Restless Bandit Problems and Computation of Index Policies

Nicolas Gast

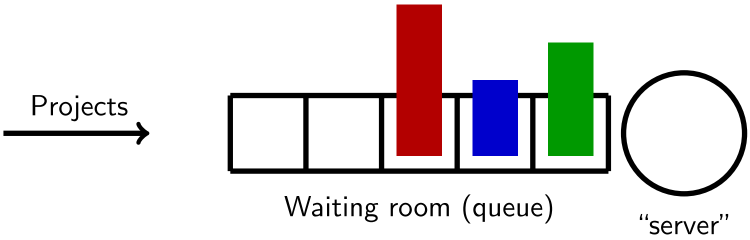
joint work with our two students Kimang Khun, Chen Yan,
co-supervised with Bruno Gaujal

Inria

AEP – Grenoble – July, 2022

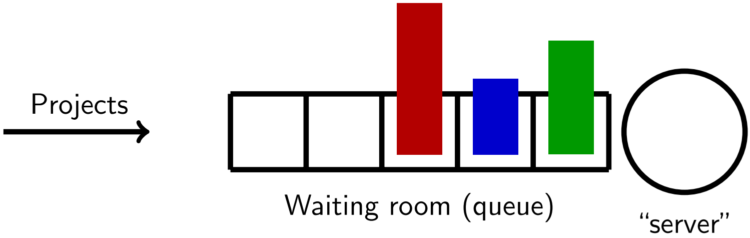
Motivation: What to work on?

Motivation: What to work on?



- Examples: research projects, tasks allocations, electric vehicle charging, wireless scheduling,...
- We allow preemption (preempt-resume).

Motivation: What to work on?



If you know the project sizes and you want to minimize the waiting time: use SRPT (Shortest Remaining Processing Time).

- "Strongly optimal" [Schrage, 1966]

Is SRPT always optimal?

Is SRPT always optimal?

Short answer: no

- Projects have different rewards.
- Impatient customers (research completed by other team)
- Durations are unknown.
- ...

Objective of the talk: How can we do better?

Outline

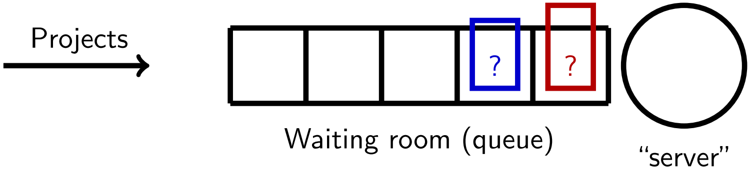
Intuitive Example: What is an Index policy?

Definition: Resltess Bandits and Whittle index

How to Compute Indices: A Sub-Cubic Algorithm

Conclusion

Example: scheduling with random job durations



Example: How to schedule with unknown durations?

Intuition suggests SERPT (shortest **expected** remaining processing time)

SERPT is in general not optimal

Example: two jobs of sizes X and Y with:

- $X = 10 - \varepsilon$
- $Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$

SERPT is in general not optimal

Example: two jobs of sizes X and Y with:

- $X = 10 - \varepsilon$
- $Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$

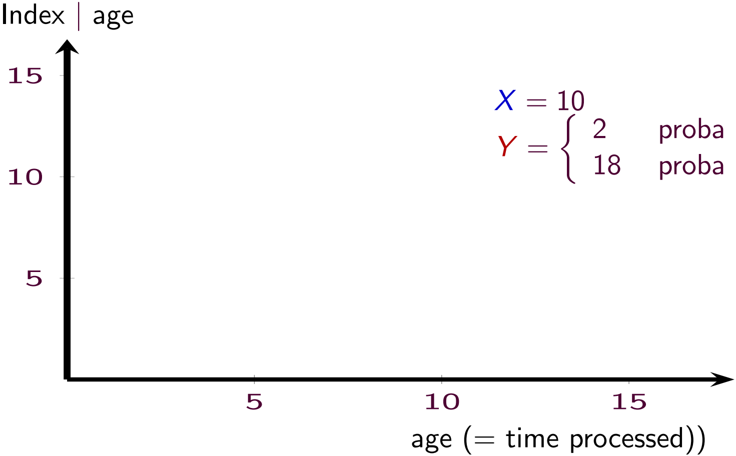
For this model: Gittins index policy is optimal:

- Gittins index policy: serves job with smallest index first.

