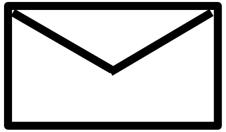
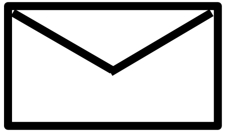
# Efficient Representation of Numerical Optimization Problems for SNARKs

Sebastian Angel, Andrew Blumberg, Eleftherios Ioannidis, Jess Woods

# Example: School Admissions



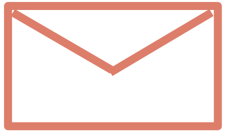
1. School **commits** to secret admissions criteria
2. Students apply to school



3. School **commits** to students' materials


4. School can then **prove (in zero knowledge)** that they admitted the best (most optimal) applicants given its criteria, without leaking criteria or information about specific students

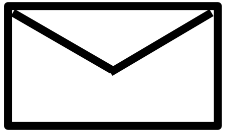
# Example: School Admissions



1. School **commits** to secret admissions criteria

   GPA > 3.0, SAT > 500, sports + music > 1, incoming class enrollment < 200

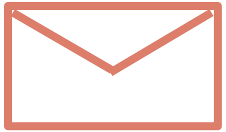
2. Students apply to school



3. School **commits** to students' materials


4. School can then **prove (in zero knowledge)** that they admitted the best (most optimal) applicants given its criteria, without leaking criteria or information about specific students

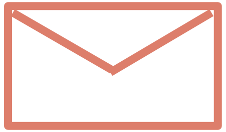
# Example: School Admissions



1. School **commits** to secret admissions criteria

   **GPA > 3.0, SAT > 500, sports + music > 1, incoming class enrollment < 200**

2. Students apply to school



3. School **commits** to students' materials

   **Jane Doe, GPA 3.5, SAT 720, sports; Joe Public, GPA 2.9, SAT 530, music; …**


4. School can then **prove (in zero knowledge)** that they admitted the best (most optimal) applicants given its criteria, without leaking criteria or information about specific students

