Fast projection onto the top-k-sum constraint

Jianting Pan, Ming Yan

School of Data Science,
The Chinese University of Hong Kong, Shenzhen

December 9, 2025

Optimization problem

Optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{x}\in\mathbb{R}^n} \ \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{a}\|^2 \\ &\text{s.t. } x \in \mathcal{T}^r_{(k)} := \left\{ \boldsymbol{x}\in\mathbb{R}^n : \mathrm{T}_{(k)}(\boldsymbol{x}) := \sum_{i=1}^{k} \vec{x}_i \le r \right\}. \end{aligned} \tag{1}$$

- $\vec{x}_1 \ge \vec{x}_2 \ge \cdots \ge \vec{x}_n$ are the components of $\boldsymbol{x}$ in nonincreasing order.

Application I (Conditional value-at-risk (CVaR))

- It measures the conditional expectation of losses exceeding a given threshold, **capturing the severity of tail events**.
- Important in portfolio optimization, quantile regression and so on.
- Mathematical

$$\phi_\beta(\boldsymbol{x}) = \inf_{\alpha\in\mathbb{R}} \left\{ \alpha + \frac{1}{(1-\beta)n} \sum_{i=1}^{n} (x_i - \alpha)_+ \right\} \leq \kappa \quad \Rightarrow \quad \boldsymbol{x} \in \mathcal{T}^r_{(k)}.$$

Application II (Ky-Fan k-norm)

- It is defined as the sum of a matrix's k largest singular values.
- Important in matrix optimization, statistics, finance and so on.
- Mathematical

$$\sum_{i=1}^{k} \sigma_i(X) \leq r \quad \Rightarrow \quad X \in \mathcal{T}_{(k)}^{r}.$$

KKT conditions overview (Necessary & Sufficient)

1. **Area Balance:**

$$g(u^*, l^*) = 0, \quad g(u, l) := \sum_{i=1}^{k} \max\{u - \vec{a}_i, 0\} - \sum_{j=k+1}^{n} \max\{\vec{a}_j - l, 0\} \tag{2}$$

2. **Top-k Sum Constraint:**

$$f(u^*, l^*) = 0, \quad f(u, l) := \sum_{i=1}^{n} \max\{a_i - u, 0\} + lk - \min\left\{\sum_{i=1}^{k} \vec{a}_i, r\right\} \tag{3}$$

with $u^* \geq \vec{a}_k \geq l^*$.

Solution Form:

$$\boldsymbol{x}^*_\alpha = \boldsymbol{a}_\alpha - (u^* - l^*)\mathbf{1}_{|\alpha|}, \quad \boldsymbol{x}^*_\beta = l^*\mathbf{1}_{|\beta|}, \quad \boldsymbol{x}^*_\gamma = \boldsymbol{a}_\gamma, \tag{4}$$

$$\alpha := \{i : a_i > u^*\}, \quad \beta := \{i : u^* \geq a_i \geq l^*\}, \quad \gamma := \{i : a_i < l^*\}$$

KKT conditions visualization

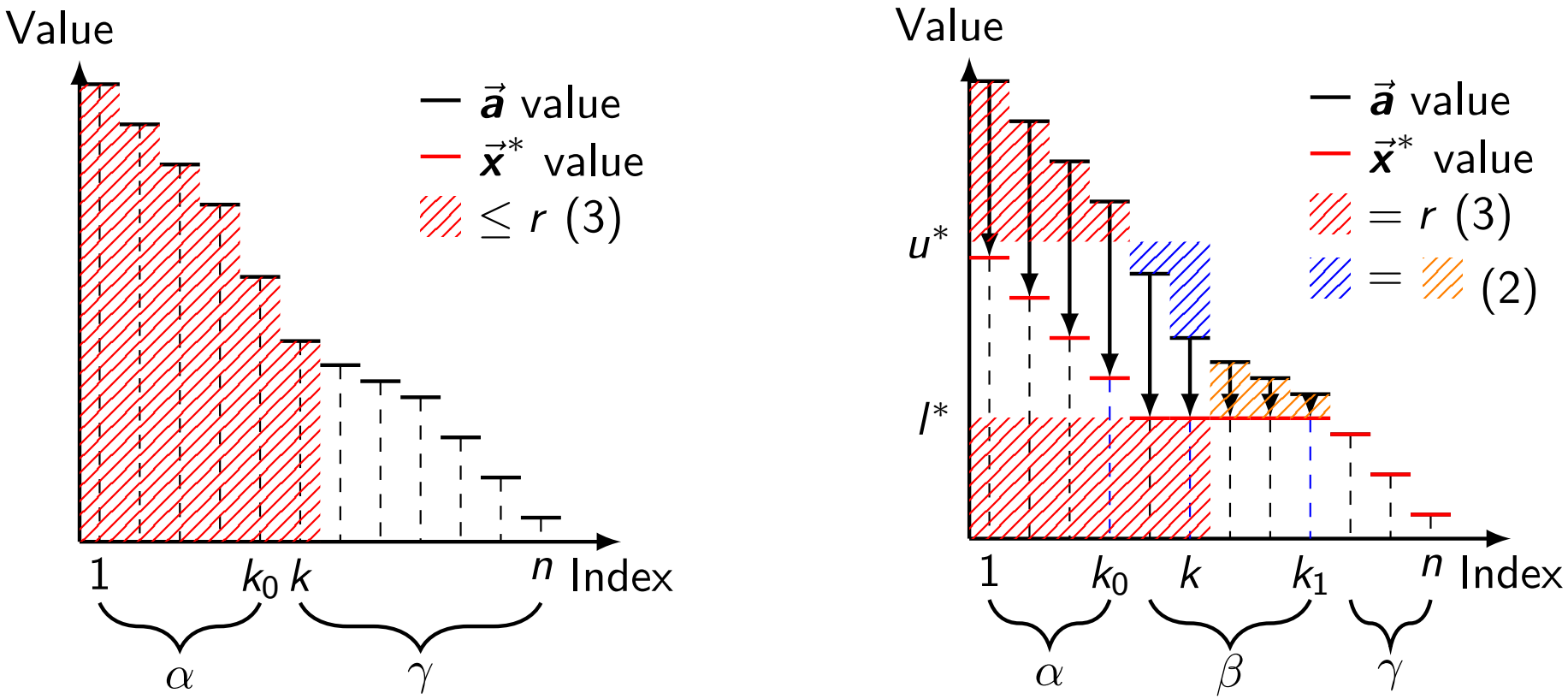


Figure 1: Schematic of sorted input $\vec{a}$ and sorted projection $\vec{x}^*$, where $k_0 = |\alpha|$ and $k_1 = |\alpha| + |\beta|$. **(Left)**: $\mathrm{T}_{(k)}(\boldsymbol{a}) \leq r$. **(Right)**: $\mathrm{T}_{(k)}(\boldsymbol{a}) > r$.

Literature review

Previous methods search (k_0, k_1) until KKT conditions are satisfied.

Algorithm	Complexity	Require sorting
[Wu et al., 2014, GRID]	$O(k(n-k))$	✓
[Roth and Cui, 2025, ESGS]	$O(n)$	✓
[Luxenberg et al., 2025]	$O(n)$	✓

Table 1: Comparison of methods.

Literature review

Previous methods search (k_0, k_1) until KKT conditions are satisfied.

Algorithm	Complexity	Require sorting
[Wu et al., 2014, GRID]	$O(k(n-k))$	✓
[Roth and Cui, 2025, ESGS]	$O(n)$	✓
[Luxenberg et al., 2025]	$O(n)$	✓

Table 1: Comparison of methods.

Computational Challenge (unsorted input)

Solving KKT requires sorting first (expensive $O(n \log n)$ operation)

Literature review

Previous methods search (k_0, k_1) until KKT conditions are satisfied.

Algorithm	Complexity	Require sorting
[Wu et al., 2014, GRID]	$O(k(n-k))$	✓
[Roth and Cui, 2025, ESGS]	$O(n)$	✓
[Luxenberg et al., 2025]	$O(n)$	✓

Table 1: Comparison of methods.

