Efficient Problem Reductions in Linear Algebra

Erich Kaltofen North Carolina State University www.kaltofen.net



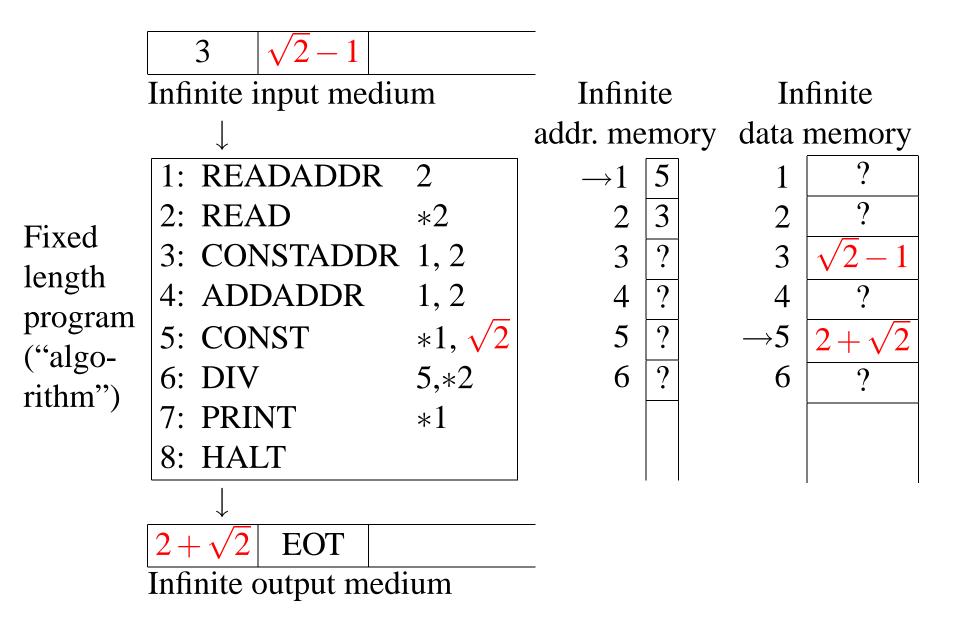
A simple example

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}^{-1} = \begin{bmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix} \in \mathbb{F}^{3n \times 3n}$$
$$\implies \text{MATMULT}_{\text{arithm.}}(n) = O(\text{MATINV}_{\text{arithm.}}(n)).$$

Further examples:

LUDECOMP(n) = O(MATMULT(n)) [Bunch and Hopcroft 1974] MATMULT(n) = O(DET(n)) [Baur and Strassen 1983] CHARPOLY $(n) = MATMULT(n)^{1+o(1)}$ [Keller-Gehrig 1985] FROBFORM $(n) = MATMULT(n)^{1+o(1)}$ [Giesbrecht 1992]

Model: algebraic RAM over $\mathbb{F} = \mathbb{Q}(\sqrt{2})$



– computes a function from $D \longrightarrow E$ where D is **infinite**

- can be programmed as a C++ template function
- defines arithmetic time and space complexities

Ambiguity through randomization: RANDOM{ADDR} *i*, *j* The operand *j* points to an address which is the cardinality of $S \subset \mathbb{F}$ from which random elements are sampled.

- Monte Carlo: "always fast, probably correct". Examples: isprime

Lemma [*DeMillo&Lipton'78, Schwartz/Zippel'79*] Let $f, g \in \mathbb{F}[x_1, ..., x_n], f \neq g, S \subseteq \mathbb{F}$.

Probability $(f(a_1, \dots, a_n) \neq g(a_1, \dots, a_n) \mid a_i \in S)$ $\geq 1 - \max\{\deg(f), \deg(g)\} / \operatorname{cardinality}(S)$

sparse polynomial interpolation, factorization, minimal polynomial of a sparse matrix Do we exactly know what the algorithm computes? E.g., in the presence of floating point arithmetic?