Running a job costs 1€/sec and you can stop anytime. If you finish the job, you earn x. Gittins index $=$ smallest x so that you running or stopping is equivalent.

SERPT is in general not optimal

Running a job costs 1€/sec and you can stop anytime. If you finish the job, you earn x. Gittins index = smallest x so that you running or stopping is equivalent.

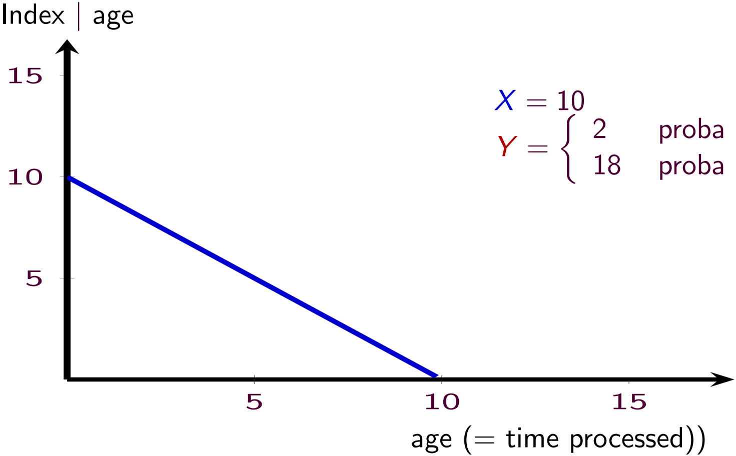


$$X = 10$$

$$Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$$

SERPT is in general not optimal

Running a job costs 1€/sec and you can stop anytime. If you finish the job, you earn x. Gittins index = smallest x so that you running or stopping is equivalent.

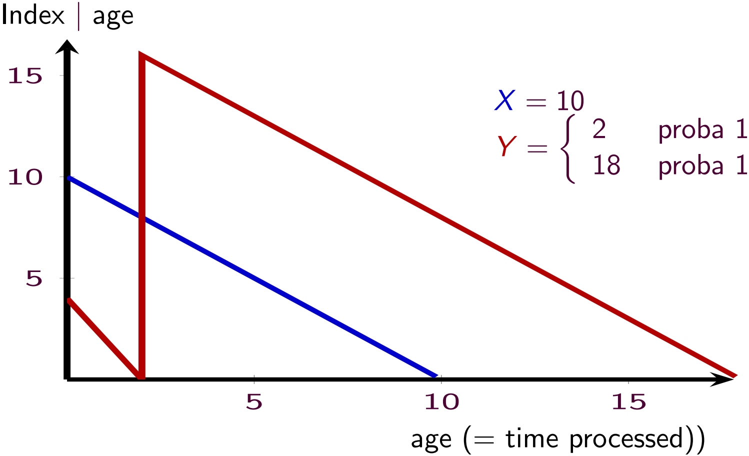


$$X = 10$$

$$Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$$

SERPT is in general not optimal

Running a job costs 1€/sec and you can stop anytime. If you finish the job, you earn x. Gittins index = smallest x so that you running or stopping is equivalent.



$X = 10$

$Y = \begin{cases} 2 & \text{proba } 1/2 \\ 18 & \text{proba } 1/2 \end{cases}$

Outline

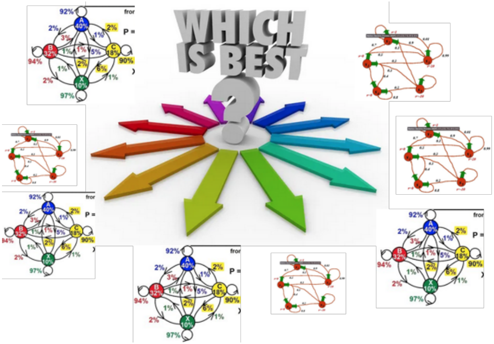
Intuitive Example: What is an Index policy?

