Dynamic resource allocation problems in communication networks:

Introduction and the Finite Horizon Restless Bandit problem

Alexandre Reiffers-Masson

Equipe Maths&Net, IMT Atlantique, CS department
LabSTICC (UMR CNRS 6285)

July 9, 2023

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Lab-STICC

Acknowledgement

Thanks the CNI center (Centre for Networked Intelligence) and IIsc to have invited me. Huge thanks to Parimal, Preetam.

This course has been also elaborate during the project Ramonaas[1] (Regional Program STIC-AmSud):

- a STIC/AMSUD project between CAPES/BR (88881.694462/2022-01);
- Ministry for Europe and Foreign Affairs/FR;
- Campus France/FR and the National Agency for Research;
- Innovation/UY (MOV_CO_2022_1_1011515)

[1] Resource Allocation Methods for Optical Networks as a Service

IMT Atlantique : -Engineering institution under the tutelage of the French Ministry of Industry
- 3 campuses North-West of France
- Member of the Institut Mines-Télécom

Track MLA:
***where?* on Brest campus**

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

when?
from 30 march to 2nd june 2023

What do in Brest?

Maths& Net

Goals: The Maths&Net team aims to design, describe, manage, secure and control various aspects of networks, in particular telecommunications networks. The team also works on other types of networks such as distributed ledgers or social networks.

Permanent members

- Prof. Sandrine Vaton (head of the team)
- Prof. Francoise Sailhan
- Associate Prof. Isabel Amigo
- Associate Prof. Alexandre Reiffers-Masson

Non-permanent members: Robin Duraz (Ph.D.), Sanaa Ghandi (Ph.D.), Ziad Tlaiss (Ph.D.), Colin Troisemaine (Ph.D.), Stanislas Mareschal de Charentenay (Ph.D.), Hajer Rejeb (Postdoc), Olivier Tsemogne Kamguia (Postdoc).

Agenda of the course

Part I: Provably efficient heuristics for solving large-scale resource allocation problems.

- **Day I:** Introduction to resource allocation problem and MDP and Restless Bandit in Finite Horizon.
- **Day II:** Weakly coupled MDP and the resolving heuristic.
- **Day III:** Constrained Finite Horizon Stochastic Optimization Problems.

Part II: Machine Learning for Resource Allocation Problems

- **Day IV:** Online and Distributed algorithm for Resource Allocation Problem with unknown parameters.
- **Day V:** Deep-learning applied to Resource Allocation Problems.

Objectives of the course

- Provably efficient heuristics for solving large-scale resource allocation problems
    1. Design heuristics and prove asymptotically optimal properties.
    2. Code the heuristic in python using *cvxpy*.

- Machine Learning for Resource Allocation Problems
    1. Learn fundamental theoretical results to study the performance of Gradient-descent type of algorithms.
    2. Learn recent theoretical results to study the performance of Neural Networks (expressive power of neural networks, convergence and generalization).
    3. Code Neural Networks and DRL.

Machine Maintenance[2]

- **Scenario:** A collection of N machines which deteriorate under usage is maintained by a set of α repairmen. Maintenance interventions will improve a machine's condition and may preempt costly breakdowns.

[2] Glazebrook, K. D., Mitchell, H. M., & Ansell, P. S. (2005). Index policies for the maintenance of a collection of machines by a set of repairmen. European Journal of Operational Research, 165(1), 267-284.

Machine Maintenance[2]

- **Scenario:** A collection of N machines which deteriorate under usage is maintained by a set of α repairmen. Maintenance interventions will improve a machine's condition and may preempt costly breakdowns.

- **Challenge:** How to *allocate* repairmen at each instant t, *knowing* the state of each machine, *to minimize the total expected deterioration of machines*.

[2] Glazebrook, K. D., Mitchell, H. M., & Ansell, P. S. (2005). Index policies for the maintenance of a collection of machines by a set of repairmen. European Journal of Operational Research, 165(1), 267-284.

Machine Maintenance[2]

- **Scenario:** A collection of N machines which deteriorate under usage is maintained by a set of α repairmen. Maintenance interventions will improve a machine's condition and may preempt costly breakdowns.

- **Challenge:** How to *allocate* repairmen at each instant t, *knowing* the state of each machine, *to minimize the total expected deterioration of machines*.

[2] Glazebrook, K. D., Mitchell, H. M., & Ansell, P. S. (2005). Index policies for the maintenance of a collection of machines by a set of repairmen. European Journal of Operational Research, 165(1), 267-284.

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \ldots, S\}$ be the state of the machine of the machine k.

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \ldots, S\}$ be the state of the machine of the machine k.
- The greater the value of $S_k(t)$, the worse the machine k performs.

