [20]. Let's say we have set of inputs $\mathcal{I}n$ as well as a set of outputs $\mathcal{O}ut$. We consider having a unit flow so that we define two vectors $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$ of size n, the first one containing the proportions of that flow coming from each source nodes whereas the other contains the proportions, this time, delivered to each target nodes. So when we have $i \notin \mathcal{I}n$, we set $\sigma_i^{\text{in}} = 0$. The same logic applies for the outputs. We therefore have mathematically:

$$\begin{cases} \sum_{i \in \mathcal{I}n} \sigma_i^{\text{in}} = 1 & \text{with all } \sigma_i^{\text{in}} \geq 0 \\ \sum_{j \in \mathcal{O}ut} \sigma_j^{\text{out}} = 1 & \text{with all } \sigma_j^{\text{out}} \geq 0 \end{cases} \tag{2.4}$$

Note that in the case of a non-unitary flow, we can just find the solution for our unit flow and then multiply the results accordingly.

2.2.2 The extended graph

We now explain how to get to a new equivalent single-input and single-output graph $G_{\text{ext}} = \{\mathcal{V}_{\text{ext}}, \mathcal{E}_{\text{ext}}\}$ from our initial graph G. We start by introducing an initial

super-node acting as a single source, coupled with an additional super-node acting as a single absorbing target (with no outgoing edges). We therefore add two new nodes such as $n_{\text{ext}} = |\mathcal{V}_{\text{ext}}| = n + 2$. In order to make these super-nodes part of the original graph, we simply have to build new directed edges to connect them to the initial inputs and outputs. Meaning it starts from the new single source and it arrives at the new single target. Furthermore, these edges will be assigned a zero cost, given the absence of any associated transportation penalties. Otherwise, the rest of the graph G remains almost the same. Only the numbering of nodes will change since it now starts with the first super-node 1 and finish with our target super-node n_{ext}. This new graph is equivalent to G since the source node provides a unit flow to the original inputs while the target node is made absorbing (no possibility to go further), and thus receives the same unit flow from the original outputs. Above you can find Fig. 2.1 showing an example of that extension.

2.2.3 The reference transition probabilities

After defining how the new nodes are linked to the initial graph, the next step is to determine the weights associated to the new edges connecting them to the initial graph.

First, we recall some properties that will help us to do that. As we already know from Sect. 2.1, the transition reference probability matrix is associated with the natural random walk on our graph G_{ext} in the standard RSP formalism. It is defined in a local way as an absorbing Markov chain for which we can extract some quantities. This include the expected number of visits (i.e., the flow) through node $i \in \mathcal{V}_{\text{ext}}$, denoted as $\bar{n}_i$, and the expected number of passages through edge $(i, j) \in \mathcal{E}_{\text{ext}}$, $\bar{n}_{ij}$. And by conservation of flows, these quantities are related together in this way: $\bar{n}_j = \sum_{i \in \mathcal{P}red(j)} \bar{n}_{ij} + \delta_{1j}$ and $\bar{n}_{ij} = \bar{n}_i p^{\text{ext}}_{ij}$ for any $j \in \mathcal{V}$. δ_{1j} translates the fact a unit flow is generated at node 1.

From the previous properties, we can define the weights in such a way that the flow constraints are properly met, as in Eq. (2.4). Starting with the directed edges going out of node 1, we have that $\bar{n}_1 = \delta_{11} = 1$. Having $\bar{n}_{1i} = \bar{n}_1 p^{\text{ext}}_{1i} = \sigma^{\text{in}}_i$ by conservation of flows, we straightly see that we need $p^{\text{ext}}_{1i} = \sigma^{\text{in}}_i$ to ensure the exact equality. From Eq. (2.1.1), we find that a valid choice for the weights is to assign them to their associated probabilities.

However, for the edges incident to node n, the weights are harder to find. Indeed, using again the conservation of flows, we have: $\bar{n}_{jn} = \bar{n}_j p^{\text{ext}}_{jn} = \sigma^{\text{out}}_j$ for any $j \in \mathcal{O}ut$. Since we don't know the specific value of the flow passing through each output nodes, we would need to compute it in order to find the probabilities. We therefore define a new weight vector, denoted as $\mathbf{w}$, whose values are chosen according to the constraint on the output flows equal to $\boldsymbol{\sigma}^{\text{out}}$.

We can now construct the adjacency matrix of the extended graph $\mathbf{A}_{\text{ext}}$. After renumbering the nodes as discussed above, we have:

$$\mathbf{A}_{\text{ext}} = \begin{array}{c} \\ 1 \\ \{2,\dots,(n-1)\}=\mathcal{V} \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 1 & \{2,\dots,(n-1)\}=\mathcal{V} & n \end{array} \\ \begin{bmatrix} 0 & \boldsymbol{\sigma}^{\text{T}}_{\text{in}} & 0 \\ \mathbf{0} & \mathbf{A} & \mathbf{w} \\ 0 & \mathbf{0}^{\text{T}} & 0 \end{bmatrix} \end{array} \tag{2.5}$$

Regarding the new costs, as previously mentioned, they are all zero. The result is therefore straightforward:

$$\mathbf{C}_{\text{ext}} = \begin{array}{c} \\ 1 \\ \{2,\dots,(n-1)\}=\mathcal{V} \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 1 & \{2,\dots,(n-1)\}=\mathcal{V} & n \end{array} \\ \begin{bmatrix} 0 & \mathbf{0}^{\mathrm{T}} & 0 \\ \mathbf{0} & \mathbf{C} & \mathbf{0} \\ 0 & \mathbf{0}^{\mathrm{T}} & 0 \end{bmatrix} \end{array} \tag{2.6}$$

The probabilities resulting from the random walk are then found using Eq. (2.1.1) from our new adjacency matrix $\mathbf{A}_{\text{ext}}$. Although, we still need to figure out a way to compute the vector $\mathbf{w}$ in order for our target flow margins $\boldsymbol{\sigma}^{\text{out}}$ to be satisfied. Article [12] covers these computations in its Appendix A, with an approach based on "killing nodes", inspired by [20]. The extended transition probability matrix leading to the proper flows is then computed from a quantity denoted as $\boldsymbol{\alpha}$:

$$\mathbf{P}_{\text{ext}} = \begin{array}{c} \\ 1 \\ \{2,\dots,(n-1)\}=\mathcal{V} \\ n \end{array} \begin{array}{c} \begin{array}{ccc} 1 & \{2,\dots,(n-1)\}=\mathcal{V} & n \end{array} \\ \begin{bmatrix} 0 & \boldsymbol{\sigma}_{\text{in}}^{\mathrm{T}} & 0 \\ \mathbf{0} & (\mathbf{I}-\mathbf{Diag}(\boldsymbol{\alpha}))\mathbf{P} & \boldsymbol{\alpha} \\ 0 & \mathbf{0}^{\mathrm{T}} & 0 \end{bmatrix} \end{array} \tag{2.7}$$

The paper also underlines the fact w_j can only be positive for target nodes $j \in \mathcal{O}ut$. For all other nodes, the weight is set to 0. The same result applies for α_j, as both quantities are directly proportional. The probabilities hence remain the same as for the original graph, as long as the random walker is not absorbed in one of the output nodes. More details around the computations of $\boldsymbol{\alpha}$ are given in Eqs. (A7)-(A9) of [12].

2.2.4 Problem statement

The goal is now to solve the randomized shortest-paths problem associated to the extended graph G_{ext} while satisfying the flow constraints. We already talked about the RSP model, in its basic form, with the single source/target in Section 2.1. The more general problem with multiple inputs and outputs, taken from [20] as well, can be written as:

$$\left|\begin{array}{lll} \underset{\{\mathrm{P}(\wp)\}_{\wp\in\mathcal{P}}}{\text{minimize}} & \phi(\mathrm{P}) = \sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\tilde{c}(\wp) + T\sum_{\wp\in\mathcal{P}} \mathrm{P}(\wp)\log\left(\dfrac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right) & \\ \text{subject to} & \sum_{j\in\mathcal{V}}\sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_i^{\text{in}} & \forall i \in \mathcal{I}n \\ & \sum_{i\in\mathcal{V}}\sum_{\wp_{ij}\in\mathcal{P}_{ij}} \mathrm{P}(\wp_{ij}) = \sigma_j^{\text{out}} & \forall j \in \mathcal{O}ut \end{array}\right. \tag{2.8}$$

where, following a bag-of-paths approach, the set of interest is extended to $\mathcal{P} = \cup_{i\in\mathcal{I}n}\cup_{j\in\mathcal{O}ut}\mathcal{P}_{ij}$.

2.2.5 The approach based on capacitity constraints

We now want to consider the new graph structure to compute the solution, which allows the use of the nice properties of single input and output RSPs. The following computations are hence derived from the results obtained in [12], inspired by [13]. It

exploits the Lagrange formulation of of Eq. 2.8 (more details in Appendix C of [12]) and provides the optimal randomized policy satisfying:

$$\begin{cases} \bar{n}_{1i} = \sigma_i^{\text{in}} & \text{for each node } i \in \mathcal{I}n \\ \bar{n}_{jn} = \sigma_j^{\text{out}} & \text{for each node } j \in \mathcal{O}ut \end{cases} \tag{2.9}$$

where $\bar{n}_{ij}$ is the flow (expected number of passages) in edge (i, j) and where σ_i satisfies Eq. (2.4). Note that the constraints recalled here are now expressed according to our extended graph G_{ext}.

In short, we can then introduce the Lagrange function associated to the previous equation. We use the same notations as for Sect. 2.1, $\mathcal{P}_{1n}$ represents therefore the set of possible paths from the input supernode 1 to the output supernode n,

$$\mathscr{L}(\mathrm{P}, \boldsymbol{\lambda}) = \underbrace{\sum_{\wp \in \mathcal{P}_{1n}} \mathrm{P}(\wp)\tilde{c}(\wp) + T \sum_{\wp \in \mathcal{P}_{1n}} \mathrm{P}(\wp) \log\left(\frac{\mathrm{P}(\wp)}{\tilde{\pi}(\wp)}\right)}_{\text{free energy, } \phi(\mathrm{P})}$$
$$+ \mu\left(\sum_{\wp \in \mathcal{P}_{1n}} \mathrm{P}(\wp) - 1\right) + \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}}\left(\bar{n}_{1i} - \sigma_i^{\text{in}}\right) + \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}}\left(\bar{n}_{jn} - \sigma_j^{\text{out}}\right) \tag{2.10}$$

where $\boldsymbol{\lambda}$ contains the Lagrange parameters $\{\lambda_i^{\text{in}}\}$ and $\{\lambda_j^{\text{out}}\}$ corresponding to each input and output constraints respectively.

In paper [13], it is shown that the dual function and gradient are easy to compute. Moreover, as the objective function is strictly convex, the support set for the path probabilities is convex and the equality constraints are linear, there exist only one global minimum with no duality gap [16]. Having this in mind, they used a common optimization procedure, which iterates between solving the primal (finding the optimal probability distribution) while considering the Lagrange parameters as fixed, and then maximizing the dual (which is always concave) with respect to the Lagrange parameters, until convergence (see Arrow–Hurwicz–Uzawa algorithm [6]).

In our case, the dual function can be written as:

$$\mathscr{L}(\mathrm{P}^*, \boldsymbol{\lambda}) = -T \log \mathcal{Z}' - \sum_{i \in \mathcal{I}n} \lambda_i^{\text{in}} \sigma_i^{\text{in}} - \sum_{j \in \mathcal{O}ut} \lambda_j^{\text{out}} \sigma_j^{\text{out}} \tag{2.11}$$

where the partition function, denoted as $\mathcal{Z}'$, is given from the augmented costs c'_{ij} on G_{ext}, which are updated from the Lagrange parameters. $\mathcal{Z}'$ is obtained through the expression $\sum_{\wp \in \mathcal{P}_{1n}} \tilde{\pi}(\wp) \exp[-\theta \tilde{c}'(\wp)]$ where the augmented costs are computed as:

$$c'_{ij} = \begin{cases} c_{ij}^{\text{ext}} + \lambda_j^{\text{in}} \\ c_{ij}^{\text{ext}} + \lambda_i^{\text{out}} \\ c_{ij}^{\text{ext}} \end{cases} = \begin{cases} \lambda_j^{\text{in}} & \text{when } i = 1 \text{ and } j \in \mathcal{I}n \\ \lambda_i^{\text{out}} & \text{when } i \in \mathcal{O}ut \text{ and } j = n \\ c_{ij}^{\text{ext}} & \text{otherwise} \end{cases} \tag{2.12}$$

The Lagrange parameters updates are in turn determined, following the maximization

of the dual function, according to these expressions:

$$\begin{cases} \lambda_k^{\text{in}} = \frac{1}{\theta}\left(\log z'_{kn} - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log z'_{ln} \right) \text{ for each } k \in \mathcal{I}n \\ \lambda_l^{\text{out}} = \frac{1}{\theta}\left(\log z'_{1l} - \log\left(\frac{\sigma_l^{\text{out}}}{p_{ln}^{\text{ext}}} \right) \right. \\ \qquad \left. - \sum_{k \in \mathcal{O}ut} \sigma_k^{\text{out}} \left[\log z'_{1k} - \log\left(\frac{\sigma_k^{\text{out}}}{p_{kn}^{\text{ext}}} \right) \right] \right) \text{ for each } l \in \mathcal{O}ut \end{cases} \tag{2.13}$$

where z'_{kn} are the entries of the fundamental matrix $\mathbf{Z}$:

$$\mathbf{Z} = \mathbf{I} + \mathbf{W} + \mathbf{W}^2 + \cdots = (\mathbf{I} - \mathbf{W})^{-1} \tag{2.14}$$

where $\mathbf{W}$ is obtained as the elementwise product $\mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$. The idea behind the resulting algorithm is therefore to compute the necessary elements of the fundamental matrix from the current augmented costs as in Eq. (2.14). Then we compute the augmented cost following Eq. (2.12), Eq. (2.13) and iterates both of these steps until convergence of the parameters. The optimal policy (transition probabilities) is found, at the end, from the augmented costs (see Eq. B19 of [12]).

Note that the Lagrange parameters are undefined up to a translation. Here, they have been shifted to get $\sum_{k \in \mathcal{I}n} \lambda_k^{\text{in}} \sigma_k^{\text{in}} = 0$ and $\sum_{l \in \mathcal{O}ut} \lambda_l^{\text{out}} \sigma_l^{\text{out}} = 0$, but it's not the only way to do so.

From these new expressions, one thing is important to notice. We indeed see that our initial problem of margin-constrained randomized shortest-paths can be viewed as a classical unconstrained RSP on a newly defined extended graph where edges have been added and associated costs have been updated (augmented costs) in order to satisfy the initial constraints. This interpretation is detailed initially in Eqs. (21)-(23) of [13], where the Lagrange parameters can be seen as additional "virtual" costs.

2.2.6 The resulting algorithm

The resulting algorithm is presented below and has been taken from [12].

Algorithm 1 Solving the relative entropy-regularized optimal transport on a graph problem with multiple sources and targets, called the margin-constrained bag-of-paths model.

Input:

– A weighted directed graph G_{ext} containing n nodes, derived from a strongly connected graph G as detailed this Section. Node 1 is the source supernode and node n the absorbing, target, supernode. The indegree of node 1 and the outdegree of node n are both equal to 0.

– The set of input nodes $\mathcal{I}n$ (only connected to the source supernode 1) and output nodes $\mathcal{O}ut$ (only connected to the target supernode n).

