Bounded Integer Linear Constraint Solving via Lattice Search

Joe Hendrix and Ben Jones
Galois, Inc

| galois |

Overview

- We present an algorithm for solving systems of integer linear constraints where each linear form has lower and upper bounds.
  - We call these bounded integer linear problems.
- We implemented this in a tool called BLT.
- BLT has proven many orders of magnitude faster on a class of preimage problems arising from JPEG.

Overview

- We present an algorithm for solving systems of integer linear constraints where each linear form has lower and upper bounds.
  - We call these bounded integer linear problems.
- We implemented this in a tool called BLT.
- BLT has pr… …s of preimage proble…

On one of the “easier” problems, we ran several SMT solvers for months without success.

BLT takes under a second to find a solution.

Bounded Integer Linear Constraints

- Bounded integer linear constraint problems have the form below:

$$
\begin{aligned}
& l_1 \leq a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq u_1 \\
\wedge\ & l_2 \leq a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq u_2 \\
& \cdots \\
\wedge\ & l_m \leq a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq u_m
\end{aligned}
$$

- Both the lower and upper bounds l_i and u_i coefficients a_{ij} are rational constants.

- The free variables x_j are all integers.

Bounded Integer Linear Constraints

- Bounded integer linear constraint problems have the form below:

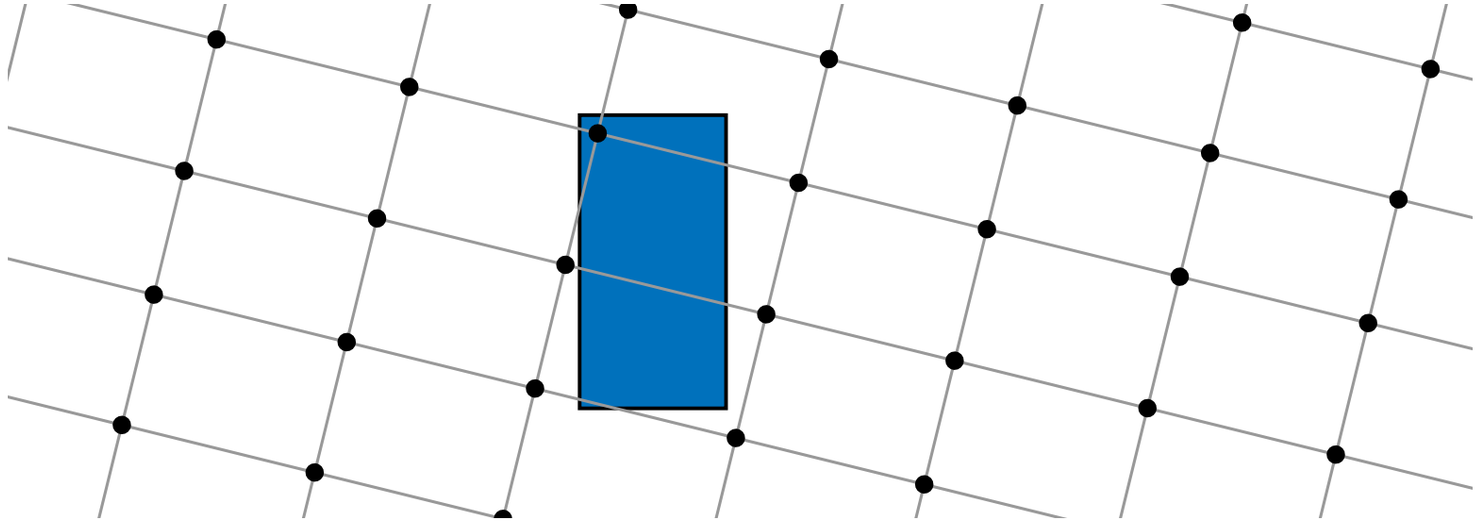
$$l \leq Ax \leq u$$

- Both the lower and upper bounds l_i and u_i coefficients a_{ij} are rational constants.
- The free variables x_j are all integers.

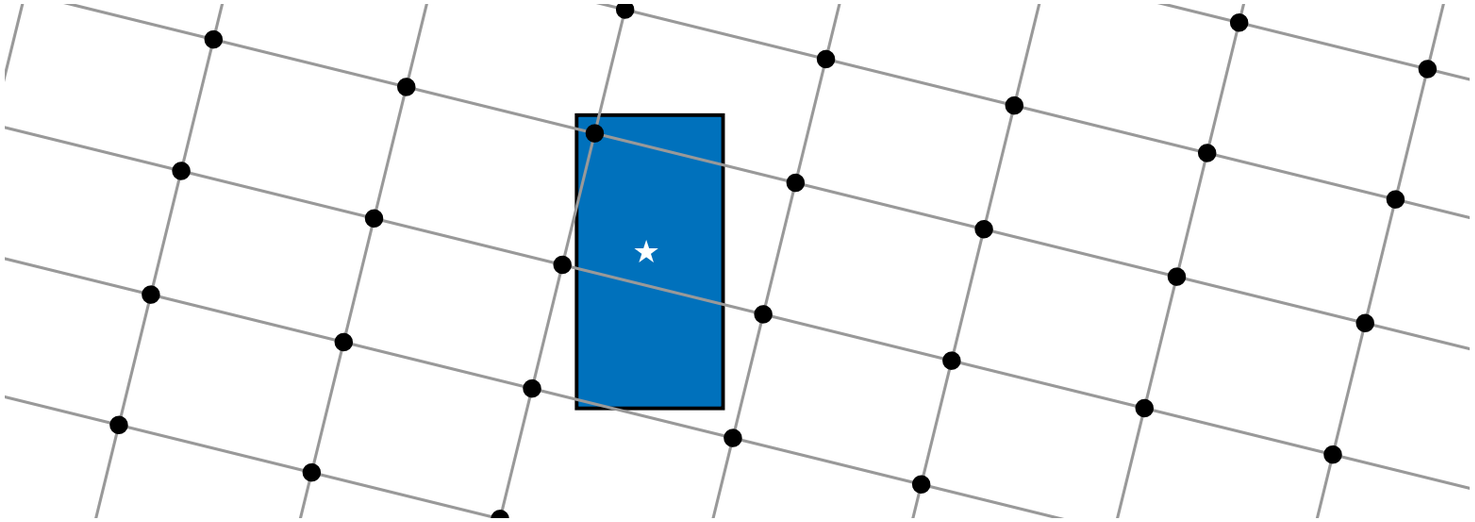
- The two problems below are equivalent:
  - Finding a solution to $l \leq Ax \leq u$.
  - Finding a point in the intersection of the lattice generated by the columns of A and hyper-rectangle with corners l and u.

$$l \leq \mathbf{A}_0 x_0 + A_1 x_{1\ +} + \ldots + A_n x_n \leq u$$

- The two problems below are equivalent:
  - Finding a solution to $\boldsymbol{l} \leq \boldsymbol{Ax} \leq \boldsymbol{u}$.
  - Finding a point in the intersection of the lattice generated by the columns of $\boldsymbol{A}$ and hyper-rectangle with corners $\boldsymbol{l}$ and $\boldsymbol{u}$.



- The two problems below are equivalent:
  - Finding a solution to $\boldsymbol{l} \leq \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{u}$.
  - Finding a point in the intersection of the lattice generated by the columns of $\boldsymbol{A}$ and hyper-rectangle with corners $\boldsymbol{l}$ and $\boldsymbol{u}$.