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \ldots, S\}$ be the state of the machine of the machine k.
- The greater the value of $S_k(t)$, the worse the machine k performs.
- **Remark:** Usually the evolution of the state of machine k is modelled by a *controlled Markov chain*.

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \ldots, S\}$ be the state of the machine of the machine k.
- The greater the value of $S_k(t)$, the worse the machine k performs.
- **Remark:** Usually the evolution of the state of machine k is modelled by a *controlled Markov chain*.
- At each instant $t = 0, \ldots, N-1$ a repairman is checking the machine k (action $a = 1$) or no one is checking (action $a = 0$). We can assume that

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \dots, S\}$ be the state of the machine of the machine k.
- The greater the value of $S_k(t)$, the worse the machine k performs.
- **Remark:** Usually the evolution of the state of machine k is modelled by a *controlled Markov chain*.
- At each instant $t = 0, \dots, N-1$ a repairman is checking the machine k (action $a = 1$) or no one is checking (action $a = 0$). We can assume that

$$\mathbb{P}(S(t+1) = 0 | S(t) = s, A(t) = 1) \quad = \quad 1$$

Model: Evolution of the state of a machine

- Let $S_k(t) \in \{1, \ldots, S\}$ be the state of the machine of the machine k.
- The greater the value of $S_k(t)$, the worse the machine k performs.
- **Remark:** Usually the evolution of the state of machine k is modelled by a *controlled Markov chain*.
- At each instant $t = 0, \ldots, N-1$ a repairman is checking the machine k (action $a = 1$) or no one is checking (action $a = 0$). We can assume that

$$\begin{array}{rcl} \mathbb{P}(S(t+1) = 0 | S(t) = s, A(t) = 1) & = & 1 \\ \mathbb{P}(S(t+1) = s' | S(t) = s, A(t) = 0) & = & I\{s' \geq s\} p_{ss'} \end{array} \quad (1)$$

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. ''Deadline scheduling as restless bandits.'' IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.
  2. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs.

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.
  2. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs.

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.
  2. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs.
- **Reward and cost:**

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.
  2. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs.
- **Reward and cost:**
  1. The charging station obtains a reward of $1 - c$ for each unit of electricity provides.

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Deadline Scheduling (for charging electric vehicles)[3]

- **Set-up:** Charging station has N charging spots and enough power to charge M vehicles at each round.
- **System dynamic:**
  1. When a charging spot is available, a new vehicle may join the system and occupy the spot.
  2. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs.
- **Reward and cost:**
  1. The charging station obtains a reward of $1 - c$ for each unit of electricity provides.
  2. If the station cannot fully charge the vehicle by the time it leaves, the station needs to pay a penalty proportional to the amount of unfulfilled charge.

---

[3] Yu, Zhe, Yunjian Xu, and Lang Tong. "Deadline scheduling as restless bandits." IEEE Transactions on Automatic Control 63, no. 8 (2018): 2343-2358.

Model

States: Let $T_k(t) = d_k - t$ be the lead time to deadline d_k and $B_k(t)$ be the amount of electricity. The state of the k-th spot in defined as

Model

States: Let $T_k(t) = d_k - t$ be the lead time to deadline d_k and $B_k(t)$ be the amount of electricity. The state of the k-th spot in defined as

$$S_k(t) = \begin{cases} (0,0) & \text{if no job is at the } k\text{-th position,} \\ (T_k(t), B_k(t)) & \text{otherwise.} \end{cases}$$

Model

States: Let $T_k(t) = d_k - t$ be the lead time to deadline d_k and $B_k(t)$ be the amount of electricity. The state of the k-th spot in defined as

$$S_k(t) = \begin{cases} (0,0) & \text{if no job is at the } k\text{-th position,} \\ (T_k(t), B_k(t)) & \text{otherwise.} \end{cases}$$

Evolution: The (Markovian) evolution of the state is given by:

Model

States: Let $T_k(t) = d_k - t$ be the lead time to deadline d_k and $B_k(t)$ be the amount of electricity. The state of the k-th spot in defined as

$$S_k(t) = \begin{cases} (0,0) & \text{if no job is at the } k\text{-th position,} \\ (T_k(t), B_k(t)) & \text{otherwise.} \end{cases}$$

Evolution: The (Markovian) evolution of the state is given by:

$$S_k(t+1) = \begin{cases} (T_k(t) - 1, [B_k(t) - a_k(t)]_+) & \text{if } T_k(t) > 1, \\ (T, B) \text{ with prob. } Q(T, B) & \text{otherwise,} \end{cases}$$

where $a_k(t)$ is the amount of electricity given to spot k at instant t.

Other applications

- Wireless Communication;
- Web Crawling;
- Congestion Control;
- Queuing Systems;
- Cluster and Cloud computing;
- Target Tracking;
- Clinical Trials.

