From GPUs to RRAMs: Distributed In-Memory Primal–Dual Hybrid Gradient Method for Solving Large-Scale Linear Optimization Problems

Huynh Q. N. Vo[1,2] M. T. R. Chowdhury[3,2] Paritosh Ramanan[1] Gözde Tutuncuoglu[3] Junchi Yang[2] Feng Qiu[2] Murat Yildirim[4]

[1]Oklahoma State University (IEM) • [2]Argonne National Laboratory (ESIA) • [3]Wayne State University (ECE) • [4]Wayne State University (ISE)

Full paper arXiv:2509.21137

Codebase disys-lab.github.io/meliso

ABSTRACT

Large-scale linear optimization underpins decision-making across science, engineering, and industry, yet its growing computational demand increasingly exceeds the efficiency of **conventional von Neumann hardware**.

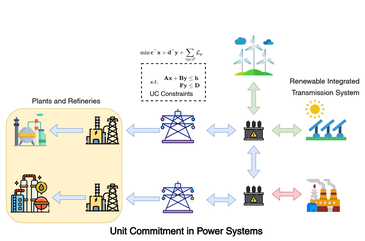
We present the **first primal-dual hybrid gradient (PDHG) linear-program solver co-designed for in-memory computing** on **resistive-RAM (RRAM) crossbar arrays**. Evaluated in a physics-based device simulation, the solver attains **accuracy comparable to a GPU** while reducing energy and latency by up to **three orders of magnitude**.

MOTIVATION — Where linear programs arise

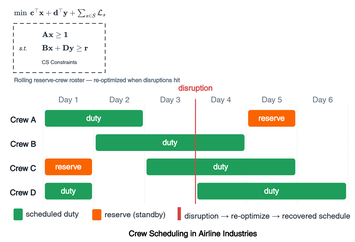
A **linear program (LP)** optimizes a linear **objective function** (e.g. cost, energy, or profit) subject to linear **constraints** that encode real-world limits (budgets, capacities, supply and demand):

$$\min_{\mathbf{x}\in\mathbb{R}^n} \ \mathbf{c}^\top\mathbf{x}, \qquad \text{s.t.} \qquad \mathbf{A}\mathbf{x}\le\mathbf{b};\ \ l\le x_i\le u,\ \forall x_i\in\mathbf{x}$$

Such problems underpin optimal decision-making across modern infrastructure:



Unit Commitment in Power Systems

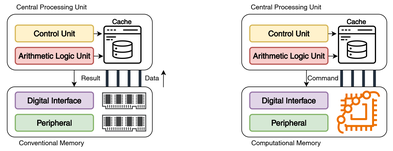


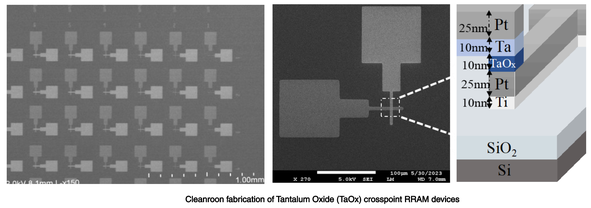
Crew Scheduling in Airline Industries

MOTIVATION — The memory wall and in-memory computing

Conventional von Neumann architectures separate the **processor** from the **memory**. For the large matrices underlying these problems, most energy and time is consumed *transferring data* between the two — the **memory wall**.

In-memory computing via RRAM technology eliminates this overhead: RRAM cells encode numerical values as electrical conductances arranged in a grid (a *crossbar*); thus, applying input voltages yields a complete matrix-vector product as output currents in **a single analog step**.





