Chapter 1
Linear Programming

Paragraph 4
The Simplex Algorithm

What we did so far

- By combining ideas of a specialized algorithm with a geometrical view on the problem, we developed an algorithm idea:

> Find a feasible corner (somehow).
> Check neighboring corners and see if one is better.
> Move over to the next corner until no better neighboring solution exists.

- We studied the mathematical foundations of this algorithm sketch. We know what corners are (basic feasible solutions), and we have proven that, if it exists, an optimal solution can always be found in a corner.

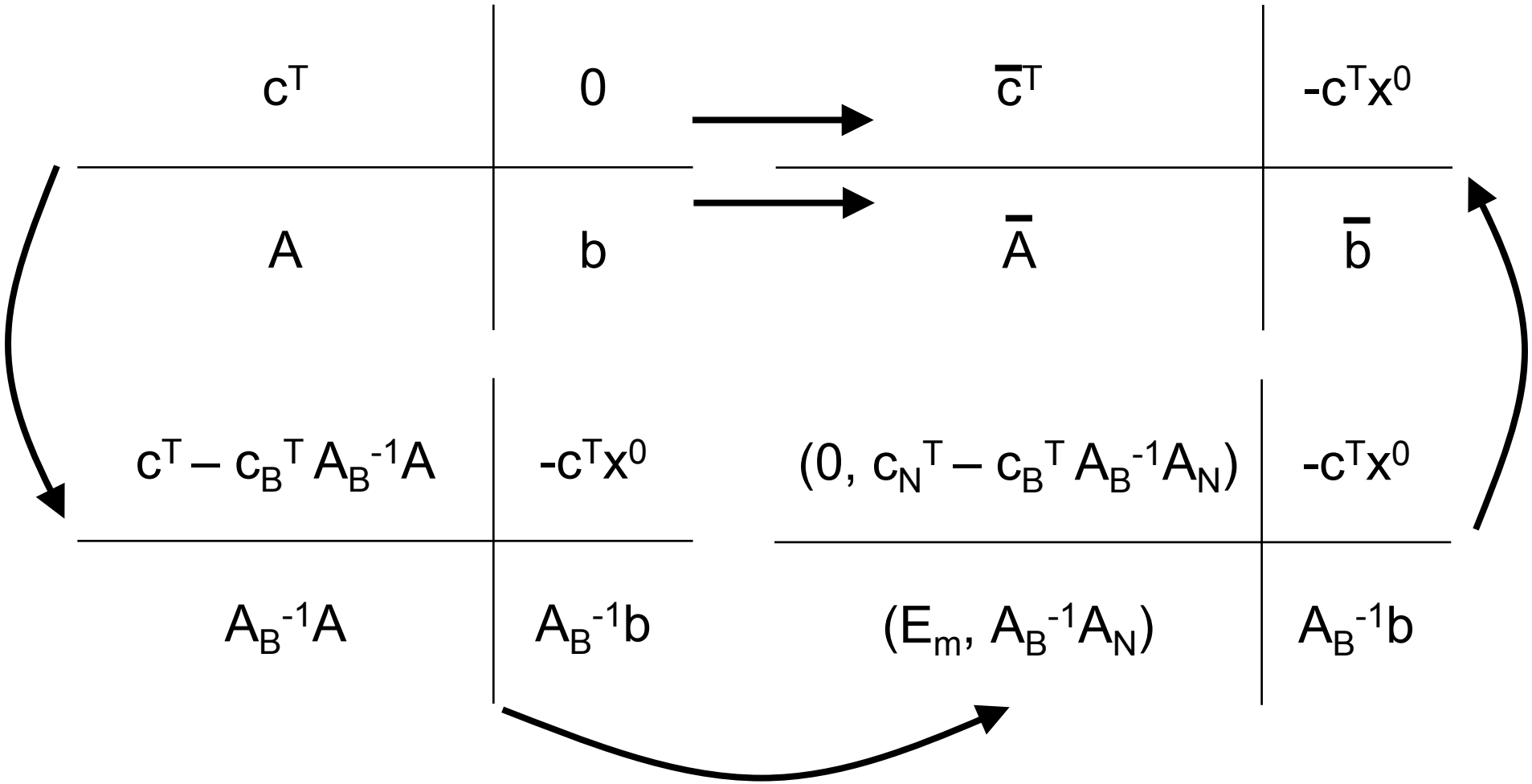
What we did so far

- We have seen how to hop from one corner to another by following the edges of a polyeder, which are given by solutions of the homogenous equation system.
- We have introduced the notion of relative costs that tell us whether following an edge to a neighboring corner will yield to an objective function decrease or not.
- Now let us formulate an algorithm for solving any given linear programming problem!

The Simplex Form

- We have seen that different forms of linear programs are “equivalent”:
  - Standard Form
  - Canonical Form
  - LPs that contain unbounded variables
- We consider exclusively LPs in Standard Form for which we sketch an algorithm in the following.
  - Min $c^T x$
  - $Ax = b$
  - $x \geq 0$

Our Data-Structure: The Simplex Tableau



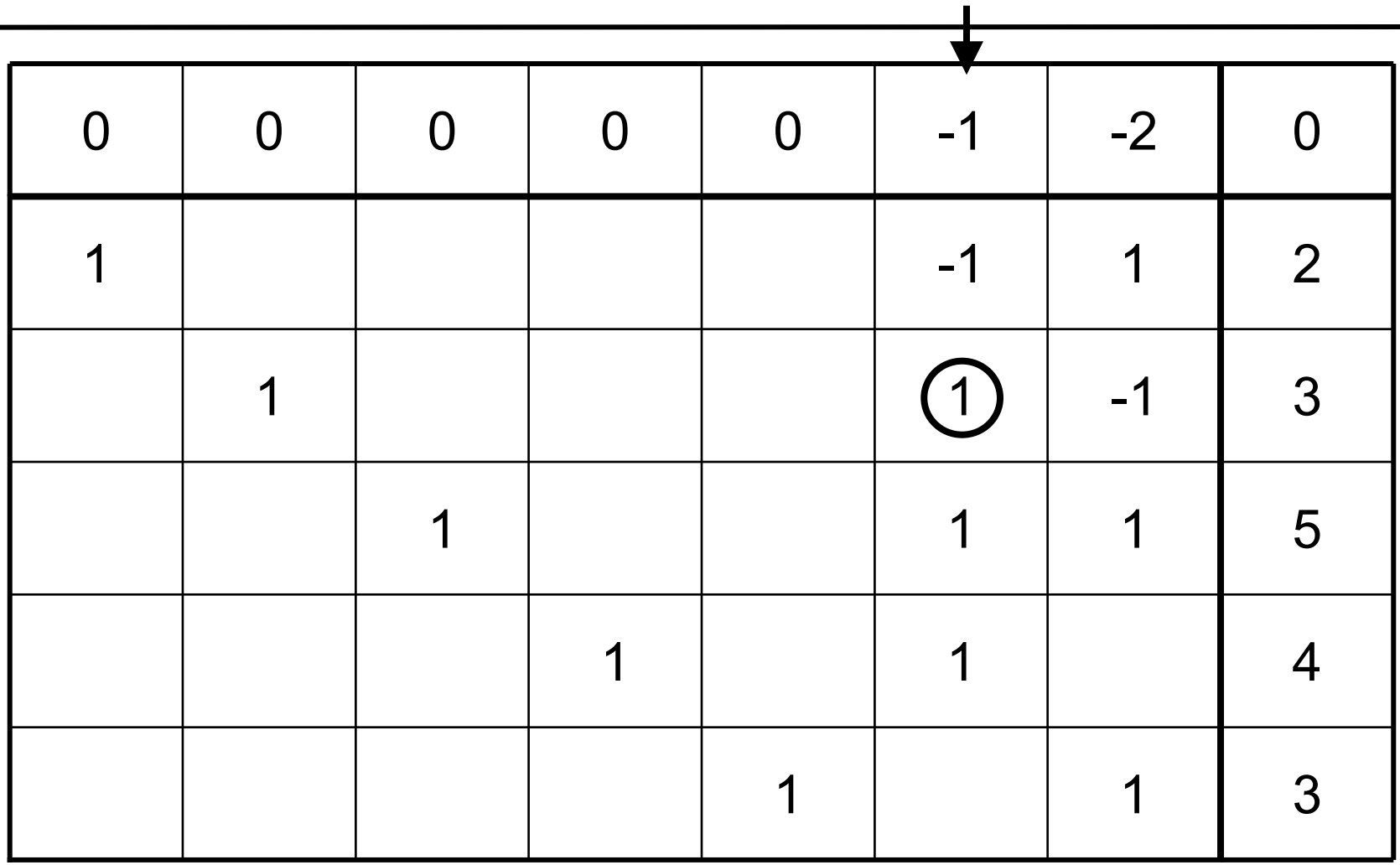
The Simplex-Tableau: A Perspective

- Assume our initial tableau corresponds to a basic feasible solution.
- When performing a basis change, all parts of the tableau change as if a Gaussian elimination step (with pivot element in the A_N-part) was applied on the entire tableau.
- Therefore: By arranging the parts of a linear program (A, b, c, and objective value) in the form of a tableau, we merely facilitate the description of the core optimization step of the simplex algorithm.
- The tableau as a data-structure is more of a software design element than an efficient data-structure in the understanding of algorithmic computer science.