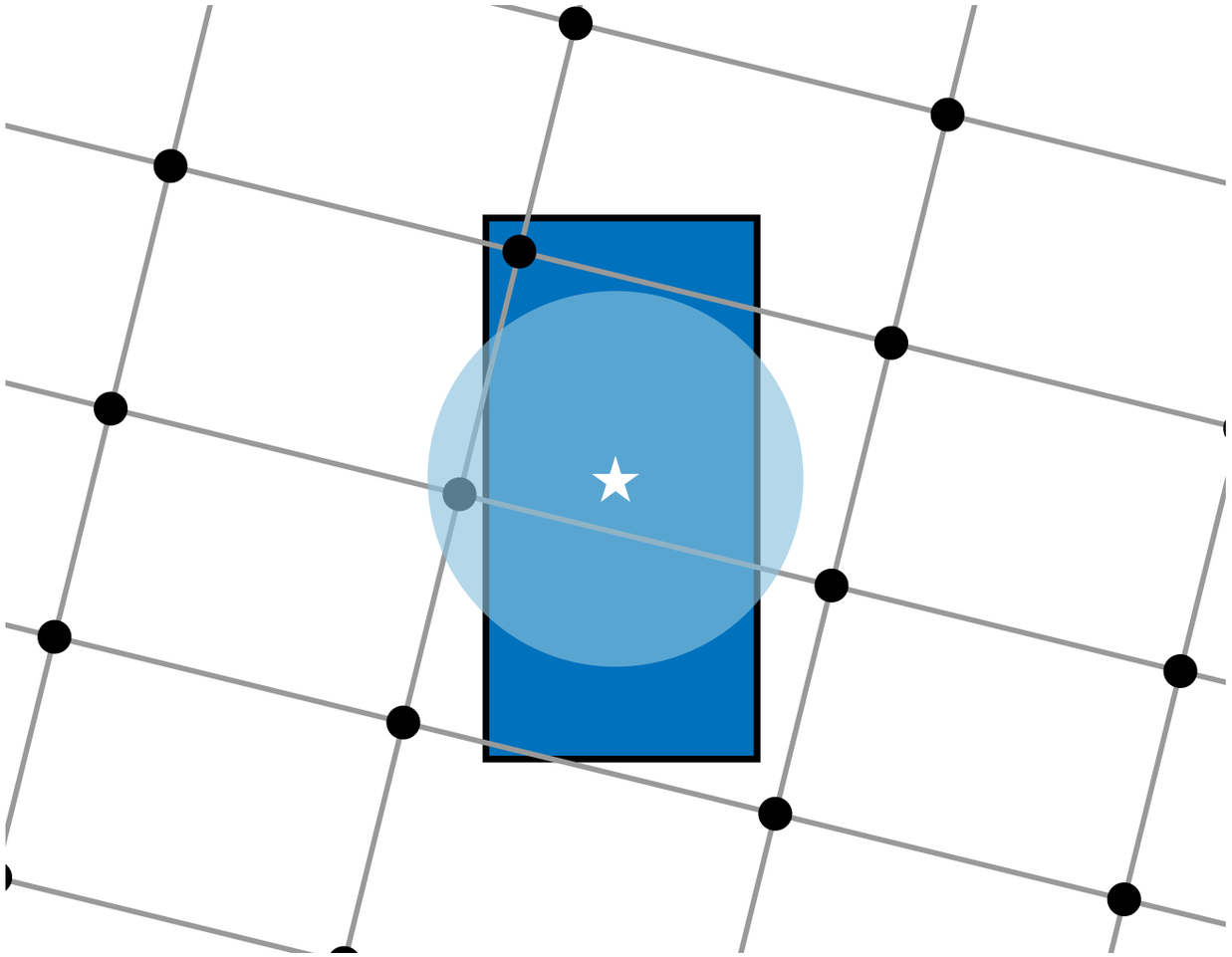


- Intuitively, it makes sense to look for points near the center of the rectangle.

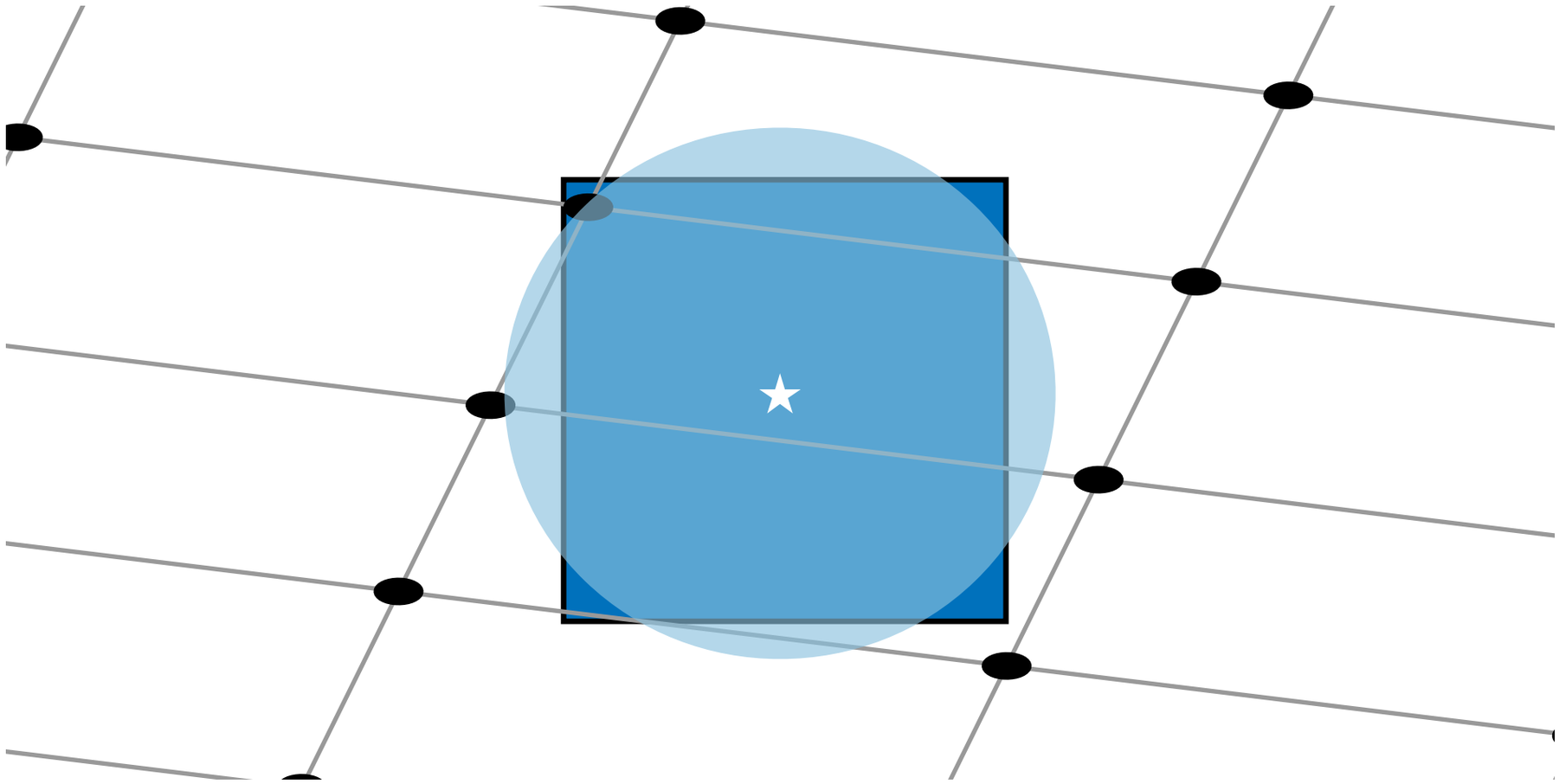
Closest Vector Problem

- Is there an algorithm to find a lattice point close to a given location?
  - Yes! Schnorr-Euchner is a well known exponential-time algorithm for this problem.
  - It incrementally builds up a partial assignment making guesses using a **nearest hyperplane** heuristic.
  - It detects when there are no real solutions within a given distance of the target point and backtracks.
- However, it uses the L^2 distance—not what we want.