# Optimization Problems

- Resource allocation
- Stocks and marketing
- Transportation
- Scheduling

→ Linear programming

- Circuit manufacturing
- Matrix completion

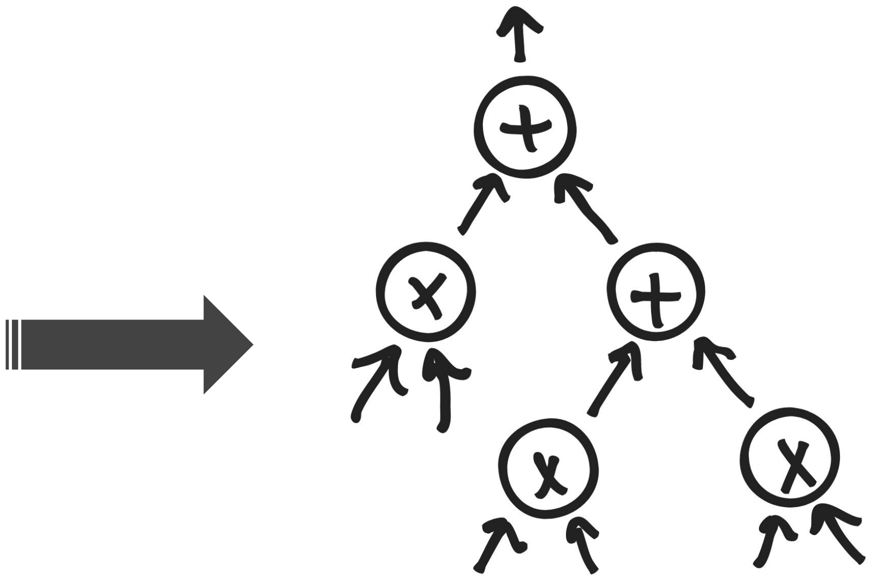
→ Semidefinite programming

- Neural network training

→ Stochastic gradient descent

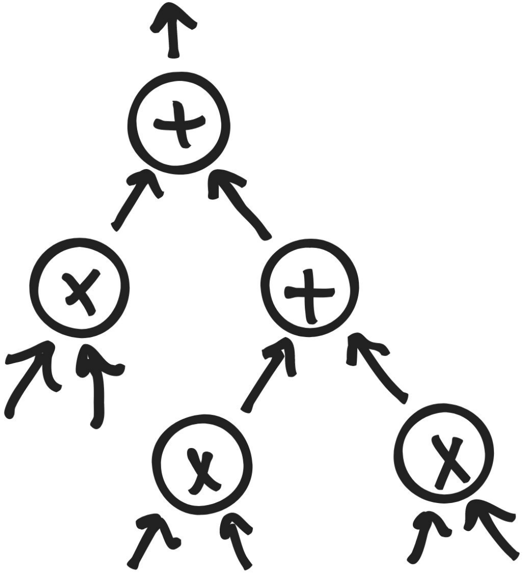
# Naive Solution

```
void simplex(Tableau tableau) {
    add_slack_variables(tableau);

    while( true ) {
        int pivot_col = find_pivot_column(tableau);
        if( pivot_col < 0 ) { // solution found
            optimal_vector(tableau);
            break;
        }

        int pivot_row = find_pivot_row(tableau, pivot_col);
        if (pivot_row < 0) { // unbounded (no solution)
            break;
        }

        float pivot = tableau.mat[pivot_row][pivot_col];
        for(int k = 0; k < tableau.cols; k++) {
            tableau.mat[pivot_row][k] =
             tableau.mat[pivot_row][k] / pivot;
        }

        for(int l = 0; l < tableau.rows; l++) {
            float multiplier = tableau.mat[l][pivot_col];
            if(l != pivot_row) {
                for(int m = 0; m < tableau.cols; m++) {
                    tableau.mat[l][m] = tableau.mat[l][m] -
                   (multiplier * tableau.mat[pivot_row][m]);
                }
            }
        }
        optimal_vector(tableau);
    }
}
```



# Naive Solution

```
void simplex(Tableau tableau) {
    add_slack_variables(tableau);

    while( true ) {
        int pivot_col = find_pivot_column(tableau);
        if( pivot_col < 0 ) { // solution found
            optimal_vector(tableau);
            break;
        }

        int pivot_row = find_pivot_row(tableau, pivot_col);
        if (pivot_row < 0) { // unbounded (no solution)
            break;
        }

        float pivot = tableau.mat[pivot_row][pivot_col];
        for(int k = 0; k < tableau.cols; k++) {
            tableau.mat[pivot_row][k] =
             tableau.mat[pivot_row][k] / pivot;
        }

        for(int l = 0; l < tableau.rows; l++) {
            float multiplier = tableau.mat[l][pivot_col];
            if(l != pivot_row) {
                for(int m = 0; m < tableau.cols; m++) {
                    tableau.mat[l][m] = tableau.mat[l][m] -
                   (multiplier * tableau.mat[pivot_row][m]);
                }
            }
        }
        optimal_vector(tableau);
    }
}
```



# Otti

Don't reason about solver implementation.

Leverage nondeterminism to reason about the solution.

Automated & Practical!

# The rest of the talk

- We'll follow Otti through a **linear programming** problem
    - (See the paper for methods on **semidefinite programming** and **stochastic gradient descent**)

- Examine how to use Otti

- Limitations and Conclusion

# Linear Programming Problems

Find a vector $\boldsymbol{x}$ that maximizes $\boldsymbol{c}^T \bullet \boldsymbol{x}$

Subject to: $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$

$\boldsymbol{x} \geq \boldsymbol{0}$

# Linear Programming Problems

Find a vector $\boldsymbol{x}$ that maximizes $\boldsymbol{c}^T \bullet \boldsymbol{x}$

Subject to: $\mathbf{A}\boldsymbol{x} \leq \boldsymbol{b}$

$\boldsymbol{x} \geq 0$

Find a vector $\boldsymbol{y}$ that minimizes $\boldsymbol{b}^T \bullet \boldsymbol{y}$

Subject to: $\mathbf{A}^T\boldsymbol{y} \geq \boldsymbol{c}$

$\boldsymbol{y} \geq 0$

# Certificate of Optimality for Linear Programming

Find a vector $\boldsymbol{x}$ that maximizes $\boldsymbol{c}^T \bullet \boldsymbol{x}$

Subject to: $\mathbf{A}\boldsymbol{x} \le \boldsymbol{b}$

$\boldsymbol{x} \ge 0$

Find a vector $\boldsymbol{y}$ that minimizes $\boldsymbol{b}^T \bullet \boldsymbol{y}$

Subject to: $\mathbf{A}^T\boldsymbol{y} \ge \boldsymbol{c}$

$\boldsymbol{y} \ge 0$

Given solutions $x^*$, $y^*$ ...

