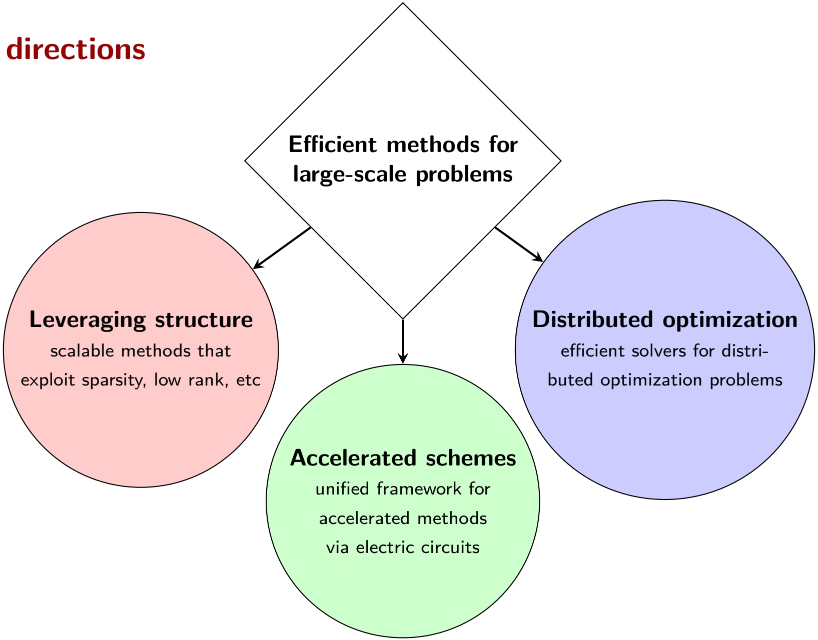
Design and Analysis of Efficient Algorithms for Large-Scale Problems

Tetiana Parshakova

Flatiron Institute

1/6/25

Research directions



Reference:

T. Parshakova, T. Hastie, E. Darve, and S. Boyd. (2024).
Factor fitting, rank allocation, and partition in multilevel low rank matrices.
To appear in *Optimization, Discrete Mathematics, and Applications to Data Sciences*. Springer.

T. Parshakova, T. Hastie, and S. Boyd. (2024).
Fitting multilevel factor models. *Submitted*.

Leveraging structure

scalable methods that exploit sparsity, low rank, etc

efficient solvers for distributed optimization problems

Accelerated schemes

unified framework for accelerated methods via electric circuits

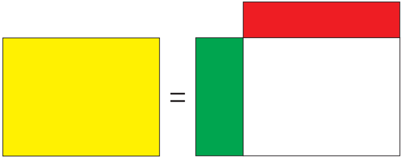
Low rank data

- in many applications data is organized in a matrix, $A \in \mathbf{R}^{m \times n}$
  - gene expressions in cells
- in practice the data is often approximately low rank [Eckart+Young36, Candès+Recht09, Udell+16]

$$A_{ij} \approx b_i^T c_j, \qquad b_i, c_j \in \mathbf{R}^r, \qquad r \ll \min\{m, n\}$$

  - per-cell coefficients and per-gene factors

Low rank matrix approximation



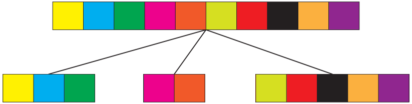
- find $B \in \mathbf{R}^{m \times r}$ and $C \in \mathbf{R}^{n \times r}$ such that $A \approx BC^T$

$$\text{minimize} \quad \|A - BC^T\|_F^2 = \sum_{i,j=1}^{m,n} (A_{ij} - b_i^T c_j)^2$$

- storage compression from mn to $(m+n)r$
- fast matrix-vector multiplication from mn flops to $2(m+n)r$
- interpretable factors
- solved via the singular value decomposition (SVD), proposed in 1907 [Schmidt07]

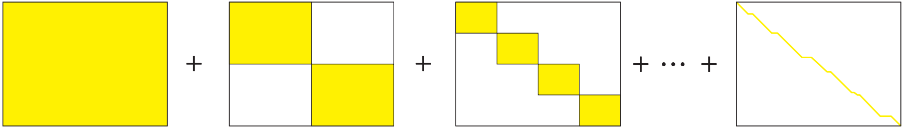
Hierarchically structured data

- biology: cells, tissues, organs
- geography: cities, states, countries
- finance: industries, groups, sectors
- healthcare: patients, clinics, regions
- education: students, classrooms, schools



Contiguous multilevel low rank matrices

- an $m \times n$ contiguous multilevel low rank (MLR) matrix A with L levels

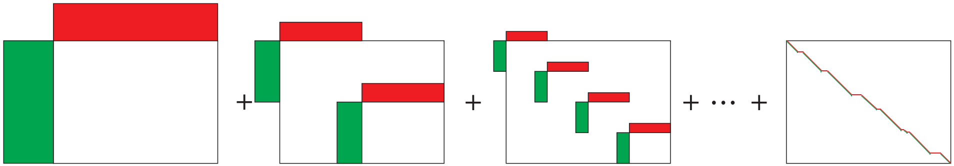


$$A = A^1 + \cdots + A^L, \qquad A^l = \mathbf{diag}(A_{l,1}, \ldots, A_{l,p_l})$$

- groups in partitions are contiguous ranges of row/column indices

Contiguous multilevel low rank matrices

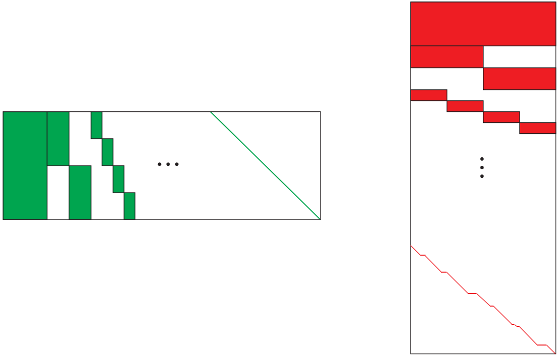
- an $m \times n$ contiguous multilevel low rank (MLR) matrix A with L levels