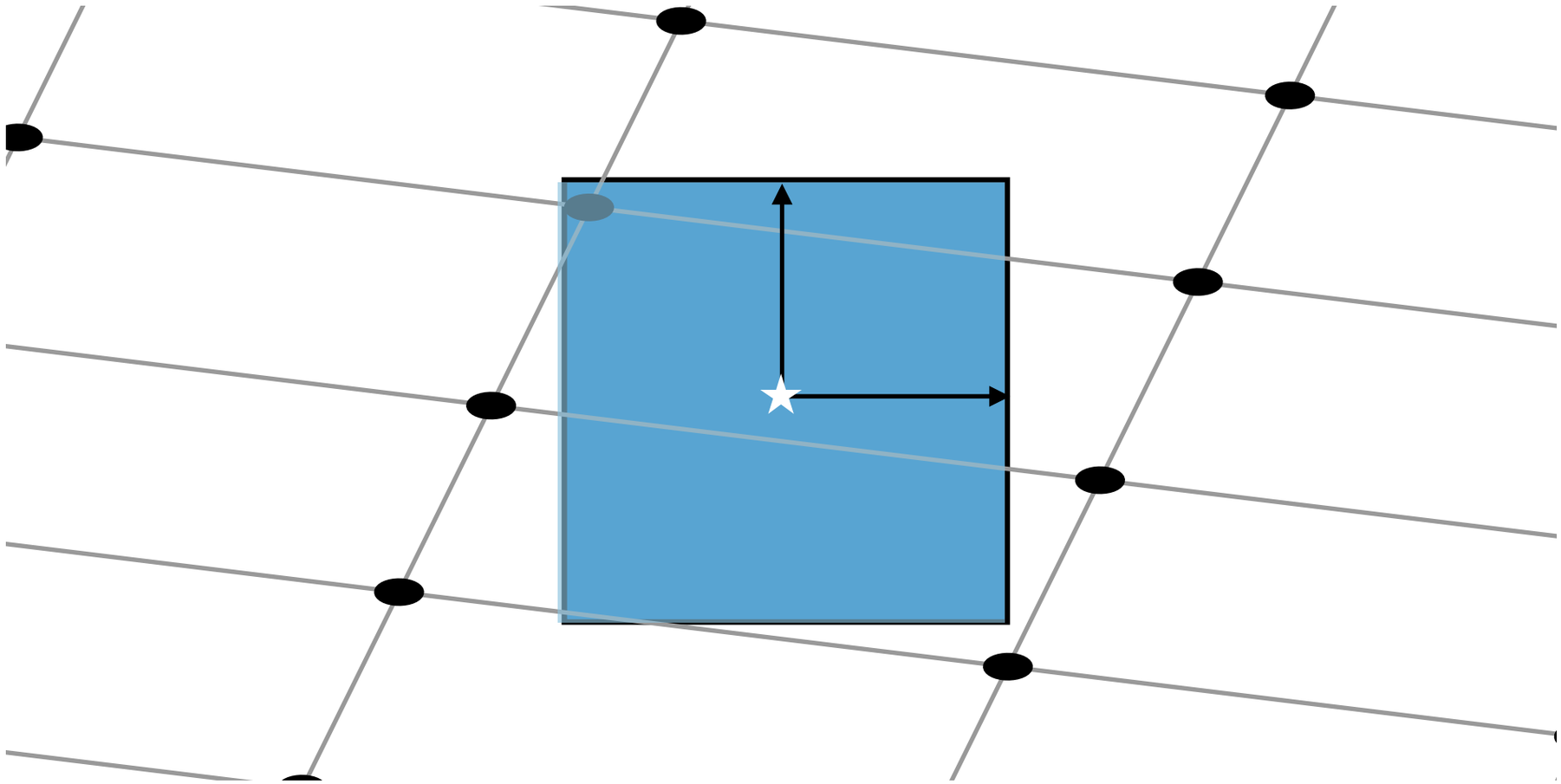
L^2 Approximation



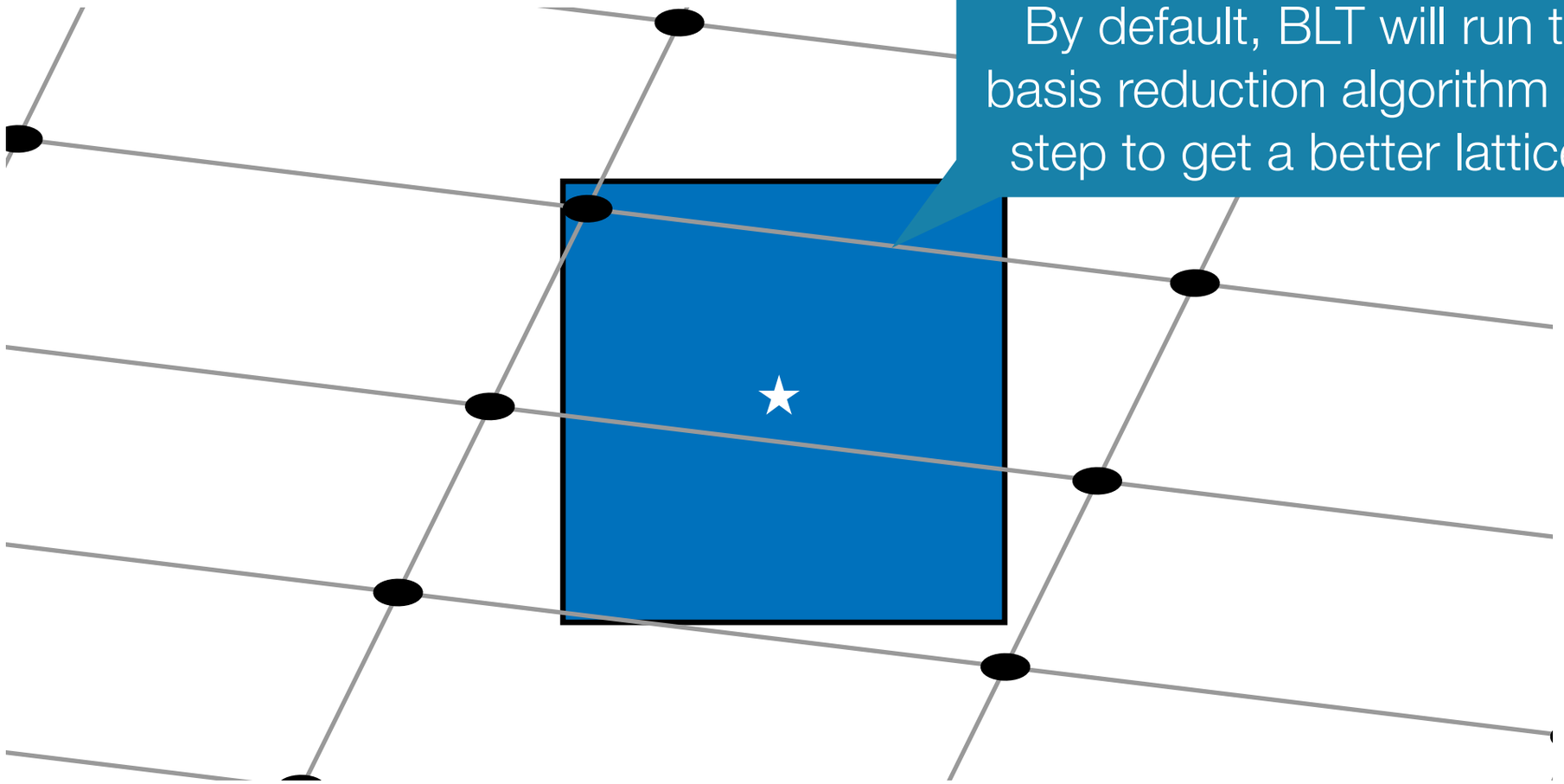
Normalize Dimensions



Use L∞ Distance

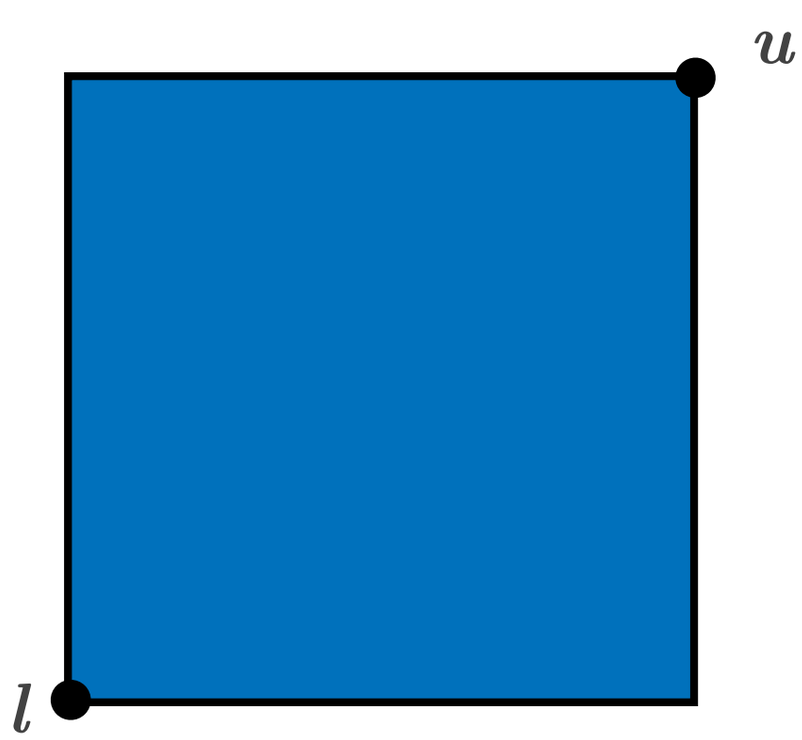


Normalize Dimensions

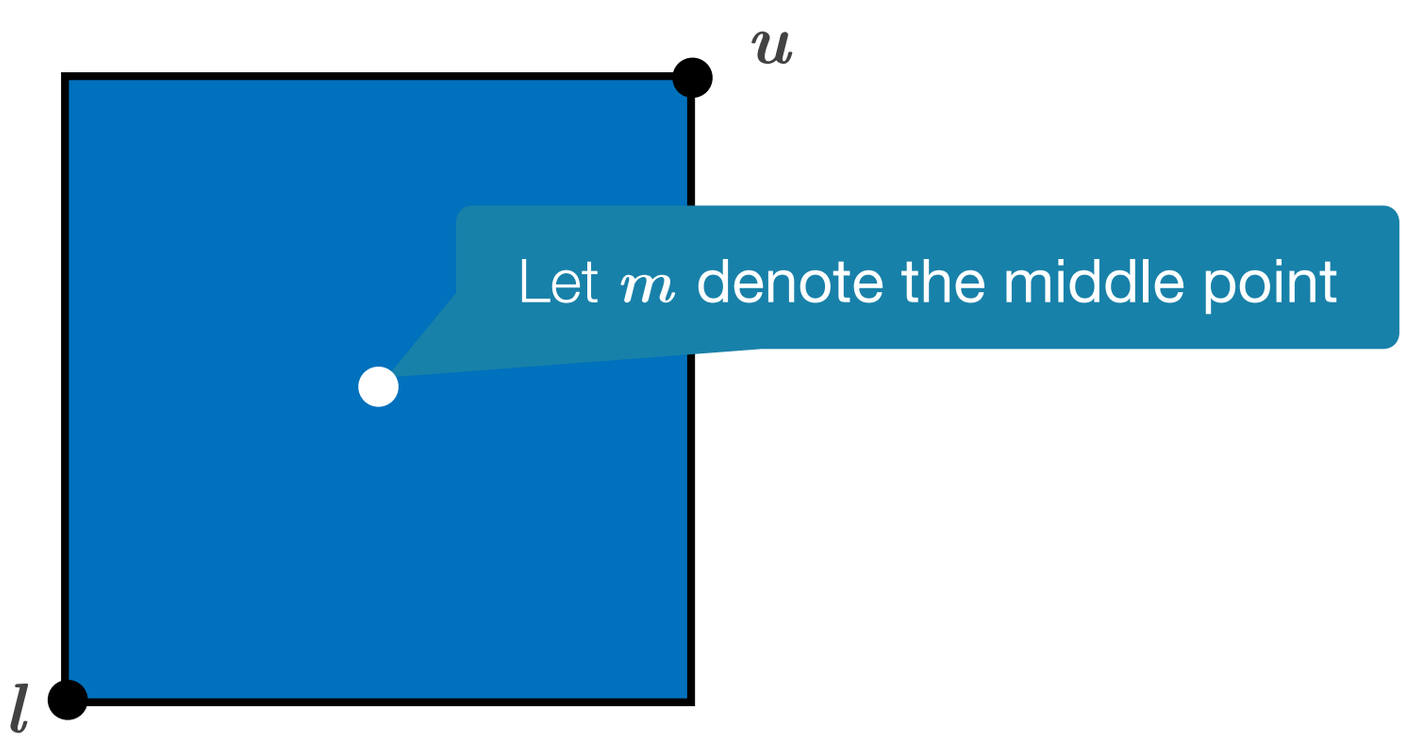


The Algorithm

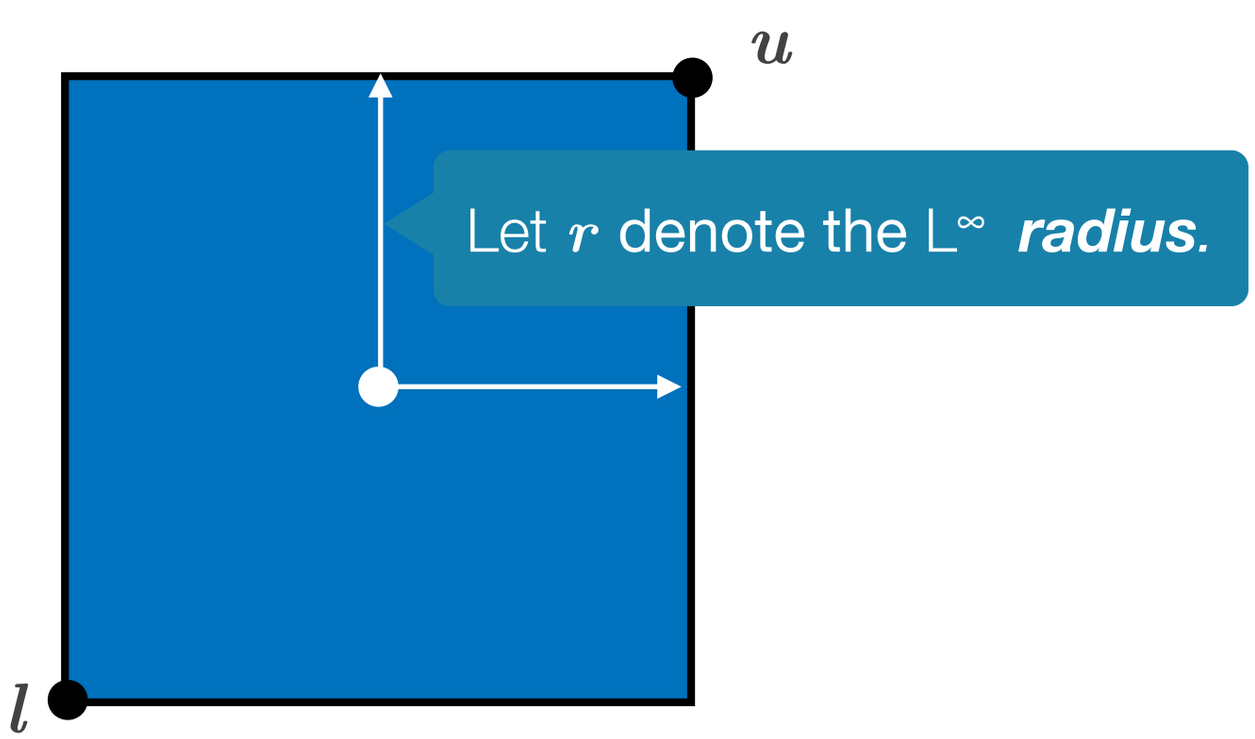
The Hypercube



The Hypercube



The Hypercube



L^∞ Distance

- The L^∞ distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$ is just the largest difference between two coefficients.

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) := \max\{|x_i - y_i| \mid i = 1, \ldots, n\}$$

- This also extends to subsets $X, Y \subseteq \mathbb{R}^n$

$$d_\infty(X, Y) := \min_{(\boldsymbol{x}, \boldsymbol{y}) \in X \times Y} d_\infty(\boldsymbol{x}, \boldsymbol{y})$$

- When X and Y can be described by a system of linear inequalities, then the distance can be solved by linear programming.

$$
\begin{aligned}
d_\infty(X,Y) &= \min\{\ d_\infty(x,y) \mid x \in X,\ y \in Y\ \} \\
&= \min\left\{\ \max_i\{|x_i - y_i|\} \mid x \in X,\ y \in Y\ \right\} \\
&= \min\{\ t \mid |x_i - y_i| \le t,\ x \in X,\ y \in Y\ \} \\
&= \min\{\ t \mid x_i - y_i \le t,\ -(x_i - y_i) \le t,\ x \in X\, y \in Y\ \}
\end{aligned}
$$

Lattice Definitions

- A **Lattice** over a fixed basis $\{\boldsymbol{a}_1\, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_n\}$ is the set of points that can be expressed as a integral sum of the vectors in the basis.

