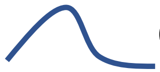
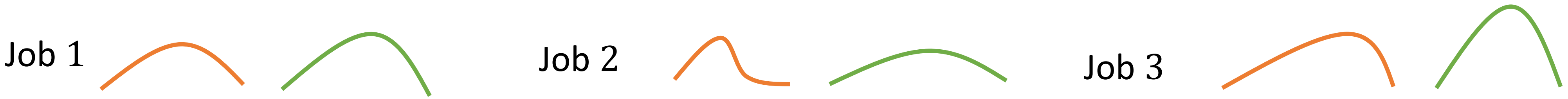
Configuration Balancing for Stochastic Request

Franziska Eberle, Anupam Gupta, Nicole Megow

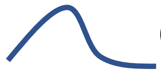
Benjamin Moseley, Rudy Zhou

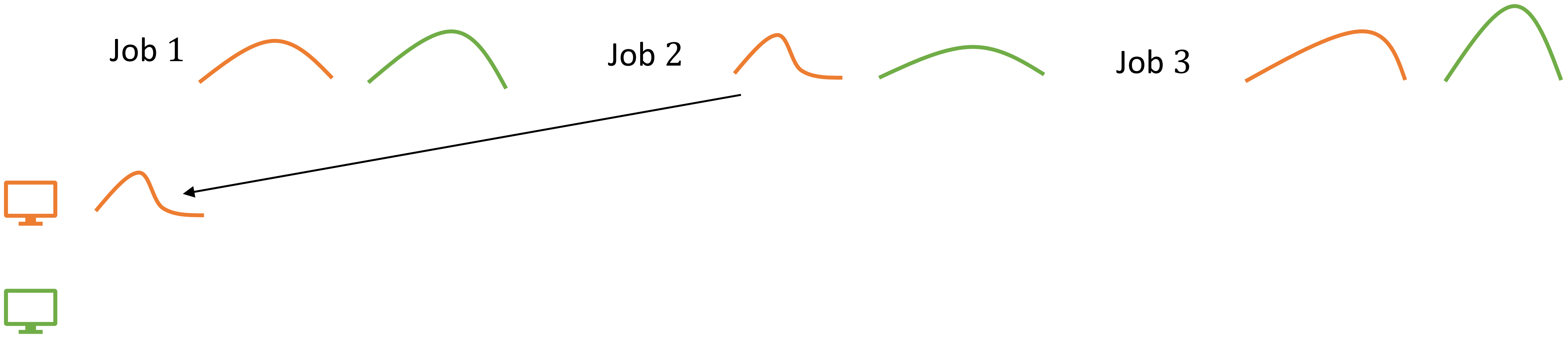
Stochastic Load Balancing

- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i

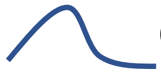


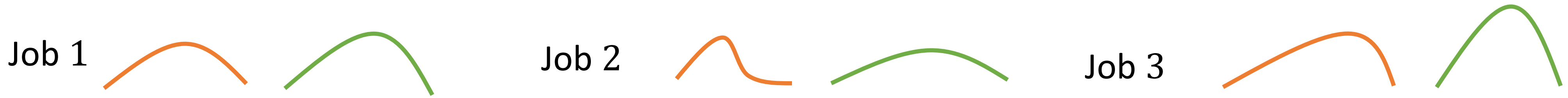
Stochastic Load Balancing

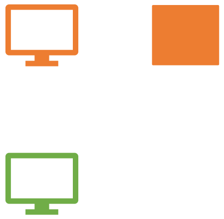
- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i



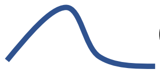
Stochastic Load Balancing

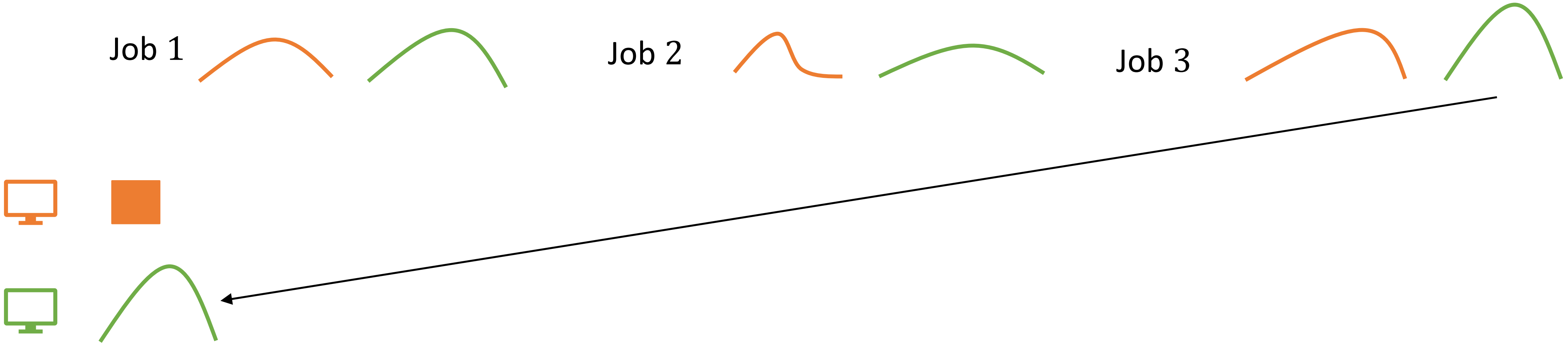
- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i