A quick recall on Markov chain

Definition

A *Markov Chain* consists of a finite set $\mathcal{S}$ (called the state space) together with a countable family of random variables $S(0), S(1), S(2), \ldots$ with values in $\mathcal{S}$ such that for all $t \geq 0$:

A quick recall on Markov chain

Definition

A *Markov Chain* consists of a finite set $\mathcal{S}$ (called the state space) together with a countable family of random variables $S(0), S(1), S(2), \ldots$ with values in $\mathcal{S}$ such that for all $t \geq 0$:

$$\mathbb{P}(S(t+1) = s' | S(0) = s_0, \ldots, S(t) = s_t) = \mathbb{P}(S(t+1) = s' | S(t) = s_t), \tag{2}$$

A quick recall on Markov chain

Definition

A *Markov Chain* consists of a finite set $\mathcal{S}$ (called the state space) together with a countable family of random variables $S(0), S(1), S(2), \ldots$ with values in $\mathcal{S}$ such that for all $t \geq 0$:

$$\mathbb{P}(S(t+1) = s' | S(0) = s_0, \ldots, S(t) = s_t) = \mathbb{P}(S(t+1) = s' | S(t) = s_t), \tag{2}$$

- We refer to this fundamental equation as the *Markov property*.

A quick recall on Markov chain

Definition

A *Markov Chain* consists of a finite set $\mathcal{S}$ (called the state space) together with a countable family of random variables $S(0), S(1), S(2), \dots$ with values in $\mathcal{S}$ such that for all $t \geq 0$:

$$\mathbb{P}(S(t+1) = s' | S(0) = s_0, \dots, S(t) = s_t) = \mathbb{P}(S(t+1) = s' | S(t) = s_t), \tag{2}$$

- We refer to this fundamental equation as the *Markov property*.
- We set for every $s, s' \in \mathcal{S}$:

$$p_{ss'} := \mathbb{P}(S(t+1) = s' | S(t) = s).$$

A quick recall on Markov chain

Definition

A *Markov Chain* consists of a finite set $\mathcal{S}$ (called the state space) together with a countable family of random variables $S(0), S(1), S(2), \ldots$ with values in $\mathcal{S}$ such that for all $t \geq 0$:

$$\mathbb{P}(S(t+1) = s' | S(0) = s_0, \ldots, S(t) = s_t) = \mathbb{P}(S(t+1) = s' | S(t) = s_t), \tag{2}$$

- We refer to this fundamental equation as the *Markov property*.
- We set for every $s, s' \in \mathcal{S}$:

$$p_{ss'} := \mathbb{P}(S(t+1) = s' | S(t) = s).$$

The matrix $P := [[p_{ss'}]]_{s,s'}$ is called the *transition matrix*.

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

- A finite state space $\mathcal{S} := \{1, \ldots, d\}$,

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

- A finite state space $\mathcal{S} := \{1, \ldots, d\}$,
- A finite action space $\mathcal{A} := \{0, 1\}$,

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

- A finite state space $\mathcal{S} := \{1, \ldots, d\}$,
- A finite action space $\mathcal{A} := \{0, 1\}$,

and where:

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

- A finite state space $\mathcal{S} := \{1, \ldots, d\}$,
- A finite action space $\mathcal{A} := \{0, 1\}$,

and where:

- $S(t) \in \mathcal{S}$ is the state of the bandit at the discrete decision time $t \in \{0, \cdots, T\}$,

One arm Restless Bandit

A **One Arm Restless Bandit** is a MDP defined on:

- A finite state space $\mathcal{S} := \{1, \ldots, d\}$,
- A finite action space $\mathcal{A} := \{0, 1\}$,

and where:

- $S(t) \in \mathcal{S}$ is the state of the bandit at the discrete decision time $t \in \{0, \cdots, T\}$,
- $A(t) \in \mathcal{A}$ is the action taken by the decision maker at the discrete decision time $t \in \{0, \cdots, T\}$.

One arm Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

One arm Restless Bandit

For each time-step $t = 0, \ldots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;

One arm Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) \in \mathcal{A}$;

One arm Restless Bandit

For each time-step $t = 0, \dots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) \in \mathcal{A}$;
3. The decision-maker collects the reward $r_{S(t)}^{A(t)}$;

One arm Restless Bandit

For each time-step $t = 0, \dots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) \in \mathcal{A}$;
3. The decision-maker collects the reward $r^{A(t)}_{S(t)}$;
4. The Markov process evolves to $S(t+1) = s'$ with probability $p^{A(t)}_{S(t),s'}$.

One arm Restless Bandit

For each time-step $t = 0, \ldots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) \in \mathcal{A}$;
3. The decision-maker collects the reward $r_{S(t)}^{A(t)}$;
4. The Markov process evolves to $S(t+1) = s'$ with probability $p_{S(t),s'}^{A(t)}$.

