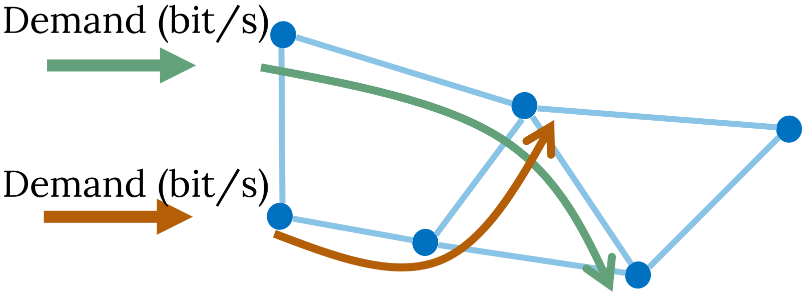
Solving **Max-Min Fair** Resource Allocations Quickly on **Large** Graphs

Pooria Namyar, Behnaz Arzani, Srikanth Kandula, Santiago Segarra,
Daniel Crankshaw, Umesh Krishnaswamy, Ramesh Govindan, Himanshu Raj

USC Viterbi School of Engineering
Microsoft
RICE

Example of Resource Allocation

Route demands in the WAN



SWAN (Microsoft), B4 (Google)

Requirements

Efficient

Utilize Resources
Maximize Profit

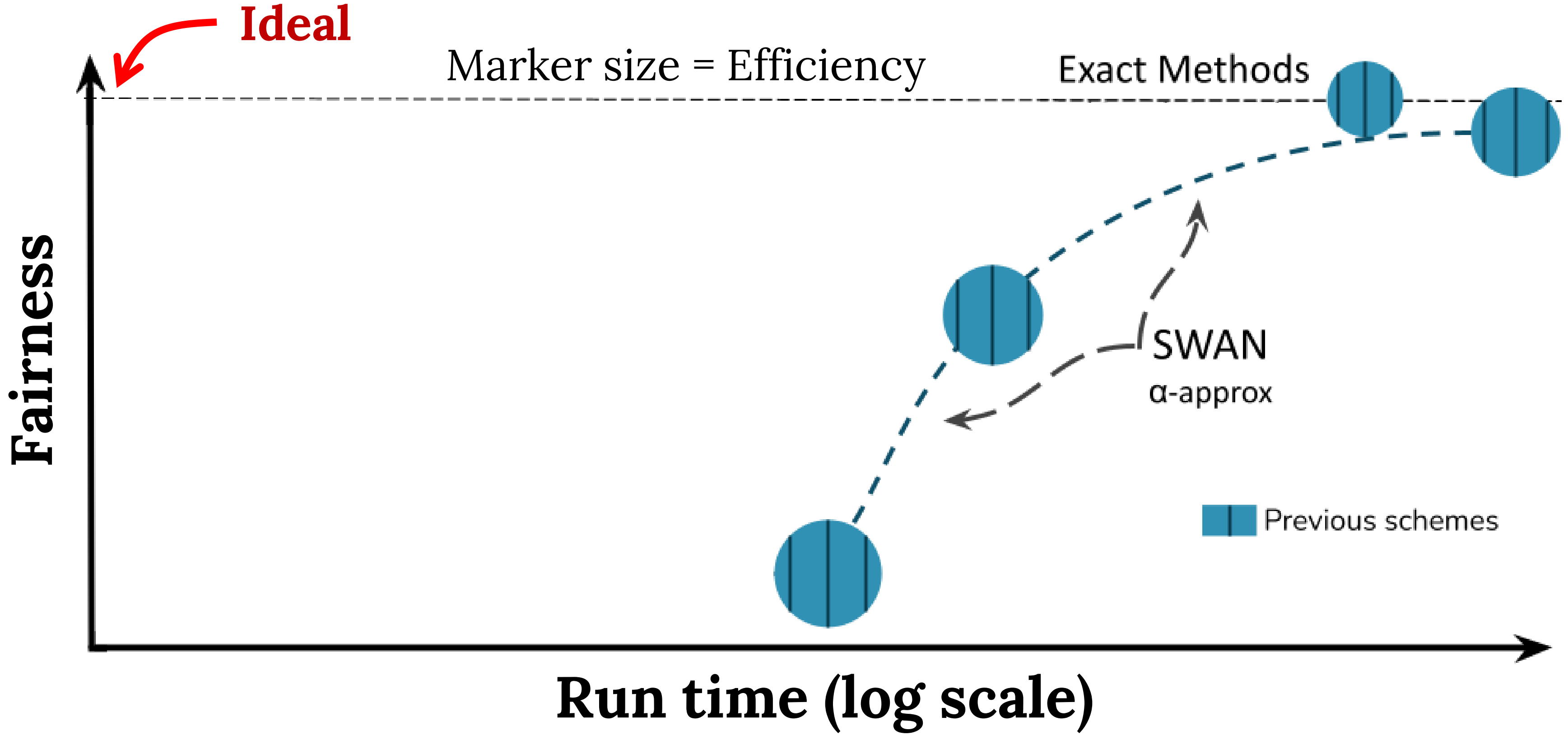
Fair

Across Tenants
and Services

Fast

React to changes
quickly

Existing Fair and Efficient Allocators are Slow



Ideal

Marker size = Efficiency

Exact Methods

SWAN

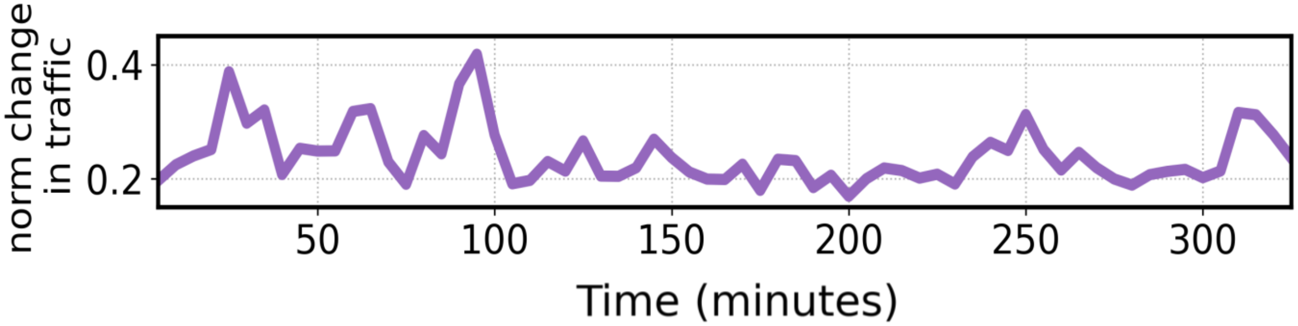
α-approx

▥ Previous schemes

Fairness

Run time (log scale)

