Time-Constrained Restless Bandits and the Knapsack Problem for Perishable Items

Peter Jacko & José Niño-Mora

CS 2006, Prague

Motivation

- Perishable product
  - ⋆ product with associated deadline after which it becomes worthless, if not sold
  - ⋆ arises in food industry ("best before" date), fashion industry (seasonal goods), etc.
- Q: How to select perishable products to be promoted?
  - ⋆ cannot ignore perishability!
  - ⋆ likely to be PSPACE-hard
- Similar problems in task management, project selection

An Application: Task Management

(Prepare)
Classes
Due: tomorrow

(Have)
Lunch
Due: before 1pm

Investigate
Due: one month

?

(Evaluate)
Homeworks
Due: one week

(Write)
Paper
Due: two weeks

(Look for)
Funding
Due: next year

Knapsack Problem

- Classical 0-1 Knapsack Problem for items in $\mathcal{I}$:

$$\begin{aligned} \max_{\boldsymbol{x}} & \sum_{i \in \mathcal{I}} v_i x_i \\ \text{subject to } & \sum_{i \in \mathcal{I}} w_i x_i \leq W \\ & x_i \in \{0, 1\} \text{ for all } i \in \mathcal{I} \end{aligned} \qquad \text{(KP)}$$

- There are 2 stages:
  - ⋆ stage 0: select items to put into knapsack
  - ⋆ stage 1: obtain rewards v_i

Characterization of a Perishable Item

- Selection stages: $0, 1, \ldots, T_i - 1$
  - ⋆ occupies space w_i
  - ⋆ if in knapsack, it remains unsold with probability p_i
  - ⋆ if not in knapsack, it remains unsold with probability $q_i > p_i$
  - ⋆ once sold, it never resurrects

- Final stage (deadline): T_i
  - ⋆ pay cost $c_i > 0$ if not sold (“bad” state)
  - ⋆ no cost if sold (“good” state)

Knapsack Problem for Perishable Items

- Let $T = \max_{i \in \mathcal{I}} \{T_i\}$ be the time horizon
- Selection stages: $0, 1, \ldots, T - 1$
- A dynamic and stochastic problem (MDP)
- Aim: to minimize the total expected cost to pay
- Reduces to (KP), when $T_i = 1$, $q_i = 1$, $p_i = 0$, and $c_i = v_i$ for all $i \in \mathcal{I}$
- (KP) is NP-hard $\Longrightarrow$ KPPI is at least NP-hard

Intuitive Solution

- To each item assign a priority of choosing it
- At each selection stage, put the items with highest priority into the knapsack
- Surprisingly: such behavior is often nearly optimal
- Questions to answer:
  - ⋆ How to assign priorities to items?
  - ⋆ How far from optimality is it?
  - ⋆ Is it better than other strategies (policies)?

Dynamic Programming Formulation

$$D_T(\boldsymbol{z}_T) = \sum_{i \in \mathcal{I}_T^0} c_i z_{(T,i)} \qquad \text{(DP)}$$

$$D_s(\boldsymbol{z}_s) = \sum_{i \in \mathcal{I}_s^0} c_i z_{(s,i)} + \min_{\substack{\boldsymbol{y}_s \le \boldsymbol{z}_s^+ \\ \sum_{i \in \mathcal{I}_s^+} w_i y_{(s,i)} \le W}} \left\{ \sum_{\boldsymbol{m}_s \le \boldsymbol{z}_s^+} \mathbb{P}^{\boldsymbol{y}_s}[\boldsymbol{m}_s] D_{s+1}(\boldsymbol{z}_s^+ - \boldsymbol{m}_s) \right\}$$

- Solving a system of an exponential number of equations for an exponential number of vectors $\boldsymbol{z}_s$ at every stage
  - ⋆ tractability problem: curse of dimensionality
  - ⋆ no interpretation

Bandit Machine

Multi-Armed Bandit Problem

- There are K independent arms
- In every time epoch, one arm must be pulled
- Rested arms are frozen (no work, no reward, no change)
- Solved by Gittins et al. in early 1970's
- Assigned a Gittins index to each arm and its state
- Optimal policy: index policy
- Decomposes K-dim. problem to K one-dimensional

Gittins Index

- Arm k when in state x has the Gittins index

$$\nu_k(x) = \max_{\tau > 0} \frac{\mathbb{E}\left\{\sum_{t=0}^{\tau-1} \beta^t r_k(x_k(t)) | x_k(0) = x\right\}}{\mathbb{E}\left\{\sum_{t=0}^{\tau-1} \beta^t | x_k(0) = x\right\}} \qquad \text{(GI)}$$

- Maximal attainable expected reward per expected work
- I.e., indicates the “worth” of pulling the arm
- Calculated in $\mathcal{O}(n^3)$ by an adaptive-greedy algorithm

Restless Bandit Problem

- Drops the freezing property
- Allows to pull parallely exactly M arms
- Whittle's relaxation '88: pull M arms on average
  - ⋆ an upper bound and a heuristic for "some" RBP
- Papadimitriou & Tsitsiklis '99: deterministic version is PSPACE-hard
- Niño-Mora '01: Sufficient condition for existence of an index-policy heuristic using marginal productivity index

Common features of RBP and KPPI

- Problems of allocation of a scarce resource
- Arms to pull $\Longleftrightarrow$ Items to select
- M hands to use $\Longleftrightarrow$ Knapsack's space W
- Probability to win $\Longleftrightarrow$ Probability to sell

Contrasting RBP with KPPI

- Restless Bandit Problem:
  - ⋆ $w_i = 1$ for all $i \in \mathcal{I}$
  - ⋆ condition is equality
  - ⋆ infinite horizon
  - ⋆ stationary costs

- Knapsack Problem for Perishable Items:
  - ⋆ w_i is arbitrary
  - ⋆ condition is inequality
  - ⋆ finite horizon
  - ⋆ only one cost at deadline