Computational Challenge (unsorted input)

Solving KKT requires sorting first (expensive $O(n \log n)$ operation)

Why is the sorting necessary?

KKT conditions visualization (another perspective)

Define $F(u) = \{l : f(u, l) = 0\}$, $\quad G(u) = \{l : g(u, l) = 0\}$, i.e.

$$F(u) = \frac{1}{k}\left(\min\left\{\sum_{i=1}^{k}\vec{a}_i, r\right\} - \sum_{i=1}^{n}\max\{a_i - u, 0\}\right), \tag{5}$$

$$G(u) = \frac{1}{|\mathcal{J}|}\left(\sum_{j\in\mathcal{J}} a_j - \sum_{i=1}^{k}\max\{u - \vec{a}_i, 0\}\right), \tag{6}$$

where $\mathcal{J} = \{j : \vec{a}_{k+1} \geq \vec{a}_j > G(u)\}$.

KKT conditions visualization (another perspective)

Define $F(u) = \{l : f(u, l) = 0\}$, $\quad G(u) = \{l : g(u, l) = 0\}$, i.e.

$$F(u) = \frac{1}{k}\left(\min\left\{\sum_{i=1}^{k}\vec{a}_i, r\right\} - \sum_{i=1}^{n}\max\{a_i - u, 0\}\right), \tag{5}$$

$$G(u) = \frac{1}{|\mathcal{J}|}\left(\sum_{j\in\mathcal{J}} a_j - \sum_{i=1}^{k}\max\{u - \vec{a}_i, 0\}\right), \tag{6}$$

where $\mathcal{J} = \{j : \vec{a}_{k+1} \geq \vec{a}_j > G(u)\}$.

Properties:

- $F(u)$ is piecewise linear, non-decreasing, concave.
- $G(u)$ is piecewise linear, decreasing.

KKT conditions visualization (another perspective)

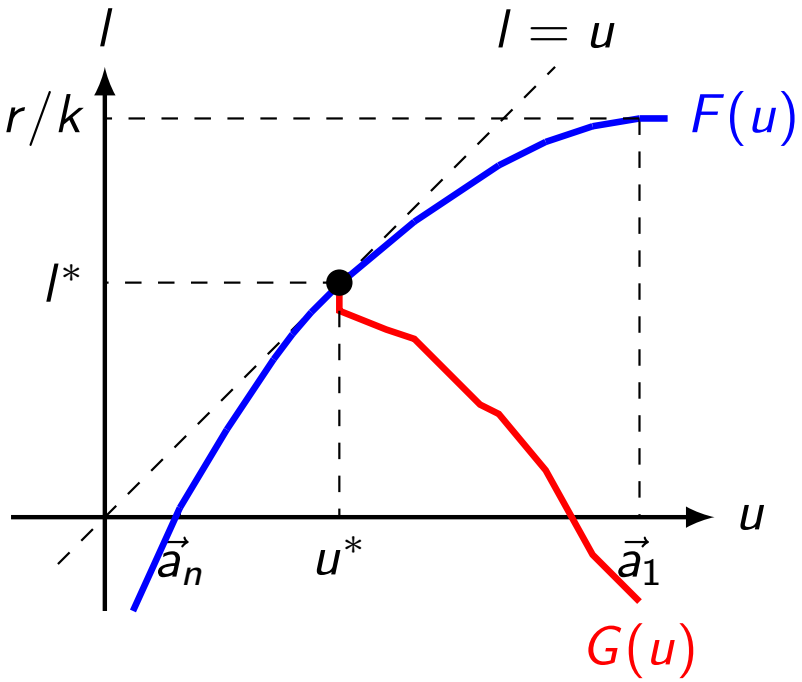


Figure 2: $r \geq \sum_{i=1}^{k} \vec{a}_i$

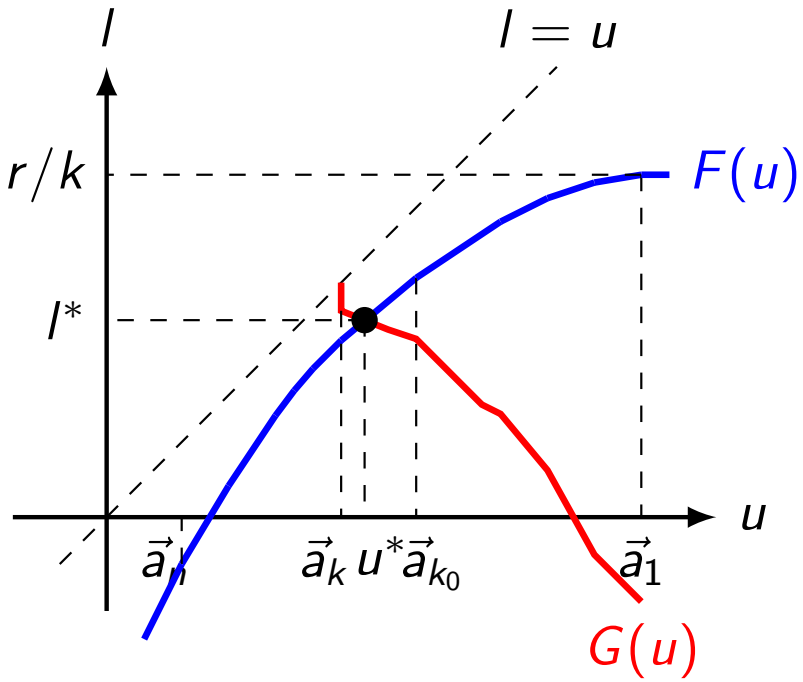


Figure 3: $r < \sum_{i=1}^{k} \vec{a}_i$

KKT conditions visualization (another perspective)