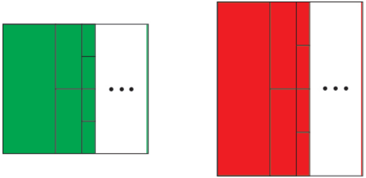
$$A_{l,k} = B_{l,k} C_{l,k}^T, \qquad B_{l,k} \in \mathbf{R}^{m_{l,k} \times r_l}, \qquad C_{l,k} \in \mathbf{R}^{n_{l,k} \times r_l}$$

- groups in partitions are contiguous ranges of row/column indices

Two-matrix form



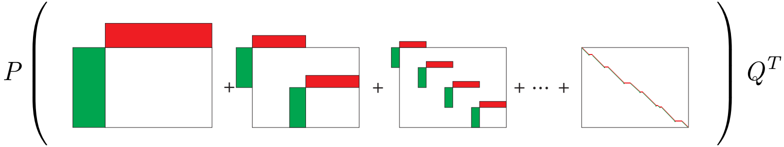
(a) factor form $A = \tilde{B}\tilde{C}^T$



(b) compressed factor form
with $(m+n)r$ coefficients, where
the MLR-rank of A $r = r_1 + \cdots + r_L$

Multilevel low rank matrices

- general $m \times n$ MLR matrix has the form



  - $P \in \mathbf{R}^{m \times m}$ is the row permutation matrix
  - $Q \in \mathbf{R}^{n \times n}$ is the column permutation matrix
- general hierarchical partition of the row and column index sets

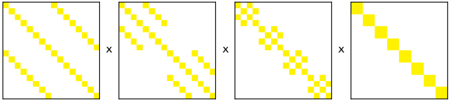
Multilevel low rank matrices

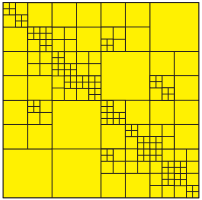
MLR matrix with MLR-rank r

- permutations P and Q
- the number of levels L
- the block dimensions $m_{l,k}$ and $n_{l,k}$, $l = 1, \ldots, L$, $k = 1, \ldots, p_l$
- the two matrices B and C
- ranks r_i s.t. $r_1 + \cdots + r_L = r$

Related work

- Hierarchical matrices
  - $\mathcal{H}$-matrix [Greengard+Rokhlin87,Hackbusch99]
  - $\mathcal{H}^2$-matrix [Greengard+Rokhlin87,Hackbusch+Borm02, Darve00]
  - hierarchically off-diagonal low-rank (HODLR) [Aminfar+16]
  - hierarchical semiseparable (HSS) matrix [Chandrasekaran+06]
- block low rank matrices [Amestoy+15]
- butterfly matrices [Parker95]
  - Monarch matrices [Dao+22]





Example: Distance matrix

- distance matrix for Venice roadmap
- $n = 5893$ nodes and 12098 edges
- $L = 14$ levels and MLR-rank $r = 98$
- compression ratio $30 : 1$

Method	Error (%)	Storage ($\times 10^5$)
LR	0.72	5.78
LR+D	0.71	5.78
HODLR	2.50	5.79
Monarch	0.87	5.88
MLR	$\mathbf{0.37}$	5.78

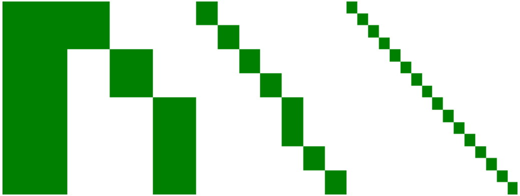
Properties of MLR matrices

- matrix-vector multiply in $2(m+n)r$ flops vs mn in the dense case
- linear system solve
  - via recursive Sherman-Morrison-Woodbury in $O(nr^2)$ vs $O(n^3)$ in the dense case
  - via direct sparse solver
- k largest eigenvalues, total cost at iteration k
  - Arnoldi iteration with $O(nrk+nk^2)$ vs $O(n^2k+nk^2)$ dense case
  - Lanczos algorithm with $O(nrk+nk)$ vs $O(n^2k+nk)$ dense case

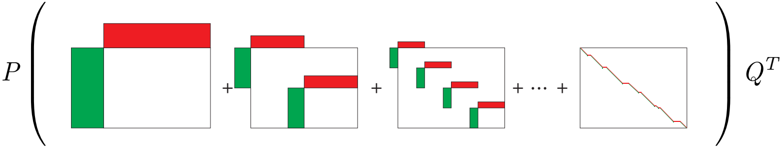
Example: Linear system solve

- solve $Ax = b$
  - A PD MLR matrix, with $n = 10^5$ and compression ratio $750 : 1$
- dense matrix in single precision requires 37Gb

- direct dense solve using Cholesky
  - extrapolated time (from 10s for $10^4 \times 10^4$ matrix) is **2.7h** on M2 chip
- recursive SMW
  - solve in **200ms** on M2 chip: $\times 50000$ faster than the dense one



Fitting problems



- how to fit the factors?
- how to allocate ranks across levels?
- how to choose hierarchical partition?

Summary of contributions

- MLR matrices are natural extensions for low rank matrices
- fast linear algebra and storage compression
- Frobenius norm and MLE-based fitting methods
- model general hierarchical structures
- identify factors explaining data at global and local scales
- applications to real-world distance matrices, asset covariance matrices, kernel matrices
- open-source packages
  - `mlrfit`: https://github.com/cvxgrp/mlr_fitting
  - `mfmodel`: https://github.com/cvxgrp/multilevel_factor_model

Future directions

- compress NNs by replacing dense layers with MLR
- learning graph structure
  - single-cell gene expression datasets: genes x cells
  - bacterial/metagenomic datasets: microbial features x samples
- scalable fitting methods for latent variable graphical models
  - build on ideas from chordal embedding and randomized graph sparsification
- iterative graph neural network (GNN) integration
  - use the conditional independence graph to guide GNNs, updating the graph as new embeddings emerge

Reference:

S. Boyd, **T. Parshakova**, E. Ryu, and J. Suh. (2024).
Optimization algorithm design via electric circuits.
Accepted (Spotlight) to Conference on
Neural Information Processing System.