An Instruction List

- Given a tableau corresponding to a basic feasible solution.
- Choose a column t with relative costs $\overline{c}_t < 0$.
- If no such column exists: x^0 is optimal – return.
- Determine i such that
  $\lambda = \overline{b}_i / \overline{a}_i^t = \min \{ \overline{b}_k / \overline{a}_k^t \mid \overline{a}_k^t > 0\}$ (Note: $\overline{b}_i = x^0_{B(i)}$).
- If no such i exists, then the linear optimization problem is unbounded and no optimal solution exists – return.
- Update the tableau by performing a Gaussian elimination step around pivot element $\overline{a}_i^t$.
- Repeat.

An Example

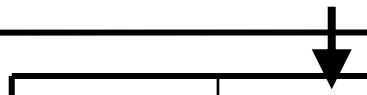


0	0	0	0	0	-1	-2	0
1					-1	1	2
	1				1	-1	3
		1			1	1	5
			1		1		4
				1		1	3

An Example

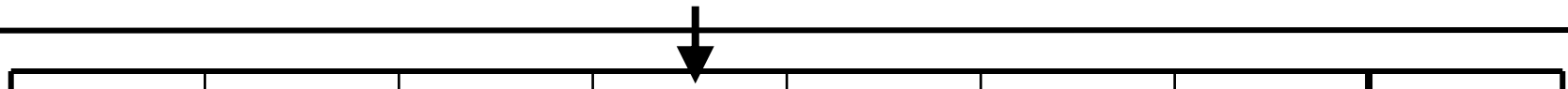
0	1	0	0	0	0	-3	3
1	1					0	5
	1				1	-1	3
	-1	1				2	2
	-1		1			1	1
				1		1	3

An Example



0	-2	0	3	0	0	0	6
1	1		0				5
	0		1		1		4
	(1)	1	-2				0
	-1		1			1	1
	1		-1	1			2

An Example



0	0	2	-1	0	0	0	6
1		-1	2				5
		0	1		1		4
	1	1	-2				0
		1	-1			1	1
		-1	1	1			2

An Example

0	0	1	0	1	0	0	8
1		1		-2			1
		1		-1	1		2
	1	-1		2			4
		0		1		1	3
		-1	1	1			2

Is Our Instruction List an Algorithm that Solves LP?

- Termination?
- Precondition: Finding a basic solution is easy (Gaussian elimination) – but finding a basic feasible solution?
- What is the worst-case runtime?

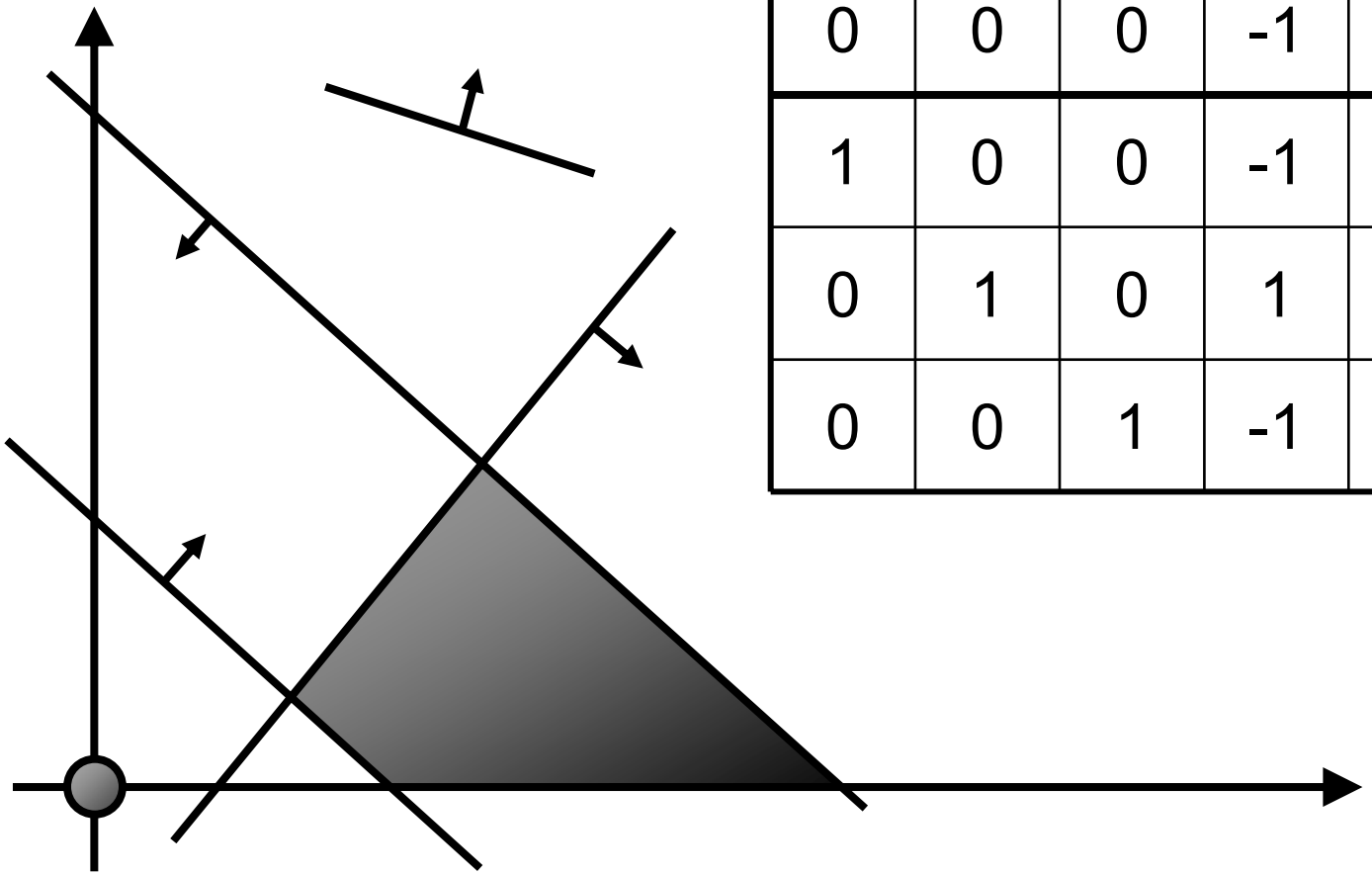
Termination

- We have some choices to make that are arbitrary so far:
  - If there is more than one column t with negative relative costs, which one should we choose?
  - If there is more than one row i for which λ is minimal, which one should we choose?
- Bland's Anticycling Algorithm
  - Choose both t and B(i) minimal.
  - Prove this guarantees termination!

Is Our Instruction List An Algorithm That Solves LP?