-Las Vegas: "always correct, probably fast". Examples: polynomial factorization in $\mathbb{Z}_p[x]$, where $p \gg 2$. Determinant of a sparse matrix **Theorem** [Strassen '73; Baur and Strassen '83; see Giesbrecht '92] Suppose you have a Monte Carlo randomized algorithm on a algebraic random access machine that can compute the determinant of an $n \times n$ matrix in D(n) arithmetic operations.

Then you have a Monte Carlo randomized algorithm on a random access machine that can multiply two $n \times n$ matrices in O(D(n)) arithmetic operations.

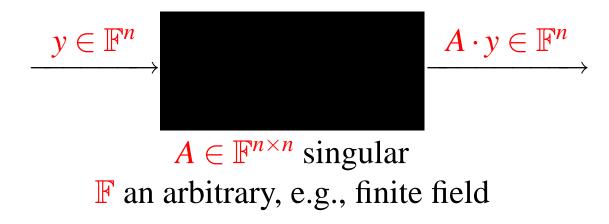
No proof is known for Las Vegas or deterministic algorithms.

Proof requires

- Eliminate superfluous tests on algebraic elements by random evaluation [DeMillo&Lipton '78/Schwartz/Zippel '79]
- MATINV \leq DET: reverse mode of automatic differentiation and [Baur&Strassen '83]: $(A^{-1})_{i,j} = \frac{(-1)^{i+j}}{\det(A)} \cdot \frac{\partial \det(A)}{\partial a_{j,i}}$
- $-C(C+I) \le MATINV: C(C+I) = (C^{-1} (C+I)^{-1})^{-1} \text{ and entries}$ in C(C+I) are algebraically independent [Strassen '73]
- Eliminate divisions from straight-line program for C(C+I)[Strassen/Ungar '73]
- $-MATMULT \le \text{division-free-} C(C+I): \begin{bmatrix} 0 & A \\ 0 & B \end{bmatrix} \cdot \begin{bmatrix} I & A \\ 0 & I+B \end{bmatrix} = \begin{bmatrix} 0 & A+AB \\ 0 & B+B^2 \end{bmatrix}$

Black box linear algebra

The black box model of a matrix



Perform linear algebra operations, e.g., $A^{-1}b$ [Wiedemann 86] with

O(n)black box calls and $n^2(\log n)^{O(1)}$ arithmetic operations in \mathbb{F} andO(n)intermediate storage for field elements

LINSOLVE0: Given black box $A \in \mathbb{F}^{n \times n}$, compute $w \neq 0$ such that Aw = 0.

Used in sieve-based integer factoring algorithms,

Las Vegas singularity and Monte-Carlo non-singularity tests.

NONSINGULAR
$$\leq$$
 LINSOLVE0: For $Ax = b$ solve $\begin{bmatrix} A & b \\ 0^{1 \times n} & 0 \end{bmatrix} w = 0$
and compute $x = \frac{1}{w_{n+1}} \begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix}$.

Harder (?) problem LINSOLVE1: Given black box $A \in \mathbb{F}^{n \times n}$ (possibly singular) and b, compute x such that Ax = b.

Results from Kaltofen & Saunders 1991

Random sampling in the nullspace is equivalent to LINSOLVE1:

$\texttt{RANDOM-LINSOLVE0}{\leq}L\texttt{INSOLVE1}$

select a random vector y and solve Ax = b for b = Ay using LINSOLVE1. Return w = x - y. Note that y is known to LINSOLVE1 only up to a shift by any

nullspace vector.

LINSOLVE1 \leq RANDOM-LINSOLVE0 Solve $\begin{bmatrix} A & b \end{bmatrix} w = 0$ by random-LINSOLVE0. With probability $1 - 1/|\mathbb{F}|$ we have $w_{n+1} \neq 0$: consider a basis $w = \sum_{i=1}^{r} c_i w^{[i]}$ and $w_{n+1}^{[1]} \neq 0$. For any choice of c_2, \ldots, c_r only one c_1 yields $w_{n+1} = 0$. Results from Kaltofen & Saunders 1991 continued

LINSOLVE1 \leq LINSOLVE0+RANK For $r = \operatorname{rank}(A)$ use a preconditioner $\begin{bmatrix} \widetilde{A}^{[r]} & \widetilde{A}^{[1,2]} \\ \widetilde{A}^{[2,1]} & \widetilde{A}^{[2,2]} \end{bmatrix} = B^{[1]} \cdot A \cdot B^{[2]}$

such that the $r \times r$ top-left submatrix $\widetilde{A}^{[r]}$ is non-singular.