Objective: Maximize the expected total sum of rewards over the T time-steps.

One arm Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) \in \mathcal{S}$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) \in \mathcal{A}$;
3. The decision-maker collects the reward $r_{S(t)}^{A(t)}$;
4. The Markov process evolves to $S(t+1) = s'$ with probability $p_{S(t),s'}^{A(t)}$.

Objective: Maximize the expected total sum of rewards over the T time-steps.

´ **Knowns parameters:** $\mathcal{S}$, $\mathcal{A}$, reward $R := [[r_s^a]]_{s,a}$, Horizon T, transition matrix $P^a := [[p_{s,s'}^a]]_{s,s'}$.

Control Policies

Definition

- A **general control policy** π is a mapping from each *possible history* $h_t = \{s_\tau, a_\tau\}_{0 \le \tau \le t}$ to $a_t = \pi_t(h_t)$.

Control Policies

Definition

- A **general control policy** π is a mapping from each *possible history* $h_t = \{s_\tau, a_\tau\}_{0 \le \tau \le t}$ to $a_t = \pi_t(h_t)$.
- A **Markov control policy** π depends on the current state and time only: $a_t = \pi_t(s_t)$.

Control Policies

Definition

- A **general control policy** π is a mapping from each *possible history* $h_t = \{s_\tau, a_\tau\}_{0 \leq \tau \leq t}$ to $a_t = \pi_t(h_t)$.
- A **Markov control policy** π depends on the current state and time only: $a_t = \pi_t(s_t)$.
- A **randomized control policy**, the action a_t is chosen according to a probability distribution $\pi_t(a|h_t)$ over A_t .

Control Policies

Definition

- A **general control policy** π is a mapping from each *possible history* $h_t = \{s_\tau, a_\tau\}_{0 \leq \tau \leq t}$ to $a_t = \pi_t(h_t)$.
- A **Markov control policy** π depends on the current state and time only: $a_t = \pi_t(s_t)$.
- A **randomized control policy**, the action a_t is chosen according to a probability distribution $\pi_t(a|h_t)$ over A_t .

Remarks:

1. Randomized Markov policies suffice to achieve the optimum in classical constrained MDP. *In this course, we will only such policies.*

Control Policies

Definition

- A **general control policy** π is a mapping from each *possible history* $h_t = \{s_\tau, a_\tau\}_{0 \le \tau \le t}$ to $a_t = \pi_t(h_t)$.
- A **Markov control policy** π depends on the current state and time only: $a_t = \pi_t(s_t)$.
- A **randomized control policy**, the action a_t is chosen according to a probability distribution $\pi_t(a|h_t)$ over A_t .

Remarks:

1. Randomized Markov policies suffice to achieve the optimum in classical constrained MDP. *In this course, we will only such policies.*
2. When a randomized Markov policy π_t is used, the probability that the Markov process evolves to $S(t+1) = s'$ and action $A(t) = a$, knowing $S(t) = s$ is given by $p^a_{s,s'}\pi^a(s)$.

Mathematical formulation of the problem

For a given randomized Markov policy $\pi := [\pi_t]_{0 \le t \le T-1}$, we define the cumulative reward:

Mathematical formulation of the problem

For a given randomized Markov policy $\pi := [\pi_t]_{0 \leq t \leq T-1}$, we define the cumulative reward:

$$V_1^{\pi}(x^0, T) := \mathbb{E}_{x^0}\left[\sum_{t=0}^{T-1} R_{S_t}^{A_t}\right].$$

Mathematical formulation of the problem

For a given randomized Markov policy $\pi := [\pi_t]_{0 \le t \le T-1}$, we define the cumulative reward:

$$V_1^{\pi}(x^0, T) := \mathbb{E}_{x^0}\left[\sum_{t=0}^{T-1} R_{S_t}^{A_t}\right].$$

The **Value function** is given by:

$$V_1^*(x^0, T) = \min_{\pi} V_1^{\pi}(x^0, T)$$

LP formulation

Let us define the following LP problem:

$$
\begin{aligned}
\min_{y \geq 0} \quad & \sum_{t=0}^{T-1} \sum_{s,a} R_s^a y_{a,s}(t) \\
\text{s.t.} \quad & y_{s,0}(t) + y_{s,1}(t) = x_s(t),\ \forall t \in [[0, T-1]],\ \forall s \in \mathcal{S}, \\
& x_s(t) = \sum_{s'} \sum_{a} y_{s',a}(t-1) p_{s',s}^a \ \forall t \in [[1, T-1]],\ \forall s \in \mathcal{S}, \\
& x_s(0) = x^0,\ \forall s \in \mathcal{S}
\end{aligned}
\tag{3}
$$

