Action Decomposable MDP A Linear Programming formulation for queuing network problems

Ariel Waserhole, Vincent Jost, Jean-Philippe Gayon

Action Decomposable MDP
A Linear Programming formulation for queuing network problems

Ariel Waserhole [1,2] Vincent Jost[2] Jean-Philippe Gayon[1]

[1] G-SCOP, Grenoble-INP [2] LIX CNRS, École Polytechnique Palaiseau

August 31, 2012

Abstract

Markov decision processes (MDP) have provided general frameworks for many control, decision making, and optimization stochastic problems. In this paper we are interested in a special class queuing control that can be model with a Continuous-Time MDP but that have an exponential actions state space with the classic methods. The Event Based Dynamic Programming approach deals about this problem and gives some algorithms (value iteration) to compute the best policy polynomially. However there is no formal definition on the subclass of MDP problem they can tackle. The first contribution of this paper to define this class, naming it "Action Decomposable MDP". The second contribution is to give a new MDP Linear Programming formulation using "Action Decomposability" that contributes to extend MDP solution techniques. Finally we give some examples of application of this framework and give numeric experiments showing the interest of using the action decomposition properties.

1 Introduction

1.1 Context

Markov decision processes (MDP) have provided general frameworks for many control, decision making, and optimization stochastic problems. In this paper we are interested in a special class queuing control that can be model with a Continuous-Time MDP. For instance, as schemed figure 1, for a single server and n classes of clients, if we want to set an entrance price p_i, and hence the arrival rate $\lambda(p_i)$ of customer i in order to optimize a given gain, we could model this problem with a Continuous-Time MDP. However with the classic MDP framework in each state of the process we would have to set the price of each class of clients giving possibly an exponential number of states of action.

In the literature, the Event Based Dynamic Programming approach deals about this problem and gives some algorithms to compute the best policy polynomially. However there is no formal definition on the subclass of MDP problem they can tackle. This is the first contribution of this paper to define this class, naming it "Action Decomposable MDP" or simply "Decomposable MDP" in the following. The second contribution is to give a new MDP Linear Programming formulation using 'Action Decomposability" that contributes to extend MDP solution techniques.

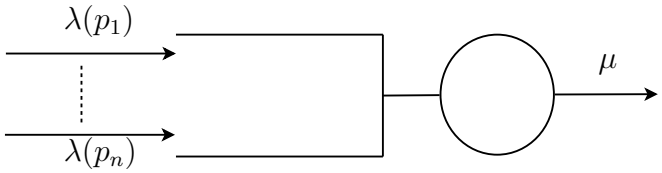

Figure 1: Queuing example.

1.2 Literature review

If the reader is not familiar with the notion of Markovian processes we refer him to the books of Puterman [13] or Bertsekas [2, 3]. For results more focused on Continuous-Time MDP, the book of Guo et al. [8] specialized on the question with notably a chapter really accurate on the Linear Programming Approach.

Bellman [1] first introduced Dynamic programming to solve Markov decision processes. The use of linear programming to solve dynamic programming formulations appeared later in Depenoux [5] and Manne [12]. Manne [12] studies an average reward Markov decision model with an infinite planning horizon. Depenoux [5] provides a linear program for the discounted version of the problem in [12] by linearizing the functional equations of the corresponding dynamic program. For the close relationship between Dynamic Programming and Linear Programming we refer to Byktahtakn [4].

Howard [9] combines dynamic programming with Markov chain theory to develop Markov decision processes. He also contributes to the solution of infinite horizon problems by developing the policy iteration method as an alternative to the backward induction method of Bellman [1], which is known as value iteration. The policy iteration algorithm generates a sequence of stationary policies by evaluating and improving the policies until the optimal policy is obtained. Again to have a good overview of the different classic techniques to sove MDP we refer to the books of Puterman [13] or Bertsekas [2, 3].

Event-based dynamic programming (dp) was first formulated in Koole [10]. Koole introduced event-based dp as a systematic approach for deriving monotonicity results of optimal policies for various queueing and resource sharing models. Citing himself " Event-based dp deals with event operators, which can be seen as building blocks of the value function. Typically it associates an operator with each basic event in the system, such as an arrival at a queue, a service completion, etc. Event-based dp focuses on the underlying properties of the value and cost functions, and allows us to study many models at the same time."

Later Koole [11] extended his theory to all kinds of monotonicity results applied to Stochastic scheduling problems that he defined as: A system with multiple customers classes having different service time distributions and needed to be assigned to one or more servers.

2 Decomposable Continuous Time Markov Decision Process

In this section we first try to give a rigorous definition of the sub class of Markov Decision Process where the Event Based Dynamic Programming can be apply. Then we study both the average reward and discounted reward criterion on infinite horizon. We give for each both the original and decomposed formulation of the equations of optimality and the Linear program corresponding. We finally discuss on the advantages to have a Linear Programming formulation.

2.1 Definitions

Let's first recall the classic CT-MDP formulation and give our definition of the Decomposable CT-MDP.

Definition 1 (Continuous Time Markov Decision Process, CT-MDP) *A Continuous Time Markov Decision Process is a stochastic control problem involving a finite state space S and a finite decision space $A(s)$ representing the actions available at each state s. When an action a is selected in state s, the systems get a stage reward $h_s(a)$ per unit of time and evolves to state t with an instant reward $r_{s,t}(a)$ after an exponentially distributed time with rate $\lambda_{s,t}(a)$. Future rewards might be discounted by a factor $\beta \in]0,1[$. The problem is to find the best policy which maximizes the discounted or average reward.*

Definition 2 (Decomposable Continuous Time Markov Decision Process , D-CT-MDP) *A Decomposable Continuous Time Markov Decision Process is a special class of CT-MDP when in each state $s \in S$, the action set $A(s)$ can be expressed as a Cartesian product of n_s independent subsets: $A(s) = A_1(s) \times \ldots \times A_{n_s}(s)$. When an action $a_i \in A_i(s)$ is selected in state s, the system get a stage reward $h_s^i(a_i)$ per unit of time and evolves to state t with an instant reward $r_{s,t}^i(a_i)$ after an exponentially distributed time with rate $\lambda_{s,t}^i(a_i)$. Future rewards might be discounted by a factor $\beta \in]0,1[$. In this paper we are looking at the problem of finding the best policy which maximizes the discounted or average reward .*

In the end, depending the action $a = (a_1, \ldots, a_{n_s})$ we choose in state s, the reward per unit of time staying in state s worth $h_s(a) = \sum_{i=1}^{n_s} h_s^i(a_i)$, the transition rate to go from state s to state t worth $\lambda_{s,t}(a) = \sum_{i=1}^{n_s} \lambda_{s,t}^i(a_i)$ with an instant reward $r_{s,t}(a) = \sum_{i=1}^{n_s} \frac{\lambda_{s,t}^i(a_i)}{\lambda_{s,t}(a)} r_{s,t}^i(a_i)$. Figure 2 gives a graphical representation of the independence of each subset of action $A_i(s)$. We could add holding cost, and

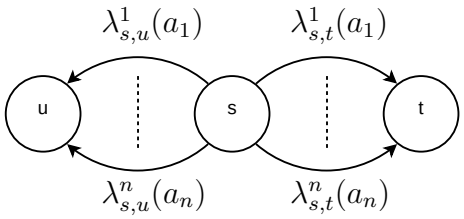

Figure 2: Graphical representation of a decomposable CT-MDP.

