Differentiable Ranking and Sorting using Optimal Transport

M. Cuturi O. Teboul J.-P. Vert

`google_research/soft_sort/`

Google AI Brain Team

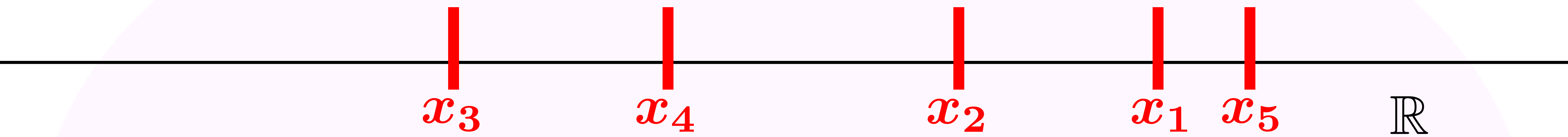
@ East Exhibition Hall B + C **#60**

Ranking and Sorting numbers ...

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$

Ranking and Sorting numbers ...

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$



Sorting permutation $\sigma(\mathbf{x}) = (3, 4, 2, 1, 5)$

Ranking and Sorting numbers ...

$$\mathbf{x} = (x_1, x_2, x_3, x_4, x_5)$$

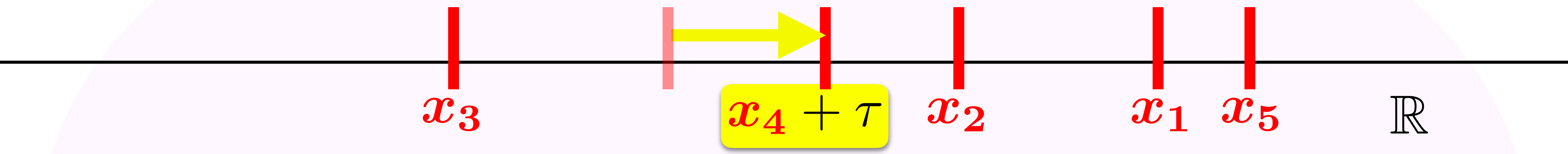
Sorting permutation $\sigma(\mathbf{x}) = (3, 4, 2, 1, 5)$

$$\underbrace{R\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right)}_{:= \sigma(\mathbf{x})^{-1}} = \begin{bmatrix} 4 \\ 3 \\ 1 \\ 2 \\ 5 \end{bmatrix} \qquad \underbrace{S\left(\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}\right)}_{:= \mathbf{x}_{\sigma(\mathbf{x})}} = \begin{bmatrix} x_3 \\ x_4 \\ x_2 \\ x_1 \\ x_5 \end{bmatrix}$$

… are **not** differentiable operations

When adding a small perturbation τ to x_4,

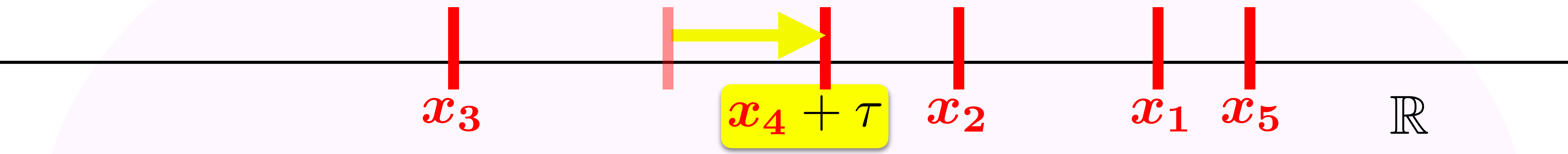
$$\mathbf{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \boldsymbol{x_4} + \tau, \boldsymbol{x_5})$$



… are **not** differentiable operations

When adding a small perturbation τ to x_4,

$$\mathbf{x} = (x_1, x_2, x_3, x_4+\tau, x_5)$$

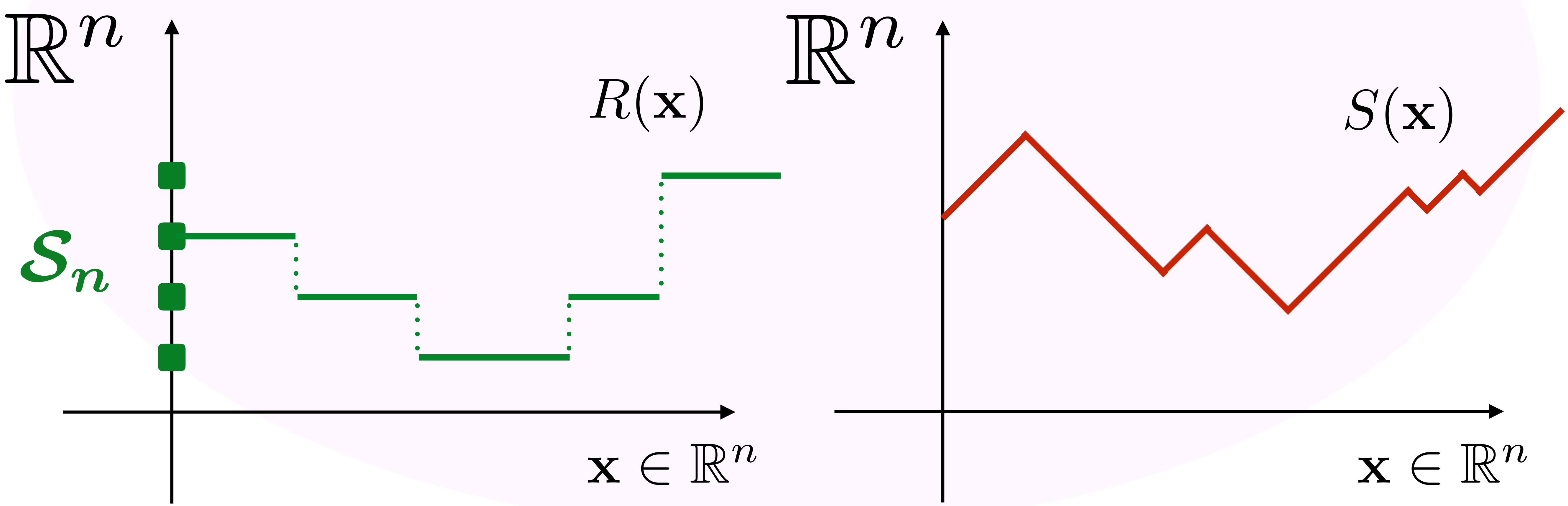


R is unchanged, S varies only in one coordinate.

$$R\left(\begin{bmatrix}x_1\\x_2\\x_3\\x_4+\tau\\x_5\end{bmatrix}\right)=\begin{bmatrix}4\\3\\1\\2\\5\end{bmatrix} \qquad S\left(\begin{bmatrix}x_1\\x_2\\x_3\\x_4+\tau\\x_5\end{bmatrix}\right)=\begin{bmatrix}x_3\\x_4+\tau\\x_2\\x_1\\x_5\end{bmatrix}$$