Main Results

- Finite-horizon problem with deadline costs can be formulated as an RBP
  - ⋆ augmented state space: state space $\times$ selection stages
- Marginal productivity indices (MPI's) can be obtained in closed form (uncommon!)
- An index-policy heuristic based on MPI's can be defined in a similar way as for RBP
- The heuristic is nearly optimal

Marginal Productivity Index

- Closed form of MPI for item i

$$\nu_i = \frac{c_i(1-p_i)(q_i-p_i)p_i^{T-1}}{1-q_i+(q_i-p_i)p_i^{T-1}} \qquad \text{(MPI)}$$

- Some properties of MPI for an item in isolation:
  - ⋆ item with higher probability of remaining unsold if not selected (q_i) gets higher priority
  - ⋆ item with closer deadline T_i gets higher priority
  - ⋆ MPI is proportional to cost c_i and positive

MPI Heuristic

- "Solve a (KP) at each stage using MPI's: $v_i = \nu_i$"
- Reduces a stochastic and dynamic problem to a simpler deterministic problem
- Considers only the current situation, not any future
- Provides an excellent performance: avg. gap $< 5\%$
- Systematically outperforms naïve policies
- (KP) solved efficiently up to millions of items, e.g. by COMBO algorithm (Martello, Pisinger, & Toth 1999)

Other Policies

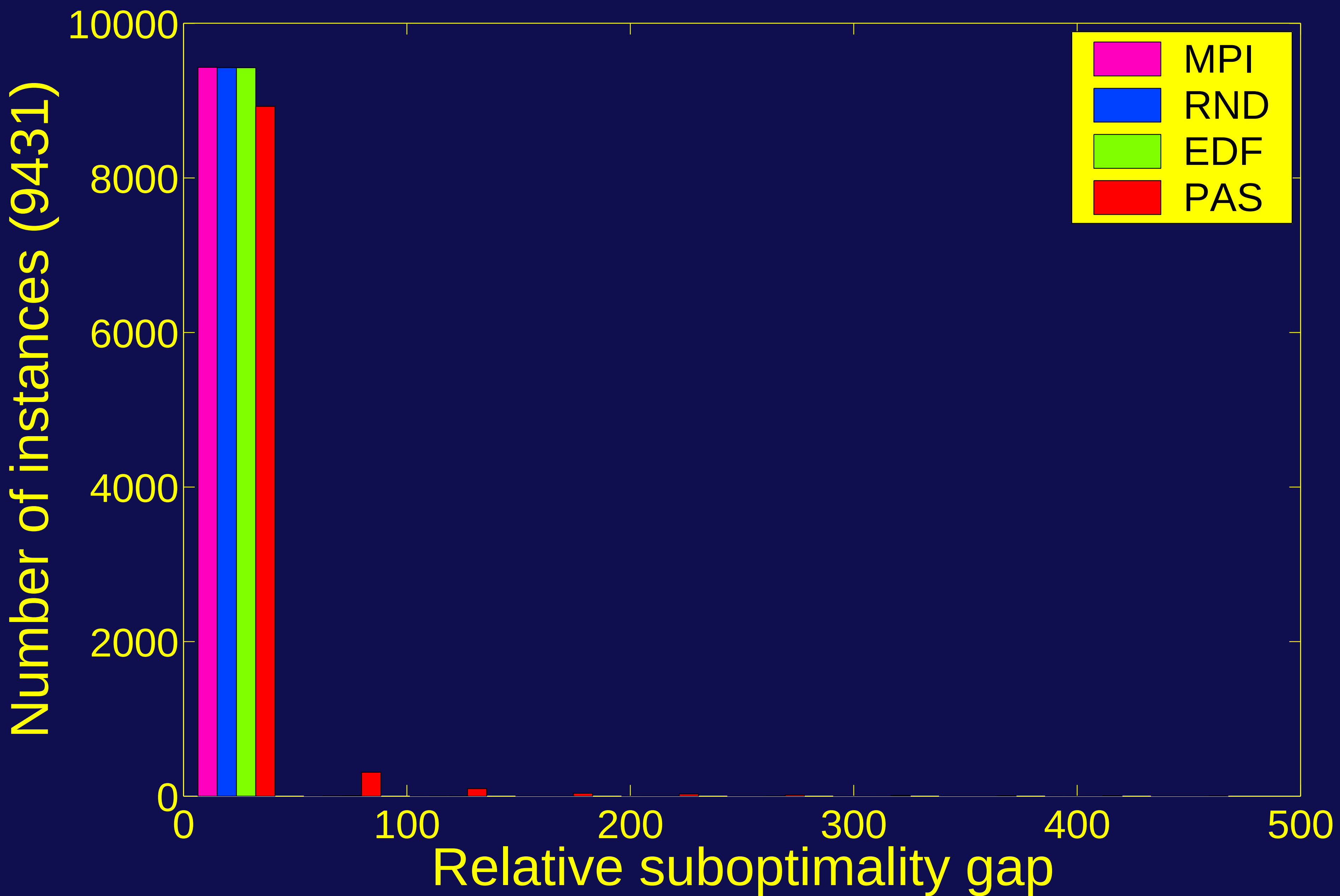
- MIN: best-case policy (optimal solution)
- PAS: passive policy (empty knapsack)
- RND: random solution heuristic
  - order the items randomly
  - select items for knapsack following the order
- EDF: Earlier-Deadline-First heuristic
  - "myopic" strategy, but often used in practice
  - surprisingly, in general behaves worse than RND

