Learning to match in online platforms

Milan Vojnovic

Department of Statistics

LSE

Seminar of the School of Information Sciences & Technology,
Athens University of Economics and Business, Athens, Greece, 12 December 2019

Outline

- Part I – Adaptive matching for expert systems with uncertain task types
- Part II – Test score approach to team selection

Part I

Adaptive matching for expert systems with uncertain task types

Joint work with L. Gulikers, L. Massoulie, and V. Shah

Operations Research, accepted 2019

Motivating application scenarios

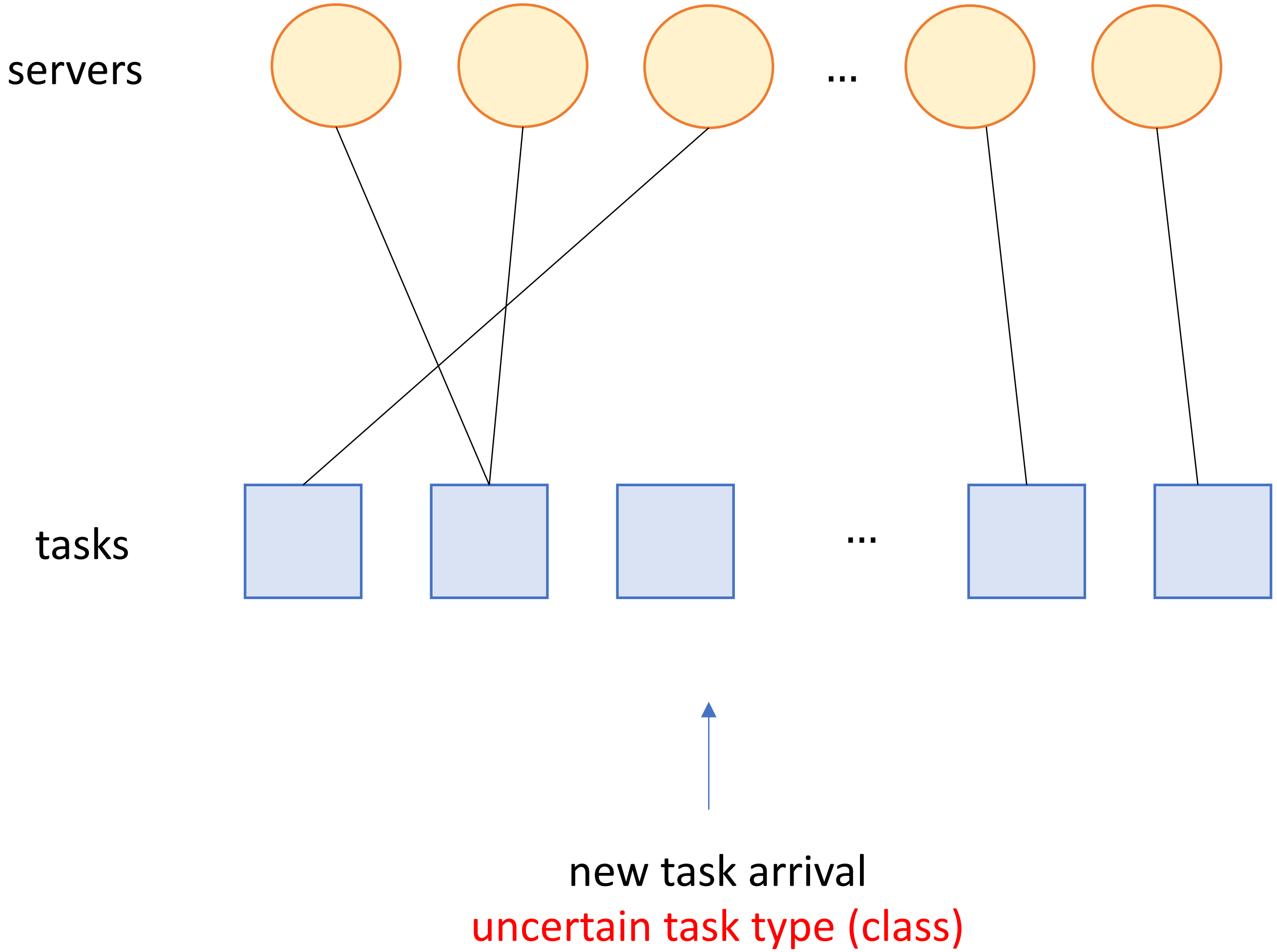
employers – employees

cars – passengers

travelers – housing facilities

questions – answers

Matching problem formulation

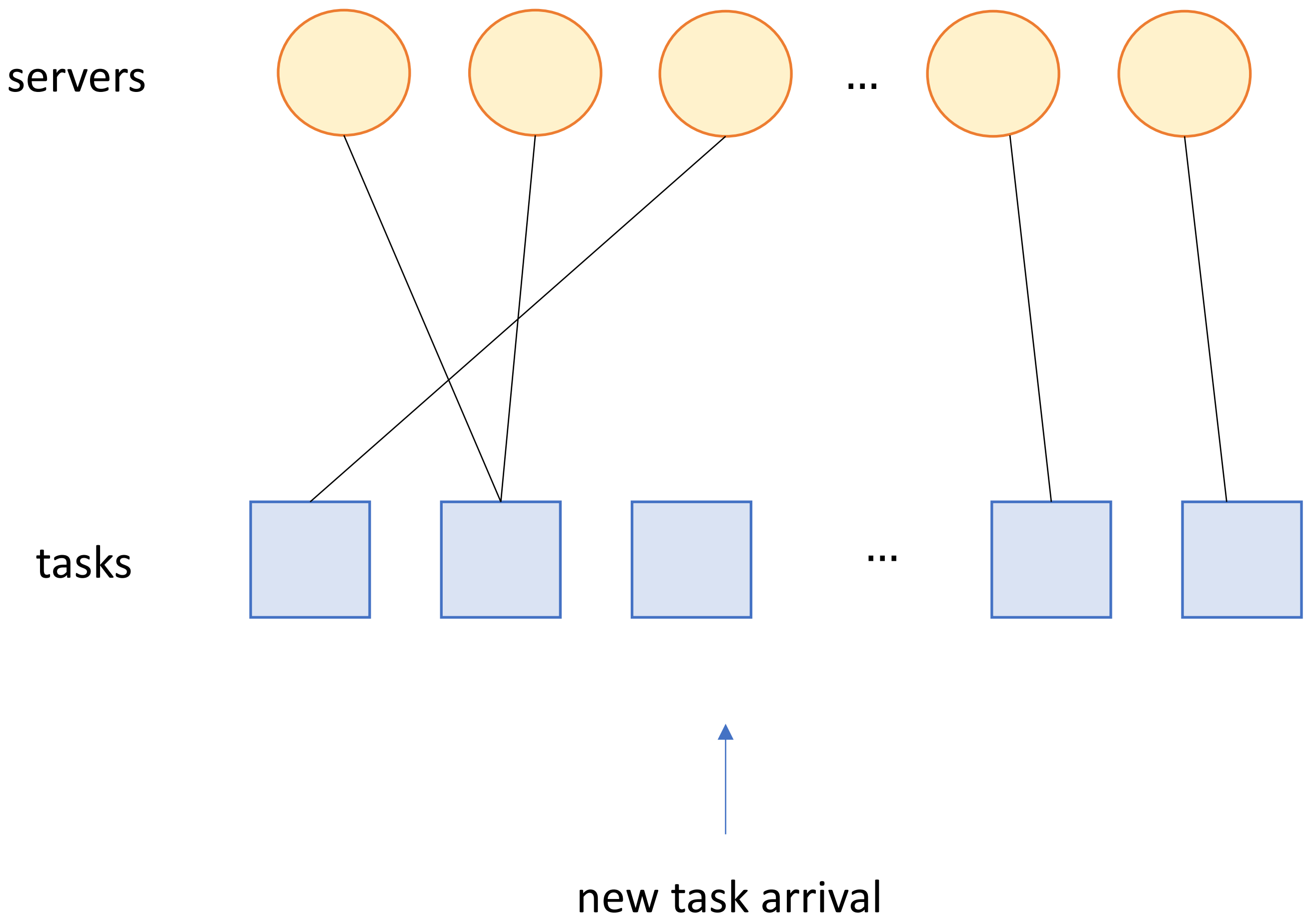


Key questions

- What throughput can be achieved by service systems with uncertain task types by learning while matching tasks to servers?