Equivalence

Lemma: Let y^* be a solution of (3). If for all $0 \leq t \leq T-1$, for all $s \in \mathcal{S}$ and for all $a \in \mathcal{A}$, we define

$$\pi_t(a|s) = \begin{cases} y_s^a(t)/x_s^a(t), & \text{if } x_s^a(t) > 0, \\ 0 & \text{otherwise,} \end{cases}$$

then

$$V_1(x^0, T) = V_1^\pi(x^0, T).$$

N-Arms Restless Bandit

A **N-arms Restless Bandit** is composed of N statistically MDPs where:

N-Arms Restless Bandit

A **N-arms Restless Bandit** is composed of N statistically MDPs where:

- $S_k(t) \in \mathcal{S}$ is the state of the arm k at the discrete decision time $t \in \{0, \cdots, T\}$,

N-Arms Restless Bandit

A **N-arms Restless Bandit** is composed of N statistically MDPs where:

- $S_k(t) \in \mathcal{S}$ is the state of the arm k at the discrete decision time $t \in \{0, \cdots, T\}$,
- $A_k(t) \in \mathcal{A}$ is the action taken by the decision maker at the discrete decision time $t \in \{0, \cdots, T\}$.

N-Arms Restless Bandit

A **N-arms Restless Bandit** is composed of N statistically MDPs where:

- $S_k(t) \in \mathcal{S}$ is the state of the arm k at the discrete decision time $t \in \{0, \cdots, T\}$,
- $A_k(t) \in \mathcal{A}$ is the action taken by the decision maker at the discrete decision time $t \in \{0, \cdots, T\}$.
- We assume that the decision maker chooses a fraction $0 < \alpha < 1$ of the N arms to be activated.

N-Arms Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

N-Arms Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \ldots, S_N(t)] \in \mathcal{S}^N$;

N-Arms Restless Bandit

For each time-step $t = 0, \ldots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \ldots, S_N(t)] \in \mathcal{S}^N$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) := [A_1(t), \ldots, A_N(t)] \in \mathcal{A}^N$, such that $\sum_k A_k(t) \leq N\alpha$;

N-Arms Restless Bandit

For each time-step $t = 0, \ldots, T-1$*:*

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \ldots, S_N(t)] \in \mathcal{S}^N$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) := [A_1(t), \ldots, A_N(t)] \in \mathcal{A}^N$, such that $\sum_k A_k(t) \leq N\alpha$;
3. The decision-maker collects the reward $\sum_k r_{S_k(t)}^{A_k(t)}$;

N-Arms Restless Bandit

For each time-step $t = 0, \dots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \dots, S_N(t)] \in \mathcal{S}^N$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) := [A_1(t), \dots, A_N(t)] \in \mathcal{A}^N$, such that $\sum_k A_k(t) \leq N\alpha$;
3. The decision-maker collects the reward $\sum_k r_{S_k(t)}^{A_k(t)}$;
4. For every k, the arm k evolves to $S_k(t+1) = s'$ with probability $p_{S_k(t),s'}^{A_k(t)}$.

N-Arms Restless Bandit

For each time-step $t = 0, \ldots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \ldots, S_N(t)] \in \mathcal{S}^N$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) := [A_1(t), \ldots, A_N(t)] \in \mathcal{A}^N$, such that $\sum_k A_k(t) \leq N\alpha$;
3. The decision-maker collects the reward $\sum_k r^{A_k(t)}_{S_k(t)}$;
4. For every k, the arm k evolves to $S_k(t+1) = s'$ with probability $p^{A_k(t)}_{S_k(t),s'}$.

Objective: Maximize the expected total sum of rewards over the T time-steps.

N-Arms Restless Bandit

For each time-step $t = 0, \dots, T-1$:

1. The decision-maker gets full knowledge of the current system state $S(t) := [S_1(t), \dots, S_N(t)] \in \mathcal{S}^N$;
2. Once $S(t)$ has been observed, the decision-maker chooses a control $A(t) := [A_1(t), \dots, A_N(t)] \in \mathcal{A}^N$, such that $\sum_k A_k(t) \leq N\alpha$;
3. The decision-maker collects the reward $\sum_k r_{S_k(t)}^{A_k(t)}$;
4. For every k, the arm k evolves to $S_k(t+1) = s'$ with probability $p_{S_k(t),s'}^{A_k(t)}$.

Objective: Maximize the expected total sum of rewards over the T time-steps.

´ **Knowns parameters:** $\mathcal{S}$, $\mathcal{A}$, reward $R := [[r_s^a]]_{s,a}$, Horizon T, transition matrix $P^a := [[p_{s,s'}^a]]_{s,s'}$.

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

- **Arms are exchangeable.** Two arms in the same state and for which the same action is chosen provide the *same reward* and have the *same transition probabilities*.