Relative Suboptimality Gap

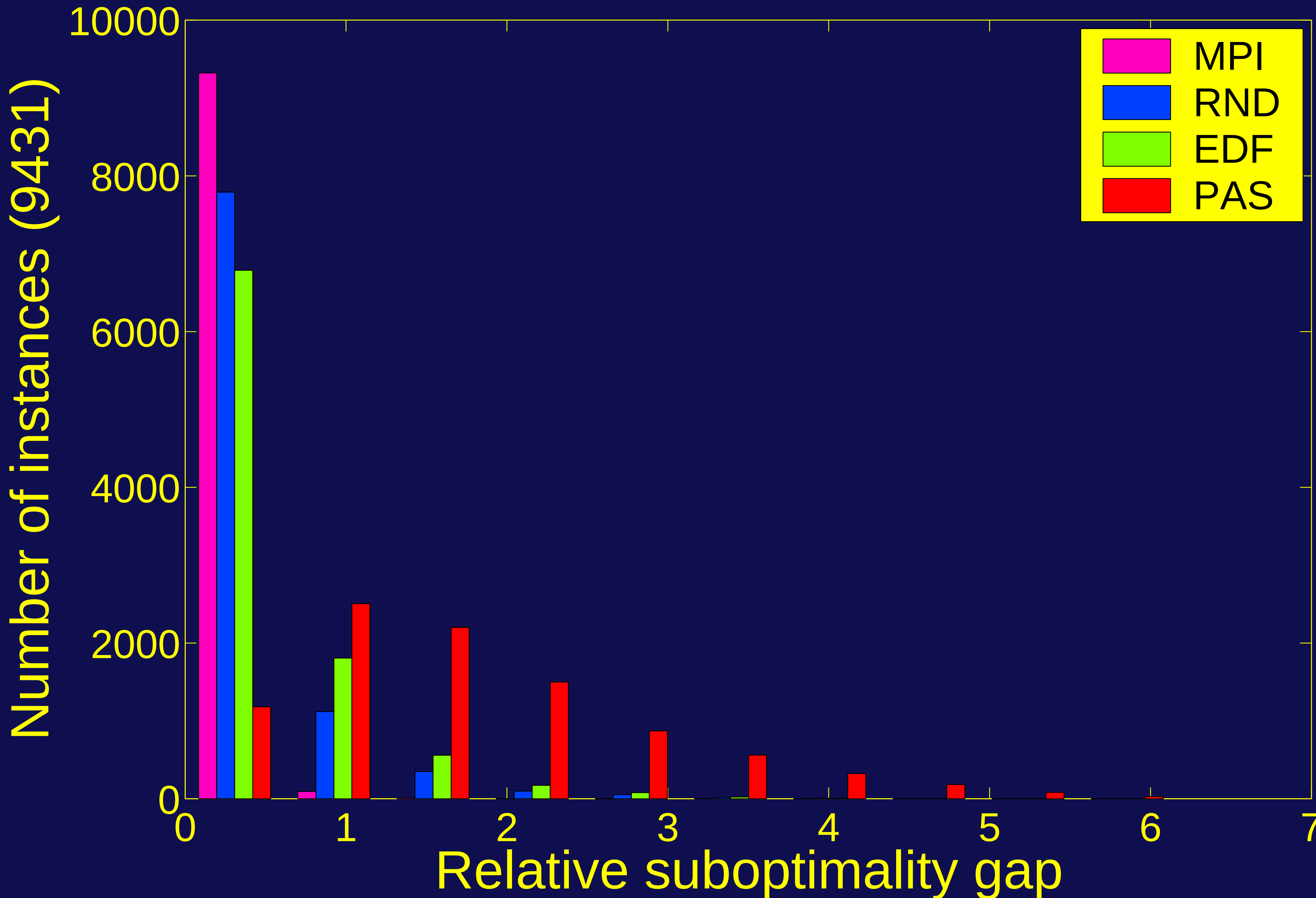
- Relative suboptimality gap of policy π