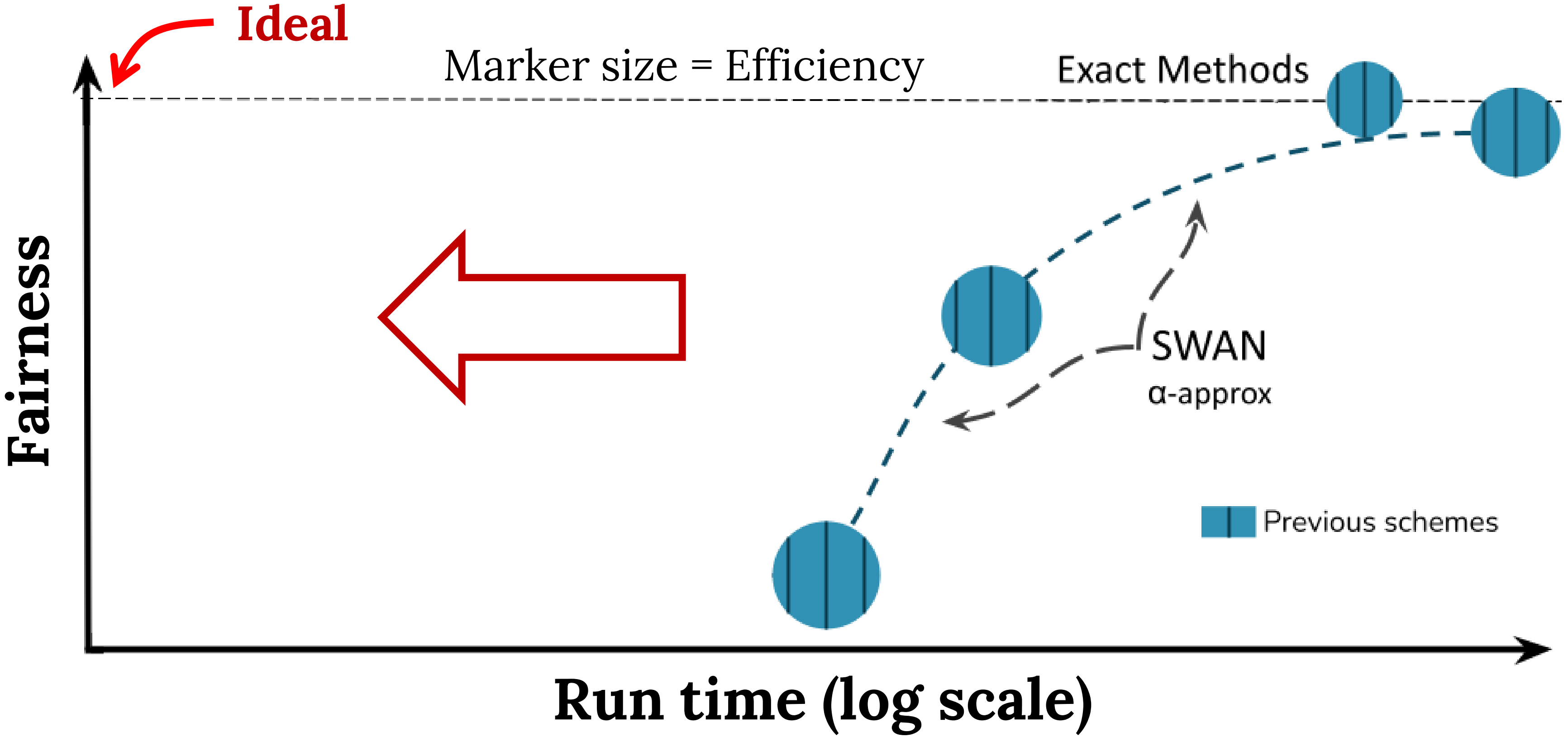
Speed Matters in Fair and Efficient Allocation



Slow Allocator: 30% drop in Efficiency.

60% drop in Fairness.

Existing Fair and Efficient Allocators are Slow



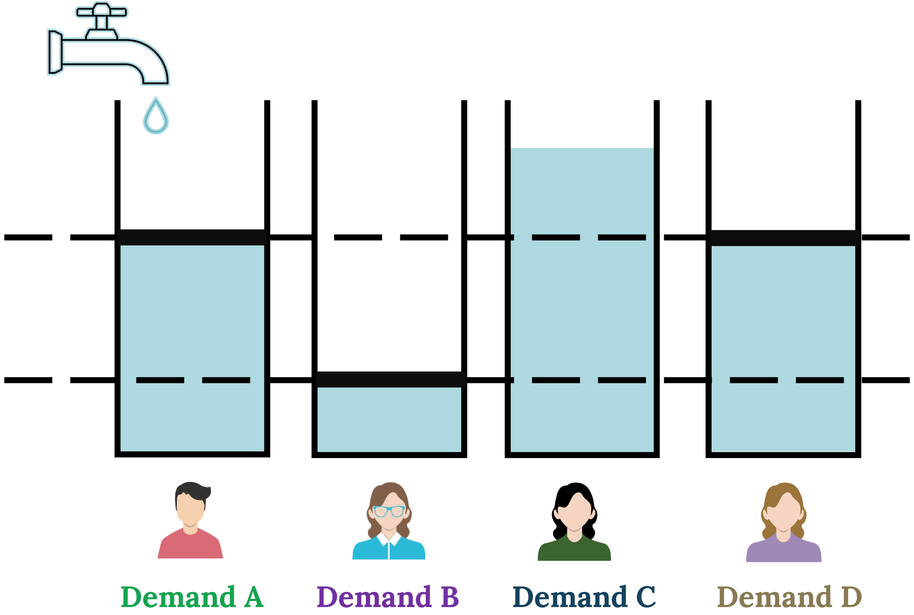
Max-Min Fair Resource Allocation

Common in practice:

- B4 (Google)
- SWAN (Microsoft)

Cannot allocate more to A and D.