This new definition is subject to different remarks.

Remark 1 *The formula to compute $r_{s,t}(a)$ might seems complicated to compute but in fact usually we don't need to do it because problems are state decomposed in an intrinsic manner.*

Remark 2 *We could define a stage reward h'_s independent of the action chosen in state s. However to a have it dependent allows us to model for example a decision upon the usage of a certain type, numbers, of resources which change the transitions rates but also the stage reward (or cost) h.*

Remark 3 *We have in this definition rewards per transition and stage rewards. We could have only considered one or the other since by a small and easy transformation you can include stage reward in transition reward and reciprocally.*

We define some notions about the policies we can consider in such MDP.

Definition 3 (Deterministic policies) *A policy is said deterministic if for each state s we select always one action $a \in A(s)$ in case of CT-MDP, or $a_i \in A_i(s) \forall i = 1 \dots n_s$ in case of D-CT-MDP.*

Definition 4 (Randomized policies) *In CT-MDP, a policy is said randomized if for each state s we have a "discrete" probability $p(a|s)$ to select action $a \in A(s)$ with $\sum_{a \in A(s)} p(a|s) = 1$.*

Definition 5 (Decomposable randomized policies) *In D-CT-MDP, a policy is said decomposable randomized if for each state s we have a "discrete" probability $p^i(a_i|s)$ to select action $a_i \in A_i(s) \forall i \in [1 \dots n_s]$ with $\sum_{a_i \in A_i(s)} p^i(a_i|s) = 1, \ \forall i \in [1 \dots n_s]$..*

In the following, without loss of generality, we assume that each state $s \in S$ has $n_s = n$ subsets of independent actions $A_i(s)$ which contain each one exactly k actions. We will also only consider and speak about stationary policies, i.e. policies which take always for each state s the same decision, because they are trivially dominant since we are using exponential distribution with the memoryless property, c.f. [13].

2.2 Average Reward criterion

When decisions are made frequently, we can consider that there is no discount rate for future and therefore study the average reward criterion.

2.2.1 Bellman's equations

Recall that Bellman's equations of optimality are defined for Discrete Time MDP (DT-MDP). Therefore we have to make a "uniformization" by state to transform the CT-MDP into a DT-MDP.

To do so we use the following notations, let $\Lambda^i_{s,t} := \max_{a_i \in A_i(s)} \lambda^i_{s,t}(a_i)$, $\Lambda_{s,t} := \sum_{i=1}^n \Lambda^i_{s,t}$, $\Lambda^i_s := \sum_{t \in S} \Lambda^i_{s,t}$ and $\Lambda_s := \sum_{t \in S} \Lambda_{s,t}$, $\forall s,t \in S,\ \forall i \in [1..n]$.

We have that the optimal average reward g^* satisfies with some vector v the following Bellman's equations of optimality [1]:

$$\forall s \in S, \quad \frac{g}{\Lambda_s} + v(s) = T(v(s)). \tag{1}$$

With

$$\forall s \in S, \quad T(v(s)) = \max_{a \in A(s)} \left\{ \frac{1}{\Lambda_s} \left(h_s(a) + \sum_{t \in S} \left[\lambda_{s,t}(a) \times [v(t) + r_{s,t}(a)] + (\Lambda_{s,t} - \lambda_{s,t}(a)) \times v(s) \right] \right) \right\}$$

$$= \max_{\substack{(a_1,\ ...,\ a_n) \\ \in\ (A_1(s) \times ... \times A_n(s))}} \left\{ \frac{1}{\Lambda_s} \sum_{i=1}^{n} \left(h^i_s(a_i) + \sum_{t \in S} \left[\lambda^i_{s,t}(a_i) \times [v(t) + r^i_{s,t}(a_i)] + (\Lambda^i_{s,t} - \lambda^i_{s,t}(a_i)) \times v(s) \right] \right) \right\}. \tag{2}$$

And using that subset of actions A_i are independent, we can reduce the complexity by rewriting the equations (2) as follows $\forall s \in S$:

$$T(v(s)) = \frac{1}{\Lambda_s} \sum_{i=1}^{n} \left[\max_{a_i \in A_i(s)} \left\{ h^i_s(a_i) + \sum_{t \in S} \left[\lambda^i_{s,t}(a_i) \times [v(t) + r^i_{s,t}(a_i)] + (\Lambda^i_{s,t} - \lambda^i_{s,t}(a_i)) \times v(s) \right] \right\} \right]. \tag{3}$$

Finally to compute the best MDP policy we can now use the value iteration algorithm on Bellman's equations of optimality (1). We take operator T defined equations (2) or more efficiently (as we'll see in the section 3.2) on equations (3). Otherwise another method is to use a Linear Programming formulation given next section.

2.2.2 Classic CT-MDP Linear Program

From Manne [12] we know that it is possible to to compute the best MDP average reward policy through a Linear Program Indeed as it explained again in detail for instance in Puterman [13], chapter 8.8, page 391, we can construct a Linear Program from equations (2).

If the Weak Accessibility condition holds (see Bertsekas [3], Chapter 4.2, page 198), which is the case of an ergodic chain, then the Bellman's equations (1) have a solution and the optimal average reward g^* is independent from the initial state. Moreover g^* is the smallest g which satisfies with some vector v the following inequalities:

$$\frac{g}{\Lambda_s} + v(s) \geq T(v(s)), \quad \forall s \in S. \tag{4}$$

With equations (4), we can formulate the following linear program which has for solution g^*:

Classic Primal - Average Reward

$$\begin{aligned}
\min \quad & g \\
\text{s.t.} \quad & g \geq \sum_{t \in S} \left[\lambda_{s,t}(a) \times [v(t) + r_{s,t}(a)] + h_s(a) + (\Lambda_{s,t} - \lambda_{s,t}(a)) \times v(s)\right] - \Lambda_s \times v(s) \\
& \qquad\qquad \forall s \in S,\ \forall a \in A(s) = A_1(s) \times \ldots \times A_n(s) \\
& v(s) \in \mathbb{R} \qquad \forall s \in S \\
& g \in \mathbb{R}
\end{aligned}$$

Classic Dual - Average Reward

$$\begin{aligned}
\max \ & \sum_{s \in S} \sum_{a \in A} \pi_s(a) \times \left[h_s(a) + \sum_{t \in S} \lambda_{s,t}(a) \times r_{s,t}(a)\right] \\
\text{s.t.} \ & \sum_{a \in A(s)} \sum_{t \in S} \pi_s(a) \times \lambda_{s,t}(a) = \sum_{t \in S} \sum_{a \in A(t)} \pi_t(a) \times \lambda_{t,s}(a) \qquad \forall s \in S \\
& \sum_{s \in S} \sum_{a \in A(s)} \pi_s(a) = 1 \\
& \pi_s(a) \geq 0 \qquad \forall s \in S, \forall a \in A(s) = A_1(s) \times \ldots \times A_n(s)
\end{aligned}$$

An advantage of the dual formulation is that we can interpret the dual variables $\pi_s(a)$ as the average amount of time spent in state s choosing action a.