$$\text{gap}(\pi) = \frac{z^{\pi} - z^{\mathsf{MIN}}}{z^{\mathsf{MIN}}}$$

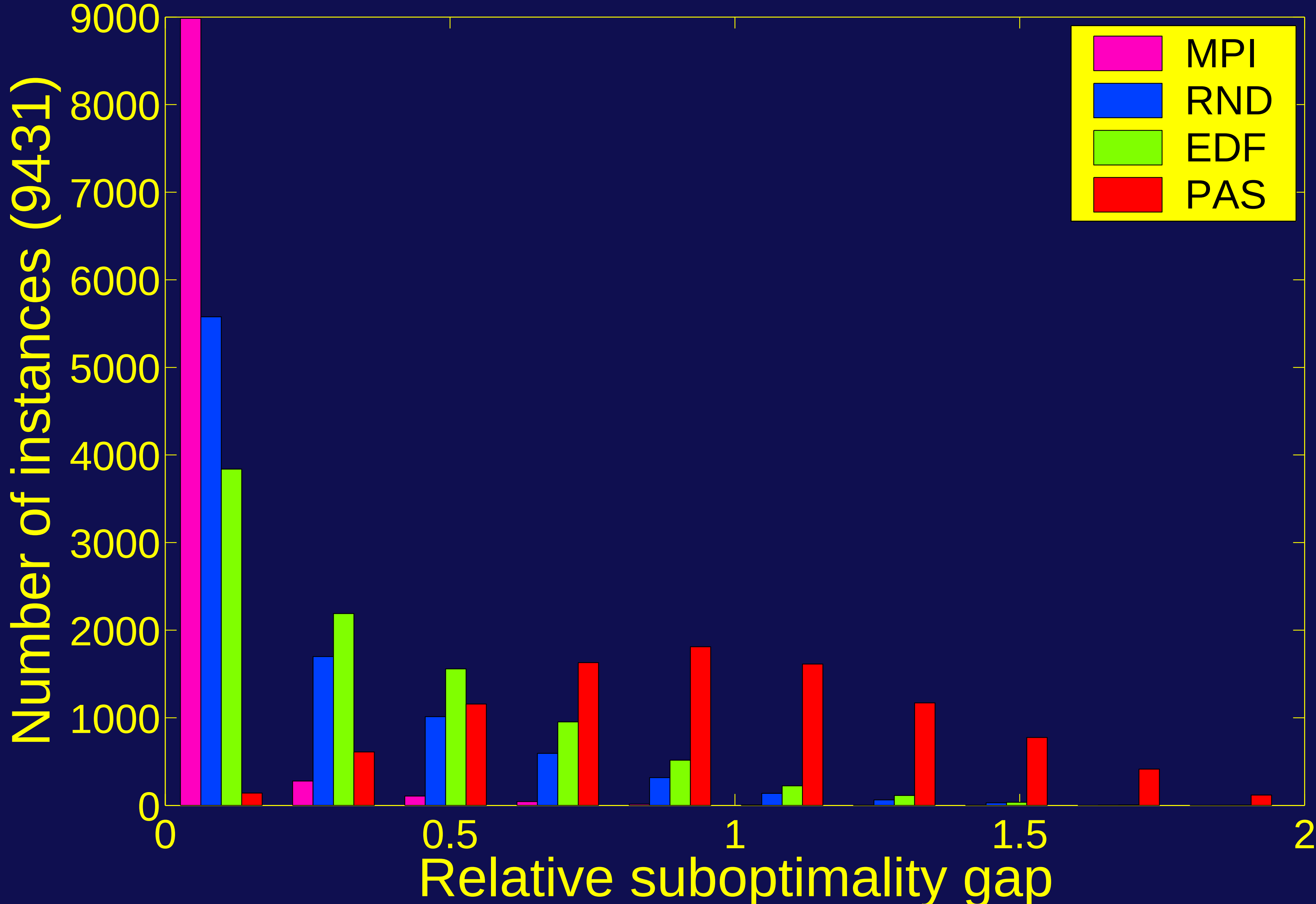
Histogram of gap



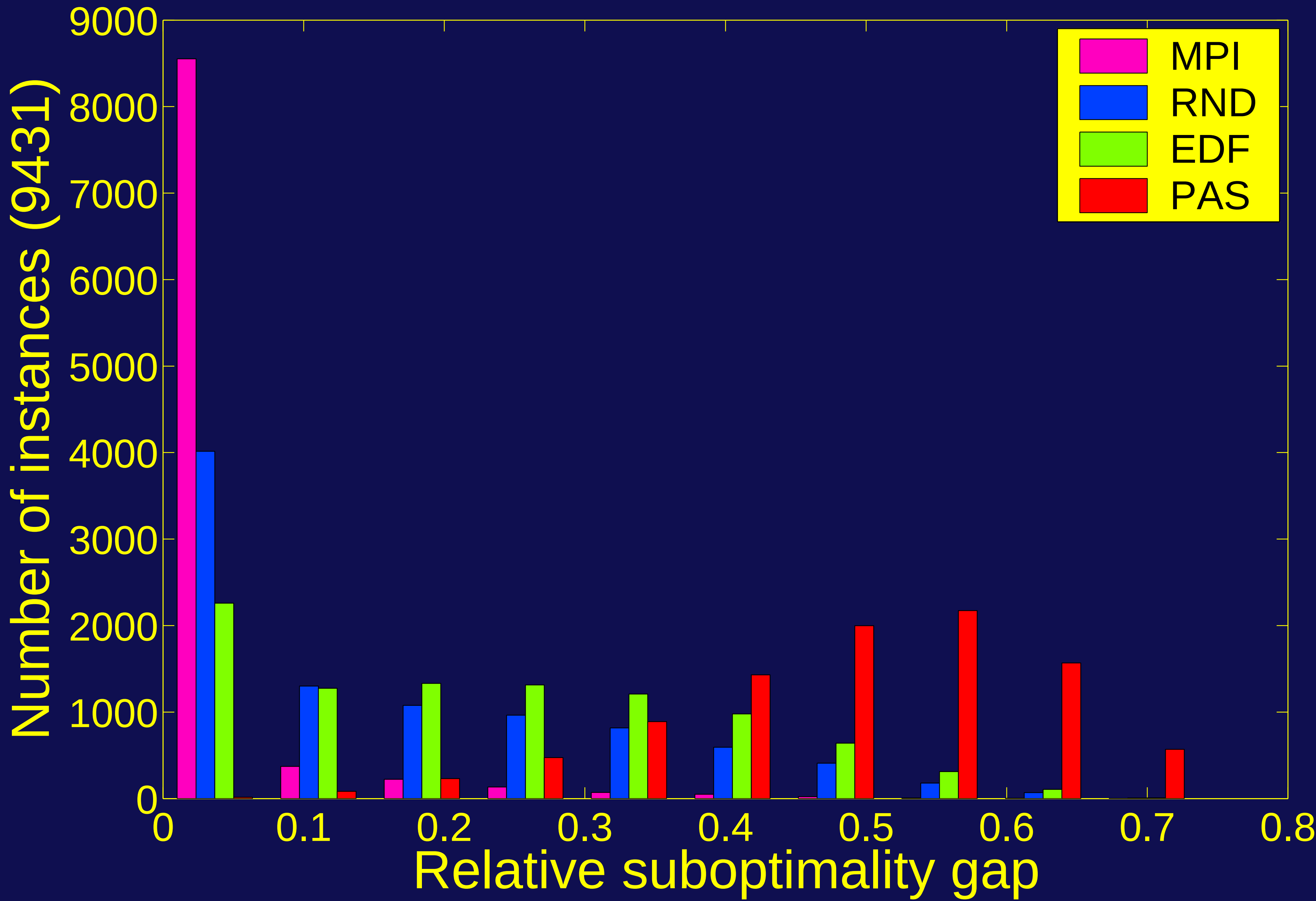
Histogram of gap ($\log$ transformation)