Primal feasibility: $\mathbf{A}\boldsymbol{x}^* \le \boldsymbol{b}$

$\boldsymbol{x}^* \ge 0$

Dual feasibility: $\mathbf{A}^T\boldsymbol{y}^* \ge \boldsymbol{c}$

$\boldsymbol{y}^* \ge 0$

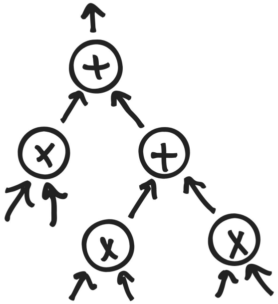
**Strong Duality:** $\boldsymbol{c}^T \bullet \boldsymbol{x}^* = \boldsymbol{b}^T \bullet \boldsymbol{y}^*$

# Certificate of Optimality for Linear Programming

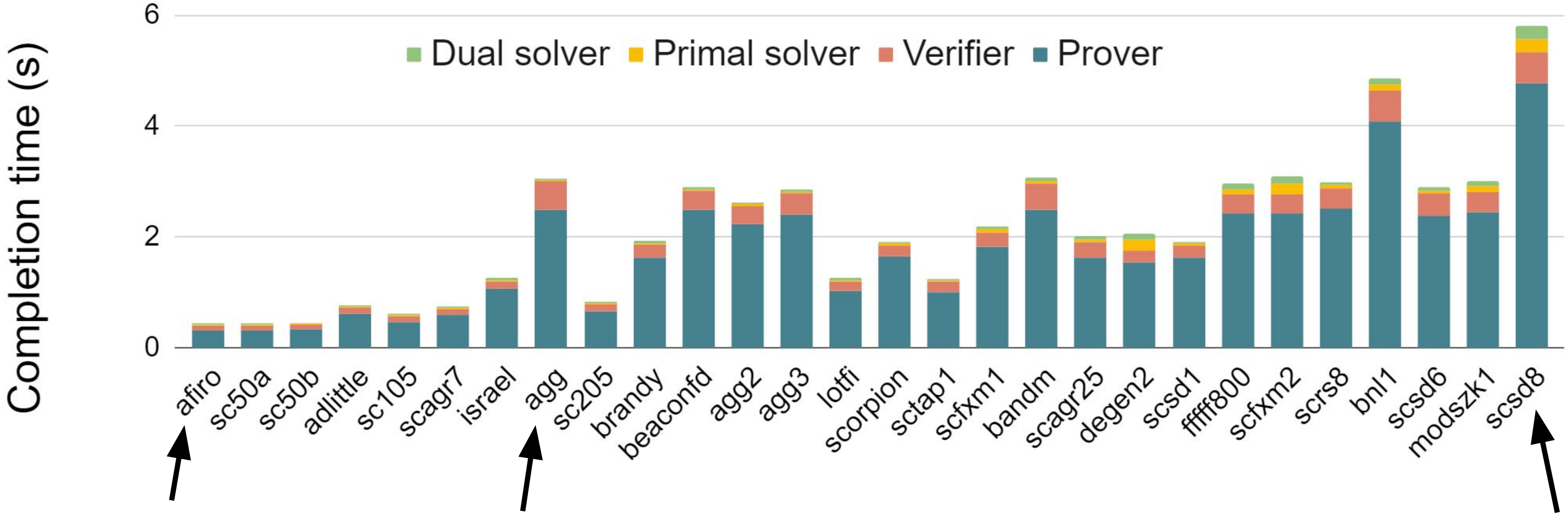
Primal feasibility: $Ax^* \le b$
$x^* \ge 0$