Note the the classical dual linear program formulation has $|S| \times k^n$ variables and $|S| \times k^n + |S| + 1$ constraints .

2.2.3 Decomposable CT-MDP Linear Program

Using the Event Based Programming approach we can formulate a new Linear Program. To do so we first formulate the next lemma.

Lemma 1 *For any finite sets $S,\ I,\ A$, any set of data coefficients $\gamma_{s,i,a} \in \mathbb{R}\ :\ s \in S,\ i \in I,\ a \in A$, the value*

$$g^* = \max_{s \in S} \left\{ \sum_{i \in I} \max_{a \in A} \{\gamma_{s,i,a}\} \right\}$$

is the solution of the following linear program:

$$\begin{array}{lll} \textit{minimize} & g & \\ \textit{subject to} & m(s,i) \geq \gamma_{s,i,a} & \forall s \in S,\ \forall i \in I,\ \forall a \in A \\ & g \geq \sum_{i \in I} m(s,i) & \forall s \in S \\ & m(s,i) \in \mathbb{R} & \forall s \in S,\ \forall i \in I \\ & g \in \mathbb{R} & \end{array}$$

Proof : Let g^* be an optimal solution of this linear program. It is trivial that $g^* \geq \max_{s \in S} \left\{ \sum_{i \in I} m(s,i) \right\}$, and since we are minimizing g without any other constraints on it, we have therefore $g^* = \max_{s \in S} \left\{ \sum_{i \in I} m(s,i) \right\}$.

For any optimal solution g^*, there exists $s' \in S$ such that $\sum_{i \in I} m(s',i) = \max_{s \in S} \left\{ \sum_{i \in I} m(s,i) \right\} = g^*$ and $\forall i \in I,\ m(s',i) = \max_{a \in A} \{\gamma_{s',i,a}\}$.

Indeed otherwise it would mean that for all $s' \in S$ such that $\sum_{i \in I} m(s',i) = \max_{s \in S} \left\{ \sum_{i \in I} m(s,i) \right\} = g^*$ there exists $i' \in I$ such that $m(s',i') > \max_{a \in A} \{\gamma_{s',i',a}\}$ and we would have a strictly better solution with $m(s',i') = \max_{a \in A} \{\gamma_{s',i',a}\}$.

In the end it gives us that $g^* = \max_{s \in S} \left\{ \sum_{i \in I} \max_{a \in A} \{\gamma_{s,i,a}\} \right\}$. □

Again if the Weak Accessibility condition holds then the Bellman's equation (1) has a solution and the optimal average reward g^* is solution of the following equations:

$$g^* = \min\ g \quad \text{such that} \quad \frac{g}{\Lambda_s} + v(s) \geq T(v(s)), \quad \forall s \in S, .$$

$$\Leftrightarrow$$

$$\begin{aligned} & g^* = \max_{s \in S} \{\Lambda_s \times (T(v(s)) - v(s))\} \\ & = \max_{s \in S} \left\{ \sum_{i=1}^{n} \left[\max_{a_i \in A_i(s)} \left\{ h_s^i(a_i) + \sum_{t \in S} \left[\lambda_{s,t}^i(a_i) \times [v(t) + r_{s,t}^i(a_i)] + (\Lambda_{s,t}^i - \lambda_{s,t}^i(a_i)) \times v(s) \right] - \Lambda_s^i \times v(s) \right\} \right] \right\} \\ & = \max_{s \in S} \left\{ \sum_{i=1}^{n} \left[\max_{a_i \in A_i(s)} \left\{ h_s^i(a_i) + \sum_{t \in S} \lambda_{s,t}^i(a_i) \times [v(t) - v(s) + r_{s,t}^i(a_i)] \right\} \right] \right\}. \end{aligned}$$

Thus from lemma 1, g^* is the solution of the following linear program:

Decomposable Primal - Average Reward

$$\min \quad g \tag{5a}$$

$$\text{s.t.} \quad m(s,i) \geq h_s^i(a_i) + \sum_{t \in S} \lambda_{s,t}^i(a_i) \times [v(t) - v(s) + r_{s,t}^i(a_i)] \quad \forall s \in S,\ \forall i \in [1..n],\ \forall a_i \in A_i(s) \tag{5b}$$

$$g \geq \sum_{i=1}^{n} m(s,i) \qquad \forall s \in S \tag{5c}$$

$$m(s,i) \in \mathbb{R} \qquad \forall s \in S,\ \forall i \in [1..n] \tag{5d}$$

$$v(s) \in \mathbb{R} \qquad \forall s \in S \tag{5e}$$

$$g \in \mathbb{R} \tag{5f}$$

Note that we can rewrite equations (5b) as follows to construct the dual more easily:

$$\forall s \in S, \quad \forall i \in [1..n], \quad \forall a_i \in A_i(s),$$
$$m(s,i) + \sum_{t \in S} \lambda_{s,t}^i(a_i) \times v(s) - \sum_{t \in S} \lambda_{s,t}^i(a_i) \times v(t) \geq h_s^i(a_i) + \sum_{t \in S} \lambda_{s,t}^i(a_i) \times r_{s,t}^i(a_i).$$

Decomposable Dual - Average Reward

$$\max \sum_{s \in S} \sum_{i=1}^{n} \sum_{a_i \in A_i} \pi_s^i(a_i) \times \left[h_s^i(a_i) + \sum_{t \in S} \lambda_{s,t}^i(a_i) \times r_{s,t}^i(a_i) \right] \tag{6a}$$

$$\text{s.t.} \sum_{i=1}^{n} \sum_{a_i \in A_i(s)} \pi_s^i(a_i) \times \sum_{t \in S} \lambda_{s,t}^i(a_i) = \sum_{t \in S} \sum_{i=1}^{n} \sum_{a_i \in A_i(t)} \pi_t^i(a_i) \times \lambda_{t,s}^i(a_i) \qquad \forall s \in S \tag{6b}$$

$$\sum_{a_i \in A_i(s)} \pi_s^i(a_i) = \pi_s \qquad \forall s \in S,\ \forall i \in [1..n] \tag{6c}$$

$$\sum_{s \in S} \pi_s = 1 \tag{6d}$$

$$\pi_s^i(a_i) \geq 0 \qquad \forall s \in S,\ \forall i \in [1..n],\ \forall a_i \in A_i(s) \tag{6e}$$

$$\pi_s \geq 0 \qquad \forall s \in S \tag{6f}$$

Again an advantage of the Dual formulation is that we can interpret π_s as the average amount of time spent in state s and $\pi_s^i(a_i)$ as the average amount of time spent in state s choosing action $a_i \in A_i(s)$ among all actions $A_i(s)$. This interpretation allows to see equations (6b) as the flow conservation constraints, and equations (6c) and (6d) as the Markov Chain Stationary distribution constraints.

Note that the decomposable dual linear program formulation has $|S|\times(k\times n+1)$ variables and $|S|\times((k+1)\times n+2)+1$ constraints.