– The $n \times n$ transition matrix $\mathbf{P}_{\text{ext}}$ associated with G_{ext}. See Eq. (2.7) and Algorithm 1 in [20] for the computation of $\mathbf{P}_{\text{ext}}$ and $\boldsymbol{\alpha}$.

– The $n \times n$ non-negative cost matrix $\mathbf{C}_{\text{ext}}$ associated with G_{ext} (see Eq. (2.6)). These original costs are equal to zero for edges starting in node 1 and ending in $\mathcal{I}n$ as well as edges starting in $\mathcal{O}ut$ and ending in node n.

– The $n \times 1$ vectors of input flows, $\boldsymbol{\sigma}_{\text{in}}$, and output flows, $\boldsymbol{\sigma}_{\text{out}}$, both non-negative and summing to 1.

– The inverse temperature parameter θ.

Output:

– The $n \times n$ randomized policy defined by the transition matrix $\mathbf{P}^*$.

$\boldsymbol{\lambda}_{\text{in}} \leftarrow \mathbf{0}$; $\boldsymbol{\lambda}_{\text{out}} \leftarrow \mathbf{0}$ ▷ initialize $n \times 1$ Lagrange parameter vectors

$\mathbf{C}' \leftarrow \mathbf{C}_{\text{ext}}$ ▷ initialize the augmented costs matrix

repeat[main iteration loop]

  $\mathbf{W}' \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ ▷ compute $\mathbf{W}'$ matrix (elementwise exponential and multiplication $\circ$)

  Solve $(\mathbf{I} - \mathbf{W}')\mathbf{z}'_n = \mathbf{e}_n$ ▷ backward variables (column n of the fundamental matrix $\mathbf{Z}'$) with elements z'_{kn} (n is fixed)

  for all $k \in \mathcal{I}n$ **do** ▷ initialize Lagrange parameters associated with source nodes

    $\lambda_k^{\text{in}} \leftarrow \frac{1}{\theta} \log z'_{kn}$ ▷ compute Lagrange parameters

  end for

  for all $k \in \mathcal{I}n$ **do** ▷ update Lagrange parameters and costs associated with source nodes

    $\lambda_k^{\text{in}} \leftarrow \lambda_k^{\text{in}} - \sum_{k' \in \mathcal{I}n} \sigma_{k'}^{\text{in}} \lambda_{k'}^{\text{in}}$ ▷ normalize Lagrange parameters

    $c'_{1k} \leftarrow \lambda_k^{\text{in}}$ ▷ update augmented costs (recall that $c_{1k}^{\text{ext}} = 0$ for all $k \in \mathcal{I}n$))

  end for

  $\mathbf{W}' \leftarrow \mathbf{P}_{\text{ext}} \circ \exp[-\theta \mathbf{C}']$ ▷ update $\mathbf{W}'$ matrix

  Solve $\left(\mathbf{I} - \mathbf{W}'^{\mathrm{T}}\right)\mathbf{z}'_1 = \mathbf{e}_1$ ▷ forward variables (row 1 of the fundamental matrix $\mathbf{Z}'$) with elements z'_{1k} (1 is fixed)

  for all $l \in \mathcal{O}ut$ **do** ▷ initialize Lagrange parameters associated with target nodes

    $\lambda_l^{\text{out}} \leftarrow \frac{1}{\theta} \log z'_{1l} - \frac{1}{\theta} \log \left(\frac{\sigma_l^{\text{out}}}{p_{ln}^{\text{ext}}} \right)$ ▷ compute Lagrange parameters

  end for

  for all $l \in \mathcal{O}ut$ **do** ▷ update Lagrange parameters and costs associated with target nodes

    $\lambda_l^{\text{out}} \leftarrow \lambda_l^{\text{out}} - \sum_{l' \in \mathcal{O}ut} \sigma_{l'}^{\text{out}} \lambda_{l'}^{\text{out}}$ ▷ normalize Lagrange parameters

    $c'_{ln} \leftarrow \lambda_l^{\text{out}}$ ▷ update augmented costs (recall that $c_{ln}^{\text{ext}} = 0$ for all $l \in \mathcal{O}ut$)

  end for

until convergence of $\boldsymbol{\lambda}_{\text{in}}$, $\boldsymbol{\lambda}_{\text{out}}$

$\mathbf{P}^* \leftarrow (\mathbf{Diag}(\mathbf{z}'_n))^{-1} \mathbf{W}' \mathbf{Diag}(\mathbf{z}'_n)$ ▷ compute optimal policy

return $\mathbf{P}^*$

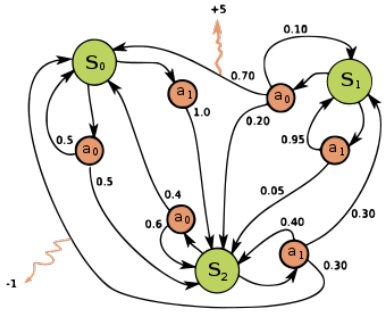

Figure 2.2: Example of classical MDP with states S_i, their associated control actions a_j and some defined rewards (or costs in our context). Taken from [41].

2.3 Markov Decision Processes (MDP)

Before talking about our interest for the MDP model, we can start by recalling its definition as well as its link with Markov chains [33].

A Markov chain is a mathematical model that describes a sequence of events in which the probability of each event depends only on the state at the previous event. The key feature of a Markov chain is the Markov property, which states that the future state of the system depends only on the present state and not on any previous states.

Markov decision processes are an extension of Markov chains that allow decision-making in random environments. The system moves from one state to another based on transition probabilities, just like in a Markov chain. However, the difference is that the system also includes taking decisions among multiple actions, based on the current state of the system. The chosen action affects the probability distribution of following transitions, as well as the received reward (or cost).

Formally, an MDP is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function that specifies the probability of moving from one state k to another k' given an action a $P(s_{t+1} = k'|s_t = k, u(s_t) = a) = p(k'|k, a)$, a reward function that specifies the reward received for each of these transitions $r(k'|k, a)$, and sometimes a discount factor γ that determines the relative importance of future rewards when considering an action. The goal is hence to find the set of actions that maximizes the expected cumulative reward (or minimizes the expected cumulative cost) until a target, goal state is reached. This optimal set defines a deterministic state-to-action policy while the action-to-state probabilities are provided by the environment which can be random or not.

Solutions for finite MDPs can be found using methods of dynamic programming. One popular method is the Value Iteration:

$$\begin{cases} \hat{V}(k) & = \min_{a \in U(k)} \left(c(a|k) + \sum_{k'=1}^{\infty} p(k'|k, a) \gamma \hat{V}(k') \right), \quad k \neq d \\ \hat{V}(d) & = 0, \quad \text{where d is the destination node} \end{cases} \tag{2.15}$$

where $\hat{V}(k)$ contains the partial discounted expected rewards to be earned by following the optimal action when starting at state k.

The link between MDPs and Markov chains is that an MDP can be considered as a Markov chain with decision nodes. At each node, the system move to a new state based on the chosen action, but the probability distribution of the next state depends only on the current state and the chosen action, and not on any previous states or actions. This satisfies the Markov property as well, which allows us to use the same properties for analyzing MDPs as we do for analyzing Markov chains.

We can already see some similarities between this model and the standard RSP problem of Sect. 2.1 we described initially and whose solution provides a biased Markov Chain as recalled just below. The next section is about how to link these two frameworks and the resulting reinterpretation of MDPs.

2.3.1 MDP as a Markov chain on a bipartite graph

The goal is thus to redefine Markov Decision Processes in a Randomized Shortest-Path formulation where we aim at finding a stochastic policy according to some level of entropy. That means, finding the least-expected-cost policy subject to a KL regularization term, monitoring the degree of randomness between the optimal deterministic solution and the purely random behavior.

In this context, it corresponds to finding the optimal transition probabilities of a Markov chain defined on the states but also on the actions. It means minimizing the expected cost needed to reach the single goal state from the single initial state, while having a fixed level of entropy and transition probabilities from actions to states. This is in line with the local interpretation of the optimal randomized policy we discussed with Eq. (2.3) from Sect. 2.1. We will see more precisely how it is linked later in this section.

[7] cover the description of the entire model we are recalling here. It starts with the definition of a new bipartite graph with state and action nodes. It describes how the reference transition probabilities are modified and then states the associated optimization problem. It then finishes by establishing a method (inspired by [29]) based on Lagrange parameters to compute the optimal stochastic policy.

2.3.2 The bipartite graph

MDPs can be modelled as a directed bipartite graph G_b (see Figure 2.3). The first set of nodes is the original set of states $\mathcal{S}$, whereas the other nodes are the possible actions associated to each of these states. We denoted as $n_{\mathcal{S}}$ and $n_{\mathcal{A}}$ the total number of states and actions. For the states, it is straightforward $n_{\mathcal{S}} = |\mathcal{S}|$. For the actions, we need to consider every possible choice from each state, so we define $\mathcal{A}(k)$ as the set of actions available in state k. $\mathcal{A}$ corresponds therefore to the union of all theses sets over the states $\mathcal{S}$. Note that, even if the same action is achievable from different states, we must have one distinct action corresponding to each distinct state. We therefore have $n_{\mathcal{A}} = |\mathcal{A}| = |\mathcal{A}(1)| + |\mathcal{A}(2)| + \cdots + |\mathcal{A}(n_{\mathcal{S}} - 1)|$ where $|\mathcal{A}(n_{\mathcal{S}})| = 0$ since the last state $n_{\mathcal{S}}$ is the absorbing target state with no further actions. About the numbering, we assumed here that the state nodes are indexed from 1 to $n_{\mathcal{S}}$ and the actions are hence starting from $n_{\mathcal{S}} + 1$ to $n_{\mathcal{S}} + n_{\mathcal{A}}$.

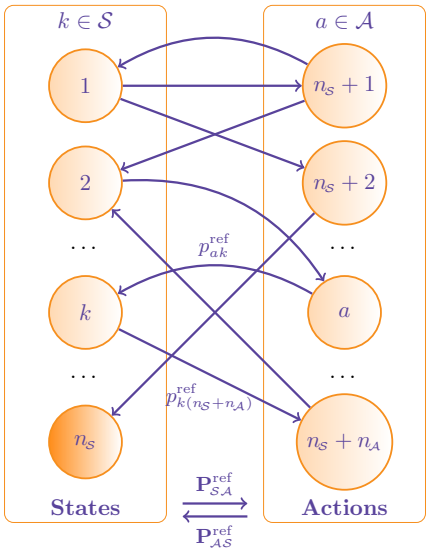

Figure 2.3: Standard MDP modeled as a bipartite graph G_b with $n_\mathcal{S}$ states and $n_\mathcal{A}$ control actions. We indeed have node 1 as initial state and node $n_\mathcal{S}$ as absorbing target state. Given also the reference probabilities p^{ref}_{ka} and p^{ref}_{ak} gathered in matrices. The first one being the policy and the second one being provided by the environment. The graph is from [26].

2.3.3 The reference transition probabilities

We now describe how the reference transition probabilities are modified according to this new bipartite graph G_b. We consider the case of a natural random walk as for standard RSPs when there is no prior information (see Eq. 2.1.1). The actions are thus chosen with a complete uncertainty, translated through a uniform distribution. We define the probabilities as $p^{\text{ref}}_{ka} = 1/|\mathcal{A}(k)|$ for each state $k \in \mathcal{S}$. Walkers in state k jump to some action a with p^{ref}_{ka} to which a non-negative cost is associated, denoted as c_{ka} as for standard MDPs. The state-to-action reference transition probability matrix $\mathbf{P}^{\text{ref}}_{\mathcal{AS}}$ and cost matrix $\mathbf{C}_{\mathcal{SA}}$ are therefore built from these expressions. Note that the last row of both matrices is set to 0 since the goal state has no successors.

Concerning the action-to-state probabilities, we assume the agents are moving from an action a to a state k with a probability fixed by the environment. As for standard MDPs, these transitions cannot be modified and there is no additional cost associated. We can now express the matrices related to our bipartite graph G_b:

$$\mathbf{P}^{\text{ref}} = \begin{array}{c} \\ \mathcal{S} \\ \mathcal{A} \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{S} & \mathcal{A} \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{P}^{\text{ref}}_{\mathcal{SA}} \\ \mathbf{P}^{\text{ref}}_{\mathcal{AS}} & \mathbf{0} \end{bmatrix} \end{array}, \quad \mathbf{C} = \begin{array}{c} \\ \mathcal{S} \\ \mathcal{A} \end{array} \begin{array}{c} \begin{array}{cc} \mathcal{S} & \mathcal{A} \end{array} \\ \begin{bmatrix} \mathbf{0} & \mathbf{C}_{\mathcal{SA}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \end{array} \tag{2.16}$$

Now that the structure is defined, and following derivations from paper [7], we are able to reformulate MDPs in the RSP framework but defined in a more local way, at the edge level as in [19, 18]. This will allow to adjust the constraints on action-to-state probabilities more easily. Moreover, an advantage of this approach is that it can adapted more easily to other divergence regularizations (e.g. Tsallis entropy).

2.3.4 Randomized MDP problem statement

First and as shown in Appendix A of [29], KL-regularized objective function is equivalent, at a local level to:

$$\phi(\mathbf{P}) = \sum_{(i,j)\in\mathcal{E}} \bar{n}_{ij} c_{ij} + T \sum_{a\in\mathcal{V}\setminus i} \bar{n}_i \sum_{j\in\mathcal{S}ucc(i)} p_{ij} \log \frac{p_{ij}}{p_{ij}^{\text{ref}}} \tag{2.17}$$

The notation, previously recalled in Subsection 2.2.3, denotes the expected number of visits through node $i \in \mathcal{V}$ as $\bar{n}_i$, and the expected number of passages through edge $(i,j) \in \mathcal{E}$ as $\bar{n}_{ij}$. According to conservation of flows, $\bar{n}_i$ can be computed using the equation $\bar{n}_j = \sum_{i\in\mathcal{P}red(j)} \bar{n}_{ij} p_{ij}^{\text{ref}} + \delta_{1j}$ (as described initially in [3]), where $\mathcal{P}red(j)$ is the set of nodes that have a directed edge incident to node j. δ_{1j} refers to the first node having no predecessors and an injected unit flow. It's also worth noting that this local formulation is closely related to the dual linear program used to solve standard MDPs (more details in [33]), which includes here a KL divergence regularization term.

We can also add the alternative function when using r-Tsallis divergence [40, 28]:

$$\phi(\mathbf{P}) = \sum_{(i,j)\in\mathcal{E}} \bar{n}_{ij} c_{ij} + \tfrac{T}{r-1} \sum_{a\in\mathcal{V}\setminus i} \bar{n}_i \sum_{j\in\mathcal{S}ucc(i)} p_{ij} \left(\left(\frac{p_{ij}}{p_{ij}^{\text{ref}}} \right)^{r-1} - 1 \right) \tag{2.18}$$

which we won't go into the details here since it's a similar derivation and it's already developed in [7] and initially in [29]. We will use this other formulation later in the experiments as it has the nice property of providing sparse policies.