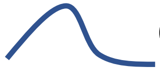


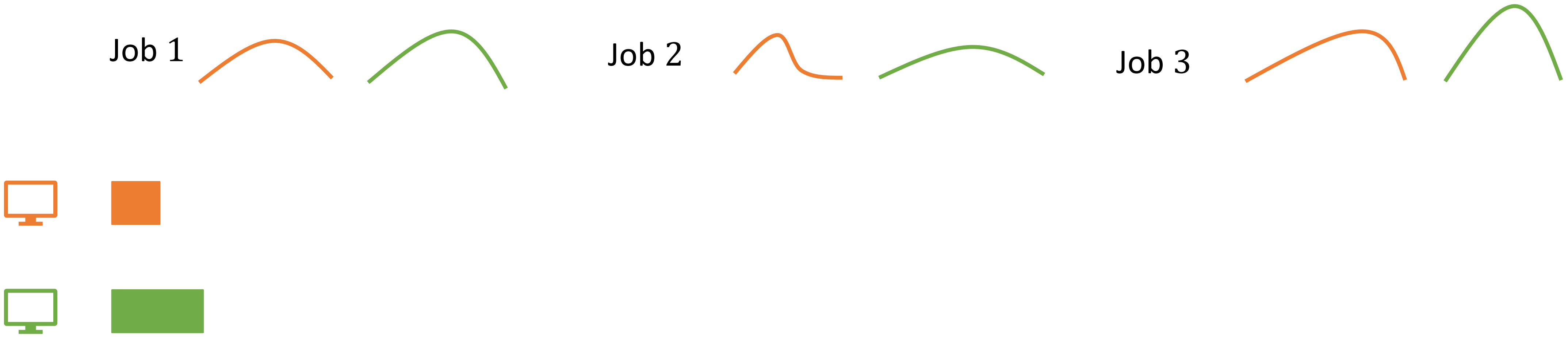
Stochastic Load Balancing

- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i

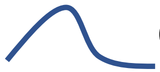


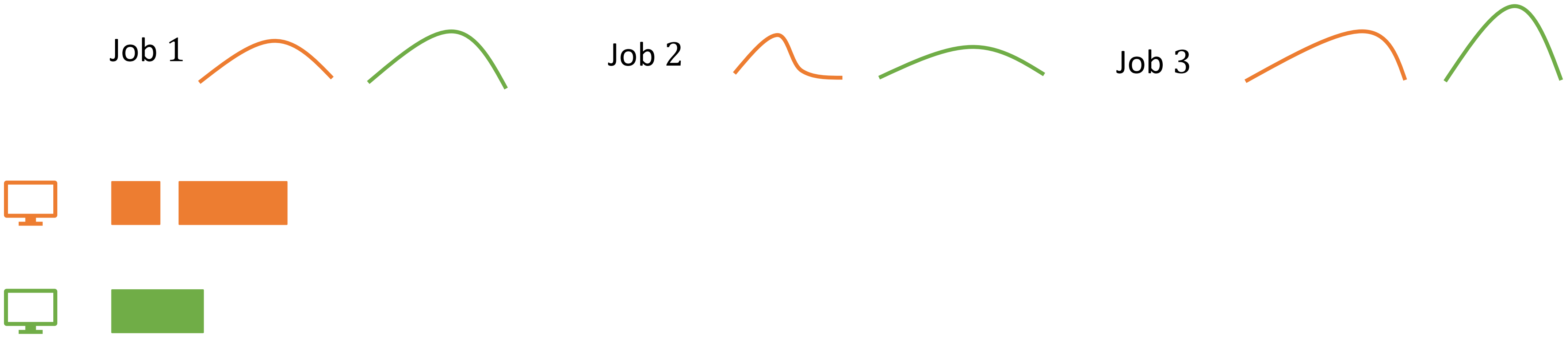
Stochastic Load Balancing

- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i

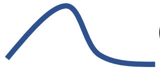


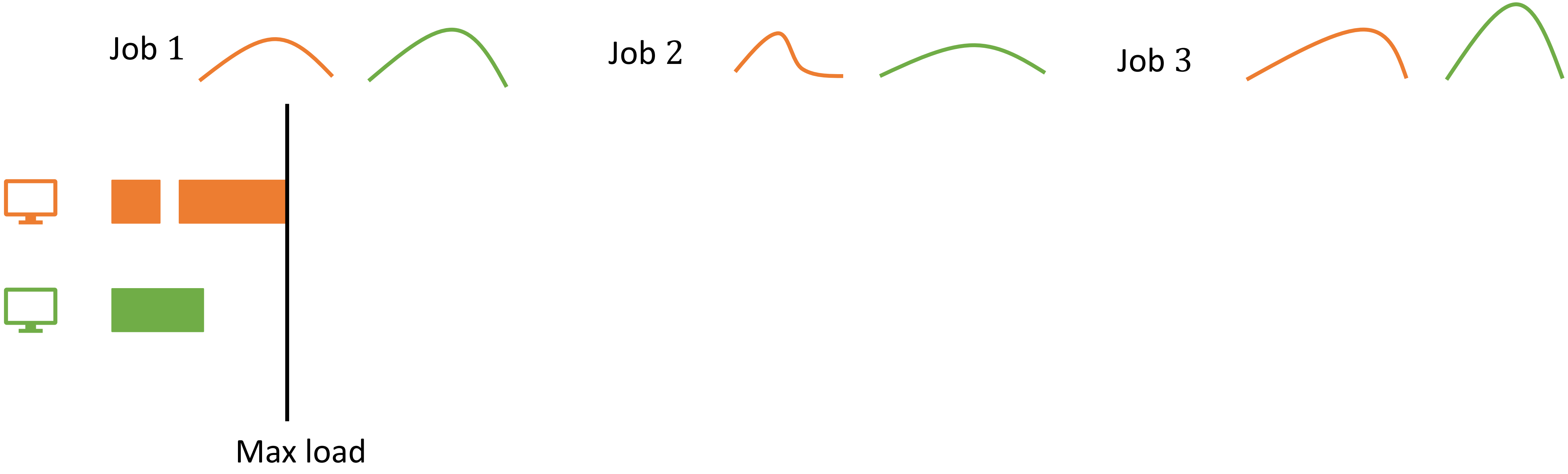
Stochastic Load Balancing

- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i



Stochastic Load Balancing

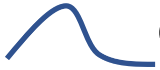
- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i