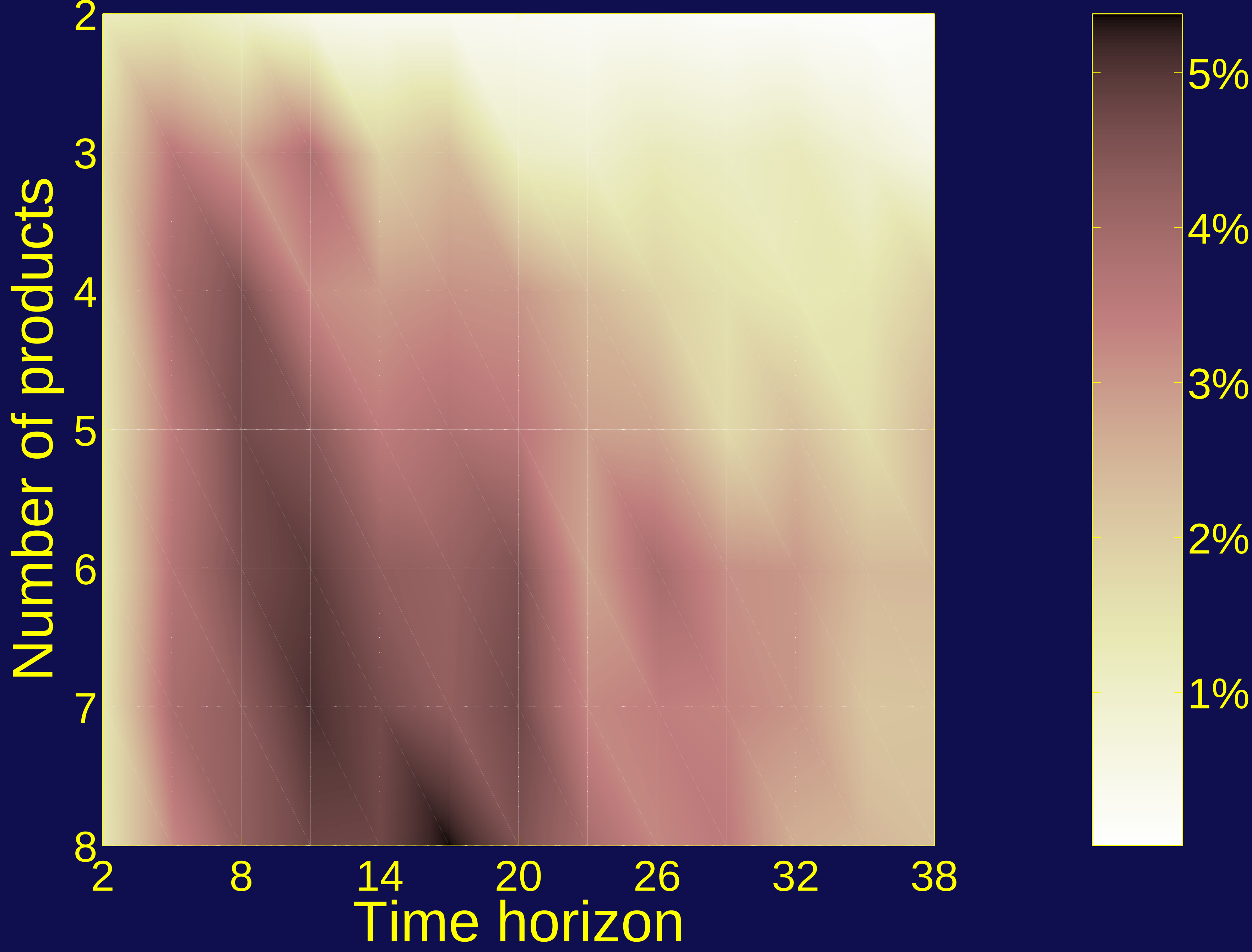
Histogram of gap (2-log transformation)



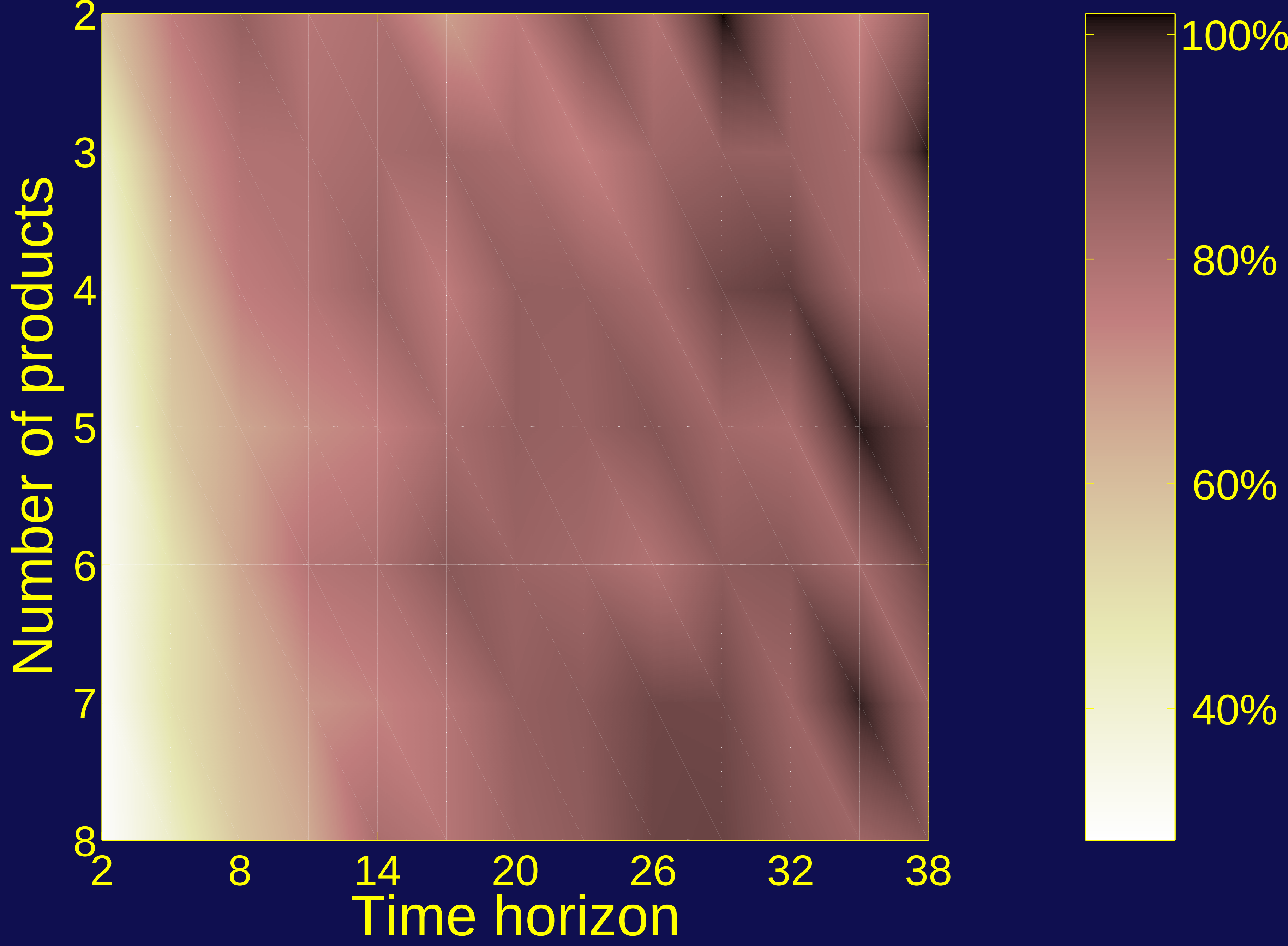
Histogram of gap (4-log transformation)