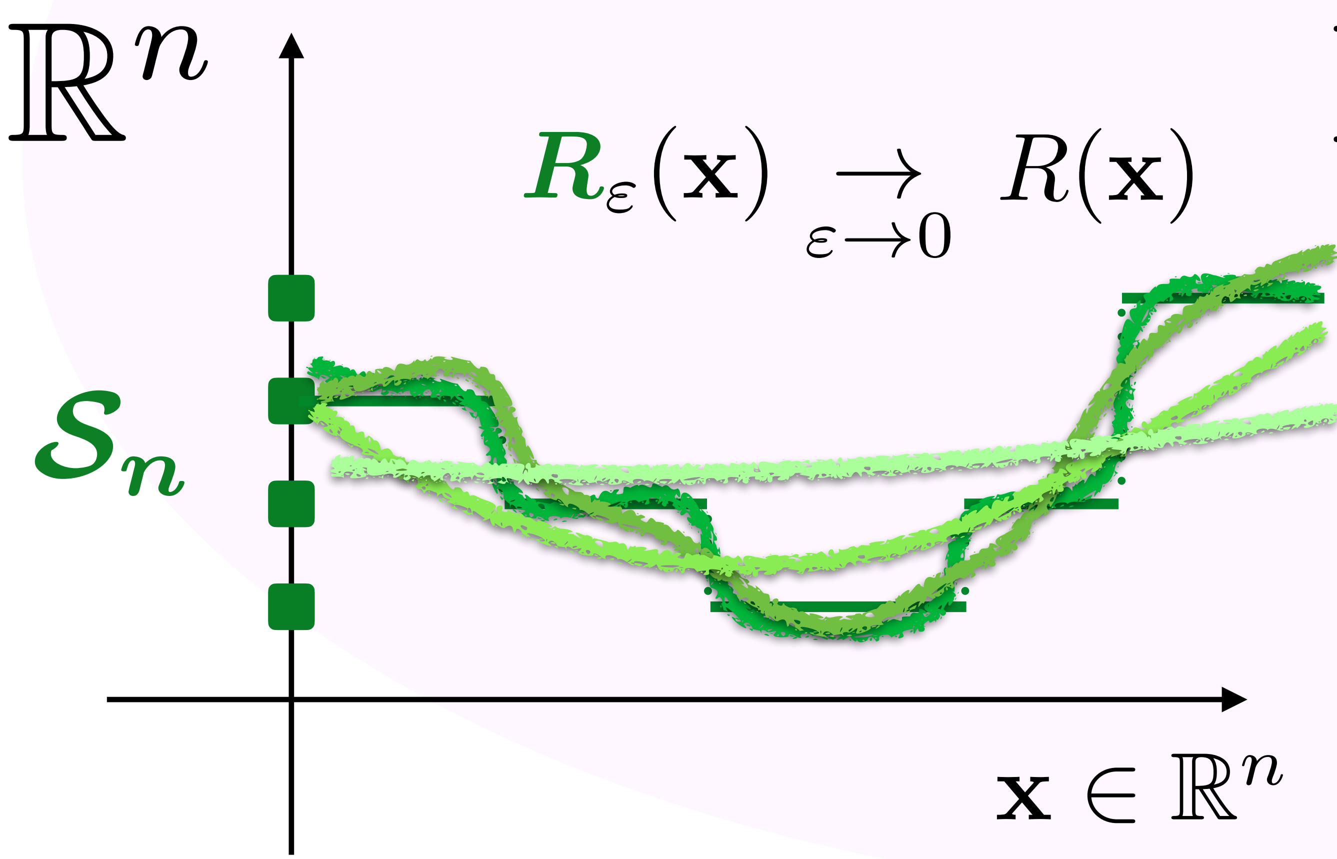
Soft Ranks and Sort Operators

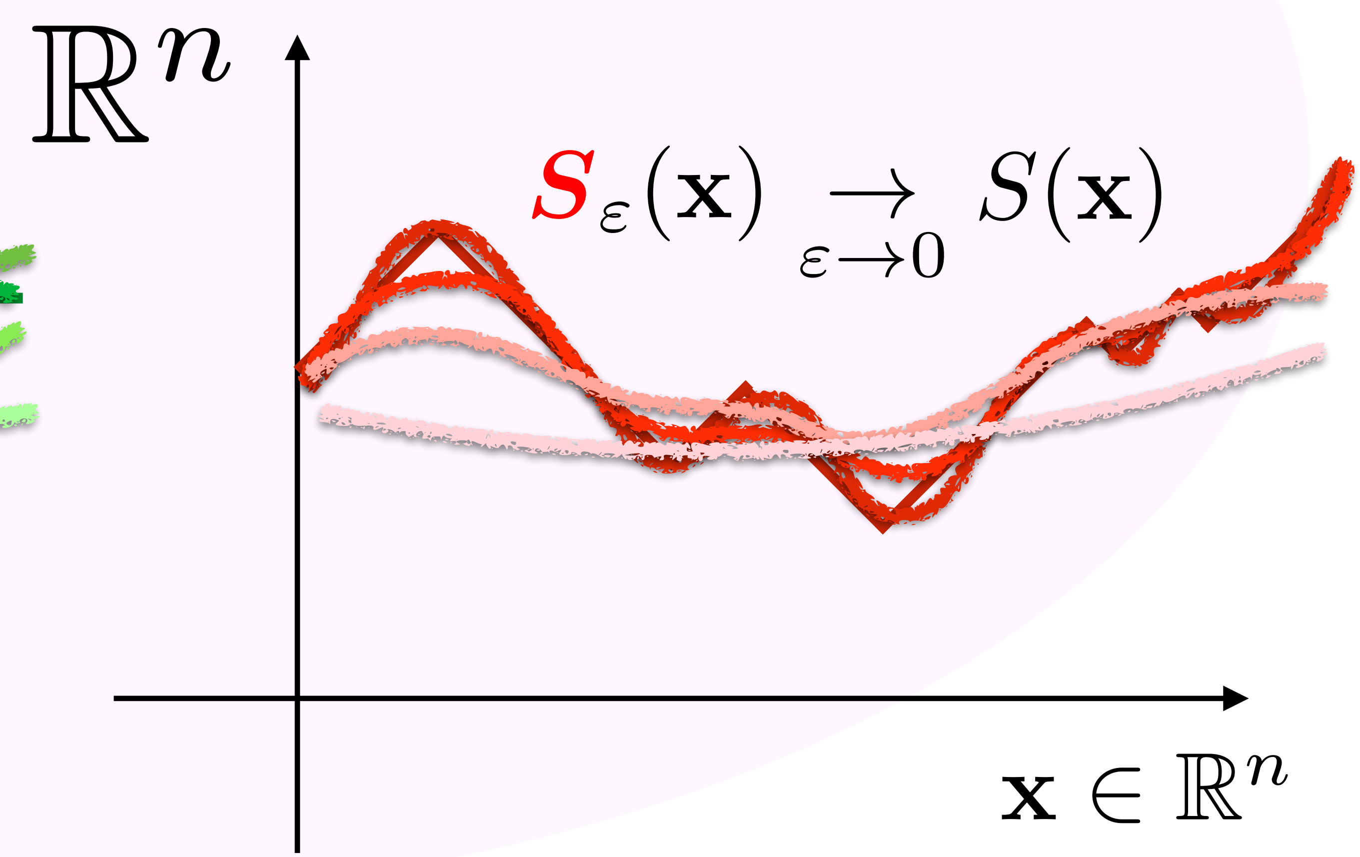
Problem: Despite appearing frequently in ML, R & S **are staircase/broken lines** like functions.