Now and in order to obtain the optimal policy, we have to minimize the KL-regularized objective function with respect to the local transition probabilities p_{ka}^{ref} instead of paths probabilities:

$$\begin{aligned}
\underset{\{p_{ka}\}_{k\in\mathcal{S},a\in\mathcal{A}(k)}}{\text{minimize}} \quad & \phi_1^{\mathcal{S}} = \sum_{k\in\mathcal{S}} \sum_{a\in\mathcal{A}(k)} \bar{n}_k p_{ka} \left(c_{ka} + T \log \frac{p_{ka}}{p_{ka}^{\text{ref}}} \right) \\
\text{subject to} \quad & \bar{n}_k = \sum_{a\in\mathcal{P}red(k)} \bar{n}_a p_{ak}^{\text{ref}} + \delta_{1k} \quad \text{for each state } k \in \mathcal{S} \\
& \bar{n}_a = \sum_{k\in\mathcal{P}red(a)} \bar{n}_k p_{ka} \quad \text{for each action } a \in \mathcal{A} \\
& \sum_{a\in\mathcal{A}(k)} p_{ka} = 1 \quad \text{for all } k \neq n_{\mathcal{S}} \\
& p_{ka} \geq 0 \quad \text{for all states and actions } k \in \mathcal{S}, a \in \mathcal{A}(k) \\
& \bar{n}_k, \bar{n}_a \geq 0 \quad \text{for all states and actions } k \in \mathcal{S}, a \in \mathcal{A}(k)
\end{aligned} \tag{2.19}$$

where node 1 is the initial state and $n_{\mathcal{S}}$ is the goal state. Notice also that each state probabilities summed over the actions to 1, except for the absorbing state since it has no successor nodes.

This brings the formulation of the associated Lagrange function, without the non-negativity constraints since it's unnecessary when using KL divergence regularization

as recalled in Subsection 2.1.2:

$$\mathscr{L}(\mathbf{P}_{\mathcal{SA}}, \boldsymbol{\lambda}) = \sum_{k \in \mathcal{S}} \sum_{a \in \mathcal{A}(k)} \bar{n}_k p_{ka} \left(c_{ka} + T \log \frac{p_{ka}}{p_{ka}^{\text{ref}}} \right) + \sum_{k \in \mathcal{S} \setminus n_{\mathcal{S}}} \mu_k \left(1 - \sum_{a \in \mathcal{A}(k)} p_{ka} \right)$$
$$+ \sum_{k \in \mathcal{S}} \lambda_k^{\mathcal{S}} \left(\sum_{a \in \mathcal{P}red(k)} \bar{n}_a p_{ak}^{\text{ref}} + \delta_{1k} - \bar{n}_k \right) + \sum_{a \in \mathcal{A}} \lambda_a^{\mathcal{A}} \left(\sum_{k \in \mathcal{P}red(a)} \bar{n}_k p_{ka} - \bar{n}_a \right) \quad (2.20)$$

From this function, we can derive an algorithm using Karush–Kuhn–Tucker (KKT) necessary conditions [10], as suggested in [29], which is based on the method of Lagrange parameters (often called costates in the optimal control literature).

Note that [7] also adapted the Value Iteration method of Eq. (2.15) in order to include an entropy regularization term and thus exploration. We won't go into the details of this method here.

2.3.5 The costate-based algorithm

Following the previous Lagrange formulation and with the help of some nonlinear optimization and optimal control background (see, e.g. [10] for details), the p_{ka} can be considered as the control variables whereas $\bar{n}_k$ can be regarded as the state variables. It means the control variables are chosen by the agent to manipulate state variables (that are uniquely defined), as it is the case here through the flow conservation constraints. In Lagrange formulation, they can therefore be considered independent.

Still from [7] and in order to obtain the randomized optimal policy, they followed a two-step optimization procedure. Starting by minimizing the Lagrange function with respect to the transition probabilities p_{ka}, while fixing the Lagrange parameters:

$$p_{ka} = \frac{p_{ka}^{\text{ref}} \exp[-\theta(c_{ka} + \lambda_a^{\mathcal{A}})]}{\sum_{a' \in \mathcal{A}(k)} p_{ka'}^{\text{ref}} \exp[-\theta(c_{ka'} + \lambda_{a'}^{\mathcal{A}})]} \quad \text{for each } k \in \mathcal{S} \setminus n_s \text{ and } a \in \mathcal{A}(k) \quad (2.21)$$

where the sum-to-one constraint is imposed. It shows that it only depends on the Lagrange parameters, which can be interpreted as "additional costs" in a similar way as in Subsect. 2.2.5 when using a closely related approach with flow constraints instead (Eq. 2.10). They can then be computed, by minimizing, this time, with respect to the expected number of visits both $\bar{n}_k$ for the states and $\bar{n}_a$ for the actions:

$$\begin{cases} \lambda_k^{\mathcal{S}} = \sum_{a \in \mathcal{A}(k)} p_{ka} \left(c_{ka} + \lambda_a^{\mathcal{A}} + T \log \frac{p_{ka}}{p_{ka}^{\text{ref}}} \right) & \text{for states } k \in \mathcal{S} \\ \lambda_a^{\mathcal{A}} = \sum_{k \in \mathcal{S}ucc(a)} p_{ak}^{\text{ref}} \lambda_k^{\mathcal{S}} & \text{for actions } a \in \mathcal{A} \end{cases} \quad (2.22)$$

where we can get $\lambda_a^{\mathcal{A}}$ appearing in Eq. (2.21). Notice that $p_{n_{\mathcal{S}}a}$ and $\lambda_{n_{\mathcal{S}}}^{\mathcal{S}}$ are 0 since the target node has no successors within the set $\mathcal{S}$. From the previous two equations, both steps are iterated until convergence.

It has also been shown in Appendix A of [29] that the Lagrange parameters, at the optimum, can be interpreted as the free energy distance $\phi_i(P^*)$ from each nodes (states or actions) defined in Eq. (2.17). Indeed, the above parameters updates are equivalent to:

$$\begin{cases} \lambda_k^{\mathcal{S}} = -\frac{1}{\theta} \log \left(\sum_{a \in \mathcal{A}(k)} p_{ka}^{\text{ref}} \exp[-\theta(c_{ka} + \lambda_k^{\mathcal{A}})] \right) & \text{for } k \in \mathcal{S} \\ \lambda_a^{\mathcal{A}} = \sum_{k \in \mathcal{S}ucc(a)} p_{ak}^{\text{ref}} \lambda_k^{\mathcal{S}} & \text{for } a \in \mathcal{A} \end{cases} \tag{2.23}$$

which corresponds to the recurrence formula used to compute the free energy distances. It is true only when using KL divergence and it's a nice property we use to develop the new algorithm presented in the next chapter.

The previously stated method is based on the fact the objective function of Eq. (2.17) is strictly convex with respect to edge flows, as shown first in [3]. This function is expressed in terms of transition probabilities and number of visits to nodes, but can also be expressed in terms of edge flows or transition probabilities only. The authors of [29] thus conjectured that the objective function is also convex with respect to the transition probabilities based on the strict increasing, differentiable, and one-to-one correspondence between the two quantities for a unit input flow. Then, because the domain is convex as well, it should be a global minimum. It is still important to note that this remains a conjecture and it would require a more formal assurance. Their simulations suggested also that strong duality holds in all cases.

They also established a similar reasoning when using Tsallis entropy (Eq. 2.18) in their Appendix B.

Chapter 3

Randomized MDP with multiple inputs and outputs

So far, we developed a way of solving multiple inputs and outputs RSP problems using an extended graph and an approach based on capacity constraints. We also presented a way of seeing Markov Decision Process as standard RSP problems with additional constraints from the environment and a method to find the associated transition probabilities. It's now time to combine both, meaning we want to use MDP's approach to handle several inputs and outputs and that is what this chapter will be about.

The goal is now to find the optimal stochastic policy for carrying an input flow from multiple margin-constrained source nodes to multiple margin-constrained target nodes minimizing the expected transportation cost regularized by an entropy term while fixing probabilities from the environment.

In order to do that, we again need to define a new graph structure that regroups ideas from both contexts. We then define the probabilities and introduce an approach that allowed us to adapt the algorithm presented in Subsection 2.2.6 to the case of a MDP problem.

3.1 Definition of the extended bipartite graph

We need to build a new directed bipartite graph with state and action nodes as in Subsect. 2.3.2 but here, with some additional edges and nodes translating the fact we want an equivalent single input and output extended graph as in Subsect. 2.2.2. We will call this new graph $G_{b,\text{ext}}$.

We thus have, as before from G_b, the left part of the nodes defined as the states of the initial MDP, whereas the right nodes are the possible actions available at each nodes. Concerning the input part, we first add a state super-node acting as single source. We do have a certain number of inputs n_{in}, so we follow by linking to this state new control action nodes corresponding to the initial multiple sources. Each action is therefore connected to each input states from the original bipartite graph. Similarly for the outputs, a new action node is connected to each of the original output states. We then link the latest to a second new state super-node acting as a single absorbing target. Even if it doesn't look necessary as it is only pointed out by the associated target action, the absorbing node must be a state in order to keep consistency with the MDP structure. We hence have a new set of state nodes $\mathcal{A}_{\text{ext}}$ with $n_{\mathcal{S}_{\text{ext}}} = n_{\mathcal{S}} + 2$ and a new set of action nodes $\mathcal{A}_{\text{ext}}$ with $n_{\mathcal{A}_{\text{ext}}} = n_{in} + n_{\mathcal{A}} + 1$. In

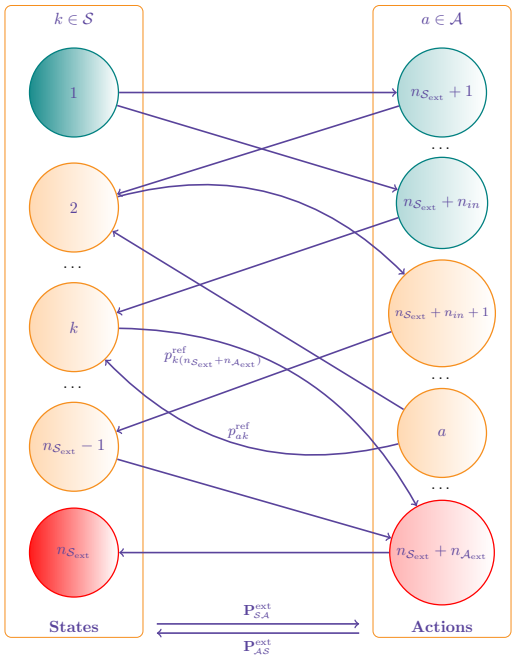

Figure 3.1: Extended MDP modeled as a bipartite graph $G_{b,\text{ext}}$ with $n_{\mathcal{S}_{\text{ext}}}$ states and $n_{\mathcal{A}_{\text{ext}}}$ control actions. We have node 1 as single source supernode in dark green connected to all newly added input actions corresponding to each original input in light green. We have the new "goal" action gathering from initial target states in light red connected to the absorbing state node $n_{\mathcal{S}_{\text{ext}}}$ in dark red. Given also the reference probabilities p^{ref}_{ka} and p^{ref}_{ak} gathered in matrices, the first one being the reference random walk policy while the second one is provided by the environment.

order to be consistent as well with the extended graph structure of Subsection 2.2.2, $\mathcal{I}n$ refers now to the n_{in} new action nodes while the set $\mathcal{O}ut$ doesn't change.

The edges incident to new nodes have zero cost since there is no related charge considering they are artificial. The rest of the graph G_b remains almost the same. The numbering is only modified according to the new sets and the action goal node is indexed with $n = n_{\mathcal{S}_{\text{ext}}} + n_{\mathcal{A}_{\text{ext}}}$. This new graph is, as for G_{ext} and G in Subsect. 2.2.2, still equivalent to G_b and Fig. 3.1 shows an illustrative example.

3.1.1 The reference transition probabilities

We can now describe how the reference transition probabilities are adapted according to graph $G_{b,ext}$ using results already developed in the previous Chapter.

In Subsect. 2.2.3, we talked about the reference random walk behavior, that can be seen as a Markov Chain on the graph. We could use associated properties of conservation of flows and link the probabilities at the source and target to the initially specified flows σ_i^{in} and σ_i^{out}.

In Subsect. 2.3.3, we also defined the reference transition probabilities from a natural random walk and an uniform distribution among the available actions. We now need to reconcile both approaches. Here is what we got for the state-to-action

probability matrix:

$$\mathbf{P}^{\text{ext}}_{\mathcal{SA}} = \begin{matrix} 1 \\ \{2,\dots,(n_{\mathcal{S}_{\text{ext}}}-1)\}=\mathcal{S} \\ n_{\mathcal{S}_{\text{ext}}} \end{matrix} \begin{bmatrix} \overset{\{1,\dots,n_{in}\}}{\boldsymbol{\sigma}_{\text{in}}^{\text{T}}} & \overset{\{(n_{in}+1),\dots,(n_{\mathcal{A}_{\text{ext}}}-1)\}=\mathcal{A}}{\mathbf{0}^{\text{T}}} & \overset{n_{\mathcal{A}_{\text{ext}}}}{0} \\ \mathbf{0} & (\mathbf{I}-\mathbf{Diag}(\boldsymbol{\sigma}_{\text{out}}))\mathbf{P}_{\mathcal{SA}} & \boldsymbol{\sigma}_{\text{out}} \\ \mathbf{0}^{\text{T}} & \mathbf{0}^{\text{T}} & 0 \end{bmatrix} \tag{3.1}$$

Indeed, as for the RSP with multiple inputs and outputs, we have by conservation of flows and using the fact the input super-node is injected a unit flow, that the probabilities at the source are exactly equal to the specified flows.

At the target, again, it's more complicated as we don't know the specific values of the flows passing through each output node. We also cannot use the same strategy used in the non-stochastic case for finding the probabilities since some of them need to be fixed here. In order to avoid having to deal with such complications, we will later use a method that allows to exactly get the required values for the flows even without having consistent reference probabilities. We therefore assign our probabilities in a more simple way similar to the inputs. We just need to normalize the initial state-to-action matrix to have well-defined probabilities regarding the whole graph.

Concerning now the action-to-state probabilities, we have to consider the links at the start between the n_{in} input actions and their associated states. We also included the relationship between the action output supernode and the additional absorbing state. Among the initial states and actions, it doesn't change and everything is provided by the environment:

$$\mathbf{P}^{\text{ext}}_{\mathcal{AS}} = \begin{matrix} \{1,\dots,n_{in}\} \\ \{(n_{in}+1),\dots,(n_{\mathcal{A}_{\text{ext}}}-1)\}=\mathcal{A} \\ n_{\mathcal{A}_{\text{ext}}} \end{matrix} \begin{bmatrix} \overset{1}{\mathbf{0}} & \overset{\{2,\dots,(n_{\mathcal{S}_{\text{ext}}}-1)\}=\mathcal{S}}{\sum} & \overset{n_{\mathcal{S}_{\text{ext}}}}{\mathbf{0}} \\ \mathbf{0} & \mathbf{P}_{\mathcal{AS}} & \mathbf{0} \\ 0 & \mathbf{0}^{\text{T}} & 1 \end{bmatrix} \tag{3.2}$$

where $\sum$ is a $n_{in} \times n_{\mathcal{S}_{\text{ext}}}$ matrix defined as:

$$\left[\sum\right]_{ij} = \begin{cases} 1 & \text{when } i \in \mathcal{A}_{\text{ext}}(1) \text{ and } j \in Succ(i) \\ 0 & \text{otherwise} \end{cases} \tag{3.3}$$

$\mathcal{A}_{\text{ext}}(1)$ corresponds to the actions available at the input node, indexed from 1 to n_{in}. $Succ(i)$ is the set of successor state nodes of action i, which is reduced to one of the initial input state in this case.
Regarding the costs, as every new edges are of zero cost, the result is straightforward:

$$\mathbf{C}^{\text{ext}}_{\mathcal{SA}} = \begin{matrix} 1 \\ \{2,\dots,(n_{\mathcal{S}_{\text{ext}}}-1)\}=\mathcal{S} \\ n_{\mathcal{S}_{\text{ext}}} \end{matrix} \begin{bmatrix} \overset{\{1,\dots,n_{in}\}}{\mathbf{0}^{\text{T}}} & \overset{\{(n_{in}+1),\dots,(n_{\mathcal{A}_{\text{ext}}}-1)\}=\mathcal{A}}{\mathbf{0}^{\text{T}}} & \overset{n_{\mathcal{A}_{\text{ext}}}}{0} \\ \mathbf{0} & \mathbf{C}_{\mathcal{SA}} & \mathbf{0} \\ \mathbf{0}^{\text{T}} & \mathbf{0}^{\text{T}} & 0 \end{bmatrix} \tag{3.4}$$

with no cost associated to action-to-state transitions as assumed in Subsection 2.3.3. The extended bipartite structure is determined and the goal is now to use the results of both MDPs and RSPs with multiple inputs and outputs to solve our new problem. The idea is therefore to use the extended RSP algorithm on the new bipartite graph,

to which we need to adjust the state-to-action probabilities to their fixed values. The next section presents a way to do it and the resulting algorithm.