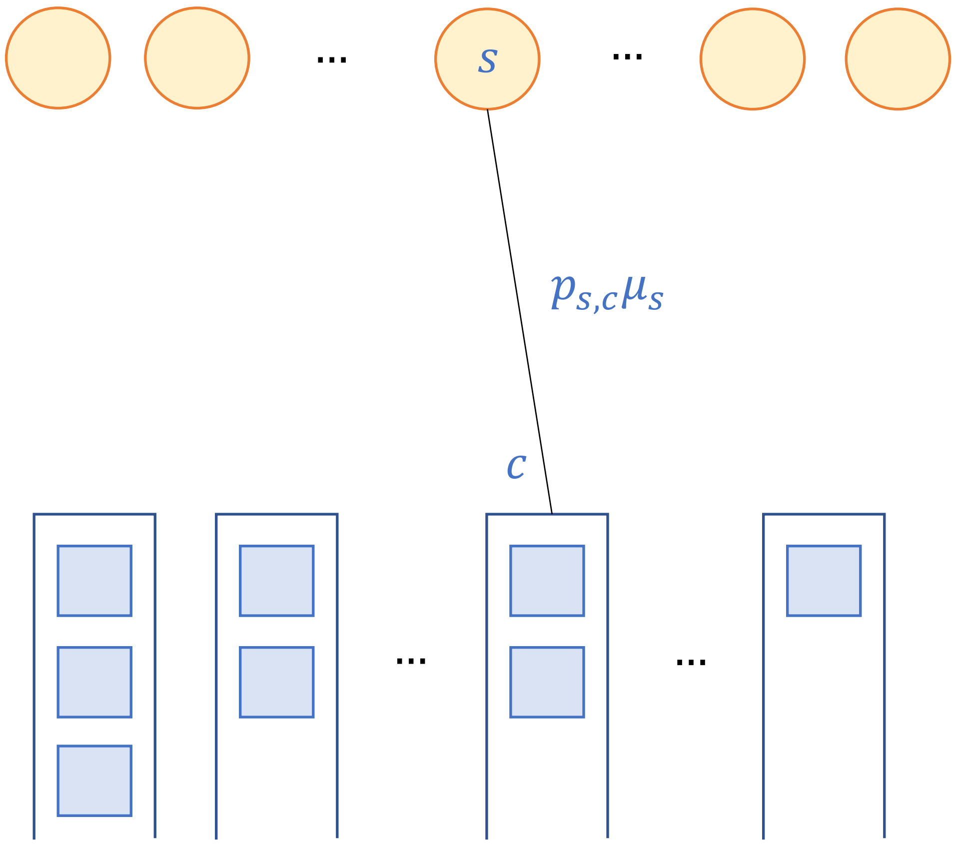
- What policies can achieve optimal throughput?

Problem formulation



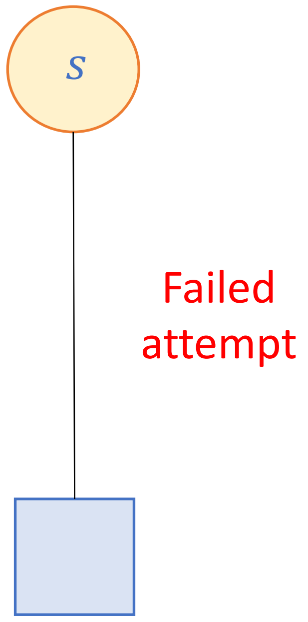
- Each task is of a hidden (latent) class, from a finite set C of classes
- Each server can serve at most 1 task at any time with processing rate μ_s
- Server s solves a task of class c according to an independent Bernoulli $(p_{s,c})$ random variable
- Bayesian framework: prior distribution for class type π

Classical case: scheduling flexible servers



- No uncertainty:
  - Known task classes
  - Known processing rates
- Goal:
  - Minimize a long-term cost, defined as a function of queue sizes or job waiting tasks
- Optimality of simple policies in some regimes:
  - $c\mu$-scheduling policy

Learning from failures



Probability of failure:

$$\psi_s(z) = \sum_{c \in C} (1 - p_{s,c}) z_c$$

Prior distribution of task type:

Posterior distribution of task type:

$$z \quad \mapsto \quad z' = \phi_s(z) = \left(\frac{(1 - p_{s,c}) z_c}{\psi_s(z)}, c \in C \right)$$

Optimal stability region

- **Thm** Assume there exits server s such that $p_{s,c} > 0$ for all $c \in C$.

  If there are variables $v_{s,c} \geq 0$ and $\delta_s > 0$ for $s \in S$ and $c \in C$ such that

  $$\lambda\pi_{z'} + \sum_{s\in S, z\in Z:\phi_s(z)=z'} v_{s,z}\psi_s(z) = \sum_{s\in S} v_{s,z'} \text{ for all } z' \in Z \quad \text{(flow conservation)}$$

  and

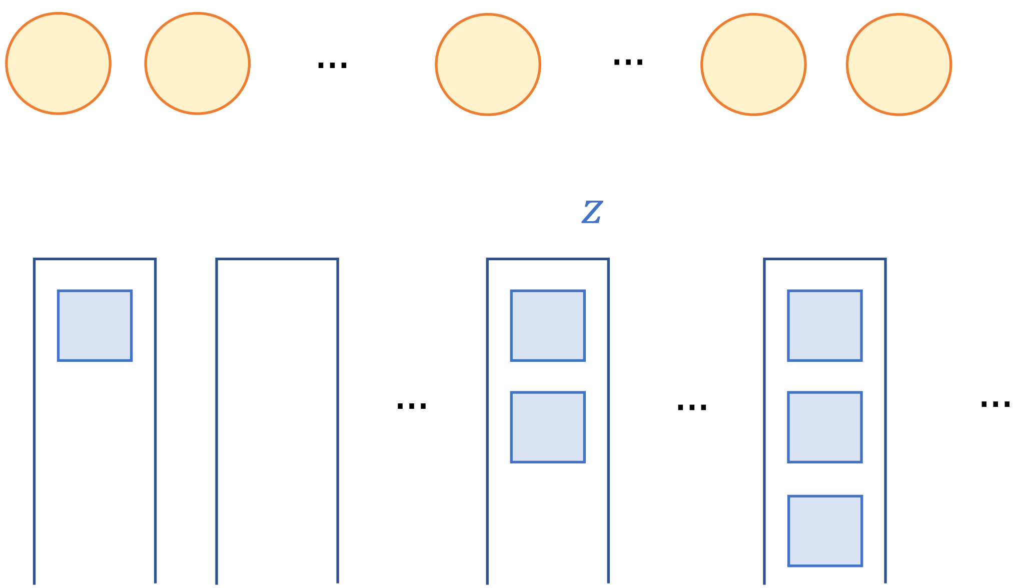
  $$\sum_{z'\in Z} v_{s,z'} + \delta_s \leq \mu_s \text{ for all } s \in S \quad \text{(capacity constraint)}$$

  then, there exists a policy under which the system is stable.

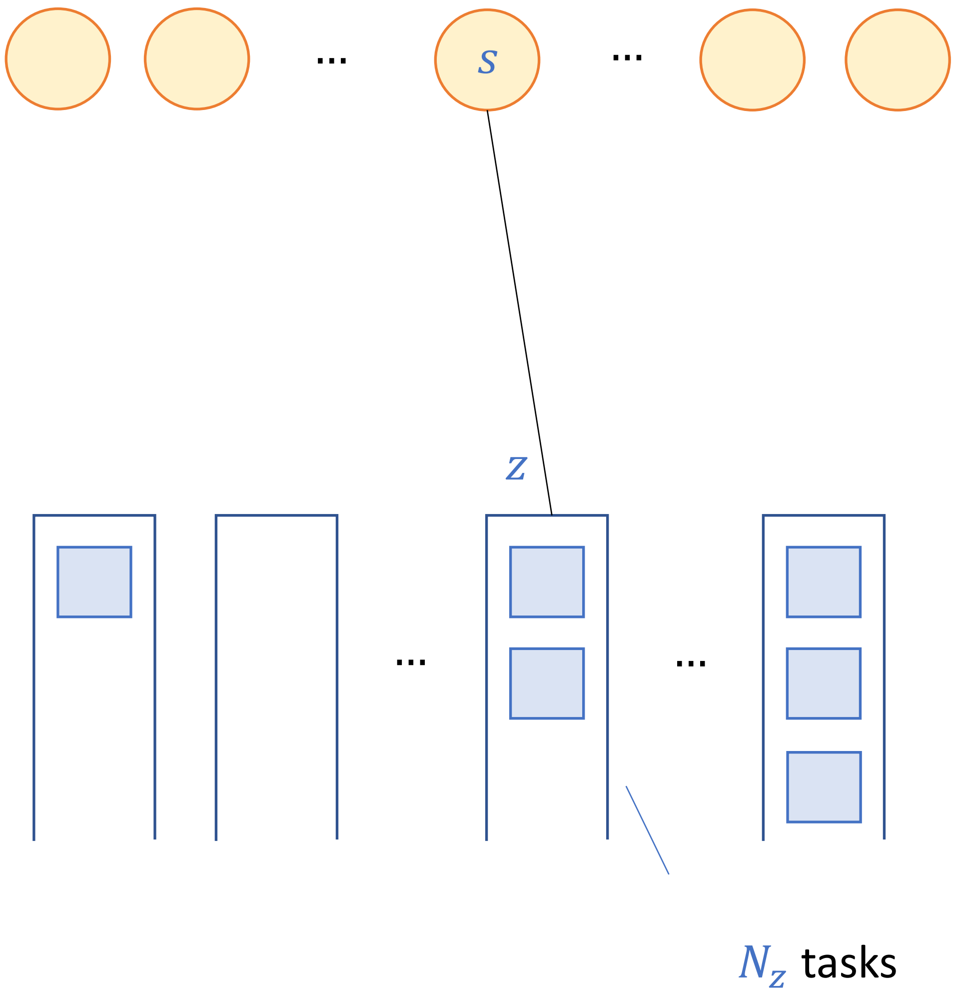
  Otherwise, there is no policy under which the system is stable.

