Learning to Backfill (when you have a simulator)

Eric Gaussier, David Glesser, Denis Trystram, Valentin Reis, and many more

Université Grenoble Alpes

firstname.lastname@imag.fr

December 16, 2015

Problem Statement

On-line non-preemptive execution of a stream of unconstrained rigid parallel jobs on a HPC system (incl. topology).

Simplified Problem

On-line non-preemptive execution of a stream of independent rigid parallel jobs on a HPC platform represented by a pool of m identical undistinguishable resources.

Simplified Problem, cont.

Job j from the submission stream has:

- Submission date r_j unknown in advance.
- Resource requirement q_j known at submission date.
- Running time p_j unknown in advance.
- Requested running time know at submission date (**bad** upper bound on p_j).
- Additional metadata and historical information.

Performance Metric (a.k.a. Scheduling Objective)

$$\mathrm{bsld}_j = \max\left(\frac{wait_j + p_j}{\max(p_j, \tau)}, 1\right) \quad \tau = 600s(10mn)$$

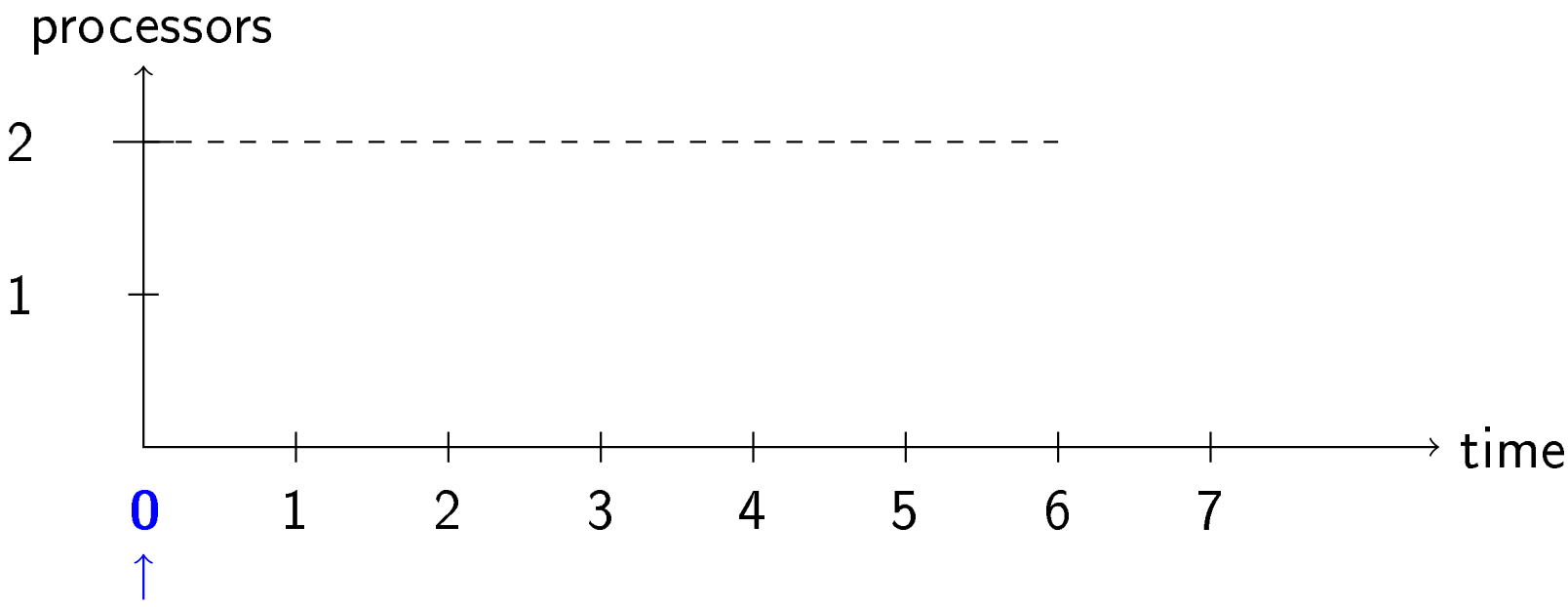
Performance Metric (a.k.a. Scheduling Objective)

$$\text{bsld}_j = \max\left(\frac{wait_j + p_j}{\max(p_j, \tau)}, 1\right) \quad \tau = 600s(10mn)$$

For k jobs $\{1, \ldots, k\}$, we are interested in the Average Bounded Slowdown:

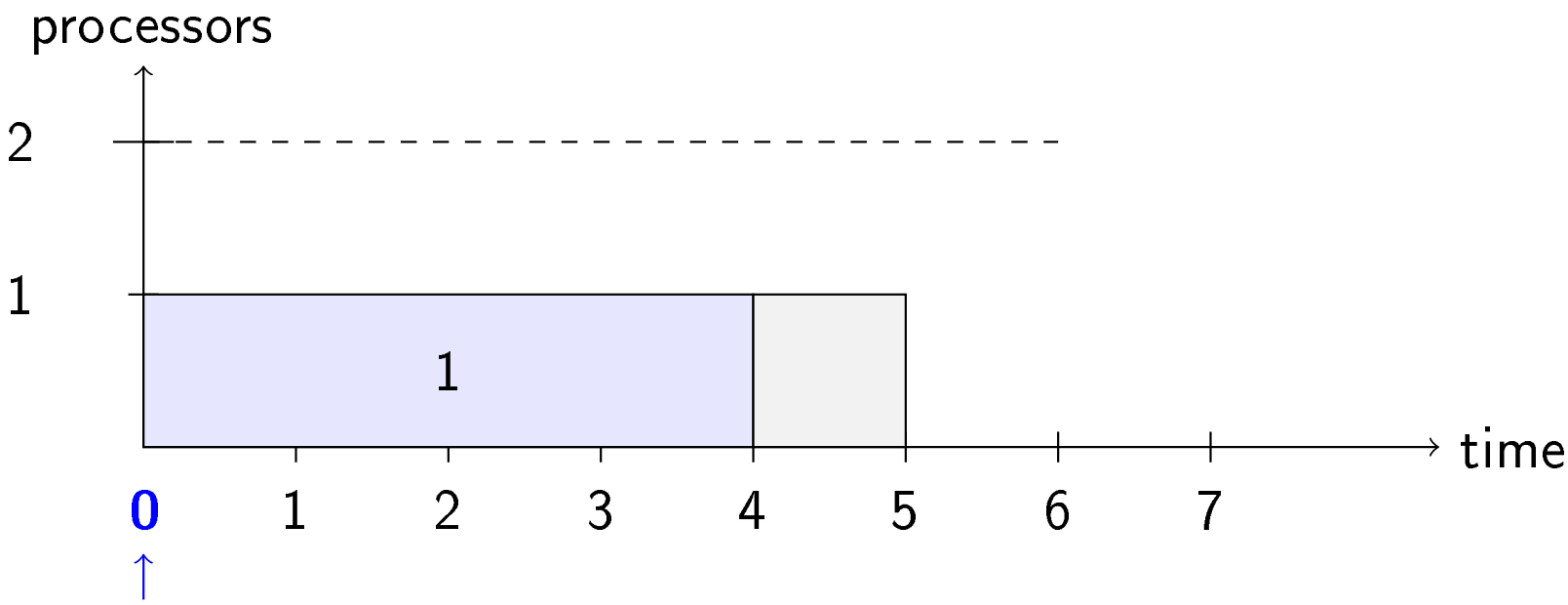
$$\frac{1}{k}\sum_{j=1}^{k} \max\left(\frac{wait_j + p_j}{\max(p_j, \tau)}, 1\right)$$