scalable methods that
exploit sparsity, low rank, etc

efficient solvers for distri-
buted optimization problems

Accelerated schemes

unified framework for
accelerated methods
via electric circuits

Distributed convex optimization problem

$$
\begin{array}{ll}
\text{minimize} & f(x) \\
\text{subject to} & x \in \mathcal{R}(E^\intercal)
\end{array}
$$

- f: $\mathbf{R}^m \to \mathbf{R} \cup \{\infty\}$ is closed, convex, and proper
- n nets $N_1, \ldots, N_n$ forming a partition of $\{1, \ldots, m\}$
- $E \in \mathbf{R}^{n \times m}$ is a selection matrix

$$
E_{ij} = \begin{cases} +1 & \text{if } j \in N_i \\ 0 & \text{otherwise} \end{cases}
$$

Example: Consensus problem

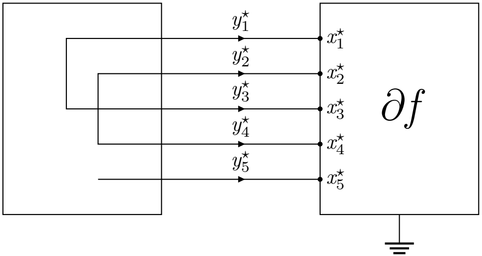
$$\begin{array}{ll} \underset{x_1,\ldots,x_N \in \mathbf{R}^{m/N}}{\text{minimize}} & f_1(x_1) + \cdots + f_N(x_N) \\ \text{subject to} & x_1 = \cdots = x_N \end{array}$$

- $x = (x_1, \ldots, x_N) \in \mathbf{R}^m$ is the decision variable
- $f(x) = f_1(x_1) + \cdots + f_N(x_N)$ is block-separable
- $E^\intercal = (I, \ldots, I) \in \mathbf{R}^{m \times m/N}$

Circuit interpretation: KKT conditions

$$
\begin{array}{lll}
y & \in & \partial f(x) \quad \text{(stationarity)} \\
x & \in & \mathcal{R}(E^\intercal) \quad \text{(primal feasibility)} \\
y & \in & \mathcal{N}(E) \quad \text{(dual feasibility)}
\end{array}
$$

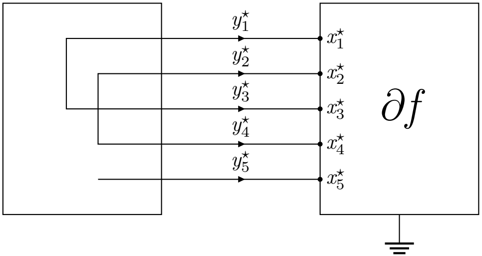
Static interconnect



Circuit interpretation: KKT conditions

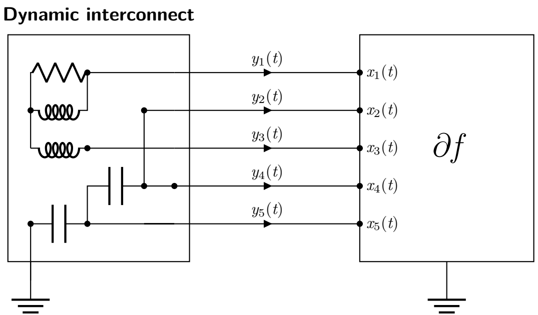
$$\begin{aligned} y &\in \partial f(x) \quad \text{(nonlinear resistor)} \\ x &\in \mathcal{R}(E^\intercal) \quad \text{(KVL)} \\ y &\in \mathcal{N}(E) \quad \text{(KCL)} \end{aligned}$$

Static interconnect



Circuit interpretation: Dynamic interconnect

$$
\begin{aligned}
y(t) &\in \partial f(x(t)) && \text{(nonlinear resistor)} \\
v(t) &= A^\intercal \begin{bmatrix} x(t) \\ e(t) \end{bmatrix} && \text{(KVL)} \\
Ai(t) &= \begin{bmatrix} -y(t) \\ 0 \end{bmatrix} && \text{(KCL)} \\
v_{\mathcal{R}}(t) &= D_{\mathcal{R}} i_{\mathcal{R}}(t) && \text{(resistor)} \\
v_{\mathcal{L}}(t) &= D_{\mathcal{L}} \frac{d}{dt} i_{\mathcal{L}}(t) && \text{(inductor)} \\
i_{\mathcal{C}}(t) &= D_{\mathcal{C}} \frac{d}{dt} v_{\mathcal{C}}(t) && \text{(capacitor)}
\end{aligned}
$$



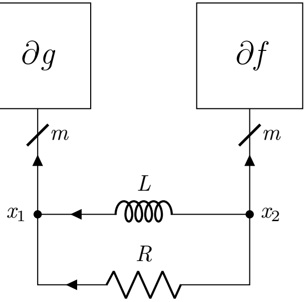
Circuits for classical algorithms: DRS

- V-I relations

$$
\begin{aligned}
x_1 &= \mathbf{prox}_{Rg}(x_2 + Ri_{\mathcal{L}}) \\
x_2 &= \mathbf{prox}_{Rf}(x_1 - Ri_{\mathcal{L}}) \\
\frac{d}{dt} i_{\mathcal{L}} &= \frac{1}{L}(x_2 - x_1)
\end{aligned}
$$

- Douglas–Rachford splitting

$$
\begin{aligned}
x_1^{k+1} &= \mathbf{prox}_{Rg}(x_2^k + Ri_{\mathcal{L}}^k) \\
x_2^{k+1} &= \mathbf{prox}_{Rf}(x_1^{k+1} - Ri_{\mathcal{L}}^k) \\
i_{\mathcal{L}}^{k+1} &= i_{\mathcal{L}}^k + \frac{h}{L}(x_2^{k+1} - x_1^{k+1})
\end{aligned}
$$



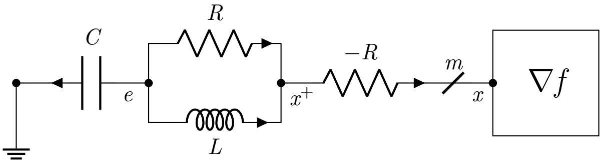
Circuits for classical algorithms: Nesterov acceleration