Throughput optimal policy: challenges

- Natural approach: associate a queue with each task type z
- Challenge: an infinite number of queues (unlike to classical queueing systems)



Naïve greedy policy



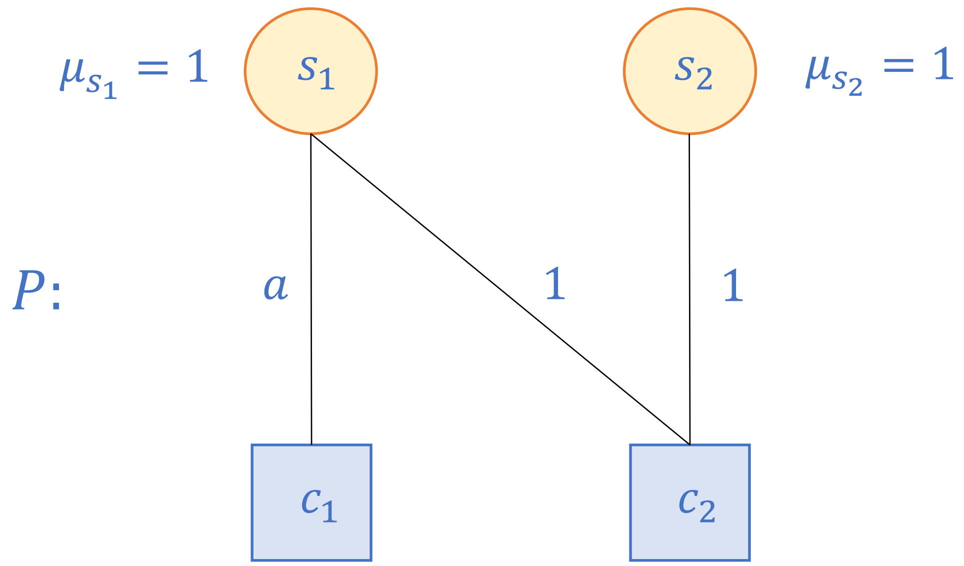
- At each time when there is a free server s and a task waiting to be served, assign s to a task with maximum success probability according to the posterior distribution of task class:

$$z(s) \in \operatorname{argmax}_{z \in Z : N_z > 0} \left(1 - \psi_s(z)\right)$$

with random tie break

Not throughput optimal

Special case: Asymmetric (a) system



$$\psi_{s_1}(z) = \frac{1}{2}(1-a) \qquad \psi_{s_1}(z') = (1-a)$$

$$\psi_{s_2}(z) = \frac{1}{2} \qquad \psi_{s_2}(z') = 1$$

- Arrival type:

$$(z_{c_1}, z_{c_2}) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

- Upon a failed attempt for a task of type z, the task becomes of type z' where

$$(z'_{c_1}, z'_{c_2}) = (1,0)$$

- Set of task types $Z = \{z, z'\}$

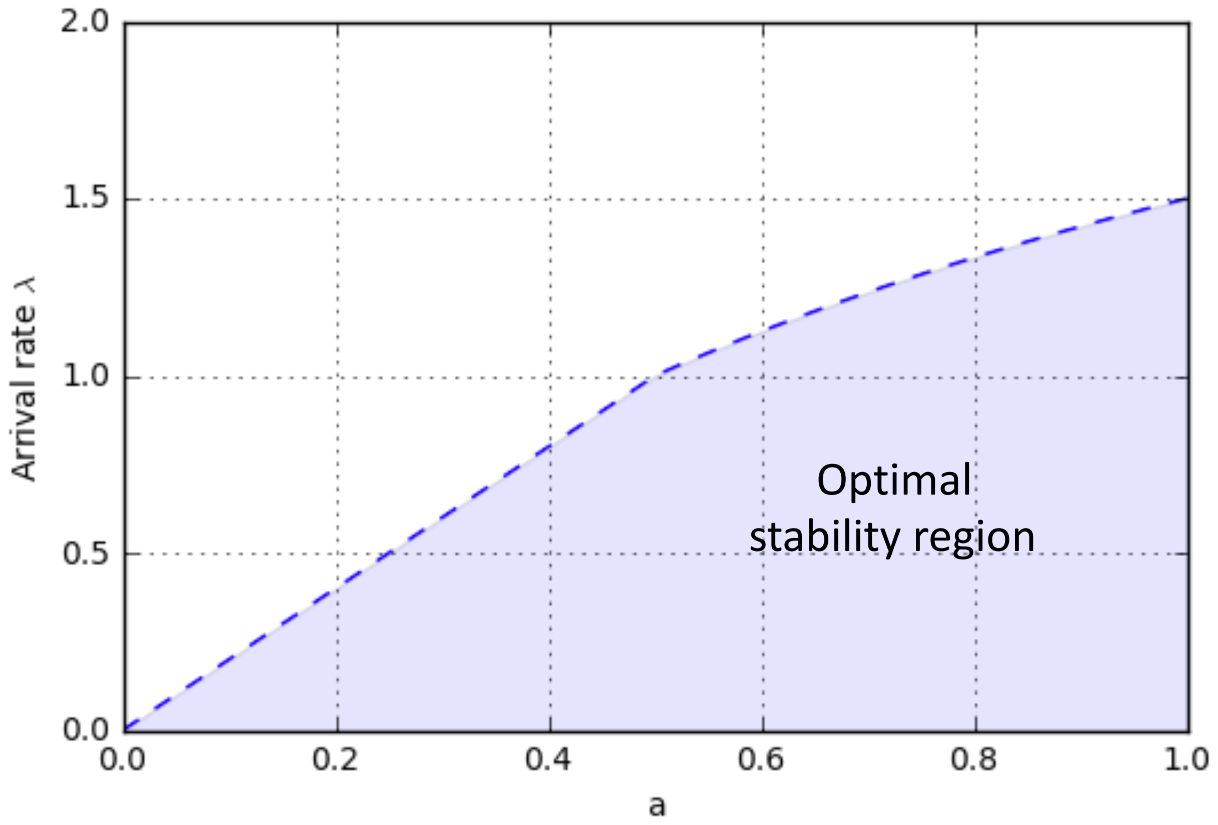
Asymmetric (a) system: optimal stability region

- Optimal stability region:

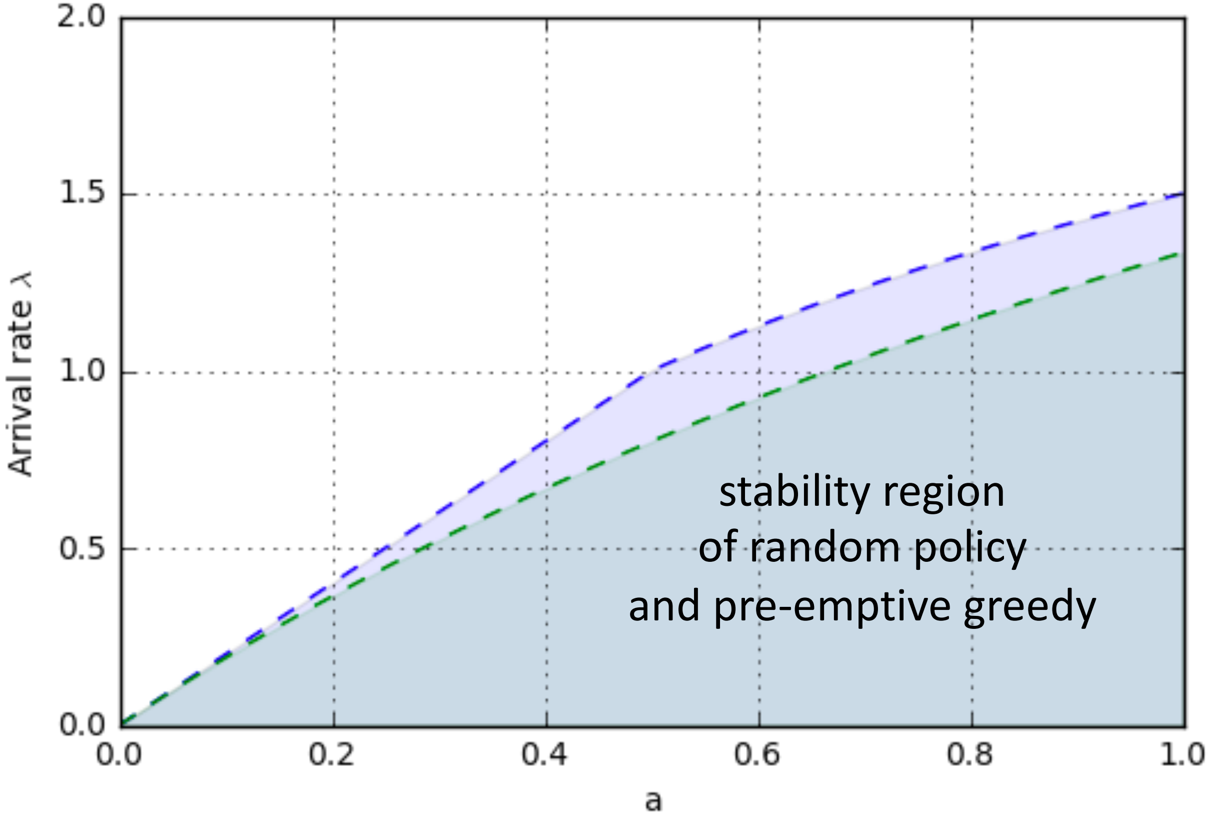
$$\lambda < \lambda^{\star}(\mathrm{a})$$

