Algebraic approach to exact algorithms, Part II: Fast Matrix Multiplication

Łukasz Kowalik

University of Warsaw

ADFOCS, Saarbrücken, August 2013

(Square) matrix multiplication

Problem

Given two matrices $n \times n$: A and B.
Compute the matrix $C = A \cdot B$.

Naive algorithm

$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$.
Time: $O(n^3)$ arithmetical operations.

W.l.o.g. $n = 2^k$.
Let us partition **A**, **B**, **C** into blocks of size $(n/2) \times (n/2)$:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

Then

$$\mathbf{C} = \left[\begin{array}{c|c} \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} & \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\ \hline \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} & \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2} \end{array} \right]$$

We get the recurrence $T(n) = 8T(n/2) + O(n^2)$, hence $T(n) = O(n^3)$.
(The last level dominates, it has $8^{\log_2 n} = n^3$ nodes.)

Matrix multiplication: Divide and conquer (2)

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{1,1} & \mathbf{A}_{1,2} \\ \mathbf{A}_{2,1} & \mathbf{A}_{2,2} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{1,1} & \mathbf{B}_{1,2} \\ \mathbf{B}_{2,1} & \mathbf{B}_{2,2} \end{bmatrix}$$

A new approach (Strassen 1969):

$$\mathbf{M}_1 := (\mathbf{A}_{1,1} + \mathbf{A}_{2,2})(\mathbf{B}_{1,1} + \mathbf{B}_{2,2}) \qquad \mathbf{M}_2 := (\mathbf{A}_{2,1} + \mathbf{A}_{2,2})\mathbf{B}_{1,1}$$
$$\mathbf{M}_3 := \mathbf{A}_{1,1}(\mathbf{B}_{1,2} - \mathbf{B}_{2,2}) \qquad \mathbf{M}_4 := \mathbf{A}_{2,2}(\mathbf{B}_{2,1} - \mathbf{B}_{1,1})$$
$$\mathbf{M}_5 := (\mathbf{A}_{1,1} + \mathbf{A}_{1,2})\mathbf{B}_{2,2} \qquad \mathbf{M}_6 := (\mathbf{A}_{2,1} - \mathbf{A}_{1,1})(\mathbf{B}_{1,1} + \mathbf{B}_{1,2})$$
$$\mathbf{M}_7 := (\mathbf{A}_{1,2} - \mathbf{A}_{2,2})(\mathbf{B}_{2,1} + \mathbf{B}_{2,2}).$$

Then:

$$\mathbf{C} = \left[\begin{array}{c|c} \mathbf{A}_{1,1}\mathbf{B}_{1,1} + \mathbf{A}_{1,2}\mathbf{B}_{2,1} & \mathbf{A}_{1,1}\mathbf{B}_{1,2} + \mathbf{A}_{1,2}\mathbf{B}_{2,2} \\ \hline \mathbf{A}_{2,1}\mathbf{B}_{1,1} + \mathbf{A}_{2,2}\mathbf{B}_{2,1} & \mathbf{A}_{2,1}\mathbf{B}_{1,2} + \mathbf{A}_{2,2}\mathbf{B}_{2,2} \end{array}\right]$$
$$= \left[\begin{array}{c|c} \mathbf{M}_1 + \mathbf{M}_4 - \mathbf{M}_5 + \mathbf{M}_7 & \mathbf{M}_3 + \mathbf{M}_5 \\ \hline \mathbf{M}_2 + \mathbf{M}_4 & \mathbf{M}_1 - \mathbf{M}_2 + \mathbf{M}_3 + \mathbf{M}_6 \end{array}\right]$$

We get the recurrence $T(n) = 7T(n/2) + O(n^2)$ hence $T(n) = O(7^{\log_2 n}) = O(n^{\log_2 7}) = O(n^{2.81})$.

State-of-art algorithms

ω constant

$\omega = \inf\{p \ : \ \forall\epsilon > 0$ one can multiply two $n \times n$ matrices in $O(n^{p+\epsilon})$ time$\}$

Trivial lower bound: $\omega \geq 2$.

Theorem (Coppersmith and Winograd 1990)

$\omega \leq 2.376$.

Theorem (Stothers 2010)

$\omega \leq 2.3736$.

Theorem (Vassilevska-Williams 2011)

$\omega \leq 2.3727$.