Dual feasibility: $A^T y^* \ge c$
$y^* \ge 0$

**Strong Duality:** $c^T \bullet x^* = b^T \bullet y^*$



# How well does Otti work for Linear Programming?




**28 equations,**
**32 variables,**
**20 KB proof size**

**489 equations,**
**163 variables,**
**48 KB proof size**

**398 equations,**
**2750 variables,**
**81 KB proof size**

# Flow of Otti

Find a vector ***x*** that maximizes **objective function**

Subject to **constraints**

Prover

$x^*, y^* =$

```
void simplex(Tableau tableau) {
    add_slack_variables(tableau);

    while( true ) {
        int pivot_col = find_pivot_column(tableau);
        if( pivot_col < 0 ) { // solution found
            optimal_vector(tableau);
            break;
        }

        int pivot_row = find_pivot_row(tableau, pivot_col);
        if (pivot_row < 0) { // unbounded (no solution)
            break;
        }

        float pivot = tableau.mat[pivot_row][pivot_col];
        for(int k = 0; k < tableau.cols; k++) {
            tableau.mat[pivot_row][k] =
                       tableau.mat[pivot_row][k] / pivot;
        }

        for(int l = 0; l < tableau.rows; l++) {
            float multiplier = tableau.mat[l][pivot_col];
            if(l != pivot_row) {
                for(int m = 0; m < tableau.cols; m++) {
                    tableau.mat[l][m] = tableau.mat[l][m] -
                     (multiplier * tableau.mat[pivot_row][m]);
                }
            }
        }
        optimal_vector(tableau);
    }
}
```