Cannot allocate more to B.

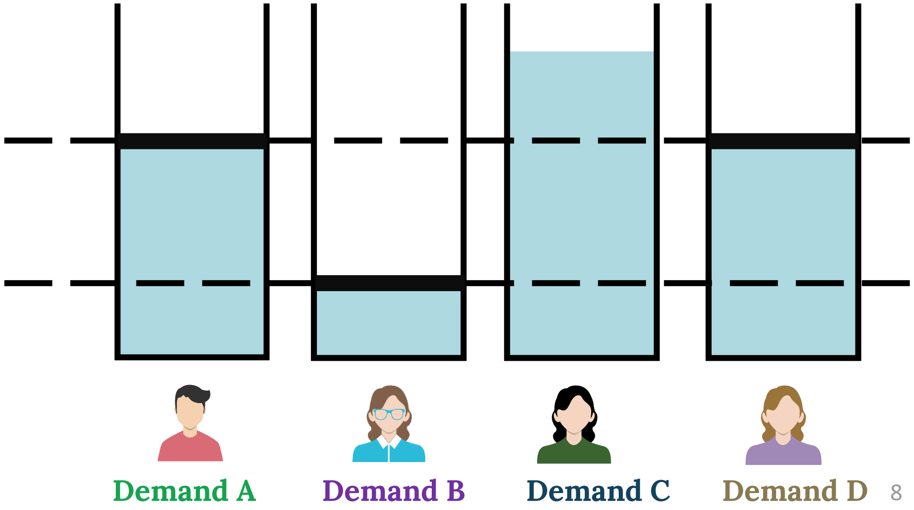


Existing Max-Min Fair Allocators

Iterate

(1) Maximize the minimum allocation among remaining demands.

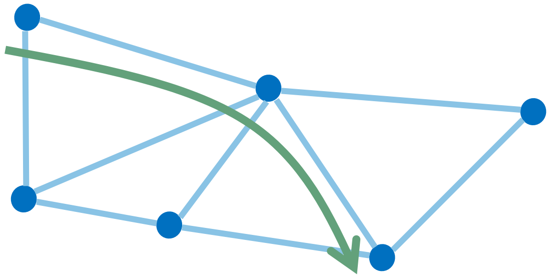
(2) Fix the demands that cannot receive more.



