

High performance solution of linear optimization problems 2

Parallel techniques for the simplex method

Alternative methods for linear optimization

Julian Hall

School of Mathematics
University of Edinburgh
jajhall@ed.ac.uk

ALOP Autumn School, Trier

31 August 2017



THE UNIVERSITY
of EDINBURGH

- Exploiting parallelism
 - Background
 - Data parallelism by exploiting LP structure
 - Task parallelism for general LP problems
- Interior point methods
 - Numerical linear algebra requirements
 - “Matrix-free”
 - Quadratic programming (QP) problems
- Application in machine learning
- Pointers to the future

Simplex method: Computation

Standard simplex method (SSM): Major computational component

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\hat{N}$	$\hat{\mathbf{b}}$
	$\hat{\mathbf{c}}_N^T$	

Update of tableau: $\hat{N} := \hat{N} - \frac{1}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \hat{\mathbf{a}}_p^T$

where $\hat{N} = B^{-1}N$

- Hopelessly inefficient for sparse LP problems
- Prohibitively expensive for large LP problems

Revised simplex method (RSM): Major computational components

Pivotal row via $B^T \pi_p = \mathbf{e}_p$ **BTRAN** and $\hat{\mathbf{a}}_p^T = \pi_p^T N$ **PRICE**

Pivotal column via $B \hat{\mathbf{a}}_q = \mathbf{a}_q$ **FTRAN** Represent B^{-1} **INVERT**

Parallelising the simplex method

History

- SSM: Good parallel efficiency of $\hat{N} := \hat{N} - \frac{1}{\hat{a}_{pq}} \hat{\mathbf{a}}_q \hat{\mathbf{a}}_p^T$ **was** achieved
Many! (1988–date)
- Only parallel *revised* simplex is worthwhile: goal of H and McKinnon in 1992!

Parallel primal revised simplex method

- Overlap computational components for different iterations
Wunderling (1996), H and McKinnon (1995-2005)
- Modest speed-up **was** achieved on general sparse LP problems

Parallel dual revised simplex method

- Only immediate parallelism is in forming $\pi_p^T N$
- When $n \gg m$ significant speed-up **was** achieved
Bixby and Martin (2000)

Parallelising the simplex method: Matrix-vector product $\pi_p^T N$ (PRICE)

- **In theory:**

- Partition $N = [N_1 \ N_2 \ \dots \ N_P]$, where P is the number of processes
- Compute $\pi_p^T N_j$ on process j for $j = 1, 2, \dots, P$
- Execution time is reduced by a factor P : **linear speed-up**

- **In practice**

- On a distributed memory machine with N_j on processor j
Linear speed-up achieved
- On a shared memory machine
 - If N_j fits into cache for process j
Linear speed-up achieved
 - Otherwise, computation is limited by speed of reading data from memory
Speed-up limited to number of memory channels

Data parallelism by exploiting LP structure

Overview

- Written in C++ to solve stochastic MIP relaxations in parallel
- Dual simplex
- Based on NLA routines in `clp`
- Product form update

Concept

- Exploit data parallelism due to block structure of LPs
- Distribute problem over processes

Paper: Lubin, H, Petra and Anitescu (2013)

- COIN-OR INFORMS 2013 Cup
- COAP best paper prize (2013)

PIPS-S: Stochastic MIP problems

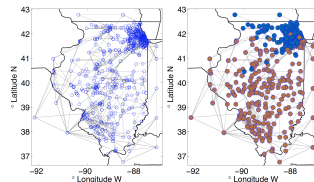
Two-stage stochastic LPs have column-linked block angular (BALP) structure