**Solver**

# Flow of Otti

Find a vector $\boldsymbol{x}$ that maximizes **objective function**

Subject to **constraints**

Prover

$x^*, y^* =$

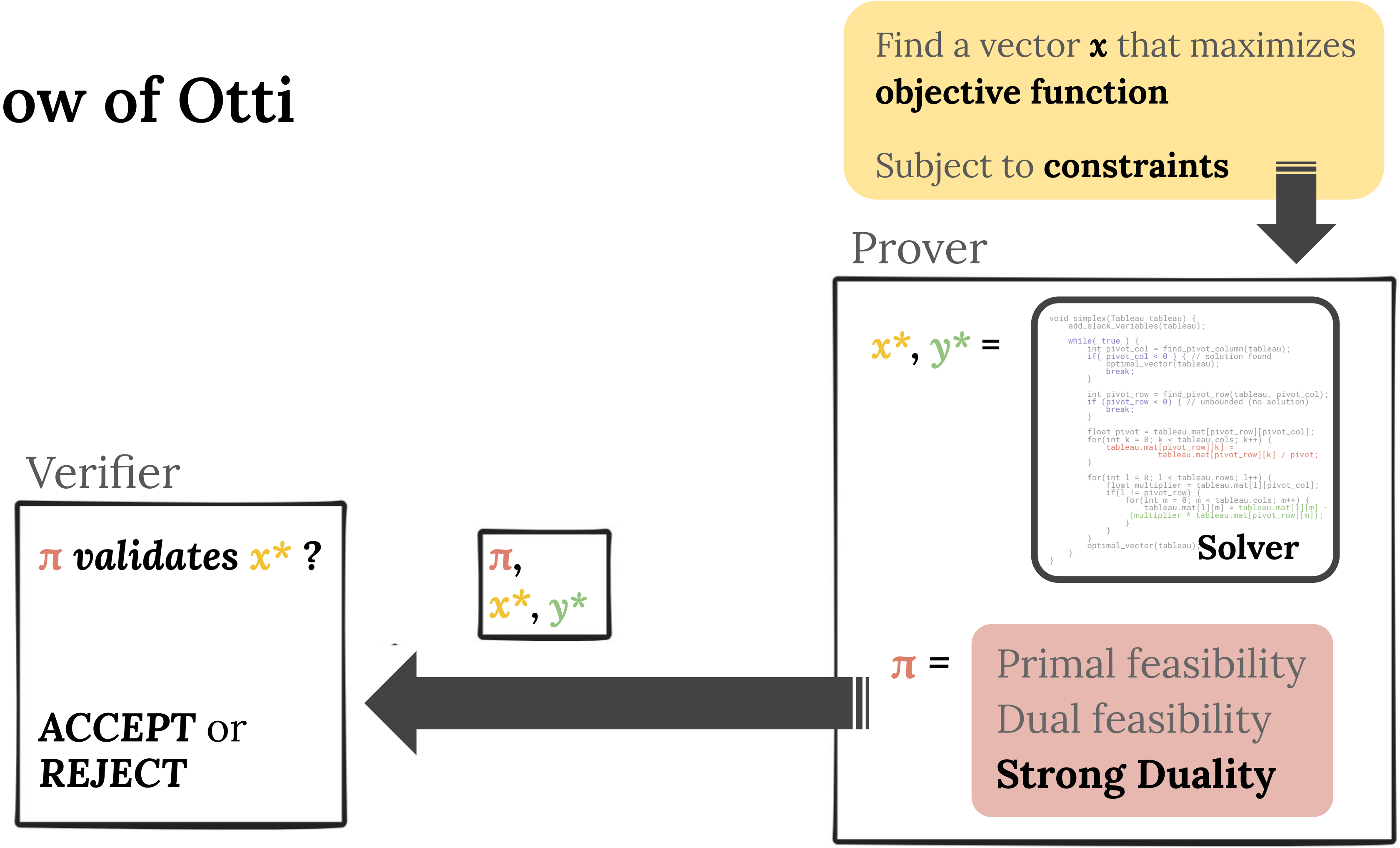
```
void simplex(Tableau tableau) {
    add_slack_variables(tableau);

    while( true ) {
        int pivot_col = find_pivot_column(tableau);
        if( pivot_col < 0 ) { // solution found
            optimal_vector(tableau);
            break;
        }

        int pivot_row = find_pivot_row(tableau, pivot_col);
        if (pivot_row < 0) { // unbounded (no solution)
            break;
        }

        float pivot = tableau.mat[pivot_row][pivot_col];
        for(int k = 0; k < tableau.cols; k++) {
            tableau.mat[pivot_row][k] =
                           tableau.mat[pivot_row][k] / pivot;
        }

        for(int l = 0; l < tableau.rows; l++) {
            float multiplier = tableau.mat[l][pivot_col];
            if(l != pivot_row) {
                for(int m = 0; m < tableau.cols; m++) {
                    tableau.mat[l][m] = tableau.mat[l][m] -
                 (multiplier * tableau.mat[pivot_row][m]);
                }
            }
        }
        optimal_vector(tableau);
    }
}
```

**Solver**

$\pi =$

Primal feasibility
Dual feasibility
**Strong Duality**

# Flow of Otti



Find a vector x that maximizes
objective function
Subject to constraints
Prover
x*, y* =
void simplex(Tableau tableau) {
add_slack_variables(tableau);
while( true ) {
int pivot_col = find_pivot_column(tableau);
if( pivot_col < 0 ) { // solution found
optimal_vector(tableau);
break;
}
int pivot_row = find_pivot_row(tableau, pivot_col);
if (pivot_row < 0) { // unbounded (no solution)
break;
}
float pivot = tableau.mat[pivot_row][pivot_col];
for(int k = 0; k < tableau.cols; k++) {
tableau.mat[pivot_row][k] =
tableau.mat[pivot_row][k] / pivot;
}
for(int l = 0; l < tableau.rows; l++) {
float multiplier = tableau.mat[l][pivot_col];
if(l != pivot_row) {
for(int m = 0; m < tableau.cols; m++) {
tableau.mat[l][m] = tableau.mat[l][m] -
(multiplier * tableau.mat[pivot_row][m]);
}
}
}
optimal_vector(tableau);
}
}
Solver
π =
Primal feasibility
Dual feasibility
Strong Duality
π,
x*, y*
Verifier
π validates x* ?
ACCEPT or
REJECT