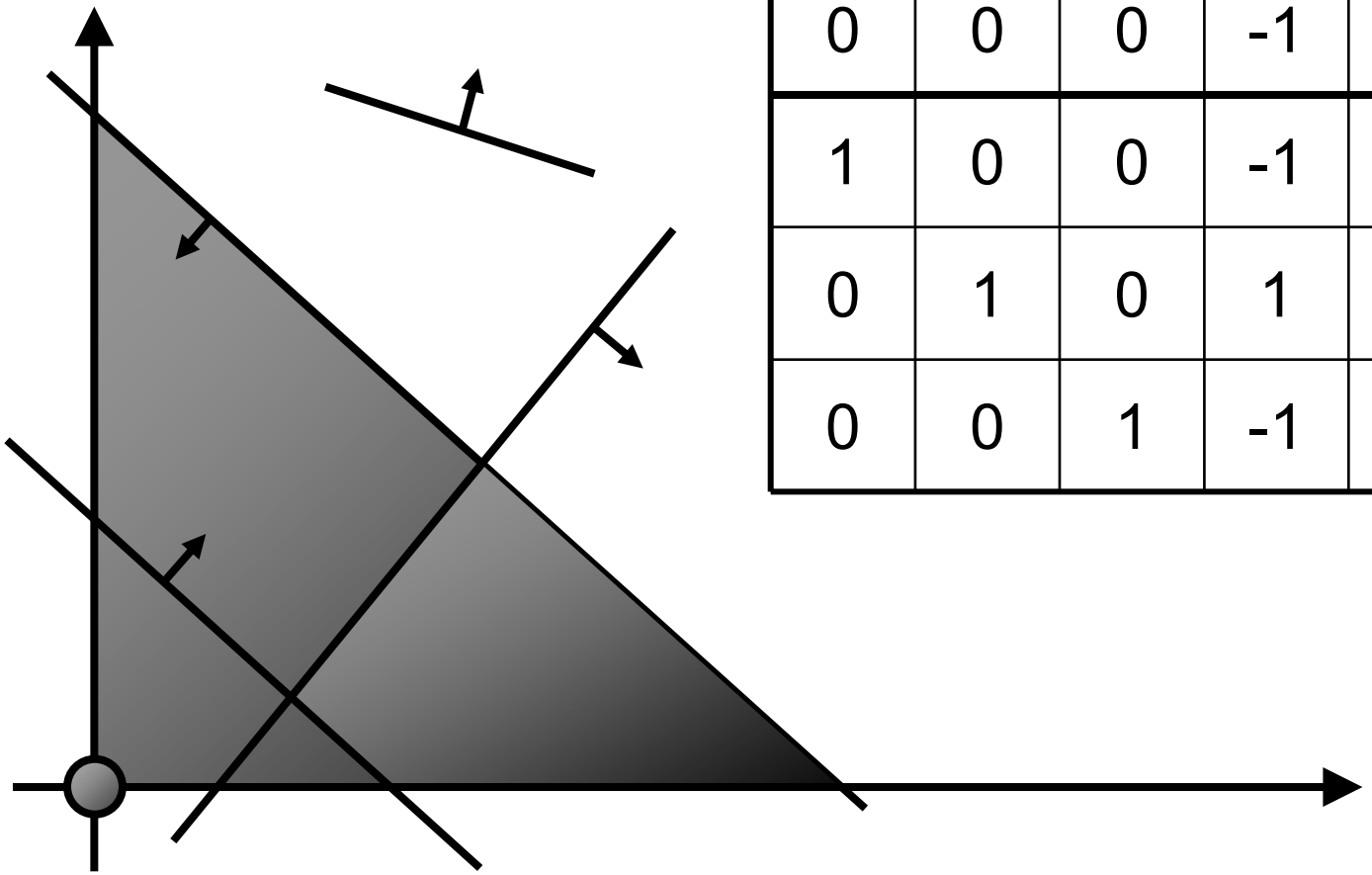
- Termination?
- Precondition: Finding a basic solution is easy (Gaussian elimination) – but finding a basic feasible solution?
- What is the worst-case runtime?

How to find the first basic feasible solution



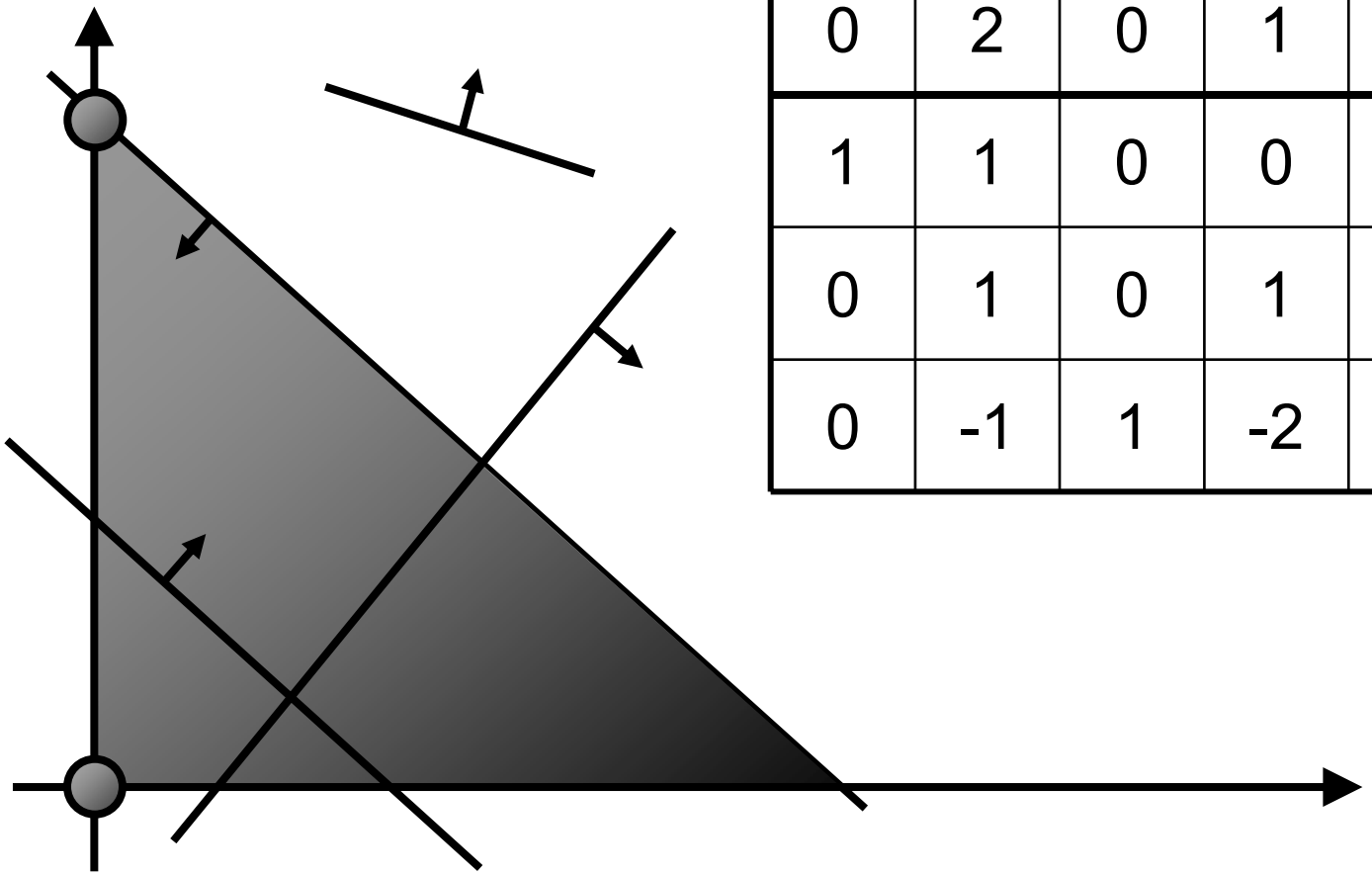
0	0	0	-1	-2	0
1	0	0	-1	-1	-2
0	1	0	1	1	5
0	0	1	-1	1	-1