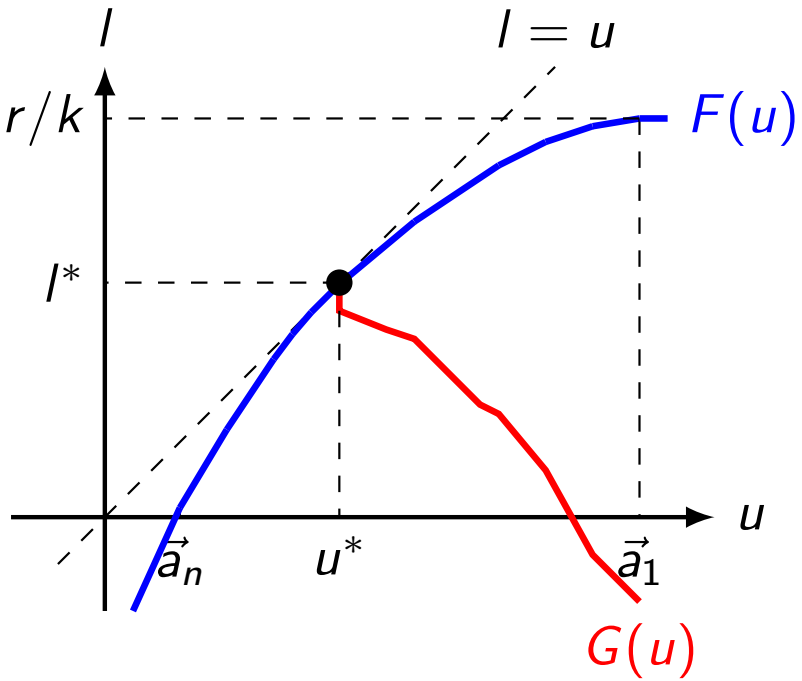


Figure 2: $r \geq \sum_{i=1}^{k} \vec{a}_i$

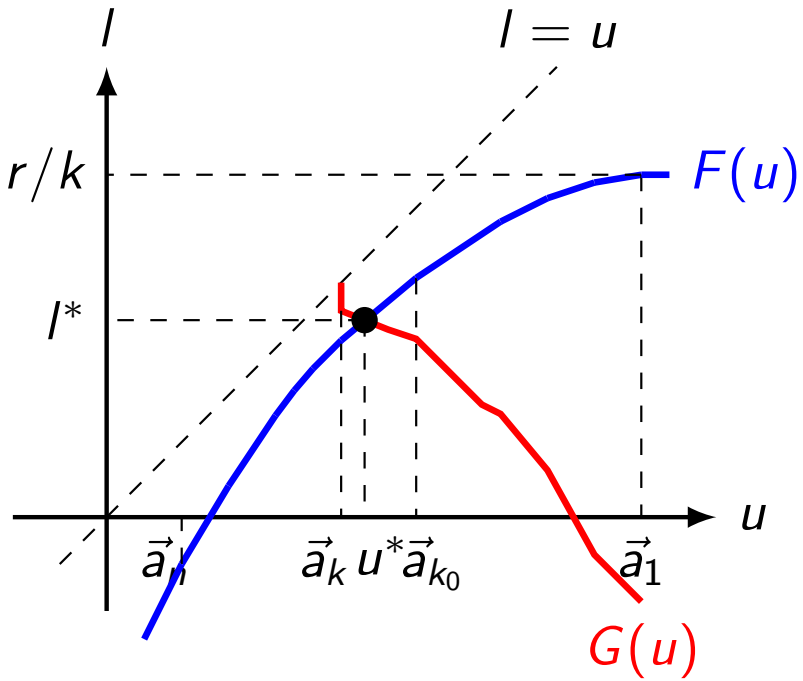


Figure 3: $r < \sum_{i=1}^{k} \vec{a}_i$

Key insights: Searching (k_0, k_1) is equivalent to solving $F(u) = G(u)$.

Relaxed KKT conditions

Goal: Avoid sorting by relaxing conditions.

Relaxed KKT conditions

Goal: Avoid sorting by relaxing conditions.

Modified $F(u)$:

Modified $F(u)$:

$$F_r(u) = \frac{1}{k}\left(\min\left\{\cancel{\sum_{i=1}^{k}\vec{a}_i}, r\right\} - \sum_{i=1}^{n}\max\{a_i - u, 0\}\right)$$

Relaxed KKT conditions

Goal: Avoid sorting by relaxing conditions.

Modified $F(u)$:

Modified $F(u)$:

$$F_r(u) = \frac{1}{k}\left(\min\left\{\cancel{\sum_{i=1}^{k}\vec{a}_i}, r\right\} - \sum_{i=1}^{n}\max\{a_i - u, 0\}\right)$$

Relationship between $F(u)$ and $F_r(u)$:

- $r \le \sum_{i=1}^{k}\vec{a}_i$: $F_r(u) = F(u)$.
- $r > \sum_{i=1}^{k}\vec{a}_i$: $F_r(u) > F(u)$.

Properties (Follow $F(u)$):

- $F_r(u)$ is piecewise linear, non-decreasing, concave.

Relaxed KKT conditions

Modified $G(u)$:

$$G_r(u) = \min\{l : g_r(u, l) = 0\},$$

where

$$g_r(u, l) = \sum_{i=1}^{n} (\max\{a_i - u, 0\} - \max\{a_i - l, 0\}) + k(u - l).$$

Relaxed KKT conditions

Modified $G(u)$:

$$G_r(u) = \min\{l : g_r(u, l) = 0\},$$

where

$$g_r(u, l) = \sum_{i=1}^{n} (\max\{a_i - u, 0\} - \max\{a_i - l, 0\}) + k(u - l).$$

Relationship between $G(u)$ and $G_r(u)$:

- $g(u, l) = g_r(u, l)$ for $(u, l) \in [\vec{a}_{k+1}, \infty) \times (-\infty, \vec{a}_k]$.
- $G_r(u) = G(u)$ for $u > \vec{a}_k$;
- $G_r(u) = \vec{a}_{k+1}$ for $u \in [\vec{a}_{k+1}, \vec{a}_k]$, while $G(u) = [\vec{a}_{k+1}, \vec{a}_k]$ for $u = \vec{a}_k$.

$$u^* \geq \vec{a}_k \rightarrow \textbf{Relaxed: } u > \vec{a}_{k+1}.$$

Properties (Follow $G(u)$):

- $G_r(u)$ is piecewise linear, decreasing on $u \in [\vec{a}_k, \infty)$.

Solving relaxed KKT

Intersection Point u_r^*: Solve $F_r(u) = G_r(u)$

Cases:

1. **No Intersection:** $r \geq \mathrm{T}_{(k)}(\boldsymbol{a})$, solution is $\boldsymbol{x}^* = \boldsymbol{a}$
2. **Intersection with $u_r^* > \vec{a}_k$:** $r < \mathrm{T}_{(k)}(\boldsymbol{a})$, $u^* = u_r^*$ and $l^* = F(u^*)$
3. **Intersection with $\vec{a}_{k+1} < u_r^* \leq \vec{a}_k$:** $r < \mathrm{T}_{(k)}(\boldsymbol{a})$, $u^* = \vec{a}_k$ and $l^* = F(u^*)$

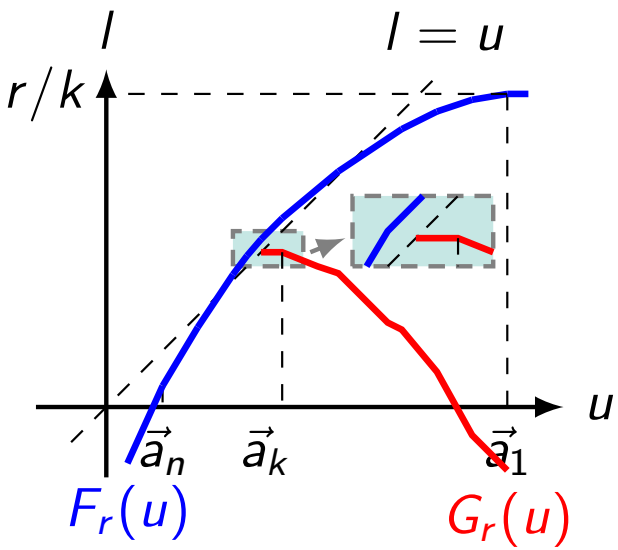


Figure 4: Case 1

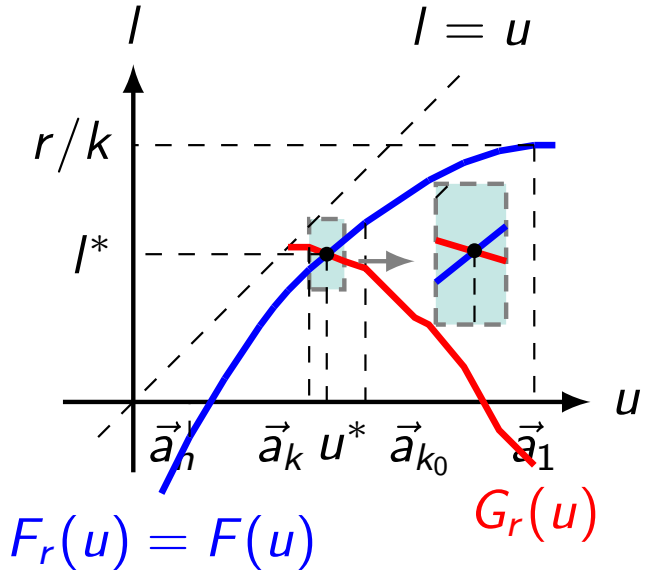


Figure 5: Case 2

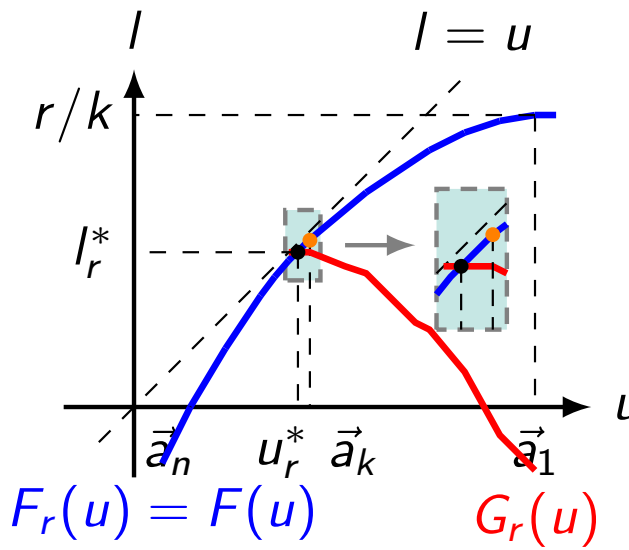


Figure 6: Case 3

Advantages of relaxation

- **No Sorting:** Reduces practical complexity from $O(n \log n)$ to $O(n)$ (numerical results)
- **Same Solution:** Relaxed conditions recover original KKT solution
- **Scalability:** Efficient for large-scale problems