$$\begin{array}{llllllllll}
 \text{minimize} & \mathbf{c}_0^T \mathbf{x}_0 & + & \mathbf{c}_1^T \mathbf{x}_1 & + & \mathbf{c}_2^T \mathbf{x}_2 & + & \dots & + & \mathbf{c}_N^T \mathbf{x}_N & & \\
 \text{subject to} & \mathbf{A} \mathbf{x}_0 & & & & & & & & & = & \mathbf{b}_0 \\
 & \mathbf{T}_1 \mathbf{x}_0 & + & \mathbf{W}_1 \mathbf{x}_1 & & & & & & & = & \mathbf{b}_1 \\
 & \mathbf{T}_2 \mathbf{x}_0 & & & + & \mathbf{W}_2 \mathbf{x}_2 & & & & & = & \mathbf{b}_2 \\
 & \vdots & & & & & & \ddots & & & \vdots \\
 & \mathbf{T}_N \mathbf{x}_0 & & & & & & & + & \mathbf{W}_N \mathbf{x}_N & = & \mathbf{b}_N \\
 & \mathbf{x}_0 \geq \mathbf{0} & & \mathbf{x}_1 \geq \mathbf{0} & & \mathbf{x}_2 \geq \mathbf{0} & & \dots & & \mathbf{x}_N \geq \mathbf{0} & &
 \end{array}$$

- Variables $\mathbf{x}_0 \in \mathbb{R}^{n_0}$ are **first stage** decisions
- Variables $\mathbf{x}_i \in \mathbb{R}^{n_i}$ for $i = 1, \dots, N$ are **second stage** decisions
Each corresponds to a **scenario** which occurs with modelled probability
- The objective is the expected cost of the decisions
- In stochastic MIP problems, some/all decisions are discrete

PIPS-S: Stochastic MIP problems

- Power systems optimization project at Argonne
 - Integer second-stage decisions
 - Stochasticity from wind generation
 - Solution via branch-and-bound
 - Solve root using parallel IPM solver PIPS
 - Solve nodes using parallel dual simplex solver PIPS-S
- Lubin, Petra *et al.* (2011)



PIPS-S: Stochastic MIP problems

Convenient to permute the LP thus:

$$\begin{array}{llllllllll}
 \text{minimize} & \mathbf{c}_1^T \mathbf{x}_1 & + & \mathbf{c}_2^T \mathbf{x}_2 & + & \dots & + & \mathbf{c}_N^T \mathbf{x}_N & + & \mathbf{c}_0^T \mathbf{x}_0 \\
 \text{subject to} & W_1 \mathbf{x}_1 & & & & & & & & + T_1 \mathbf{x}_0 = \mathbf{b}_1 \\
 & & & W_2 \mathbf{x}_2 & & & & & & + T_2 \mathbf{x}_0 = \mathbf{b}_2 \\
 & & & & & \ddots & & & & \vdots \\
 & & & & & & & W_N \mathbf{x}_N & + & T_N \mathbf{x}_0 = \mathbf{b}_N \\
 & & & & & & & & & A \mathbf{x}_0 = \mathbf{b}_0 \\
 & \mathbf{x}_1 \geq \mathbf{0} & & \mathbf{x}_2 \geq \mathbf{0} & & \dots & & \mathbf{x}_N \geq \mathbf{0} & & \mathbf{x}_0 \geq \mathbf{0}
 \end{array}$$

PIPS-S: Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

$$B = \begin{bmatrix} W_1^B & & T_1^B \\ & \ddots & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$

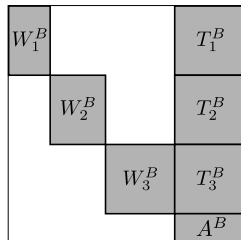
- W_i^B are columns corresponding to n_i^B basic variables in scenario i

- $\begin{bmatrix} T_1^B \\ \vdots \\ T_N^B \\ A^B \end{bmatrix}$ are columns corresponding to n_0^B basic first stage decisions

PIPS-S: Exploiting problem structure

- Inversion of the basis matrix B is key to revised simplex efficiency
- For column-linked BALP problems

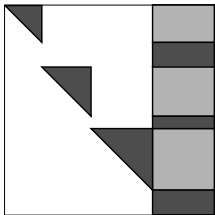
$$B = \begin{bmatrix} W_1^B & & & T_1^B \\ & \ddots & & \vdots \\ & & W_N^B & T_N^B \\ & & & A^B \end{bmatrix}$$



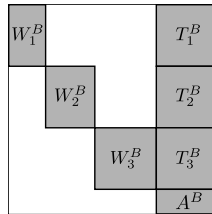
- B is nonsingular so
 - W_i^B are “tall”: full column rank
 - $[W_i^B \ T_i^B]$ are “wide”: full row rank
 - A^B is “wide”: full row rank
- Scope for parallel inversion is immediate and well known