Lemma 2 *Solutions of the decomposable linear program are decomposable randomized policies.*

Proof : In state s , the "discrete" probability $p^i(a_i|s)$ to choose action a_i out of all action $A_i(s)$ equals to $p^i(a_i|s)=\frac{\pi_s^i(a_i)}{\pi_s}$, and we have moreover $\sum_{a_i\in A_i(s)}p^i(a_i|s)=1$. □

Corollary 1 *Decomposable randomized policies are dominant among randomized policies since they are the solutions of the Decomposable Dual for Average Reward which provides the optimal policy. We have the probability $p(a|s)$ to choose action $a=(a_1,\ldots,a_n)$ in state s equal to $p(a|s)=\prod_{i=1}^n p^i(a_i|s)=\prod_{i=1}^n\frac{\pi_s^i(a_i)}{\pi_s}$. This gives us finally the value of Classic Dual variables $\pi_s(a)=\pi_s\times\prod_{i=1}^n\frac{\pi_s^i(a_i)}{\pi_s}$.*

Theorem 1 *The decomposable linear program solves the Decomposable Continuous Time Markov Decision Process and gives deterministic policies as optimal solutions.*

Proof : To prove this theorem we are going to study the dual version of the Average reward linear program which is more intuitive. We are going to show that dual equations define a polytope and that its vertices represent deterministic strategy solutions: i.e. $\forall s\in S,\ \forall i\in[1..n]$ there exists only one action selected a_j such that $\pi_s^i(a_j)\neq 0$ and $\forall a_i\in A_i(s)\setminus\{a_j\},\ \pi_s^i(a_i)=0$.

First, all variables of the problem are bounded: $\forall s\in S,\ \ 0\leq\pi_s\leq 1$ and $\forall s\in S,\ \forall i\in[1..n],\ \forall a_i\in A_i(s),\ 0\leq\pi_s^i(a_i)\leq 1$, therefore the space of feasible solutions is a bounded polyhedron, that is a polytope.

Secondly, we have $|S|\times n\times k+|S|$ variables, hence a vertex of the feasible polytope should satisfy at least $|S|\times n\times k+|S|$ constraints with equality (we say tight). Since we can remove one of the $|S|$ flow constraints of equations (6b) without changing the solution, we already have $|S|+|S|\times n$ equality constraints in our LP. Therefore only $|S|\times n\times(k-1)$ constraints still remain to be tight.

Assume now that $\forall s\in S,\ \pi_s\neq 0$, otherwise if $\exists s\in S$ with $\pi_s=0$ we could reduce our problem by removing state s without changing the optimal solution. Since $\forall s\in S,\ \pi_s\neq 0$ it means that we have at least one variable $\pi_s^i(a_i)$ in each of the $|S|\times n$ constraints (6c) not equal to 0. Hence $\forall s\in S,\ \forall i\in[1..n],\ \exists a_j\in A_i(s)$ such that $\pi_s^i(a_j)\neq 0$.

In the end we have to tight $|S|\times n\times(k-1)$ constraints (6e) out of $|S|\times n\times k$ with $|S|\times n$ different not tight. Therefore there exists only one action selected a_j such that $\pi_s^i(a_j)\neq 0$ and $\forall a_i\in A_i(s)\setminus\{a_j\},\ \ \pi_s^i(a_i)=0$ and a vertex of the solution polygon defines a deterministic policy. □

2.3 Discounted Reward criterion

We study now the discounted reward criterion, i.e. when future rewards are discounted by factor $\beta \in]0,1[$.

In this section in order to have a better interpretation of the formulation we will use a positive scalar α_s, $s \in S$ which satisfies $\sum_{s\in S} \alpha_s = 1$. Any other positive constant would work but the sum equals to 1 allows an interpretation as a initial state probability distribution over S.

2.3.1 Bellman's equations

Again to use Bellman's equation of optimality defined on Discrete Time MDP (DT-MDP), we use an uniformization by state to transform our CT-MDP into a DT-MDP.

To do so we use the following notations, let $\Lambda^i_{s,t} := \max_{a_i \in A_i(s)} \lambda^i_{s,t}(a_i)$, $\Lambda_{s,t} := \sum_{i=1}^n \Lambda^i_{s,t}$, $\Lambda^i_s := \sum_{t\in S} \Lambda^i_{s,t}$ and $\Lambda_s := \sum_{t\in S} \Lambda_{s,t}$, $\forall s \in S,\ \forall i \in [1..n]$. We also define $\beta^i = \frac{\beta}{n},\ \forall i \in [1..n]$

We have then that the optimal expected discounted reward per state v^* satisfies the following Bellman's equations of optimality [1]:

$$\forall s \in S, \quad v(s) = T(v(s)). \tag{7}$$

With

$$\begin{aligned}
\forall s \in S, \quad T(v(s)) &= \max_{a\in A(s)} \left\{ \frac{1}{\beta + \Lambda_s} \left(h_s(a) + \sum_{t\in S} \left[\lambda_{s,t}(a) \times (v(t) + r_{s,t}(a)) + (\Lambda_{s,t} - \lambda_{s,t}(a)) \times v(s) \right] \right) \right\} \\
&= \max_{\substack{(a_1,\ ...,\ a_n) \\ \in\ (A_1(s)\times ... \times A_n(s))}} \left\{ \sum_{i=1}^n \left[\frac{1}{\beta + \Lambda_s} \left(h^i_s(a_i) + \sum_{t\in S} \left[\lambda^i_{s,t}(a_i) \times (v(t) + r^i_{s,t}(a_i)) + (\Lambda^i_{s,t} - \lambda^i_{s,t}(a_i)) \times v(s) \right] \right) \right] \right\}
\end{aligned} \tag{8}$$

And using that subset of actions A_i are independent, we can reduce the complexity by rewriting the equations (8) as follows $\forall s \in S$:

$$T(v(s)) = \sum_{i=1}^n \left[\max_{a_i \in A_i(s)} \left\{ \frac{1}{\beta + \Lambda_s} \left(h^i_s(a_i) + \sum_{t\in S} \left[\lambda^i_{s,t}(a_i) \times (v(t) + r^i_{s,t}(a_i)) + (\Lambda^i_{s,t} - \lambda^i_{s,t}(a_i)) \times v(s) \right] \right) \right\} \right]. \tag{9}$$

Finally to compute the best MDP policy we can now use the value iteration algorithm on Bellman's equations of optimality (7). We take operator T defined equations (8) or more efficiently (as we'll see in the section 3.2) on equations (9). Otherwise another method is to use a Linear Programming formulation given next section.

2.3.2 Classic CT-MDP Linear Program

From D'epenoux [5] we know that it is possible to to compute the best MDP discounted reward policy with a Linear Program. As it is well explained in Puterman [13], chapter 6.9, page 223, we can construct a Linear Program from equations (8).