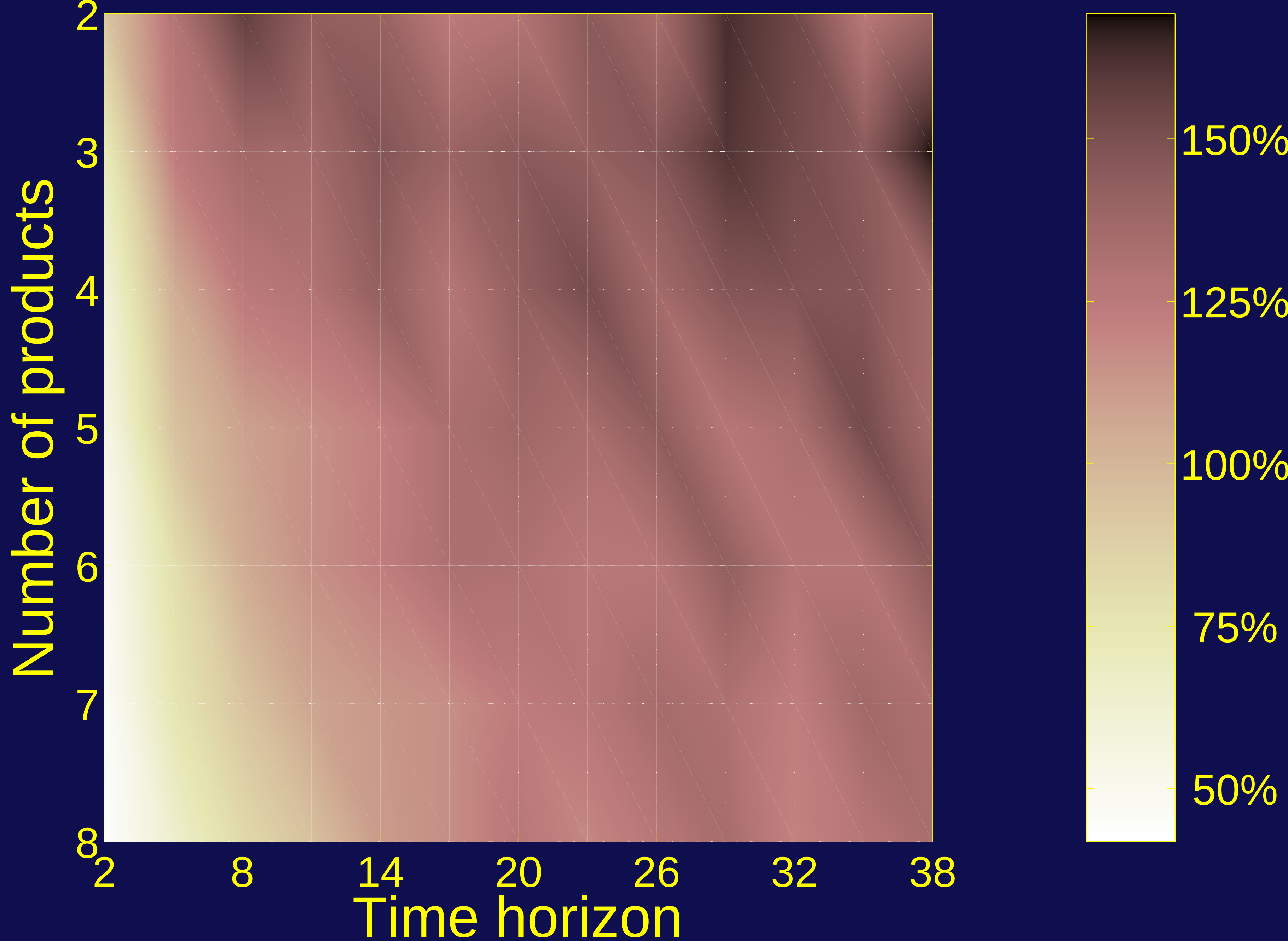
Average gap(MPI)



Average gap(RND)



Average gap(EDF)