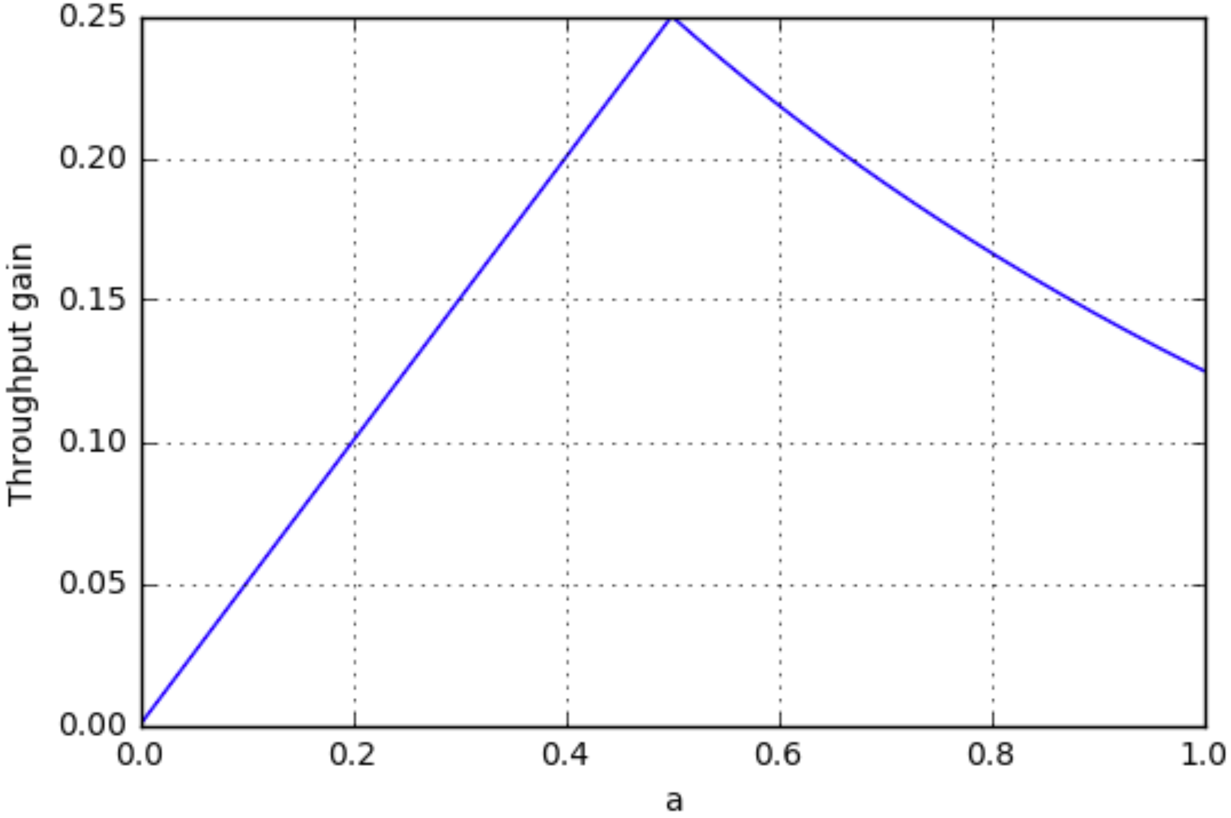
where

$$\lambda^{\star}(a) = \min\left\{2a, \frac{3a}{a+1}\right\}$$



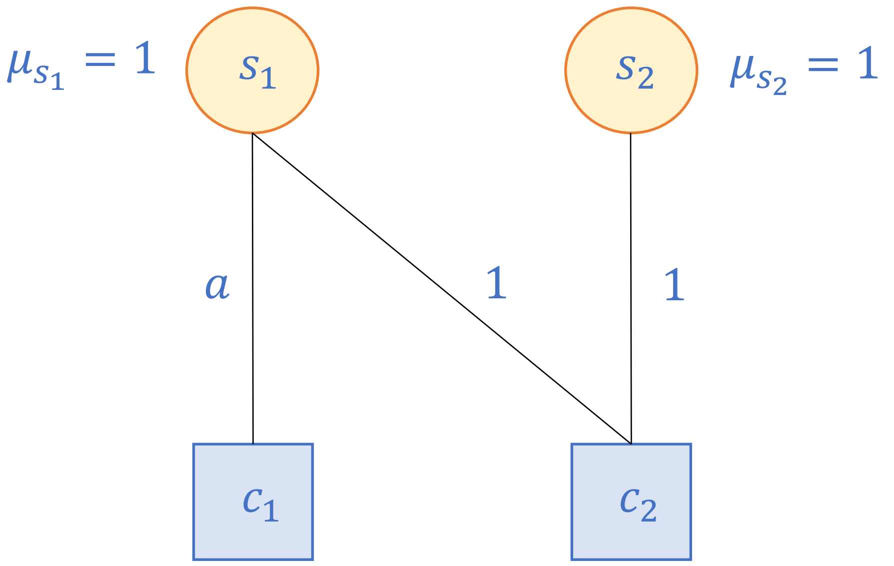
Stability region of random and greedy policies





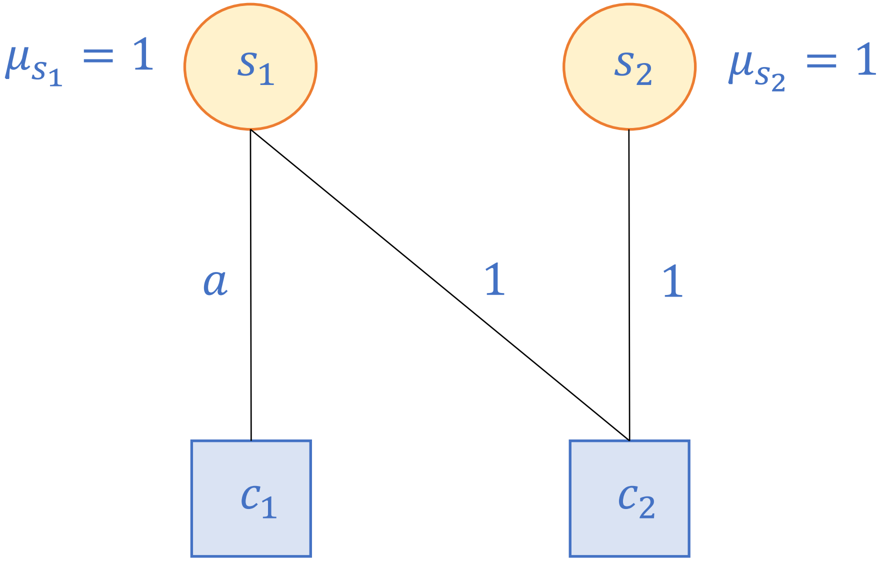
$$\lambda < \lambda^{\star}(a) = \frac{4a}{2+a}$$

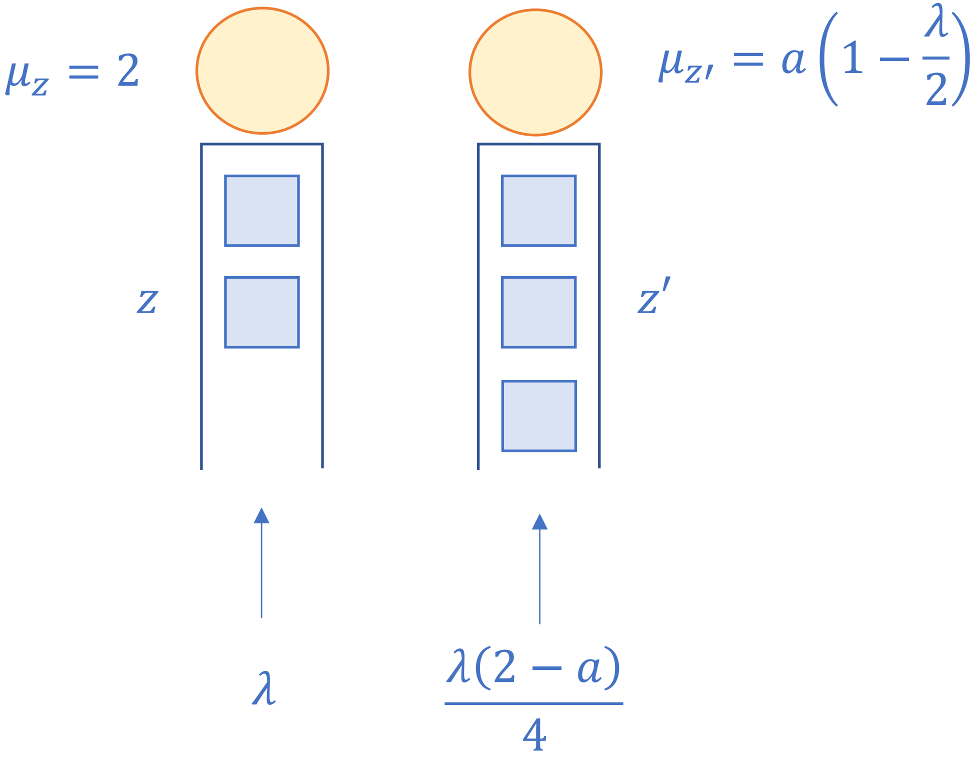
Optimal stability region: intuition



- For small values of a, the main bottleneck is s_1 serving tasks of class a
- The extra capacity of server s_2 can be used to identify class c_1 tasks
- For large values of a, both servers are bottleneck, and thus identifying class c_2 tasks results in a throughput loss

Intuition (cont'd)