Soft Ranks and Sort Operators

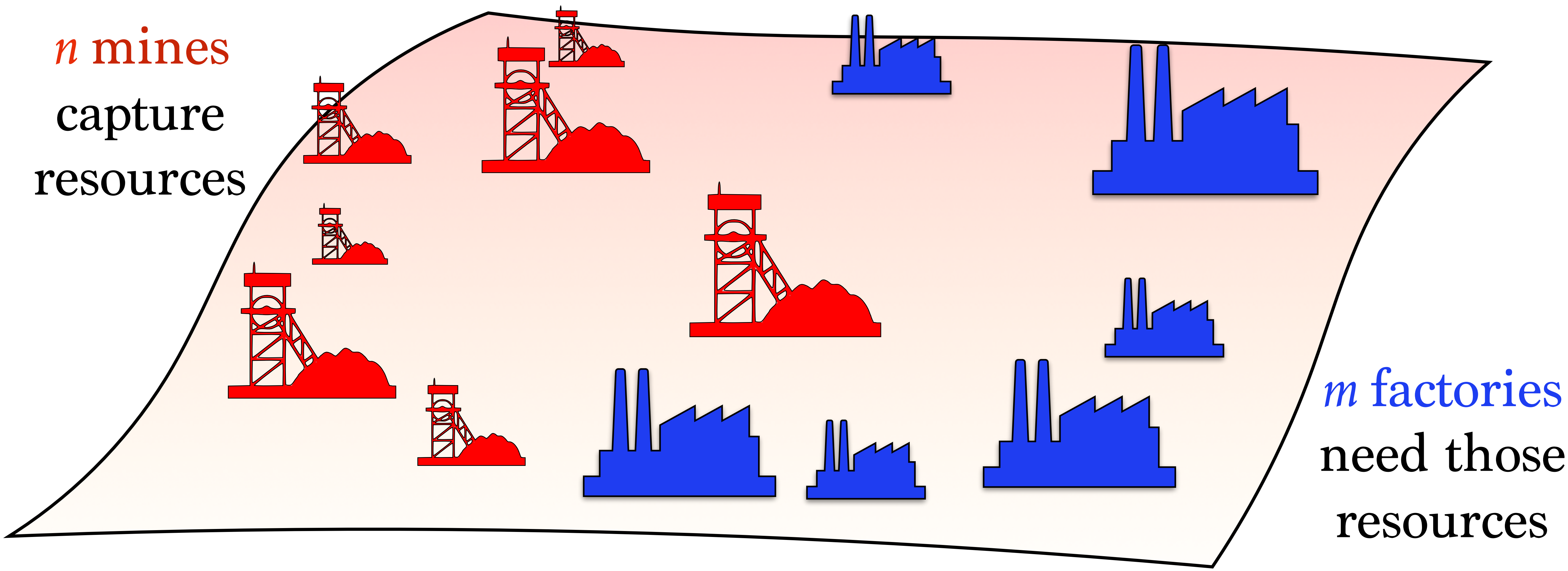
Goal: approximation functions for *R/S*, **arbitrarily close** to the true *R/S* vector outputs, programatically differentiable everywhere.





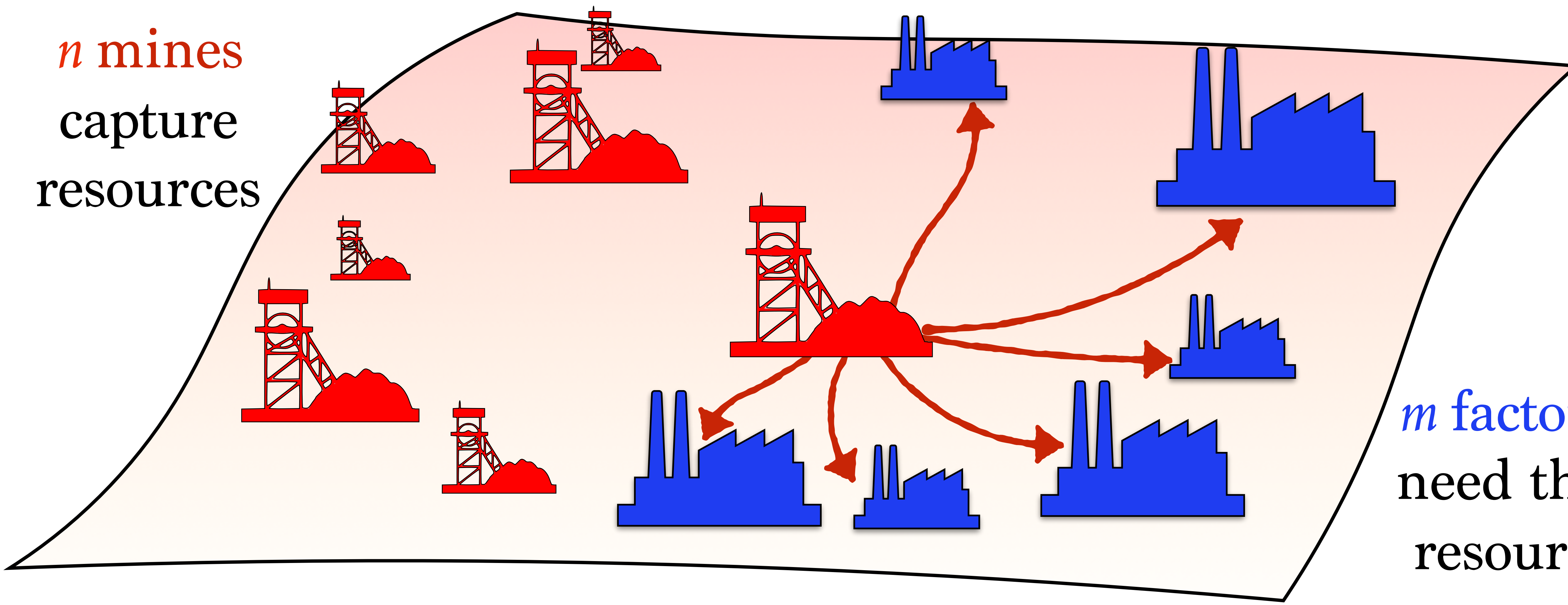
Our Tool: *Optimal Transport*

Sketch: mined resources must be dispatched towards factories.