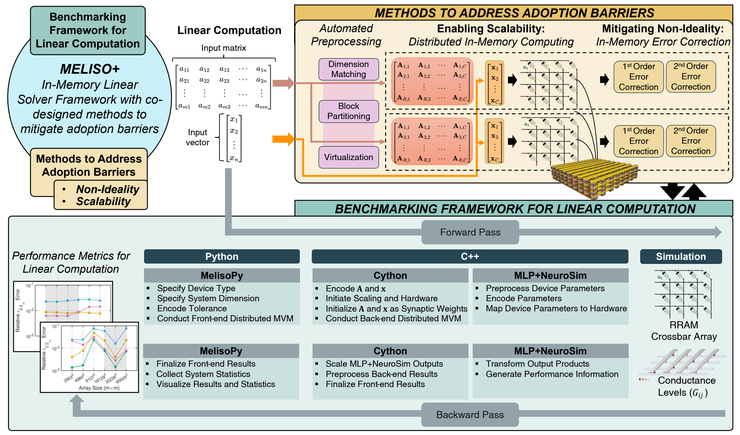
Cleanroom fabrication of Tantalum Oxide (TaOx) crosspoint RRAM devices

CHALLENGES — Each generation of solver trades one limitation for another

- **Exact digital solvers:** Classical methods (e.g. **Gurobi**) rely on repeated **matrix factorizations**. They return accurate, certified solutions but incur prohibitive memory overhead at scale.
- **GPU first-order methods:** Newer methods such as **PDHG** exploit the abundance of GPUs to avoid factorization altogether, yet remain bound by the **von Neumann bottleneck**.
- **Analog in-memory computing:** Devices such as **RRAM** are **noisy and non-ideal**. Additionally, the **writing operations are expensive** in RRAMs, so the problem instances (a.k.a. the matrices) *must not* be re-encoded each iteration.

METHODOLOGY — The MELISO+ simulation framework

We first build **MELISO+**, an in-memory linear-solver framework that simulates **RRAMs** under realistic device physics to measure **accuracy**, **latency**, and **energy**. It establishes how to perform the core primitive (**matrix-vector multiplication**, abbreviated as **MVM**) reliably on RRAM crossbar arrays by addressing two adoption barriers: **scalability** and **device non-ideality**.



- **Message Passing Interface (MPI)** framework is employed to distribute the problem instances to multiple crossbar arrays (MCAs).
- **Device properties** (e.g., device-to-device and cycle-to-cycle variations) are simulated to observe their effects on MVM results.

METHODOLOGY — In-memory optimization with PDHG

We then configure MELISO+ to solve linear programs entirely in memory using the **PDHG** method. Recast as a saddle-point problem,

$$\min_{l\le\mathbf{x}\le u}\ \max_{\mathbf{y}}\ \mathbf{c}^\top\mathbf{x}+\langle\mathbf{A}\mathbf{x}-\mathbf{b},\,\mathbf{y}\rangle,$$

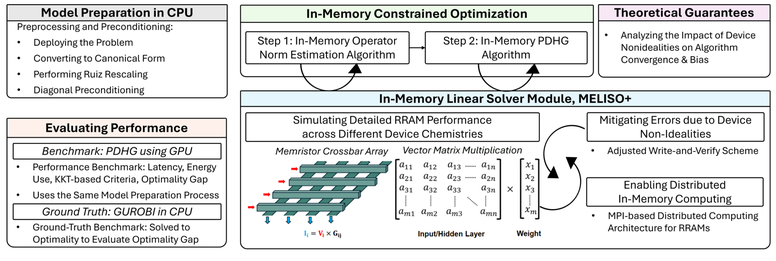
each iteration touches the matrix *only* through matrix–vector products:

$$\begin{aligned}\mathbf{y}_{k+1}&=\Pi_Y\big[\mathbf{y}_k+\sigma(\mathbf{A}\bar{\mathbf{x}}_k-\mathbf{b})\big],\\ \mathbf{x}_{k+1}&=\Pi_{[l,u]}\big[\mathbf{x}_k-\tau(\mathbf{c}+\mathbf{A}^\top\mathbf{y}_{k+1})\big],\\ \bar{\mathbf{x}}_{k+1}&=\mathbf{x}_{k+1}+\theta(\mathbf{x}_{k+1}-\mathbf{x}_k).\end{aligned}$$

Here, the products $\mathbf{A}\bar{\mathbf{x}}_k$ and $\mathbf{A}^\top\mathbf{y}_{k+1}$ are the in-memory MVMs; Π are cheap projections, $\{\sigma,\tau\}$ is the dual-primal step size, and θ is the extrapolation term.