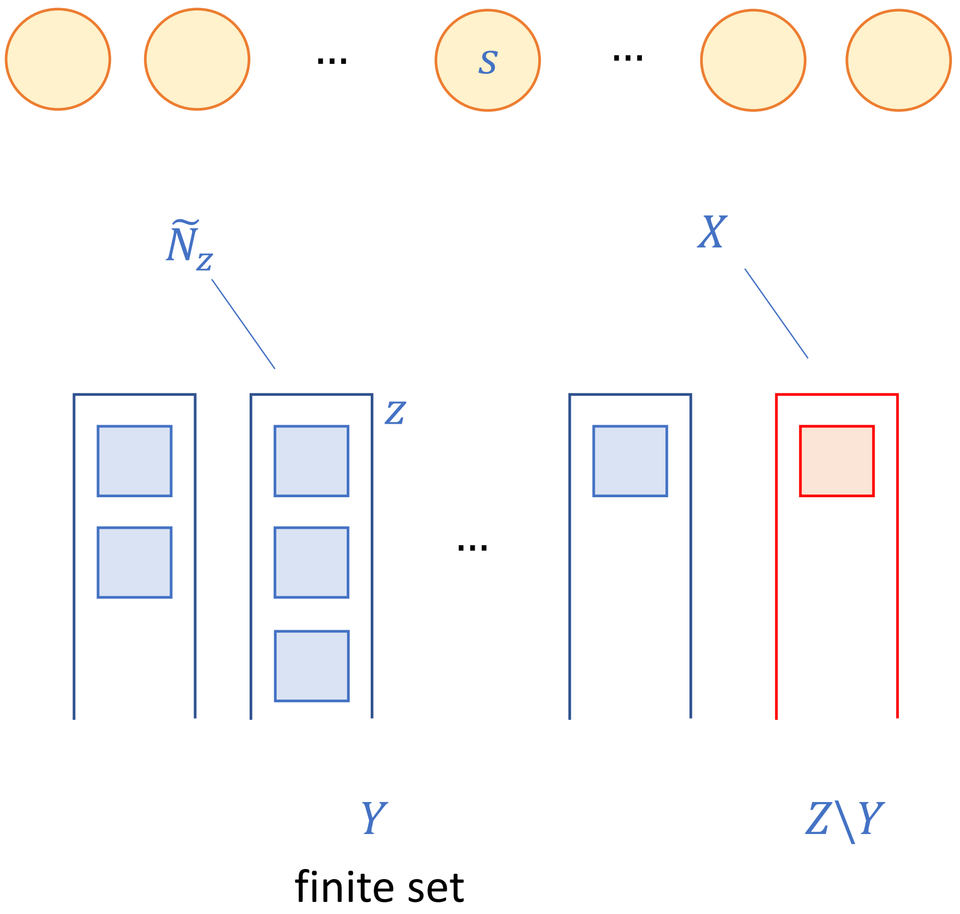


$$\psi_{s_1}(z) = \frac{1}{2}(1-a) \qquad \psi_{s_1}(z') = (1-a)$$

$$\psi_{s_2}(z) = \frac{1}{2} \qquad \psi_{s_2}(z') = 1$$

Backpressure (Y) policy

- Key idea: bundling task types such that the total number of queues is finite



Backpressure (Y) priority index:

$$w_{s,z}(\tilde{N},X) = \begin{cases} \tilde{N}_z - \psi_s(z)\tilde{N}_{\phi_s(z)} & \text{if } \phi_s(z) \in Y \\ \tilde{N}_z - \psi_s(z)X & \text{if } \phi_s(z) \in Z\backslash Y \end{cases}$$

Backpressure (Y) policy

- **Algorithm**: when assigning sever s, if

$$X \leq \frac{\sum_{s' \in S} \mu_{s'} \max_{z \in Y:\, \tilde{N}_z > 0} w_{s',z}(\tilde{N}, X)}{\min_{c \in C} \sum_{s' \in S} p_{s',c} \mu_{s'}}$$

then, assign a task of type in $B_s(\tilde{N}, X)$ to s with random tie break where

$$B_s(\tilde{N}, X) = \arg\max_{z \in Y:\, \tilde{N}_z > 0} w_{s,z}(\tilde{N}, X)$$

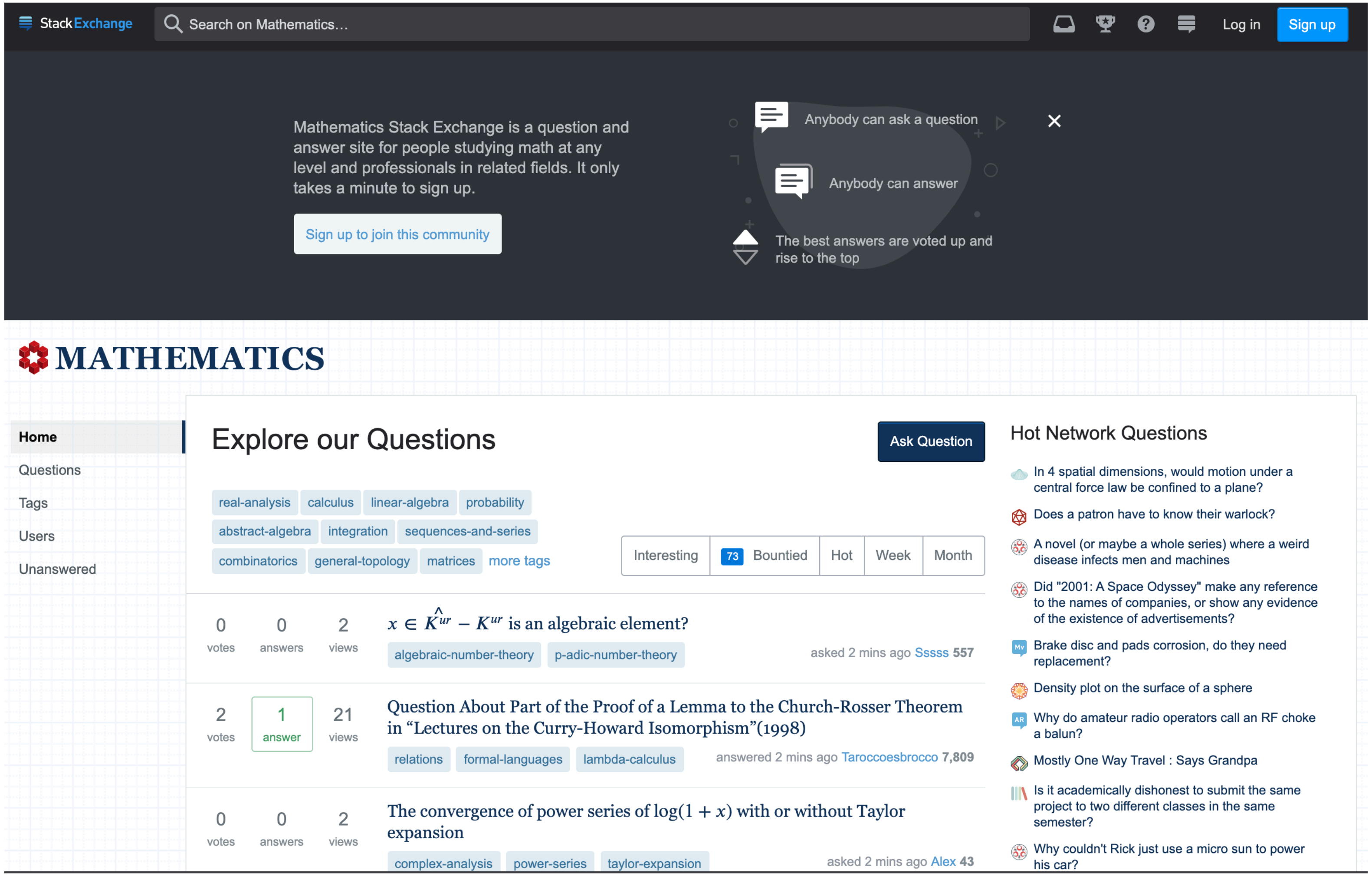
else, assign a task chosen uniformly at random from $Z \setminus Y$

Throughput optimality of Backpressure (Y)

- **Thm**: Assume there exits server s such that $p_{s,c} > 0$ for all $c \in C$.

  If the sufficient conditions for stability hold, then there exists a finite subset Y of the set of task classes Z such that Backpressure (Y) policy is throughput optimal.