- V-I relations

$$\begin{aligned}\frac{d}{dt}i_{\mathcal{L}} &= D_{\mathcal{L}}^{-1}(v_{\mathcal{C}} - x^{+})\\ \frac{d}{dt}v_{\mathcal{C}} &= -D_{\mathcal{C}}^{-1}\nabla f(x).\end{aligned}$$

- Nesterov acceleration

$$\frac{d^2}{dt^2}x + 2\sqrt{\mu}\frac{d}{dt}x + \sqrt{s}\frac{d}{dt}\nabla f(x) + (1+\sqrt{\mu s})\nabla f(x) = 0$$



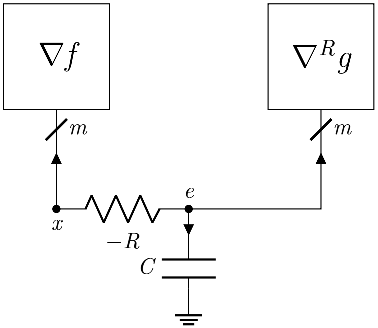
Circuits for classical algorithms: Proximal gradient

- V-I relations

$$\begin{aligned} i_{\mathcal{C}} &= -\nabla f(x) - \nabla^R g(e) \\ v_{\mathcal{C}} &= x - R\nabla f(x) \end{aligned}$$

- Proximal gradient method

$$x^{k+1} = \mathbf{prox}_{Rg}(I - R\nabla f)(x^k)$$



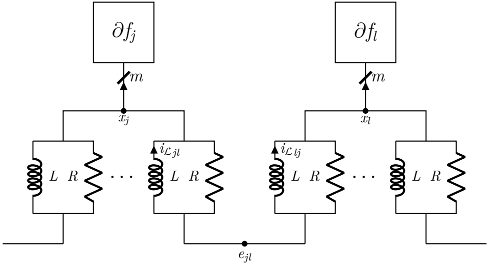
Circuits for classical algorithms: DADMM

- V-I relations

$$
\begin{aligned}
x_j &= \mathbf{prox}_{(R/|\Gamma_j|)f_j}\left(\frac{1}{|\Gamma_j|}\sum_{l\in\Gamma_j}(Ri_{\mathcal{L}\,jl}+e_{jl})\right)\\
e_{jl} &= \frac{1}{2}(x_j+x_l)\\
\frac{d}{dt}i_{\mathcal{L}\,jl} &= \frac{1}{L}(e_{jl}-x_j)
\end{aligned}
$$

- Decentralized ADMM

$$
\begin{aligned}
x_j^{k+1} &= \mathbf{prox}_{(R/|\Gamma_j|)f_j}\left(\frac{1}{|\Gamma_j|}\sum_{l\in\Gamma_j}(Ri_{\mathcal{L}\,jl}^{k}+e_{jl}^{k})\right)\\
e_{jl}^{k+1} &= \frac{1}{2}(x_j^{k+1}+x_l^{k+1})\\
i_{\mathcal{L}\,jl}^{k+1} &= i_{\mathcal{L}\,jl}^{k}+\frac{1}{R}(e_{jl}^{k+1}-x_j^{k+1})
\end{aligned}
$$



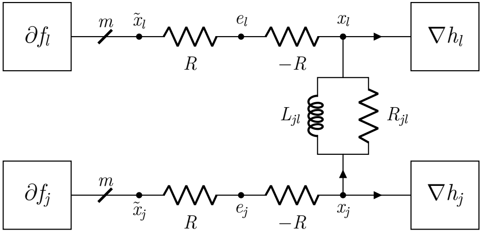
Circuits for classical algorithms: PG-EXTRA

- V-I relations

$$
\begin{aligned}
x_j &= \mathbf{prox}_{Rf_j}\left(\sum_{l=1}^{N} W_{jl}x_l - R\nabla h_j(x_j) - w_j\right) \\
\frac{d}{dt}w_j &= x_j - \sum_{l=1}^{N} W_{jl}x_l
\end{aligned}
$$

- PG-EXTRA

$$
\begin{aligned}
x^{k+1} &= \mathbf{prox}_{Rf}\left(Wx^k - R\nabla h(x^k) - w^k\right) \\
w^{k+1} &= w^k + \frac{1}{2}(I - W)x^k
\end{aligned}
$$



Energy dissipation

- in continuous time, energy dissipation leads to convergence (Thm 2.2, [Boyd+2024])
- $\mathcal{E}(t) = \frac{1}{2}\|v_{\mathcal{C}}(t) - v_{\mathcal{C}}^\star\|_{D_{\mathcal{C}}}^2 + \frac{1}{2}\|i_{\mathcal{L}}(t) - i_{\mathcal{L}}^\star\|_{D_{\mathcal{L}}}^2$
- $\frac{d}{dt}\mathcal{E} \le -\langle x(t) - x^\star, y(t) - y^\star\rangle \le 0$
- $\lim_{t\to\infty} x(t) = x^\star$

- not every discretization leads to a convergent algorithm

Automatic discretization

- find discretization preserving the proof structure (Lemma 4.1, [Boyd+2024])
- $\mathcal{E}_k = \frac{1}{2}\|v_{\mathcal{C}}^k - v_{\mathcal{C}}^\star\|_{D_{\mathcal{C}}}^2 + \frac{1}{2}\|i_{\mathcal{L}}^k - i_{\mathcal{L}}^\star\|_{D_{\mathcal{L}}}^2$
- $\mathcal{E}_{k+1} - \mathcal{E}_k + \eta\langle x^k - x^\star, y^k - y^\star\rangle \leq 0$ for some $\eta > 0$
- $\sum_{k=1}^{\infty}\langle x^k - x^\star, y^k - y^\star\rangle < \infty$
- $\lim_{k\to\infty} x^k = x^\star$

- automate using computer-assisted proof framework PEP
  - open-source package `ciropt`: `https://github.com/cvxgrp/optimization_via_circuits`