3.2 The approach based on free energy

Starting with the inputs, we can recall a property involving the free energy when solving a standard randomized MDP through the costate-based algorithm presented in Subsect. 2.3.5. Indeed, it says that at the optimum, the Lagrange parameters are exactly equal to the free energy potentials ϕ_k associated to each node (see Eq. 2.23).

From Appendix C of paper [12], we also know that, in the extended graph, the parameters updates for the inputs given by Eq. (2.13) can also be linked to the free energy as:

$$\lambda_k^{\text{in}} = \frac{1}{\theta}\left(\log z'_{kn} - \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} \log z'_{ln} \right) = \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} (\phi_l - \phi_k) \quad \text{for } k \in \mathcal{I}n \tag{3.5}$$

where z'_{kn} are entries of the fundamental matrix (Eq. 2.14) associated to the goal action super-node in our case, also called the "backward variables". It therefore corresponds to the optimal Lagrange parameters of the MDP and we can use that relationship to combine both approaches and get for the updates:

$$\lambda_k^{\text{in}} = \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} (\phi_l(P^*) - \phi_k(P^*)) = \sum_{l \in \mathcal{I}n} \sigma_l^{\text{in}} (\lambda_l^{\mathcal{A}} - \lambda_k^{\mathcal{A}}) \qquad \text{for } k \in \mathcal{I}n \tag{3.6}$$

where $\lambda_k^{\mathcal{A}}$ are related to action input nodes of the set $\mathcal{I}n$. It means that, by solving the MDP problem, we can fix the action-to-state probabilities while returning the optimal free energy potentials used in the updates of the algorithm. The costs can then, in turn, be augmented and we just need to maintain these steps until convergence of the Lagrange parameters. The margins will also be satisfied since the initial unit flow doesn't depend on the fixed state-to-action transition probabilities. The results hence are derived from the same computations as for the non-stochastic case.

Concerning the outputs, we cannot use the same strategy just used as the deterministic algorithm isn't supposed to manipulate constraints on the probabilities. We will thus use a more flexible method based on Algorithm 2 of [13] that is developed initially to also handle capacity constraints on standard RSP problems. In fact, as in in Subsect. 2.2.5 with Arrow-Hurwicz-Uzawa algorithm [6] and from the convexity of the problem, it exploits Lagrangian duality and proposes a simple gradient-based updating rule, initially from [34] to find a global optimal solution. Through the dual function maximization step, it is found that its gradient is given by:

$$\frac{\partial \mathcal{L}(\lambda)}{\partial \lambda_{ij}} = \bar{n}_{ij} - \sigma_{ij} \tag{3.7}$$

which make senses since it corresponds to the difference between the current flow passing through edge (i, j) and the specified flow we want to fix. Following the additional "virtual" costs interpretation of the Lagrange parameters (Eqs. 21-23

of [13]), the associated update is used to later compute the augmented cost. The quantity $\bar{n}_{ij}$ is computed from the current augmented costs as in the algorithm of Sect. 2.2 at each iteration.

In order to conciliate this with the MDP's approach, we just need to compute the expected number of visits on edges $\bar{n}_{ka}$[1] after solving the associated MDP problem with the augmented costs. This way, the state-to-action transition probabilities are adjusted. Once done, we can update the costs following the gradient-based updating rule:

$$\lambda_k^{\text{out}} = \max(\lambda_k^{\text{out}} + \alpha(\bar{n}_{kn} - \sigma_k^{\text{out}}), 0) \qquad \text{for } k \in \mathcal{O}ut \tag{3.8}$$

$$c'_{kn} = c_{kn}^{\text{ext}} + \lambda_k^{\text{out}} = \lambda_k^{\text{out}} \qquad \text{for } k \in \mathcal{O}ut \tag{3.9}$$

where, as stated before, n is the output action super-node. The second equality of Eq. (3.9) is explained by the initial zero cost associated to edges incident to this node. The augmented cost should increase/decrease in order to reduce/raise the current flow and reach the specified value. The method is finally said to converge as long as the dual function is concave (which is trivial), the step α is positive, is not too large, and the problem is feasible [34].

We hence need to iterate between solving the MDP problem and updating the costs following the two different approaches for the inputs and the ouputs until convergence of the parameters. At the end, the optimal randomized policy is found from the last MDP computations.

Regarding convexity, from [7], it is supposed from conjecture that the MDP problem (Eq. 2.3.4) solved here using the costate-based algorithm is convex. The RSP problem issued from the extended graph of Subsect. 2.2.2 is shown to be convex as it only considers additional linear constraints compared to the standard RSP problem. We can use the same argument in this case considering standard MDPs and say, without formal assurance, that our problem is supposed to be convex. Our small set of simulations suggested that strong duality seems to hold but once again, it remains a conjecture.

We found a way to solve this problem at the output level, but at the expense of performance, as the gradient descent algorithm cannot compete against the extended RSP algorithm. In order to improve the descent, we included ADAM optimizer, which is an adaptive learning rate method firstly described in [24]. It combines the idea of RMSProp optimizer and the Momentum method. It thus keeps a weighted average of the squared gradients related to each weights (here, $\boldsymbol{\lambda}_{\text{out}}$) and divides the gradient by the square root of this average. It also replaces the gradient at each iteration by a linear combination of the current gradient and the previous one. It's done through forgetting factors for both quantities. The stepsize remains in turn constant. We chose this particular method because it showed great flexibility and performance when tuning efficiently the parameters.

Following Subsection 2.3.4, we stated a local equivalent approach to compute the

[1]For which we can use properties of standard MDPs: $\bar{n}_{ka} = p_{ka}\bar{n}_k$ where $\bar{n}_k$ is computed from the transition probabilities as in Eq. (2.19).

optimal stochastic policy associated to standard MDP. As mentionned, it has the advantage of making the use of other divergence regularizations easier. Since our algorithm is based on this method, it is possible to extend it to Tsallis r-divergence as well, which is interesting as it favors sparse policies when getting closer to the optimum. In order to do so, we would however need to use the gradient approach for the inputs instead of flow matching. Indeed, the property used to link free energy distances is only applicable when using KL divergence.

3.3 The developed algorithm

The resulting algorithm is presented in Algorithm 2. The different steps of the process are the following:

1. Describe the extended bipartite graph $G_{b,\text{ext}}$ from the initial bipartite graph G_b and compute its associated quantities (transition probabilities and costs) as described in Section 2.3.

2. Initialize the Lagrange parameters to 0.

3. Iterate the following steps until convergence, update first the quantities associated to source nodes through flow matching, and then update the quantities associated to target nodes from gradient descent:

   - The free energy potential and the expected number of visits through edges are computed from the MDP algorithm (Eq. 2.3.5) on $G_{b,\text{ext}}$.
   - The Lagrange parameters are updated (Eqs. 3.6 and 3.8). Respectively from the flow matching approach of Sect. (2.2) and the method based on gradient descent.
   - The augmented costs are updated (Eqs. 2.12 and 3.9).

4. Compute the optimal stochastic policy (transition probabilities) from the augmented costs on $G_{b,\text{ext}}$ given by Eq. (2.21) of the costate-based algorithm from Subsect. 2.3.5.

The time complexity of the algorithm comes mainly from the linear equations that need to be solved at each step of the costate-based algorithm when updating the Lagrange parameters in Eq. (2.22). It's of order $\mathcal{O}(k(n_{\mathcal{S}_{\text{ext}}})^3)$ where k corresponds to the number of iterations needed for convergence. It takes into account both the inner steps of the MDP algorithm and the ones of the main iteration loop. That parameter can be reduced due to the use of ADAM method optimizing the gradient step. Moreover, as our extended bipartite graph is sparse, the performance can also be improved by bringing more efficient methods for solving sparse systems of linear equations.

Algorithm 2 Solving the MDP-based stochastic optimal transport on a graph problem with multiple sources and targets.

Input:

– A bipartie graph $G_{b,\text{ext}}$ containing $n_{\mathcal{S}_{\text{ext}}}$ left state nodes and $n_{\mathcal{A}_{\text{ext}}}$ right action nodes, derived from the standard MDP based bipartite graph G_b, as detailed in this Section. Node 1 is the source state supernode and node $n = n_{\mathcal{S}_{\text{ext}}} + n_{\mathcal{A}_{\text{ext}}}$ the target action supernode. The indegree of node 1 is 0 while the outdegree of node n is 1 as it goes to the absorbing state supernode.

– The set of input action nodes $\mathcal{I}n$ (only connected to the source supernode 1) and output state nodes $\mathcal{O}ut$ (only connected to the target action supernode n).

– The $n_{\mathcal{S}_{\text{ext}}} \times n_{\mathcal{A}_{\text{ext}}}$ state-to-action transition matrix $\mathbf{P}^{\text{ext}}_{\mathcal{SA}}$ as well as the $n_{\mathcal{A}_{\text{ext}}} \times n_{\mathcal{S}_{\text{ext}}}$ action-to-state transition matrix $\mathbf{P}^{\text{ext}}_{\mathcal{AS}}$ associated with $G_{b,\text{ext}}$. See Eq. (3.1, 3.2).

– The $n_{\mathcal{S}_{\text{ext}}} \times n_{\mathcal{A}_{\text{ext}}}$ non-negative cost matrix $\mathbf{C}^{\text{ext}}_{\mathcal{SA}}$ associated with $G_{b,\text{ext}}$ (see Eq. (3.4)). These costs are equal to zero for edges starting in node 1 and ending in $\mathcal{I}n$ as well as edges starting in $\mathcal{O}ut$ and ending in node n.

– The $n_{\mathcal{S}} \times 1$ vectors of input flows, $\boldsymbol{\sigma}_{\text{in}}$, and output flows, $\boldsymbol{\sigma}_{\text{out}}$, both non-negative and summing to 1.

– The inverse temperature parameter θ.

Output:

– The $n_{\mathcal{S}_{\text{ext}}} \times n_{\mathcal{A}_{\text{ext}}}$ randomized policy defined by the transition matrix $\mathbf{P}^*$.

$\boldsymbol{\lambda}_{\text{in}} \leftarrow \mathbf{0}$; $\boldsymbol{\lambda}_{\text{out}} \leftarrow \mathbf{0}$ ▷ initialize $n_{\mathcal{S}_{\text{ext}}} \times 1$ Lagrange parameter vectors

$\mathbf{C}' \leftarrow \mathbf{C}_{\text{ext}}$ ▷ initialize the augmented costs matrix (recall that $\mathbf{C}^{\text{ext}}_{\mathcal{AS}} = 0$)

repeat[main iteration loop]

  $\phi_a, \bar{n}_{kn} \leftarrow$ Solve MDP($\mathbf{C}'$) ▷ free energy and current flows associated to action and state nodes

  ▷ from costate-based algorithm developed in Eq.(2.3.5)

  for all $k \in \mathcal{I}n$ **do** ▷ initialize Lagrange parameters associated with source nodes

    $\lambda^{\text{in}}_k \leftarrow -\phi_{a,k}$ ▷ compute Lagrange parameters

  end for

  for all $k \in \mathcal{I}n$ **do** ▷ update Lagrange parameters and costs associated with source nodes

    $\lambda^{\text{in}}_k \leftarrow \lambda^{\text{in}}_k - \sum_{k' \in \mathcal{I}n} \sigma^{\text{in}}_{k'} \lambda^{\text{in}}_{k'}$ ▷ normalize Lagrange parameters

    $c'_{1k} \leftarrow \lambda^{\text{in}}_k$ ▷ update augmented costs (recall that $c^{\text{ext}}_{1k} = 0$ for all $k \in \mathcal{I}n$))

  end for

  for all $l \in \mathcal{O}ut$ **do** ▷ gradient descent: update quantities associated to output nodes

    $\delta_l \leftarrow \bar{n}_{ln} - \sigma^{\text{out}}_l$ ▷ compute the difference between the current flow and σ^{out}_l

    $\Delta_l \leftarrow$ ADAM(δ_l) ▷ return the update computed from ADAM method [24]

    $\lambda^{\text{out}}_l \leftarrow \max(\lambda^{\text{out}}_l + \Delta_l, 0)$ ▷ update the Lagrange parameters

    $c'_{ln} \leftarrow \lambda^{\text{out}}_l$ ▷ update augmented costs (recall that $c^{\text{ext}}_{ln} = 0$ for all $l \in \mathcal{O}ut$))

  end for

until convergence of $\boldsymbol{\lambda}_{\text{in}}$, $\boldsymbol{\lambda}_{\text{out}}$

$\mathbf{P}^* \leftarrow$ Solve MDP($\mathbf{C}'$) ▷ compute optimal policy

return $\mathbf{P}^*$

Chapter 4

Simulations and results

This chapter covers the more practical part associated to the algorithm we developed. The main interest is to see how algorithms, from the RSP context, can be applied to a medium-sized problem and to check if these behave as we would expect. We focused our computations on a network representative of a neighborhood of Louvain-la-Neuve. An illustrative example of smaller size is also included in order to introduce the process.

The model studied in this paper is rather general as it combines the extension to multiple inputs and outputs as well as the MDP reformulation. We aim at discovering if such a general method could be useful in some situations. It's already not competitive with more optimized models developed in more specific contexts. We still did a brief study on how we slightly improve the performance of our algorithm at the end of the chapter.

For each of the following simulations, we tried at least one other configuration of margins in order to see if the conclusions we had could be generalized. The conclusions stated here were thus verified to an extent.

4.1 Illustrative example: a simple board game

In this section, we first show how the algorithm presented in Sect. 3.3 can be applied to a transportation problem on a small board game example. Let us consider a board as described in Fig. 4.1. Each square represents a state and to each state corresponds four actions translating every possible directions. The cost associated to these actions is constant with a value of 1 and such that the cost can be interpreted as the number of moves with no preferred direction. Randomness is then brought through some barriers of different permeability. It means that, at some square, there are different probabilities p of following some chosen direction or of staying in place on the contrary. We consider that some players at the top two corners must move in order to reach the bottom two corners.

The bipartite graph is constructed from the previously defined states and actions, which is then extended as explained in Section 3.1. We also need to specify the partition of flows among the inputs and outputs. We defined these in vector form as $\boldsymbol{\sigma}_{\text{in}} = [1/2; 1/2]$ and $\boldsymbol{\sigma}_{\text{out}} = [8/10; 2/10]$. It's then possible to determine the matrices associated to the graph, that is the reference transition probability matrix and cost matrix. They are defined following the instructions of Subsect. 3.1.1 when no prior information is provided concerning the weights. The state-to-action probabilities are provided from the different barriers described before.