$$\mathcal{L} = \left\{ \sum_{i=1}^{n} c_i \boldsymbol{a}_i \mid c_i \in \mathbb{Z} \right\}$$

- Given a partial mapping θ from variables to integers, we can define the **sublayer** of θ as follows:

$$\mathcal{L}_\theta^{\mathbb{Z}} := \left\{ \sum_{i=1}^{n} c_i \boldsymbol{a}_i \mid c_i \in \mathbb{Z},\, i \in \mathrm{dom}(\theta) \Rightarrow c_i = \theta(i) \right\}$$

Lattice Definitions

- To prune solutions, we use the real affine linear space below:

$$\mathcal{L}_\theta^{\mathbb{R}} := \left\{ \sum_{i=1}^{n} c_i \boldsymbol{a}_i \mid c_i \in \mathbb{R},\, i \in \mathrm{dom}(\theta) \Rightarrow c_i = \theta(i) \right\}$$

BLT Algorithm

- Our search procedure maintains a partial assignment θ from variables to integers with the invariant:

$$d_\infty(\mathcal{L}_\theta^{\mathbb{R}}, m) \leq r$$

- We start with the empty assignment.

- At each step we first choose an unbound variable y to add to the substitution.

- BLT chooses the assignment to y by finding the value $c \in \mathbb{R}$ that minimizes the distance:

$$d_\infty(\mathcal{L}^{\mathbb{R}}_{\theta[y \mapsto c]}, m)$$

- The first assignment to y is chosen by rounding c to the nearest integer

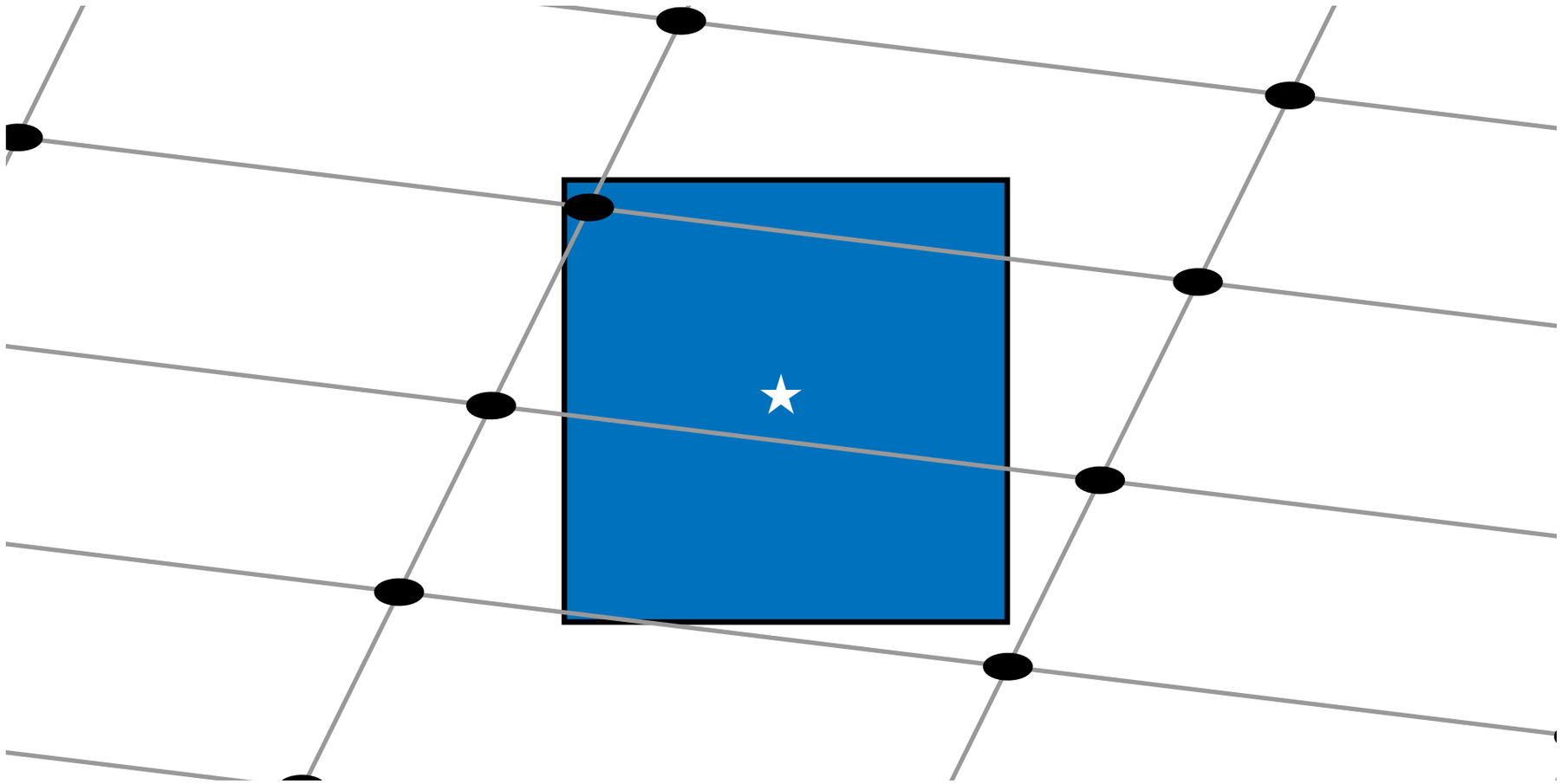
- If we need to backtrack, we incrementally explore values starting from those near c and working outwards

$$\ldots, \lfloor c-1 \rfloor, \lfloor c \rfloor, \lceil c \rceil, \lceil c+1 \rceil, \ldots$$