How to find the first basic feasible solution



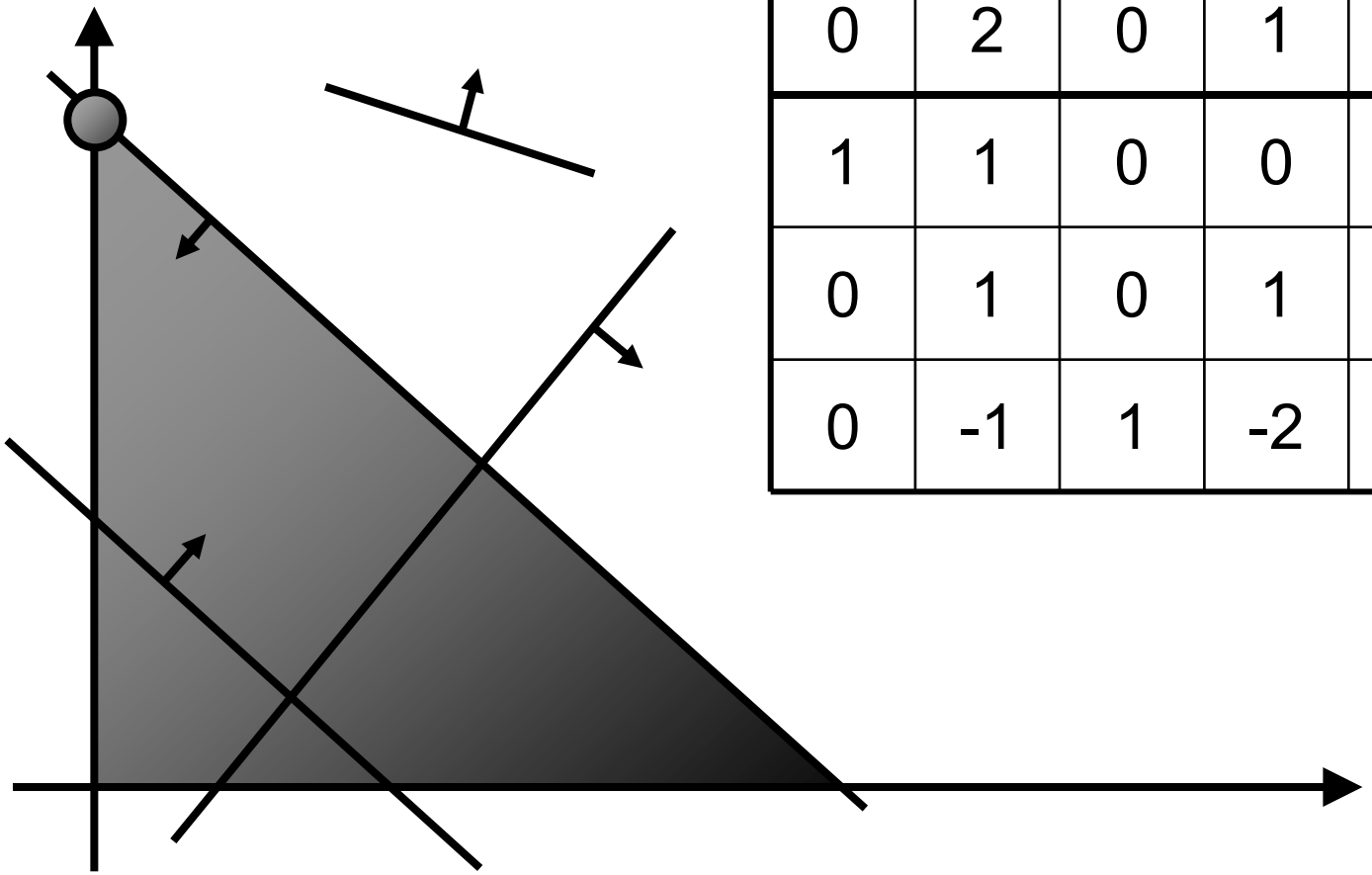
0	0	0	-1	-2	0
1	0	0	-1	-1	-2
0	1	0	1	1	5
0	0	1	-1	1	-1

How to find the first basic feasible solution



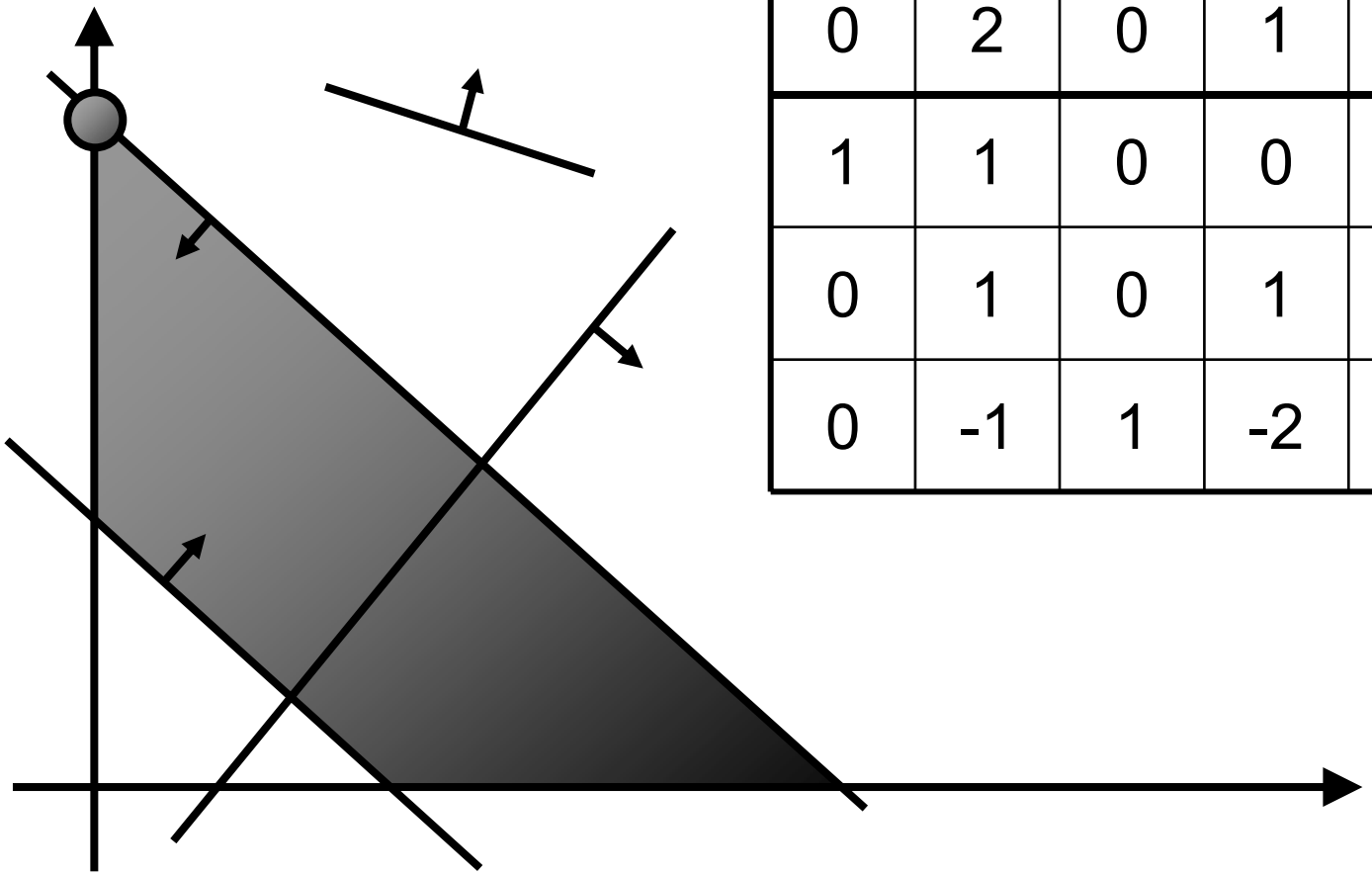
0	2	0	1	0	10
1	1	0	0	0	3
0	1	0	1	1	5
0	-1	1	-2	0	-6