Existing discretization

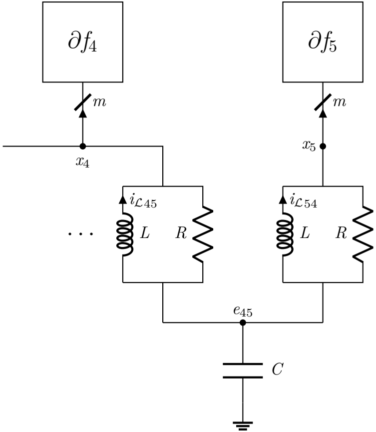
previous discretization studies divide into two categories

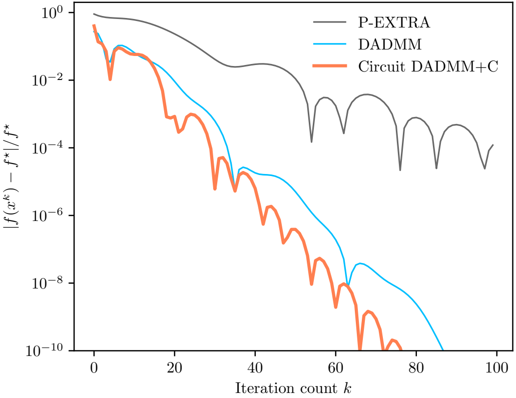
- apply standard discretization schemes or their variants [Runge+1895; Kutta+1901]
  - focus on the convergence of the discretized sequence to the solution trajectory in $[0, T]$
  - not our interest, fails when $T \to \infty$ [Iserles2009]
- special rules tailored to the specific dynamics [Alvarez+2001; Su+2016; Wibisono+2016; Attouch+2019; Wilson+2019]
  - strategy cannot work in general

we provide a novel discretization methodology

- aim to find parameters that preserve the proof structure
- find such parameters automatically by leveraging PEP

Numerical results: DADMM+C





Summary of contributions

- introduce a framework for designing optimization algorithms via RLC circuits
  - design dynamic circuit that converges to the solution
  - discretize to obtain convergent algorithm
- electric circuits for standard methods
  - Nesterov acceleration, proximal point method, prox-gradient, primal decomposition, dual decomposition, DYS, DRS, decentralized gradient descent, diffusion, DADMM and PG-EXTRA
- convergence proof of circuit dynamics based on energy dissipation
- PEP-based automated discretization that preserves proof structure
  - open-source package `ciropt`: https://github.com/cvxgrp/optimization_via_circuits

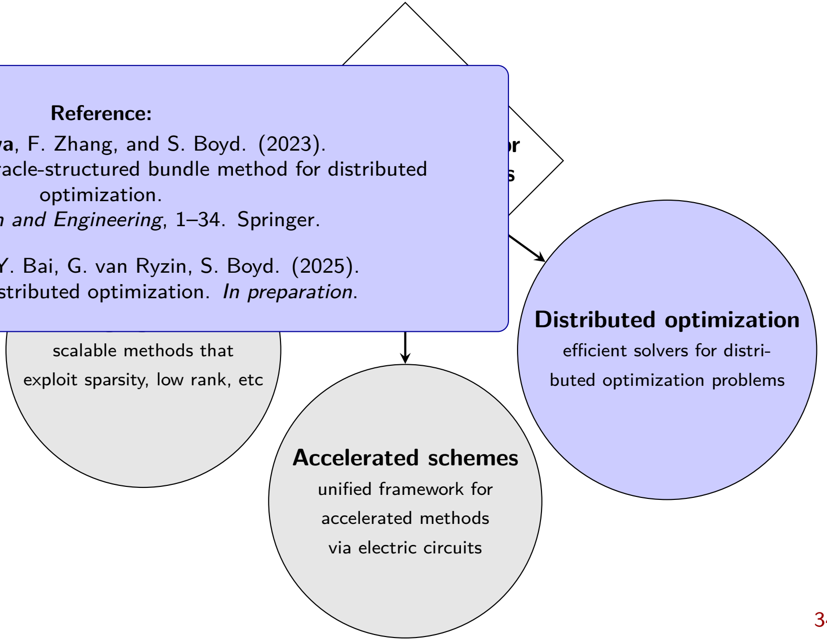
Future directions

- extending the framework for stochastic programming
- extracting convergence rates
  - energy dissipation over multiple steps
- include methods with time dependent step sizes
  - using time dependent electric components
- key question: how to automate the development of accelerated algorithms?
  - search over admissible circuit designs
  - find circuits which result in fast relaxation, *e.g.*, critical damping
  - guidance on how to design fast optimization method from circuit architecture

Reference:

T. Parshakova, F. Zhang, and S. Boyd. (2023).
Implementation of an oracle-structured bundle method for distributed optimization.
Optimization and Engineering, 1–34. Springer.

T. Parshakova, Y. Bai, G. van Ryzin, S. Boyd. (2025).
Price directed distributed optimization. *In preparation*.



Oracle-structured distributed optimization problem

$$\text{minimize} \quad h(x) = f(x) + g(x)$$

- $x = (x_1, \ldots, x_M) \in \mathbf{R}^n$ is variable, $x_i \in \mathbf{R}^{n_i}$
- $f(x) = \sum_{i=1}^M f_i(x_i)$ is block separable
- f_i convex, accessed by value/subgradient oracle
- $g : \mathbf{R}^n \to \mathbf{R} \cup \{\infty\}$ is convex structured objective function
- coordinator can solve an optimization problem involving g

Our goals

classical setting	**our setting**
agents easy to query	agents costly to query
coordinator performs simple operations (*e.g.*, averaging)	coordinator can do more than average

we are interested in methods that

- find good points in tens of iterations or fewer
- have *zero* hyper-parameters to tune
- handle agent delays and failures

Methods

- (accelerated) proximal subgradient
- ADMM, Douglas-Rachford
- cutting plane/bundle methods

- we've settled on cutting-plane/bundle methods
- these methods build up a piecewise linear model of each f_i

Agent objective minorants

- algorithm maintains a *minorant* $\hat{f}_i$: $\hat{f}_i(x) \leq f_i(x)$ for all x
- at iteration k, query each agent i at $x_i^{(k+1)}$ to get

$$f_i(x_i^{(k+1)}), \qquad q_i^{(k+1)} \in \partial f_i(x^{(k+1)})$$

- update minorant of agent i

$$\hat{f}_i^{(k+1)}(x_i) = \max\left(\hat{f}_i^{(k)}(x_i),\ f_i(x_i^{(k+1)}) + (q_i^{(k+1)})^T(x_i - x_i^{(k+1)})\right)$$

- update minorant of h

$$\hat{h}^{(k+1)}(x) = \hat{f}_1^{(k+1)}(x) + \cdots + \hat{f}_M^{(k+1)}(x_M) + g(x)$$

Proximal minorant algorithm

Algorithm

given $x^{(0)} \in \textbf{dom}\, h$, $h(x^{(0)})$, initial minorants $\hat{f}_i^{(0)}$ and stepsize $\rho^{(0)}$.