Again if the Weak Accessibility condition holds (see Bertsekas [3], Chapter 4.2, page 198)), which is the case of an ergodic chain, then the Bellman's equation (7) has a solution and the optimal expected reward per state v^* is the vector v with the smallest value $\sum_{s\in S} \alpha_s \times v(s)$ which satisfies:

$$v(s) \geq T(v(s)), \quad \forall s \in S. \tag{10}$$

With equations (10), we can formulate the following linear program which has for solution v^*:

Classic Primal - Discounted Reward

$$\begin{aligned}
&\min \sum_{s\in S} \alpha_s \times v(s) \\
&\text{s.t. } (\beta + \sum_{t\in S} \lambda_{s,t}(a)) \times v(s) - \sum_{t\in S} \lambda_{s,t}(a) \times v(t) \geq h_s(a) + \sum_{t\in S} \lambda_{s,t}(a) \times r_{s,t}(a) \quad \forall s \in S,\ \forall a \in A(s) \\
&\quad v(s) \in \mathbb{R} \qquad\qquad \forall s \in S
\end{aligned}$$

Classic Dual - Discounted Reward

$$\begin{aligned}
&\max \sum_{s\in S} \sum_{a\in A(s)} \widetilde{\pi}_s(a) \times \left(h_s(a) + \sum_{t\in S} \lambda_{s,t}(a) \times r_{s,t}(a) \right) \\
&\text{s.t. } \sum_{a\in A(s)} (\beta + \sum_{t\in S} \lambda_{s,t}(a)) \times \widetilde{\pi}_s(a) - \sum_{t\in S} \sum_{a\in A(t)} \lambda_{t,s}(a) \times \widetilde{\pi}_t(a) = \alpha_s \qquad \forall s \in S \\
&\quad \widetilde{\pi}_s(a) \geq 0 \qquad \forall s \in S,\ \forall a \in A(s) = A_1(s) \times \ldots \times A_n(s)
\end{aligned}$$

Again an advantage of the dual formulation is that we can interpret the dual variables $\widetilde{\pi}_s(a)$ as the total discounted joint probability under initial state distribution α_s that the system occupies state $s \in S$ and chooses action $a \in A(s)$. Some other interpretations can be retrieved in Puterman [13] p.226.

Note that the classic dual linear program has $|S|\times k^n$ variables and $|S|\times(k^n + 1)$ constraints.

2.3.3 Decomposable CT-MDP Linear Program

Using Event Based Programming approach we can construct a new Linear Program. To do so we first state the next lemma.

Lemma 3 *For any finite sets $S,\ I,\ A$, any set of data coefficients $\alpha_s,\ \gamma_{s,t,i,a},\ \delta_{s,i,a} \in \mathbb{R}\ :\ s,t \in S,\ i \in I,\ a \in A$, the vector $v \in \mathbb{R}^{|S|}$ with the smallest value $\sum_{s\in S} \alpha_s \times v(s)$ satisfying:*

$$0 \geq \max_{s\in S}\left\{\sum_{i\in I}\max_{a\in A}\left\{\sum_{t\in S}\gamma_{s,t,i,a}\times v(s)+\delta_{s,i,a}\right\}\right\}$$

is the solution of the following linear program:

$$\begin{array}{lll}
\textit{minimize} & \displaystyle\sum_{s\in S}\alpha_s \times v(s) & \\
\textit{subject to} & m(s,i) \geq \displaystyle\sum_{t\in S}\gamma_{s,t,i,a}\times v(s)+\delta_{s,i,a} & \forall s\in S,\ \forall i\in I,\ \forall a\in A \\
& 0 \geq \displaystyle\sum_{i\in I} m(s,i) & \forall s\in S \\
& m(s,i)\in\mathbb{R} & \forall s\in S,\ \forall i\in I \\
& v(s)\in\mathbb{R} & \forall s\in S
\end{array}$$

Proof : It is clear that we are minimizing $\sum_{s\in S}\alpha_s \times v(s)$. Now for any vector v we can see that $\forall s \in S,\ \forall i \in I\ m(s,i) \geq \max_{a\in A}\left\{\sum_{t\in S}\gamma_{s,t,i,a}\times v(s)+\delta_{s,i,a}\right\}$, and we have $\forall s \in S\sum_{i\in I} m(s,i) \leq 0$. Therefore any vector v solution must satisfy $0 \geq \sum_{i\in I}\max_{a\in A}\left\{\sum_{t\in S}\gamma_{s,t,i,a}\times v(s)+\delta_{s,i,a}\right.$ S and therefore $0 \geq \max_{s\in S}\left\{\sum_{i\in I}\max_{a\in A}\left\{\sum_{t\in S}\gamma_{s,t,i,a}\times v(s)+\delta_{s,i,a}\right\}\right\}$. □

Again if the Weak Accessibility condition holds then Bellman's equations (7) have a solution and the optimal expected reward per state v^* is the solution of the following equations:

$$\min\sum_{s\in S}\alpha_s \times v(s)$$

Such that:

$$v(s) \geq T(v(s)),\quad \forall s\in S.$$

$$\Leftrightarrow$$

$$\begin{aligned}
0 &\geq \max_{s\in S}\{T(v(s))-v(s)\} \\
&\geq \max_{s\in S}\{(\beta+\Lambda_s)\times(T(v(s))-v(s))\} \\
&\geq \max_{s\in S}\left\{\sum_{i=1}^{n}\left[\max_{a_i\in A_i(s)}\left\{h_s^i(a_i)\right.\right.\right. \\
&\quad \left.\left.\left.+\sum_{t\in S}\left[\lambda_{s,t}^i(a_i)\times[v(t)+r_{s,t}^i(a_i)]+(\Lambda_{s,t}^i-\lambda_{s,t}^i(a_i))\times v(s)\right]-(\Lambda_s^i+\beta^i)\times v(s)\right\}\right]\right\} \\
&\geq \max_{s\in S}\left\{\sum_{i=1}^{n}\left[\max_{a_i\in A_i(s)}\left\{h_s^i(a_i)+\sum_{t\in S}\lambda_{s,t}^i(a_i)\times[v(t)-v(s)+r_{s,t}^i(a_i)]-\beta^i\times v(s)\right\}\right]\right\}.
\end{aligned}$$

Thus from lemma 3, v^* is the solution of the following linear program.

Decomposable Primal - Discounted Reward

$$\min \sum_{s\in S} \alpha_s \times v(s) \tag{11a}$$

$$\text{s.t. } m(s,i) \geq h_s^i(a_i) + \sum_{t\in S} \lambda_{s,t}^i(a_i) \times [v(t) - v(s) + r_{s,t}^i(a_i)] - \beta^i \times v(s) \tag{11b}$$

$$\forall s \in S,\ \forall i \in [1..n],\ \forall a_i \in A_i(s) \tag{11c}$$

$$\sum_{i=1}^{n} m(s,i) \leq 0 \qquad \forall s \in S \leq 0 \tag{11d}$$

$$m(s,i) \in \mathbb{R} \qquad \forall s \in S,\ \forall i \in [1..n] \tag{11e}$$

$$v(s) \in \mathbb{R} \qquad \forall s \in S \tag{11f}$$

To construct easily the dual equations (11c) can be rewritten:

$$\forall s \in S, \quad \forall i \in [1..n], \quad \forall a_i \in A_i(s),$$
$$m(s,i) + \left(\beta_i + \sum_{t\in S} \lambda_{s,t}^i(a_i)\right) \times v(s) - \sum_{t\in S} \lambda_{s,t}^i(a_i) \times v(t) \geq h_s^i(a_i) + \sum_{t\in S} \lambda_{s,t}^i(a_i) \times r_{s,t}^i(a_i).$$