How to find the first basic feasible solution



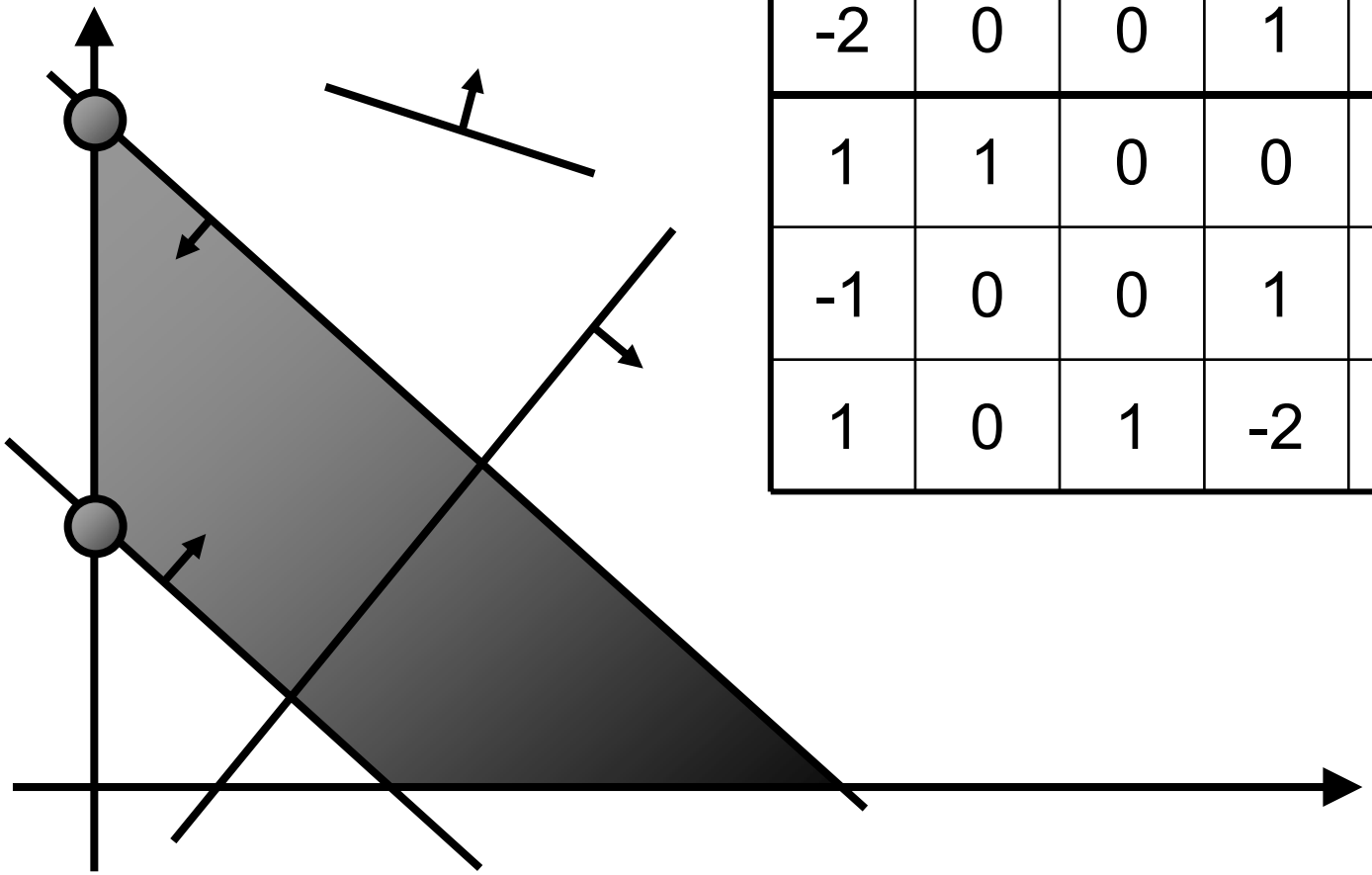
0	2	0	1	0	10
1	1	0	0	0	3
0	1	0	1	1	5
0	-1	1	-2	0	-6

How to find the first basic feasible solution



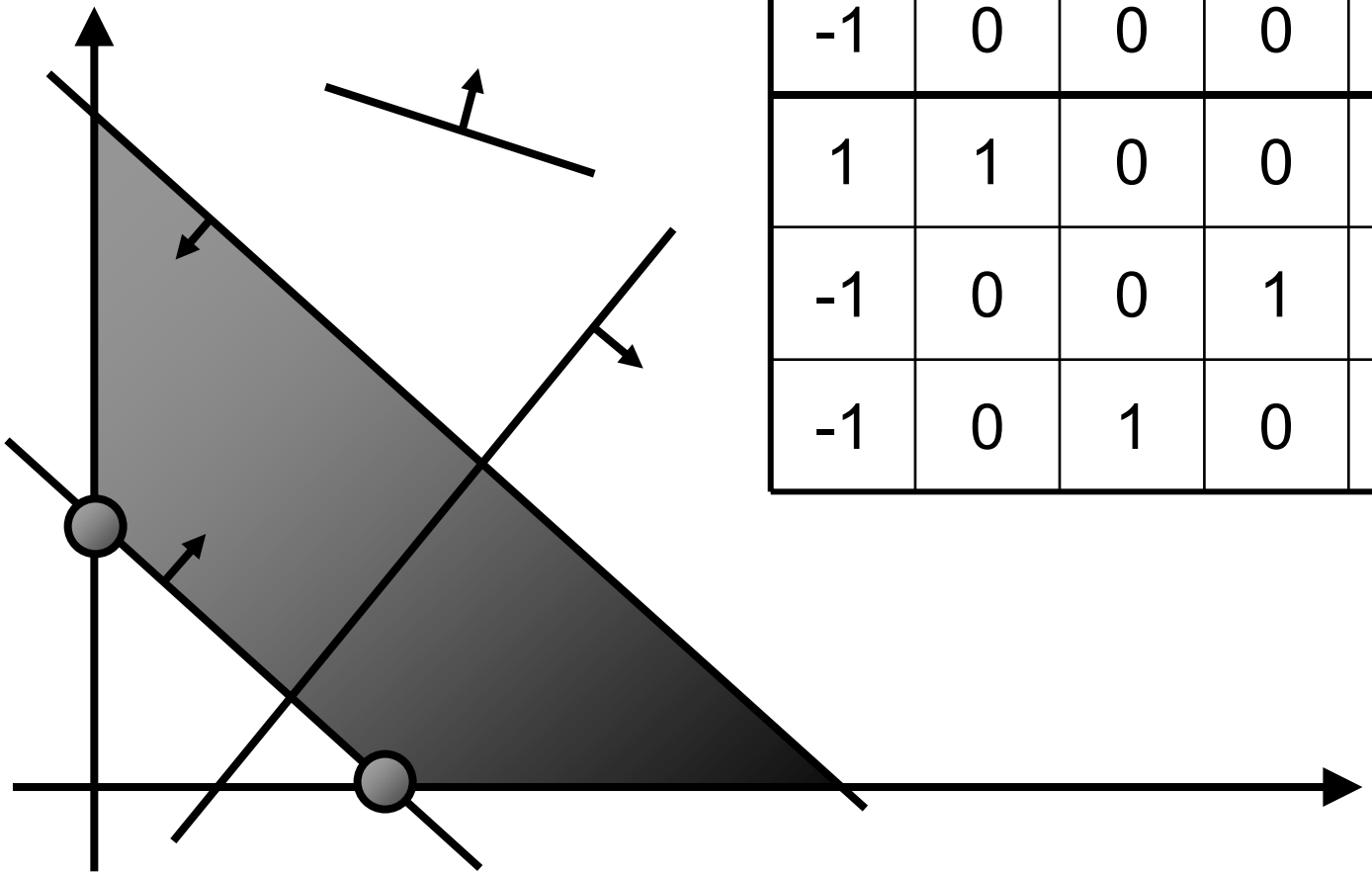
0	2	0	1	0	10
1	1	0	0	0	3
0	1	0	1	1	5
0	-1	1	-2	0	-6