The FCFS with EASY-Backfilling policy



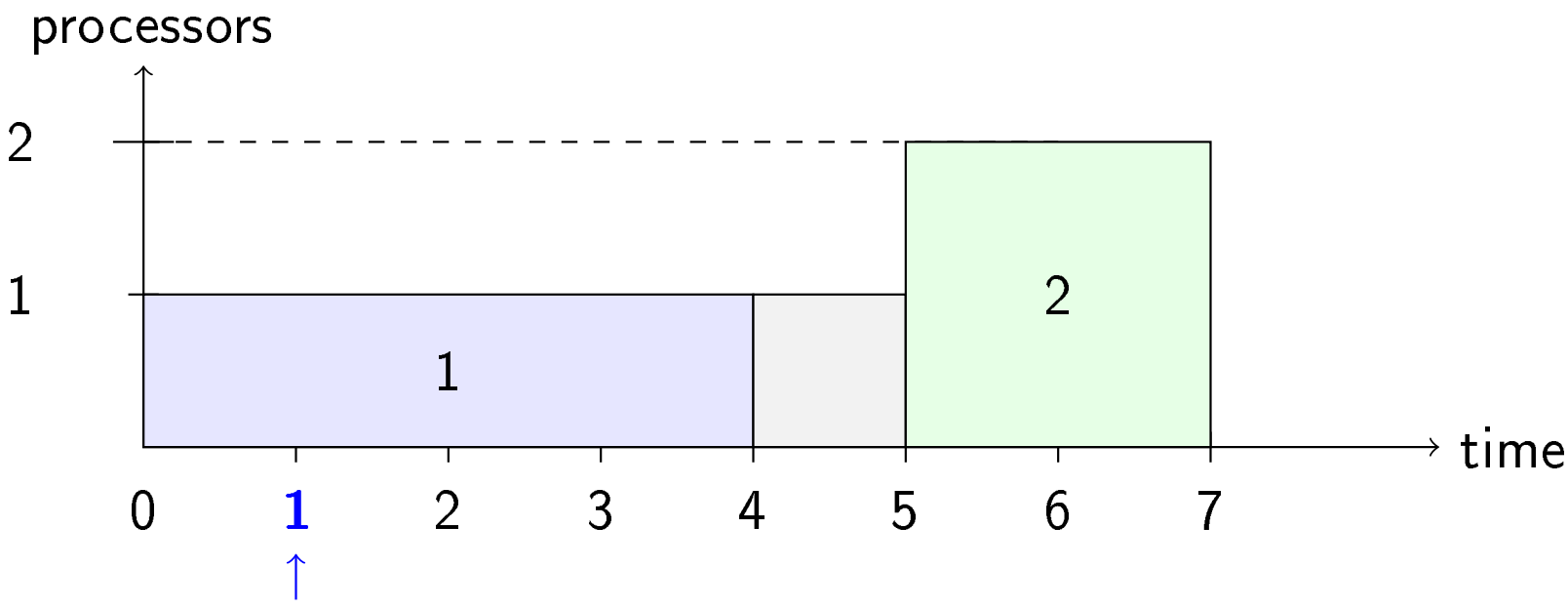
Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



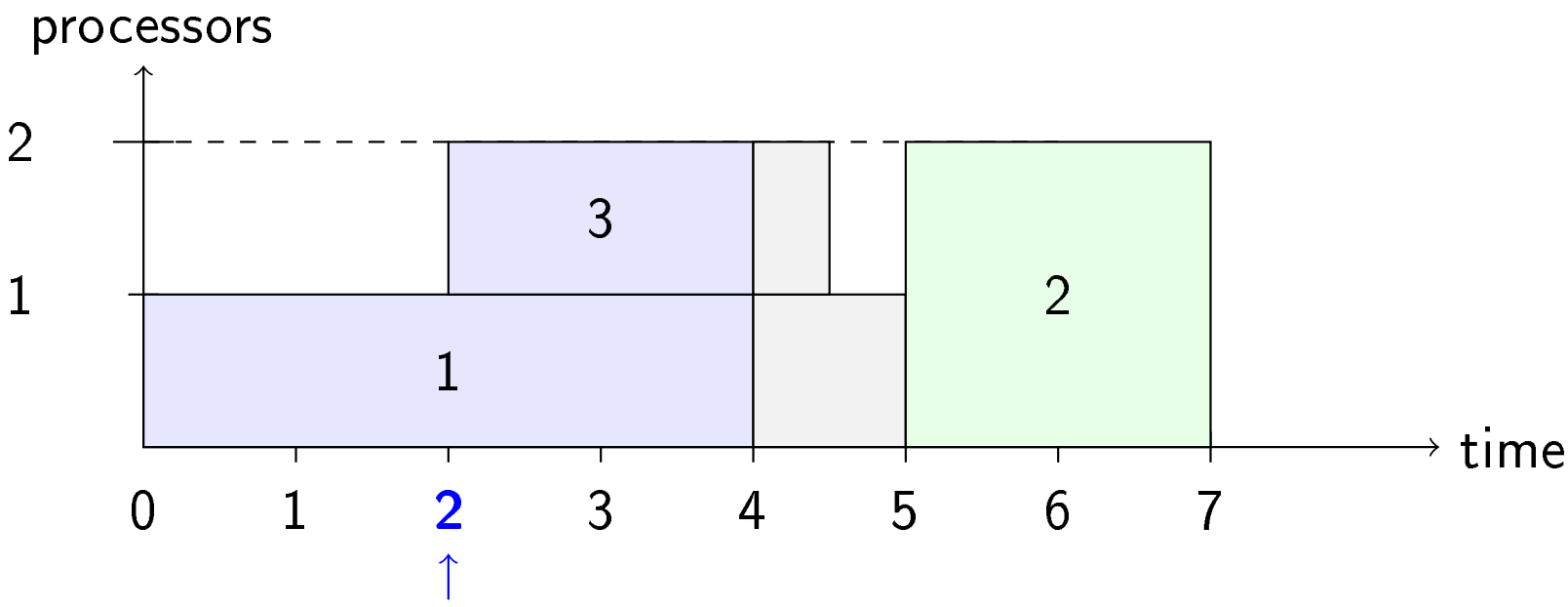
Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