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

- **Arms are exchangeable.** Two arms in the same state and for which the same action is chosen provide the *same reward* and have the *same transition probabilities*.
- **Implication:** It is equivalent to use a control on $S(t)$ or on the *empirical distribution of states* at every instant t.

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

- **Arms are exchangeable.** Two arms in the same state and for which the same action is chosen provide the *same reward* and have the *same transition probabilities*.
- **Implication:** It is equivalent to use a control on $S(t)$ or on the *empirical distribution of states* at every instant t.

New notations:

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

- **Arms are exchangeable.** Two arms in the same state and for which the same action is chosen provide the *same reward* and have the *same transition probabilities*.
- **Implication:** It is equivalent to use a control on $S(t)$ or on the *empirical distribution of states* at every instant t.

New notations:

- $M_s^{(N)}(t) :=$ the fraction of arms in state s at time t.
  $M^{(N)}(t) := [M_s^{(N)}(t)]_{s \in \mathcal{S}}$ is the associated vector.

A new state representation

To simplify the mathematical formulation of the problem, we make the following observation:

- **Arms are exchangeable.** Two arms in the same state and for which the same action is chosen provide the *same reward* and have the *same transition probabilities*.
- **Implication:** It is equivalent to use a control on $S(t)$ or on the *empirical distribution of states* at every instant t.

New notations:

- $M_s^{(N)}(t) :=$ the fraction of arms in state s at time t. $M^{(N)}(t) := [M_s^{(N)}(t)]_{s\in\mathcal{S}}$ is the associated vector.
- $Y_{s,a}^{(N)}(t) :=$ the fraction of arms in state s at time t for which decision a is taken. $Y^{(N)}(t) := [Y_{s,a}^{(N)}(t)]_{s\in\mathcal{S},a\in\mathcal{A}}$ is the associated vector.

Mathematical Formulation

$$\min_{\pi} \quad \sum_{t=0}^{T-1} \mathbb{E} \sum_{s,a} r_s^a Y_{a,s}^{(N)}(t) := V_{opt}^{(N)}(m(0), T) \tag{4a}$$

s.t. Arms follow the Markovian evolution generated by $\Pi_n p_{s_n, s_n'}^{a_n}$, (4b)

$$Y_{0,s}^{(N)}(t) + Y_{1,s}^{(N)}(t) = M_s^{(N)}(t),\ \forall t \in [[0, T-1]],\ \forall s \in \mathcal{S}, \tag{4c}$$

$$\sum_{s} Y_{s,a}^{(N)}(t) \leq \alpha\ \forall t \in [[0, T-1]],, \tag{4d}$$

$$M_s^{(N)}(0) = m_s(0),\ \forall s \in \mathcal{S}, \tag{4e}$$

where $m_s(0) = \frac{1}{N} \sum_{k=1}^{N} I\{S_k(0) = s\}$, for all $s \in \mathcal{S}$.

Difficulty

The key difficulty of the N-Arms Restless Bandit problem is coming from:

$$\sum_{s} Y_{s,a}^{(N)}(t) \leq \alpha \, \forall t \in [[0, T-1]],$$

which couples all the arms together.

Challenge of the day:

How to design an efficient heuristic to solve such problem?

Outline of the approach

1. **Relaxation:** Classical approach is to relax this constraint and consider a problem where this constraint has to be satisfied only in expectation:

Outline of the approach

1. **Relaxation:** Classical approach is to relax this constraint and consider a problem where this constraint has to be satisfied only in expectation:

$$\sum_{s} \mathbb{E}[Y_{s,a}^{(N)}(t)] \leq \alpha, \ \forall t \in [[0, T-1]].$$

Outline of the approach

1. **Relaxation:** Classical approach is to relax this constraint and consider a problem where this constraint has to be satisfied only in expectation:

$$\sum_s \mathbb{E}[Y_{s,a}^{(N)}(t)] \leq \alpha, \ \forall t \in [[0, T-1]].$$

2. **Interpolation:** Construct a sequence of decision rules $\pi_t : \Delta^d \to \Delta^{2d}$ which is optimal for the relaxed problem.