Note: $B^{[i]}$ can be sparse, Toeplitz, or "butterfly" matrices [Chen et al. 2002]; we have a black box algorithm for $\widetilde{A}^{[r]}$.

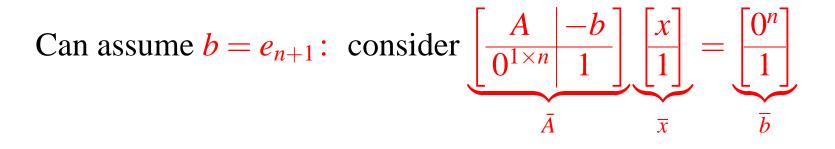
Solve the $r \times r$ non-singular system $\widetilde{A}^{[r]}z^{[r]} = \widetilde{b}^{[r]}$ where $B^{[1]}b = \begin{bmatrix} \widetilde{b}^{[r]} \\ \vdots \end{bmatrix}$ and return $x = B^{[2]} \begin{bmatrix} z^{[r]} \\ 0 \end{bmatrix}$. When is PROBLEM1 <= PROBLEM2?

Both PROBLEM2 and the black box matrix act as oracles.
 E.g., PROBLEM2 is solved for a preconditioned black box matrix.

- The algorithm for PROBLEM2 could be for a fixed or a generic coefficient field. E.g., PROBLEM2 is solved over a field extension.

- -O(1) versus $(\log n)^{O(1)}$ deceleration.
 - E.g., PROBLEM2 is called log *n* times or on matrices of bigger dimensions.
 - The black box matrix is called O(n) times.
 - Note: LINSOLVE1 \leq PRECONDNIL+Wiedemann/Lanczos

$LinSolve1 \leq LinSolve0$



Why
$$\widetilde{A} = \overline{A} - \overline{b}c^T = [\overline{A}_{*,1} - c_1\overline{b} \mid \dots \mid \overline{A}_{*,n+1} - c_{n+1}\overline{b}],$$

 c_j random, $\widetilde{A} \widetilde{w} = 0$ does not yield $\overline{x} = 1/(c^T \widetilde{w}) \cdot \widetilde{w}$ for all \overline{A} :

Suppose $\overline{A} = [0 \mid 0 \mid \overline{A}_{*,3} \mid ...]$ and the algorithm for LINSOLVE0 first checks if two columns are dependent. Then for any choice of c_j we always get $c^T \widetilde{w} = 0$.

Must hide
$$\overline{b}$$
 better: $\widetilde{A} = \overline{A}B - \overline{b}c^T$ or $\widetilde{A} = (\overline{A} - \overline{b}c^T)B$

The curse of soft-O

 $\log_2 n < n^{1/3 - 1/5}$ for $n \ge n_0$: $n_0 \ge 10^{12}$.

 $\log_2 n < n^{1/5 - 1/7}$ for $n \ge n_0$: $n_0 \ge 10^{37}$.

 $(\log_2 n)^2 < n^{1/2}$ for $n \ge n_0$: $n_0 \ge 2^{16} = 65536$.

Reductions and bit complexity

By CRA, MATMULT has bit complexity $\leq n^{\omega} (\log ||AB||)^{1+o(1)}$, where ω is the exponent of the arithmetic complexity.

DET: $\leq n^{3.19} (\log ||A||)^{1+o(1)}$ [Eberly, Giesbrecht, Villard 2000] $\leq n^{3.03} (\log ||A||)^{1+o(1)}$ [Kaltofen 1995/2000] $\leq n^{2.70} (\log ||A||)^{1+o(1)}$ [Kaltofen, Villard 2001] $\leq n^{2.38} (\log ||A||)^{1+o(1)}$ [Storjohann 2002]

The argonautical quest: How do preserve bit complexity when computing high degree objects?