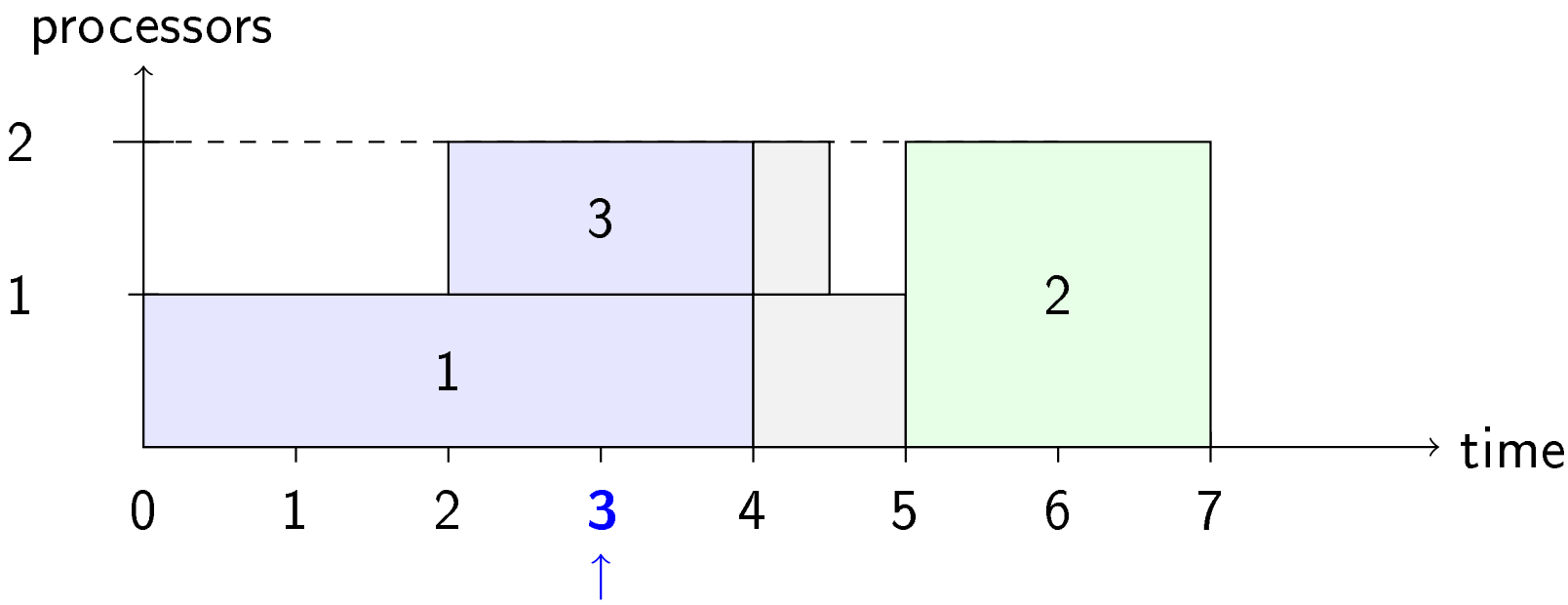
Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



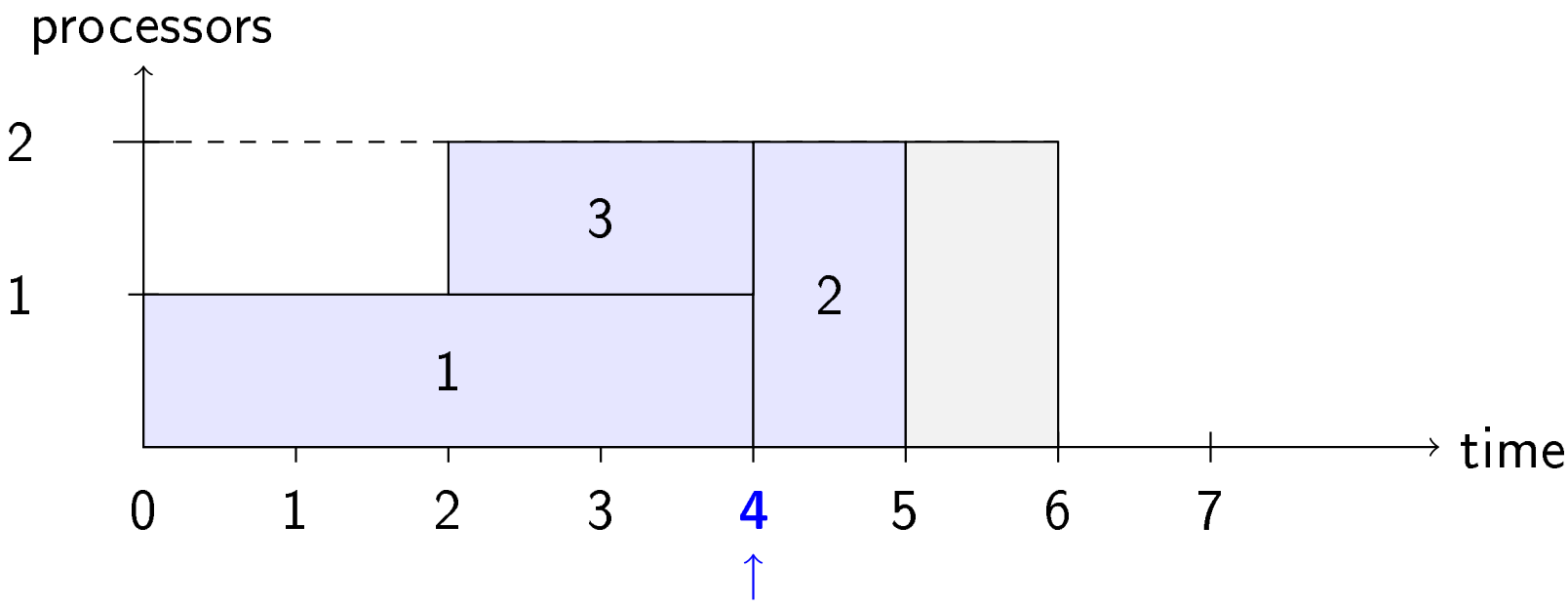
Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