Existing Max-Min Fair Allocators

Single-Path Waterfilling

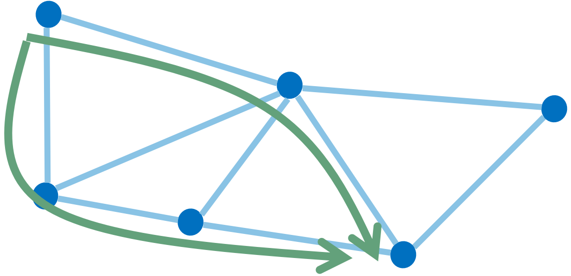
Example: K-waterfilling



- Fast
- Unfair and Inefficient

Iterative Optimization-based

Example: SWAN



Multi-Path → Optimization

- Fair and Efficient
- Slow

Our Solution: Soroush

Multi-path Waterfilling

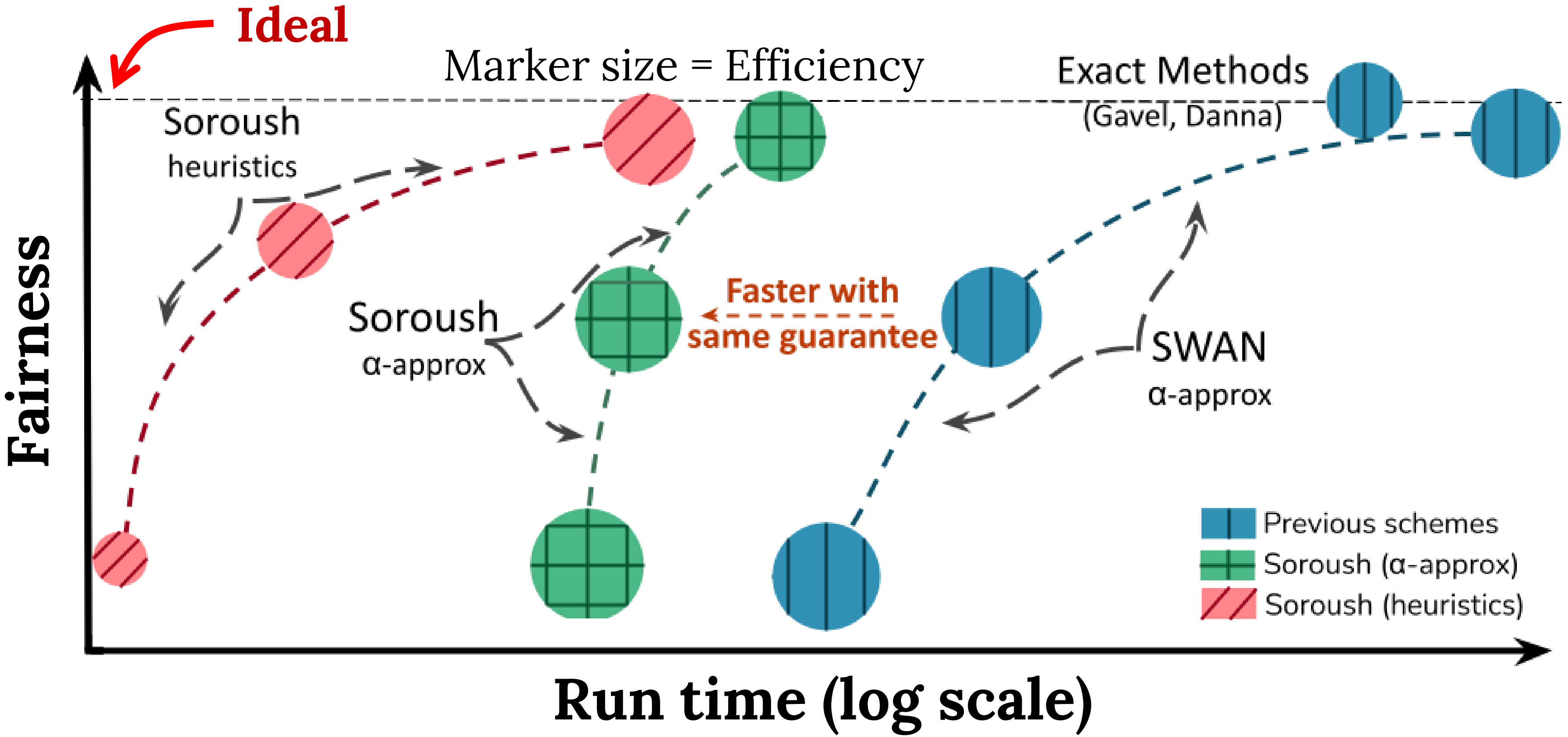
Fast single-shot optimization

Adaptive Waterfiller

Geometric Binner

Equi-depth Binner

Soroush Empirically Pareto-dominates Prior Work



Our Solution: Soroush

Multi-path Waterfilling

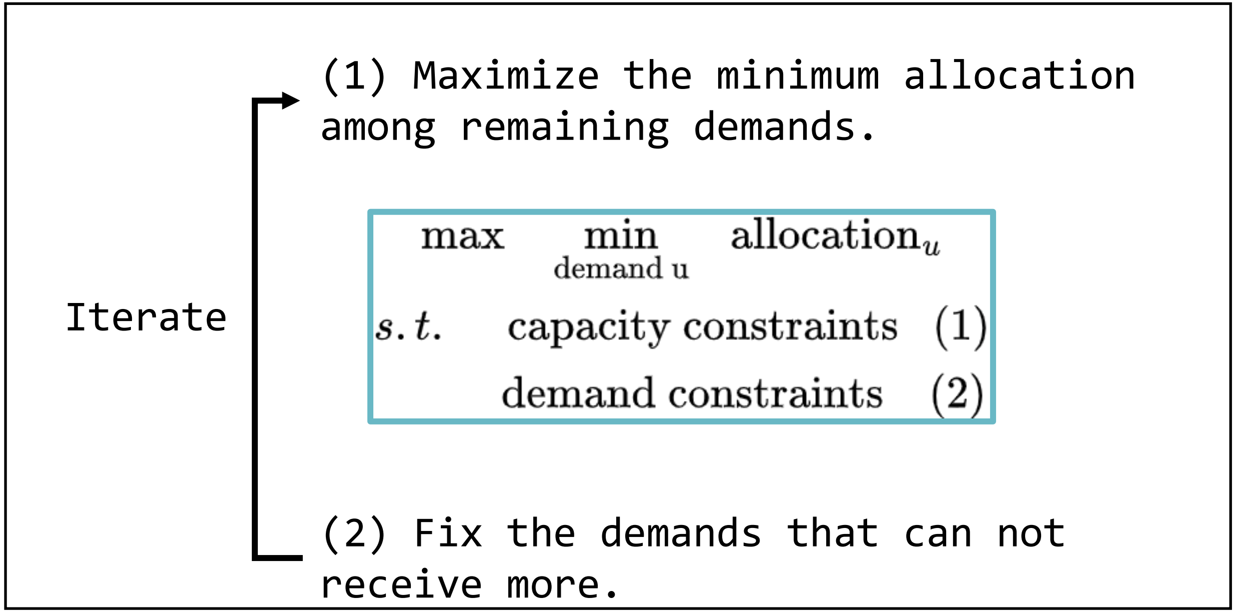
Fast single-shot optimization

Adaptive Waterfiller

Geometric Binner

Equi-depth Binner

Towards Single-Shot Max-Min Fair Allocation

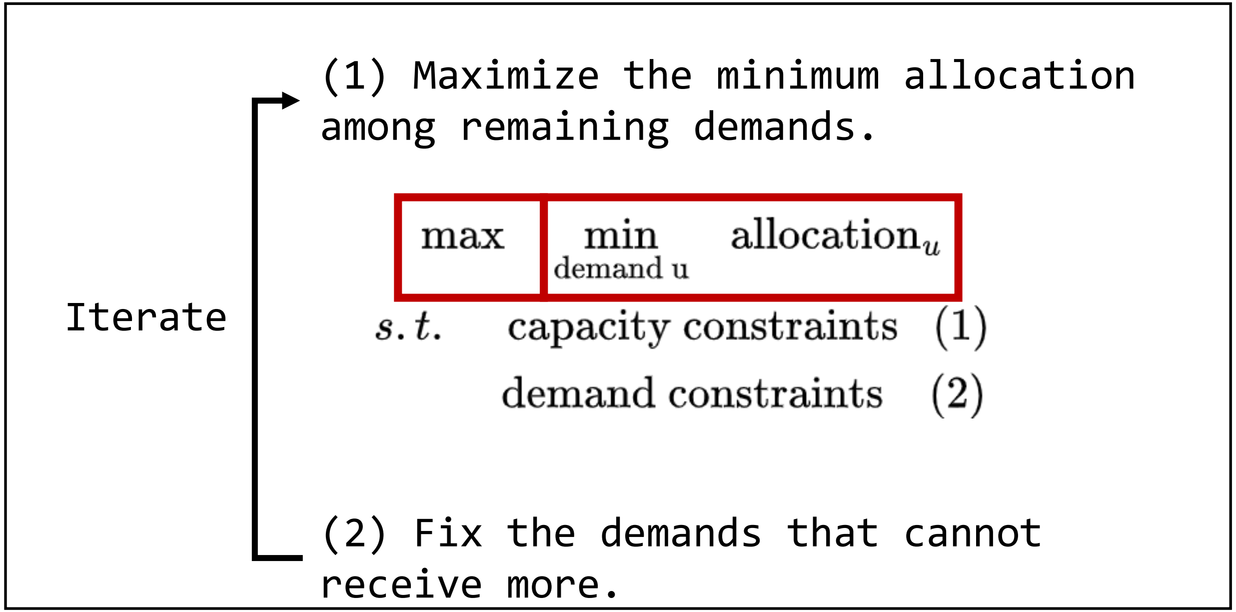

Iterate

(1) Maximize the minimum allocation among remaining demands.

$$\begin{aligned} &\max \quad \min_{\text{demand u}} \quad \text{allocation}_u \\ &s.t. \quad \text{capacity constraints} \quad (1) \\ &\qquad\;\; \text{demand constraints} \quad (2) \end{aligned}$$

(2) Fix the demands that can not receive more.