Decomposable Dual - Discounted Reward

$$\max \sum_{s\in S} \sum_{i=1}^{n} \sum_{a_i \in A_i(s)} \widetilde{\pi}_s^i(a_i) \times \left(h_s^i(a_i) + \sum_{t\in S} \lambda_{s,t}^i(a_i) \times r_{s,t}^i(a_i)\right) \tag{12a}$$

$$\text{s.t. } \sum_{a_i \in A_i(s)} \widetilde{\pi}_s^i(a_i) = \widetilde{\pi}_s \qquad \forall s \in S,\ \forall i \in [1..n] \tag{12b}$$

$$\sum_{i=1}^{n} \sum_{a_i \in A_i(s)} \left(\beta^i + \sum_{t\in S} \lambda_{s,t}^i(a_i)\right) \times \widetilde{\pi}_s^i(a_i) - \sum_{t\in S} \sum_{i=1}^{n} \sum_{a_i \in A_i(t)} \lambda_{t,s}^i(a_i) \times \widetilde{\pi}_t^i(a_i) = \alpha_s \qquad \forall s \in S \tag{12c}$$

$$\widetilde{\pi}_s^i(a_i) \geq 0 \qquad \forall s \in S,\ \forall i \in [1..n],\ \forall a_i \in A_i(s) \tag{12d}$$

$$\widetilde{\pi}_s \geq 0 \qquad \forall s \in S \tag{12e}$$

We can interpret $\widetilde{\pi}_s$ as the total discounted joint probability under initial state distribution α_s that the system occupies state $s \in S$. While $\widetilde{\pi}_s^i(a_i)$ represents the total discounted joint

probability under initial state distribution α_s that the system occupies state $s \in S$ and choose action $a_i \in A_i(s)$ out of $A_i(s)$.

Note that the decomposable dual linear program has $|S|\times(k\times n+1)$ variables and $|S|\times((k+1)\times n+2)$ constraints.

Lemma 4 *Solutions of the decomposable linear program are decomposable randomized policies.*

Proof : In state s , the "discrete" probability $p^i(a_i|s)$ to choose action a_i out of all action $A_i(s)$ equals to $p^i(a_i|s) = \frac{\widetilde{\pi}^i_s(a_i)}{\widetilde{\pi}_s}$, and we have moreover $\sum_{a_i\in A_i(s)} p^i(a_i|s) = 1$. □

Corollary 2 *Decomposable randomized policies are dominant over randomized policies since they are the solutions of the Decomposable Dual for Discounted Reward which gives the optimal policy. We have the probability $p(a|s)$ to choose action $a = (a_1, \ldots, a_n)$ in state s equal to $p(a|s) = \prod_{i=1}^n p^i(a_i|s) = \prod_{i=1}^n \frac{\widetilde{\pi}^i_s(a_i)}{\widetilde{\pi}_s}$. This gives us finally the value of Classic Dual variables $\widetilde{\pi}_s(a) = \widetilde{\pi}_s \times \prod_{i=1}^n \frac{\widetilde{\pi}^i_s(a_i)}{\widetilde{\pi}_s}$.*

Theorem 2 *Optimal solutions of the decomposable linear program are deterministic policies.*

Proof : To prove this theorem we are going to study the dual version of the Discounted Reward linear program which is more intuitive. We are going to show that dual equations define a polyedron and that its vertices represent deterministic strategy solutions: i.e. $\forall s \in S,\ \forall i \in [1..n]$ there exists only one action selected a_j such that $\widetilde{\pi}^i_s(a_j) \neq 0$ and $\forall a_i \in A_i(s) \setminus \{a_j\},\ \widetilde{\pi}^i_s(a_i) = 0$.

First, all variables of the problem are bounded: $\forall s \in S,\ \ 0 \leq \widetilde{\pi}_s \leq 1$ and $\forall s \in S,\ \forall i \in [1..n],\ \forall a_i \in A_i(s),\ \ 0 \leq \widetilde{\pi}^i_s(a_i) \leq 1$, therefore the space of feasible solutions is a bounded polyhedron, that is a polytope.

Secondly, we have $|S|\times n \times k + |S|$ variables, hence a vertex of the feasible polytope should at least tight $|S|\times n \times k + |S|$ constraints. We have already $|S|+|S|\times n$ equality constraints in our LP. Hence $|S|\times n \times (k-1)$ constraints still remain to be tight.

Now assume that $\forall s \in S,\ \widetilde{\pi}_s \neq 0$, otherwise if $\exists s \in S$ such that $\widetilde{\pi}_s = 0$ we could reduce our problem by removing state s without changing the solution. Since $\forall s \in S,\ \widetilde{\pi}_s \neq 0$ it means that we have at least one variable $\widetilde{\pi}^i_s(a_i)$ in each of the $|S|\times n$ constraints (12b) not equal to 0. Hence $\forall s \in S,\ \forall i \in [1..n],\ \exists a_j \in A_i(s)\ /\ \widetilde{\pi}^i_s(a_j) \neq 0$.

In the end we have to tight $|S|\times n \times (k-1)$ constraints (12d) out of $|S|\times n \times k$ with $|S|\times n$ different not tight. Therefore there exists only one action selected a_j such that $\widetilde{\pi}^i_s(a_j) \neq 0$ and $\forall a_i \in A_i(s)\setminus\{a_j\},\ \widetilde{\pi}^i_s(a_i) = 0$ and a vertex of the solution polygon defines a deterministic policies.

□

2.4 Advantages of a LP formulation against DP

First let recall that this particular Linear Programming formulation for the Decomposable CT-MDP, which uses the paradigm of event based dynamic programming allows us to have a complexity polynomial in the number of independent subsets of actions: $|S|\times(k \times n + 1)$ variables and $|S|\times((k + 1) \times n + 2)$ constraints; Whereas the classical linear programming formulation grows exponentially: $|S|\times k^n$ variables and $|S|\times(k^n + 1)$ constraints. This has a huge impact on the computational results given in the next section.

Secondly even if the Linear Programming formulation is slower to solve than a Dynamic Programming one, as is it said in Puterman [13] and showed again in a following section, this Mathematical Programming approach offers some advantages:

- The dual Linear Program formulation is really simple to write and does not need the uniformization necessary to the DP which is sometimes source of waste of time and error.
- Linear Program formulation can help to characterize the polyhedral structure of discrete optimization problems, c.f. [4].
- It is possible to use sensitive analysis methods of Linear Programming, c.f. [7] for a good survey.
- We can also use Approximate Linear Programming techniques, c.f. [6] for details.
- Moreover we have the ability to add extra constraints in the linear program formulation which is impossible in the Dynamic Program. Examples are given next section.

2.4.1 Average Reward criterion constraints

Stationary distribution constraints We can constrain the stationary distribution to force a quality of service q on a subset $T \subset S$ of state:

$$\sum_{s \in T} \pi_s \geq q.$$

Nevertheless we have to be aware that such constraint can have for optimal strictly randomized (mixed) policies.