Experimental results: Math StackExchange



StackExchange Search on Mathematics…

MATHEMATICS

Home
Questions
Tags
Users
Unanswered

Proving $\lim_{n\to\infty} \dfrac{\Phi^{n+1} - (1-\Phi)^{n+1}}{\Phi^n - (1-\Phi)^n} = \Phi$

Asked today Active today Viewed 39 times

2

$\Phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio

I'm having hard time using proving that

$$\lim_{n\to\infty} \frac{\Phi^{n+1} - (1-\Phi)^{n+1}}{\Phi^n - (1-\Phi)^n} = \Phi$$

dividing both the numerator and denominator by Φ^n doesn't help, neither does $\Phi^n - (1-\Phi^n) = (2\Phi+1)\sum_{i=0}^{n-1} \Phi^i(1-\Phi)^{n-1-i}$

Where is the trick?

calculus sequences-and-series golden-ratio

share cite improve this question

asked 1 hour ago
Fritjof Larsson
107 ▲5
New contributor

4 I think that dividing the numerator and denominator by Φ^n is helpful. – Lord Shark the Unknown 1 hour ago

add a comment

2 Answers

active oldest votes

5

Hint:

$$\Phi - 1 = \frac{\sqrt{5}-1}{2} = \frac{5-1}{2(\sqrt{5}+1)} = \frac{2}{\sqrt{5}+1} < 1 \text{ and } > 0$$

$$\implies |1-\Phi| < 1 \text{ and } \left|\frac{1-\Phi}{\Phi}\right| < 1$$

Divide the numerator and the denominator by Φ^n

share cite improve this answer edited 1 hour ago

answered 1 hour ago
lab bhattacharjee
243k ●15 ■170 ▲292

1 @user1992, Thanks for the observation – lab bhattacharjee 1 hour ago

1 @user1992, Rectified – lab bhattacharjee 1 hour ago

add a comment

2

Use How do I prove Binet's Formula?

if $F(m) = \dfrac{\alpha^m - \beta^m}{\alpha - \beta}$ with α, β are the roots of

$$t^2 - t - 1 = 0$$

we can prove

$$F_{n+2} = F_{n+1} + F_n$$

$$\frac{F_{n+2}}{F_{n+1}} = 1 + \frac{1}{\frac{F_{n+1}}{F_n}}$$

If $\lim_{n\to\infty} \dfrac{F_{n+2}}{F_{n+1}} = r > 0,$

$$r = 1 + \frac{1}{r} \iff r^2 - r - 1 = 0, r = ?$$

share cite improve this answer

answered 1 hour ago
lab bhattacharjee
243k ●15 ■170 ▲292

add a comment

Dataset

702,286 questions
994,138 answers

For each (user, tag) pair, the success probability estimated by empirical frequency

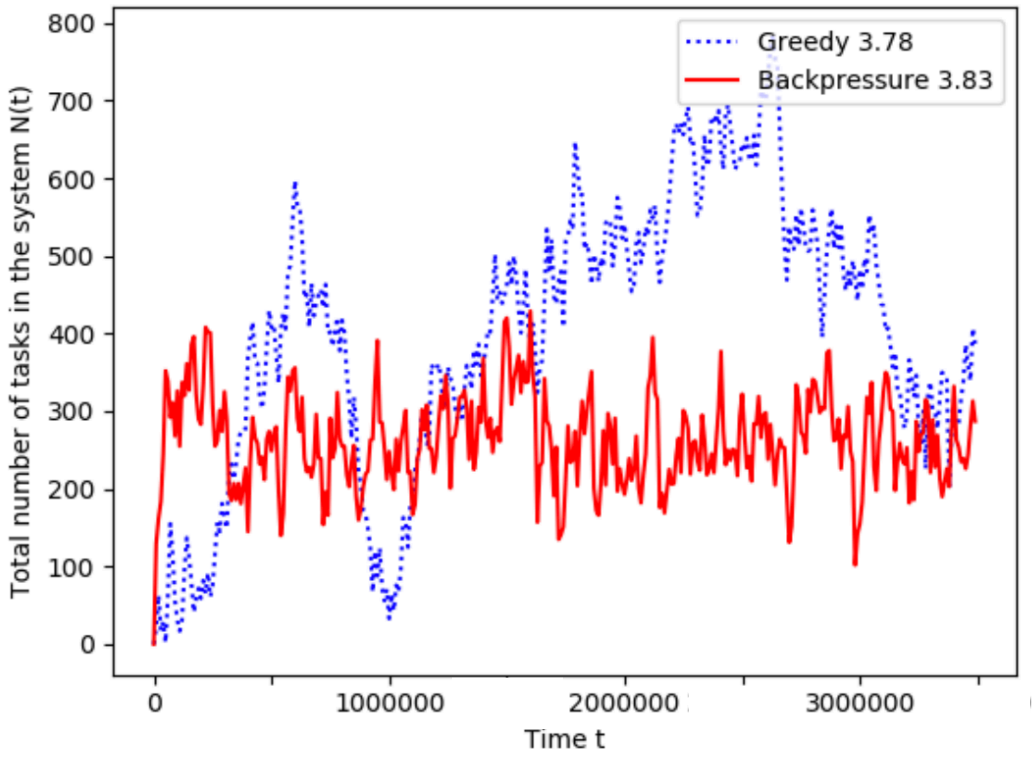
Expert classes computed by using k-means clustering

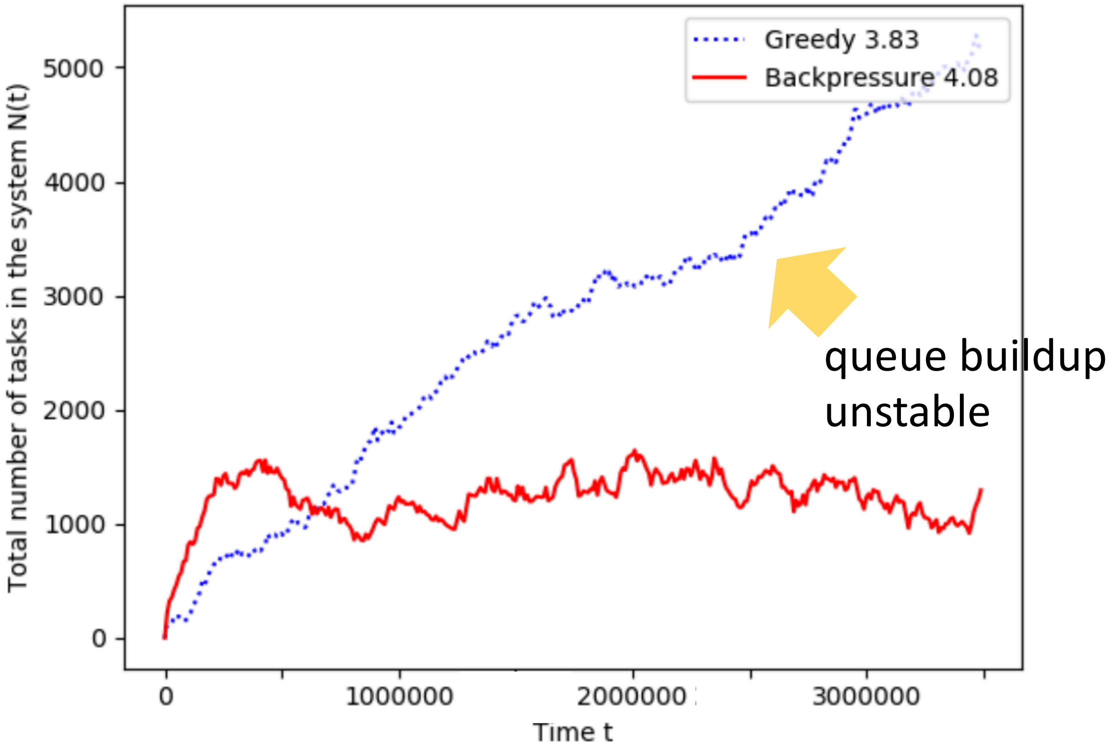
Inferred expert skills:

	Expert Clusters									
Tags	1	2	3	4	5	6	7	8	9	10
calculus	.32	**.39**	.30	**.35**	**.37**	**.47**	.28	.16	.26	**.41**
real-analysis	.17	**.41**	.25	.32	.23	**.49**	**.40**	.10	.10	**.44**
linear-algebra	**.46**	.29	.05	**.36**	.14	**.48**	.26	.31	.07	**.43**
probability	.07	**.49**	.02	.33	.02	**.50**	.06	.02	**.46**	.04
abstract-algebra	.02	.05	.03	.32	.02	**.38**	.23	**.50**	.01	.27
integration	.09	**.43**	.05	.19	**.44**	**.45**	.03	.01	.06	**.37**
sequences-and-series	.05	.32	.16	.31	.20	**.45**	.09	.04	.06	.33
general-topology	.02	.10	.03	.16	.02	**.43**	**.50**	.07	.02	.31
combinatorics	.03	.14	.06	**.43**	.04	**.37**	.02	.06	.19	.05
matrices	.27	.15	.02	.31	.02	**.44**	.06	.11	.02	.34
complex-analysis	.02	.19	.08	.16	.14	**.50**	.09	.05	.01	**.44**
Size	165	188	313	200	179	183	231	187	178	176