Our Tool: *Optimal Transport*

Sketch: mined resources must be dispatched towards factories.



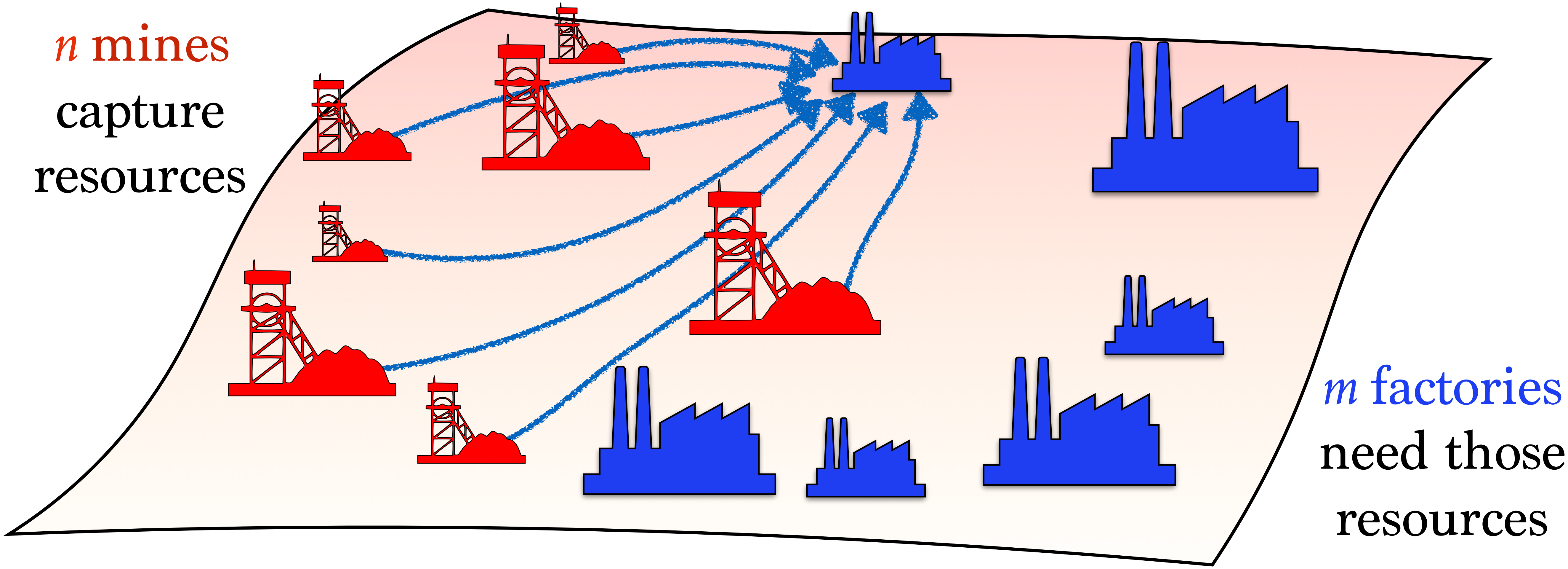
n mines capture resources

m factories need those resources

Each of the n mines must dispatch its production

Our Tool: *Optimal Transport*

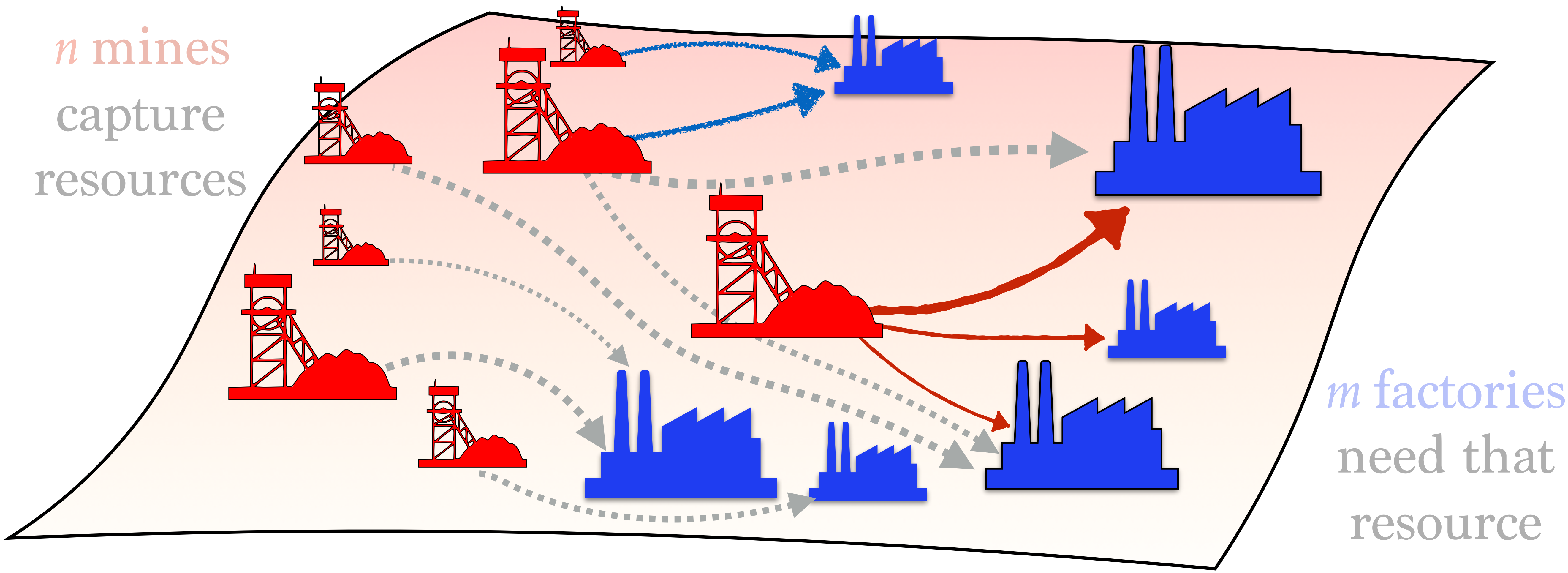
Sketch: mined resources must be dispatched towards factories.