Advantages of relaxation

- **No Sorting:** Reduces practical complexity from $O(n \log n)$ to $O(n)$ (numerical results)
- **Same Solution:** Relaxed conditions recover original KKT solution
- **Scalability:** Efficient for large-scale problems

How to find the intersection point u_r^*?

Calculating $G_r(u)$

Key idea: Construct a lower bound and increase the bound.

Key inequality:

$$\rho = \frac{\sum_{a\in \boldsymbol{v}} a - (k-m)u}{|\boldsymbol{v}| - k + m} \leq G_r(u) = \frac{\sum_{a\in \boldsymbol{b}_T} a - (k-m)u}{|\boldsymbol{b}_T| - k + m}, \tag{7}$$

where $\boldsymbol{b}_T := \{a_i : i \in \mathcal{I}(\boldsymbol{a}, G_r(u), u)\}$, $\boldsymbol{v} \subseteq \boldsymbol{b} := \{a_i : a_i < u\}$ with $|\boldsymbol{v}| > k - m$ and $m = |\{i : a_i \geq u\}|$.

Calculating $G_r(u)$

Key idea: Construct a lower bound and increase the bound.

Key inequality:

$$\rho = \frac{\sum_{a\in \boldsymbol{v}} a - (k-m)u}{|\boldsymbol{v}| - k + m} \leq G_r(u) = \frac{\sum_{a\in \boldsymbol{b}_T} a - (k-m)u}{|\boldsymbol{b}_T| - k + m}, \tag{7}$$

where $\boldsymbol{b}_T := \{a_i : i \in \mathcal{I}(\boldsymbol{a}, G_r(u), u)\}$, $\boldsymbol{v} \subseteq \boldsymbol{b} := \{a_i : a_i < u\}$ with $|\boldsymbol{v}| > k - m$ and $m = |\{i : a_i \geq u\}|$.

- Add b_n to $\boldsymbol{v}$: $G_r(u) \geq \rho \hookleftarrow \rho + (b_n - \rho) / (|\boldsymbol{v}| - (k-m))$
  - **Key insights**: $b_n > \rho \rightarrow$ add b_n to $\boldsymbol{v} \rightarrow \rho$ increases.
- Remove b_n from $\boldsymbol{v}$: $G_r(u) \geq \rho \hookleftarrow \rho + (\rho - b_n)/(|\boldsymbol{v}| - (k-m)$.
  - **Key insights**: $b_n \leq \rho \rightarrow$ remove b_n from $\boldsymbol{v} \rightarrow \rho$ increases.
- ρ **converges to** $G_r(u)$ **after several iterations.**

Calculating $G_r(u)$

Algorithm 1: G-Searching

Data: u; $m = \left|\mathcal{I}(\boldsymbol{a}, u, +)\right|$; $H = k - m$; $\boldsymbol{b} = \{a_i : i \in \mathcal{I}(\boldsymbol{a}, -, u)\}$
Result: ρ

1 **if** $H = 0$ **then return** $\rho = \max\{a_j : a_j < u\}$;
2 Initialize $\boldsymbol{v} \leftarrow \{b_1, b_2, \ldots, b_N\}$ with $N > H$ (e.g., $N = H + 1$);
3 $\rho \leftarrow (\sum_{b \in \boldsymbol{v}} b - Hu)/(|\boldsymbol{v}| - H)$;
4 **for** $n = N + 1, ..., |\boldsymbol{b}|$ **do**
5 | **if** $b_n > \rho$ **then** append b_n to $\boldsymbol{v}$ and set $\rho \leftarrow \rho + (b_n - \rho) \,/\, (|\boldsymbol{v}| - H)$;
6 **end**
7 **while** $|\boldsymbol{v}|$ *changes* **do**
8 | **foreach** $b \in \boldsymbol{v}$ **do**
9 | | **if** $b \leq \rho$ **then** remove b from $\boldsymbol{v}$ and set $\rho \leftarrow \rho + (\rho - b)/(|\boldsymbol{v}| - H)$;
10 | **end**
11 **end**

Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Initialization

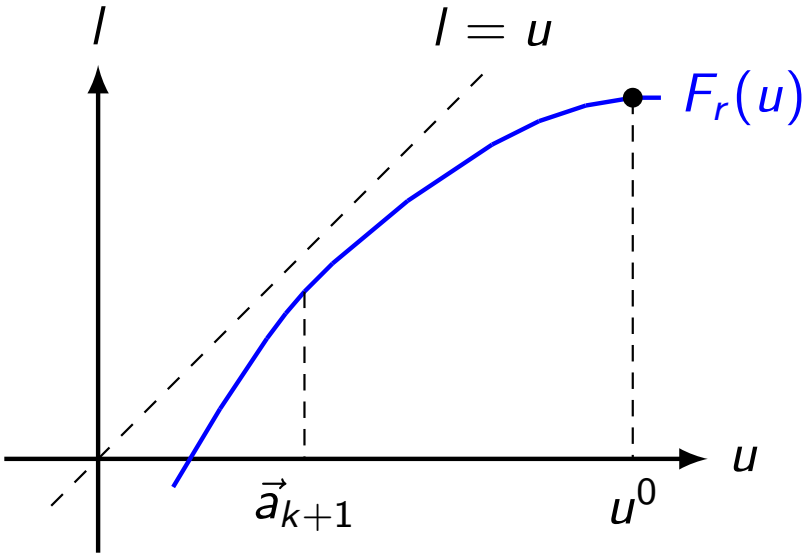
Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.

Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

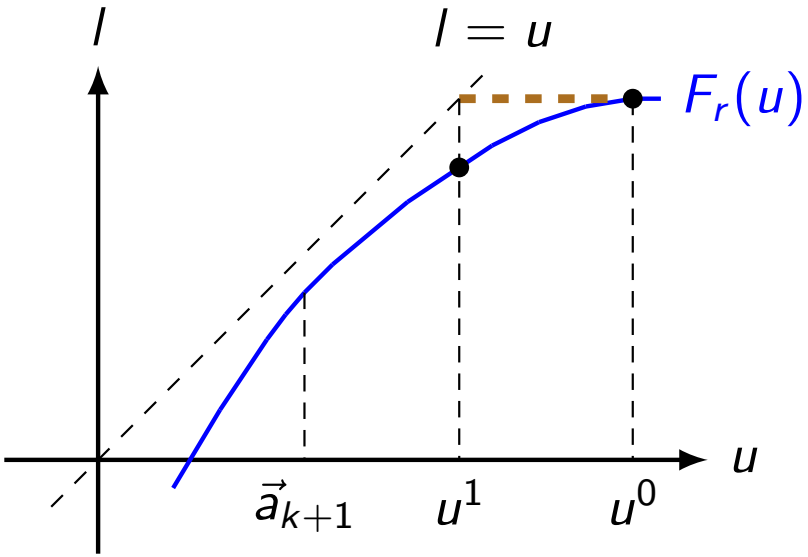
Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.



Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

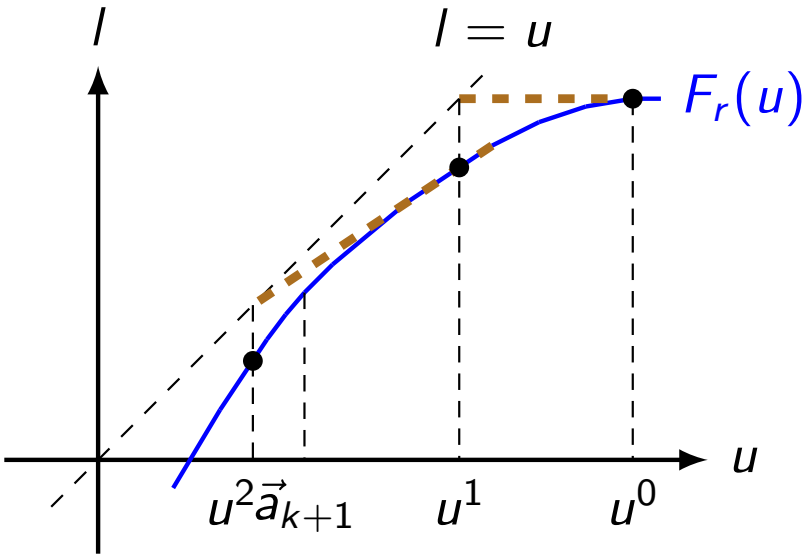
Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.



Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.



Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.

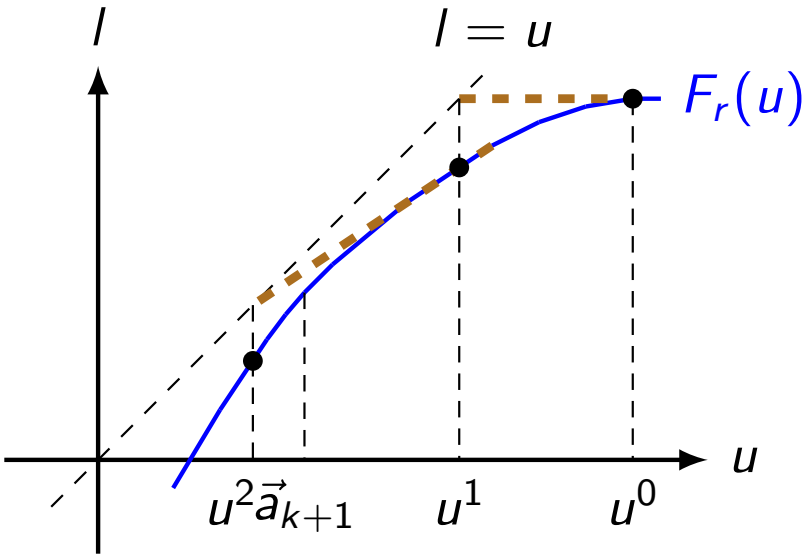


Figure 7: $r < \sum_{i=1}^{k} \vec{a}_i$

Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.

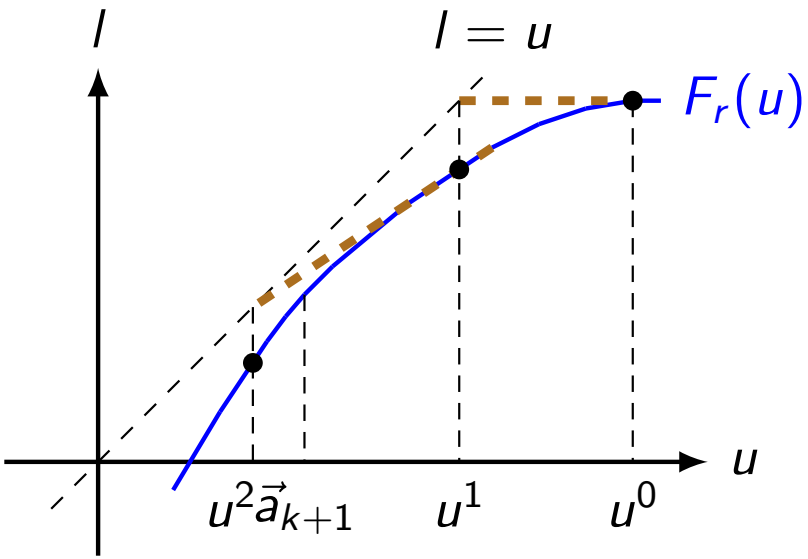


Figure 7: $r < \sum_{i=1}^{k} \vec{a}_i$

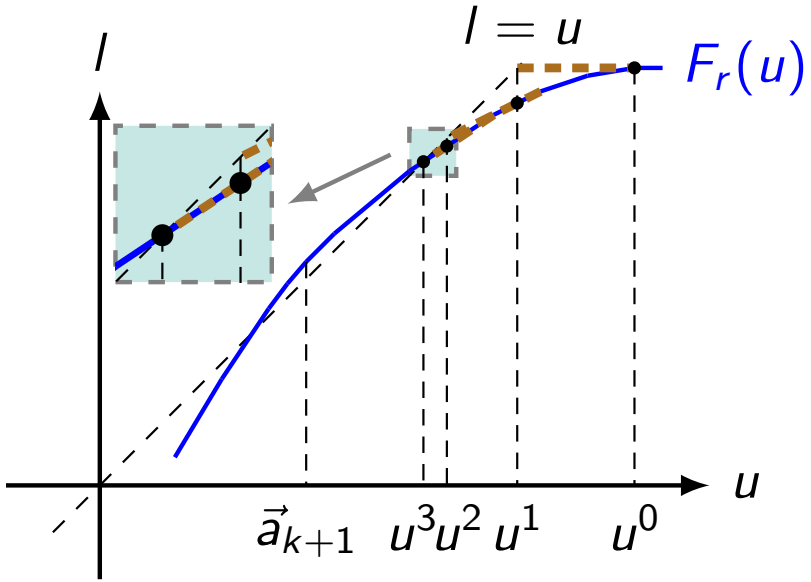


Figure 8: $r \geq \sum_{i=1}^{k} \vec{a}_i$

Initialization

Goal: determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.

Key property: $r < \sum_{i=1}^{k} \vec{a}_i \iff F_r(u)$ has no intersection with $l = u$.