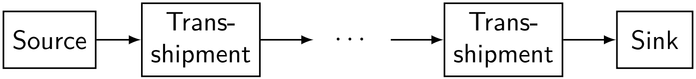
for $k = 0, 1, \ldots$

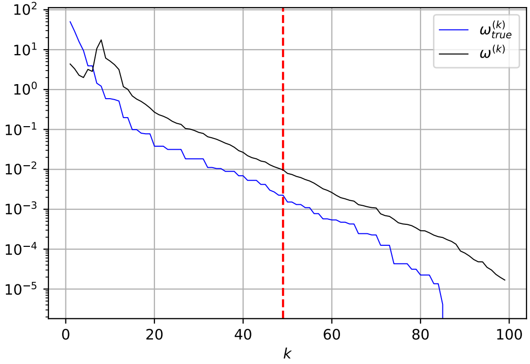
1. *Check stopping criterion.*
2. *Update iterate.* $x^{(k+1)} = \operatorname{argmin}_x \left(\hat{h}^{(k)}(x) + (\rho^{(k)}/2)\|x - x^{(k)}\|_2^2\right)$.
3. *Query agents.* Evaluate $f_i(x_i^{(k+1)})$ and $q_i^{(k+1)} \in \partial f_i(x_i^{(k+1)})$.
4. *Update minorants.* Update $\hat{f}_i^{(k+1)}$, $i = 1, \ldots, M$, and $\hat{h}^{(k+1)}$.

- relative duality gap with $L^{(k)} = \min_x \hat{h}^{(k)}(x)$, $U^{(k)} = \min\{U^{(k-1)}, h(x^{(k)})\}$,

$$\omega^{(k)} = \frac{U^{(k)} - L^{(k)}}{\min\{|U^{(k)}|, |L^{(k)}|\}}$$

Numerical results: Optimality gap (relative)





Summary of contributions

- assemble several methods into a single algorithm
  - disaggregate partially exact bundle method
  - diagonal preconditioning
  - level bundle methods
- *zero* hyper-parameters to tune
- handle agent delays and failures
- works well on wide range of practical problems
  - achieves 1% accuracy in *tens of iterations*
- open-source package `OSBDO`: https://github.com/cvxgrp/OSBDO

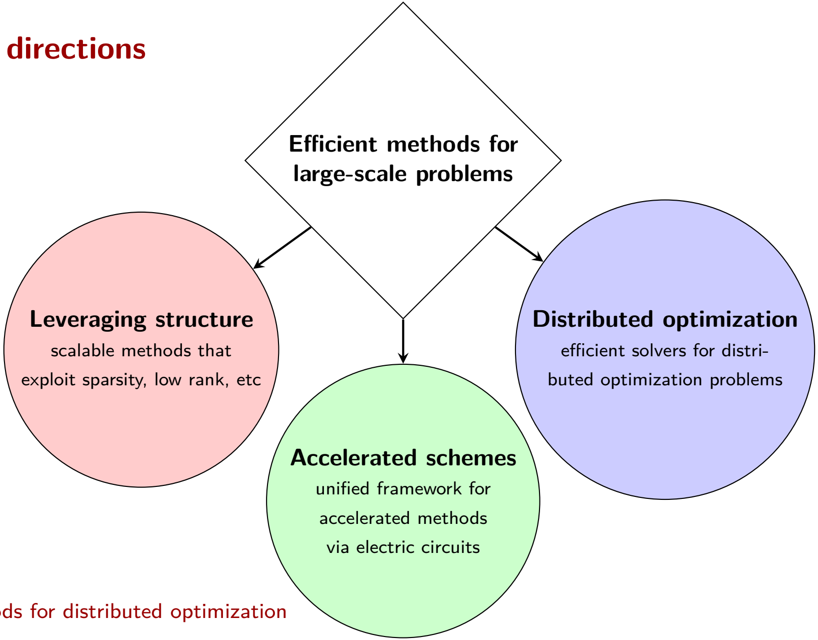
Future directions

setting: substantial computational cost per agent

- develop postprocessing method
  - uses low-precision optimal dual variable
  - recovers close to feasible primal point
  - uses only parallel calls to agents; avoids sequential calls
- recover near-feasible point with tolerable suboptimality using parallel agent calls

- combining first-order methods with localization methods
  - faster ADMM with analytic centers

Research directions



Future directions

- compress NNs by replacing dense layers with MLR
- learning graph structure from data

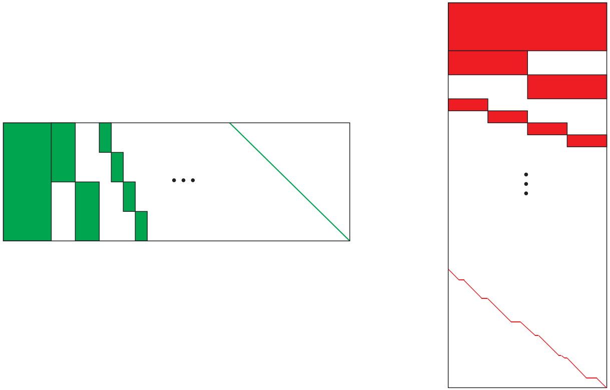
- automating the development of accelerated algorithms
  - search over admissible circuit designs
  - find circuits which result in fast relaxation, *e.g.*, critical damping

- recover near-feasible point with tolerable suboptimality using parallel agent calls
- combining first-order methods with localization methods

Thanks!