Definition: Resltess Bandits and Whittle index

How to Compute Indices: A Sub-Cubic Algorithm

Conclusion

Reslless Markov bandit problem



A decision maker faces n arms.
Time is discrete.

- Each arm is a 2-action MDP (passive / active)
- Controller can activate $m < n$ arms each time.

Policy : you observe the states. Which ones do you activate?

Indexability

An arm is a 2-action MDP: if in state s:

- Activation: earn $r(s, active)$, jump to $j \sim \mathbf{P}\left[j \mid s, active\right]$.
- Passive: earn $r(s, passive)$, jump to $j \sim \mathbf{P}\left[j \mid s, passive\right]$.

Indexability

We consider a x-subsidized MDP.

An arm is a 2-action MDP: if in state s:

- Activation: earn $r(s, active)$, jump to $j \sim \mathbf{P}[j \mid s, active]$.
- Passive: earn $r(s, passive)+x$, jump to $j \sim \mathbf{P}[j \mid s, passive]$.

Let $\pi_s =$ values of x such that "activation" is optimal. If $\pi_s = (-\infty, \lambda(s)]$, then the state is indexable and $\lambda(s)$ is its Whittle index.

Optimality of index policies

When $P(j \mid s, passive)$, the bandit is "rested". In this case, Whittle index=Gittins index.

Theorem (Gittins, Glazebrook, Weber, 90s)

For rested bandits and $m = 1$, the Gittins index policy is optimal.

- Preemptive-resume is rested: running tasks with smallest index is optimal.
- Impatient customers is restless, multi-server is $m \geq 2$.

Theorem (Weber Weiss 90s)

For restless bandits, the Whittle index policy is asymptotically optimal in the regime $n \to \infty$ and $m/n = O(1)$ under the "global attractor condition".

Are all related problems closed since the 90s?

Are all related problems closed since the 90s?

(Recently) closed questions:

- Can we analyze the performance of this Gittins index? (SOAP: [Scully et al 2018]).
- Multi-server? (Close to optimal: [Grosof et al 2019]).

Are all related problems closed since the 90s?

(Recently) closed questions:

- Can we analyze the performance of this Gittins index? (SOAP: [Scully et al 2018]).
- Multi-server? (Close to optimal: [Grosof et al 2019]).
- At which speed do Whittle index become optimal? (exponentially fast in most cases [G,Gaujal,Yan 2021])
- Can we define index for finite-horizon problems? [Frazier et al. 2019-20, G, Gaujal, Yan 2022].
- Can we leverage indexable to problem to obtain better learning algorithms? (No regret learning.[G,G,Khun, 2021])

Are all related problems closed since the 90s?

(Recently) closed questions:

- Can we analyze the performance of this Gittins index? (SOAP: [Scully et al 2018]).
- Multi-server? (Close to optimal: [Grosof et al 2019]).
- At which speed do Whittle index become optimal? (exponentially fast in most cases [G,Gaujal,Yan 2021])
- Can we define index for finite-horizon problems? [Frazier et al. 2019-20, G, Gaujal, Yan 2022].
- Can we leverage indexable to problem to obtain better learning algorithms? (No regret learning.[G,G,Khun, 2021])
- Are Whittle index hard to compute? [G, G. Khun, 2022]

Outline

Intuitive Example: What is an Index policy?

Definition: Resltess Bandits and Whittle index

How to Compute Indices: A Sub-Cubic Algorithm

Conclusion

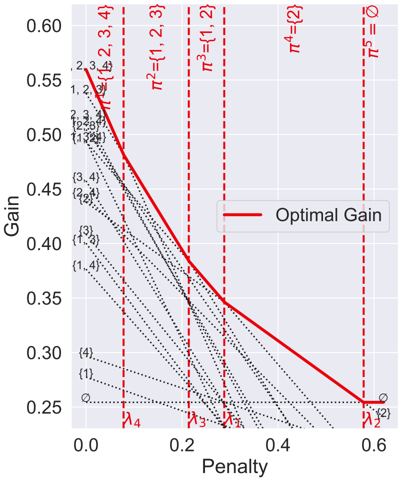
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x) =$ value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \ldots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.