PIPS-S: Exploiting problem structure

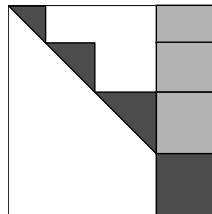
- Eliminate sub-diagonal entries in each W_i^B (independently)



- Accumulate non-pivoted rows from the W_i^B with A^B and complete elimination



- Apply elimination operations to each T_i^B (independently)



Scope for parallelism

- Parallel Gaussian elimination yields **block LU** decomposition of B
- Scope for parallelism in block forward and block backward substitution
- Scope for parallelism in PRICE

Implementation

- Distribute problem data over processes
- Perform data-parallel BTRAN, FTRAN and PRICE over processes
- Used MPI

On Fusion cluster: Performance relative to c1p

Dimension	Cores	Storm	SSN	UC12	UC24
$m + n = O(10^6)$	1	0.34	0.22	0.17	0.08
	32	8.5	6.5	2.4	0.7
$m + n = O(10^7)$	256	299	45	67	68

On Blue Gene

- Instance of UC12
- $m + n = O(10^8)$
- Requires 1 TB of RAM
- Runs from an advanced basis

Cores	Iterations	Time (h)	Iter/sec
1024	Exceeded execution time limit		
2048	82,638	6.14	3.74
4096	75,732	5.03	4.18
8192	86,439	4.67	5.14

Task parallelism for general LP problems

Overview

- Written in C++ to study parallel simplex
- Dual simplex with steepest edge and BFRT
- Forrest-Tomlin update
 - **complex** and **inherently serial**
 - **efficient** and **numerically stable**

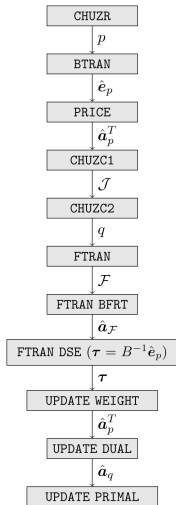
Concept

- Exploit limited task and data parallelism in standard dual RSM iterations (sip)
- Exploit greater task and data parallelism via minor iterations of dual SSM (pami)
- Test-bed for research
- Work-horse for consultancy

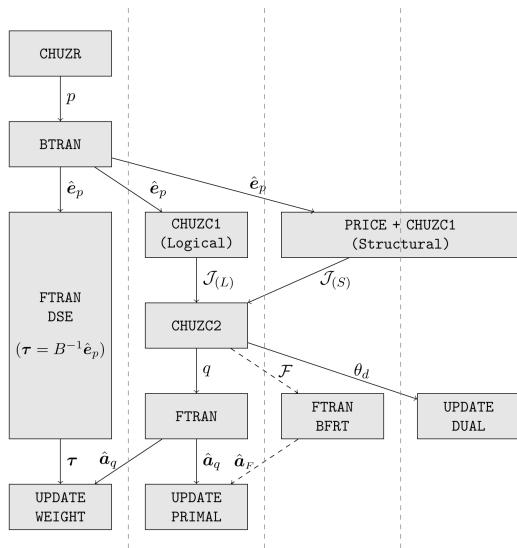
Huangfu, H and Galabova (2011–date)

h₂gmp: Single iteration parallelism with sip option

- Computational components appear sequential
- Each has highly-tuned sparsity-exploiting serial implementation
- Exploit “slack” in data dependencies



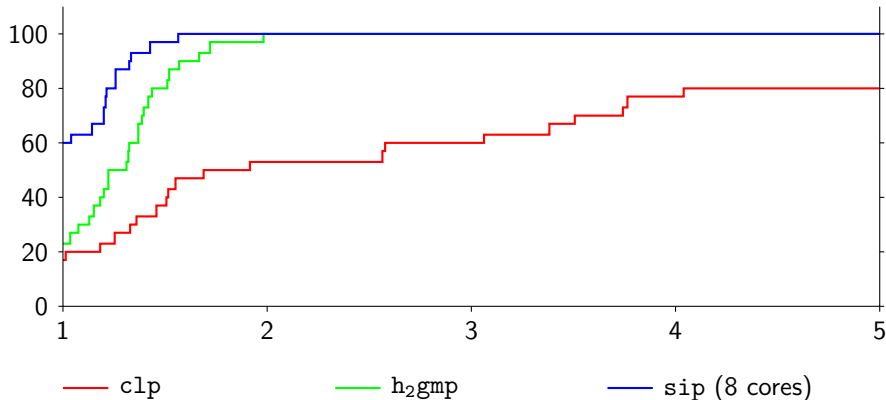
h_2gmp : Single iteration parallelism with sip option