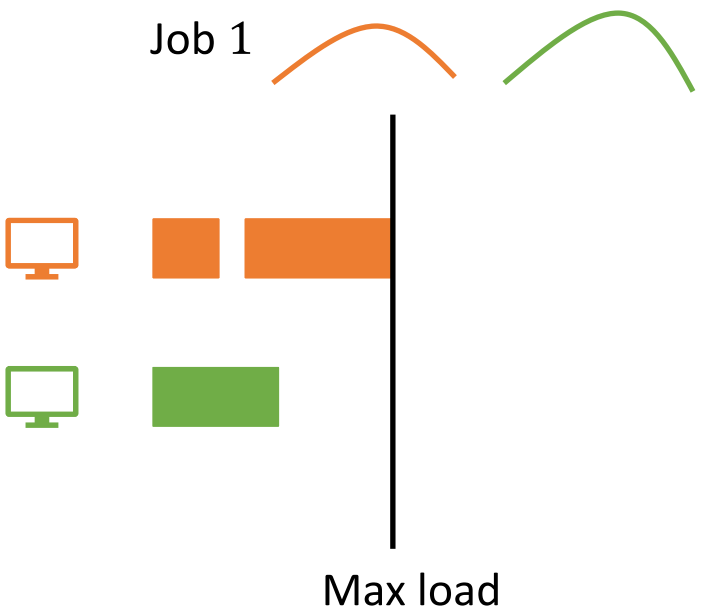


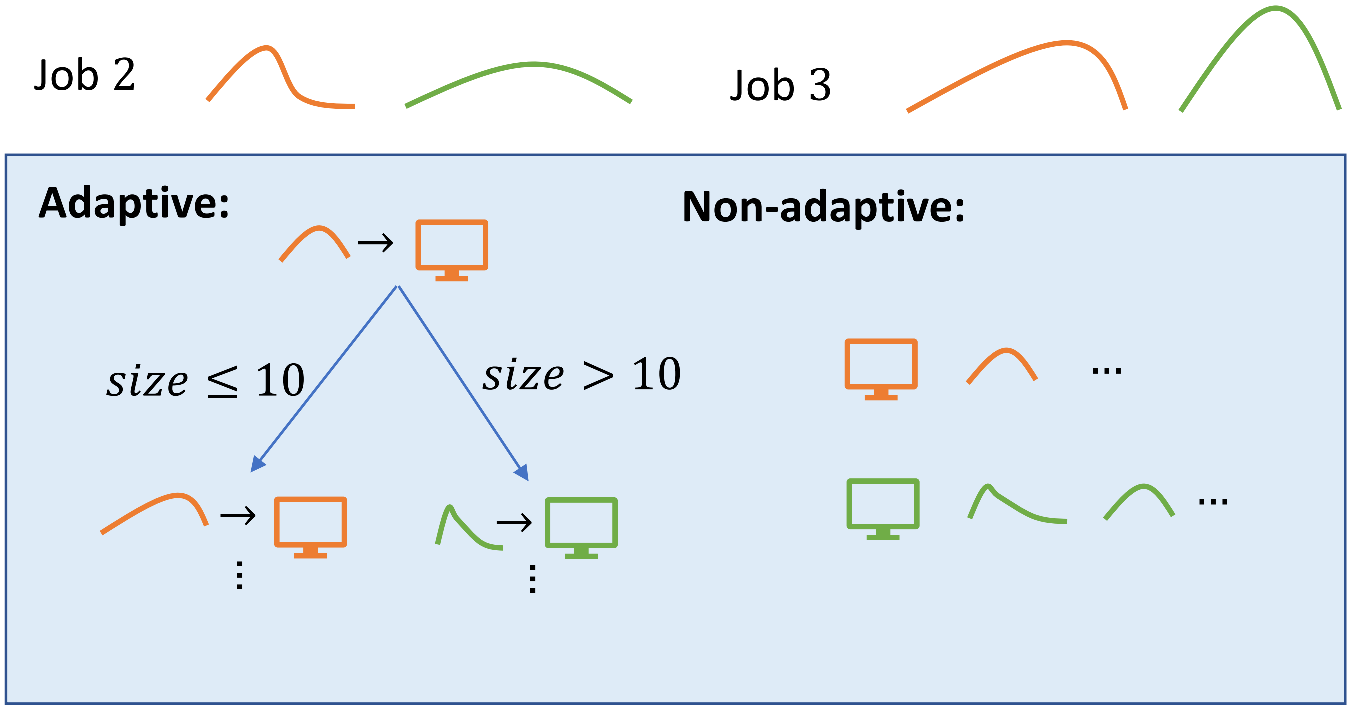
Minimize expected max load

…compared to optimal adaptive policy

Stochastic Load Balancing

- m unrelated machines
- n jobs with stochastic sizes such that job j has size $X_{ij} \sim$  on machine i





Minimize expected max load

…compared to optimal adaptive policy

Related Work

- Deterministic setting well-studied
  - 2-approximation offline (LP rounding) [Lenstra, Shmoys, Tardos, Math. Prog. 1990]
  - $O(\log m)$-competitive online (potential function) [Aspnes, Azar, Fiat, Plotkin, Waarts, J. ACM 1997]
  - Variety of generalizations (multidimensional, norm objective, etc.)

Related Work

- Deterministic setting well-studied
  - 2-approximation offline (LP rounding) [Lenstra, Shmoys, Tardos, Math. Prog. 1990]
  - $O(\log m)$-competitive online (potential function) [Aspnes, Azar, Fiat, Plotkin, Waarts, J. ACM 1997]
  - Variety of generalizations (multidimensional, norm objective, etc.)
- Stochastic setting focused on non-adaptive policies
  - Non-adaptive algorithm that $O(1)$-approximates optimal non-adaptive policy (LP rounding + effective size) [Gupta, Kumar, Nagarajan, Shen, Math. Oper. Res. 2021]
  - Adaptivity gap is $\Omega(\frac{\log m}{\log \log m})$

Our Results

Theorem: There exists an efficient algorithm for stochastic load balancing on unrelated machines that $O(\frac{\log m}{\log \log m})$-approximates the optimal adaptive policy. Further, the algorithm is non-adaptive.

- Also give $O(1)$-approximate adaptive policy for related machines

Our Results

Theorem: There exists an efficient algorithm for stochastic load balancing on unrelated machines that $O(\frac{\log m}{\log \log m})$-approximates the optimal adaptive policy. Further, the algorithm is non-adaptive.

- Also give $O(1)$-approximate adaptive policy for related machines
- First general result for stochastic load balancing compared to optimal adaptive policy
- Gives tight upper bound on adaptivity gap
- Can be generalized to variety of other resource allocation problems and online setting

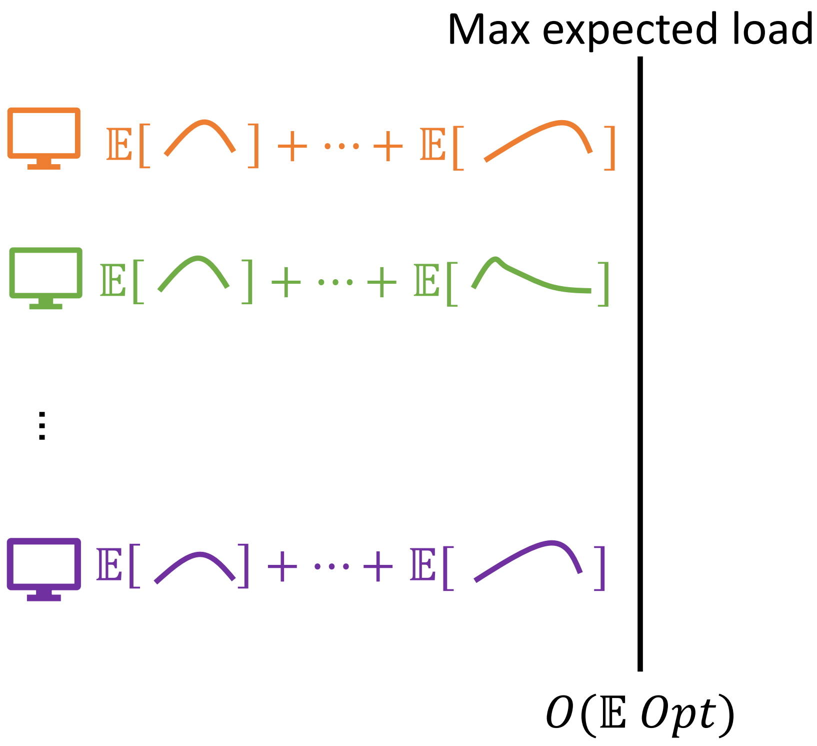
Our Results

Theorem: There exists an efficient algorithm for stochastic load balancing on unrelated machines that $O(\frac{\log m}{\log \log m})$-approximates the optimal adaptive policy. Further, the algorithm is non-adaptive.