Each of the m factories need resource as raw material

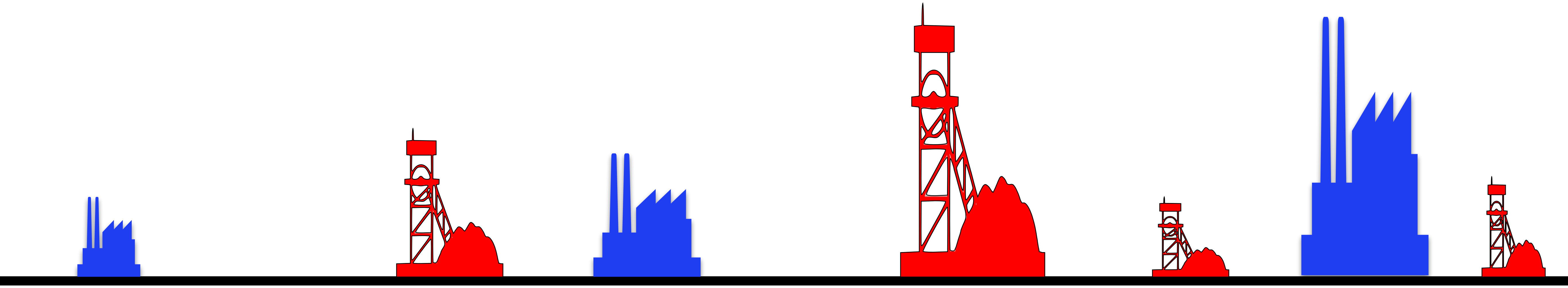
Our Tool: *Optimal Transport*

Optimal Transport computes the **least-costly** transfer plan (in terms of moving resources) in $O((n+m)\,nm \log(n+m))$



Optimal Transport in 1D & sorting

Suppose the mines and factories live in a 1D world.