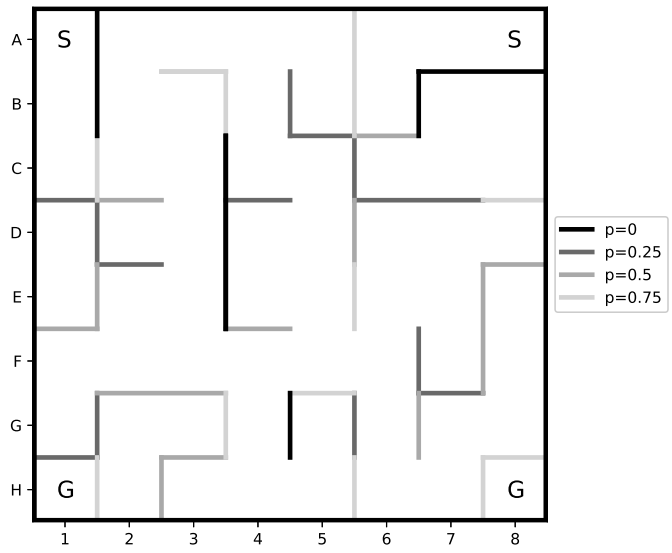

Figure 4.1: Structure of the simple board game example

Transportation solutions provided by Algorithm 2 for different θ values are provided in Fig. 4.2. We compared the results obtained when using KL divergence and when using Tsallis 2-divergence. In order to do so, we chose θ values such that the optimal expected cost of both solutions is similar. We can notice that using a value twice bigger for Tsallis brings more or less the desired result. It can be explained partially from the fact Tsallis 2-entropy graph tendancy has a magnitude two times smaller compared to Shannon entropy, as shown in [30]. We should however not forget the fact both entropies provide a different trade-off among the probabilities and thus that the solutions may differ even for a similar entropy value. As we already mentioned, Tsallis divergence tends to result in sparse policies when getting closer to the optimum. Fig. 4.3 illustrates that fact for $\theta = 1$ and $\theta = 2$.

We can observe, on Fig. 4.2, that the optimal total transportation cost (or the number of moves) of around 13.8 is reached when $\theta = 10$ and $\theta = 20$ respectively. We can see that this cost moves further and further away from its minimal value as θ decreases. It makes sense as the lower θ the inverse temperature parameter gets, the more it favors paths following the reference natural random walk behavior. Indeed, the divergence term is given more weight and the costs are no longer considered. For that reason, sub-optimal directions in terms of cost are chosen, which increases the total expected cost. This can be seen on Fig. 4.2 for low values of θ. Darker colors are used translating the fact agents are more likely to spread out on the board instead of choosing an optimal direction. The optimal directions are also sometimes different.

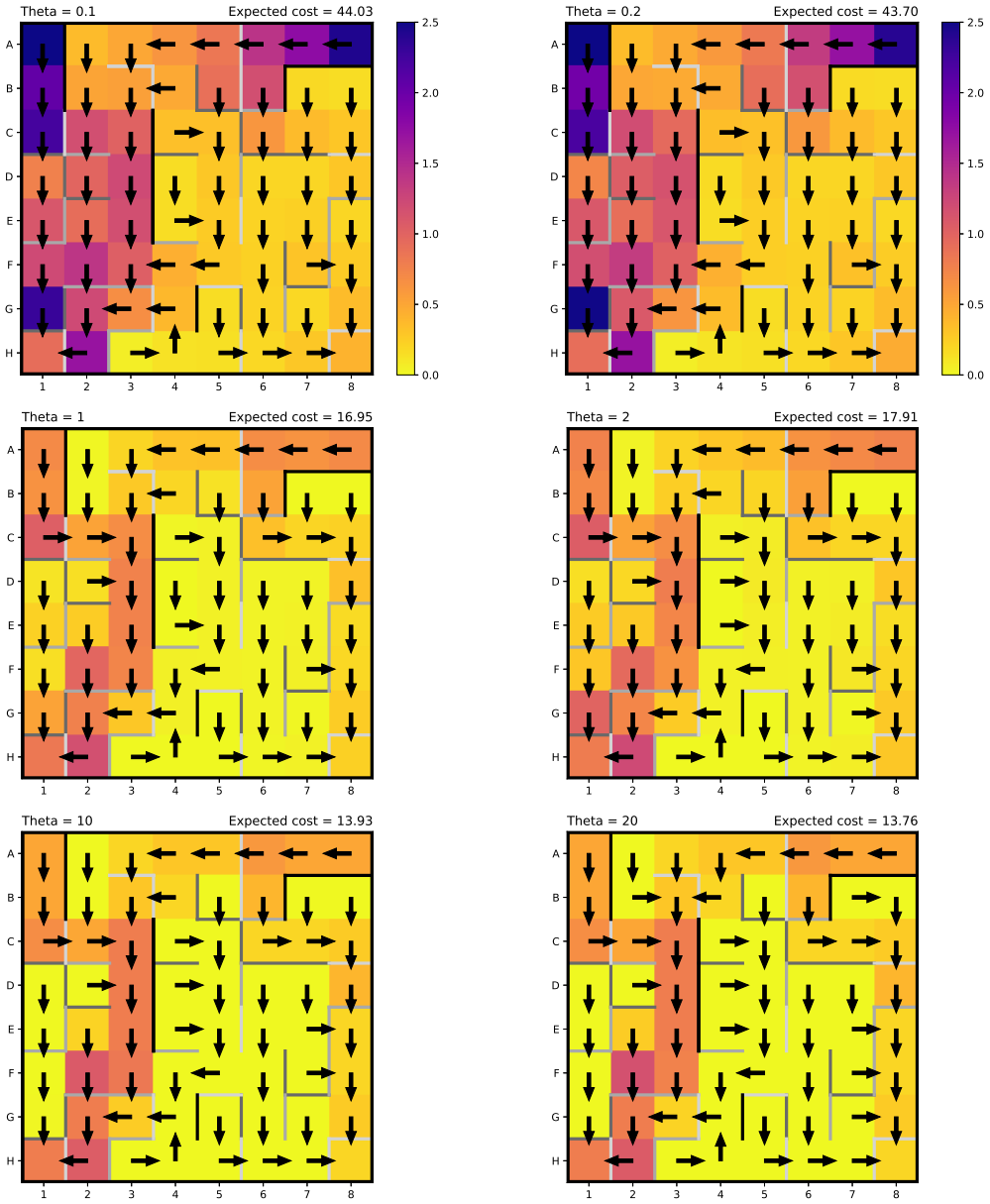

Figure 4.2: Representation of the solutions of a simple board game problem presented above for KL divergence on the left, Tsallis 2-divergence on the right and different θ values. Intensity of the colors is proportional to the expected number of visits while arrows represent the most likely direction at each square.

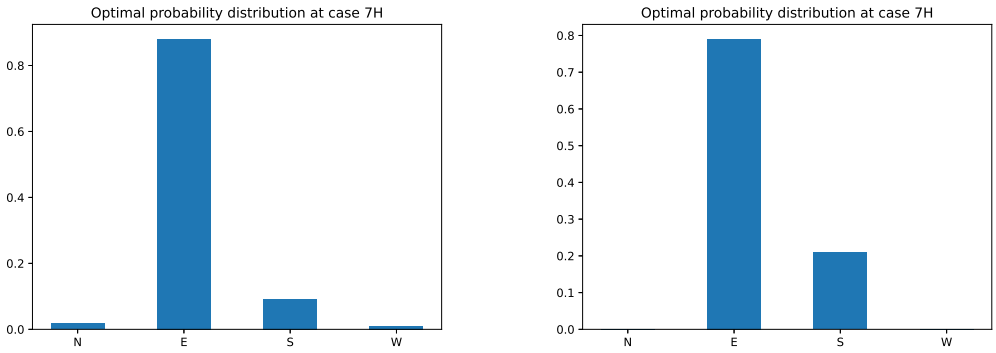

Figure 4.3: Example of optimal policy considering KL and Tsallis 2-divergence for $\theta = 1$ and $\theta = 2$ respectively at square H7.

4.2 Pseudo traffic example: a neighbourhood of Louvain-la-Neuve

4.2.1 Experimental design

This case study is based on a real-life network. In a neighborhood of the city of Louvain-la-Neuve, we consider that some provider is planning to deliver goods from several places to multiple destinations. States are hence defined to be the set of every existing place present around the area including additional simple road junctions. Actions correspond to every road reachable from every particular states. Costs associated to a particular road can then be updated with its length. Finally, we defined the reference transition probabilities associated to each road in order for the weights of the adjacency matrix a_{ij} to be inversely proportional to the costs (as introduced in Eq. 2.1.1). The associate random walker will thus prefer roads of smaller lengths, as we wanted to. The map of the neighborhood is shown in Fig. 4.4.

Note that we also led some preprocessing steps on the data used to construct the mentioned network. Some were necessary for the good behavior of our algorithm while others were only used for convenience, to decrease the size of the resulting graph. First, we removed states that weren't reachable from any other state since they would never be reached. We also let aside those who were "absorbing" in the sense they hadn't any outgoing road. We didn't consider roads that end at the same state it started (loops) as well as roads with the same starting and ending point (multiple edges). We only kept for the latter the roads with minimum length which are more interesting. We ended up with a total of 944 state nodes and 2459 action nodes.

4.2.2 Results and discussions

Simple transport example with a deterministic environment

Through this example, we show how our algorithm can now be applied to a transportation problem on a real-life road network. The goal is to highlight the effects

of exploration on this network as well as the differences between the use of KL and Tsallis 2-divergence.

We must first define the extended bipartite graph issued from the experimental design. In order to do so, we define the source flows and target flows as $\boldsymbol{\sigma}_{\text{in}} = [1/3; 1/3; 1/3]$ and $\boldsymbol{\sigma}_{\text{out}} = [1/2; 1/2]$ which represent the proportions of goods available at each input place. The state-to-action reference probabilities and costs are found as suggested above. The action-to-state transitions corresponding to a deterministic case are associated unit probabilities at the end state related to some road or action. In such case, when using KL-divergence, the results must match with the ones provided by the deterministic algorithm from paper [12]. Indeed, one of the constraints of Eq. (2.3.4) become trivial and we end up with the same problem as in paper [29] (which is an equivalent problem at the local level).

It can be seen in Fig. 4.2, which gives the solutions of our algorithm in the form of maps for different θ values. We again compared the results obtained when using KL divergence and when using Tsallis 2-divergence for similar expected costs. We can notice that the optimal total transportation cost is reached when the inverse temperature parameter is the highest and it increases as it moves closer to 0. We see that, as θ increases, the flows become higher on every edge because of the random movements of deliverers, especially around the itineraries of smaller length. Note that we specified the maximum values of the number of visits for each scenario. Indeed, the scales used to render the flows are different as we normalize the widths with respect to these maximum values. We did so as otherwise, we wouldn't be able to distinguish the same flow differences for more deterministic cases, which correspond to high θ values.

Comparing both types of regularization, we also end up with the same observations as for the illustrative example. We can see that Tsallis divergence results in more sparse policies as we have less small dark edges on the figures. This effect is moreover enhanced as we move closer to the optimum, as shown with $\theta = 10$ and $\theta = 20$.

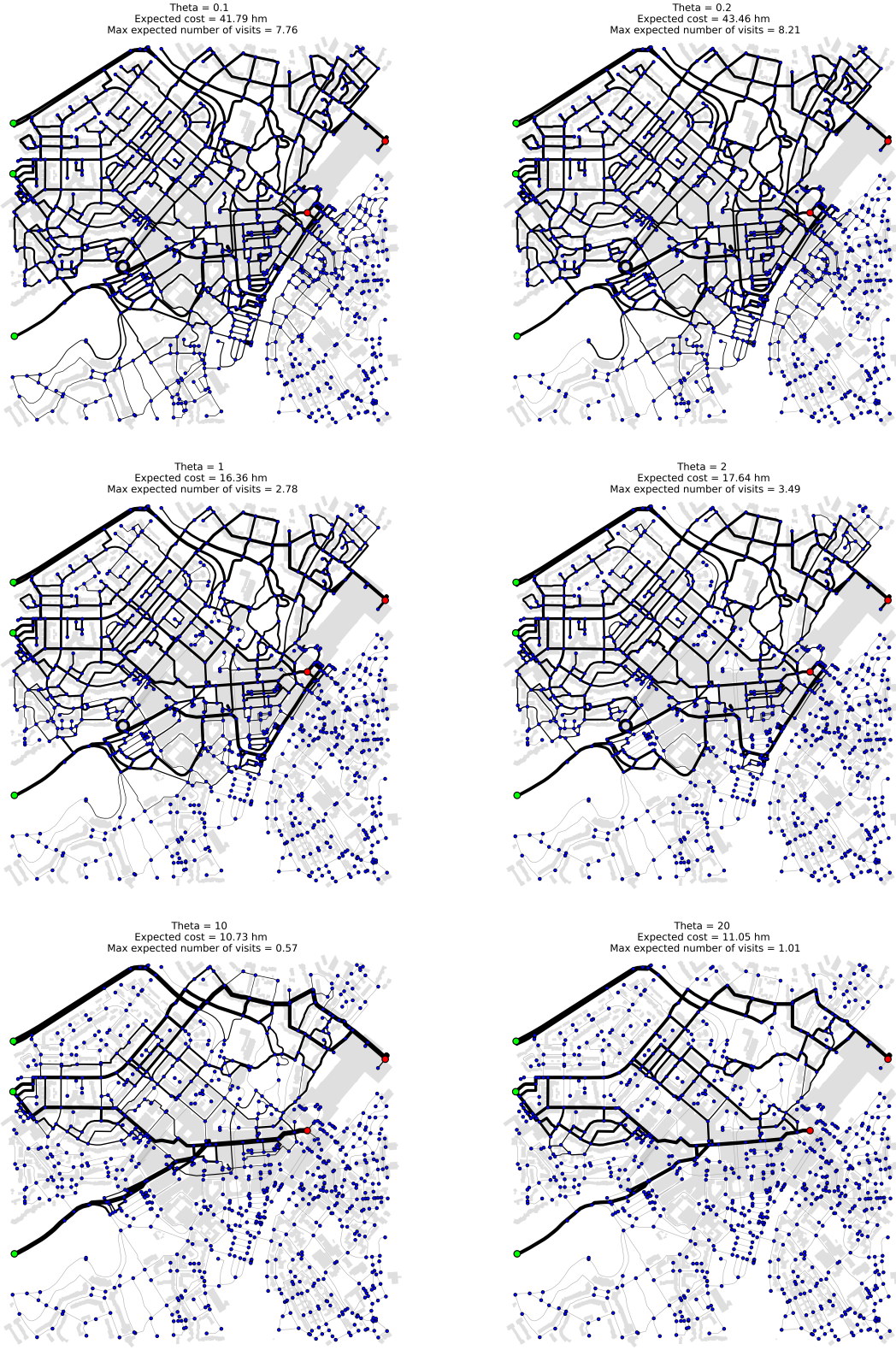

Figure 4.4: Representation of the solutions of a simple deterministic transportation problem for KL divergence on the left, Tsallis 2-divergence on the right and different θ values. The different edge widths indicate the intensity of the flow passing through that edge.