- Also give $O(1)$-approximate adaptive policy for related machines
- First general result for stochastic load balancing compared to optimal adaptive policy
- Gives tight upper bound on adaptivity gap
- Can be generalized to variety of other resource allocation problems and online setting
- **New Idea:** Show that there exists near-optimal adaptive policy that behaves similarly to a non-adaptive policy

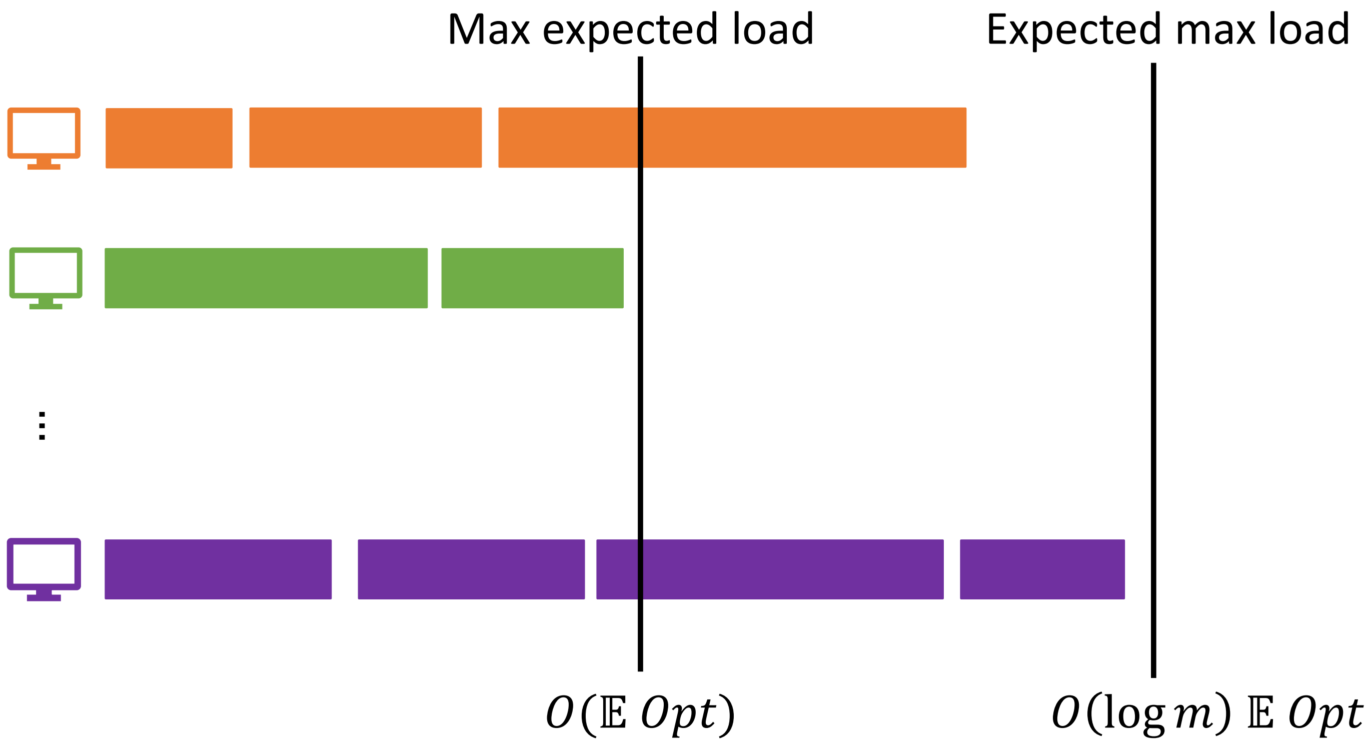
Warm Up: Small jobs

- Assume all jobs are small: $X_{ij} \in [0, \mathbb{E}\,Opt]$ for all i, j
- Suffices to control expected load on each machine



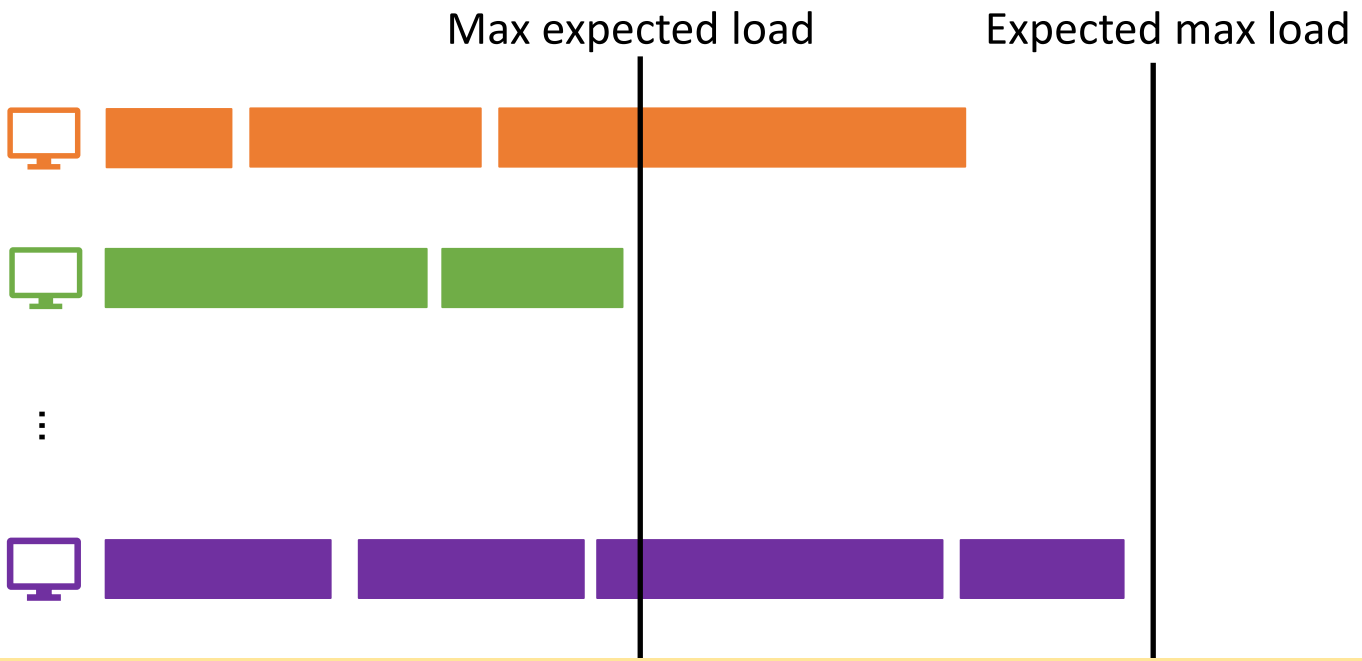
Warm Up: Small jobs

- Assume all jobs are small: $X_{ij} \in [0, \mathbb{E}\, Opt]$ for all i, j
- Suffices to control expected load on each machine (Concentration + Union)



Warm Up: Small jobs

- Assume all jobs are small: $X_{ij} \in [0, \mathbb{E}\, Opt]$ for all i, j
- Suffices to control expected load on each machine (Concentration + Union)

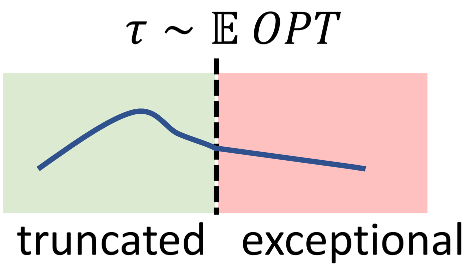


Main Challenge: How to handle jobs that aren't reasonably bounded?