A standard exercise

Problem

Given a directed/undirected n-vertex graph G

- find a triangle in G, if it exists.
- Compute the number of triangles in G

A standard exercise

Problem

Given a directed/undirected n-vertex graph G

- find a triangle in G, if it exists.
- Compute the number of triangles in G

Lemma

Let $\mathbf{A}$ be the adjacency matrix of an n-vertex graph G (directed or undirected). Let $k \in \mathbb{N}_{>0}$. Then for every $i, j = 1, \ldots, n$ the entry $\mathbf{A}^k_{i,j}$ is the number of length k walks from i to j.

Proof: Easy induction on k.

Corollary

Both problems above can be solved in $O(n^\omega)$ time.

MAX-2-SAT

Problem MAX-2-SAT

Given a 2-CNF formula ϕ with n variables, find an assignment which maximizes the number of satisfied clauses.

Example: $(x_1 \vee \neg x_2) \wedge (x_3 \vee x_2) \wedge (x_2 \vee \neg x_5) \wedge \cdots$

MAX-2-SAT

Problem MAX-2-SAT

Given a 2-CNF formula ϕ with n variables, find an assignment which maximizes the number of satisfied clauses.

Example: $(x_1 \vee \neg x_2) \wedge (x_3 \vee x_2) \wedge (x_2 \vee \neg x_5) \wedge \cdots$
In what follows we deal with the equivalent (up to a #clauses factor) problem:

MAX-2-SAT, decision version

Input: A 2-CNF formula ϕ with n variables, a number $k \in \mathbb{N}$.
Question: Is there an assignment which satisfies exactly k clauses?

MAX-2-SAT

Problem MAX-2-SAT

Given a 2-CNF formula ϕ with n variables, find an assignment which maximizes the number of satisfied clauses.

Example: $(x_1 \vee \neg x_2) \wedge (x_3 \vee x_2) \wedge (x_2 \vee \neg x_5) \wedge \cdots$
In what follows we deal with the equivalent (up to a #clauses factor) problem:

MAX-2-SAT, decision version

Input: A 2-CNF formula ϕ with n variables, a number $k \in \mathbb{N}$.
Question: Is there an assignment which satisfies exactly k clauses?

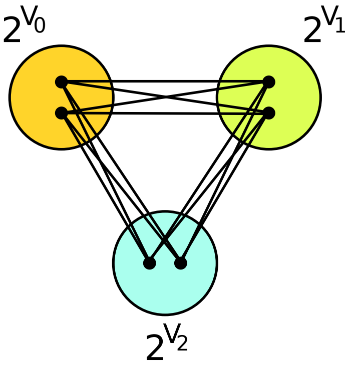
Complexity

MAX-2-SAT is NP-complete.
The naive algorithm works in $O^*(2^n)$ time.
Question: Can we do better? E.g. $O(1.9^n)$?

MAX-2-SAT (Williams 2004)

We construct an undirected graph G on $O(2^{n/3})$ vertices.

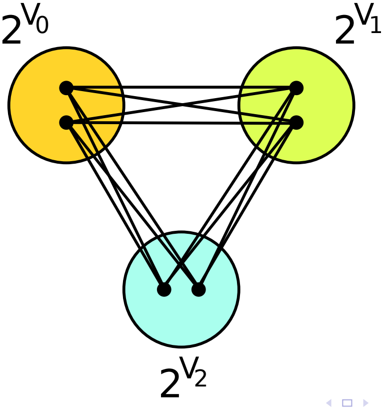
- Let us fix an arbitrary partition $V = V_0 \cup V_1 \cup V_2$ into three equal parts (as equal as possible...).
- $V(G)$ is the set of all assignments $v_i : V_i \to \{0,1\}$ for $i = 0, 1, 2$.
- For every $v \in V_i$, $w \in V_{(i+1) \bmod 3}$ graph G contains the edge vw.



MAX-2-SAT (Williams 2004)

Solution idea

- We assign weights to edges so that the weight of the *vwu* triangle in G equals the number of clauses satisfied with the assignment (v, w, u).
- Then it is sufficient to check if there is a triangle of weight k in G.



MAX-2-SAT (Williams 2004)

Solution idea

- We assign weights to edges so that the weight of the vwu triangle in G equals the number of clauses satisfied with the assignment (v, w, u).
- Then it is sufficient to check if there is a triangle of weight k in G.