More action selection constraints

Lemma 5 *The following equation force in a state $s \in S$ to select at least m and at most M actions in a set of k actions a_i^j belonging to different subsets of independent actions, $a_i^j \in A_i(s),\ j \in [1..k]$:*

$$m \leq \sum_{j=1}^{k} \frac{\pi_s^i(a_i^j)}{\pi_s} \leq M.$$

Proof : We only have to see that $\frac{\pi_s^i(a_i^j)}{\pi_s}$ is the discrete probability to take in state s action a_i^j out of all actions $A_i(s)$. See lemma 2. □

Remark 4 *We have to be aware that such constraints may lead to optimal solutions that are strictly randomized policies. In this case to ensure with a probability* 1 *to have at least* m *and at most* M *actions you would need to generate a posteriori the randomized policy sequence of action to take.*

2.4.2 Discounted Reward constraints

State value constraints Constraints on the $v(s)$. Example?

Lemma 6 *The following equation force in a state* $s \in S$ *to a select at least* m *and at most* M *actions in a set of* k *actions* a_i^j *belonging to different subsets of independent actions,* $a_i^j \in A_i(s),\ j \in [1..k]$*:*

$$m \leq \sum_{j=1}^{k} \frac{\widetilde{\pi}_s^i(a_i^j)}{\widetilde{\pi}_s} \leq M.$$

Proof : We only have to see that $\frac{\widetilde{\pi}_s^i(a_i^j)}{\widetilde{\pi}_s}$ is the discrete probability to take in state s action a_i^j out of all actions $A_i(s)$. See lemma 4. □

Remark 5 *As in remark 4, we have to be aware that such constraints may lead to optimal solutions that are strictly randomized policies.*

3 Examples

3.1 A Linear Programming Model with additional constraints

A finite number n of the same resource need to be shared into different class of clients. Assume it is a bumper car rental (or short term bicycle rental) and we have 3 client classes: the children c, the student s and the adult a arriving at rate $\lambda_c(p_c^k)$ resp. $\lambda_s(p_s^k)$ and $\lambda_a(p_a^k)$, with $p_i^k, k = 1 \ldots 3$ the 3 discrete ordered increasing prices to rent a resource at class i. All rentals are for the same amount of time following an exponential distribution of mean $\frac{1}{\mu}$.

The states S of the system are the number of vehicles rented. For each state we want to define the price to rent one vehicle for each class of client.

To attract clients there need to have enough vehicles playing on the playground so people would have fun. So we constraint the system to be two third of the time with more than half of

the vehicles rented. We can express this constraint as follows

$$\sum_{s=n/2}^{n} \pi_s \geq \frac{2}{3}.$$

Moreover the manager can't rent a vehicle to a student at a price higher that an adult, otherwise the student will pay the adult price! So if the price p_a^1 is lower than price p_s^2 and p_s^3 in each state we have to put the constraint to select at most one price:

$$\forall s \in S, \quad \frac{\pi_s(p_s^2)}{\pi_s} + \frac{\pi_s(p_s^3)}{\pi_s} + \frac{\pi_s(p_a^1)}{\pi_s} \leq 1.$$

In the end we want to maximize the average revenue.

3.2 Numeric experimentation

In this section we are considering a decomposable CT-MDP problem and look at the efficiency (in term of resolution time) between the linear programming formulation and the dynamic formulation with the classic and decomposable approach.

3.2.1 Model

Let's consider a single server and n different client classes. We have a finite queue (buffer) of size C for each class. The system state space is $S = \{(x_1, \ldots, x_n) \mid x_i < C, \ \forall i \in [1, n]\}$, and we have therefore $|S| = C^n$ states.

For each state$s$ of the system we want to decide

- A price for each client of class i to enter into the system: $r(a_i)$.
  - It is model by the selection of an action a_i out of all actions A_i with $|A_i| = k$.
  - Clients of class i will then arrive in the system following a Poisson distribution with rate $\lambda_i(a_i)$ which depends on the price $r(a_i)$ chosen.
- The class of client to process d
  - It is model by the selection of an action $d \in D$ which elect the class d to process out of all classes with $|D| = n$.
  - Clients of class i have a process time μ_i by the server.
  - The waiting time cost per unit of time for a client of class i worth h_i.

In the end the action space for each state is $A = (A_1 \times \ldots \times A_n) \times D$.

3.2.2 Bellman equations

To solve this problem with the value iteration algorithm we have to write down the Bellman equation of optimality. To do so, to uniformize the CT-MDP into a discrete one we'll need the following notations: let $\Lambda_i = \max_{a_i \in A_i} \lambda_i(a_i)$, $\Lambda = \sum_{i=1}^{n} \Lambda_i$ and $\Delta = \max_{i \in [1 \dots n]} \{\mu_i\}$. More over let the operator $\{a\}^b := a$ if the boolean expression $b = true$, 0 otherwise.

For a state $s = (x_1, \dots, x_n)$, the classic approach the operator T can be defined as follows:

$$T(v(s)) = \frac{1}{\Lambda + \Delta} \max_{(a_1, \dots, a_n, d) \in A} \left\{ \sum_{i=1}^{n} \left[x_i \times h_i + \{\lambda_i(a_i) \times (v(s + e_i) + r(a_i))\}^{x_i < C} \right] \right.$$
$$\left. + \{\mu_d \times v(s - e_d)\}^{x_d > 0} + (\Lambda - \sum_{i=1}^{n} \{\lambda_i(a_i)\}^{x_i < C} + \Delta - \{\mu_d\}^{x_d > 0}) \times v(s) \right\}.$$

For the decomposable approach the operator T can be defined as follows:

$$T(v(s)) = \frac{1}{\Lambda + \Delta} \left(\sum_{i=1}^{n} \left[x_i \times h_i + \max_{\substack{a_i \in A_i \\ x_i < C}} \{\lambda_i(a_i) \times (v(s + e_i) + r(a_i)) + (\Lambda_i - \lambda_i(a_i)) \times v(s)\} \right] \right.$$
$$\left. + \max_{\substack{d \in D \\ x_d > 0}} \{\mu_d \times v(s - e_d) + (\Delta - \mu_d) \times v(s)\} \right).$$

3.2.3 Linear program

Classic With $a = (a_1, \dots, a_n, d) \in A$, we can write the classical Linear Programming:

$$\max \sum_{s \in S} \sum_{a \in A} \pi_s(a) \times \sum_{i=1}^{n} \left[x_i \times h_i + \{\lambda_i(a_i) \times r_i(a_i)\}^{x_i < C} \right]$$
$$\text{s.t.} \sum_{a \in A} \pi_s(a) \left[\{\mu_d\}^{x_d > 0} + \sum_{i=1}^{n} \{\lambda_i(a_i)\}^{x_i < C} \right] = \sum_{i=1}^{n} \sum_{a \in A} \left[\{\pi_{s - e_i}(a) \times \lambda_i(a_i)\}^{x_i > 0} + \{\pi_{s + e_d}(a) \times \mu_d\}^{x_d < C} \right]$$
$$\sum_{s \in S} \sum_{a \in A} \pi_s(a) = 1$$
$$\pi_s(a) \geq 0 \qquad \forall s \in$$