Truncation

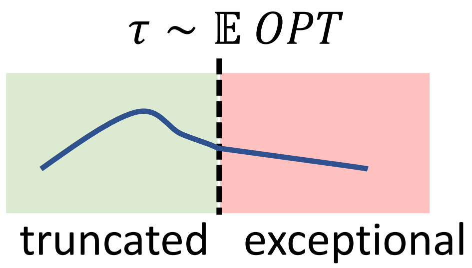
- Problem is easy if jobs are small $\Rightarrow$ make jobs small and deal with big jobs separately

Truncation



- Problem is easy if jobs are small ⇒ make jobs small and deal with big jobs separately
- Given truncation threshold τ,
- the truncated part of X_{ij} is: $X_{ij}^T = X_{ij} \cdot 1_{X_{ij} \leq \tau}$
- the exceptional part is: $X_{ij}^E = X_{ij} \cdot 1_{X_{ij} > \tau}$

Truncation



- Problem is easy if jobs are small ⇒ make jobs small and deal with big jobs separately
- Given truncation threshold τ,
- the truncated part of X_{ij} is: $X_{ij}^T = X_{ij} \cdot 1_{X_{ij} \leq \tau}$
- the exceptional part is: $X_{ij}^E = X_{ij} \cdot 1_{X_{ij} > \tau}$
- ⇒ handle truncated parts by controlling max expected load

Question: How to control expected max load of exceptional parts?

Exceptional parts

- Bound contribution of exceptional parts: $\mathbb{E}\left[\max_i \sum_{j\to i} X_{ij}^E\right]$
- Only have trivial upper bound:

$$\mathbb{E}\left[\max_i \sum_j X_{ij}^E \cdot 1_{j\to i}\right] \leq \sum_i \sum_j \mathbb{E}[X_{ij}^E \cdot 1_{j\to i}]$$

Exceptional parts

- Bound contribution of exceptional parts: $\mathbb{E}\left[\max_i \sum_{j\to i} X_{ij}^E\right]$
- Only have trivial upper bound:

$$\mathbb{E}\left[\max_i \sum_j X_{ij}^E \cdot 1_{j\to i}\right] \le \sum_i \sum_j \mathbb{E}[X_{ij}^E \cdot 1_{j\to i}]$$

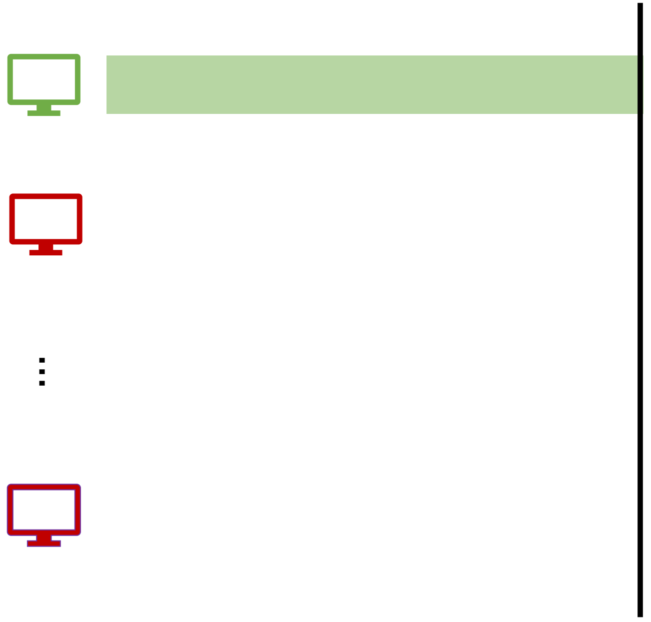
Total expected exceptional load

- **Algorithm goal:** assign jobs to machines non adaptively such that each machine has expected truncated load $O(\mathbb{E}\, Opt)$ and the total expected exceptional load $O(\mathbb{E}\, Opt)$

Question: Does there exist such an assignment?

Benefit of Adaptivity

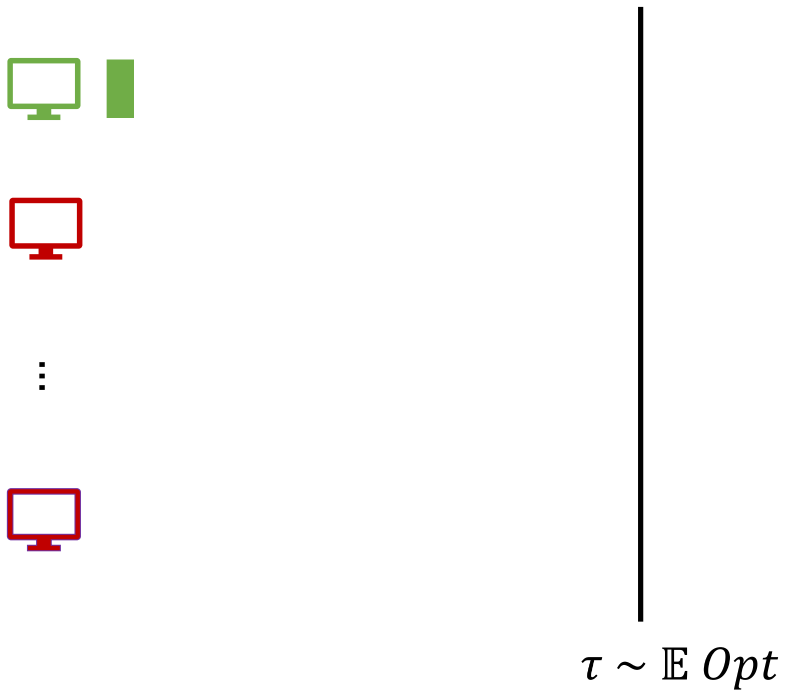
- One fast machine, $m-1$ slow
- One Bernoulli job, $m-1$ deterministic