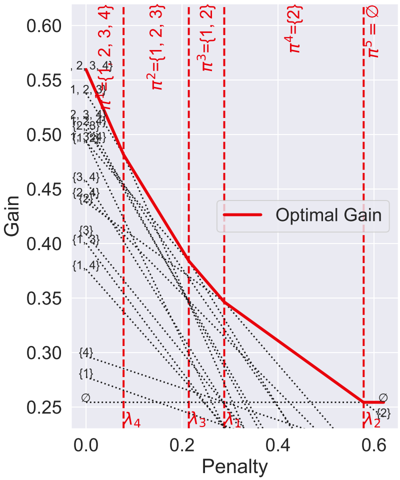
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \ldots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.



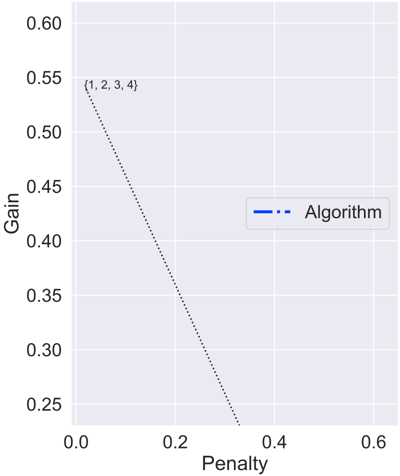
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.
2. $\pi_*(-\infty) = \{1 \dots S\}$.



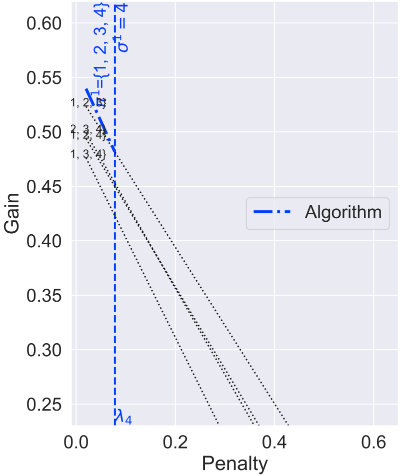
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.
2. $\pi_*(-\infty) = \{1 \dots S\}$.
3. Computing $g_{\pi\setminus\{i\}}(x)$ from $g_\pi(x)$ is easy.



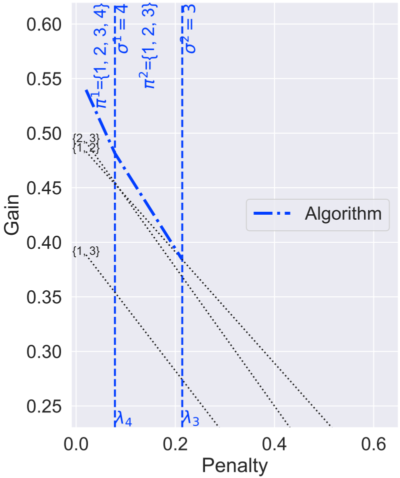
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.
2. $\pi_*(-\infty) = \{1 \dots S\}$.
3. Computing $g_{\pi \setminus \{i\}}(x)$ from $g_\pi(x)$ is easy.



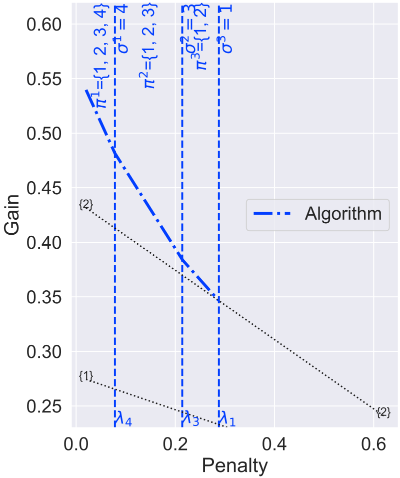
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x)$ = value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.
2. $\pi_*(-\infty) = \{1 \dots S\}$.
3. Computing $g_{\pi \setminus \{i\}}(x)$ from $g_\pi(x)$ is easy.