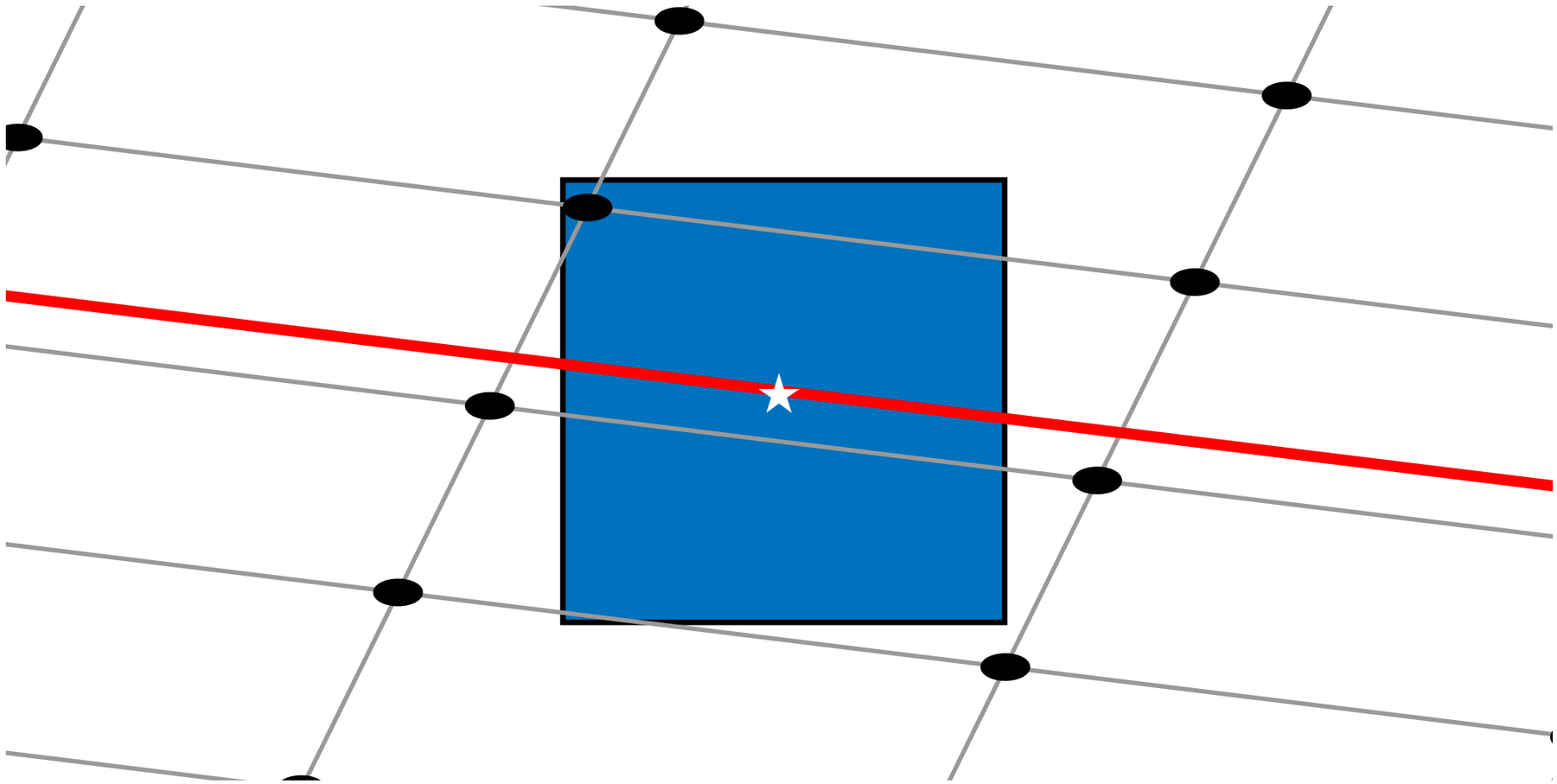
- We show in the paper that the number of values is finite, and thus our procedure will eventually terminate.