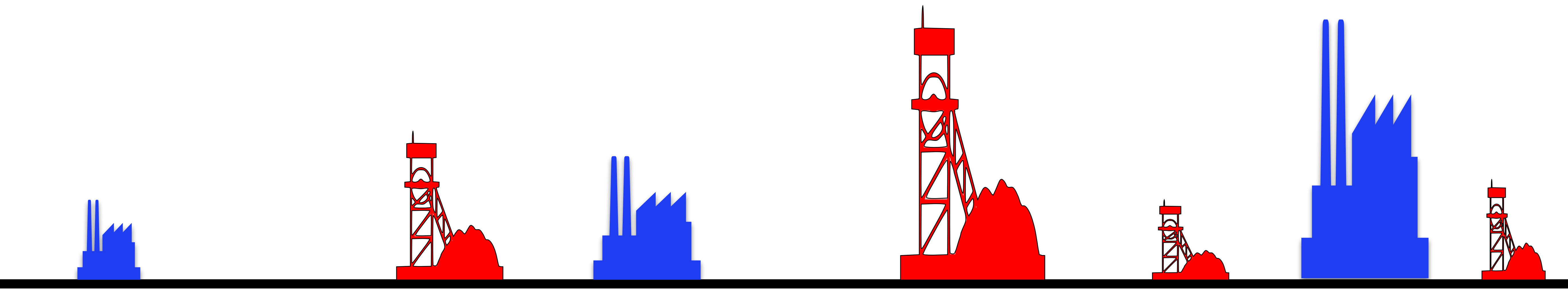


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

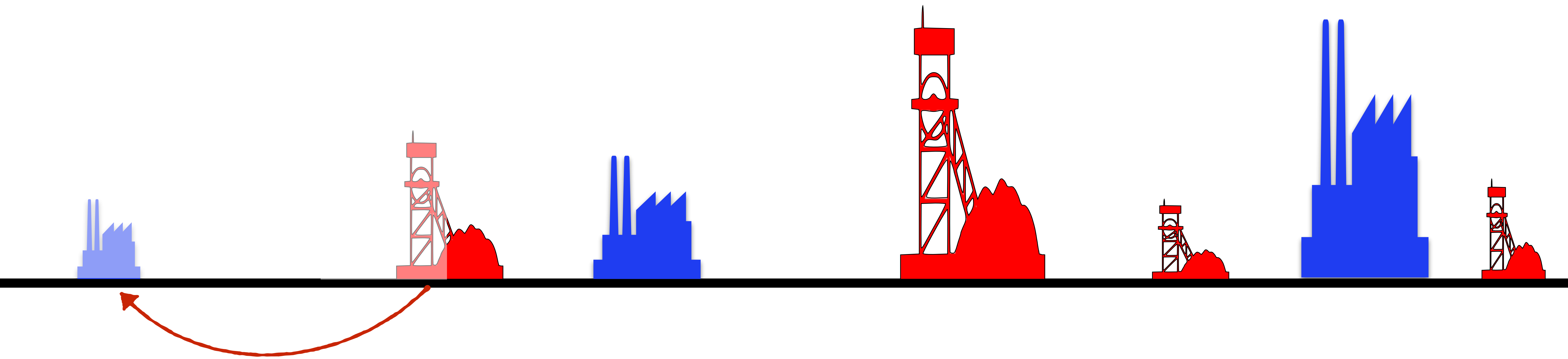


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

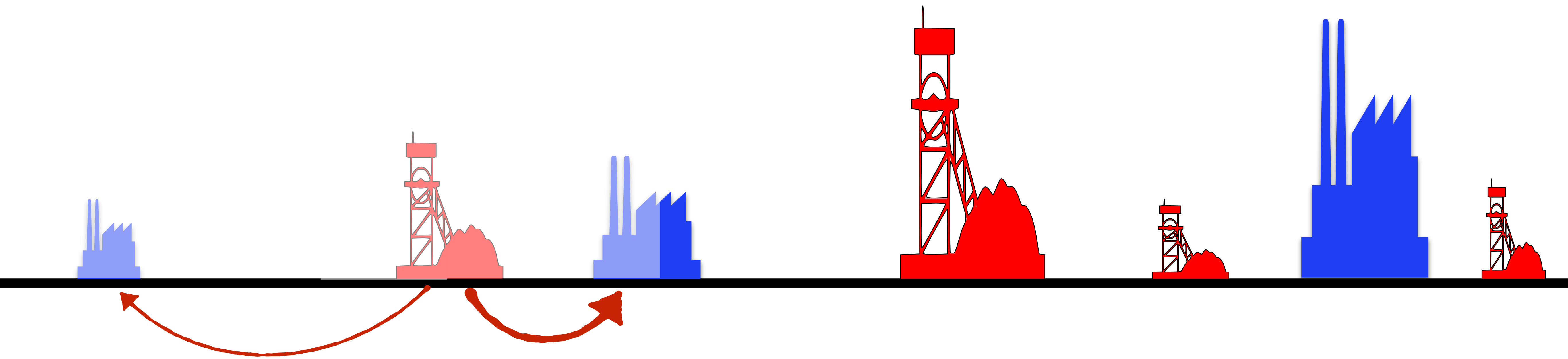


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

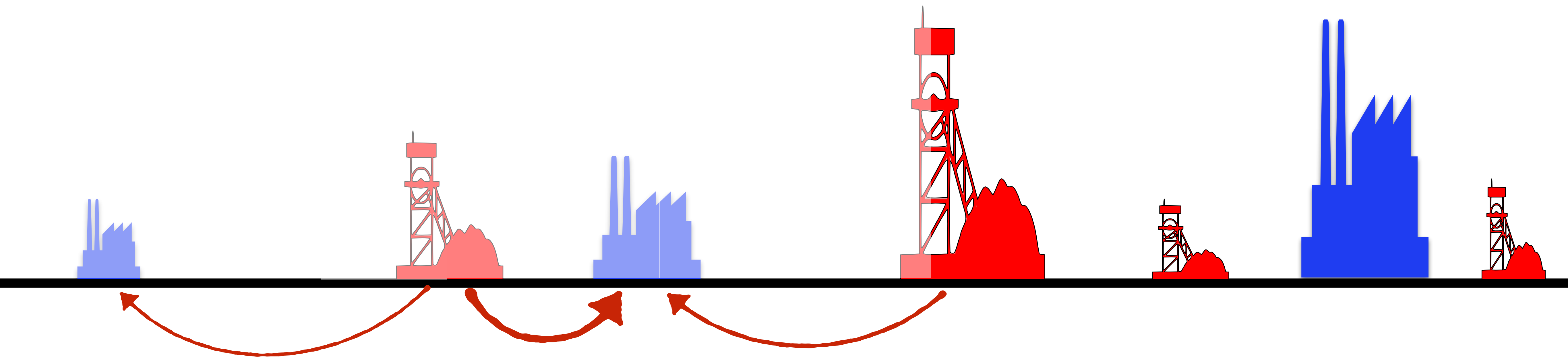


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

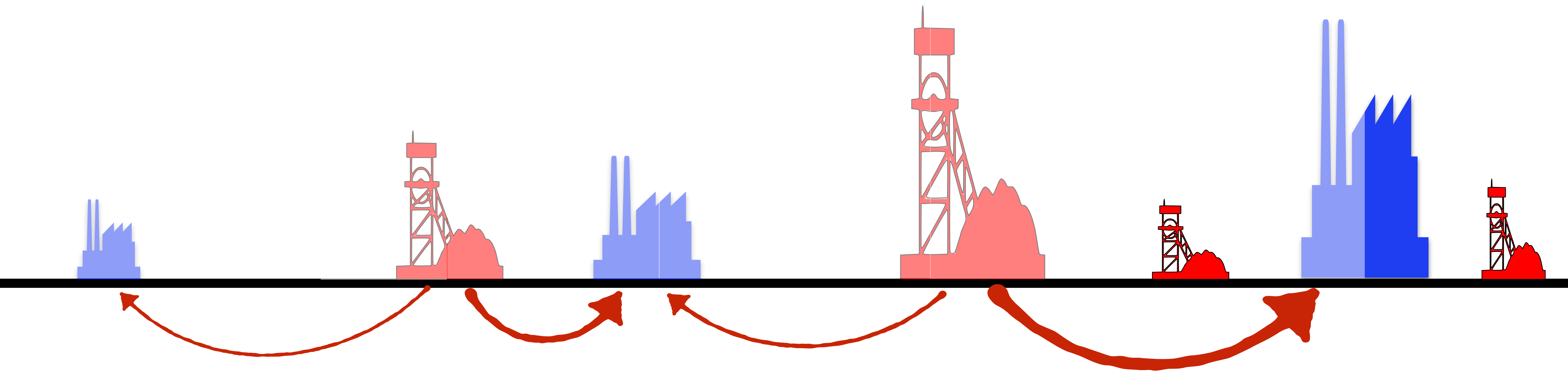


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

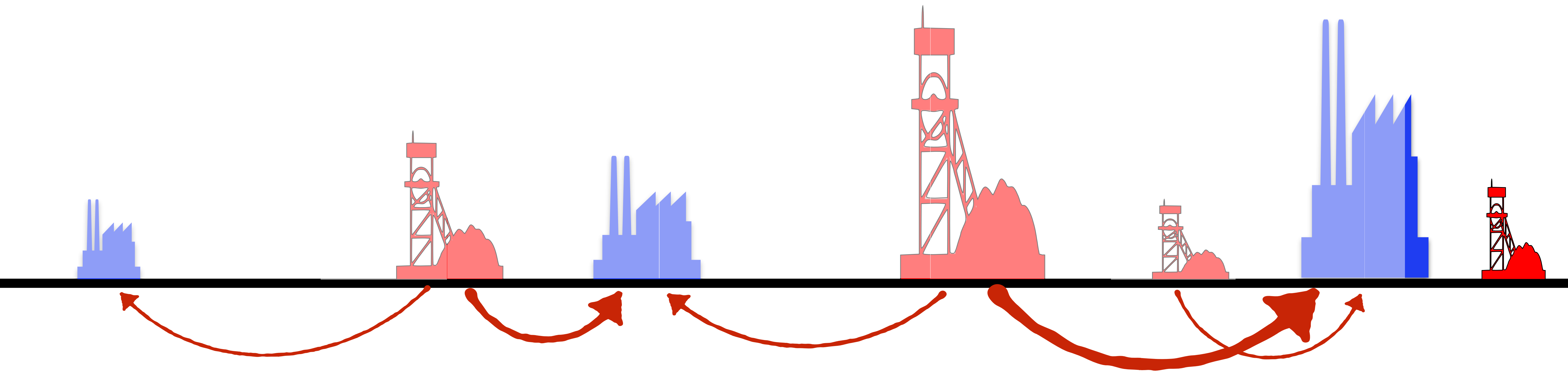


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

“Repeat until termination: the leftmost mine that’s left transfers as much as it can to leftmost factory still in need of resource.”