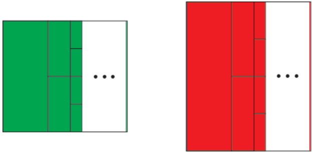
Factor form

- arrange factors such that $A = \tilde{B}\tilde{C}^T$



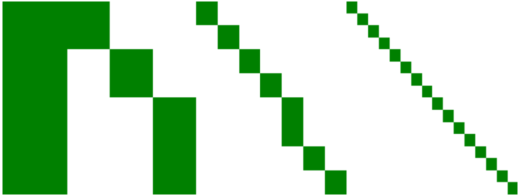
Compressed form

- $$B^l = \begin{bmatrix} B_{l,1} \\ \vdots \\ B_{l,p_l} \end{bmatrix} \in \mathbf{R}^{m \times r_l}, \qquad C^l = \begin{bmatrix} C_{l,1} \\ \vdots \\ C_{l,p_l} \end{bmatrix} \in \mathbf{R}^{n \times r_l}$$
- $$B = \begin{bmatrix} B^1 & \cdots & B^L \end{bmatrix} \in \mathbf{R}^{m \times r}, \qquad C = \begin{bmatrix} C^1 & \cdots & C^L \end{bmatrix} \in \mathbf{R}^{n \times r}$$
- $r = r_1 + \cdots + r_L$ is the MLR-rank of A



Example: Linear system solve

- solve $Ax = b$ with A positive definite MLR matrix
- $n = 10^5$
- dense matrix in single precision requires 37Gb
- hierarchical partition $p_1 = 1$, $p_2 = 3$, $p_3 = 7$, $p_4 = 16$, $p_5 = 10^5$
- ranks $r_1 = 30$, $r_2 = 20$, $r_3 = 10$, $r_4 = 5$, $r_5 = 1$
- compression ratio $750 : 1$



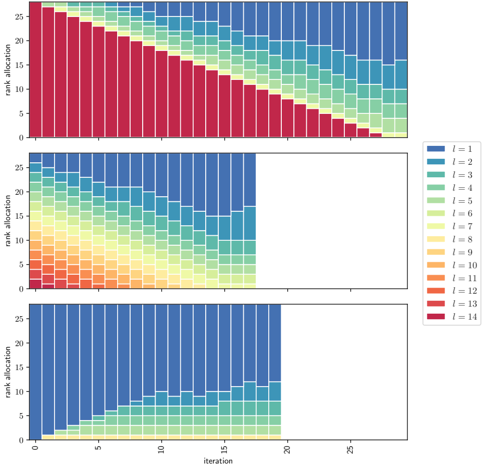
Example: Linear system solve

- direct dense solve using Cholesky
  - extrapolated time (from 10s for $10^4 \times 10^4$ matrix) is **2.7h** on M2 chip
- recursive SMW
  - solve in **200ms** on M2 chip
- MLR solve is $\times 50000$ faster than the dense one

Example: Discrete Gauss transform matrix

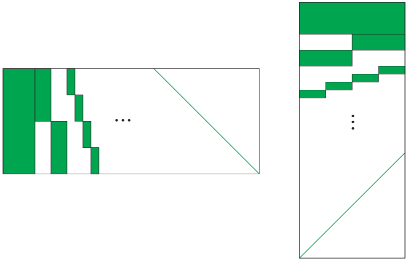
- $A_{ij} = e^{-\|t_i - s_j\|_2^2/h^2}$ and $s_j, t_i \in \mathbf{R}^d$
- $m = 5000,\ n = 7000,\ r = 28,\ L = 14,\ d = 3$, and $h = 0.2$
- compression ratio $100:1$

Method	Error (%)	Storage ($\times 10^5$)
LR	41.8	3.36
HODLR	72.5	3.39
Monarch	44.0	3.60
MLR bottom	**16.8**	3.36
MLR uniform	21.8	3.36
MLR top	25.8	3.36



PSD MLR

- symmetric positive semidefinite (PSD) MLR matrices
  - each block $A_{l,k} = B_{l,k} B_{l,k}^T$ is PSD

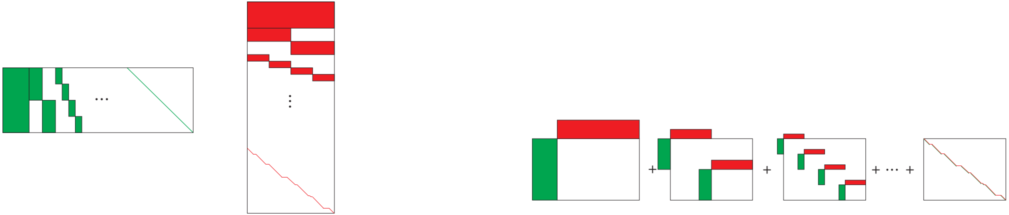


- PSD MLR is a covariance matrix in multilevel factor model (MFM) [Aitkin+81]

$$\Sigma = \begin{bmatrix} F & D^{1/2} \end{bmatrix} \begin{bmatrix} F & D^{1/2} \end{bmatrix}^T = FF^T + D$$

Factor fitting

- fix hierarchical partition and rank allocation
- optimize $\|P^T A Q - \hat{A}(B, C)\|_F^2$ over the factors B and C



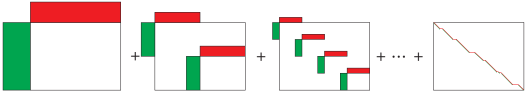
(a) alternating least squares (b) block coordinate descent

Rank allocation

- fix hierarchical partition
- optimize over the factors B and C and ranks $r_1, \ldots, r_L$ s.t. $r_1 + \cdots + r_L = r$

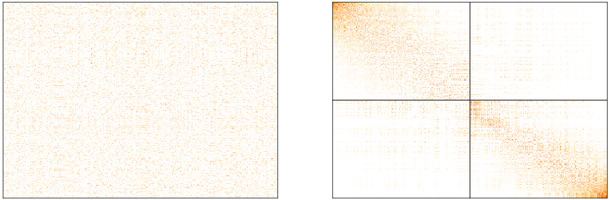
rank exchange algorithm

$$R = P^T A Q - \sum_{j \neq l} \mathbf{blkdiag}(B_{j,1} C_{j,1}^T, \ldots, B_{j,p_j} C_{j,p_j}^T)$$