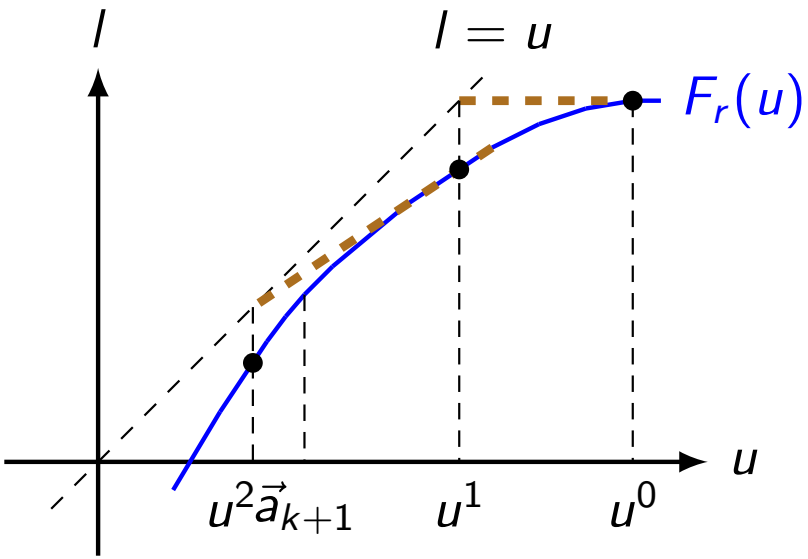


Figure 7: $r < \sum_{i=1}^{k} \vec{a}_i$

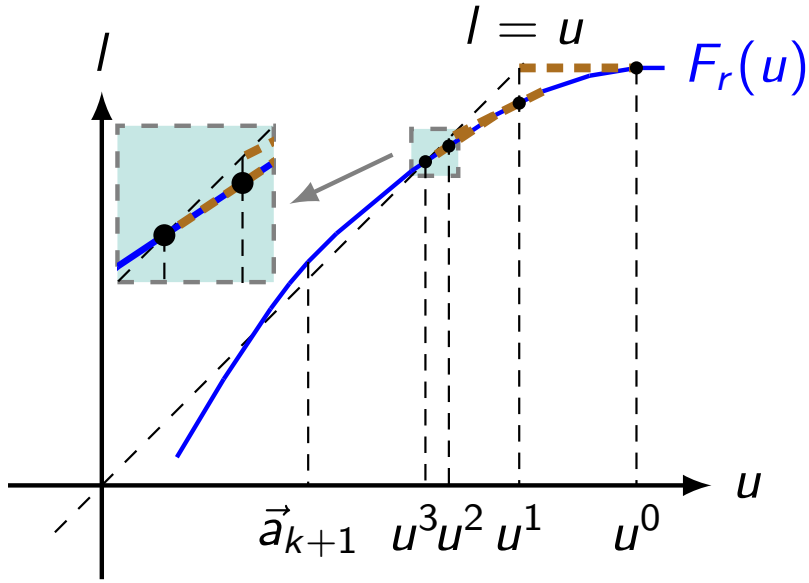


Figure 8: $r \geq \sum_{i=1}^{k} \vec{a}_i$

Focus on $r < \sum_{i=1}^{k} \vec{a}_i$.

Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \Longleftrightarrow u > u_r^*$.

Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \Longleftrightarrow u > u_r^*$.

Idea: Two Pointers: u_L (Left), u_R (Right) approach u_r^*.

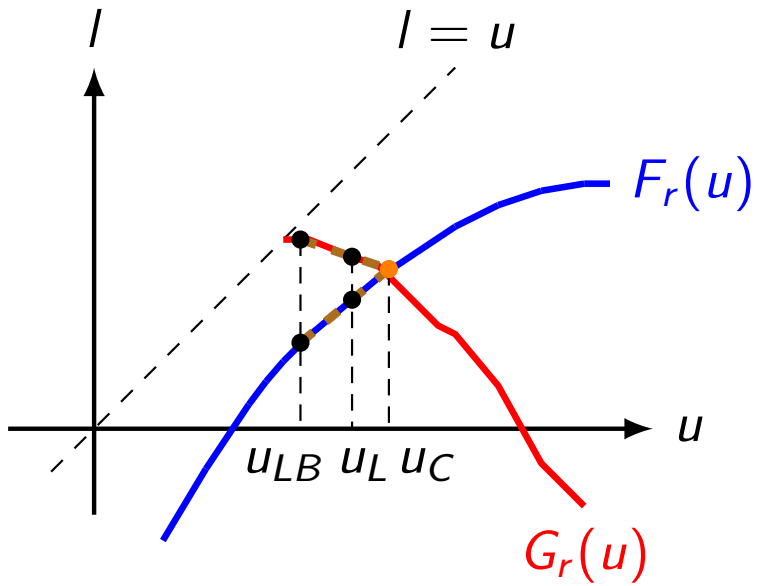
Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \Longleftrightarrow u > u_r^*$.

Idea: Two Pointers: u_L (Left), u_R (Right) approach u_r^*.

- **Case**: $u_R = \infty$
- **Updating rules**:
  - If $D(u_C) < 0 \rightarrow u_C > u_r^* \rightarrow u_R \mapsto u_C$.
  - Otherwise, $u_{LB} \mapsto u_L, u_L \mapsto u_C$
  - Generate u_C



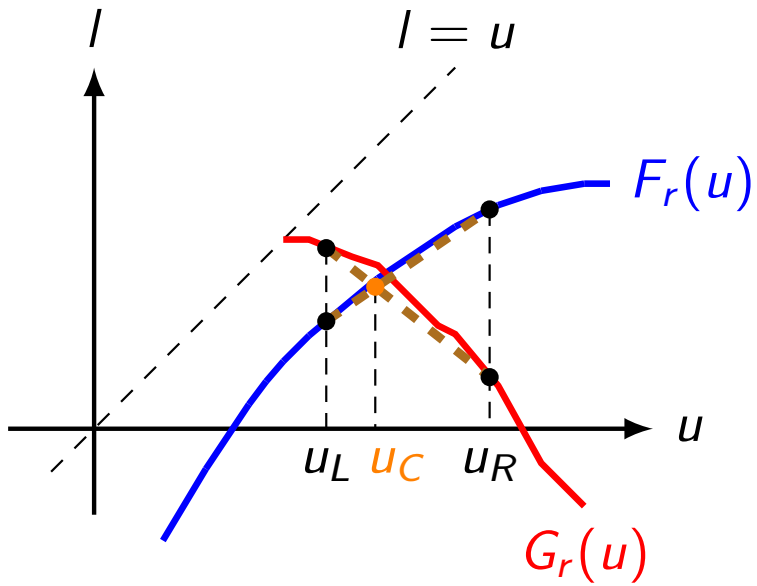
Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \iff u > u_r^*$.

Idea: Two Pointers: u_L (Left), u_R (Right) approach u_r^*.

- **Case**: $u_L > -\infty, u_R < \infty$
- **Updating rules**:
  - Update u_R or u_L based on $D(u_C)$.



Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \Longleftrightarrow u > u_r^*$.

Idea: Two Pointers: u_L (Left), u_R (Right) approach u_r^*.

Stopping criterion:

- u_r^* is found ($|D(u_C)| \leq \epsilon$)
- A narrow interval containing u_r^* is found. That is, $\vec{a}_{j+1} \leq u_L \leq u_r^* \leq \vec{a}_j$.

Pivot

Goal: find u_r^* or a narrow region containing u_r^*.

Fact: $D(u) := G_r(u) - F_r(u) > 0 \Longleftrightarrow u > u_r^*$.

Idea: Two Pointers: u_L (Left), u_R (Right) approach u_r^*.

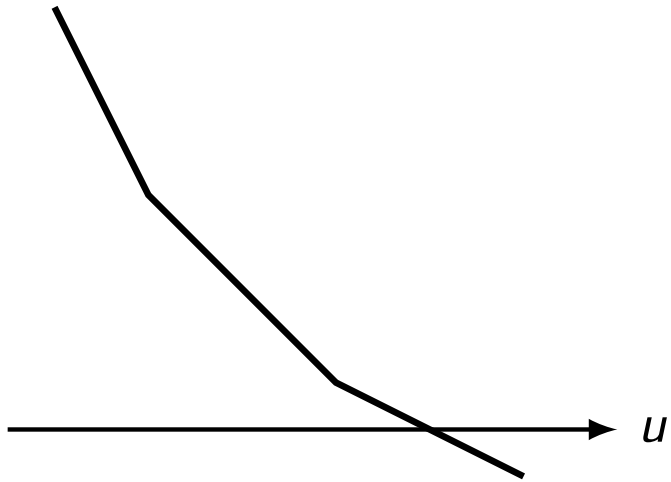
Stopping criterion:

- u_r^* is found ($|D(u_C)| \leq \epsilon$)
- A narrow interval containing u_r^* is found. That is, $\vec{a}_{j+1} \leq u_L \leq u_r^* \leq \vec{a}_j$.

Why do we require such a narrow region?

Exact

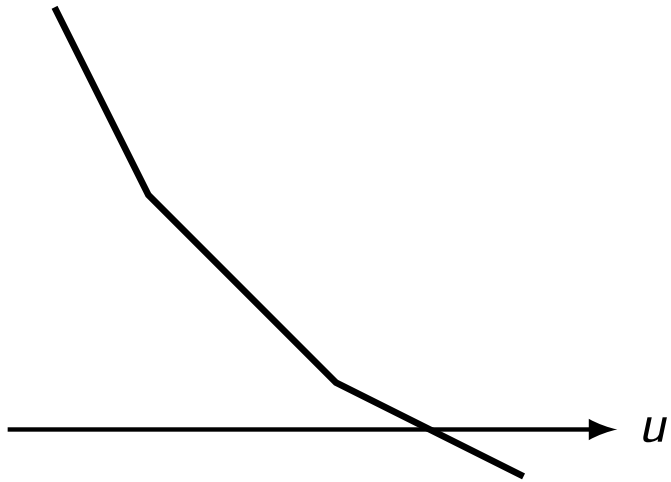
Fact: $D(u) := G_r(u) - F_r(u)$ is convex and piecewise linear on this interval.