# What the Verifier sees

Primal feasibility: $Ax^* \leq b$

$x^* \geq 0$

Dual feasibility: $A^T y^* \geq c$

$y^* \geq 0$

**Strong Duality:** $c^T \bullet x^* = b^T \bullet y^*$

A, b, c

solution $x^*$, $y^*$

$\_ \bullet x^* \leq \_$

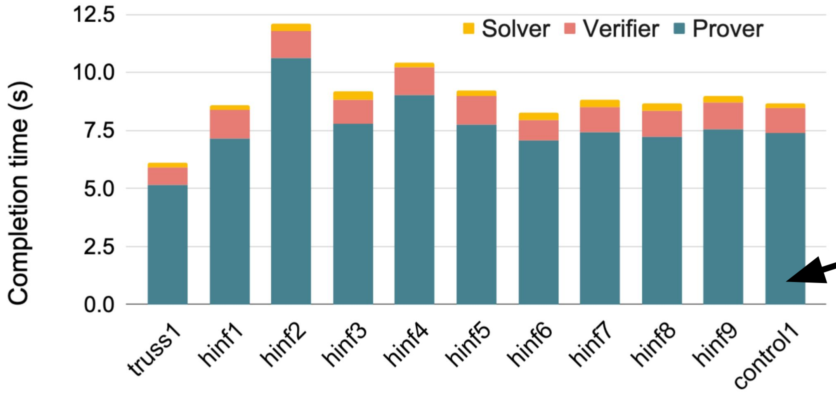
$x^* \geq 0$

$\_^T \bullet y^* \geq \_$

$y^* \geq 0$

$\_^T \bullet x^* = \_^T \bullet y^*$

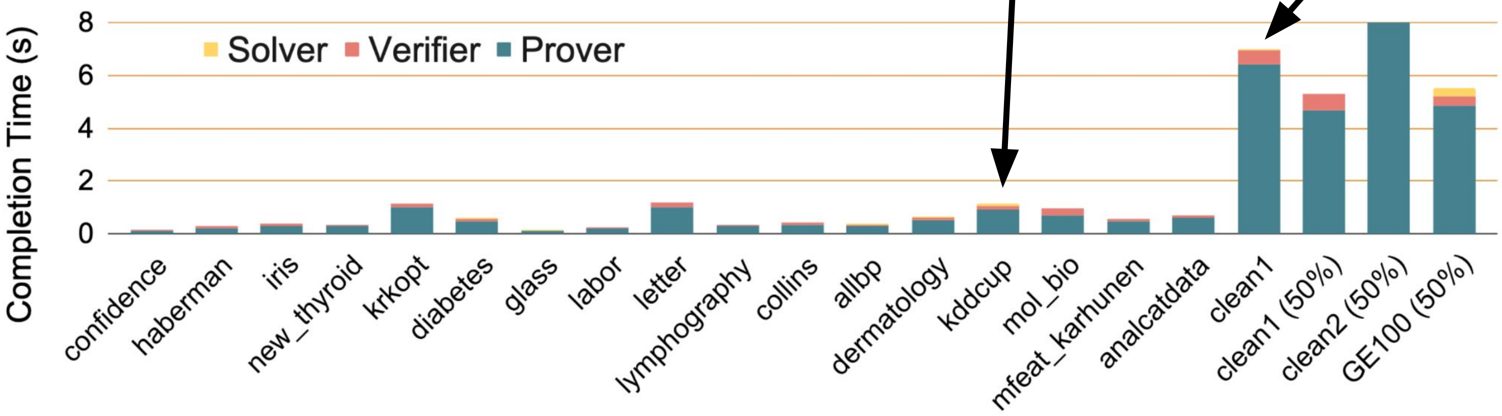
# How well does Otti work for SDP and SGD?




21 equations,
15 x 15 matrices,
80 KB proof size

494,020 data points,
41 features,
29 KB proof size

476 data points,
168 features,
80 KB proof size



# Limitations

- Still an upper size limit
- Compilation is still a bottleneck
- Problems must have specific structure
- Overhead (where the baseline is unverifiable solvers)

# Takeaways

Otti . . .

- Demonstrates the effectiveness of **nondeterministic checkers**
- **Automates** entire pipeline
- Proves optimality on **real world datasets**
- **Over 4 orders of magnitude faster** than any existing approach

**https://github.com/eniac/otti**