- Parallel PRICE to form $\hat{\mathbf{a}}_p^T = \boldsymbol{\pi}_p^T N$
- Other computational components serial
- Overlap any independent calculations
- Only four worthwhile threads unless $n \gg m$ so PRICE dominates
- More than Bixby and Martin (2000)
- Better than Forrest (2012)

Huangfu and H (2014)

h_2gmp : clp vs h_2gmp vs sip



Performance on spectrum of 30 significant LP test problems

- sip on 8 cores is 1.15 times faster than h_2gmp
- h_2gmp (sip on 8 cores) is 2.29 (2.64) times faster than clp

h2gmp: Multiple iteration parallelism with pami option

- Perform standard dual simplex minor iterations for rows in set $\mathcal{P}$ ($|\mathcal{P}| \ll m$)
- Suggested by Rosander (1975) but never implemented efficiently *in serial*

	$\mathcal{N}$	RHS
$\mathcal{B}$	$\hat{\mathbf{a}}_{\mathcal{P}}^T$	$\hat{\mathbf{b}}$
		$\hat{b}_{\mathcal{P}}$
	$\hat{\mathbf{c}}_N^T$	

- Task-parallel multiple BTRAN to form $\boldsymbol{\pi}_{\mathcal{P}} = B^{-1} \mathbf{e}_{\mathcal{P}}$
- Data-parallel PRICE to form $\hat{\mathbf{a}}_{\mathcal{P}}^T$ (as required)
- Task-parallel multiple FTRAN for primal, dual and weight updates

Huangfu and H (2011–2014)
COAP best paper prize (2015)

h2gmp: Performance and reliability

Extended testing using 159 test problems

- 98 Netlib
- 16 Kennington
- 4 Industrial
- 41 [Mittelman](#)

Exclude 7 which are “hard”

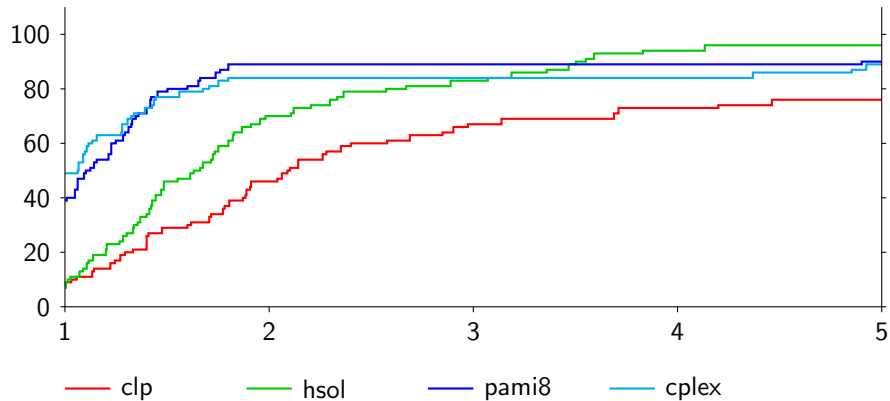
Performance

Benchmark against c1p (v1.16) and cplex (v12.5)