How to find the first basic feasible solution



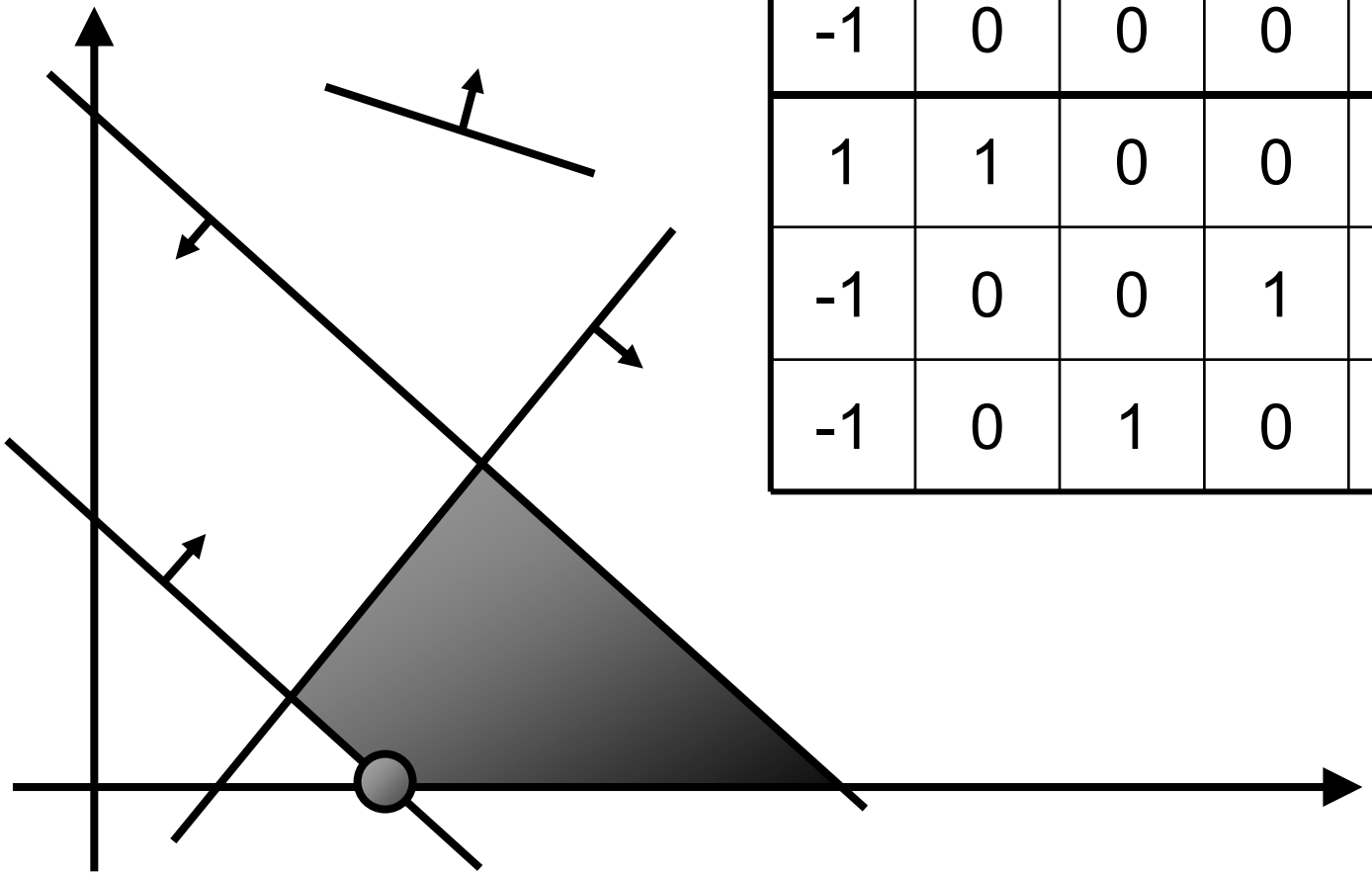
-2	0	0	1	0	4
1	1	0	0	0	3
-1	0	0	1	1	2
1	0	1	-2	0	-3

How to find the first basic feasible solution



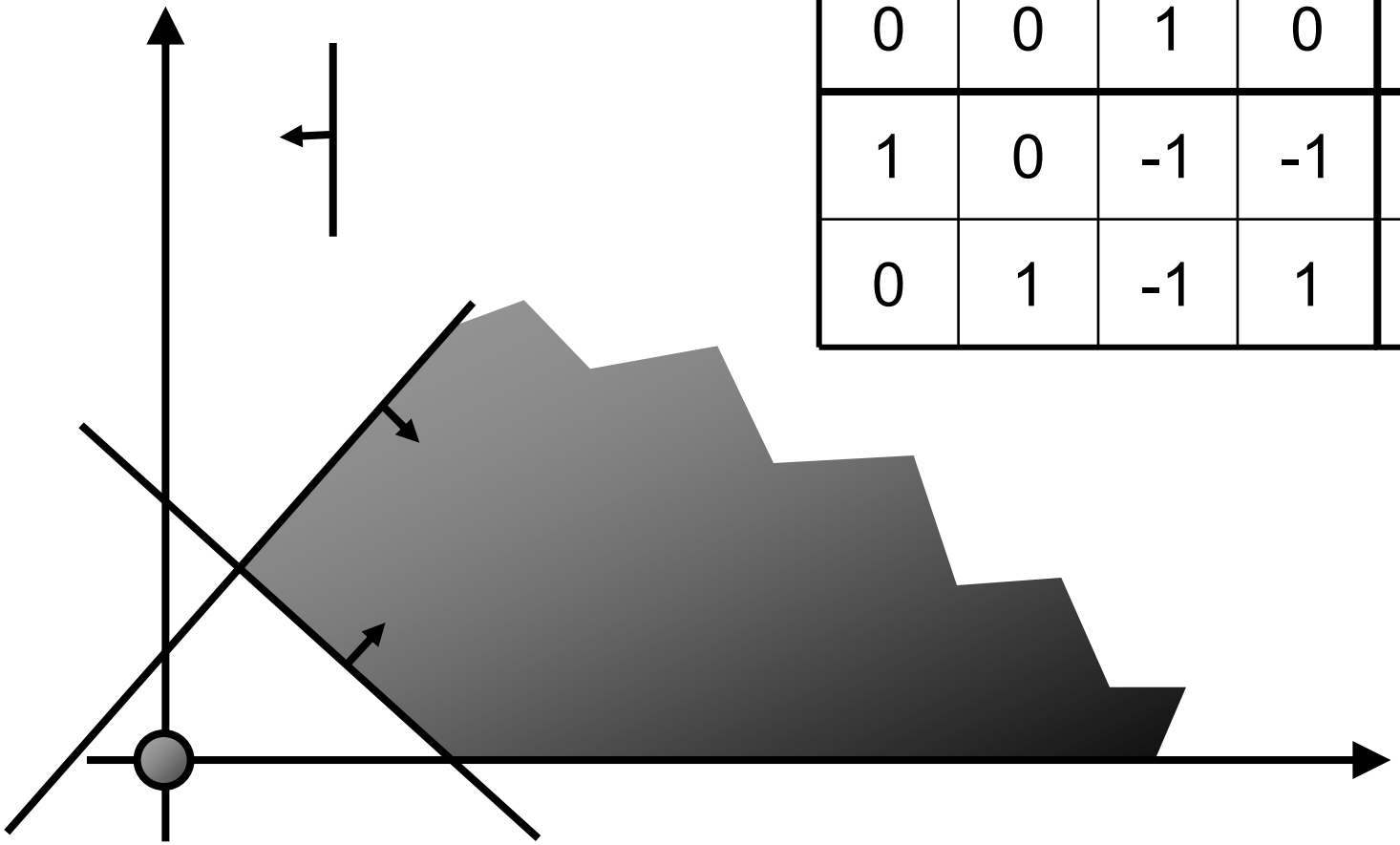
-1	0	0	0	-1	2
1	1	0	0	0	3
-1	0	0	1	1	2
-1	0	1	0	2	1