Example

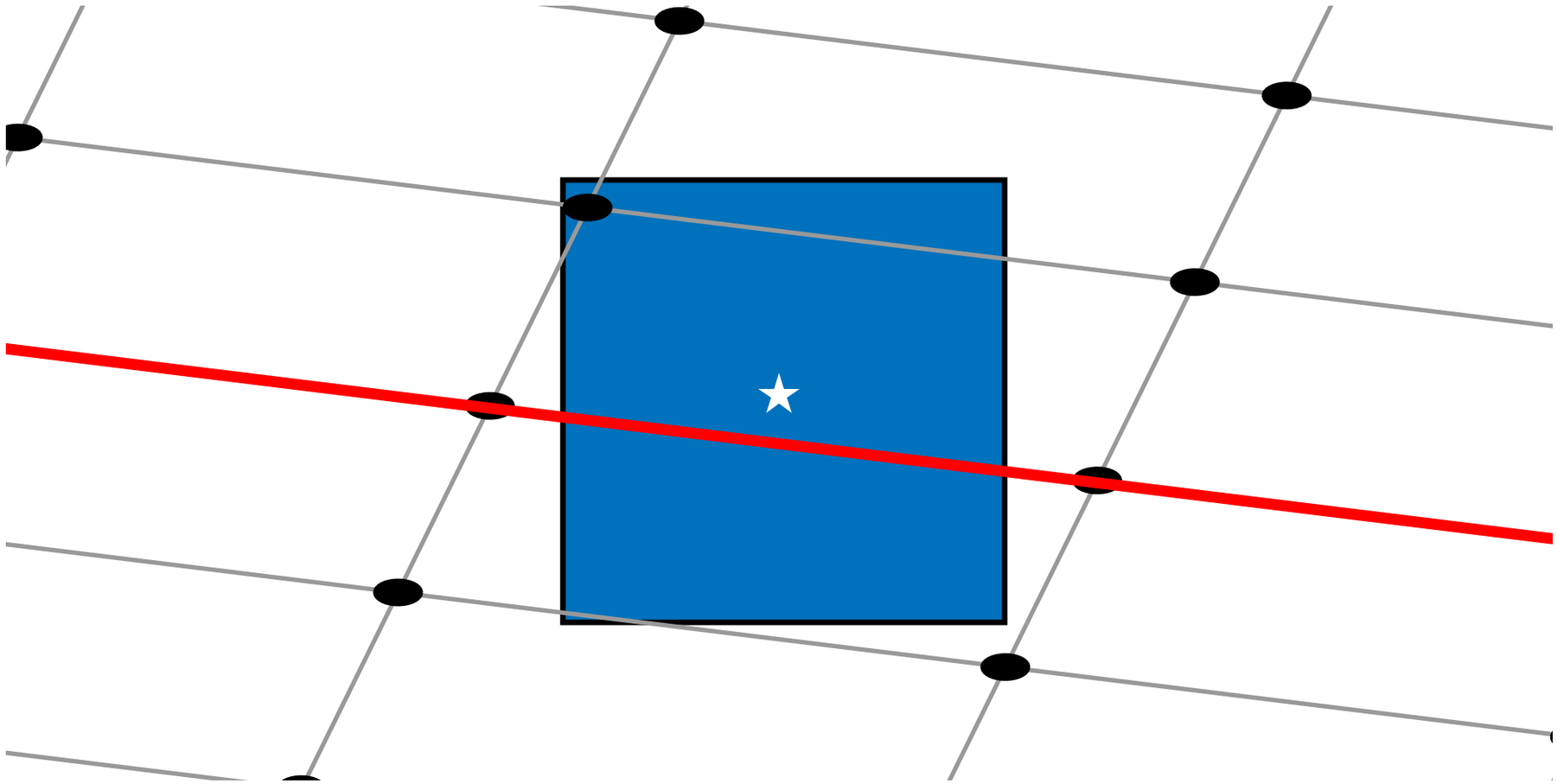
Example



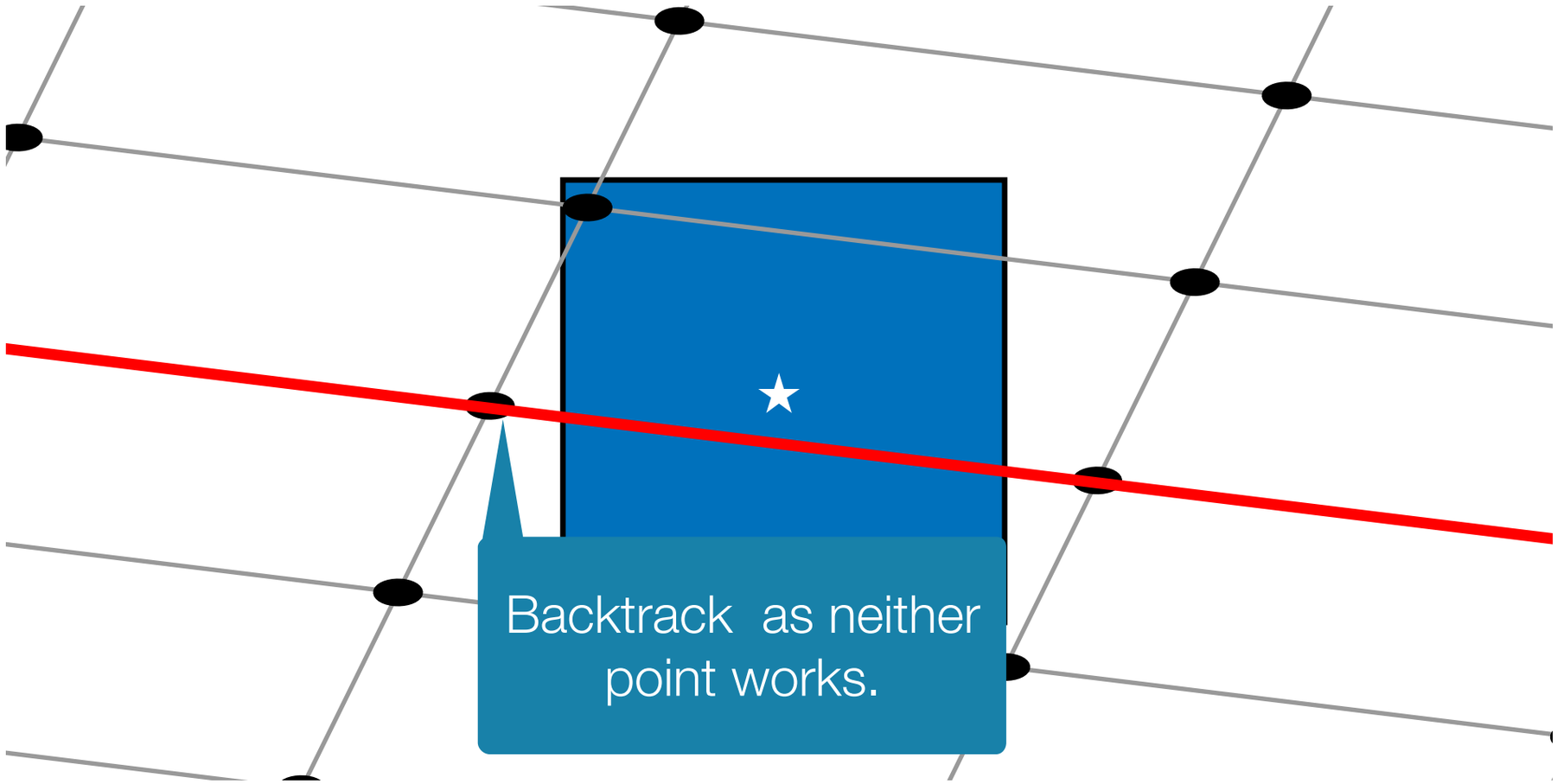
First Choice



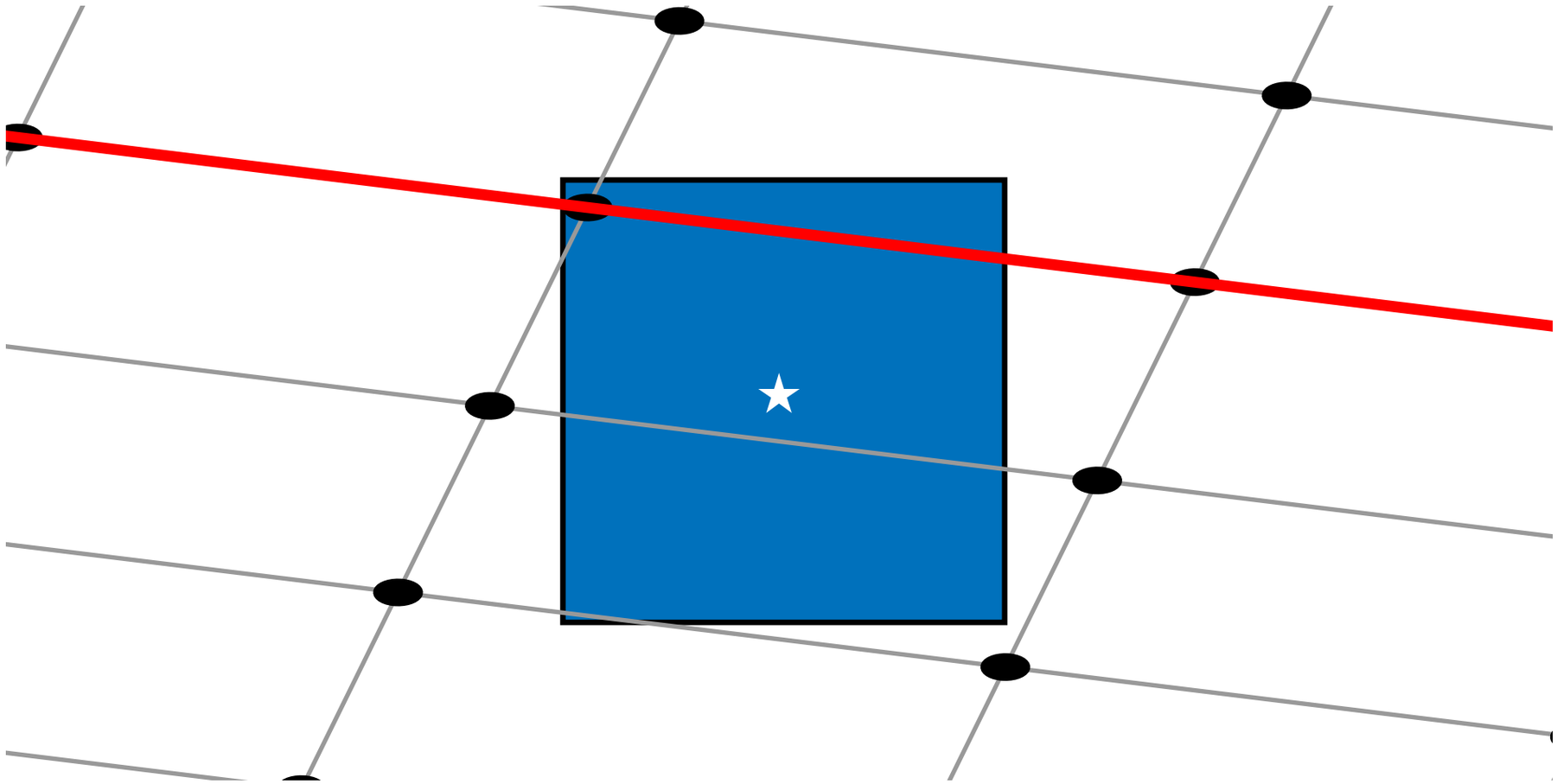
First Choice



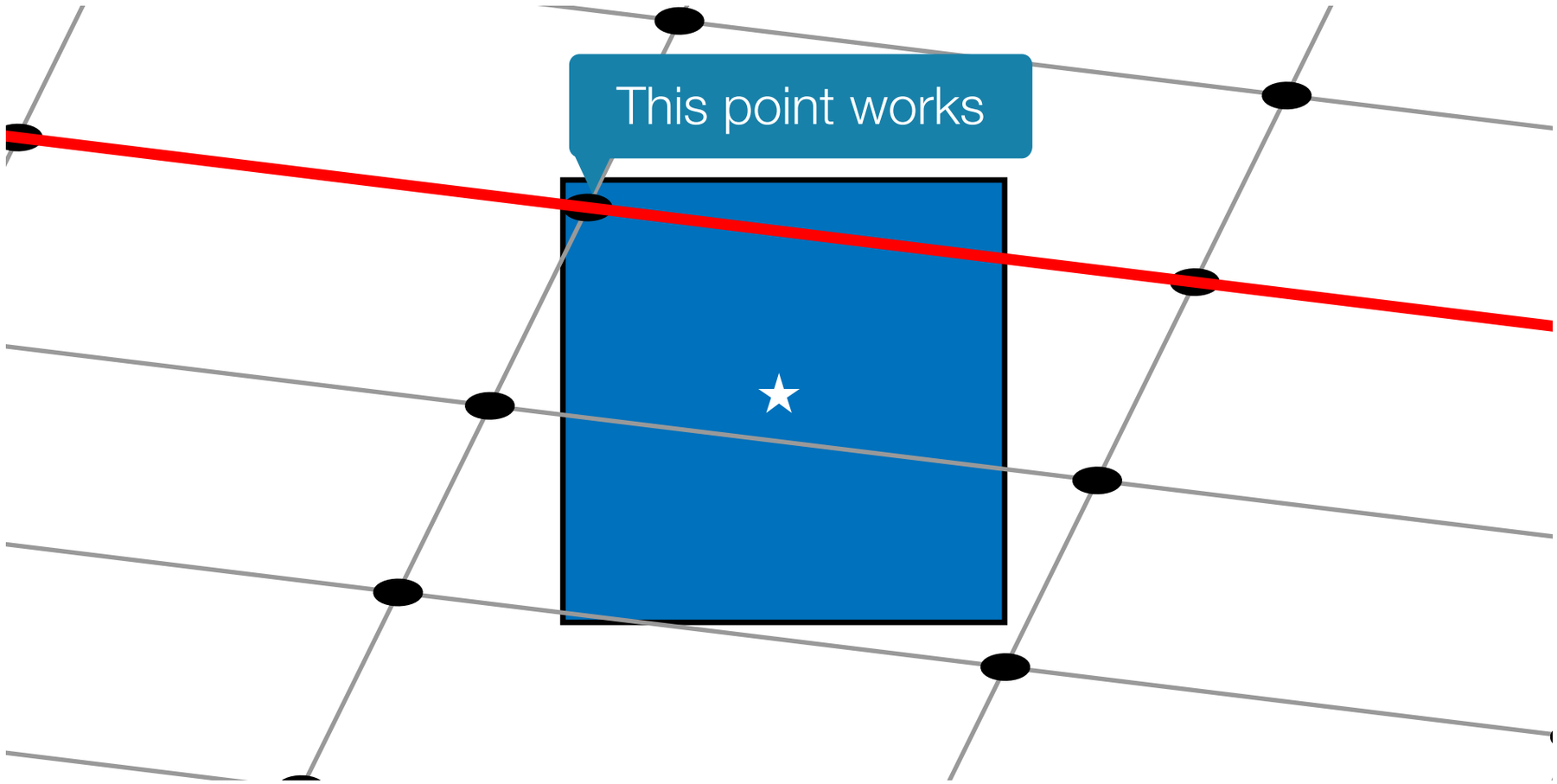
First Choice



Second Choice



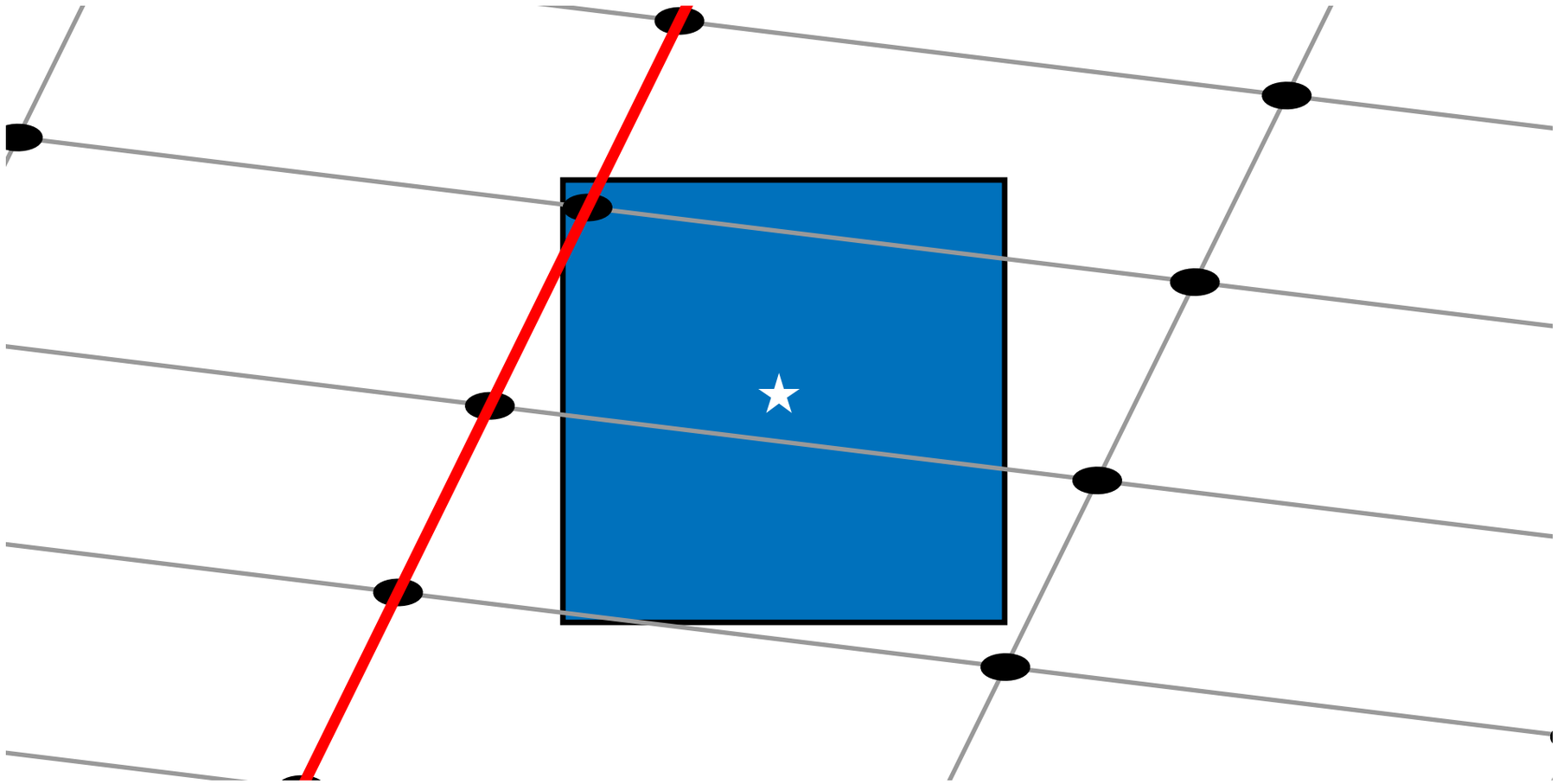
Second Choice



Longest Vector First

- This approach worked, but required backtracking.
- As an additional heuristic, BLT will always assign values to variables in the order of decreasing size of the associated basis vector.

Better assignment



JPEG Preimages

JPEG Preimages

- It turns out that JPEG decompression can be expressed as linear function from integer coefficients in a file.
- Each 8x8 block of pixels in a single channel is computed from 64 integer coefficients $\boldsymbol{C}$ that are stored in a compressed form