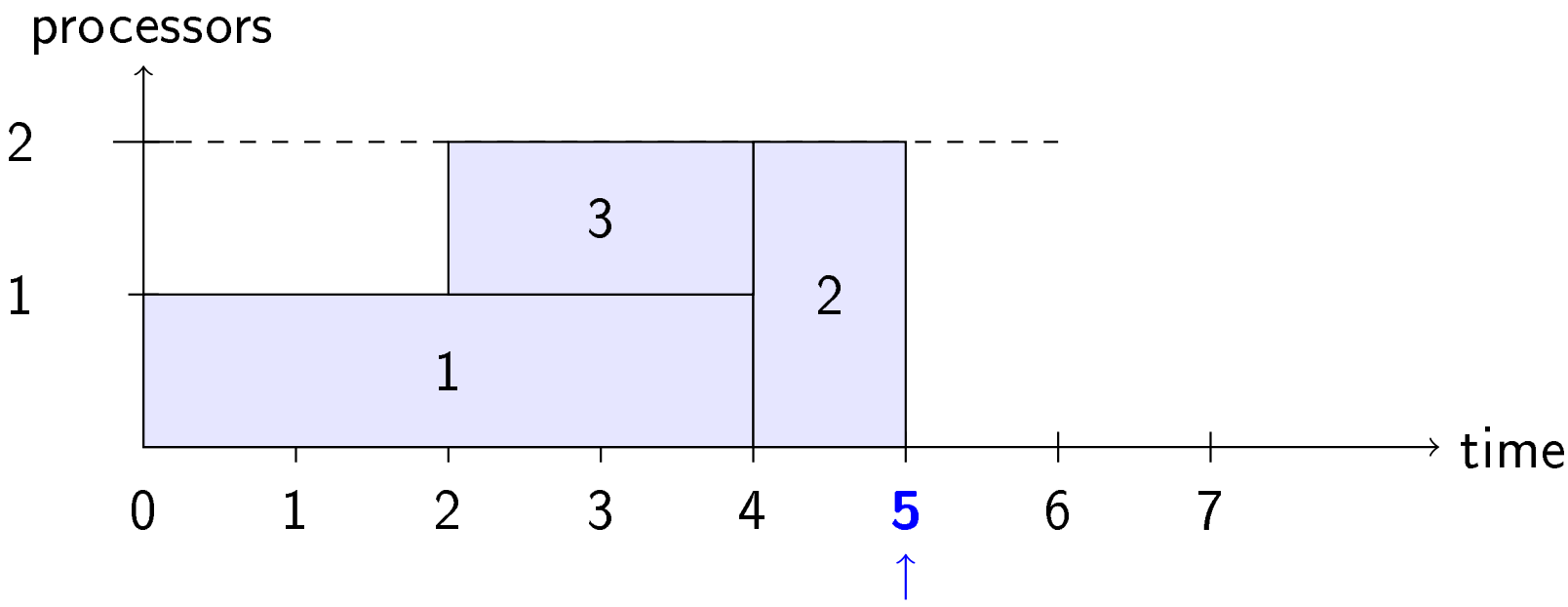
Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

The FCFS with EASY-Backfilling policy



Submission dates	$r_1 = 0$	$r_2 = 1$	$r_3 = 2$
Resource requirements	$q_1 = 1$	$q_2 = 2$	$q_3 = 1$
Requested running times	$\widetilde{p_1} = 5$	$\widetilde{p_2} = 2$	$\widetilde{p_3} = 2.5$
Running time	$p_1 = 4$	$p_2 = 1$	$p_3 = 2$

Existing solutions

Production, *do provide no-starvation guarantees*:

- EASY ("Agressive") Backfilling -*SLURM*
- Conservative Backfilling -*OAR*
- ...

Research, *might not provide no-starvation guarantees*:

- EASY++
- Runtime-Prediction based approaches (us)
- Probabilistic Backfilling
- ...
- (and many approaches that have the bad taste of running optimization solvers at decision time)