Goal: Single Fast Optimization



Iterate

(1) Maximize the minimum allocation among remaining demands.

$$\boxed{\max}\ \boxed{\min_{\text{demand u}} \quad \text{allocation}_u}$$
$$\begin{aligned} s.t. \quad & \text{capacity constraints} \quad (1) \\ & \text{demand constraints} \quad (2) \end{aligned}$$

(2) Fix the demands that cannot receive more.

Iterate

1) Find demand with minimum allocation → sort the demands

2) Maximize the minimum demand’s allocation.

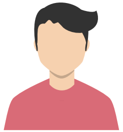
Single-Shot Max-Min Fair Allocator

1) Find the demand with minimum allocation → Sorting Network

Hongqiang Harry Liu et al., Traffic Engineering with Forward Fault Correction, SIGCOMM14

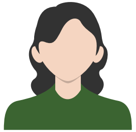
2) **Maximize the minimum demand's allocation.**

Assume we know the order of allocations



Demand A

$\leq$

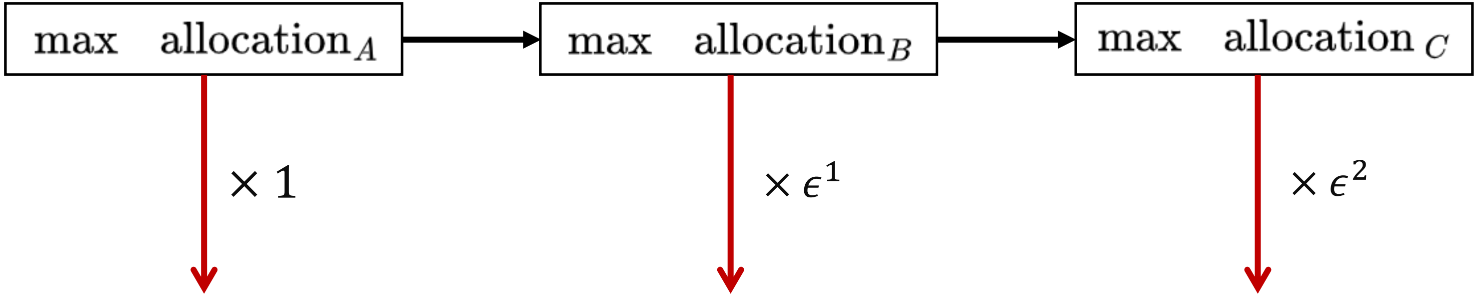


Demand B

$\leq$

Demand C

`Iterative`



$\max \quad \text{allocation}_A \rightarrow \max \quad \text{allocation}_B \rightarrow \max \quad \text{allocation}_C$

$\times 1$

$\times \epsilon^1$

$\times \epsilon^2$

`ϵ-weighting`
$(0 < \epsilon < 1)$

$$\text{allocation}_A + \epsilon \times \text{allocation}_B + \epsilon^2 \times \text{allocation}_C$$

Key: Incentivize the solver to assign in order

Single-Shot Max-Min Fair Optimization

$$max \sum_{\text{demand } k} \epsilon^k f_k$$

$$s.t.$$

$$demand\ \ constraints\ (1)$$

$$capacity\ \ constraints\ (2)$$

$$sorting\ \ network\ \ constraints\ (3)$$

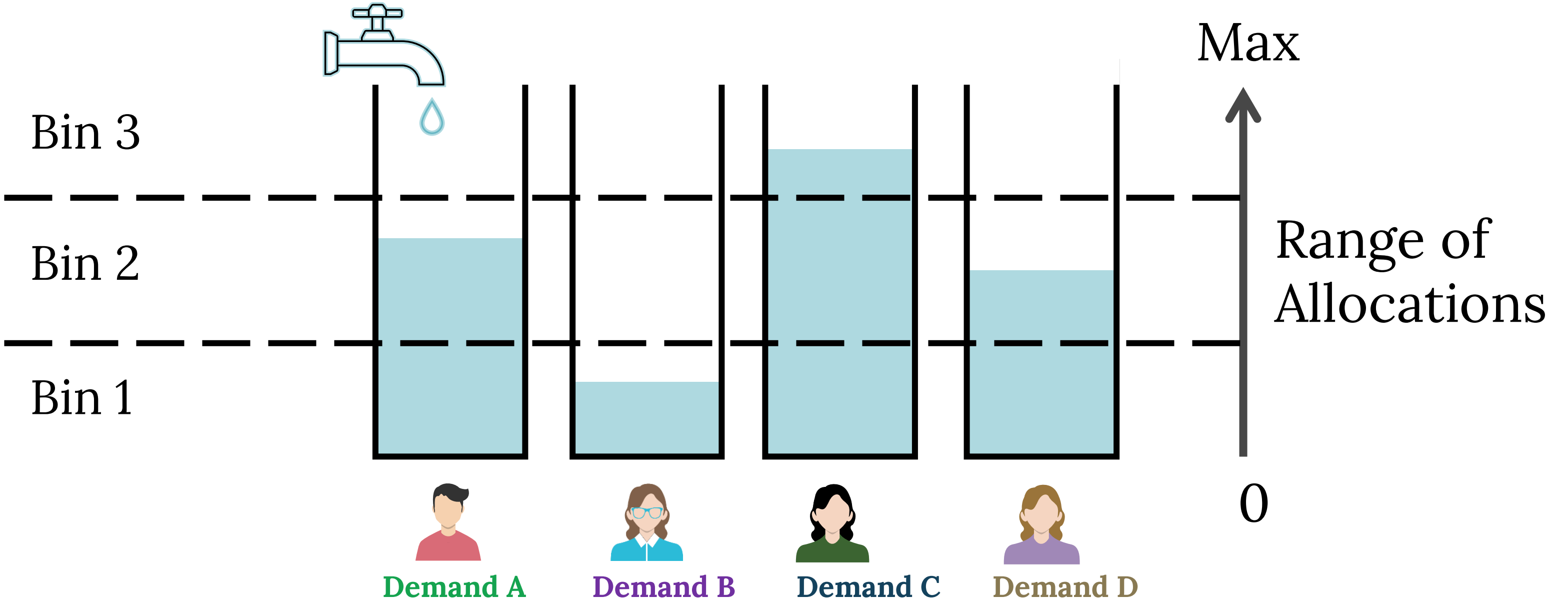
Theorem: for small enough ϵ, the optimization **yields the max-min fair solution**.