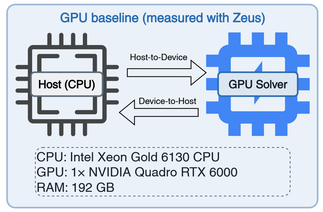
We introduce the following modifications to ensure the feasible implementation on RRAM devices:

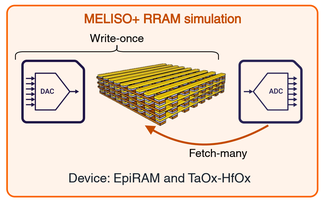
1. **Write-once encoding.** The problem is cast as a single *symmetric block matrix* $\mathbf{M}=\left[\begin{smallmatrix}0 & \mathbf{A}\\ \mathbf{A}^\top & 0\end{smallmatrix}\right]$ and programmed onto the crossbars **exactly once**; every iteration reuses this encoding, with only the input vectors changing.
2. **Distributed single accelerator.** A 4×4 tile of 64×64 crossbars (a 256×256 logical array) operates as one accelerator, eliminating inter-tile data movement.
3. **Accelerated convergence.** The PDHG iteration is augmented with **Nesterov momentum for adaptive step-size rule**, reducing the number of iterations required to reach a target tolerance.
4. **Robustness by design.** Stable operator-norm estimation (**Lanczos**) and **write-and-verify programming** mitigate device non-idealities, and we **prove convergence on noisy hardware** — no worse than a digital baseline up to a small, fixed error term.



EXPERIMENTS AND RESULTS

Benchmarked against **gpuPDLP** (a GPU version of the same method), with **Gurobi** as ground truth, on real LPs from MIPLIB-2017. Both RRAM devices (**TaO_x-HfO_x** and **EpiRAM**) match GPU/Gurobi accuracy (relative gap Δ_{rel}) at a fraction of the energy and latency.