$\tau \sim \mathbb{E}\, Opt$

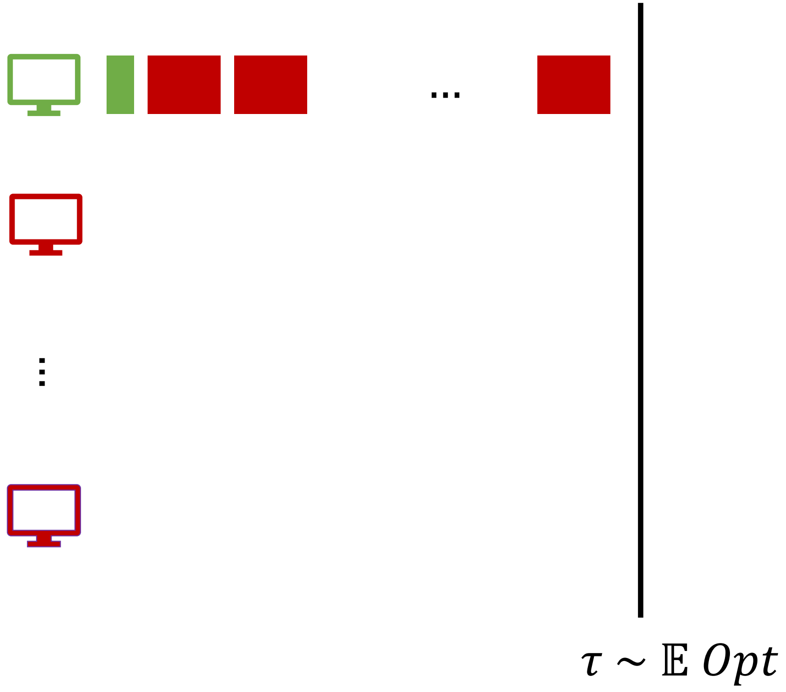
Benefit of Adaptivity

- One fast machine, $m-1$ slow
- One Bernoulli job, $m-1$ deterministic



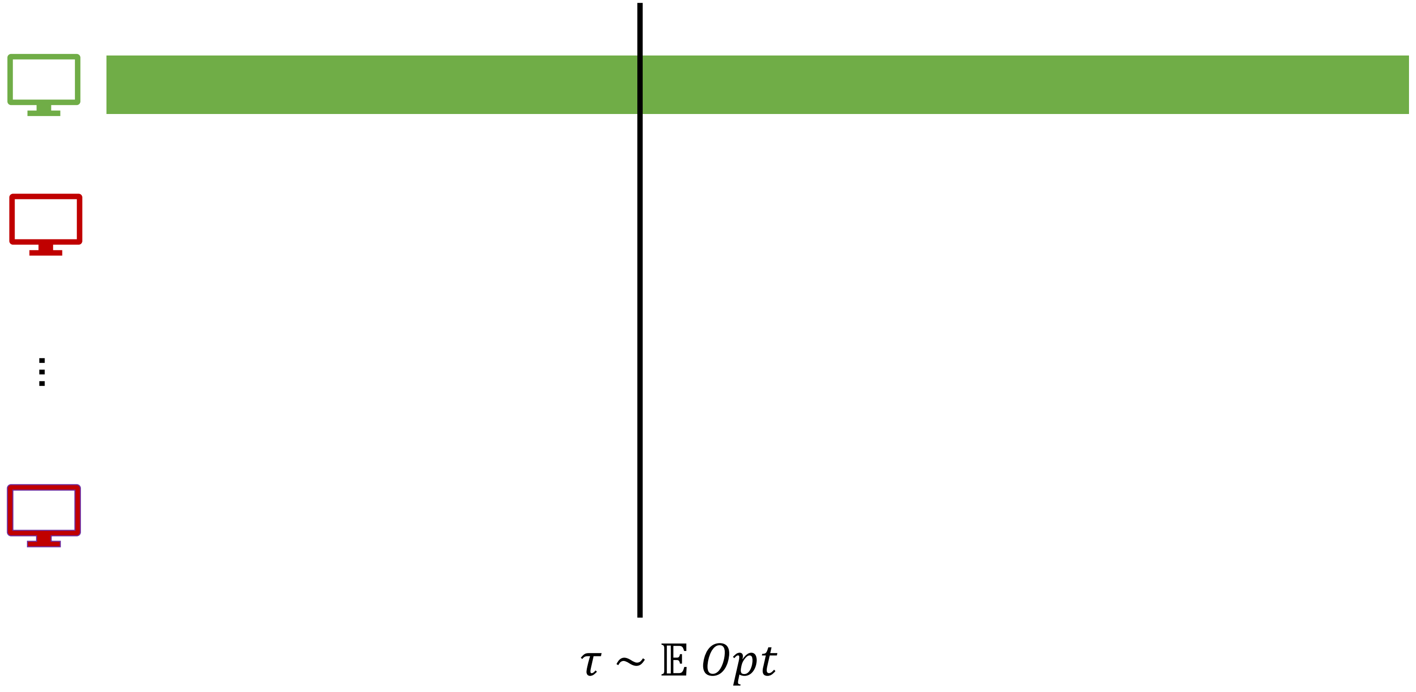
Benefit of Adaptivity

- One fast machine, $m - 1$ slow
- One Bernoulli job, $m - 1$ deterministic



Benefit of Adaptivity

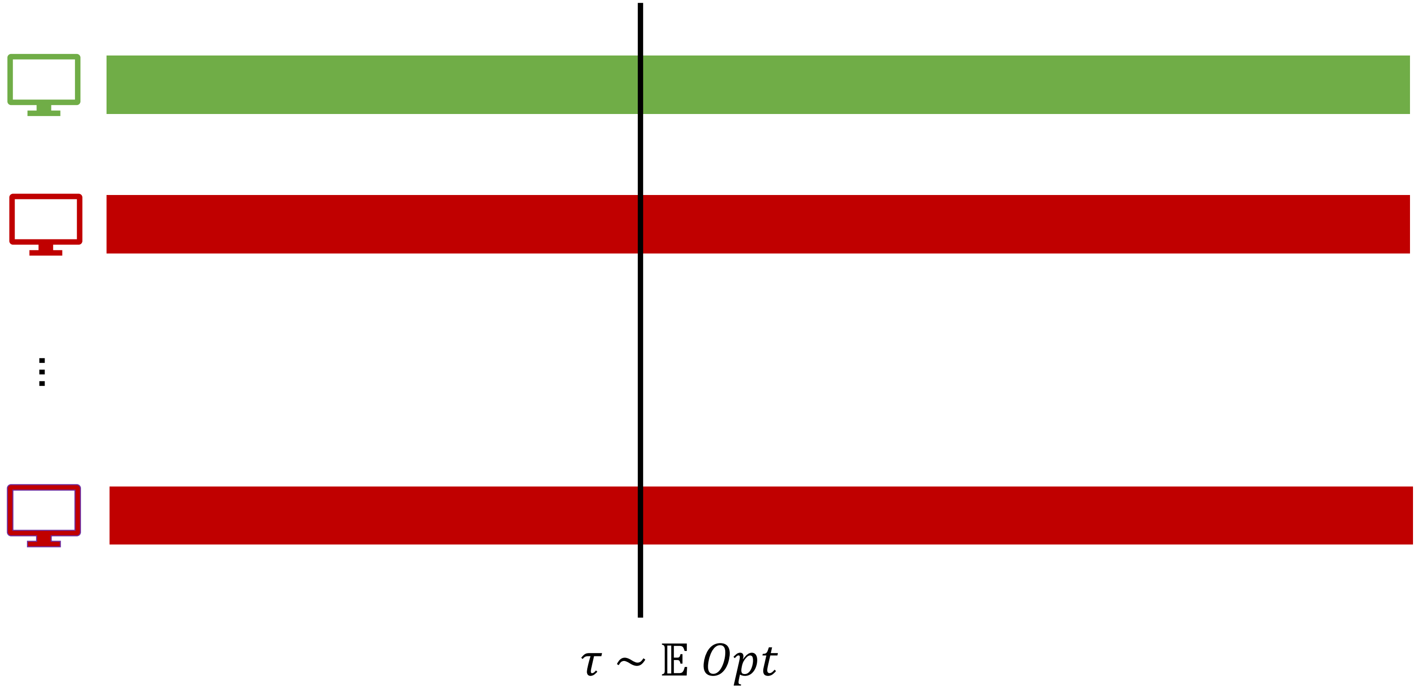
- One fast machine, $m-1$ slow
- One Bernoulli job, $m-1$ deterministic