Exact

Fact: $D(u) := G_r(u) - F_r(u)$ is convex and piecewise linear on this interval.

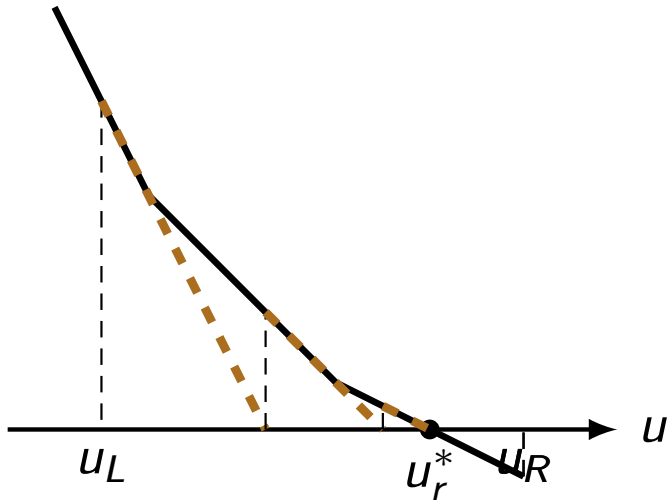
- Newton methods.



Exact

Fact: $D(u) := G_r(u) - F_r(u)$ is convex and piecewise linear on this interval.

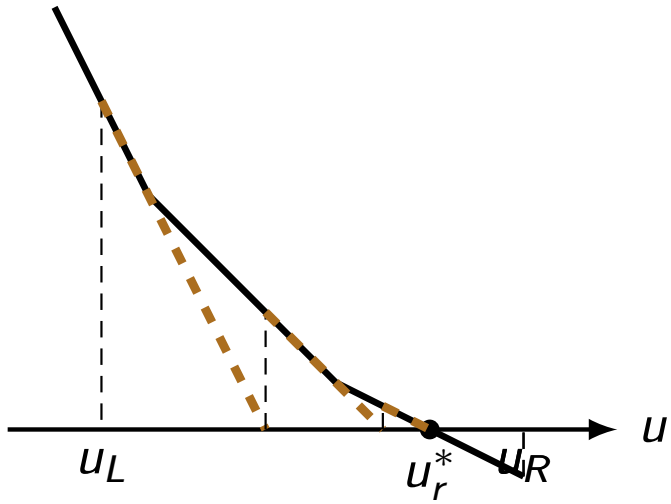
- Newton methods.
- Start from u_L.



Exact

Fact: $D(u) := G_r(u) - F_r(u)$ is convex and piecewise linear on this interval.

- Newton methods.
- Start from u_L.
- Recover $\boldsymbol{x}^*$.



Summary

1. **Initialization step**:
   - Determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.
2. **Pivot step**:
   - Roughly search u_r^*.
3. **Exact step**:
   - Exactly search u_r^* and recover $\boldsymbol{x}^*$.

Summary

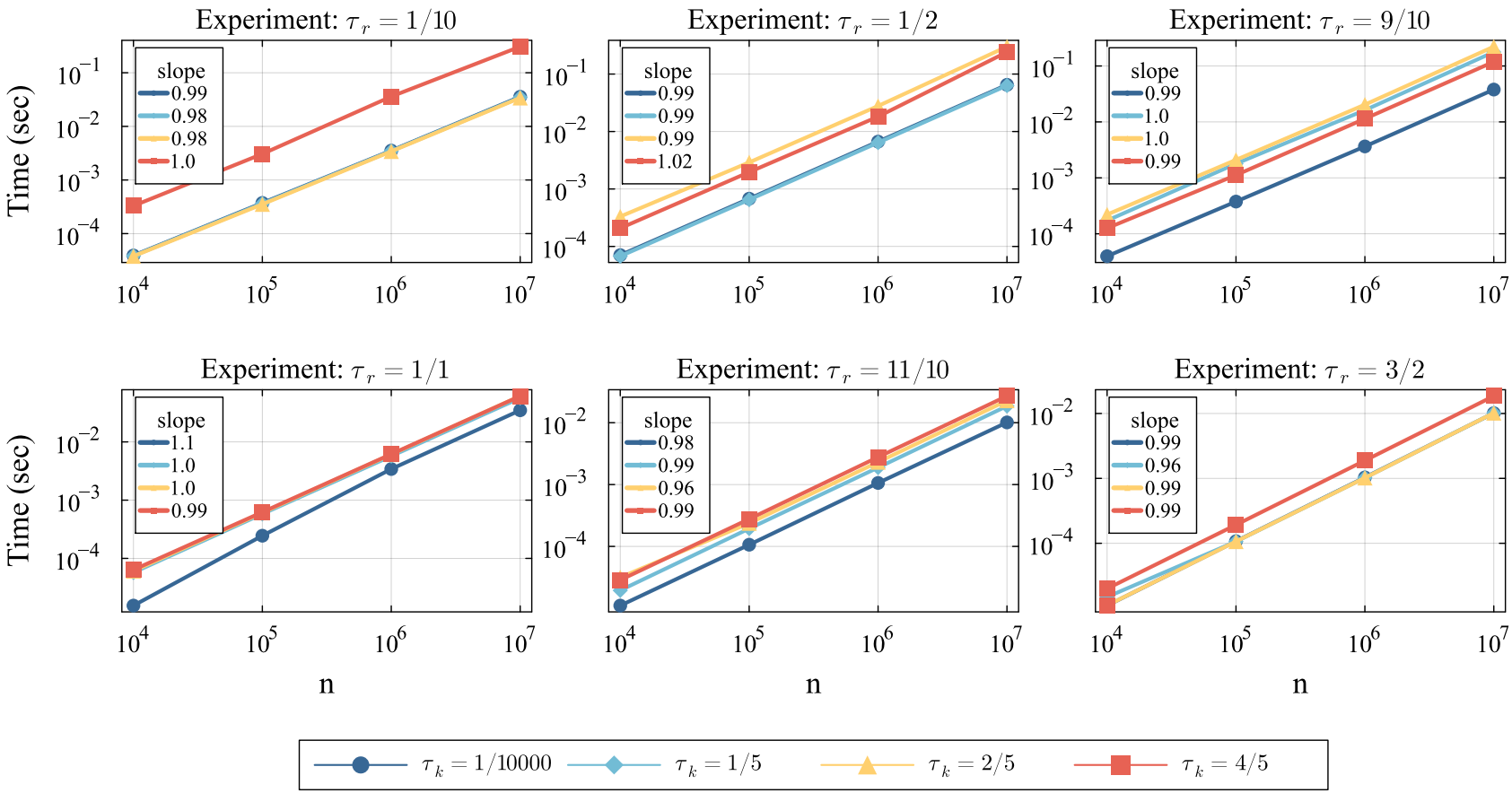
1. **Initialization step**:
   - Determine whether $r \geq \sum_{i=1}^{k} \vec{a}_i$.
2. **Pivot step**:
   - Roughly search u_r^*.
3. **Exact step**:
   - Exactly search u_r^* and recover $\boldsymbol{x}^*$.

	Per step Complexity	Iteration	Overall
Initialization	$O(n)$	k	$O((n+k)n^2)$
Pivot + Exact	$O(n^2)$	$n+k$	

Table 2: Complexity analysis

Practical complexity

Goal: Evaluate the practical complexity of EIPS.