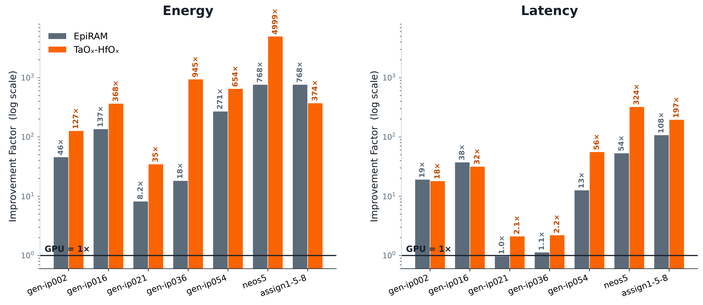


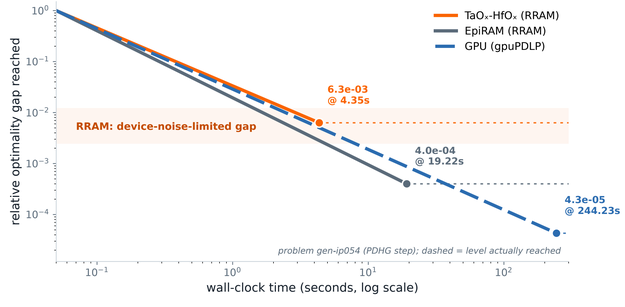


Energy and latency benchmark

Problem	Solver	Step 1: Lanczos Δ_{rel}	Step 1: Lanczos E [J]	Step 1: Lanczos t [s]	Step 2: PDHG Δ_{rel}	Step 2: PDHG E [J]	Step 2: PDHG t [s]
gen-ip002	gpuPDLP	—	11.94	0.34	1.5×10^{-5}	836.73	69.26
	EpiRAM	1.3×10^{-3}	0.75	0.34	5.0×10^{-6}	18.17	3.60
	TaO_x-HfO_x	2.0×10^{-4}	0.01	0.06	6.0×10^{-6}	6.57	3.81
gen-ip016	gpuPDLP	—	5.28	0.14	4.7×10^{-5}	2811.55	141.95
	EpiRAM	4.1×10^{-3}	2.75	1.38	1.2×10^{-4}	20.55	3.78
	TaO_x-HfO_x	7.0×10^{-4}	0.07	0.04	1.3×10^{-4}	7.64	4.47
gen-ip021	gpuPDLP	—	4.45	0.12	2.6×10^{-6}	172.17	6.09
	EpiRAM	1.8×10^{-3}	1.01	0.15	3.0×10^{-2}	20.88	6.21
	TaO_x-HfO_x	1.9×10^{-2}	0.02	0.08	3.0×10^{-2}	4.92	2.87
gen-ip036	gpuPDLP	—	2.84	0.11	4.8×10^{-6}	365.10	8.22
	EpiRAM	4.0×10^{-4}	18.42	7.13	5.5×10^{-5}	19.89	7.23
	TaO_x-HfO_x	1.7×10^{-3}	0.08	0.14	5.4×10^{-5}	0.39	3.70
gen-ip054	gpuPDLP	—	2.72	0.11	4.3×10^{-5}	4870.16	244.23
	EpiRAM	4.0×10^{-4}	4.34	3.62	3.4×10^{-3}	17.94	19.22
	TaO_x-HfO_x	6.3×10^{-3}	0.06	0.04	7.0×10^{-3}	7.44	4.35
neos5	gpuPDLP	—	2.75	0.09	5.6×10^{-4}	269.46	10.16
	EpiRAM	0	0.17	0.08	7.7×10^{-3}	0.35	0.19
	TaO_x-HfO_x	0	0.10	0.04	1.5×10^{-2}	0.05	0.03
assign1-5-8	gpuPDLP	—	2.81	0.10	3.8×10^{-5}	1417.42	410.24
	EpiRAM	2.8×10^{-3}	1.07	0.79	1.7×10^{-3}	6.54	3.79
	TaO_x-HfO_x	1.0×10^{-4}	0.54	0.19	4.2×10^{-3}	3.79	2.09

Total energy and latency measurement scheme





CONCLUSION

- The **first** PDHG linear-program solver simulated on RRAM — a true **algorithm–hardware co-design** (write-once encoding + noise-robust iteration).
- **Provable** convergence on noisy analog hardware; **orders-of-magnitude** energy and latency gains over a GPU.

REFERENCES & ACKNOWLEDGMENTS

[1] Vo, Chowdhury, Ramanan *et al.* From GPUs to RRAMs: Distributed In-Memory Primal–Dual Hybrid Gradient Method for Solving Large-Scale Linear Optimization Problems. *SIAM PP* (2026). arXiv:2509.21137.
[2] Vo, Chowdhury, Ramanan *et al.* Harnessing the full potential of RRAMs through scalable and distributed in-memory computing with integrated error correction. *Nature Communications Engineering* (2026). arXiv:2508.13298
[3] Chowdhury *et al.* The Lynchpin of In-Memory Computing: A Benchmarking Framework for Vector-Matrix Multiplication in RRAMs. *ICONS* (2024). arXiv:2409.06140
[4] Lu *et al.* A GPU Implementation of Restarted Primal-Dual Hybrid Gradient for Linear Programming in Julia. arXiv:2311.12180

Conducted at Oklahoma State University, Argonne National Laboratory, and Wayne State University. Supported in part by the National Science Foundation Secure and Trustworthy Cyberspace (SaTC) program under Award No. 2348411.