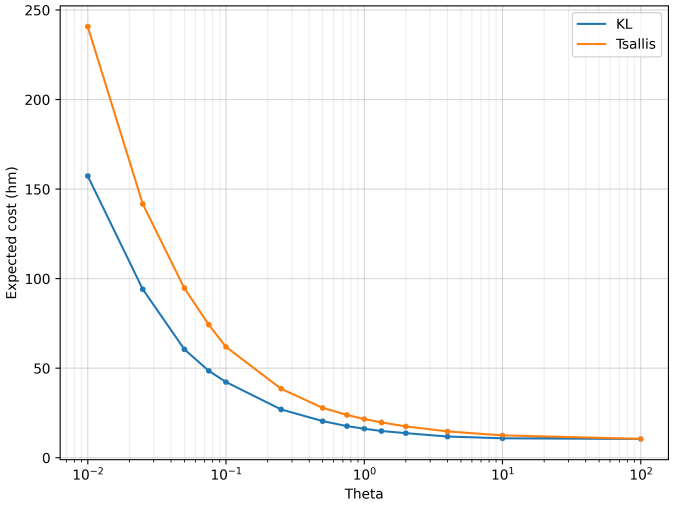

Figure 4.5: Comparison of the expected cost of a simple deterministic transportation problem between KL-regularized and Tsallis-regularized method with respect to the inverse temperature parameter θ.

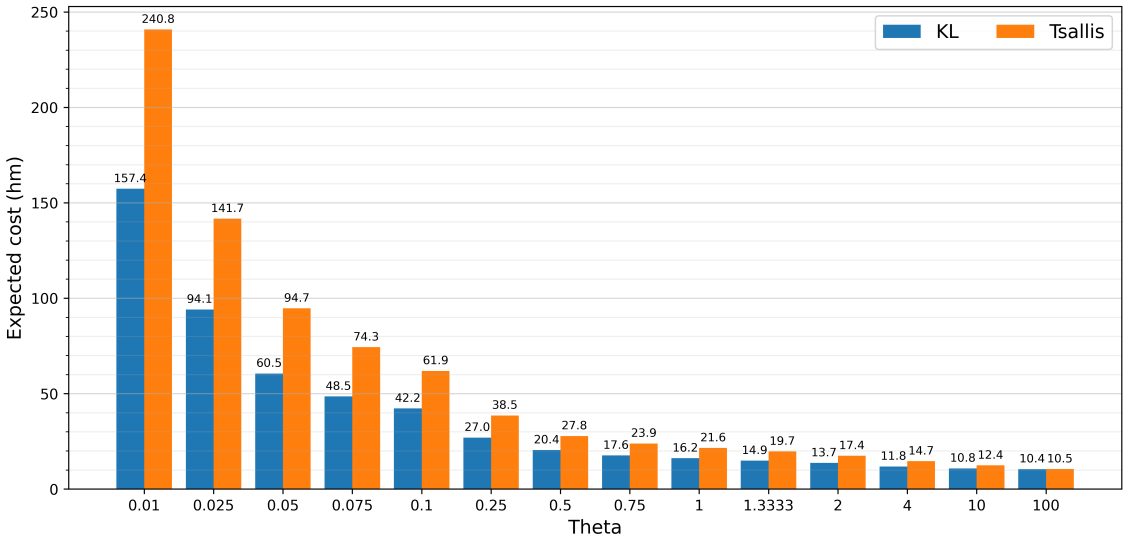

Figure 4.6: Comparison of the expected cost of a simple deterministic transportation problem between KL-regularized and Tsallis-regularized method with respect to the inverse temperature parameter θ.

The above two graphs show the evolution of the optimal expected cost, expressed in hectometers, regarding different values of θ. As we could imagine, the costs are decreasing when the parameter is increasing. With values closer to 0, the optimal policy approaches the pure random walk behavior which leads to a cost quickly diverging from the random exploration. On the contrary, around high values of θ, the final policy is progressing towards an optimal least-cost strategy. The costs are thus converging slowly to the associated optimal value. We also remark a delay when using Tsallis divergence instead of KL divergence which comes mainly from the differences of scaling between the two entropies.

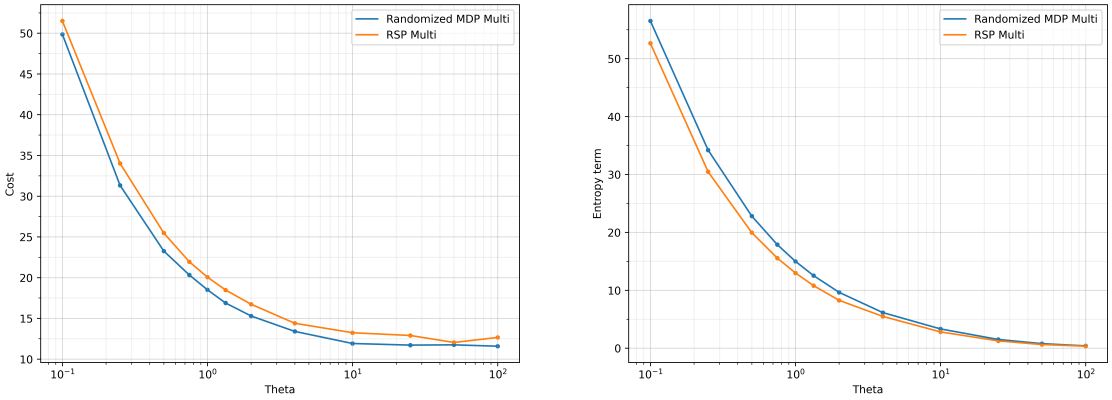

Figure 4.7: Comparison of the expected cost on the left, the entropy term on the right of a random transportation problem between KL-regularized randomized MDP and RSP methods for multiple inputs and outputs regarding the inverse temperature parameter θ.

Example with a stochastic environment: "GPS malfunction"

The goal of this subsection is to demonstrate the advantage of the MDP reformulation compared to a deterministic RSP framework through a stochastic transportation example with multiple inputs and outputs. More precisely, we will compare the results obtained using our algorithm and the method recalled in Subsect. 2.2.6, both KL-regularized. In order to introduce randomness into the environment, we imagined the context of someone following its GPS. However, this device is malfunctioning and at each state, we consider there is a 10% chance of being deviated from a direction that was chosen initially. This chance is then evenly distributed among the other possible roads reachable from the state.

Otherwise, we kept the same margins as the previous example. This way, we will be able to underline more easily some differences with the deterministic environment as well. Only the action-to-state probabilities are modified, following the GPS malfunction context, which thus become randomized.

First, Fig. 4.7 is showing us the difference of expected transportation cost and entropy between both algorithms, regarding the inverse temperature parameter θ. We see that our algorithm seems to be slightly better in general. However, it is necessary to also look at the difference of entropy before doing any conclusions. Indeed, as the MDP approach considers additional constraints provided by the environment, we cannot be sure that the associated entropy is exactly equivalent. The right figure shows that the divergence term is slightly bigger when using our algorithm. It means that, in addition to providing a better result, it considers a higher level of entropy which leads to a more randomized policy. So when comparing situations where both methods return a similar divergence term, the benefit of our method is even more noticeable. This gap of entropy is decreasing as we move towards more deterministic situations but the difference of cost still persists. Intuitively, it makes sense as since the deterministic method isn't taking into account the random deviations caused by the environment, it shouldn't be better that our method which does.

Solutions of both algorithms in the form of maps are again provided for different

θ values in Fig. 4.8. As we recalled above, the optimal policy given by the RSP is the same as the one of our algorithm when considering the previous deterministic problem. As it doesn't consider the stochasticity brought by the environment, it behaves as if there wasn't any. Indeed, if we compare the current sets of figures with Fig. 4.4, there aren't many differences in terms of flows. We perceive that there are few more edges that are visited, which come from the randomness of the problem and leads to more exploration in its own way. It then results in a higher transportation cost. Concerning our method, we notice an even more scattered pattern. We previously mentioned that the trade-off between costs and the divergence term is different as in the deterministic case. We saw that the entropy term was bigger here for the same value of θ, which could explain this difference of exploration. At similar values of entropy, the costs are thus lower than with the RSP method. But they are still higher compared to the non-stochastic scenario.

Note that the expected numbers of visits tend to behave quite well. They indeed increase together with the exploration. We were surprised however by the maximal value of our algorithm's solution, with $\theta = 10$. We investigated and found out that there was a kind of loop between two nodes around the top left source with nearly 1 probabilities. These states were therefore associated a value of about 3 whereas the maximal value afterwards was only of around 0.5. We suggested that it was caused by an unpredictable situation coming from the random environment.

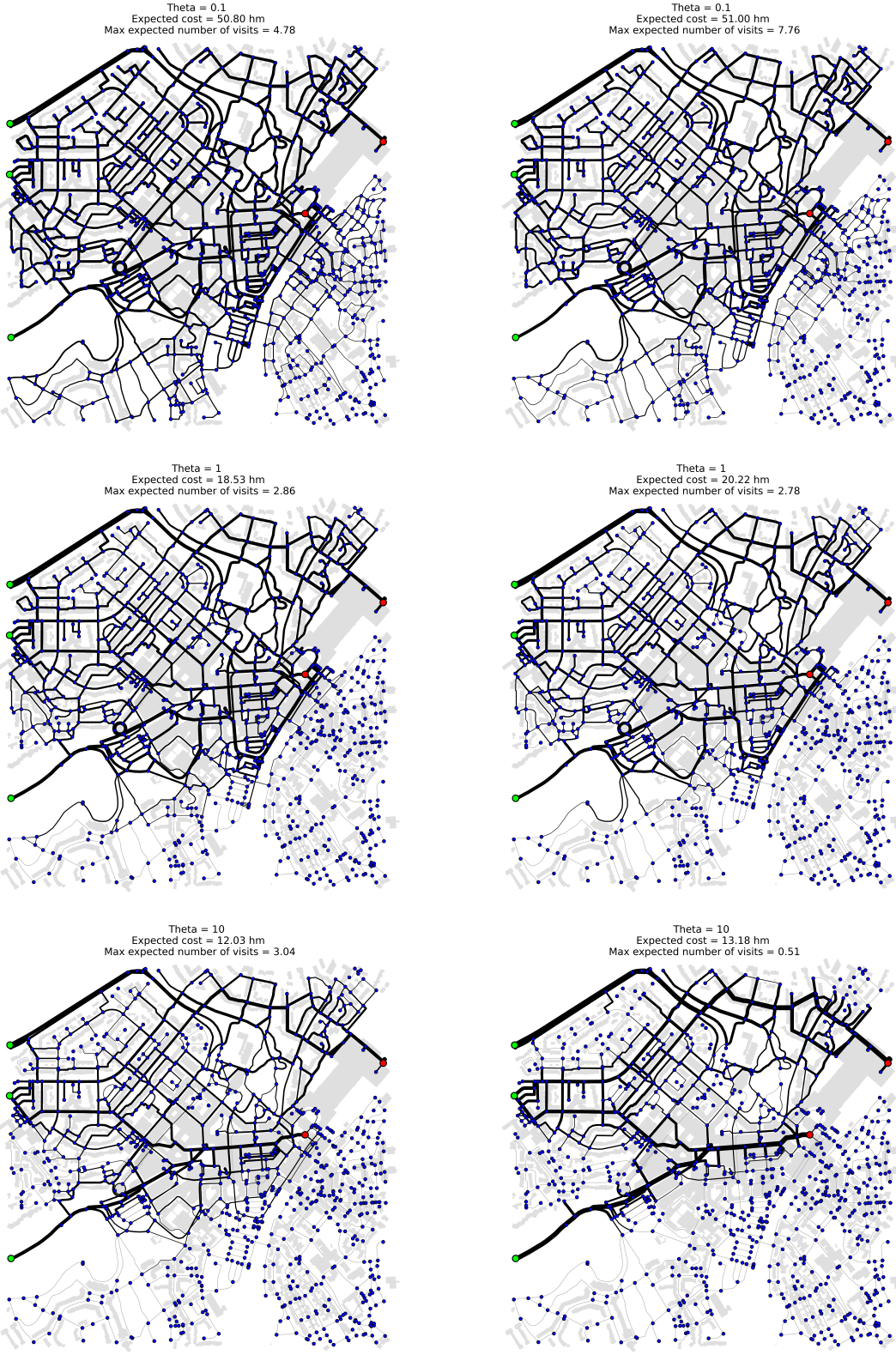

Figure 4.8: Representation of the solutions of a stochastic transportation problem for our KL-regularized randomized MDP method on the left, RSP with multiple inputs and outputs on the right and different θ values.

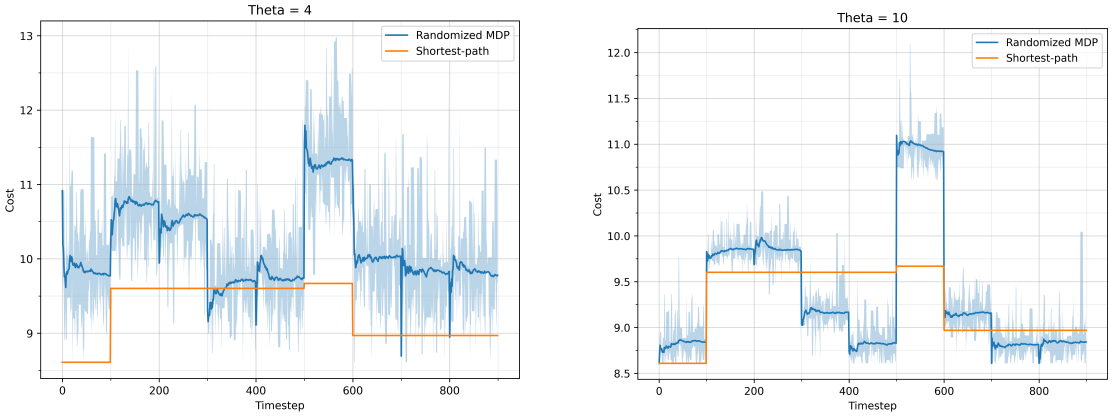

Figure 4.9: Comparison of the expected cost of a deterministic transportation problem with simple traffic between a completely deterministic policy and a KL-regularized randomized policy regarding the inverse temperature parameter θ.

Deterministic example with simple traffic: "Waze method"

We aim, through this deterministic example with simple traffic, to try to justify the use of a randomized policy instead of an optimal least-cost policy. The idea would be to inject some traffic on trajectories of lowest lengths, which are likely to be taken when following a deterministic optimal policy. The two models studied would be aware of this traffic and would change their policy accordingly. The benefit of exploration would then come from the time the traffic is finished. We consider local information in the sense that the changes happening on the map are only known if the models are using some of the roads associated to this change. More exploration would thus allow to learn about these changes and to use a policy with a lower length. It can be compared to the traffic application "Waze" which takes into account the informations given by its users in order to change the suggested path.

In more details, we would compare the results provided by our method with KL divergence and the standard shortest-path algorithm. We would do so through a transportation example with only one source and one target since one of the two policies is completely deterministic. Concerning the traffic, here is how we implemented it:

- The traffic is induced via some additional constant costs.
- The associated disturbed roads include roads used when following a least-cost policy.
- We consider two periods of traffic. The first one is defined with respect to the least-cost policy when there is no traffic. The second extends this traffic to the updated policy used when the first traffic was incorporated. Fig. 4.10 shows the associated trajectories. Second traffic concerns thus both of these paths.
- Each traffic is generated at different time and for one period. After each period, we consider 3 periods without traffic. The first period of the simulation concerns a network without any traffic.

- Models are informed of the traffic. They however don't know when the traffic is over unless they visited the previously disturbed roads. The 3 periods following a traffic thus allow to gather informations about occurring changes in this traffic.

- We consider periods of 100 time-steps. After a period, the models are able to use the informations available to update their policy.