computation: $O(n \log n + m \log m)$

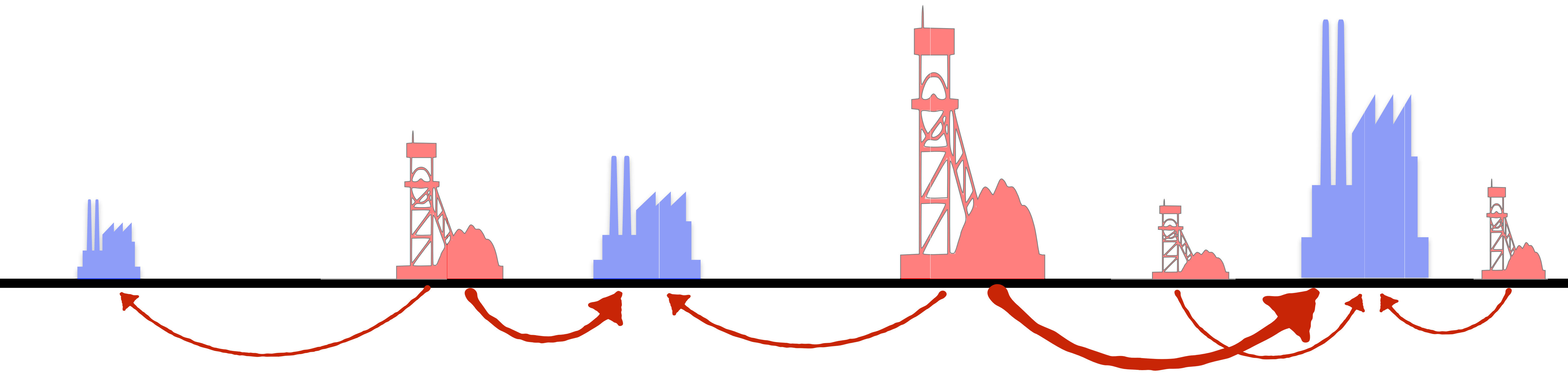


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$

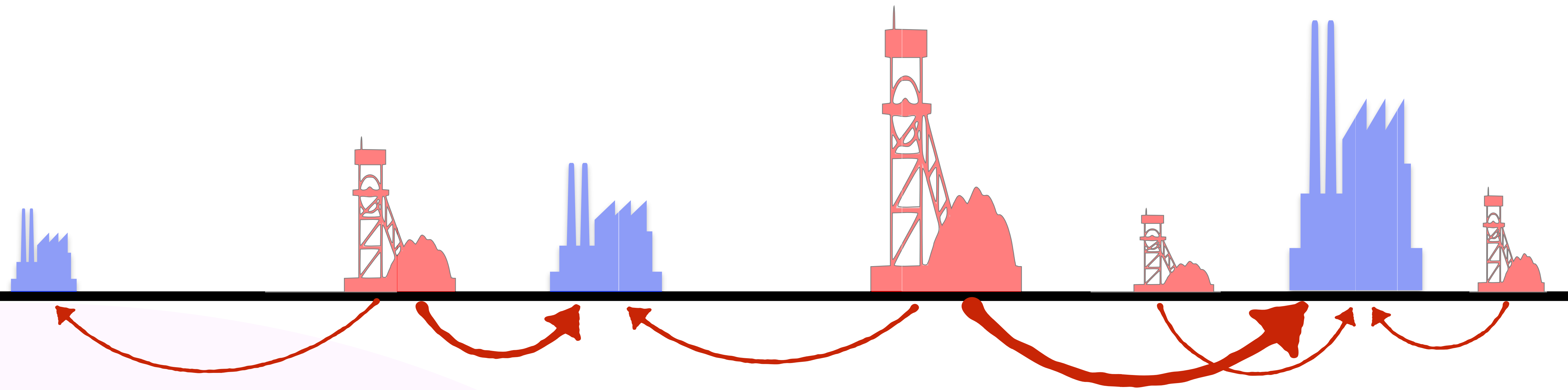


Optimal Transport in 1D & sorting

Optimal Transport *is* **solved** *using* **sorting:**

"Repeat until termination: the leftmost mine that's left transfers as much as it can to leftmost factory still in need of resource."

computation: $O(n \log n + m \log m)$



Ranking / Sorting

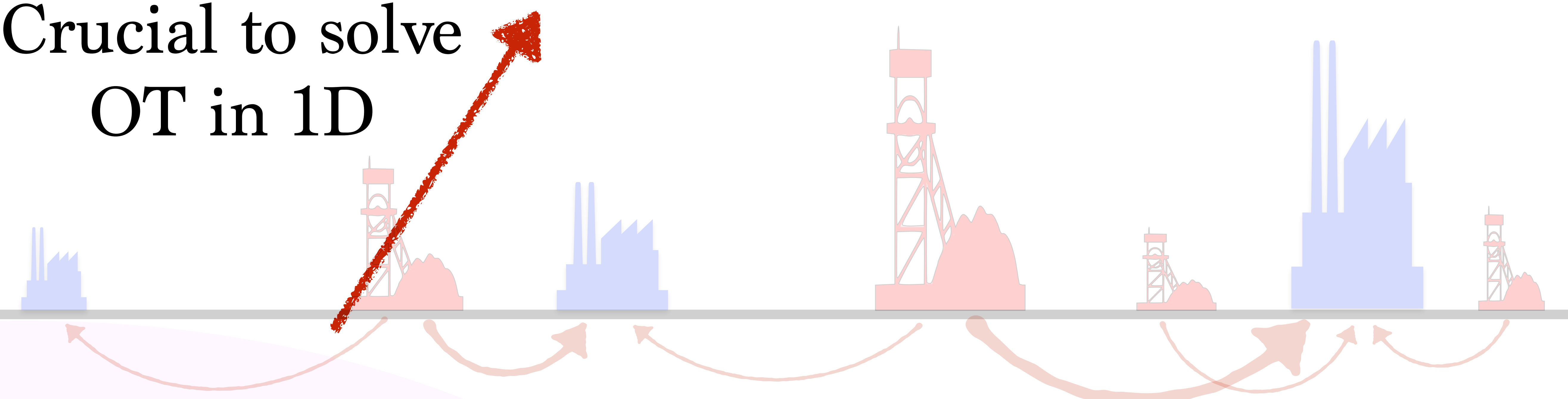
$O(n \log n + m \log m)$

Optimal Transport in 1D & sorting

Optimal Transport

$O(\,(n+m)\,nm\,\log(n+m)\,)$

Crucial to solve OT in 1D



Ranking / Sorting

$O(n \log n + m \log m)$

Our idea: Regularized OT

Optimal Transport

$O(\,(n+m)\,nm\log(n+m)\,)$

❶

generalize sorting using OT (overkill!)