How to find the first basic feasible solution



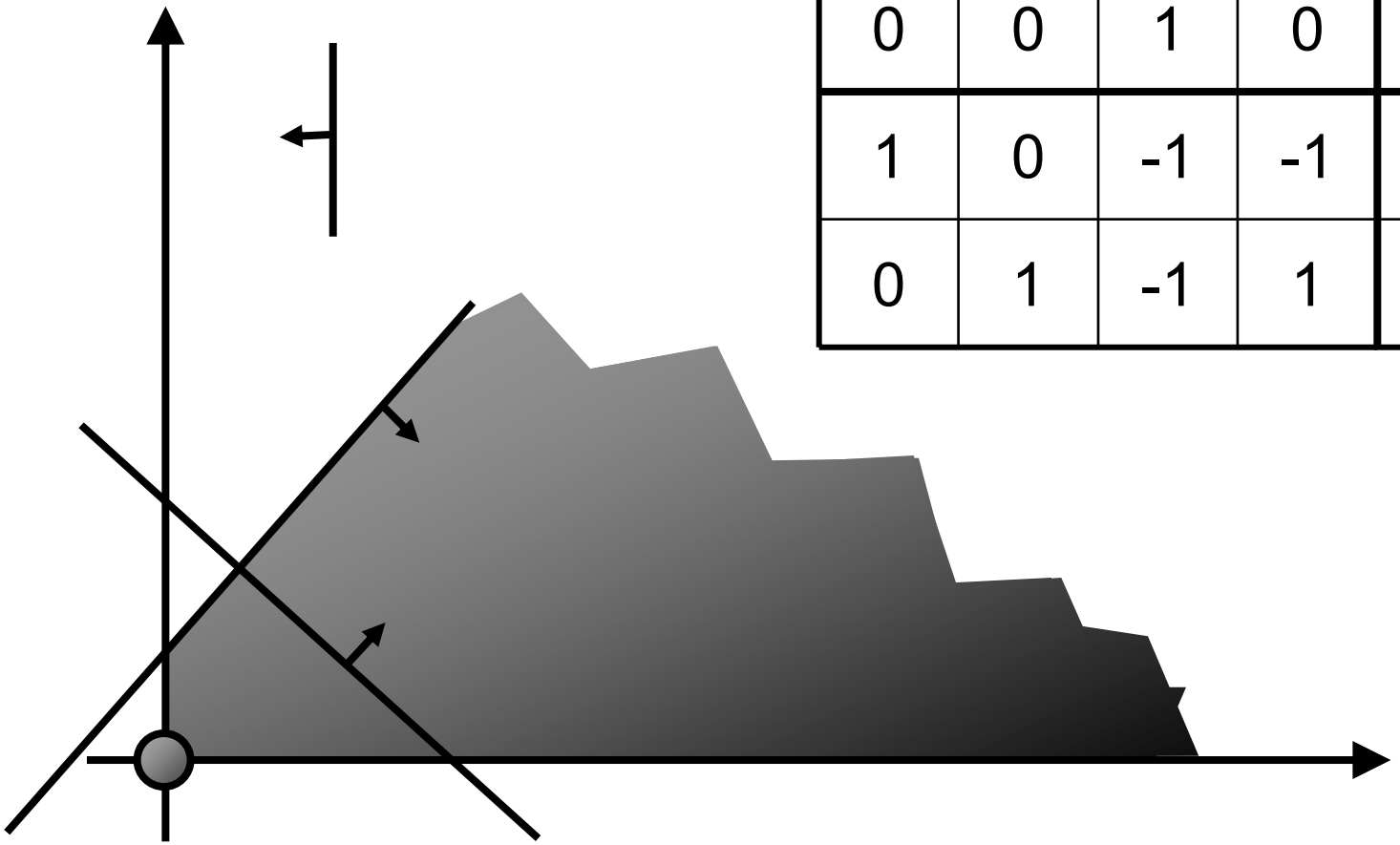
-1	0	0	0	-1	2
1	1	0	0	0	3
-1	0	0	1	1	2
-1	0	1	0	2	1