Hierarchy fitting: Nested spectral dissection

1. $\tilde{R}_1 = (A - B_{1,1} C_{1,1}^T)$
2. $R_1 = P_1^T \tilde{R}_1 Q_1$
   - permutations P_1^T, Q_1^T maximize the sum of squares of residuals within the two diagonal blocks
3. $\tilde{R}_2 = R_1 - \begin{bmatrix} B_{2,1} C_{2,1}^T & 0 \\ 0 & B_{2,2} C_{2,2}^T \end{bmatrix}$
4. $\ldots$



Multilevel factor model

$$y = Fz + e$$

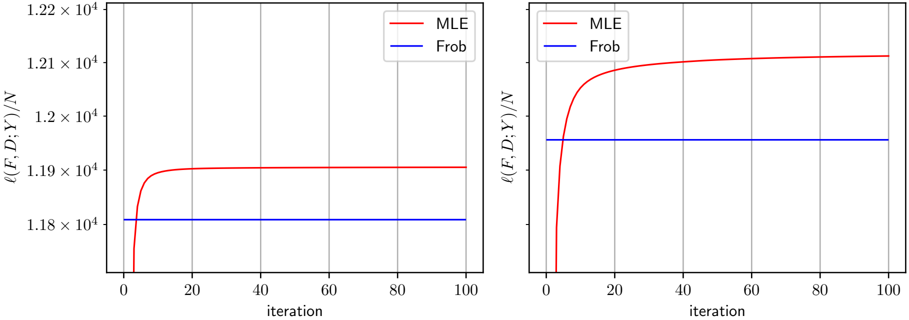
- $F \in \mathbf{R}^{n \times s}$ is structured factor loading matrix
- $z \in \mathbf{R}^s$ are factor scores, with $z \sim \mathcal{N}(0, I_s)$
- $e \in \mathbf{R}^n$ are unique terms, with $e \sim \mathcal{N}(0, D)$

Efficient computation

- computation of MLR Σ^{-1}
  - time complexity $O(nr^2 + p_{L-1} r_{\max} r^2)$
  - extra memory used is $3nr + 2p_{L-1} r_{\max} r$
- EM iteration
  - time complexity $O(p_{L-1} nr^2 + nr^3 + p_{L-1} nrN + p_{L-1} r_{\max} r^2)$

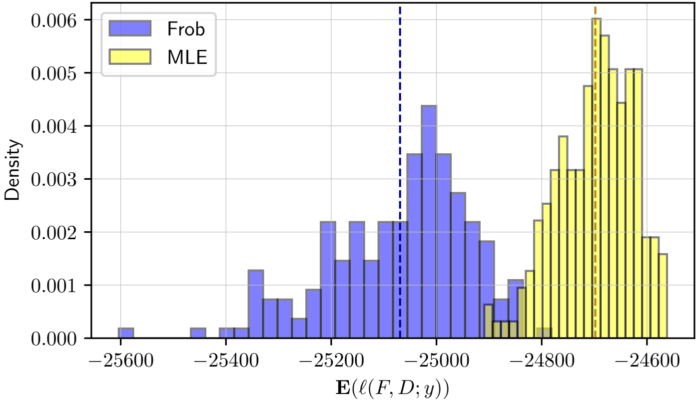
Example: Asset covariance matrix

- $n = 5000$, $L = 6$, $N = 300$, and $r = 30$
- compression ratio $80 : 1$
- log-likelihood for factor model (left) and multilevel factor model (right)



Example: Synthetic multilevel factor model

- $n = 10000$, $L = 6$, $r = 25$, $s = 174$, SNR of 4
- compression ratio $200 : 1$
- histograms over 100 runs each with sample size 200



Automatic discretization: PEP

for given discretization (α, β, h), and $\eta > 0$, dissipativity can be checked by

$$
\begin{array}{ll}
\text{maximize} & \mathcal{E}_2 - \mathcal{E}_1 + \eta\langle x^1 - x^\star, y^1 - y^\star\rangle \\
\text{subject to} & \mathcal{E}_s = \frac{1}{2}\|v_{\mathcal{C}}^s - v_{\mathcal{C}}^\star\|_{D_{\mathcal{C}}}^2 + \frac{1}{2}\|i_{\mathcal{L}}^s - i_{\mathcal{L}}^\star\|_{D_{\mathcal{L}}}^2, \quad s \in \{1, 2\} \\
& (v^1, i^1, x^1, y^1) \text{ is feasible initial point} \\
& (v^2, i^2, x^2, y^2) \text{ is generated by discrete optimization method from initial point} \\
& f \in \mathcal{F}
\end{array}
$$

- $f, v^1, i^1, x^1, y^1, v^\star, i^\star, x^\star, y^\star$ are the decision variables
- $\mathcal{F}$ is a family of functions (*e.g.*, L-smooth convex)

Agents

- when queried by coordinator at x_i, agent returns

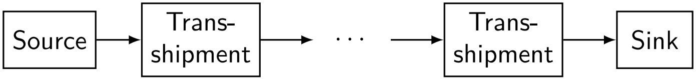
$$f_i(x_i), \qquad q_i \in \partial f_i(x_i)$$

- agents can include *private variables* z_i, with

$$f_i(x_i) = \min_{z_i} F_i(x_i, z_i)$$

- to evaluate $f_i(x_i)$ and $q_i \in \partial f_i(x_i)$ we solve an optimization problem

Example: Supply chain



- single commodity network with M trans-shipment components in series
- component i routes flows $a_i \in \mathbf{R}_+^{m_i}$ to flows $b_i \in \mathbf{R}_+^{n_i}$, with cost $f_i(a_i, b_i)$
- flow is conserved: $\mathbf{1}^T a_i = \mathbf{1}^T b_i$
- source and sink costs $\psi^{\mathrm{src}}(a_1) + \psi^{\mathrm{sink}}(b_M)$
- our instance
  - $M = 5$ with (m_i, n_i): $(20, 30)$, $(30, 40)$, $(40, 25)$, $(25, 35)$, $(35, 20)$
  - 300 variables; 4975 private variables