A comparison of the transportation lengths is shown in Fig. 4.9 for two values of θ. For $\theta = 4$ and both traffic, we see that after only one period, the expected cost goes back to its optimal value without traffic. It means the exploration is such that every disturbed roads are still visited. However, as the optimal cost without traffic is still high, the deterministic policy is better. For $\theta = 10$ and the first traffic, two periods are needed to go back to the normal case. It means it first visits some of the previous disturbed roads but not all of them. It then updates its policy and following this new trajectory, it visits the remaining ones. For the second traffic, all the roads are visited after one period. We notice here that the policy becomes better than the shortest-path since its initial cost is not much higher.

In this context, Fig. 4.10 reveals the optimal behavior on two (resp. three) periods following the first traffic for both values of θ. We indeed notice how both schemes gathers roads that were cluttered just before and how both are modified after having perceived the cost changes. The variance around the expected cost is also lower when increasing θ as the behavior gets closer to a deterministic one.

In Fig. 4.9, we expected from the deterministic policy to have 2 straight lines corresponding to the two intervals of actual traffic. Indeed, as it doesn't explore at all, it shouldn't change after the traffic is over. It's however happening here after the second traffic as the modified optimal strategy still included some of the cluttered roads.

We also have some remarks to add about this example. First, to simulate traffic, we induced an additional cost of 0.1 hm. Let's not forget that if we used an higher value, the exploration in the disturbed roads wouldn't be that high and we wouldn't have such good results. We would need to find an optimal value for θ in order to induce indeed some exploration. Even though, for higher costs, a smaller value of θ would lead to higher general costs and more variability which is not really recommended. We also see here that when incorporating the second traffic, we have a quite high gap with the standard optimal policy provided by our method even for $\theta = 10$. This is more unexpected but it's still appearing as some drawback of exploration.

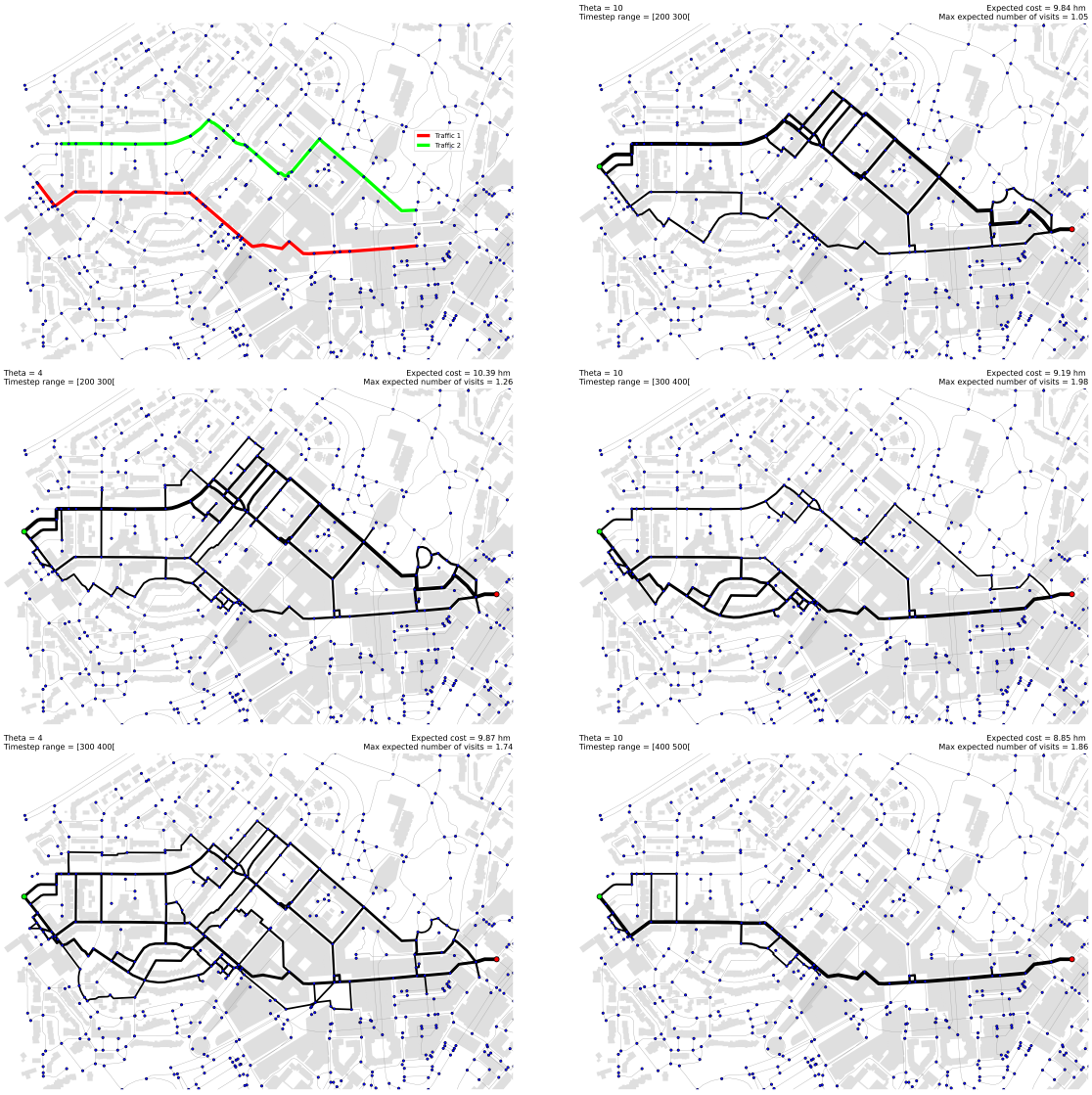

Figure 4.10: Representation of the solutions of a deterministic transportation problem with simple traffic for a completely deterministic policy on the left, a KL-regularized randomized policy on the right and during several periods of the process. It is done for two values of θ while the top left figure shows only the location of the two traffic paths.

Stochastic example with dynamic traffic and congestion

The purpose of this example is, as in the previous case, to try to justify the use of more randomized policies. However, the idea would be to include here a stochastic environment as well as multiple sources and targets. Indeed, our algorithm allows to handle both of these features and we will try to find interesting results when actually considering such an extension. An interesting way to simulate traffic, in a more realistic way as the previous example is to induce traffic on the roads but proportionally to the agents circulating around these roads. More busy trajectories would thus be more inclined to have traffic and congestion (slowdown in traffic) would occur. Concerning the randomness provided by the problem, we decided to use the same GPS malfunction context we use through the second simulation. Another

notable difference with the previous example is the level of information. Indeed, the idea here is to consider global information, which means every moving agent is aware of every traffic change happening around the map. We only accorded that knowledge to previously visited roads when considering traffic before. This could be compared to a global server providing informations to anyone on the road. So we can focus on what we might intuitively think is an advantage of exploration, which is the fact more spread out strategies would likely to decrease any start of congestion.

Precisely, we would compare the results provided by our method but with different levels of entropy. We consider a transportation example with the same margins as we used in the first example. We then update the state-to-action probabilities in the same way as the other stochastic example. Concerning the traffic, here is how we implemented it:

- The traffic is induced via some additional variable costs which are proportional to the number of visits.
- The traffic would therefore concerns every visited road.
- In order to simulate a scenario of congestion, we use a cost function of the form $f(n) = \alpha n^2$ where n is the number of visits and α, a previously chosen constant. It would induce a cost whose augmentation is increasing with the number of visits, as in real congestion.
- The model is globally informed of the traffic. We consider periods of 100 time-steps. After a period, the model is able to use the information available to update its policy.
- Fig. 4.11 shows two different graphs. The first one considers that after one period, the traffic previously induced disappears. The second graph smooths the process by using an exponential smoothing factor (more details in [9]) which still takes into account older traffic but through a factor which is decreasing exponentially over time.

The evolution of the expected cost (expressed in hectometers) is, as mentioned above, illustrated in Fig. 4.11 for two values of θ. The first graph demonstrates a cost generally lower because we indeed consider that the traffic disappears after one period. We can also distinguish more oscillations coming also from the discontinuous transition of older traffic between the periods. For both cases, we see that the more deterministic method is doing better than the other one, which is opposed to the initial fact we wanted to justify.

Fig. 4.12 shows the actual number of visits after 100 agents have passed through the network and following the two different KL-regularized randomized policies. It is done during 3 different periods of the process in order to underline how both policies react to the traffic changes. We illustrated these quantities instead of the theoretical flows to highlight the fact the exploration after a short period of time looks different from theory where expected flow values as far as 10^{-5} are represented. Compared to the theoretical solutions of previous examples, that explains why there are no narrow edges and why there are not as much difference of exploration when considering

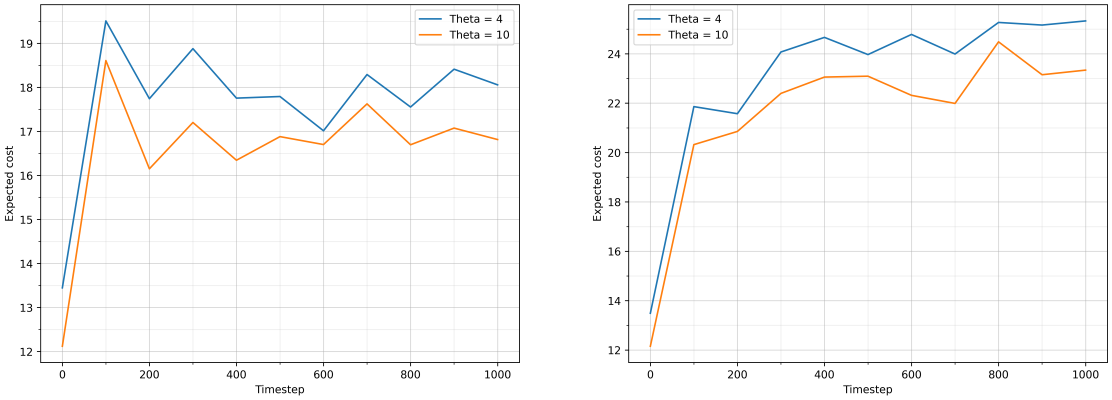

Figure 4.11: Comparison of the expected cost of a stochastic transportation problem with multiple sources and targets between two policies with varying degrees of randomness regarding the inverse temperature parameter θ.

different levels of entropy. We still see a difference, and as we progress through the simulation, we notice that the two policies are behaving well between consecutive periods. The policies are updated and we see that in general, high traffic paths are removed from one period to the other.

So we couldn't justify the exploration objective of avoiding heavy traffic by spreading out the trajectories. We can at least try to explain it. First, we used a value of $\theta = 10$ as a comparison to behave as a policy near to deterministic. As shown on the maps, we haven't seen a major difference in terms of exploration when using $\theta = 4$. If we considered longer periods, the probability to visit some nodes provided by the policy would still be so low that it wouldn't really change the cost tendency obtained. Moreover, lower values of θ would also quickly bring higher expected cost which wouldn't show better results. The random behavior certainly leads to more exploration, but it also increases the frequency of visits, including through the paths used by less randomized policies. So, in such context, a randomized strategy doesn't seem to be really convenient in the end.

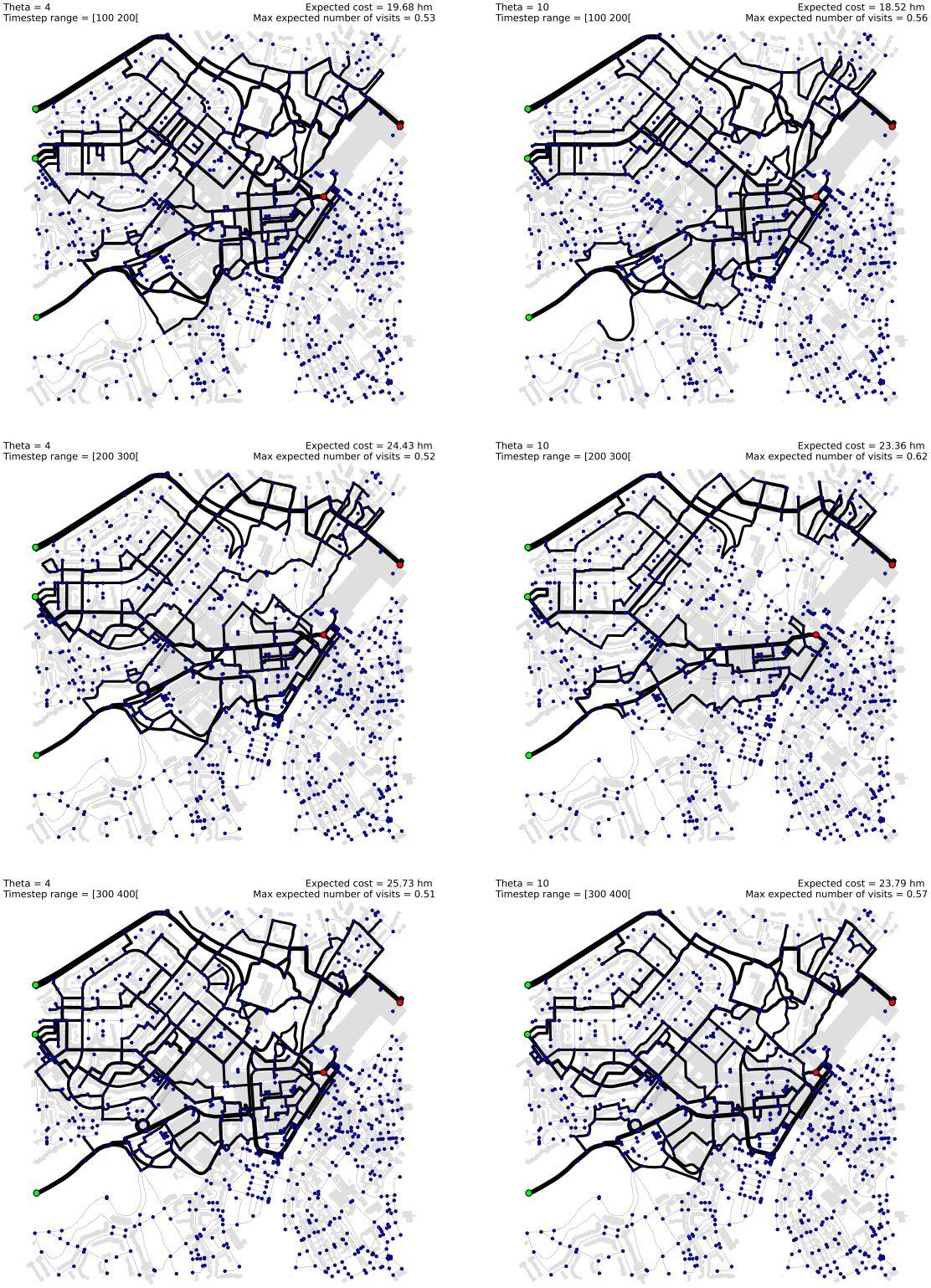

Figure 4.12: Representation of the number of visits after 100 agents have passed through the network following two KL-regularized randomized policies with varying degrees of entropy and during different periods of the simulation. It is computed in a context of stochastic transportation problem with multiple sources and targets.