Relaxed problem

$$\min_{\pi} \quad \sum_{t=0}^{T-1} \mathbb{E} \sum_{s,a} r_s^a Y_{a,s}^{(N)}(t) =: V_{rel}^{(N)}(m(0), T) \tag{5a}$$

s.t. Arms follow the Markovian evolution, (5b)

$$Y_{0,s}^{(N)}(t) + Y_{1,s}^{(N)}(t) = M_s^{(N)}(t),\ \forall t \in [[0, T-1]],\ \forall s \in \mathcal{S}, \tag{5c}$$

$$\sum_{s} \mathbb{E}[Y_{s,a}^{(N)}(t)] \leq \alpha\ \forall t \in [[0, T-1]],, \tag{5d}$$

$$M_s^{(N)}(0) = m_s(0),\ \forall s \in \mathcal{S}, \tag{5e}$$

LP formulation

Let us define the following LP problem:

$$
\begin{aligned}
\min_{y \geq 0} \quad & \sum_{t=0}^{T-1} \sum_{s,a} r_s^a y_{s,a}(t) =: V_{LP}(m(0), T) \\
\text{s.t.} \quad & y_{s,0}(t) + y_{s,1}(t) = m_s(t),\ \forall t \in [[0, T-1]],\ \forall s \in \mathcal{S}, \\
& m_s(t) = \sum_{s'} \sum_{a} y_{s',a}(t-1) p_{s',s}^a\ \forall t \in [[1, T-1]],\ \forall s \in \mathcal{S}, \\
& \sum_{s} y_{s,1}(t) \leq \alpha,\ \forall t \in [[0, T-1]],, \\
& m_s(0) = m^0,\ \forall s \in \mathcal{S}
\end{aligned}
\tag{6}
$$

LP formulation

Let us define the following LP problem:

$$
\begin{aligned}
\min_{y \geq 0} \quad & \sum_{t=0}^{T-1} \sum_{s,a} r_s^a y_{s,a}(t) =: V_{LP}(m(0),T) \\
\text{s.t.} \quad & y_{s,0}(t) + y_{s,1}(t) = m_s(t),\ \forall t \in [[0, T-1]],\ \forall s \in \mathcal{S}, \\
& m_s(t) = \sum_{s'} \sum_{a} y_{s',a}(t-1) p_{s',s}^a \ \forall t \in [[1, T-1]],\ \forall s \in \mathcal{S}, \\
& \sum_{s} y_{s,1}(t) \leq \alpha,\ \forall t \in [[0, T-1]],, \\
& m_s(0) = m^0,\ \forall s \in \mathcal{S}
\end{aligned}
\tag{6}
$$

We denote by $y^* := [[[y^*_{s,a}(t)]]]_{s,a,t}$ the optimal solution of (6) and we also define $m^* := [[m_s(t) := \sum_a y^*_{s,a}(t)]]_{s,t}$.

Equivalence

Lemma:

$$
\begin{aligned}
V_{rel}(m^0, T) &= V_{LP}(m^0, T), \\
V_{opt}^{(N)}(m(0), T) &\geq V_{LP}(m^0, T).
\end{aligned}
$$

Projection

We define the set of feasible control at time t by:

$$\mathcal{Y}(M^{(N)}(t)) := \left\{ y \in \mathbb{R}_+^{2d} | \sum_a y_{s,a} = M_s^{(N)}(t) \; \forall s \in \mathcal{S}; \; \sum_s y_{s,1} = \alpha \right\}$$

Projection

We define the set of feasible control at time t by:

$$\mathcal{Y}(M^{(N)}(t)) := \left\{ y \in \mathbb{R}_+^{2d} \middle| \sum_a y_{s,a} = M_s^{(N)}(t)\ \forall s \in \mathcal{S};\ \sum_s y_{s,1} = \alpha \right\}$$

Some observations:

Projection

We define the set of feasible control at time t by:

$$\mathcal{Y}(M^{(N)}(t)) := \left\{ y \in \mathbb{R}_+^{2d} | \sum_a y_{s,a} = M_s^{(N)}(t) \ \forall s \in \mathcal{S}; \ \sum_s y_{s,1} = \alpha \right\}$$

Some observations:

1. In general $y^*(t) \notin \mathcal{Y}(M^{(N)}(t))$;

Projection

We define the set of feasible control at time t by:

$$\mathcal{Y}(M^{(N)}(t)) := \left\{ y \in \mathbb{R}_+^{2d} | \sum_a y_{s,a} = M_s^{(N)}(t) \; \forall s \in \mathcal{S}; \; \sum_s y_{s,1} = \alpha \right\}$$

Some observations:

1. In general $y^*(t) \notin \mathcal{Y}(M^{(N)}(t))$;
2. In general $y^*(t) \in \mathcal{Y}(m^*(t))$.

Projection

We define the set of feasible control at time t by:

$$\mathcal{Y}(M^{(N)}(t)) := \left\{ y \in \mathbb{R}_+^{2d} | \sum_a y_{s,a} = M_s^{(N)}(t) \ \forall s \in \mathcal{S}; \ \sum_s y_{s,1} = \alpha \right\}$$

Some observations:

1. In general $y^*(t) \notin \mathcal{Y}(M^{(N)}(t))$;
2. In general $y^*(t) \in \mathcal{Y}(m^*(t))$.

We define the following **projection operator**:

$$\mathsf{Proj}_t(M^{(N)}) := \mathsf{argmin}_{y \in \mathcal{Y}(M^{(N)}(t))} \|y - y^*\|_2^2. \qquad (7)$$