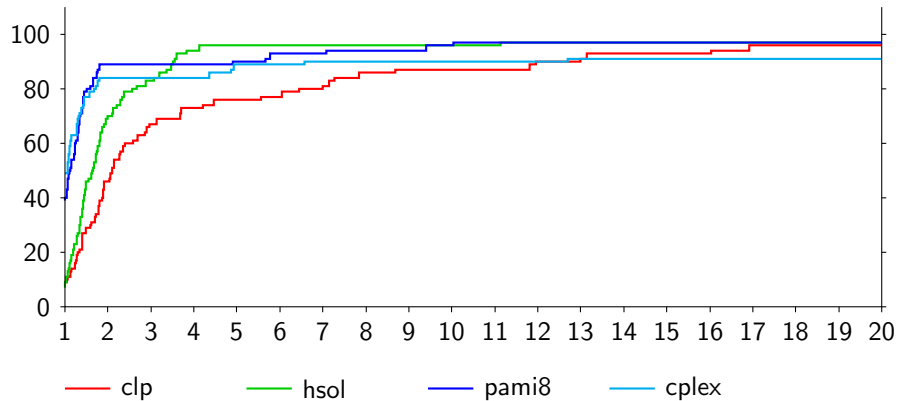
- Dual simplex
- No presolve
- No crash

Ignore results for 82 LPs with minimum solution time below 0.1s

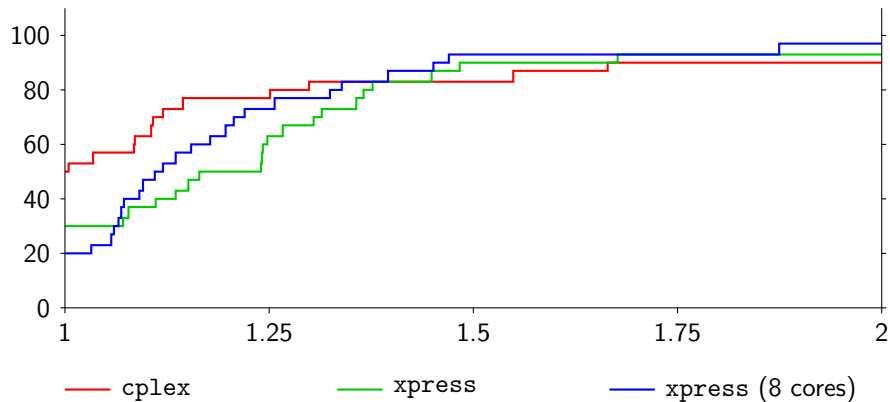
h₂gmp: Performance



h₂gmp: Reliability



h₂gmp: Impact



- pami ideas incorporated in [FICO Xpress](#) (Huangfu 2014)
- Made Xpress simplex solver the fastest commercial code

Interior point methods

Interior point methods: Traditional

Replace $\mathbf{x} \geq 0$ by **log barrier** function and solve

$$\text{maximize } f = \mathbf{c}^T \mathbf{x} + \mu \sum_{j=1}^n \ln(x_j) \quad \text{such that} \quad A\mathbf{x} = \mathbf{b}$$

- For small μ this has the same solution as the LP
- The first order optimality conditions are

$$A\mathbf{x} = \mathbf{b}; \quad A^T \mathbf{y} + \mathbf{s} = \mathbf{c}; \quad X\mathbf{S} = \mu \mathbf{e}; \quad (\mathbf{x}, \mathbf{s}) \geq \mathbf{0}$$

where X and S are diagonal matrices with entries from the vectors $\mathbf{s}$ and $\mathbf{x}$

- Interior point methods apply Newton's method to the optimality conditions
 - Not solved exactly for a given μ
 - Solved approximately for a decreasing sequence of values of μ
 - Move through *interior* of K

Interior point methods: Traditional

- Apply Newton's method to

$$A\mathbf{x} = \mathbf{b}; \quad A^T \mathbf{y} + \mathbf{s} = \mathbf{c}; \quad X\mathbf{S} = \mu \mathbf{e}; \quad (\mathbf{x}, \mathbf{s}) \geq \mathbf{0}$$

for a decreasing sequence of values of μ

- Perform a small number of (expensive) iterations: each solves

$$\begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{y} \\ \Delta \mathbf{s} \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix}$$

where $\Delta \mathbf{x}$, $\Delta \mathbf{y}$ and $\Delta \mathbf{s}$ are steps in the primal and dual variables

- Eliminate $\Delta \mathbf{s} = X^{-1}(\xi_\mu - S\Delta \mathbf{x})$ to give

$$\begin{bmatrix} -\Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{d} \end{bmatrix} \iff G\Delta \mathbf{y} = \mathbf{h}$$

where $\Theta = XS^{-1}$ and $G = A\Theta A^T$

Interior point methods: Traditional

Forming the **Cholesky** decomposition $G = LL^T$ and perform triangular solves with L

- $G = A\Theta A^T = \sum_j \theta_j \mathbf{a}_j \mathbf{a}_j^T$ is generally sparse