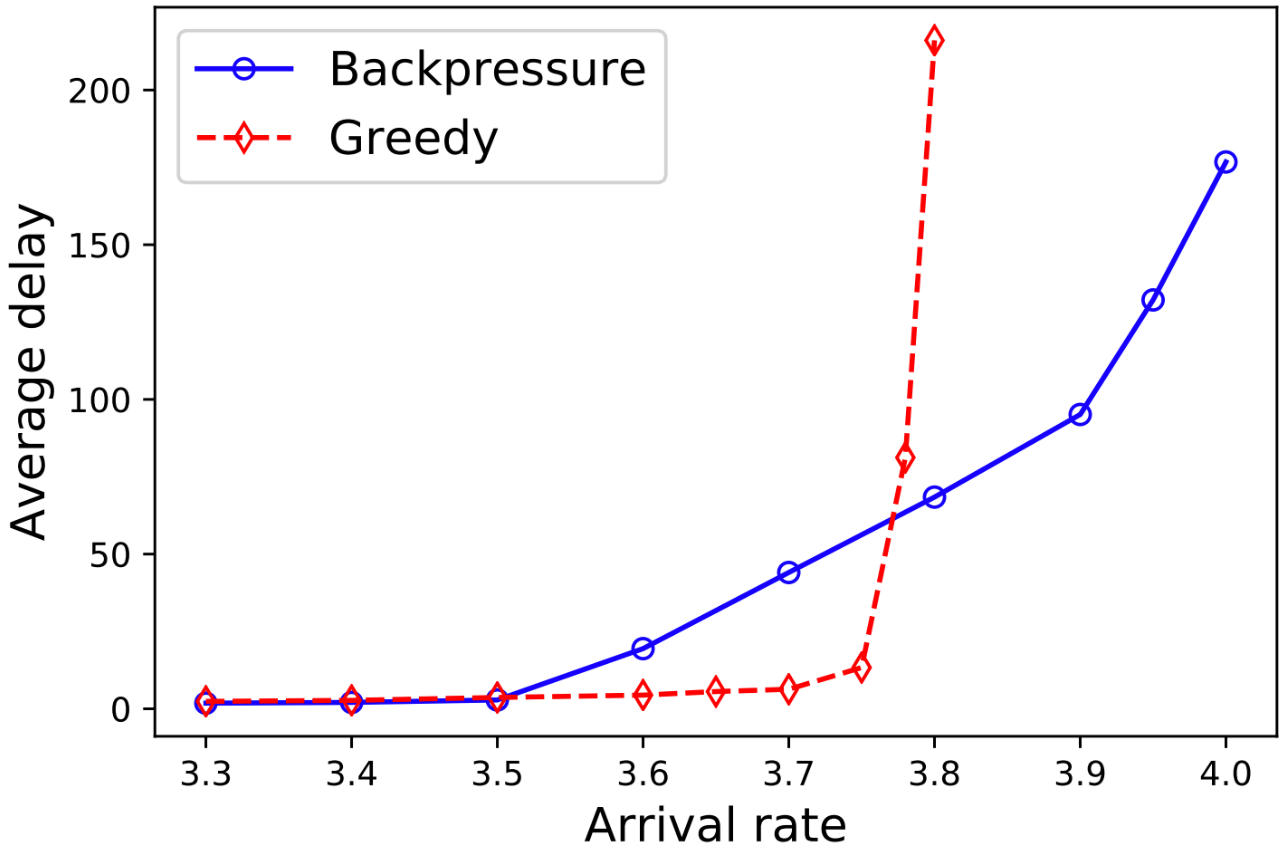
Estimated parameters used in simulations for different question arrival rates λ

Queue backlog: Backpressure vs greedy





Average delay: Backpressure vs greedy



Part I – summary points

- Backpressure type policy for assigning tasks to servers with uncertain task types
- Shown to be throughput optimal
- Greedy and random policy can be substantially suboptimal
- Backpressure policy not easy to implement, but provides guidelines for designing simple-to-implement heuristic policies

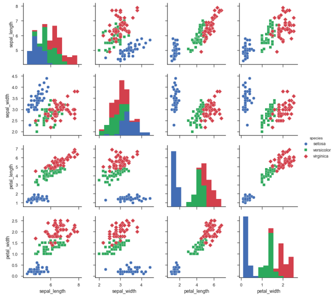
Part II

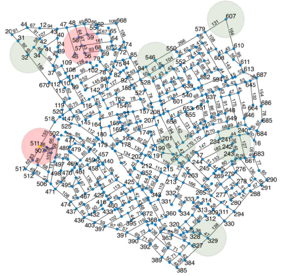
Test score approach to team selection

Joint work with S. Sekar and S. Yun

Management Science, accepted 2019

Motivating application scenarios

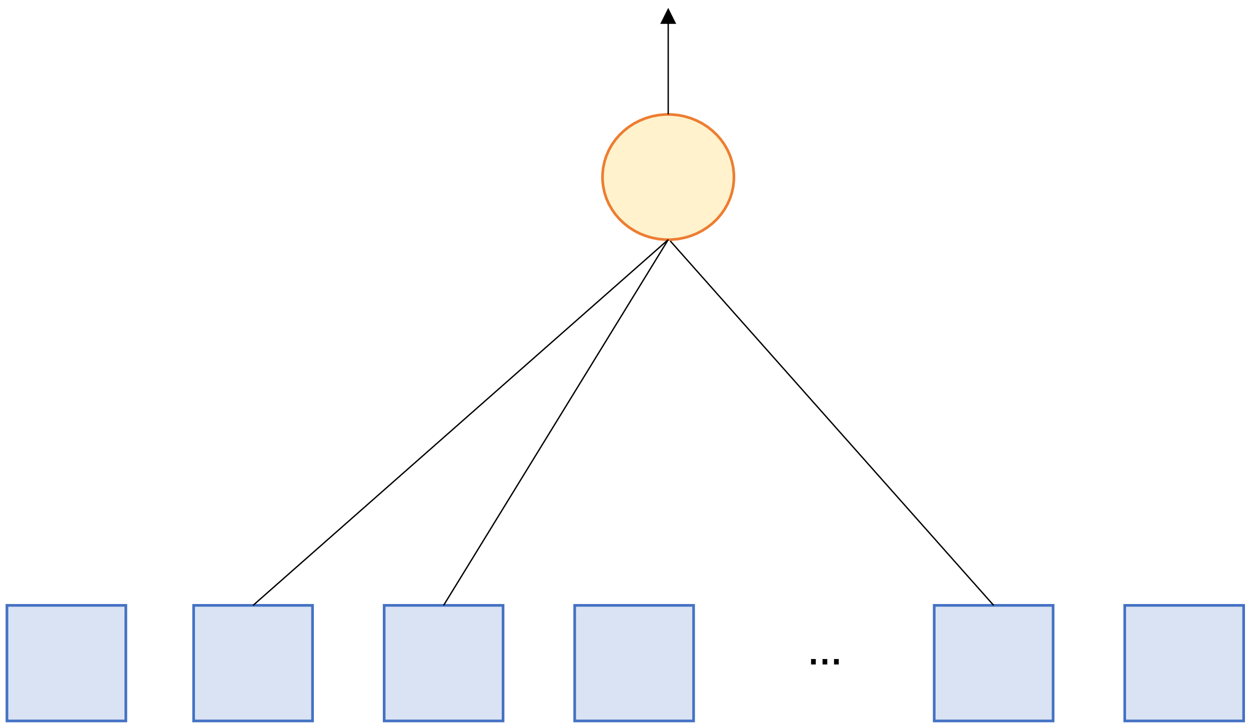




- Data summarization
- Recommender systems
- Feature selection for learning models
- Online platforms
- Combinatorial auctions
- Sensor placement
- Influence maximization in social networks

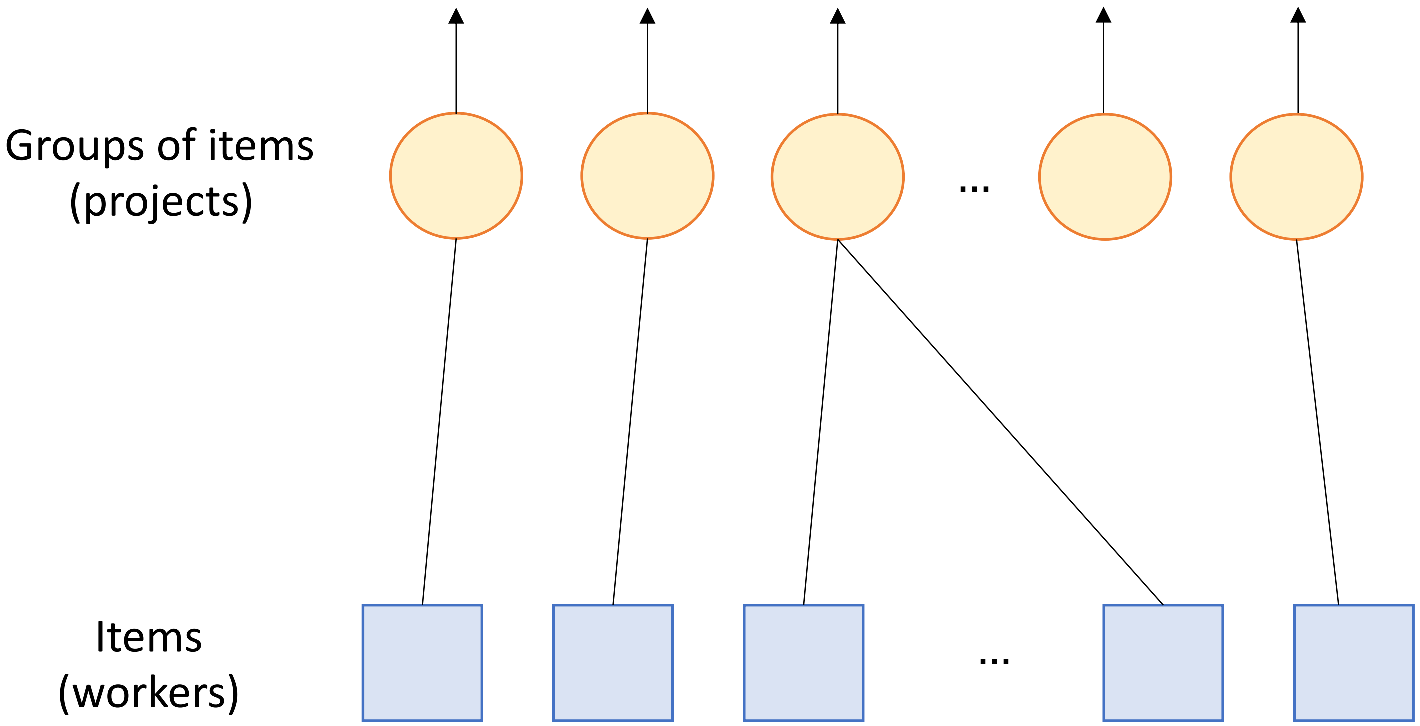
Problem formulation

- Selection of a subset of items of given cardinality from a pool of candidate items



Problem formulation (cont'd)

- Partition items to groups



Challenges

Group valuations:
value of a group of items may depend on the values of individual items in a complicated way

E.g. complements or supplements

Uncertainty:
uncertainty of individual item values may affect the expected value of a group of items in subtle ways

E.g. predictable vs high-risk high-return items

Computation complexity:
selection or assignment of items typically amounts to solving combinatorial optimization problems that are NP hard

Need for simple algorithms:
it is common assign items to groups by simple algorithms using individual item scores

E.g. select a set of items with highest individual item scores

Benefits of algorithms based on item scores

- Dynamic environments: scalability for changing pools of candidate items
  - Individual item scores only need to be computed once and do not need to be recomputed when the set of candidate items changes

- Distributed computation: algorithms for selection and assignment based on individual item scores are easy to implement in distributed systems

- Oracle queries: individual item scores may require estimating value of groups of items only for identical or similar items

- Conceptual simplicity: selection of items based on individual item scores is easy to understand by end users

Key questions

- Can algorithms that assign items to groups based on individual item scores achieve close to optimal group performance?
- If so, what are individual item scores that can guarantee this?
- How do simple, natural individual item scores perform?