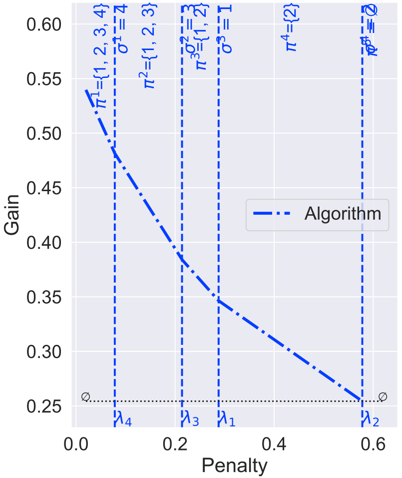
General idea: compute the index by increasing order

- A policy is $\pi \subset \{1 \dots S\}$
- $g_\pi(x) =$ value for subsidy x.
- $\pi^*(x) = \arg\max_\pi g_\pi(x)$.

We want to find the inflection points of the red curve.

Facts:

1. $g_\pi(x)$ is linear in x.
2. $\pi_*(-\infty) = \{1 \dots S\}$.
3. Computing $g_{\pi \setminus \{i\}}(x)$ from $g_\pi(x)$ is easy.



Why are the facts true?

- $g_\pi(x) = (A^\pi)^{-1}(r + x\pi)$

Why are the facts true?

- $g_\pi(x) = (A^\pi)^{-1}(r + x\pi)$
- If subsidy is $-\infty$ for all states, then we should not activate.

Why are the facts true?

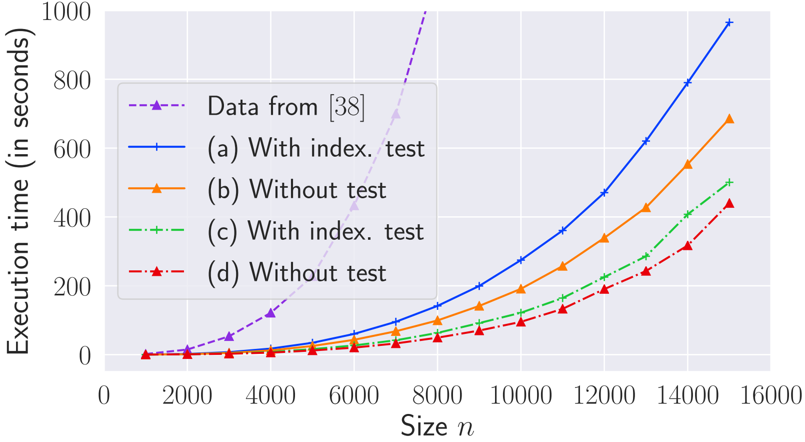
- $g_\pi(x) = (A^\pi)^{-1}(r + x\pi)$
- If subsidy is $-\infty$ for all states, then we should not activate.
- Sherman-Morisson formula: Let A be an invertible matrix, u and v vectors $1D$ such that $1 + v^T A^{-1} u \neq 0$. Then:

$$\left(A + uv^T\right)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}.$$

By using fast matrix multiplication, we can compute Whittle indices in $O(S^{2.53})$ operations (conjectured to be at least n^3 in a 2016 paper).

Simulation result

https://pypi.org/project/markovianbandit-pkg/



Outline

Conclusion

Index policies are very efficient to share resources among tasks.

- Idea: compute the "right price", and activate the cheapest.

This scales well and performs very well in practice.

- This talk: Optimality of index and computation of index.
- Open questions: learning, continuous state-spaces.

Conclusion

Index policies are very efficient to share resources among tasks.

- Idea: compute the "right price", and activate the cheapest.

This scales well and performs very well in practice.

- This talk: Optimality of index and computation of index.
- Open questions: learning, continuous state-spaces.

http://polaris.imag.fr/nicolas.gast/

- *Computing Whittle (and Gittins) Index in Subcubic Time*, G. Gaujal, Khun https://arxiv.org/abs/2203.05207
- *LP-based policies for restless bandits: necessary and sufficient conditions for (exponentially fast) asymptotic optimality*. G. Gaujal Yan. https://arxiv.org/abs/2106.10067