Performance unstability

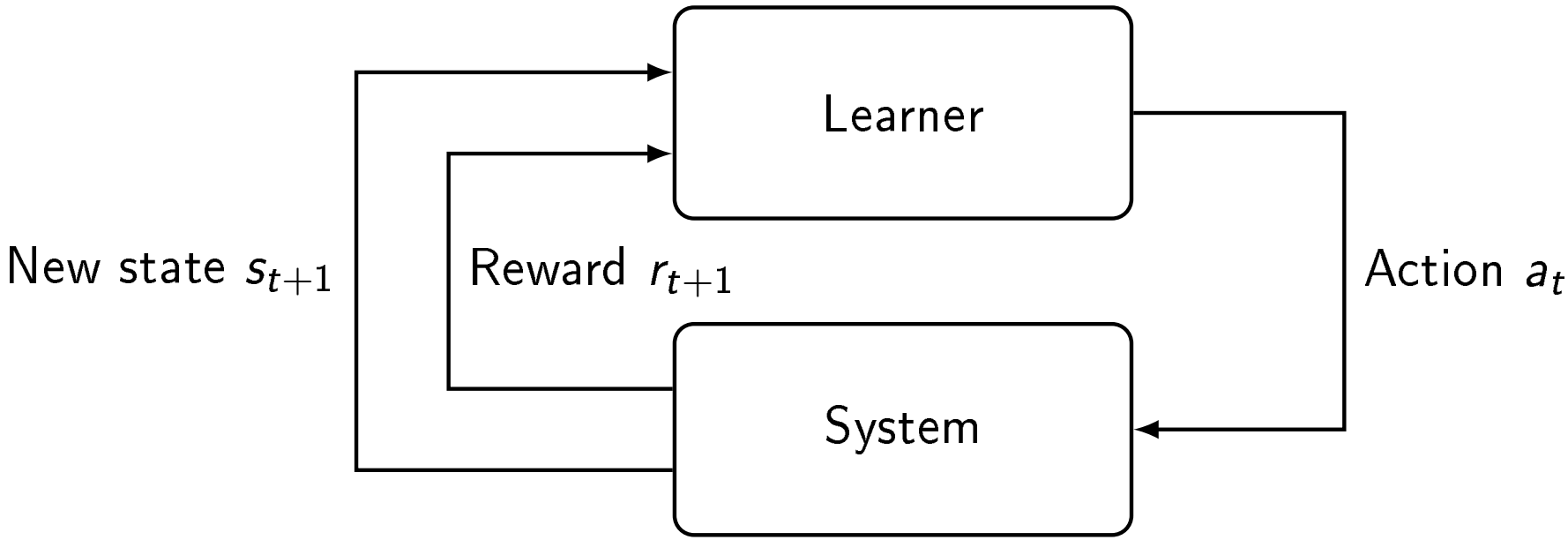
Table: Average Bounded Slowdown performance - Percentage of impromement w.r.t. EASY-BF in parenthesis.

Log	EASY	EASY++
KTH-SP2	92.6	63.5 (31%)
CTC-SP2	49.6	85.8 (-72%)
SDSC-SP2	87.9	79.4 (10%)
SDSC-BLUE	36.5	21.0 (42%)
Curie	202.1	193.5 (04%)
Metacentrum	97.6	87.2 (11%)

Our approach

Limit our decisions to the EASY-Backfill "class of policies". Search for the best job to backfill using a Reinforcement Learning (RL) approach.

RL crash course (1/4)



RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

$$(S, A, P, R, \gamma)$$

RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

$$(S, A, P, R, \gamma)$$

- S state space

RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

$$(S, A, P, R, \gamma)$$

- S state space
- A action space

RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

$$(S, A, P, R, \gamma)$$

- S state space
- A action space
- $P : s, a \in S \times A \rightarrow p(.|s, a) \in \Delta(S)$: unknown markovian transition probabilities.

($\Delta(\mathbb{X})$ is the set of all probability distributions on $\mathbb{X}$)

RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

$$(S, A, P, R, \gamma)$$

- S state space
- A action space
- $P : s, a \in S \times A \rightarrow p(.|s, a) \in \Delta(S)$: unknown markovian transition probabilities.
- $R : s, a, s' \in S \times A \times S \rightarrow p(.|s, a, s') \in \Delta(\mathbb{R})$: unknown stochastic rewards.

($\Delta(\mathbb{X})$ is the set of all probability distributions on $\mathbb{X}$)

RL crash course, cont. (2/4)

Classical RL approach : assume system to be a discounted infinite horizon markov decision process with stochastic reward and transition functions:

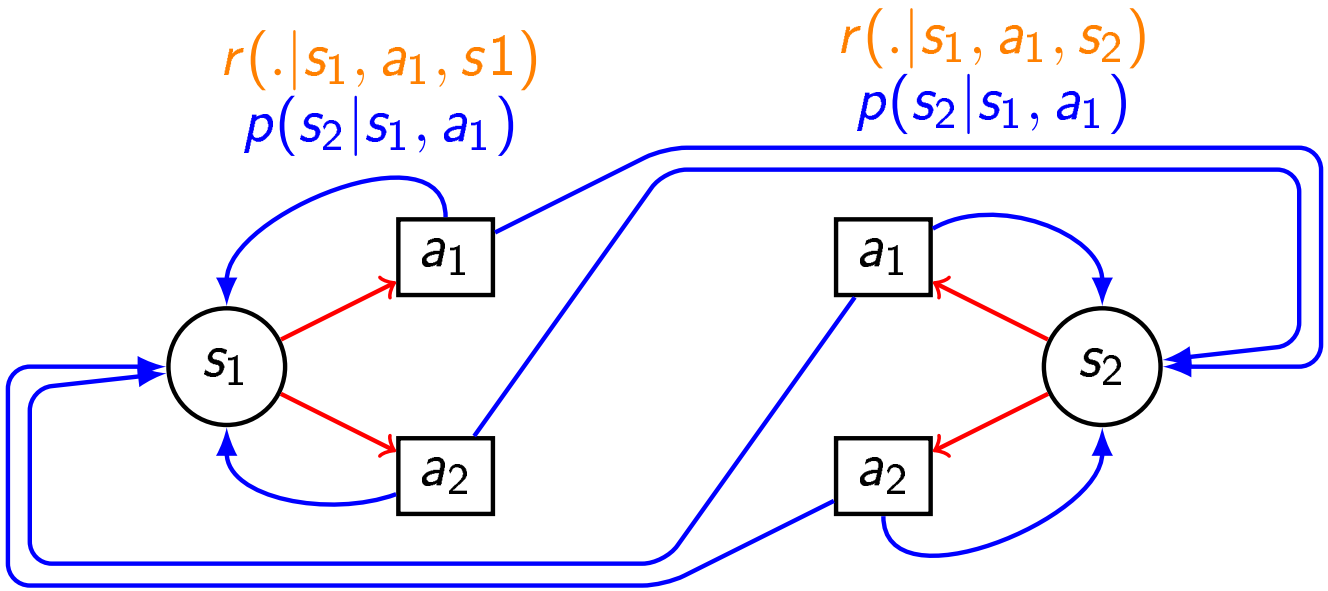
$$(S, A, P, R, \gamma)$$

- S state space
- A action space
- $P : s, a \in S \times A \rightarrow p(.|s, a) \in \Delta(S)$: unknown markovian transition probabilities.
- $R : s, a, s' \in S \times A \times S \rightarrow p(.|s, a, s') \in \Delta(\mathbb{R})$: unknown stochastic rewards.
- γ is a discount factor.