Benefit of Adaptivity

- One fast machine, $m - 1$ slow
- One Bernoulli job, $m - 1$ deterministic

Problem: Optimal adaptive policy can have total expected exceptional load $\Omega(m) \cdot \mathbb{E}\, Opt$



Structure Theorem

- For truncation threshold $\tau \sim \mathbb{E}\ Opt$, there exists an adaptive policy $\widetilde{Opt}$ such that:
  - (near optimal) $\mathbb{E}\left[\widetilde{Opt}\right] \leq 2 \cdot \mathbb{E}\ [Opt]$
  - (small total expected exceptional load) The total expected exceptional load of $\widetilde{Opt}$ is at most $2 \cdot \mathbb{E}\ [Opt]$

Structure Theorem

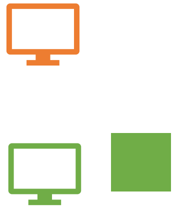
- For truncation threshold $\tau \sim \mathbb{E}\, Opt$, there exists an adaptive policy $\widetilde{Opt}$ such that:
  - (near optimal) $\mathbb{E}\left[\widetilde{Opt}\right] \leq 2 \cdot \mathbb{E}\,[Opt]$
  - (small total expected exceptional load) The total expected exceptional load of $\widetilde{Opt}$ is at most $2 \cdot \mathbb{E}\,[Opt]$
- $\Rightarrow$ natural assignment LP that ensures expected truncated load on each machine is $O(\mathbb{E}\, Opt)$ and total expected exceptional load is $O(\mathbb{E}\, Opt)$ is feasible
  - $\Rightarrow$ can round offline
  - $\Rightarrow$ can use potential function online

Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional $(X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)])$
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

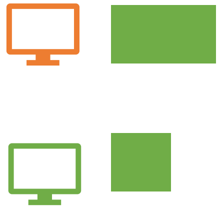


Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional $(X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)])$
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$



Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

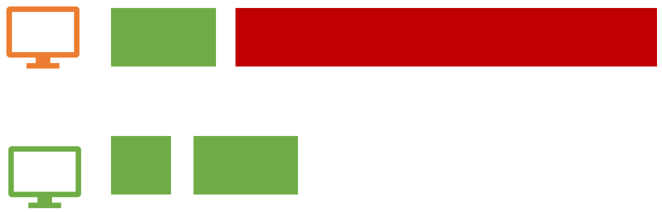
1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