Slow

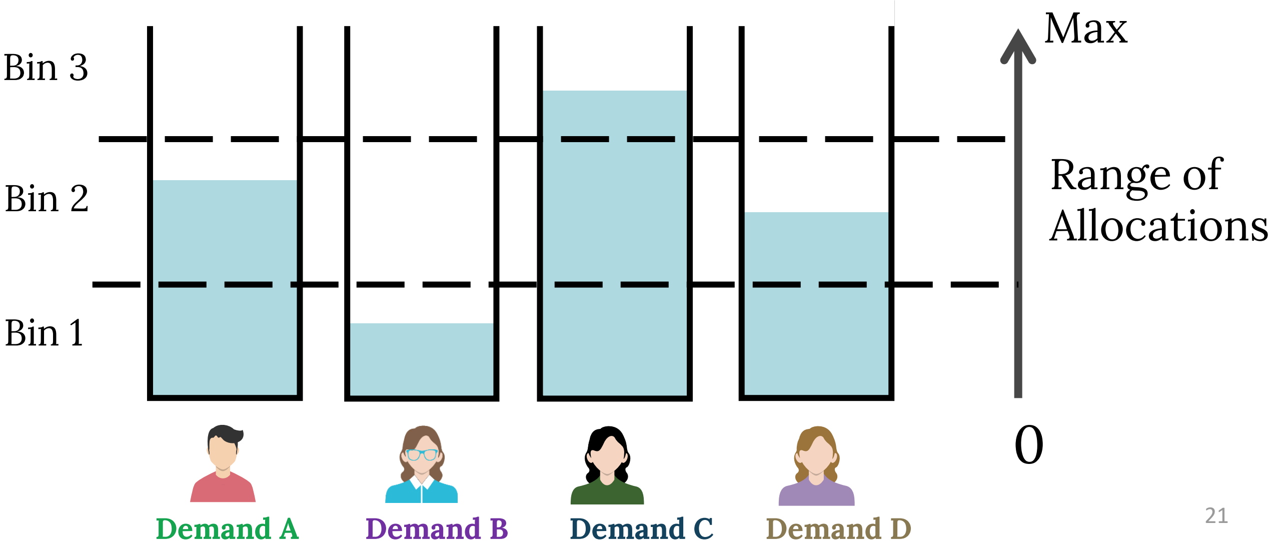
Numerical Issues
(~ million demands)

Can we make it faster?

Approx Max-Min Fair instead of Per-User



Approx Max-Min Fair instead of Per-User

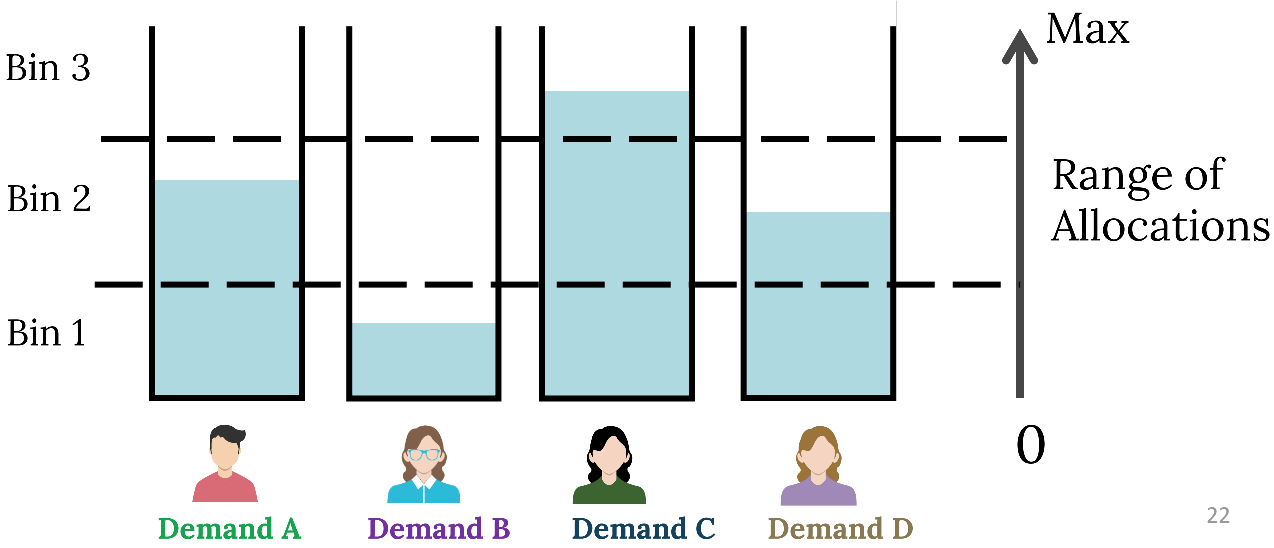


Approx Max-Min Fair instead of Per-User

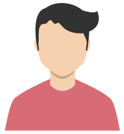
Iterative (next bin)

(1) Maximize the total allocation from a bin.

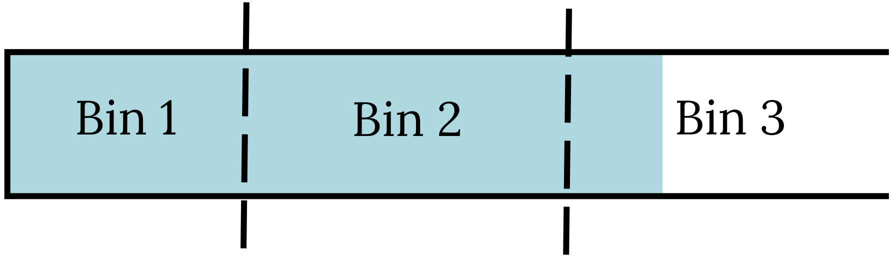
(2) Fix the demands that do not receive full rate.