($\Delta(\mathbb{X})$ is the set of all probability distributions on $\mathbb{X}$)

RL crash course, cont. (3/4)

e.g.:



RL crash course, end. (4/4)

It is possible to find a $\pi : S \to A$ (possibly randomized) that maximizes expected (discounted) rewards:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{\pi(s_t)}(s_t, s_{t+1})\right]$$

RL crash course, end. (4/4)

It is possible to find a $\pi : S \to A$ (possibly randomized) that maximizes expected (discounted) rewards:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{\pi(s_t)}(s_t, s_{t+1})\right]$$

Algorithms for small $|A|$ and $|S|$:
Q-learning, policy iteration, value iteration.

RL crash course, end. (4/4)

It is possible to find a $\pi : S \to A$ (possibly randomized) that maximizes expected (discounted) rewards:

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{\pi(s_t)}(s_t, s_{t+1})\right]$$

Algorithms for small $|A|$ and $|S|$:
Q-learning, policy iteration, value iteration.
Algorithms for large $|A|$ and $|S|$:
approximate value iteration, approximate policy iteration, policy gradient, conservative policy iteration

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)
- R noisy and not static (backfilling transitions, varying load)

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)
- R noisy and not static (backfilling transitions, varying load)

Good News:

- Joint feature mapping $\Phi(s, j) \in \mathbb{R}^n$ for a state $s \in S$ and an action $j \in J$. $n \approx 100$

Example $\Phi(a, s)$

Feature	Description
runtime_j	User estimated runtime of job j
procs_j	Number of processors required by job j
$t\text{reservation}_s$	Time of the projected start of the EASY reservation job
$\text{procs_reservation}_s$	Number of processors required by the EASY reservation job
$\text{runtime_reservation}_s$	User estimated runtime of the EASY reservation job
num_running_s	Number of jobs currently running in the system
procs_running_s	Number of processors currently used in the system
queue_length_j	Length of the backfilling queue

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)
- R noisy and not static (backfilling transitions, varying load)

Good News:

- Joint feature mapping $\Phi(s, j) \in \mathbb{R}^n$ for a state $s \in S$ and an action $j \in J$. $n \approx 100$

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)
- R noisy and not static (backfilling transitions, varying load)

Good News:

- Joint feature mapping $\Phi(s, j) \in \mathbb{R}^n$ for a state $s \in S$ and an action $j \in J$. $n \approx 100$
- Simulator.

Our RL problem

Bad News:

- $|S|$ large. (job queue, partial schedules)
- A varies with time. (list of jobs to backfill)
- P is not well approximated by a *i.i.d.* law. (hidden dependencies between jobs)
- R noisy and not static (backfilling transitions, varying load)

Good News:

- Joint feature mapping $\Phi(s, j) \in \mathbb{R}^n$ for a state $s \in S$ and an action $j \in J$. $n \approx 100$
- Simulator.
- System dynamic partially known.

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)

Preference Learning Crash Course (1/4)

You obtain preferences on a set of vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ coming from some source. When x_i is "preffered" to x_j, we write:

$$x_i \succeq x_j$$

Preference Learning Crash Course (1/4)

You obtain preferences on a set of vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ coming from some source. When x_i is "preffered" to x_j, we write:

$$x_i \succeq x_j$$

You want to find a linear scoring function on x:

$$f(x) = w^\intercal x$$

Preference Learning Crash Course (1/4)

You obtain preferences on a set of vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ coming from some source. When x_i is "preffered" to x_j, we write:

$$x_i \succeq x_j$$

You want to find a linear scoring function on x:

$$f(x) = w^\mathsf{T} x$$

which:

- Minimizes the number of violated preferences $|\{(i,j) \in [\![1,n]\!] \mid f(x_i) \leq f(x_j), x_i \succeq x_j\}|$ (optimization)

Preference Learning Crash Course (1/4)

You obtain preferences on a set of vectors $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ coming from some source. When x_i is "preffered" to x_j, we write:

$$x_i \succeq x_j$$

You want to find a linear scoring function on x:

$$f(x) = w^\intercal x$$

which:

- Minimizes the number of violated preferences $|\{(i,j) \in [\![1,n]\!] \mid f(x_i) \leq f(x_j), x_i \succeq x_j\}|$ (optimization)
- Performs well respect to new preference from the same source, e.g. iid source. (learning)

Preference Learning Crash Course (2/4)

classical simple solution (among others..): SVM-Rank [1]

$$\begin{aligned} \underset{w \in \mathbb{R}^n, \xi_{i,j} \in \mathbb{R}}{\text{argmin}} \quad & \frac{1}{2} \|w\|_2^2 + C \sum \xi_{i,j} \\ \text{s.t.} \quad & \forall (i,j), x_i \succeq x_j : w^\mathsf{T} x_i \geq w^\mathsf{T} x_j + 1 - \xi_{i,j} \end{aligned}$$

- Optimizes (an upper bound on) the number of violated preferences from $\mathbb{Q}$.
- Generalizes well thanks to $\|w\|$. (Generalization bounds exist)

Preference Learning Crash Course (3/4)

An extension: constraints are weighted with $\Delta_{i,j} > 0$ and you are interested in minimizing the sum of the weights of the violated constraints. Slack variable rescaling SVM-Rank:

$$\begin{aligned} \underset{w \in \mathbb{R}^n, \xi_{i,j} \in \mathbb{R}}{\text{argmin}} \quad & \frac{1}{2} \|w\|_2^2 + C \sum \xi_{i,j} \\ \text{s.t.} \quad & \forall (i,j), x_i \succeq x_j : w^\intercal x_i \geq w^\intercal x_j + 1 - \frac{\xi_{i,j}}{\Delta_{i,j}} \end{aligned}$$

- Generalization bounds still exist

Preference Learning Crash Course (4/4)

With preferences arriving as a stream:

$$\underset{w \in \mathbb{R}^n}{\operatorname{argmin}} \frac{1}{2} \|w\|_2^2 + C \sum \frac{w^\intercal (x_j - x_i) + 1}{\Delta_{i,j}}$$

Online learning can be done i.e. with any online convex optimization like GIGA[2] (if we want regret bounds) or using a variant of SGD[3] (if we assume stochastic source).