So long as dense columns of A are treated carefully

- Much effort has gone into developing efficient serial Cholesky codes
- Parallel codes exist: notably for (nested) block structured problems
OOPS solved a QP with 10^9 variables

Gondzio and Grothey (2006)

- **Disadvantage:** L can fill-in
Cholesky can be prohibitively expensive for large n

Interior point methods: Matrix-free

Alternative approach to Cholesky: solve $G\Delta\mathbf{y} = \mathbf{h}$ using an **iterative** method

Conjugate gradient method (CG) for $A\mathbf{x} = \mathbf{b}$ with A **symmetric** and **positive definite**

Given some estimate $\mathbf{x}^{(1)}$, let $\mathbf{s}^{(1)} = \mathbf{r}^{(1)} = \mathbf{b} - A\mathbf{x}^{(1)}$

Then, for some tolerance ϵ , repeat for $k = 1, 2, \dots$

- ① Form $\mathbf{w} = A\mathbf{s}^{(k)}$
- ② Compute the **step length** $\alpha^{(k)} = \mathbf{r}^{(k)T} \mathbf{s}^{(k)} / \mathbf{w}^T \mathbf{s}^{(k)}$
- ③ Update the **iterate** $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha^{(k)} \mathbf{s}^{(k)}$
- ④ Update the **residual** $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha^{(k)} \mathbf{w}$
- ⑤ If $\|\mathbf{r}^{(k+1)}\|_2 \leq \epsilon$ then stop
- ⑥ Compute $\beta^{(k)} = \|\mathbf{r}^{(k+1)}\|_2^2 / \|\mathbf{r}^{(k)}\|_2^2$
- ⑦ Update the **search direction** $\mathbf{s}^{(k+1)} = \mathbf{r}^{(k+1)} + \beta^{(k)} \mathbf{s}^{(k)}$

Convergence depends on clustering of eigenvalues of A

Interior point methods: Matrix-free

- Matrix G is **ill-conditioned** so CG doesn't converge
- **Preconditioned gradient method** (PCG) applies CG to $(P^{-1}G)\mathbf{x} = (P^{-1}\mathbf{b})$ using **preconditioner** P
 - Aim to cluster eigenvalues of $P^{-1}G$
 - Ideal preconditioner is $P = G$: PCG terminates in one iteration!
 - Representation of P^{-1} must be cheap to form and apply
- Main cost of PCG iteration is forming $\mathbf{w} = P^{-1}G\mathbf{s}$
 - Don't form $P^{-1}G$!
 - Don't even form G !
 - Form $\mathbf{w}$ by solving $P\mathbf{w} = \mathbf{r}$ where $\mathbf{r} = G\mathbf{s} = A[\Theta(A^T\mathbf{s})]$
 - Only requires matrix-vector products with A and A^T , and scaling with Θ

Alternative approach to Cholesky: solve $G\Delta\mathbf{y} = \mathbf{h}$ using PCG

- For preconditioner, consider $G = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_L & \\ & S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix}$ where
 - $L = \begin{bmatrix} L_{11} \\ L_{21} \end{bmatrix}$ contains the first k columns of the Cholesky factor of G
 - D_L is a diagonal matrix formed by the k largest pivots of G
 - S is the Schur complement after k pivots
- Precondition $G\Delta\mathbf{y} = \mathbf{h}$ using $P = \begin{bmatrix} L_{11} & \\ L_{21} & I \end{bmatrix} \begin{bmatrix} D_L & \\ & D_S \end{bmatrix} \begin{bmatrix} L_{11}^T & L_{21}^T \\ & I \end{bmatrix}$ where
 - S_D is the diagonal of S
 - Avoids computing S or even G !

Gondzio (2009)

Interior point methods: Matrix-free

- Can solve problems intractable using direct methods
- Only requires “oracle” returning $\mathbf{y} = \mathbf{A}\mathbf{x}$ Gondzio *et al.* (2014)
- Matrix-free IPM beats first order methods on speed and reliability for
 - ℓ_1 -regularized sparse least-squares: $n = O(10^{12})$
 - ℓ_1 -regularized logistic regression: $n = O(10^4\text{--}10^7)$
- How?
 - Preconditioner P is **diagonal**
 - $\mathbf{A}\Theta\mathbf{A}^T$ is near-diagonal!
- Says much about the “difficulty” of such problems! Fountoulakis and Gondzio (2014–2016)

Disadvantage: Not useful for all problems!