$$\mathbf{I} = \mathrm{idct}_2(\mathbf{Q}_{lvl} .* \mathbf{C}) + 128$$

JPEG Preimages

- It turns out that JPEG decompression can be expressed as linear function from integer coefficients in a file.
- Each 8x8 block of pixels in a single channel is computed from 64 integer coefficients C that are stored in a compressed form

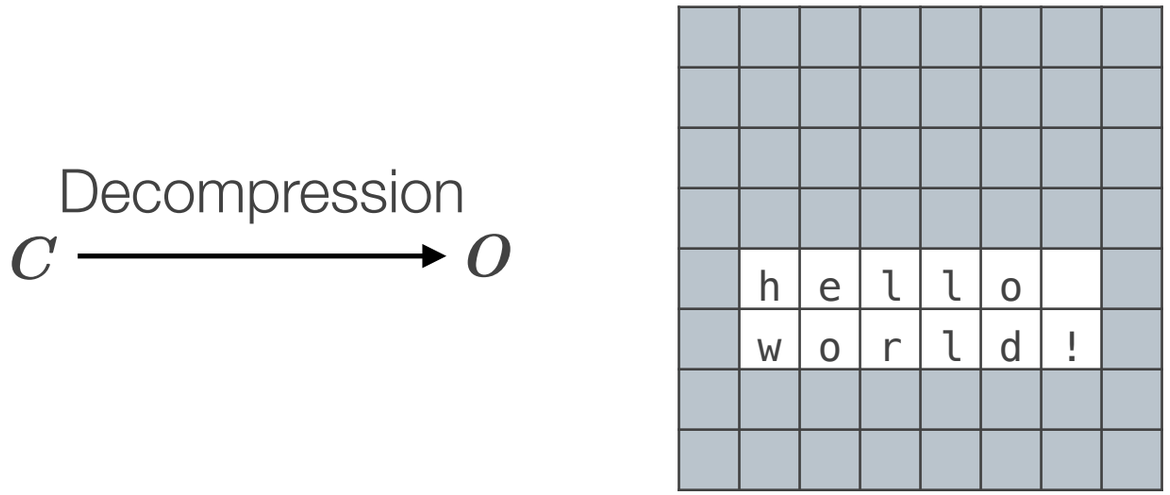
$$\mathbf{I} = \mathrm{idct}_2(\mathbf{Q}_{lvl} .* \mathbf{C}) + 128$$

The coefficients in Q_{lvl} depend on the JPEG quality level.

High Compression Ratio = Low Quality Level = Large values in Q_{lvl}

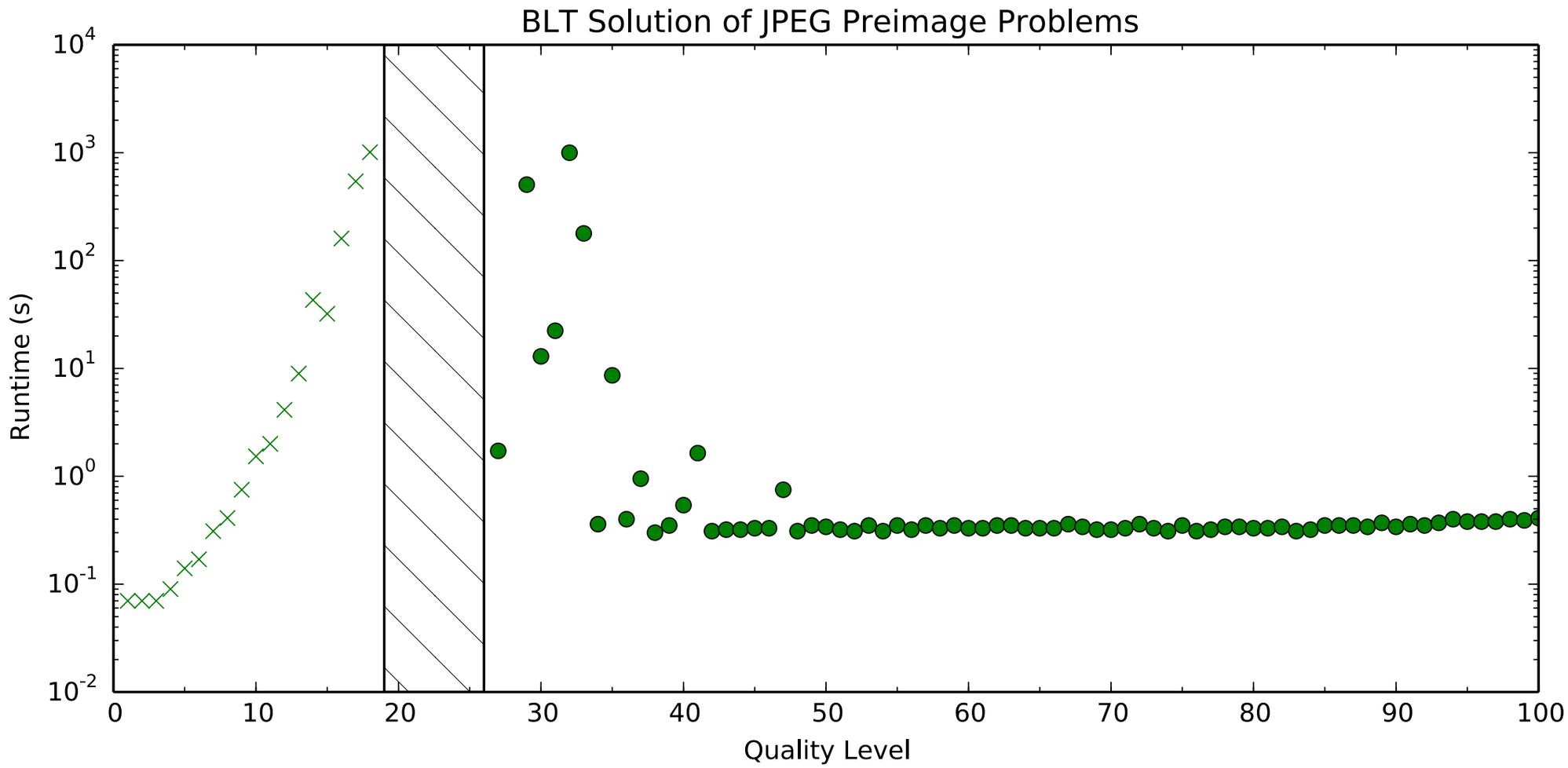
JPEG Preimages

- In the paper, we pose the problem of finding a set of compressed coefficients so that the decompressed image satisfies precise constraints on specific pixels.