Adjusted Relative Suboptimality Gap

- Classical relative suboptimality gap

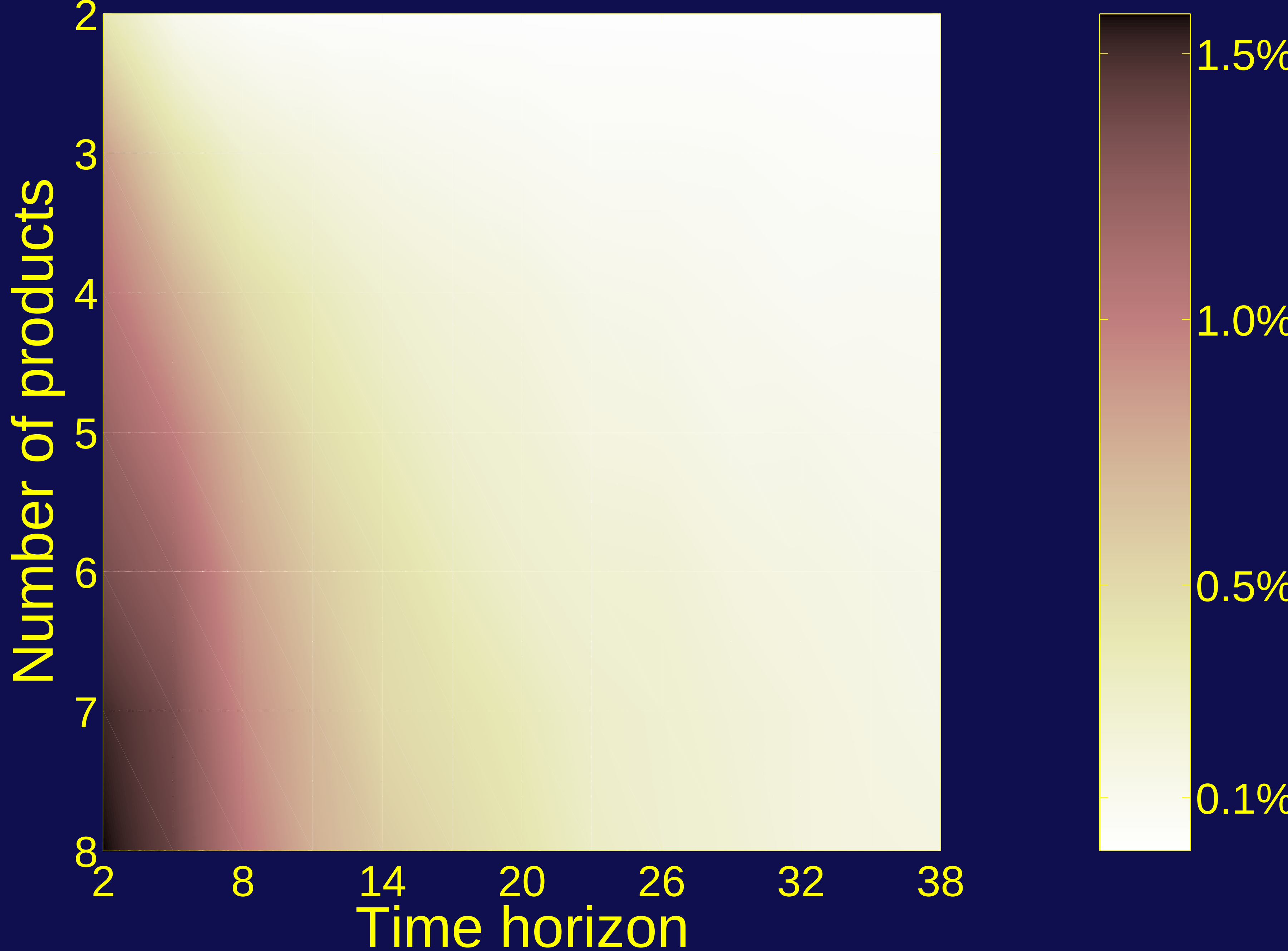
$$\mathrm{gap}(\pi) = \frac{z^{\pi} - z^{\mathsf{MIN}}}{z^{\mathsf{MIN}}}$$

- "Adjusted" relative suboptimality gap

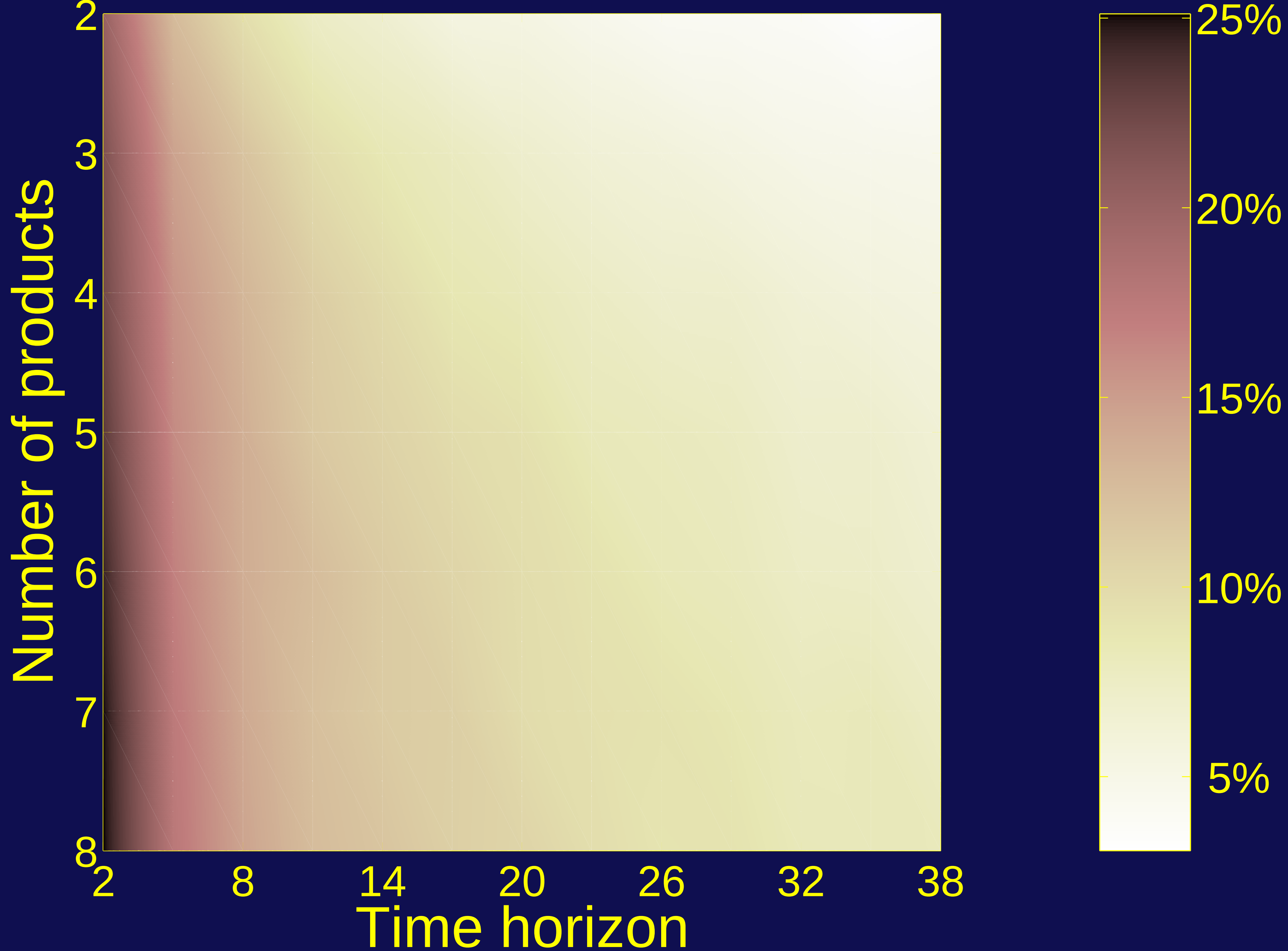
$$\text{A-gap}(\pi) = \frac{z^{\pi} - z^{\mathsf{MIN}}}{z^{\mathsf{PAS}} - z^{\mathsf{MIN}}}$$