Problem 1 How should we assign weights?
Let $c(v) =$ all the clauses satisfied under the (partial) assignment v.
Then the number of clauses satisfied under the assignment (v, w, u) amounts to:

$$\begin{aligned}|c(v) \cup c(w) \cup c(u)| = &|c(v)| + |c(w)| + |c(u)| \\ &- |c(v) \cap c(w)| - |c(v) \cap c(u)| - |c(w) \cap c(u)| \\ &+ |c(v) \cap c(w) \cap c(u)|.\end{aligned}$$

MAX-2-SAT (Williams 2004)

Solution idea

- We assign weights to edges so that the weight of the vwu triangle in G equals the number of clauses satisfied with the assignment (v, w, u).
- Then it is sufficient to check if there is a triangle of weight k in G.

Problem 1 How should we assign weights?
Let $c(v)$ = all the clauses satisfied under the (partial) assignment v.
Then the number of clauses satisfied under the assignment (v, w, u) amounts to:

$$\begin{aligned} |c(v) \cup c(w) \cup c(u)| = & |c(v)| + |c(w)| + |c(u)| \\ & - |c(v) \cap c(w)| - |c(v) \cap c(u)| - |c(w) \cap c(u)| \\ & + \underbrace{|c(v) \cap c(w) \cap c(u)|}_{0}. \end{aligned}$$

MAX-2-SAT (Williams 2004)

Solution idea

- We assign weights to edges so that the weight of the *vwu* triangle in G equals the number of clauses satisfied with the assignment (v, w, u).
- Then it is sufficient to check if there is a triangle of weight k in G.

Problem 1 How should we assign weights?
Let $c(v)$ = all the clauses satisfied under the (partial) assignment v.
Then the number of clauses satisfied under the assignment (v, w, u) amounts to:

$$\begin{aligned}|c(v) \cup c(w) \cup c(u)| = & |c(v)| + |c(w)| + |c(u)| \\ & - |c(v) \cap c(w)| - |c(w) \cap c(u)| - |c(u) \cap c(v)| \\ & + \underbrace{|c(v) \cap c(w) \cap c(u)|}_{0}.\end{aligned}$$

So, we put $\text{weight}(xy) = |c(x)| - |c(x) \cap c(y)|$.

MAX-2-SAT (Williams 2004)

We are left with verifying whether there is a triangle of weight k in G.

A trick

Consider all $O(k^2)$ partitions $k = k_0 + k_1 + k_2$. For every partition we build a graph G_{k_0,k_1,k_2} which consists only of:

- edges of weight k_0 between 2^{V_0} and 2^{V_1},
- edges of weight k_1 between 2^{V_1} and 2^{V_2},
- edges of weight k_2 between 2^{V_2} and 2^{V_0},

Then it suffices to...

MAX-2-SAT (Williams 2004)

We are left with verifying whether there is a triangle of weight k in G.

A trick

Consider all $O(k^2)$ partitions $k = k_0 + k_1 + k_2$. For every partition we build a graph G_{k_0,k_1,k_2} which consists only of:

- edges of weight k_0 between 2^{V_0} and 2^{V_1},
- edges of weight k_1 between 2^{V_1} and 2^{V_2},
- edges of weight k_2 between 2^{V_2} and 2^{V_0},

Then it suffices to... check whether there is a triangle.

Checking whether G_{k_0,k_1,k_2} contains a triangle

Corollary

- Graph G_{k_0,k_1,k_2} has $3 \cdot 2^{n/3}$ vertices.
- We can verify whether G_{k_0,k_1,k_2} contains a triangle in $O(2^{\omega n/3}) = O(1.7302^n)$ time and $O(2^{2/3n})$ space.
- Hence we can check whether G contains a triangle of weight k in $O(k^2 \cdot 2^{\omega n/3}) = O^*(1.731^n)$ time.

MAX-2-SAT (Williams 2004): Conclusion

Corollary

There is an algorithm for MAX-2-SAT running in $O^*(1.731^n)$ time and $O(2^{2/3n}) = O(1.588^n)$ space.

MAX-2-SAT (Williams 2004): Conclusion

Corollary

There is an algorithm for MAX-2-SAT running in $O^*(1.731^n)$ time and $O(2^{2/3n}) = O(1.588^n)$ space.

It is easy to modify the algorithm (how?) to get

Corollary

There is an algorithm which counts the number of optimum MAX-2-SAT solutions running in $O^*(1.731^n)$ time and $O(2^{2/3n})$ space.