Interior point methods: Quadratic programming (QP)

Quadratic programming (QP)

$$\text{minimize } f = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} \quad \text{subject to } A \mathbf{x} = \mathbf{b} \quad \mathbf{x} \geq \mathbf{0}$$

where Q is symmetric and, for convex QP, Q is positive definite

- As for LP, replace $\mathbf{x} \geq 0$ by **log barrier** function and solve

$$\text{maximize } f = \frac{1}{2} \mathbf{x}^T Q \mathbf{x} + \mathbf{c}^T \mathbf{x} + \mu \sum_{j=1}^n \ln(x_j) \quad \text{such that } A \mathbf{x} = \mathbf{b}$$

- Perform a small number of (expensive) iterations: each solves

$$\begin{bmatrix} A & 0 & 0 \\ -Q & A^T & I \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{y} \\ \Delta \mathbf{s} \end{bmatrix} = \begin{bmatrix} \xi_p \\ \xi_d \\ \xi_\mu \end{bmatrix} \iff \begin{bmatrix} -Q - \Theta^{-1} & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{x} \\ \Delta \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{d} \end{bmatrix}$$

$$\iff G \Delta \mathbf{y} = \mathbf{h} \text{ where } \Theta = X S^{-1} \text{ and } G = A(Q + \Theta^{-1})^{-1} A^T$$

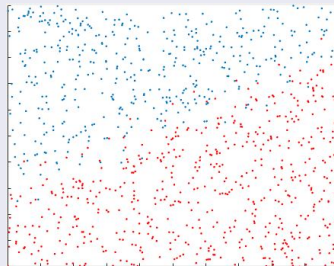
Application in machine learning

Support vector machine

- **Data:** “Training set” of n points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$
 $y_i \in \{-1, 1\}$ indicates the class of $\mathbf{x}_i \in \mathbb{R}^p$
- A **linear support vector machine** classifies a new point $\mathbf{x} \in \mathbb{R}^p$ by

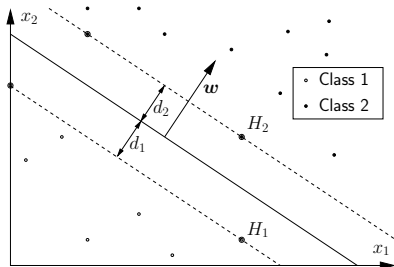
$$\mathbf{x} \mapsto \text{sign}(\mathbf{w}^T \mathbf{x} + w_0).$$

- **Problem:** Find the best **separating hyperplane** given by $\mathbf{w}$ and w_0



Example: $p = 2$; $n = 1000$;
 $y_i \in \{\text{red}, \text{blue}\}$

When classes are fully separable



- **Aim:** Find $\mathbf{w}$ and w_0 so, for all i

$$\begin{aligned} \mathbf{x}_i^T \mathbf{w} + w_0 &\geq 1 & \text{for } y_i = 1 \\ \mathbf{x}_i^T \mathbf{w} + w_0 &\leq -1 & \text{for } y_i = -1 \end{aligned}$$

- Equivalently $y_i(\mathbf{x}_i^T \mathbf{w} + w_0) - 1 \geq 0 \quad \forall i$
- Closest points (support vectors) lie on hyperplanes H_1 and H_2

$$\begin{aligned} \mathbf{x}_i^T \mathbf{w} + w_0 &= 1 & \text{for } H_1 \\ \mathbf{x}_i^T \mathbf{w} + w_0 &= -1 & \text{for } H_2 \end{aligned}$$

Best separating hyperplane has greatest **margin** $d = d_1 = d_2 = 1/\|\mathbf{w}\|_2$

Application: Machine learning

- Maximize margin by maximizing $1/\|\mathbf{w}\|_2$ subject to $y_i(\mathbf{x}_i^T \mathbf{w} + w_0) - 1 \geq 0 \forall i$
- Equivalently, solve