Algorithm

The Projection Policy

- **Input:** Initial system configuration vector $m(0)$ and time horizon T.
- **Solve** (6) to obtain y^*;
- **Set** $\hat{M}$:= m(0);
- **For** $t = 0, 2, \ldots, T-1$ **do:**
  1. *Projection step:* Compute $\hat{y}(t) := \mathrm{Proj}_t(\hat{M})$;
  2. *Rounding step:* For all $s \in \mathcal{S}$, set:

$$\hat{Y}^{(N)}_{s,a}(t) = \begin{cases} N^{-1}\lfloor N\hat{y}_{s,1}(t)\rfloor & \text{if } a = 1, \\ \hat{M}_s - N^{-1}\lfloor N\hat{y}_{s,1}(t)\rfloor & \text{otherwise.} \end{cases}$$

  3. Use control $\hat{Y}^{(N)}$ to advance to the next time-step ;
  4. Set $\hat{M}$:= current empirical distribution;

Policy construction scheme

A simple scheme can explain easily our algorithm:

Policy construction scheme

A simple scheme can explain easily our algorithm:

$$\hat{M}(t) \xrightarrow[\text{Proj. step}]{\mathsf{Proj}_t(\hat{M}(t))} \hat{y}(t) \xrightarrow[\text{Roun. step}]{} \hat{Y}^{(N)}_{s,a}(t) \xrightarrow[\text{Trans. step}]{} \hat{M}(t+1)$$

Remark: We will see in the next theorem that the projection step can be replaced by map $\pi(\cdot)$ such that:

Policy construction scheme

A simple scheme can explain easily our algorithm:

$$\hat{M}(t) \xrightarrow[\text{Proj. step}]{\mathsf{Proj}_t(\hat{M}(t))} \hat{y}(t) \xrightarrow[\text{Roun. step}]{} \hat{Y}^{(N)}_{s,a}(t) \xrightarrow[\text{Trans. step}]{} \hat{M}(t+1)$$

Remark: We will see in the next theorem that the projection step can be replaced by map $\pi(\cdot)$ such that:

1. *Admissible policy:* $\pi_t(M^{(N)}(t)) \in \mathcal{Y}(M^{(N)}(t))$,

Policy construction scheme

A simple scheme can explain easily our algorithm:

$$\hat{M}(t) \xrightarrow[\text{Proj. step}]{\text{Proj}_t(\hat{M}(t))} \hat{y}(t) \xrightarrow[\text{Roun. step}]{} \hat{Y}^{(N)}_{s,a}(t) \xrightarrow[\text{Trans. step}]{} \hat{M}(t+1)$$

Remark: We will see in the next theorem that the projection step can be replaced by map $\pi(\cdot)$ such that:

1. *Admissible policy:* $\pi_t(M^{(N)}(t)) \in \mathcal{Y}(M^{(N)}(t))$,
2. *LP-compatible policy:* $\pi_t(m^*(t)) = y^*(t)$.

Main result

Theorem

Let $\pi := \{\pi_t\}_{0 \leq t \leq T-1}$ be a continuous an admissible and continuous policy then

Main result

Theorem

Let $\pi := \{\pi_t\}_{0\leq t\leq T-1}$ be a continuous an admissible and continuous policy then

$$\lim_{N\to+\infty} V^{(N)}_{opt,\pi}(m(0),T) = V_{rel,\pi}(m(0),T).$$

Main result

Theorem

Let $\pi := \{\pi_t\}_{0 \le t \le T-1}$ *be a continuous an admissible and continuous policy then*

$$\lim_{N \to +\infty} V^{(N)}_{opt,\pi}(m(0), T) = V_{rel,\pi}(m(0), T).$$

Moreover if π *is LP-compatible then*

Main result

Theorem

Let $\pi := \{\pi_t\}_{0 \leq t \leq T-1}$ be a continuous an admissible and continuous policy then

$$\lim_{N \to +\infty} V^{(N)}_{opt,\pi}(m(0), T) = V_{rel,\pi}(m(0), T).$$

Moreover if π is LP-compatible then

$$\lim_{N \to +\infty} V^{(N)}_{opt,\pi}(m(0), T) = V^{(N)}_{LP}(m(0), T).$$

Bibliography

- The proof of the main theorem and more advance theorem can be found here: Gast, Nicolas, Bruno Gaujal, and Chen Yan. "The LP-update policy for weakly coupled Markov decision processes." arXiv preprint arXiv:2211.01961 (2022).
- If you want to find a lot of different applications, you can have a look at: Avrachenkov, Konstantin E., and Vivek S. Borkar. "Whittle index based Q-learning for restless bandits with average reward." Automatica 139 (2022): 110186.