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)
- P is not well approximated by a *i.i.d.* random variable. (?)

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)
- P is not well approximated by a *i.i.d.* random variable. (?)
- R noisy (simulated rollouts)

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)
- P is not well approximated by a *i.i.d.* random variable. (?)
- R noisy (simulated rollouts) and not static (preference learning in policy space)

Our RL problem, tricks and questions

- $|S|$ large. (approximate strategy using $\Phi(s, j)$)
- A varies with time. (preference learning on $\Phi(s, j)$)
- P is not well approximated by a *i.i.d.* random variable. (?)
- R noisy (simulated rollouts) and not static (preference learning in policy space)
- System dynamics partially known. $\rightarrow$ approximate dynamic programming?

Possible approaches (draft)

3 Approaches:

- Preference-based Approximate Policy iteration with simulated rollouts.
- Preference-based Off-policy learning (approximate Q-learning).
- Our approach - 2-stage policy ranking with simulated rollouts.

Our Approach (draft)

Goal: learn and maintain parameter $w^{(r)}$ of a job scoring function $f(\phi) = w^{(r)\mathsf{T}}\phi$.

Our Approach (draft)

Goal: learn and maintain parameter $w^{(r)}$ of a job scoring function $f(\phi) = w^{(r)\mathsf{T}}\phi$.

(1) simulation Generate and evaluate policies $\pi^{(r)}_{k=1\ldots K}$ every t_{replay} on stored job arrivals. (r-th round of simulation using stored job arrivals).

Our Approach (draft)

Goal: learn and maintain parameter $w^{(r)}$ of a job scoring function $f(\phi) = w^{(r)\mathsf{T}}\phi$.

(1) simulation Generate and evaluate policies $\pi^{(r)}_{k=1\ldots K}$ every t_{replay} on stored job arrivals. (r-th round of simulation using stored job arrivals).

(2) policy evaluation/learning Learn a job scoring model $f_k(\phi) = w_k^{\mathsf{T}}\phi$ for each policy $\pi^{(r)}_{k=1\ldots K}$ using preference learning (Solve K SVM-Rank optimization subproblems)

Our Approach (draft)

Goal: learn and maintain parameter $w^{(r)}$ of a job scoring function $f(\phi) = w^{(r)\mathsf{T}}\phi$.

(1) simulation Generate and evaluate policies $\pi^{(r)}_{k=1\ldots K}$ every t_{replay} on stored job arrivals. (r-th round of simulation using stored job arrivals).

(2) policy evaluation/learning Learn a job scoring model $f_k(\phi) = w_k^{\mathsf{T}}\phi$ for each policy $\pi^{(r)}_{k=1\ldots K}$ using preference learning (Solve K SVM-Rank optimization subproblems)

(3) learning and maximizing policy return estimate Use weighted preference learning on the $w_k^{(r)}$ to learn a policy return estimate $V^{(r)}(w) = \alpha^{(r)\mathsf{T}}w$. Finally, maximize it on w to get $w^{(r)}$.

Policy Generation (DRAFT)

Policy	Description
EASY-FCFS	FCFS backfill order.
EASY-SJBF	SJBF backfill order.
EASY-*random*	Random backfill order.
EASY-w	The current strategy: greedy backfill with respect to $f(\phi) = w^{(r)\mathsf{T}}\phi$.
EASY-w_{random}	Greedy backfill with respect to $f(\phi) = w_{\text{random}}^{\mathsf{T}}\phi$, where w_{random} is taken "randomly" in $\mathbb{R}^n$.
EASY-w_{close}	Greedy backfill with respect to $f(\phi) = w_{\text{random}}^{\mathsf{T}}\phi$, where w_{random} is taken "close to" $w^{(r)}$.

Policy Evaluation (1/2) - (draft)

For each policy $\pi_k^{(r)}$ with decisions d in $\left\{1 \dots D_k^{(r)}\right\}$.
We get:

- $v_{\pi_k}^{(r)}$ – Average Bounded Slowdown [4] of the scheduled jobs
- "preferences" list:

$$\forall j_l \in J_{k,d}^{(r)} \setminus \{j_{d,b}\}, \quad \Phi(s_d^{(r)}, j_{d,b}) \text{ preffered to } \Phi(s_d^{(r)}, j_l)$$

where $J_{k,d}^{(r)}$ is the list of eligible jobs for backfilling at decision d of the replay of policy $\pi_k^{(r)}$

Policy Evaluation (2/2) - (draft)

We learn for each $\pi_k^{(r)}$ a scoring function $f_k(\phi) = w_k^{(r)\mathsf{T}}\phi$ using SVM-Rank:

$$w_k^{(r)} = \underset{w \in \mathbb{R}^n, \xi_{k,d,l} \in \mathbb{R}}{\operatorname{argmin}} \quad \frac{1}{2}\|w\|_2^2 + C_{\text{sub}} \sum \xi_{k,d,l}$$

such that:

$$\forall j_l \in J_{k,1} \setminus \{j_b\} : w^\mathsf{T}\Phi(s_1, j_b) \geq w^\mathsf{T}\Phi(s_1, j_l) + 1 - \xi_{k,1,l}$$

$$\dots$$

$$\forall j_l \in J_{k,D_k} \setminus \{j_b\} : w^\mathsf{T}\Phi(s_{D_k}, j_b) \geq w^\mathsf{T}\Phi(s_{D_k}, j_l) + 1 - \xi_{k,D_k,l}$$