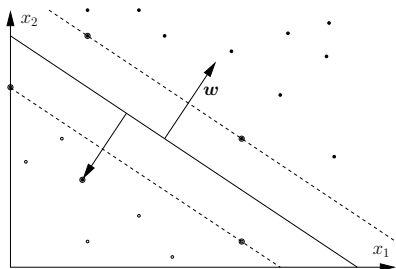
$$\text{minimize } f = \|\mathbf{w}\|_2^2 \quad \text{subject to } y_i(\mathbf{x}_i^T \mathbf{w} + w_0) - 1 \geq 0 \forall i$$

- Introduce
 - $X \in \mathbb{R}^{n \times p}$ as matrix of row vectors $\mathbf{x}_i^T$
 - $\mathbf{y} \in \{-1, 1\}^n$ as vector of class indicators y_i
 - Y as the diagonal matrix with entries y_i for $i = 1, \dots, n$
- Problem to solve is the **quadratic programming** (QP) problem

$$\text{minimize } f = \mathbf{w}^T \mathbf{w} \quad \text{subject to } -YX\mathbf{w} + \mathbf{y}w_0 \leq -\mathbf{e}$$

Application: Machine learning

When classes are not fully separable



- **Aim:** Find $\mathbf{w}$, w_0 and **slacks** $\mathbf{s} \geq \mathbf{0}$ so, for all i

$$\begin{aligned} \mathbf{x}_i^T \mathbf{w} + w_0 &\geq 1 - s_i & \text{for } y_i = 1 \\ \mathbf{x}_i^T \mathbf{w} + w_0 &\leq -1 + s_i & \text{for } y_i = -1 \end{aligned}$$

- Equivalently $y_i(\mathbf{x}_i^T \mathbf{w} + w_0) - 1 + s_i \geq 0 \quad \forall i$
- Have *two* objectives
 - Greatest margin
 - Least slacks

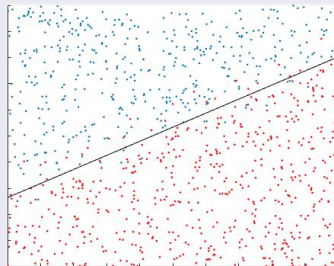
Trade off the two objectives using a multiplier $C > 0$ and solve the QP

$$\text{minimize } f = \mathbf{w}^T \mathbf{w} + \frac{C}{n} \mathbf{e}^T \mathbf{s} \quad \text{subject to} \quad -YX\mathbf{w} + \mathbf{y}w_0 - \mathbf{s} \leq -\mathbf{e} \quad \mathbf{s} \geq \mathbf{0}$$

MATLAB exercise

- **Data:** Training set of $n = 1000$ points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^2$
- **Solve:**

$$\begin{aligned} \text{minimize} \quad & f = \mathbf{w}^T \mathbf{l} \mathbf{w} + \frac{C}{n} \mathbf{e}^T \mathbf{s} \\ \text{subject to} \quad & -YX\mathbf{w} + \mathbf{y}w_0 - \mathbf{s} \leq -\mathbf{e} \\ & \mathbf{s} \geq \mathbf{0} \end{aligned}$$



High performance solution of linear optimization problems: The future

High performance solution of linear optimization problems: The future

“Optimization is being revolutionized by its interactions with machine learning and data analysis.”

- General linear optimization certainly isn't!
- Still waiting for first order methods to solve meaningful general LP problems
- Can the “fast approximate solution” philosophy be applied to (other) problems of massive practical importance?
- Speech recognition, dating apps and film preference analysis may be cool...
- ... however, we still have to
 - Eat
 - Travel
 - Manufacture things
 - Generate and transmit power
 - Make sure the data from your phone reaches a server for the ML to actually happen!

High performance solution of linear optimization problems: The future

“Optimization is being revolutionized by its interactions with machine learning and data analysis.”

- No revolution, but some *influence*
 - Fast approximate solution may be more valuable
Video stabilization for YouTube (Google)
 - Methods must be energy efficient on a phone
 - Need to be open to algorithms which are (not too!) inefficient in serial but scale naturally on multi-core
- Example: augmented Lagrangian methods
 - Useful on some ML problems
 - Fail or are uselessly inefficient on general LP problems
 - Clever “Idiot” crash (Forrest) with augmented Lagrangian and coordinate descent flavour
 - Generally useless as LP solver
 - Works well on linearizations of quadratic assignment problems
 - Can it be improved?

(Galabova and H 2016–date)