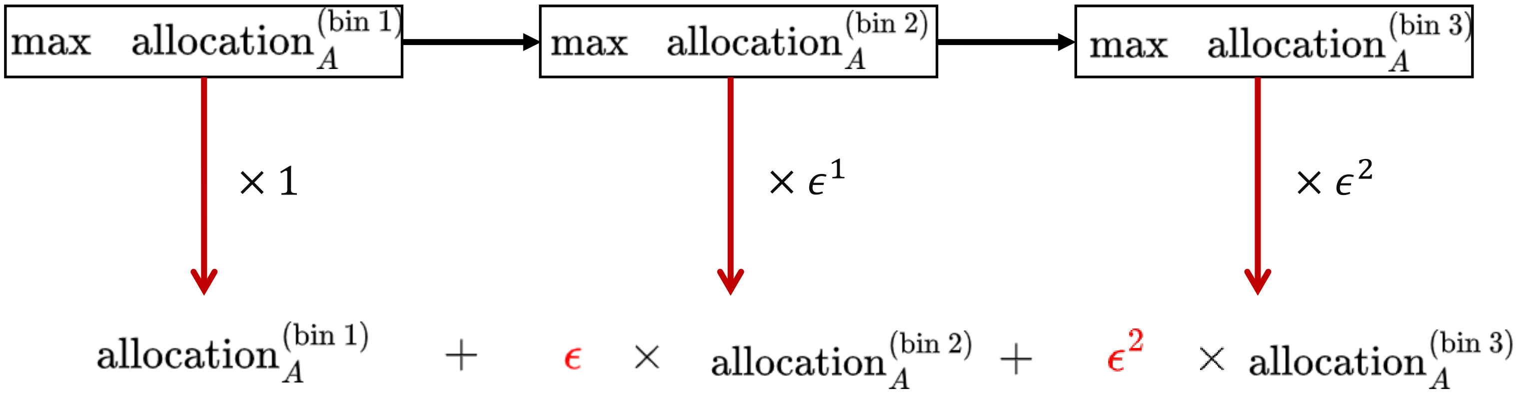


Single-Shot Approx Max-Min Fair



Demand A





Iterative

$\max \quad \text{allocation}_A^{(\text{bin 1})} \rightarrow \max \quad \text{allocation}_A^{(\text{bin 2})} \rightarrow \max \quad \text{allocation}_A^{(\text{bin 3})}$

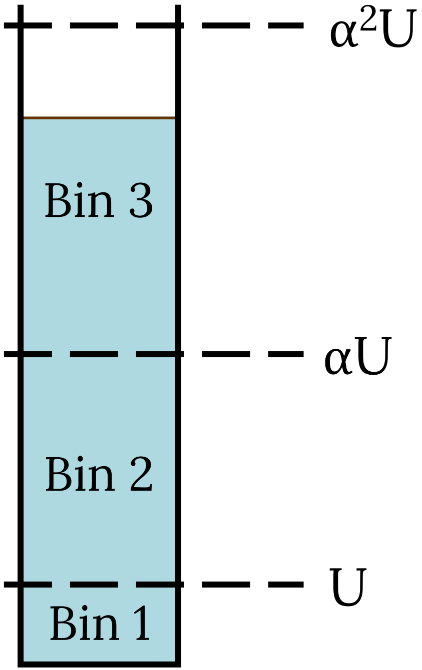
$\times 1$ $\quad \times \epsilon^1$ $\quad \times \epsilon^2$

ϵ-weighting
$(0 < \epsilon < 1)$

$\text{allocation}_A^{(\text{bin 1})} + \epsilon \times \text{allocation}_A^{(\text{bin 2})} + \epsilon^2 \times \text{allocation}_A^{(\text{bin 3})}$

Key: Incentivize the solver to allocate bins in order

Our Fast Approximate Max-Min Fair Solver



Geometric Binner (GB):
Binning + ϵ -weighting + Geometric sizes.

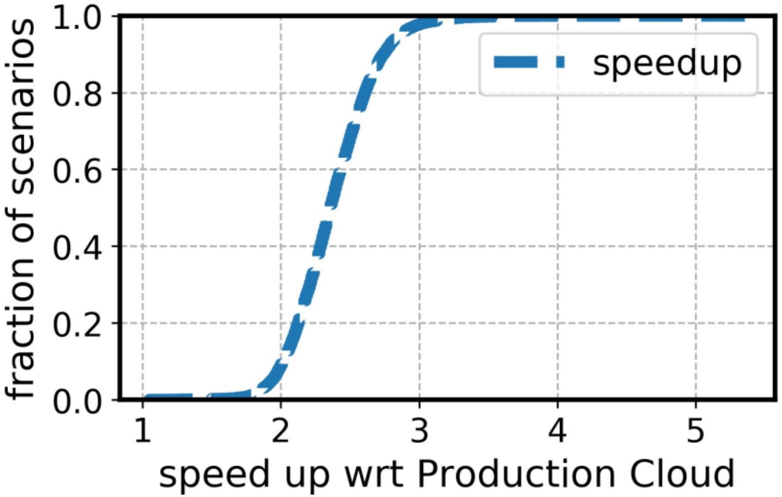
Theorem: GB's allocation is always within a α factor of optimal allocation for every demand.

Empirically and theoretically faster than existing methods.