Decomposed Using what we show in section 2.2.3 we can write the decomposable CT-MDP as follows:

$$\max \sum_{s\in S}\sum_{i=1}^{n}\left[\pi_s \times x_i \times h_i + \sum_{a_i\in A_i}\{\pi_s^i(a_i)\times\lambda_i(a_i)\times r_i(a_i)\}^{x_i<C}\right]$$

$$\text{s.t. } \sum_{i=1}^{n}\left[\sum_{a_i\in A_i}\{\pi_s^i(a_i)\times\lambda_i(a_i)\}^{x_i<C} + \{\pi_s(d)\times\mu_d\}^{x_d>0}\right]$$

$$= \sum_{i=1}^{n}\left[\sum_{a_i\in A_i(t)}\{\pi_{s-e_i}^i(a_i)\times\lambda_i(a_i)\}^{x_i>0} + \{\pi_{s+e_d}(d)\times\mu_d\}^{x_d<C}\right] \qquad \forall s\in S$$

$$\sum_{a_i\in A_i}\pi_s^i(a_i) = \pi_s^i \qquad \forall s\in S,\ \forall i\in[1..n]$$

$$\sum_{d\in D}\pi_s(d) = \pi_s \qquad \forall s\in S,\ \forall i\in[1..n]$$

$$\sum_{s\in S}\pi_s = 1$$

$$\pi_s^i(a_i)\geq 0 \qquad \forall s\in S,\ \forall i\in[1..n],\ \forall a_i\in A_i(s)$$

$$\pi_s(d)\geq 0 \qquad \forall s\in S,\ \forall d\in[1..n],$$

$$\pi_s\geq 0 \qquad \forall s\in S$$

3.2.4 Results

We create instances with n classes of client with k prices. A customer of class $i\in[1..n]$ with price $j\in[1..k]$ generates a reward $r_i(a_j)=j\times 2$, arrives with transition rate $\lambda_i(a_j)=(4-i)\times(10-r_i(a_j))$, cost $h_i=-2^{4-i}$ and can have a processing time $\mu_i=20-4\times i$. More over there is always the possibility to refuse a client, $\forall i\in[1..n]$, $r_i(0)=0$ and $\lambda_i(0)=0$.

Algorithm are tested on a Intel core 2 duo 2.4 Ghz processor with 2 Go of ram. Heuristics are written in JAVA and Linear Programs solved with Gurobi 4.6. Legend (F-M) has to be read as follow. F for Formulation: either C for Classic or D for Decomposable. M for Method: either VI-ϵ for Value Iteration at the given tolerance ϵ or LP for Linear Programming.

We want to compare the computation time of the different algorithms on a same instance. We are looking specifically at the influence of the capacity of the queue C, table 1, the number of classes n, table 2, and the number of prices proposed, table 3.

We have confronted 6 solution methods: the Value Iteration algorithm Decomposed or Classic for two values of ϵ: 10^{-3} and 10^{-5} and the classic and decomposed Linear Program.

First for classic formulation as for decomposed one, we expected the Value Iteration computation time depends on the precision asked: diminishing per 100 ϵ increase in the order of 2

the time. We also clearly see that the computation time of value iteration algorithm is from far much smaller than the Linear Program formulation. In general roughly a ratio 3 for a precision $\epsilon = 10^{-3}$ and 6 for a precision $\epsilon = 10^{-3}$.

Secondly the benefice of the decomposition appears obviously. When the number of states are growing, variation on the queue capacity C (table 1) or the number of classes n (table 2) influence less the decomposable formulation. And it is even more clear when we increase the number of proposed prices k (table 3), indeed on this table we can see that the difference of computation time between the classic and decomposed formulation increases exponentially with the number of prices.

(C,n,k)	C-VI-10^{-2}	D-VI-10^{-2}	C-VI-10^{-5}	D-VI-10^{-5}	C-LP	D-LP
(5,3,4)	0.79	0.09	2.16	0.71	4.94	0.27
(10,3,4)	18.02	1.05	34.89	1.70	101.8	7.08
(15,3,4)	82.71	4.24	143.2	7.22	4244	290

Table 1: Influence of capacity C on the algorithms computation time (in s.).

(C,n,k)	C-VI-10^{-2}	D-VI-10^{-2}	C-VI-10^{-5}	D-VI-10^{-5}	C-LP	D-LP
(10,1,4)	0	0	0	0	0.03	0.03
(10,2,4)	0.07	0.01	0.08	0.02	0.36	0.13
(10,3,4)	18.02	1.05	34.89	1.70	101.8	7.08
(5,4,4)	87.9	0.89	383	3.56	541.7	3.72

Table 2: Influence of the number of classes n on the algorithms computation time (in s.).

(C,n,k)	C-VI-10^{-2}	D-VI-10^{-2}	C-VI-10^{-5}	D-VI-10^{-5}	C-LP	D-LP
(10,3,2)	2.51	0.45	2.67	0.54	7.1	1.1
(10,3,3)	6.12	0.55	10.60	0.81	20.4	2.6
(10,3,4)	18.02	1.05	34.89	1.70	101.8	7.08
(10,3,5)	37.56	1.2	80.23	2.03	331	9.7

Table 3: Influence of the number of prices k on the algorithms computation time (in s.).

References

[1] R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 1957.

[2] D. P. Bertsekas. *Dynamic programming and optimal control. Vol. I.* 3rd ed., Athena Scientific, Belmont, MA, 2005.

[3] D. P. Bertsekas. *Dynamic programming and optimal control. Vol. II.* 3rd ed., Athena Scientific, Belmont, MA, 2005.

[4] Î. E. Byktahtakn. Dynamic programming via linear programming. *Wiley Encyclopedia of Operations Research and Management Science to appear*, 2011.

[5] F. Depenoux. A probabilistic production and inventory problem. *Management Science*, 1963, Translation of an article published in Revue Franaise de Recherche Operationnelle 14, 1960.

[6] V. L. Dos Santos Eleutrio. *Finding Approximate Solutions for Large Scale Linear Programs*. PhD thesis, ETH Zurich, 2009.

[7] C Filippi. *Sensitivity Analysis in Linear Programming*. Wiley Encyclopedia of Operations Research and Management Science, 2011.

[8] X. Guo and O. Hernández-Lerma. *Continuous-Time Markov Decision Processes: Theory and Applications*. Stochastic modelling and applied probability. Springer Verlag, 2009.

[9] R. A. Howard. Dynamic programming and markov processes. *M.I.T. Press, Cambridge, Massachusetts*, 1960.

[10] G. M. Koole. Structural results for the control of queueing systems using event-based dynamic programming. *Queueing Systems*, 1998.

[11] G. M. Koole. Stochastic scheduling with event-based dynamic programming. *Mathematical Methods of Operations Research*, 2000.

[12] A. S Manne. Linear programming and sequential decisions. *Operations Research*, 1960.

[13] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, New York, NY, 1994.

[14] A. Lippman S. Applying a new device in the optimization of exponential queuing systems. *Operation Research*, 1975.