Policy Return Estimate - (draft)

$$\alpha^{(r)} = \underset{\alpha \in \mathbb{R}^n, \xi_{i,j}^{(r)} \in \mathbb{R}}{\text{argmin}} \quad \frac{1}{2} \|\alpha\|_2^2 + C_{\text{sub}} \sum_{s=1}^{r} \sum_{i,j} \xi_{i,j}^{(s)}$$

such that

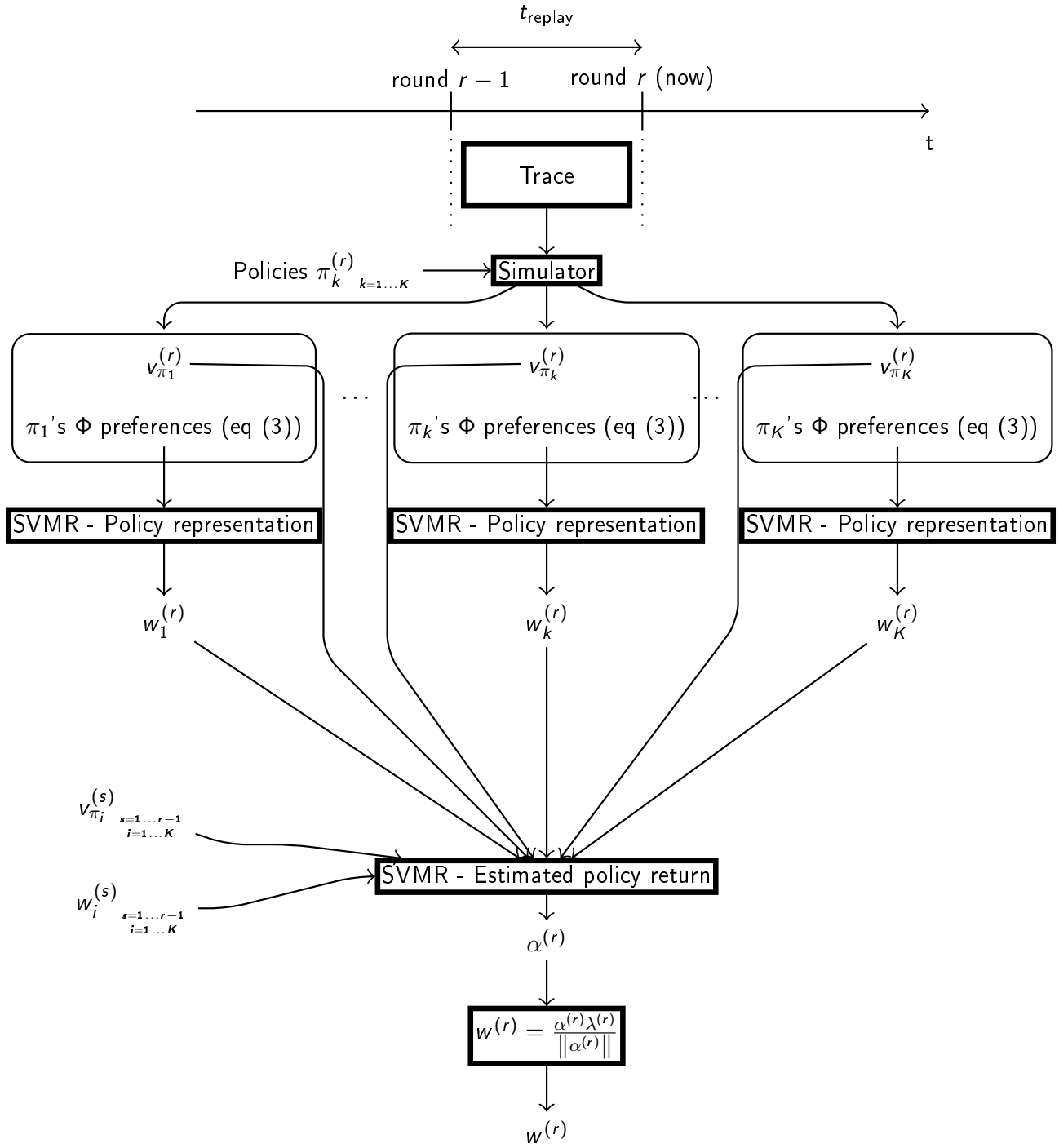
$$\forall i, j \in \{1 \dots K\} \quad \text{s.t.} \quad \delta v_{i,j}^{(1)} > 0 : \alpha^\intercal w_i^{(1)} \geq \alpha^\intercal w_j^{(1)} + 1 - \frac{\xi_{i,j}^{(1)}}{\left\| \delta v_{i,j}^{(1)} \right\|_2}$$

$$\dots$$

$$\forall i, j \in \{1 \dots K\} \quad \text{s.t.} \quad \delta v_{i,j}^{(r)} > 0 : \alpha^\intercal w_i^{(r)} \geq \alpha^\intercal w_j^{(r)} + 1 - \frac{\xi_{i,j}^{(r)}}{\left\| \delta v_{i,j}^{(r)} \right\|_2}$$

Where:

$$\forall i \in \{1 \dots r\}, \delta v_{i,j}^{(i)} = \frac{v_{\pi_i}^{(i)} - v_{\pi_j}^{(i)}}{\max_l \quad v_{\pi_l}^{(i)}}$$


t_{replay}
round $r-1$
round r (now)
t
Trace
Policies $\pi_{k\ k=1\ldots K}^{(r)}$
Simulator
$v_{\pi_1}^{(r)}$
π_1's Φ preferences (eq (3))
$v_{\pi_k}^{(r)}$
π_k's Φ preferences (eq (3))
$v_{\pi_K}^{(r)}$
π_K's Φ preferences (eq (3))
SVMR - Policy representation
SVMR - Policy representation
SVMR - Policy representation
$w_1^{(r)}$
$w_k^{(r)}$
$w_K^{(r)}$
$v_{\pi_i\ s=1\ldots r-1\ i=1\ldots K}^{(s)}$
$w_{i\ s=1\ldots r-1\ i=1\ldots K}^{(s)}$
SVMR - Estimated policy return
$\alpha^{(r)}$
$w^{(r)} = \frac{\alpha^{(r)}\lambda^{(r)}}{\|\alpha^{(r)}\|}$
$w^{(r)}$

T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 133–142, ACM, 2002.

M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," 2003.

L. Bottou, "Stochastic learning," in *Advanced Lectures on Machine Learning* (O. Bousquet and U. von Luxburg, eds.), Lecture Notes in Artificial Intelligence, LNAI 3176, pp. 146–168, Berlin: Springer Verlag, 2004.

E. Frachtenberg and D. G. Feitelson, "Pitfalls in parallel job scheduling evaluation," in *Job Scheduling Strategies for Parallel Processing*, pp. 257–282, Springer, 2005.

The End

please give us feedback