Here, $r = \tau_r \cdot \mathrm{T}_{(k)}(\boldsymbol{a}), \quad k = \tau_k \cdot n.$

Time comparing

Goal: Compare the performance of EIPS against several baseline algorithms: ESGS, PCLP, GRID, and GURO.

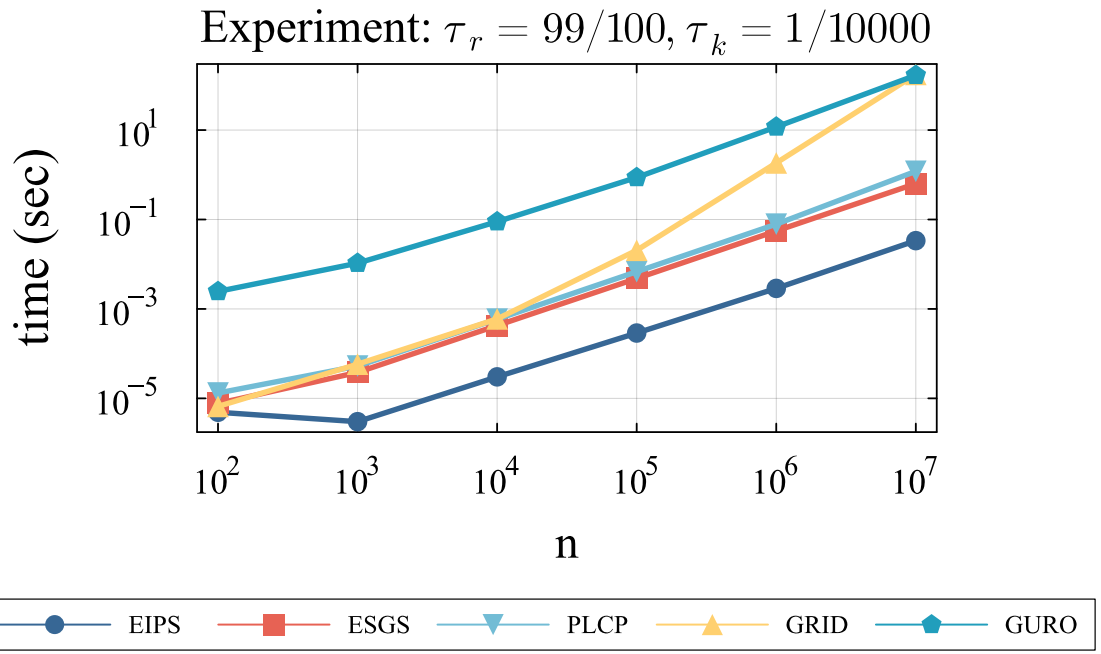
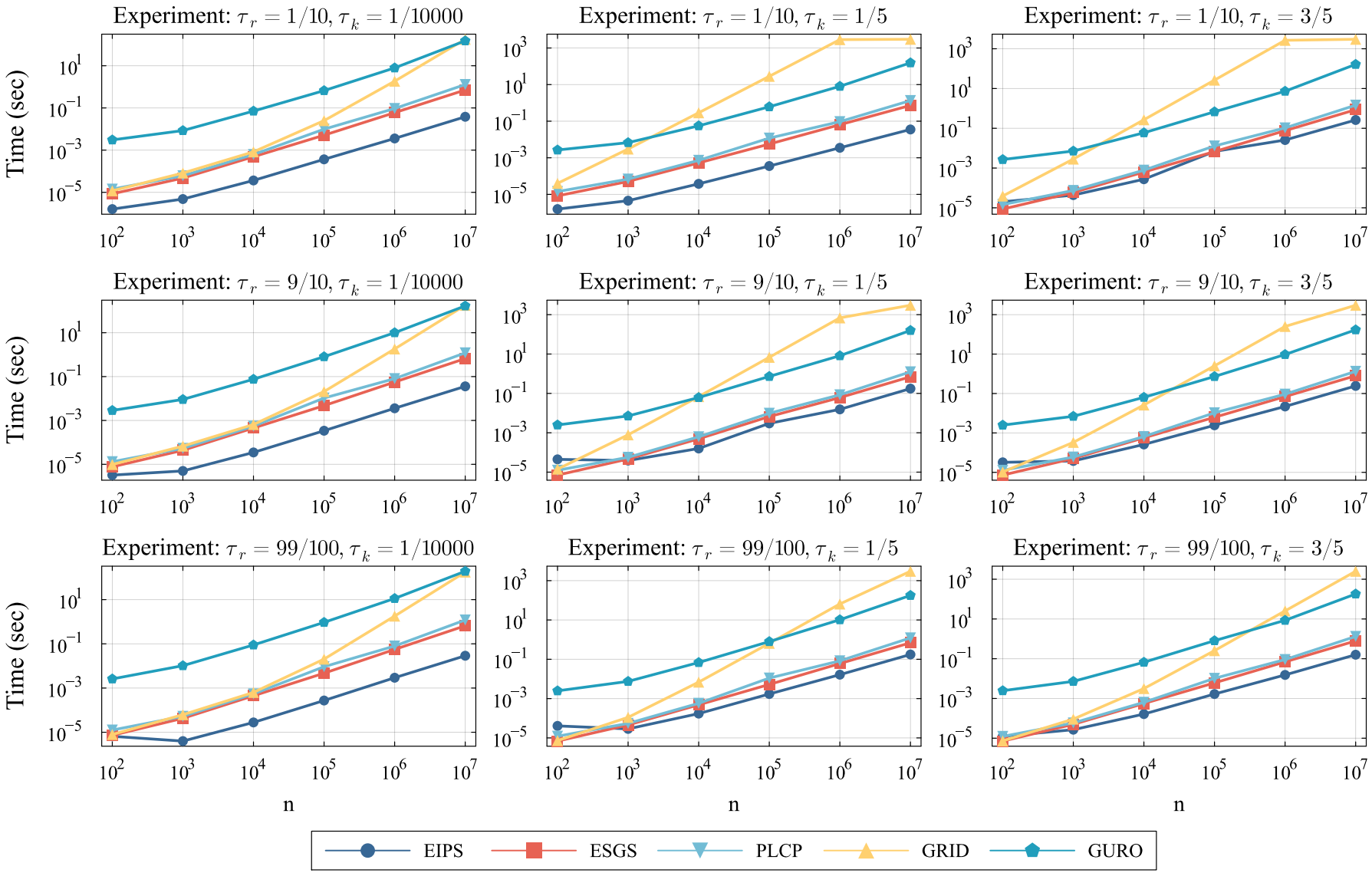


Figure 9: Mean computation time under $r < \sum_{i=1}^{k} \vec{a}_i$ averaged over 100 instances, where $k = \tau_k \cdot n,\ r = \tau_r \cdot \sum_{i=1}^{k} \vec{a}_i$.

Time comparing



Conclusion

- Consider the Euclidean projection onto the **top-k-sum** constraint.
- Introduce **relaxed KKT conditions** and provide a geometric interpretation.
- Develop an iterative approach, EIPS, that **eliminates the need for sorting** and is confirmed to converge globally to the exact solution in a finite number of iterations.
- Numerical experiments show superior computational efficiency over baseline algorithms, especially as the size n increases, with **empirical $O(n)$ complexity**.
- Future work involves applying EIPS to large-scale optimization problems like Conditional Value at Risk (CVaR).

Thank you! Questions?

References I

[Luxenberg et al., 2025] Luxenberg, E., Pérez-Piñeiro, D., Diamond, S., and Boyd, S. (2025).
An operator splitting method for large-scale cvar-constrained quadratic programs.
arXiv preprint arXiv:2504.10814.

[Roth and Cui, 2025] Roth, J. and Cui, Y. (2025).
On $O(n)$ algorithms for projection onto the top-k-sum sublevel set.
Mathematical Programming Computation, pages 1–42.

[Wu et al., 2014] Wu, B., Ding, C., Sun, D., and Toh, K.-C. (2014).
On the Moreau-Yoshida regularization of the vector k-norm related functions.
SIAM Journal on Optimization, 24(2):766–794.