Ranking / Sorting

$O(n\log n + m\log m)$

Our idea: Regularized OT

Optimal Transport

$O(\ (n+m)\ nm \log(n+m)\ )$

❶ *generalize sorting using OT (overkill!)*

Ranking / Sorting

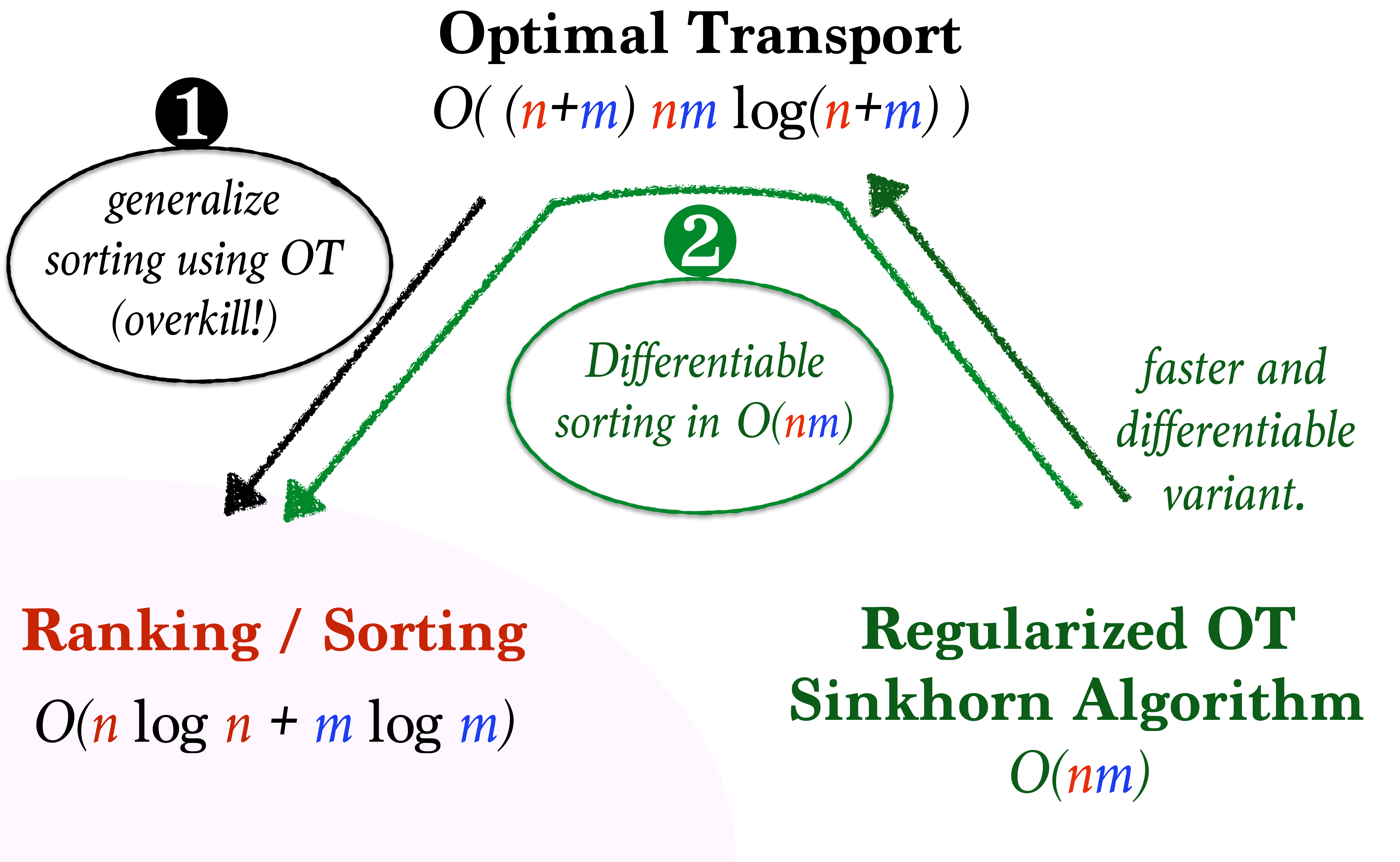
$O(n \log n + m \log m)$

faster and differentiable variant.

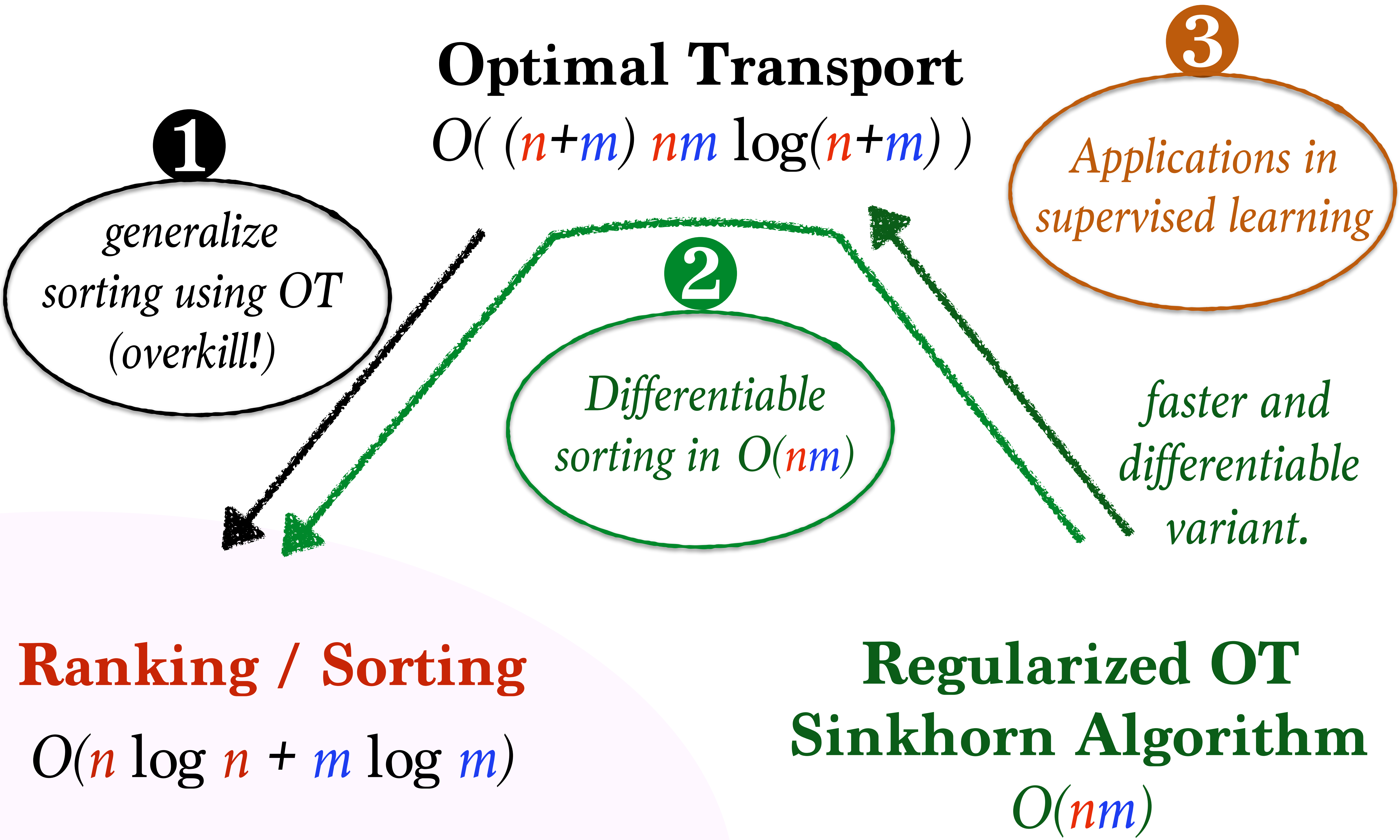
Regularized OT Sinkhorn Algorithm

$O(nm)$

Our idea: Regularized OT



Our idea: Regularized OT



At the poster, you will see...

- Algorithm, parallelism and numerical tradeoffs
  - careful **pre-processing** is needed.
  - **Sinkhorn** needs some stabilization.
- **Applications** involving *softrank* and *softsort*
  - *softrank*: our soft 0/1 loss is competitive w.r.t. XE
  - *softsort*: soft least-quantile regression
- TF and JAX implementations available

https://github.com/google-research/google-research/tree/master/soft_sort

@ East Exhibition Hall B + C **#60**