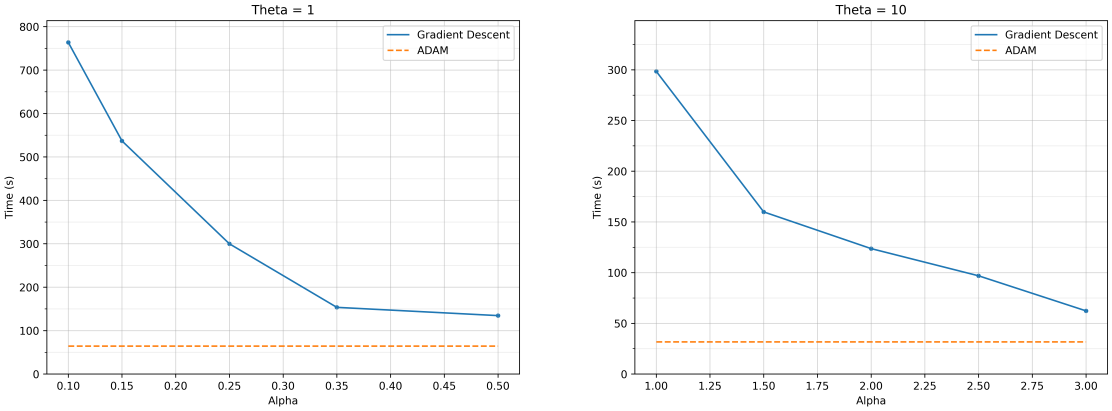

Figure 4.13: Comparison of performance of the Tsallis-regularized randomized MDP method considering the standard gradient descent update and the ADAM optimized update with constant tuned parameters.

4.3 Improving efficiency: ADAM optimizer

This section explains how we used ADAM optimizer algorithm (from [24]) to improve the performance of our method. Indeed, as mentionned in Subection 3.2, we had to use a gradient descent updating rule in order fix the margins around the outputs. It is relatively slow compared to other methods from the RSP context and it could serve to at least decrease the number of iterations. ADAM method computes an adaptive update step from the current gradient as well as gradients from the previous steps. A description is provided in Algorithm 3. It results in updating the associated parameters.

We hence did a little study on the efficiency of our method with Tsallis regularization. As the algorithm uses a gradient descent updating rule for both the inputs and ouputs, its global performance is even worse compared to the case of KL divergence which only involves the inputs. Note that the performance is already improved by the uses of more efficient methods for solving sparse systems of linear equations. We did these computations on the large road network introduced above, which is quite time expensive.

As suggested in Algorithm 3 presented below, in deep learning problems, using parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$ is generally a good choice. Since our method isn't really from the same context, it is a bit different. These parameters define the time scales over which learning rates decrease. If decay is very fast, learning rates are likely to be larger, resulting in larger movements. If they decay slowly, it will take longer for learning rates to be learned, but they will be more stable. In such a context, it's important when using stochastic gradient descent to smooth the path to the optimum by avoiding too many fluctuations. We can however be less demanding here.

Table 4.1 shows the performance results and associated parameters we obtained, for our algorithm and two values of θ. We also tested these parameters on several other values of θ in the range between 0.1 and 10. These sets of parameters were performing quite similarly in all case so that the results can be generalized to other

θ	α	β_1	β_2	ϵ	time (s)
1	1	0.3	0.999	10^{-8}	31.727
1	1	0.3	0.9	10^{-3}	24.887
10	0.2	0.4	0.999	10^{-8}	64.302
10	0.2	0.3	0.99	10^{-3}	51.201

Table 4.1: Performance of the Tsallis-regularized randomized MDP when using ADAM optimizer update with optimal parameters.

levels of entropy. The only parameter we had to modify was the stepsize α which needed to be decreased as we were getting closer to more deterministic policies. Indeed, as the cost term is given more importance in the objective function, it is also more reactive when updating the costs. It was therefore more demanding in terms of decay. Concerning other parameters, we kept similar values for β_2 because lower values tended to make the process diverge. The momentum parameter, β_1, was more flexible and we noticed that the most suited values were around $0.3 - 0.4$. The ϵ parameter is initially used to avoid dividing by zero while updating the parameters. However, we noticed that we could make the weight updates smaller by choosing a larger parameter. We indeed see that using a smaller value of β_2 was bifurcating rather at the end of the process. So using a larger value for ϵ allowed us to decrease β_2 to speed up the rest of the computations while limiting the weight updates (and adaptative behavior) near the optimum. In doing so, we got slightly better results.

Fig. 4.13 shows a small comparison of performance with the standard algorithm without optimization. The last stepsize value is the largest we could take without the process diverging. We can see that using our parameters cuts execution time by more than half, which can be worthwhile while doing many simulations. Note that these results can't be generalized to the KL-regularized algorithm. Indeed, as the objective function behaves differently, we can't be sure such parameters are adapted to such method. We also did these computations on our medium-sized road network. The obtained parameters hence can't be generalized neither to other too different graph structures.

Algorithm 3 ADAM optimizer, an adaptive step algorithm for stochastic optimization introduced in [24]. Good default settings suggested are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are elementwise.

Input:
- The stepsize α
- The exponential decay rates for the moment estimates β_1 and β_2
- The stochastic objective function $f(\theta)$
- The initial parameter vector θ_0

Output:
- The resulting parameters θ_t

$m_0 \leftarrow 0$ ▷ initialize the 1^{st} moment vector
$v_0 \leftarrow 0$ ▷ initialize the 2^{nd} moment vector
$t \leftarrow 0$ ▷ initialize the timestep
while θ_t not converged **do**
  $t \leftarrow t + 1$
  $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ ▷ get gradients of function f at timestep t
  $m_t \leftarrow \beta_1 \cdot m_{t_1} + (1 - \beta_1) \cdot g_t$ ▷ update biased 1^{st} moment estimate
  $v_t \leftarrow \beta_2 \cdot v_{t_1} + (1 - \beta_2) \cdot g_t^2$ ▷ update biased 2^{nd} moment estimate
  $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ ▷ correct the bias of 1^{st} moment estimate
  $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ ▷ correct the bias of 2^{nd} moment estimate
  $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ ▷ update the parameters
end while

Chapter 5

Conclusion

In this work, we introduced a new algorithm solving entropy-regularized shortest-paths stochastic problems on a graph with capacity constraint. This new model extends the work [7] which explore an alternative formulation of the Markov decision process in a randomized shortest-paths local framework [29]. It combines it with the results provided by papers [13, 12] which integrated flow capacity constraints to standard RSPs. This contribution hence expands the previous models to an even larger range of applications. It's done through the use of an extended bipartite graph structure which could adapt the ideas introduced in both contexts. The developed algorithm computes a policy at the edge level based on Lagrange duality which requires solving iteratively an MDP problem while updating the costs associated to the capacity constraints. This local formulation also allowed to extend the method to Tsallis divergence instead of the usual KL divergence. In the end, it provides a stochastic policy encouraging exploration, through some balance regarding exploitation. The amount of exploration is monitored by the inverse temperature parameter associated to the entropy term.

The usefulness of the algorithm is then illustrated on a medium-sized road network from a local neighborhood of Louvain-la-Neuve. The simulations demonstrated better results in comparison to the margin-constrained RSP method [12] when considering a stochastic environment. Concerning the exploration benefit, the simulations showed less convincing results when comparing several levels of randomization among different policies. Indeed, last example corresponding to a more realistic dynamic traffic context showed that more randomized behavior gave rise to higher expected costs. Even if the traffic is more spread out with increasing level of exploration, it doesn't really decrease the risks of congestion as the random behavior leads to more visits in general. Exploration seems to however be a good choice when considering local information as in the simpler static traffic example.

These results tend to be more on the theoretical side rather than relying heavily on real-world data and tendencies. We still found it interesting to apply our algorithm to such context for which we can give a clear interpretation. The model studied in this paper moreover considers a random environment which is not really observable in practice. The aim was rather to discover if such a general method could be useful in more theoretical situations. It's certainly not competitive in terms of performance with more optimized models developed in more specific contexts.

5.1 Limitations and future works

This work presents some limitations and we can underline some of them in a non-exhaustive way. First, as mentioned in Subsec. 2, the problem associated to the MDP reformulation has still not been formally proven convex with respect to the transition probabilities. The current conclusions are thus based on the global results of associated simulations so it could be interesting to prove it rigorously. We also couldn't adapt the approach used in [12] to fix the margins at the outputs. We thus used the gradient updating rule used [13] but at the cost of performance loss. It would be nice to look after a more optimized way to fix these margins in this MDP context.

Our simulations were restricted to a limited set of examples and contexts. It would be a good idea to extend our results to a larger range of problems, maybe more consistent with the use of stochastic environment. We would be able to generalize the results we already have and go further in the understanding. Non-stationary problems are especially interesting when doing exploration and we could imagine scenarios where even the margins are being modified over time. We could use the alternative Value Iteration algorithm from [7] that includes an entropy regularization term instead of the algorithm we used based on Lagrange parameters. It would allow to perform continual exploration as in paper [1].

As future works, we could explore other types of extension to the randomized shortest-path model, associated to other types of constraints. This model covers the handling of equality constraints on transition probabilities and inequality constraints on the edge flows. We could hence introduce the extension to inequality on the transition probabilities as well as inequality on the node flows. Such constraints could be adapted from a variety of real-world applications.

5.2 Skills acquired

Through the elaboration of this master thesis, I acquired new knowledge and deepened some existing ones among the contexts of randomized shortest-paths and Markov decision processes. I developed skills regarding the scientific process as the research, the writing, and the implementation, assessment of the simulations. This thesis also required working with Matlab which I never used. I also improved the diversity of my English language around these technical subjects.

Bibliography

[1] Youssef Achbany, François Fouss, Luh Yen, Alain Pirotte, and Marco Saerens. Tuning continual exploration in reinforcement learning: An optimality property of the boltzmann strategy. *Neurocomputing*, 71(13-15):2507–2520, 2008.

[2] Takashi Akamatsu. Cyclic flows, markov process and stochastic traffic assignment. *Transportation Research Part B: Methodological*, 30(5):369–386, 1996.

[3] Takashi Akamatsu. Decomposition of path choice entropy in general transport networks. *Transportation Science*, 31(4):349–362, 1997.

[4] José M Amigó, Sámuel G Balogh, and Sergio Hernández. A brief review of generalized entropies. *Entropy*, 20(11):813, 2018.

[5] Ferran Arqué, César A Uribe, and Carlos Ocampo-Martinez. Approximate wasserstein attraction flows for dynamic mass transport over networks. *Automatica*, 143:110432, 2022.

[6] Amir Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*. SIAM, 2014.

[7] Guillaume Guex Bertrand Lebichot, Pierre Leleux and Marco Saerens. Sparse randomized policies for markov decision processes based on tsallis divergence regularization. 2023. Submitted for publication.

[8] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena Scientific, 2012.

[9] Robert Goodell Brown. *Smoothing, forecasting and prediction of discrete time series*. Courier Corporation, 2004.

[10] Arthur Earl Bryson. *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[11] Ana Bušić and Sean Meyn. Action-constrained markov decision processes with kullback-leibler cost. In *Conference On Learning Theory (COLT)*, pages 1431–1444. PMLR, 2018.

[12] Sylvain Courtain, Guillaume Guex, Ilkka Kivimäki, and Marco Saerens. Relative entropy-regularized optimal transport on a graph: a new algorithm and an experimental comparison. *International Journal of Machine Learning and Cybernetics*, pages 1–26, 2022.

[13] Sylvain Courtain, Pierre Leleux, Ilkka Kivimäki, Guillaume Guex, and Marco Saerens. Randomized shortest paths with net flows and capacity constraints. *Information Sciences*, 556:341–360, 2021.

[14] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in Neural Information Processing Systems (NIPS)*, 26, 2013.

[15] Robert B Dial. A probabilistic multipath traffic assignment model which obviates path enumeration. *Transportation Research*, 5(2):83–111, 1971.

[16] Bertsekas DP. *Nonlinear Programm, 2nd edn.* Athena Scientific, 1999.

[17] Kevin Françoisse, Ilkka Kivimäki, Amin Mantrach, Fabrice Rossi, and Marco Saerens. A bag-of-paths framework for network data analysis. *Neural Networks*, 90:90–111, 2017.

[18] Guillaume Guex. Interpolating between random walks and optimal transportation routes: Flow with multiple sources and targets. *Physica A: Statistical Mechanics and its Applications*, 450:264–277, 2016.

[19] Guillaume Guex and François Bavaud. Flow-based dissimilarities: shortest path, commute time, max-flow and free energy. In *Data Science, Learning by Latent Structures, and Knowledge Discovery*, pages 101–111. Springer, 2015.

[20] Guillaume Guex, Ilkka Kivimäki, and Marco Saerens. Randomized optimal transport on a graph: framework and new distance measures. *Network Science*, 7(1):88–122, 2019.

[21] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical learning with sparsity: the lasso and generalizations*. CRC press, 2015.

[22] Edwin T Jaynes. Information theory and statistical mechanics. *Physical Review*, 106(4):620, 1957.

[23] Hilbert J Kappen. An introduction to stochastic control theory, path integrals and reinforcement learning. In *AIP Conference Proceedings*, volume 887, pages 149–181. American Institute of Physics, 2007.

[24] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Ilkka Kivimäki, Masashi Shimbo, and Marco Saerens. Developments in the theory of randomized shortest paths with a comparison of graph node distances. *Physica A: Statistical Mechanics and its Applications*, 393:600–616, 2014.

[26] Bertrand Lebichot. *Network analysis based on bag-of-paths: classification, node criticality and randomized policies.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2018.

[27] Kyungjae Lee, Sungjoon Choi, and Songhwai Oh. Sparse markov decision processes with causal sparse tsallis entropy regularization for reinforcement learning. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):1466–1473, 2018.

[28] Kyungjae Lee, Sungyub Kim, Sungbin Lim, Sungjoon Choi, and Songhwai Oh. Tsallis reinforcement learning: A unified framework for maximum entropy reinforcement learning. *arXiv preprint arXiv:1902.00137*, 2019.

[29] Pierre Leleux, Sylvain Courtain, Guillaume Guex, and Marco Saerens. Sparse randomized shortest paths routing with tsallis divergence regularization. *Data Mining and Knowledge Discovery*, 35:986–1031, 2021.

[30] Tomasz Maszczyk and Włodzisław Duch. Comparison of shannon, renyi and tsallis entropy used in decision trees. In *Artificial Intelligence and Soft Computing–ICAISC 2008: 9th International Conference Zakopane, Poland, June 22-26, 2008 Proceedings 9*, pages 643–651. Springer, 2008.

[31] P Neelakanta, J Kapur, and H Kesavan. Entropy optimization principles with applications, 1992.

[32] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017.

[33] M. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 1994.

[34] R Tyrell Rockafellar. The multiplier method of hestenes and powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[35] Marco Saerens, Youssef Achbany, François Fouss, and Luh Yen. Randomized shortest-path problems: Two related models. *Neural Computation*, 21(8):2363–2404, 2009.

[36] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[37] Evangelos Theodorou, D Krishnamurthy, and Emo Todorov. From information theoretic dualities to path integral and kullback-leibler control: Continuous and discrete time formulations. In *The Sixteenth Yale Workshop on Adaptive and Learning Systems*, volume 830, 2013.

[38] Evangelos A Theodorou and Emanuel Todorov. Relative entropy and free energy dualities: Connections to path integral and kl control. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, pages 1466–1473. IEEE, 2012.

[39] Emanuel Todorov. Linearly-solvable markov decision problems. *Advances in Neural Information Processing Systems (NIPS)*, 19, 2006.

[40] Constantino Tsallis. Generalized entropy-based criterion for consistent testing. *Physical Review E*, 58(2):1442, 1998.

[41] Wikipedia. Markov decision process. `http://en.wikipedia.org/w/index.php?title=Markov%20decision%20process&oldid=1156832827`, 2023. [Online; accessed 01-July-2023].

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**