  - ⋆ to calibrate randomly generated instances
  - ⋆ takes into account both $\min$ and $\max$ values
  - ⋆ always between 0 (best-case) and 1 (worst-case)

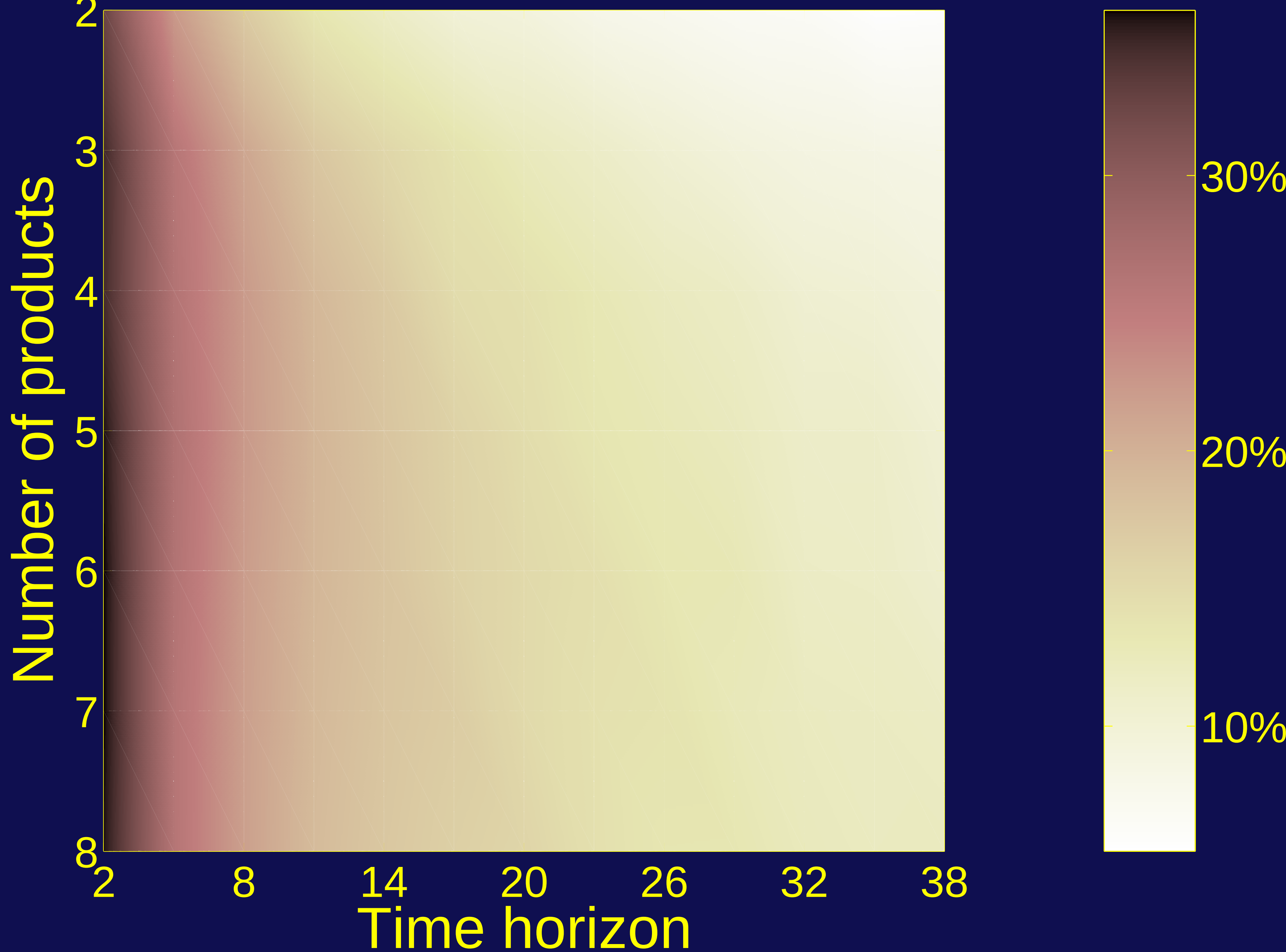
Average A-gap(MPI)



Average A-gap(RND)



Average A-gap(EDF)



Extensions of KPPI

- Geometric discounting of future
  - ⋆ slightly modified MPI's
- Stage-dependent probabilities
  - ⋆ a regularity condition on probabilities is needed
- From real-world applications:
  - ⋆ add randomly arriving perishable items
  - ⋆ items with multiple units

Thank you!