Stochastic optimization problem formulation

- Given a ground set of elements $N = \{1,2,\dots,n\}$, valuation function $f: 2^N \times \mathbf{R}^n \to \mathbf{R}_+$ feasible set $\mathcal{F} \subseteq 2^N$, and distribution P: find $S^* \in \mathcal{F}$ that is a solution to:

$$\max_{S \in \mathcal{F}} u(S) := \mathbf{E}_{\boldsymbol{X} \sim P}[f(S, \boldsymbol{X})]$$

- Assumptions:
  - $\mathcal{F} = \{S \in 2^N : |S| = k\}$
  - $f(S, \boldsymbol{x}) = g(M_S(\boldsymbol{x}))$ where $g: \mathbf{R}_+^n \to \mathbf{R}_+$ is a symmetric monotone submodular value function
  - $\boldsymbol{X} = (X_1, X_2, \dots, X_n)$ are independent random variables, $X_i \sim P_i$

Note: $M_S(\boldsymbol{x})_i = x_i$ if $i \in S$ and $M_S(\boldsymbol{x})_i = \phi$ otherwise (ϕ is a minimal element)

Examples of valuation functions

Diminish returns of total value:

$$g(\boldsymbol{x}) = \bar{g}(\textstyle\sum_{i=1}^{n} x_i)$$

where $\bar{g}$ is increasing concave

Constant elasticity of substitution (CES):

$$g(\boldsymbol{x}) = (x_1^r + x_2^r + \cdots + x_n^r)^{1/r} \text{ for } r > 0$$

diminishing returns for $r \geq 1$

Best-shot:

$$g(\boldsymbol{x}) = \max\{x_1, x_2, \dots, x_n\}$$

Success probability:

$$g(\boldsymbol{x}) = 1 - \textstyle\prod_{i=1}^{n}(1 - p(x_i))$$

where $p: \mathbf{R} \to [0,1]$, increasing

Top-r:

$$g(\boldsymbol{x}) = x_{(1)} + x_{(2)} + \cdots + x_{(r)} \text{ for } 1 \leq r \leq n$$

where $x_{(1)}, x_{(2)}, \dots, x_{(n)}$ are values $x_1, x_2, \dots, x_n$ arranged in decreasing order

Computation by using test scores

- Computation model introduced by [Kleinberg and Raghu 2015]: an algorithm has access only to (estimates) of individual item scores (test scores)

- We can think of test scores as a mapping from $(g, \mathcal{F}, P_i)$ to a real value:

$$a_i = h(g, \mathcal{F}, P_i)$$

- The sample mean version:

$$a_i = \frac{1}{T} \sum_{t=1}^{T} \varphi\left(X^{(t)}; g, \mathcal{F}, P_i\right)$$

where $\boldsymbol{x} \mapsto \varphi(\boldsymbol{x}; g, F, P_i)$ is given and $X^{(t)}$ are independent samples from P_i^d

Examples of test scores

- Mean test scores:

$$a_i = \mathbf{E}_{X_i \sim P_i}[X_i]$$

- Standard quantile test scores:

$$a_i = q_i(\theta)$$

where $q_i(\theta)$ is the θ-quantile $q_i(\theta) = \inf\{x \in \mathbf{R}: P_i(x) \geq \theta\}$

- Quantile test scores:

$$a_i = \mathbf{E}_{X_i \sim P_i}[X_i \mid P_i(X_i) \geq \theta]$$

None of these test scores can guarantee a constant-factor approximation

Main result: approximation guarantee

- Thm. Assume g is a symmetric monotone function that satisfies the extended submodularity condition: for all $\boldsymbol{x}, \boldsymbol{y}$ such that $g(\boldsymbol{x}) \leq g(\boldsymbol{y})$,

$$g(\boldsymbol{x}, z) - g(\boldsymbol{x}) \geq g(\boldsymbol{y}, z) - g(\boldsymbol{y}) \text{ for all } z \in \mathbf{R}_+$$

  Then, there exist test scores that guarantee a $(1 - 1/e)/(5 - 1/e)$-factor approximation.

- In particular, the theorem holds for replication test scores:

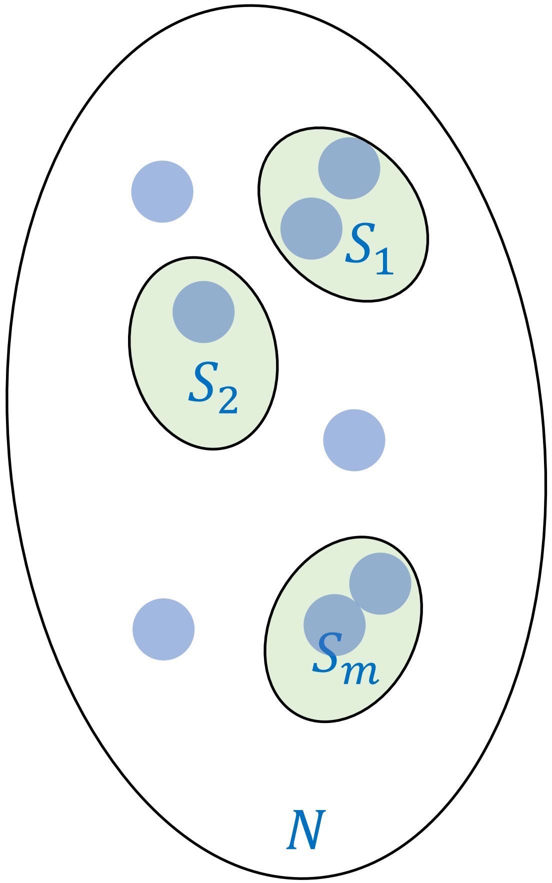
$$a_i = \mathbf{E}_{\boldsymbol{X} \sim P_i^k}[g(\boldsymbol{X})]$$

  (Expected value of a virtual set of independent copies of an item.)

- Proof based on a new approach that reduces the optimization problem to approximating the objective function by “sketch” functions

Stochastic submodular welfare maximization

maximize	$\sum_{i=1}^{m} u_j(S_j)$
over	$S_1, S_2, \dots, S_m \in 2^N$
subject to:	
	$\lvert S_j \rvert = k_j$ for $j = 1,2,\dots,m$
	$S_i \cap S_j = \emptyset$ for all $i \neq j$



$$u_j(S_j) := \mathbf{E}\left[g_j\left(M_{S_j}(X_{1,j}, \dots, X_{n,j})\right)\right]$$

$X_{i,j}$ are independent random variables, $X_{i,j} \sim P_{i,j}$

g_j is a symmetric monotone submodular value function

Approximation for welfare maximization

- Thm. Suppose that valuation functions satisfy the extended submodularity condition and let k denote the largest cardinality constraint.

  Then, there exists a test score algorithm using replication test scores that guarantees a $1/(24(\log(k) + 1)$-factor approximation.

- Proof based on the same framework as for maximizing a stochastic submodular function subject to a cardinality constraint, but using a different sketch and a more intricate greedy assignment algorithm

Greedy algorithm for welfare maximisation

Input: N, M, replication test scores $a_{i,j}^{r} = \mathbf{E}_{\boldsymbol{X}\sim P_i^r}\left[g_j(\boldsymbol{X})\right]$

Initialization: $S_1, S_2, \dots, S_m = \emptyset$, $A = N$, $P = M$

while $|A| > 0$ and $|P| > 0$ **do**:

$$(i^*, j^*) = \arg \max_{(i,j)\in A\times P} \frac{a_{i,j}^{|S_j|+1}}{|S_j|+1}$$

$S_{j^*} \leftarrow S_{j^*} \cup \{i^*\}$ and $A \leftarrow A \setminus \{i^*\}$

if $\left|S_{j^*}\right| = k_{j^*}$ **then** $P \leftarrow P \setminus \{j^*\}$

Output: $S_1, S_2, \dots, S_m$

Part II – summary points

- Test score selection of items can provide a constant-factor approximation for a broad class of submodular utility functions

- This is guaranteed by a special type of test scores: replication test scores

- Submodular welfare maximization: $\Omega(1/\log(k))$-approximation by replication test scores, where k is the maximum number of assignments to a project