Hong et al., Achieving high utilization with software-driven WAN, SIGCOMM'13

Our Method is Deployed in Microsoft WAN

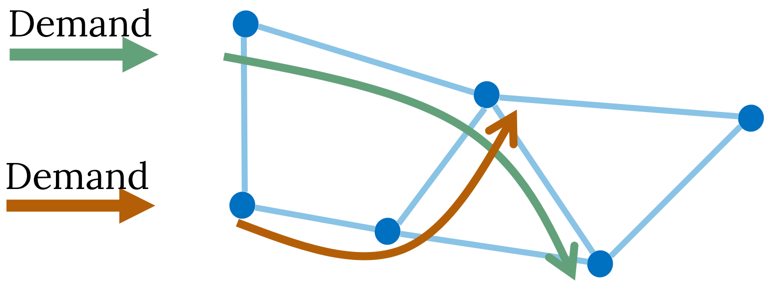
- **Matches the efficiency and fairness** of the previous iterative allocator.
- On average, 2.4x and up to more than 5x **faster.**



A Graph Model for Resource Allocation

Route demands in the WAN

SWAN (Microsoft), B4 (Google)



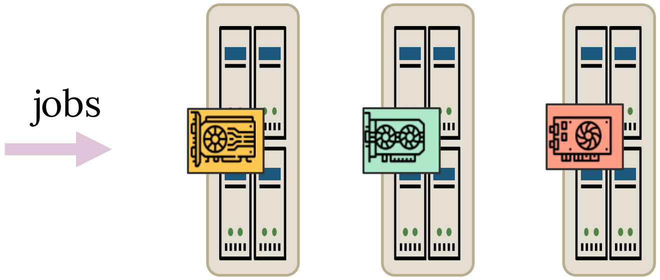
Resources: Links

Demands: Network demands

Path: Group of links we allocate together

Split jobs over multiple servers

Gavel (OSDI'20)



Resources: CPU, GPU, Memory

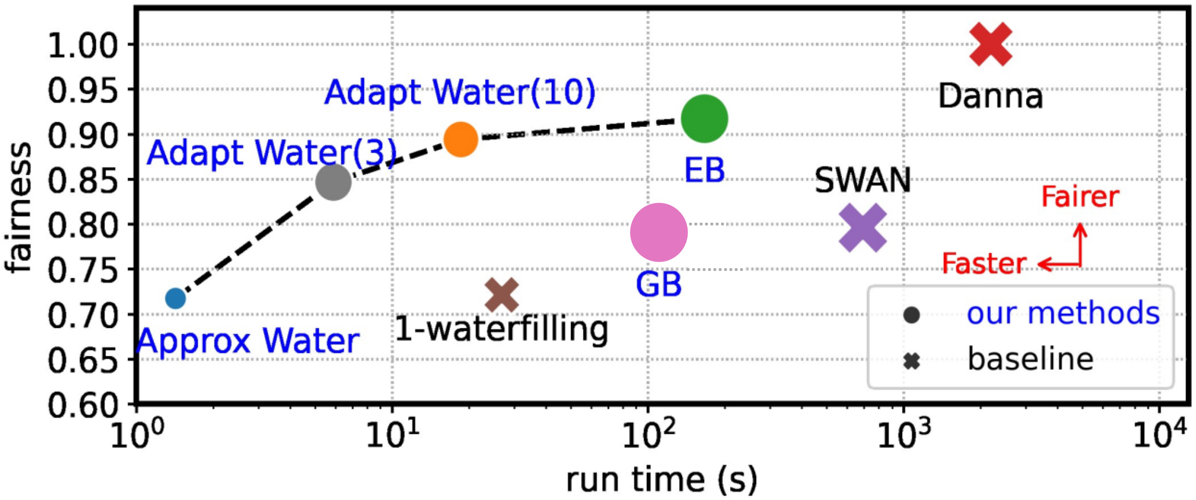
Demands: Jobs

Path: Group of resources we allocate together

Soroush Empirically Pareto-dominates Prior Work

Traffic engineering

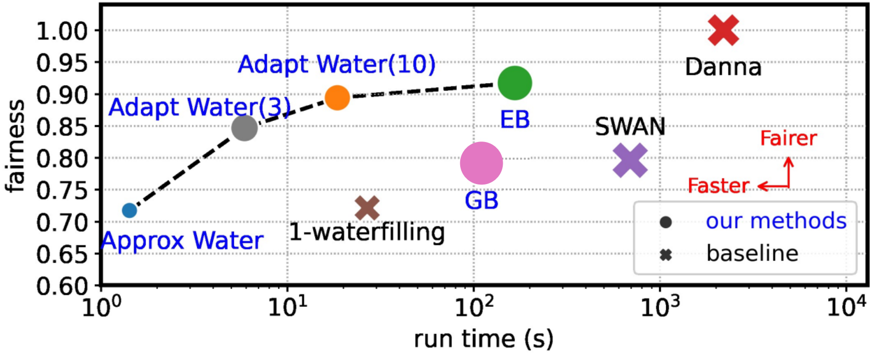
- Danna et al → exact
- SWAN → α-approximate
- 1-waterfilling → heuristic



Soroush Empirically Pareto-dominates Prior Work

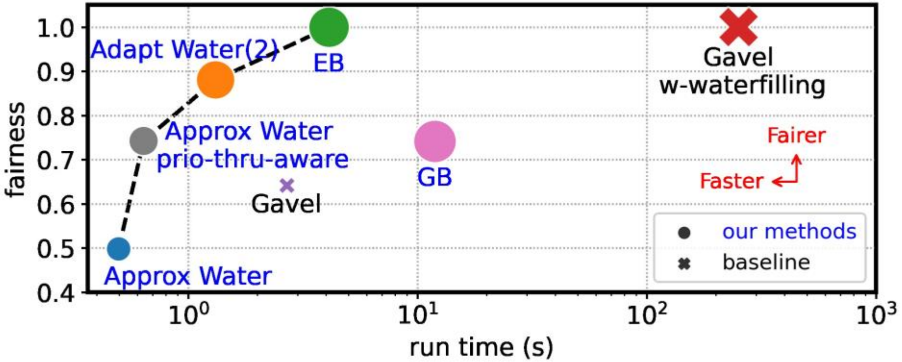
Traffic engineering

- Danna et al → exact
- SWAN → α-approximate
- 1-waterfilling → heuristic



Cluster scheduling

- Gavel w/ waterfilling → exact
- Gavel → heuristic



Soroush: General & Scalable Max-Min Fair Allocator

General Graph Model (TE, CS)

Fast & Scalable

Users can control the trade-off.

Future Work:

(1) Other domains

(2) Distributed setting

Contact: namyar@usc.edu

Code: github.com/microsoft/Soroush