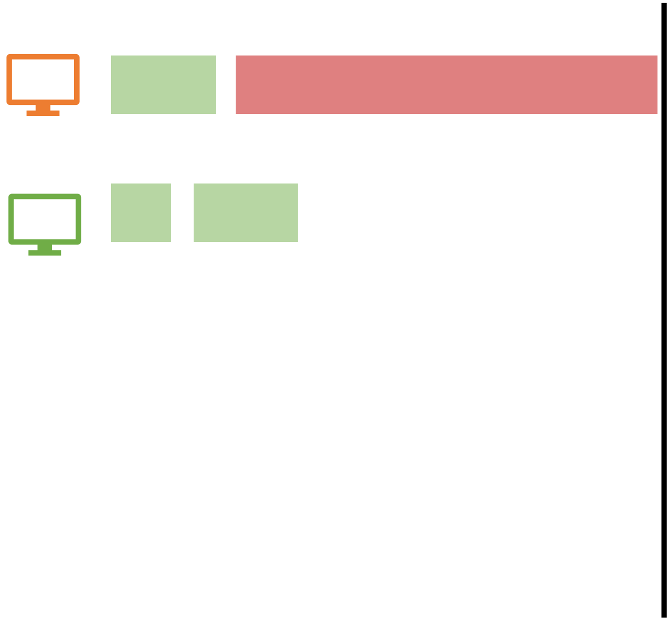


Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

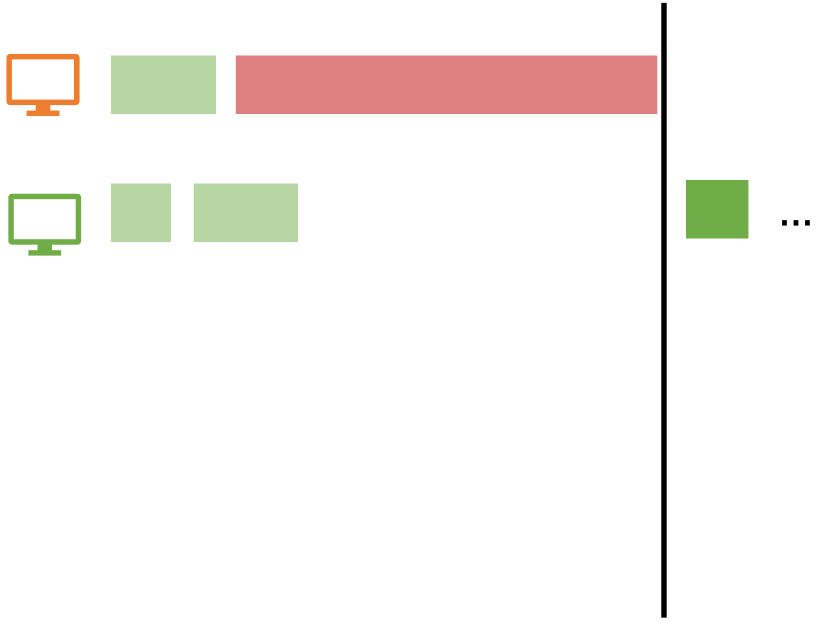


Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

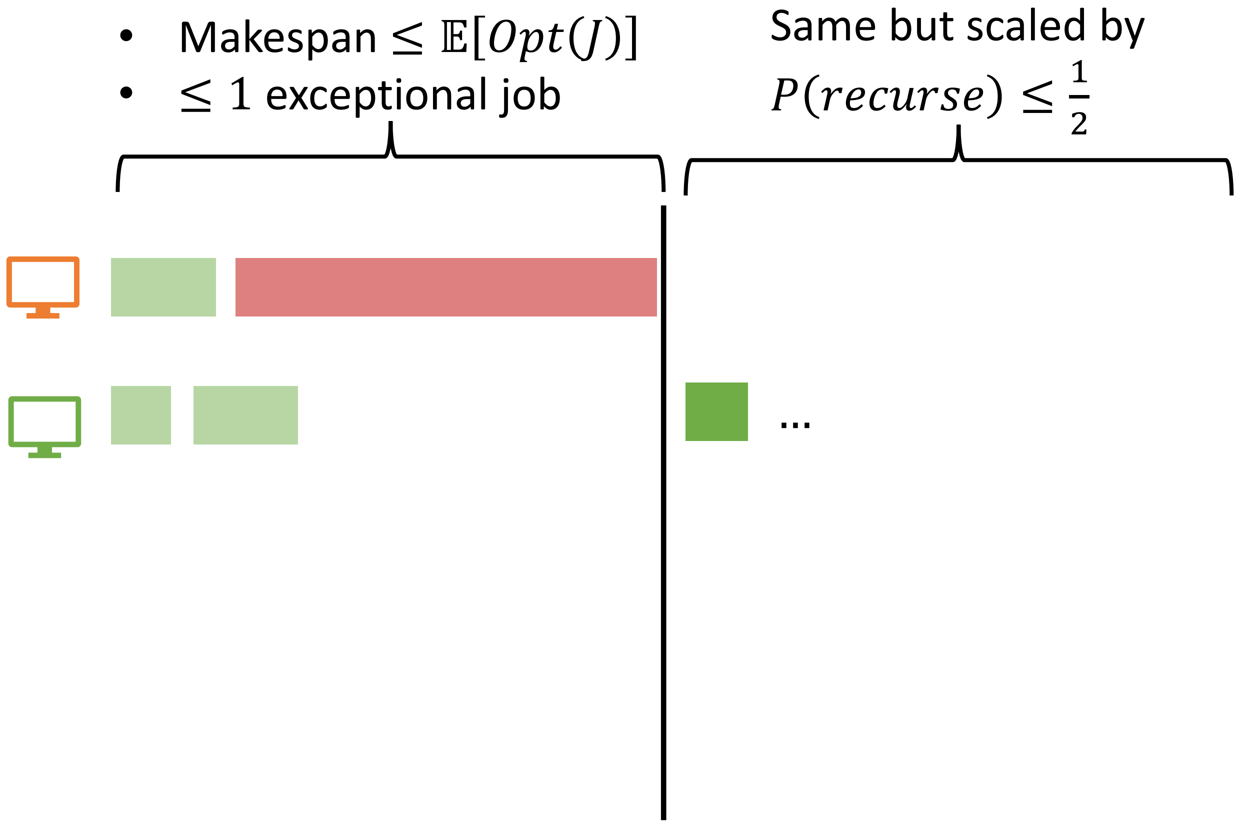


Proof

- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$



Proof

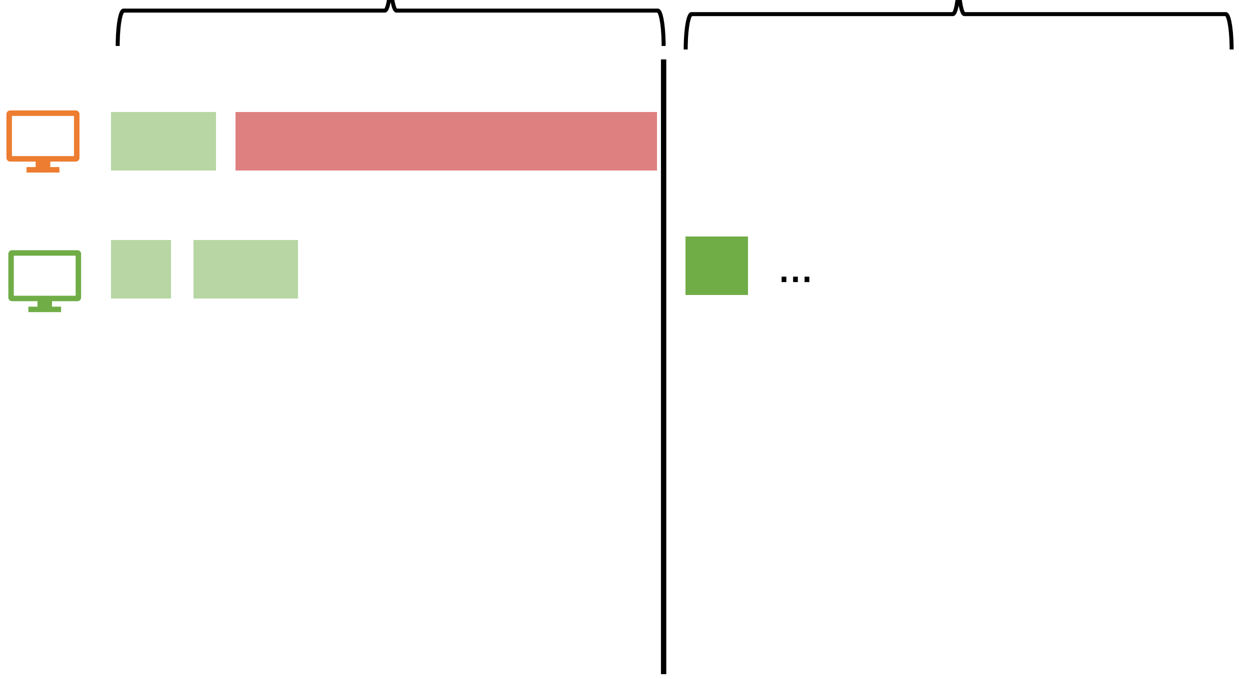
- Simulate Opt, but forget when we get unlucky

(Existential) Algorithm:

1. Given jobs J, follow optimal policy $Opt(J)$
2. If $Opt(J)$ assigns $j \to i$ such that X_{ij} becomes exceptional ($X_{ij} > \tau \geq 2 \cdot \mathbb{E}[Opt(J)]$)
3. Forget all previously-accrued machine loads, and recurse on remaining jobs $R \subset J$

- Makespan $\leq \mathbb{E}[Opt(J)]$
- ≤ 1 exceptional job

Same but scaled by $P(recurse) \leq \frac{1}{2}$



$$\mathbb{E}[Opt(J)] + \frac{1}{2}\mathbb{E}[Opt(J)] + \frac{1}{4}\mathbb{E}[Opt(J)] + \cdots$$

Structure Theorem

- For truncation threshold $\tau \sim \mathbb{E}\, Opt$, there exists an adaptive policy $\widetilde{Opt}$ such that:
  - (near optimal) $\mathbb{E}\left[\widetilde{Opt}\right] \leq 2 \cdot \mathbb{E}\,[Opt]$
  - (small total expected exceptional load) The total expected exceptional load of $\widetilde{Opt}$ is at most $2 \cdot \mathbb{E}\,[Opt]$
- $\Rightarrow$ natural assignment LP that ensures expected truncated load on each machine is $O(\mathbb{E}\, Opt)$ and total expected exceptional load is $O(\mathbb{E}\, Opt)$ is feasible
  - $\Rightarrow$ can round offline
  - $\Rightarrow$ can use potential function online

Idea: Forget when we get unlucky

Conclusion

Theorem: There exists an efficient algorithm for stochastic load balancing on unrelated machines that $O(\frac{\log m}{\log \log m})$-approximates the optimal adaptive policy. Further, the algorithm is non-adaptive.

- Many extensions: online, adaptive policies for related machines, stochastic routing, etc.
- Structure theorem for near-optimal adaptive policies

Conclusion

Theorem: There exists an efficient algorithm for stochastic load balancing on unrelated machines that $O(\frac{\log m}{\log \log m})$-approximates the optimal adaptive policy. Further, the algorithm is non-adaptive.

- Many extensions: online, adaptive policies for related machines, stochastic routing, etc.
- Structure theorem for near-optimal adaptive policies

Questions:
- Improve using adaptivity?
- Hardness of approximation?