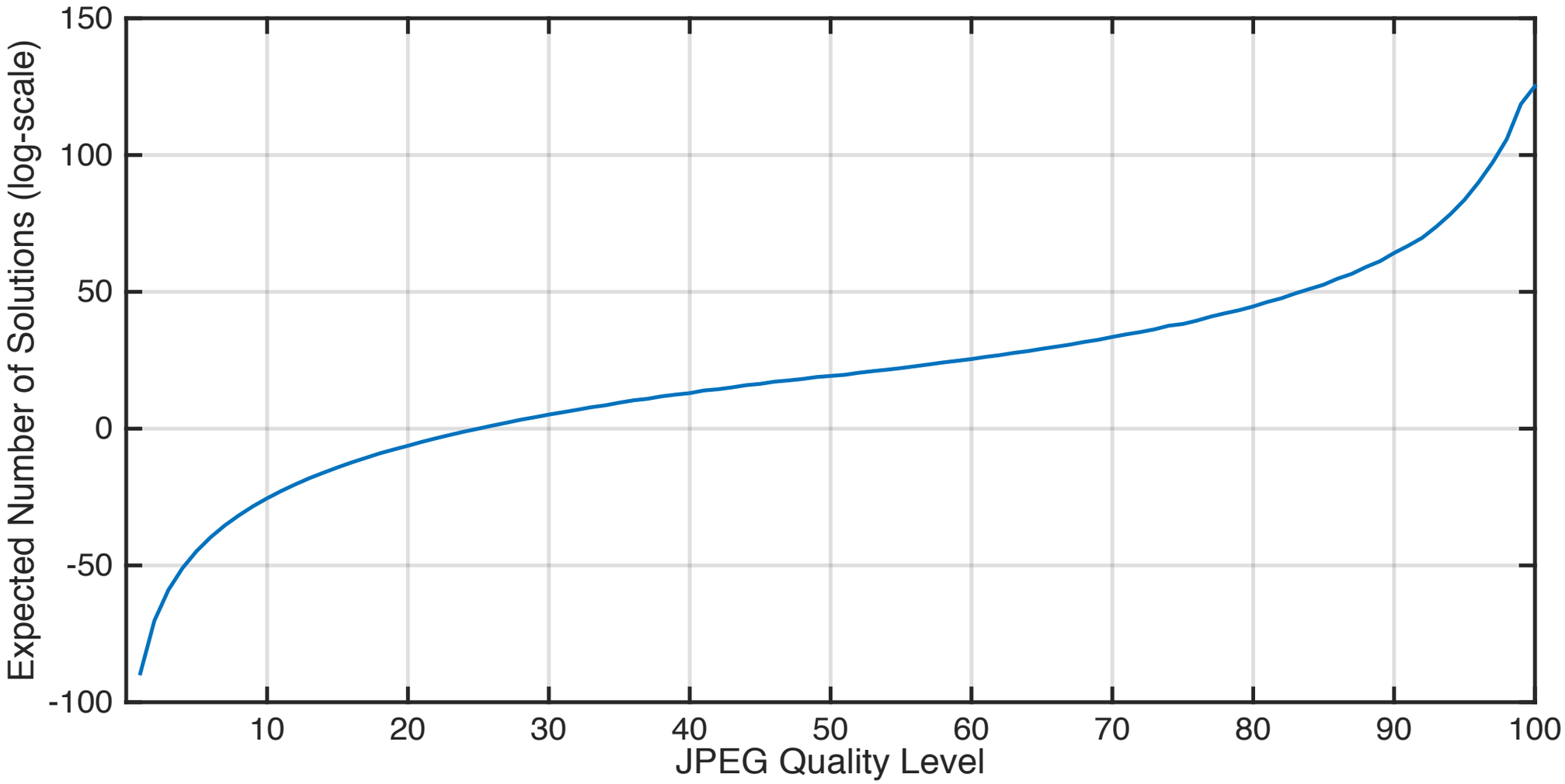


- We show how this can in turn be converted into a bounded ILP problem and solved efficiently using BLT.

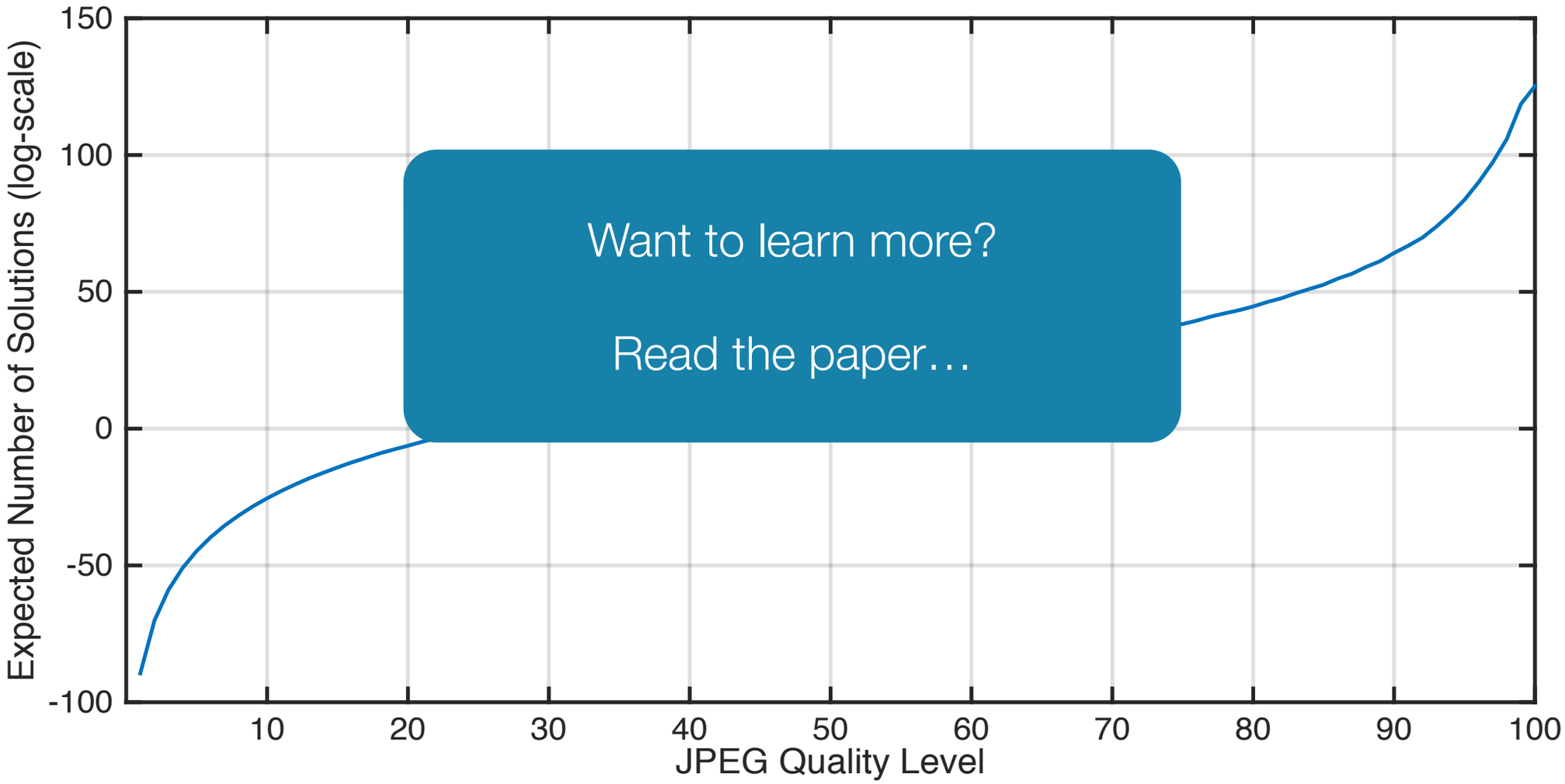
BLT Benchmark Results



Number of Solutions



BLT Benchmark Results



Implementation Experience

- The algorithm is implemented in a C++ library called BLT.
  - BLT uses GLPK for solving real linear problems.
    - GLPK uses floating point; UNSAT results may be incorrect.
- In addition to CVC4, Yices, Z3, we have also tried:
  - Bit-level version of the problem using Boolector, abc, and glucose.
  - Optimization tools such as Gurobi and GLPK.

Conclusions

- This paper describes an algorithm for solving ILP problems with the form

$$l \leq Ax \leq u$$

  - We have shown the 2-dimensional case, but the algorithm works for arbitrary systems of linear equations with this form.
  - We have not seen Schnorr-Euchner generalized in this way, nor have we seen it used for solving linear equations.
- In future work, we would like to explore ways to integrate this into SMT solvers.
- Once we have finished some remaining cleanups, we plan to release BLT and SMTLIB 2 encodings of the JPEG problem.

Thanks

We'd like to thank David Flamm, Patrick Lincoln, Dejan Jovanović, and Grant Passmore for feedback and discussions that helped develop the ideas in this paper.