How to find the first basic feasible solution



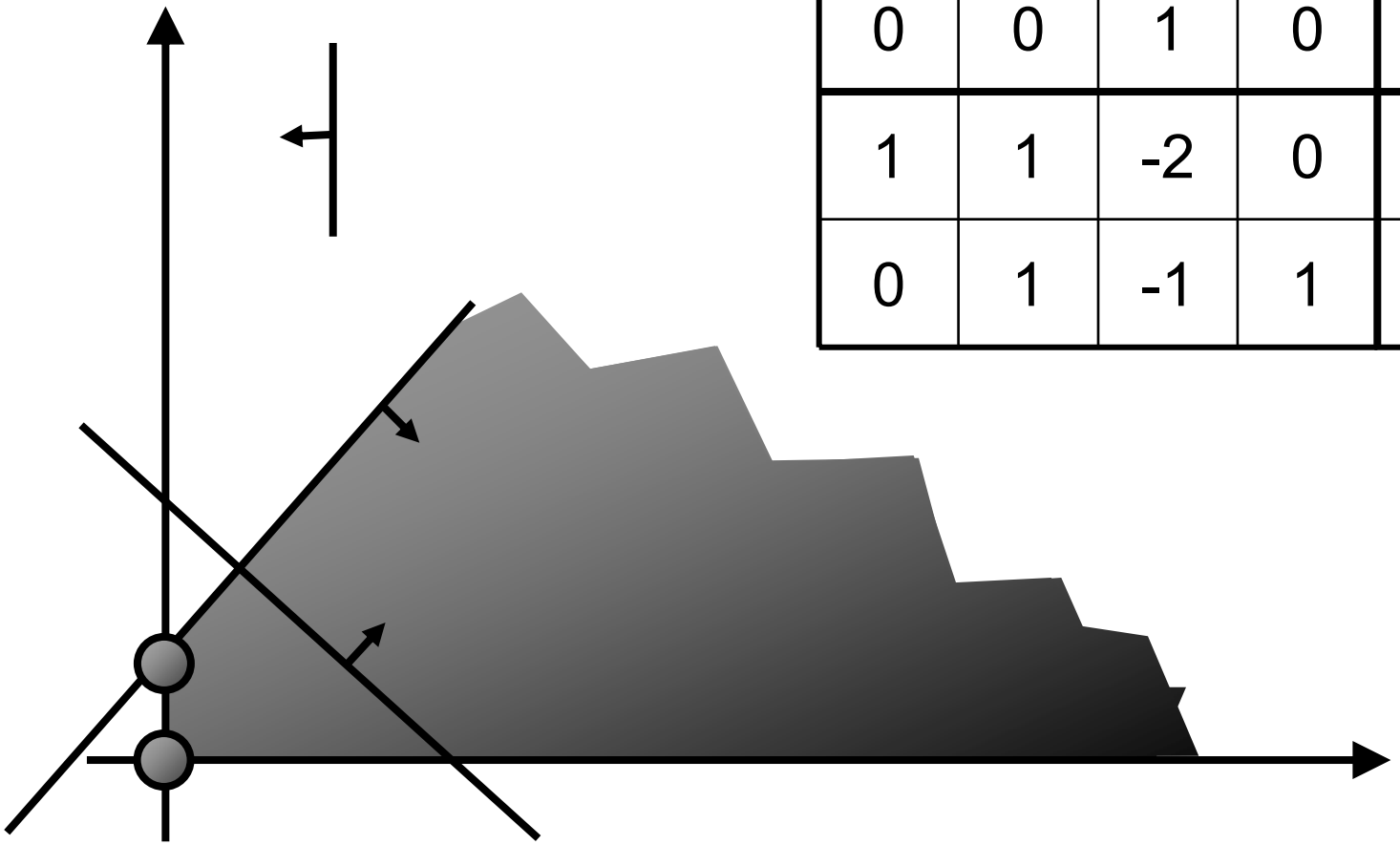
0	0	1	0	0
1	0	-1	-1	-2
0	1	-1	1	1

How to find the first basic feasible solution



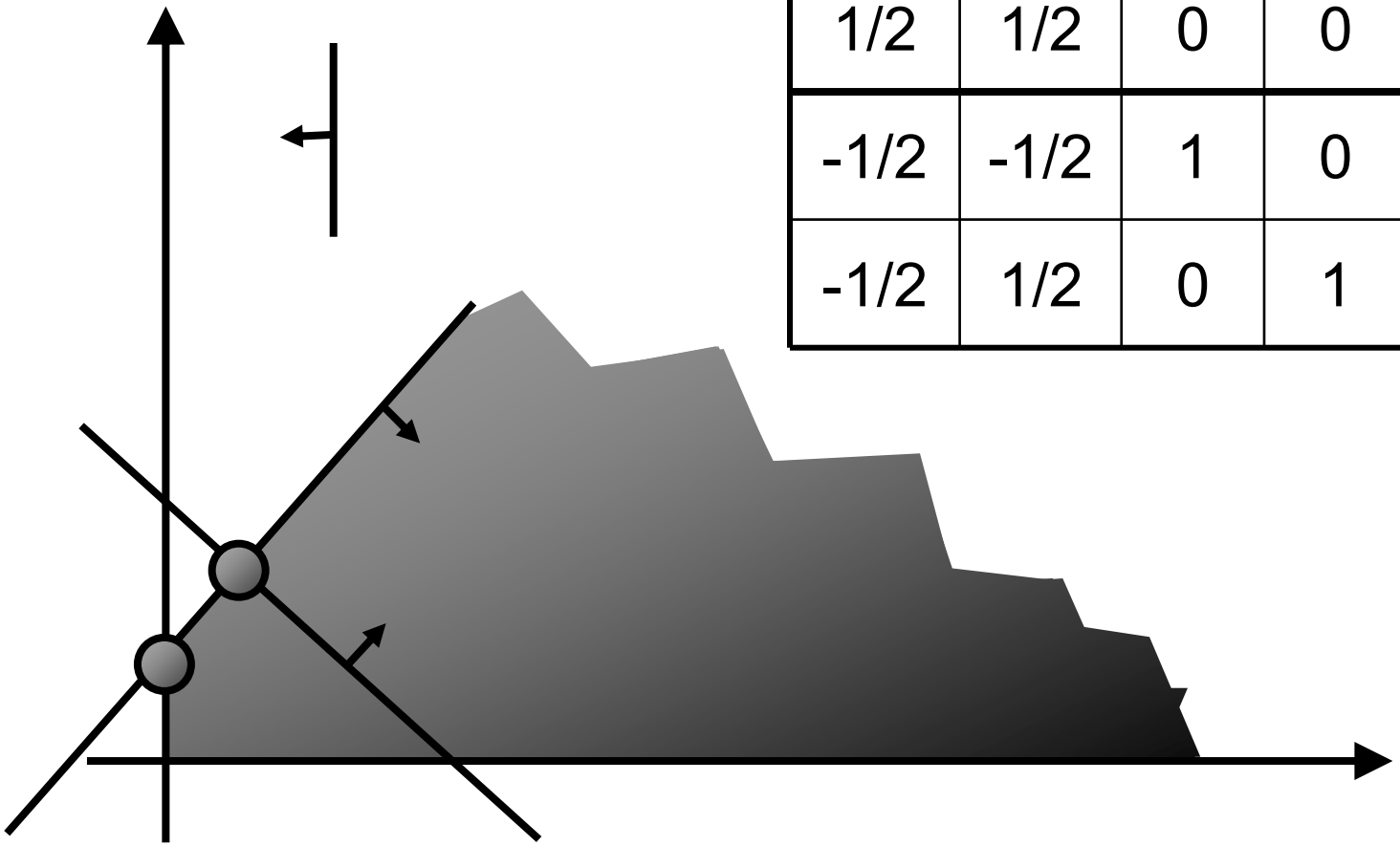
0	0	1	0	0
1	0	-1	-1	-2
0	1	-1	1	1

How to find the first basic feasible solution



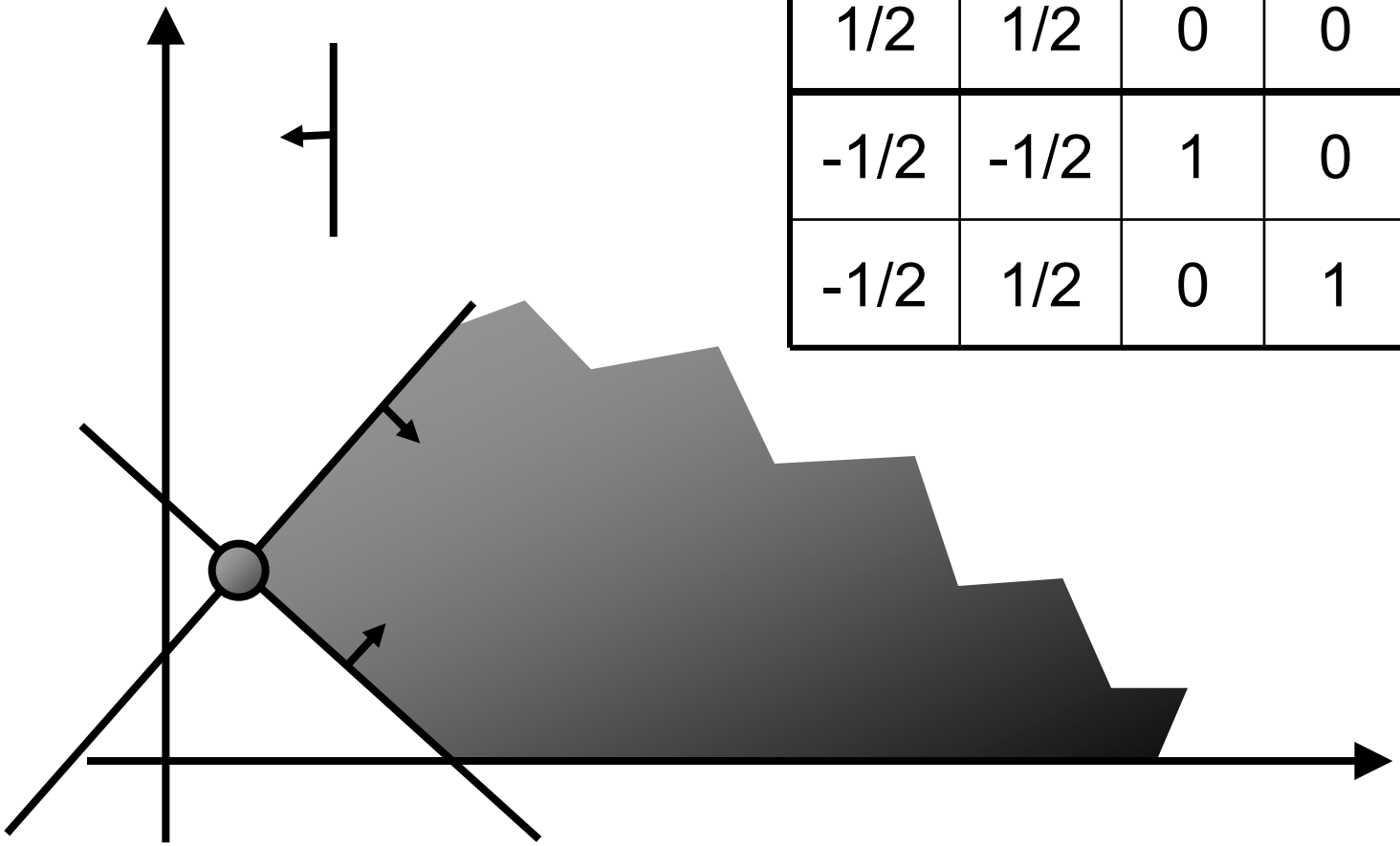
0	0	1	0	0
1	1	-2	0	-1
0	1	-1	1	1

How to find the first basic feasible solution



1/2	1/2	0	0	-1/2
-1/2	-1/2	1	0	1/2
-1/2	1/2	0	1	3/2

How to find the first basic feasible solution



1/2	1/2	0	0	-1/2
-1/2	-1/2	1	0	1/2
-1/2	1/2	0	1	3/2

How to find the first basic feasible solution

0	0	1	0	0
1	0	-1	-1	-2
0	1	-1	1	1

1	1	0	0	0	0	0
1	0	-1	0	1	1	2
0	1	0	1	-1	1	1

0	0	1	-1	0	-2	-3
1	0	-1	0	1	1	2
0	1	0	1	-1	1	1

0	2	1	1	-2	0	-1
1	-1	-1	-1	2	0	1
0	1	0	1	-1	1	1

1	1	0	0	0	0	0
1/2	-1/2	-1/2	-1/2	1	0	1/2
1/2	1/2	-1/2	1/2	0	1	3/2

How to find the first basic feasible solution

1	1	0	0	0	0	0
1/2	-1/2	-1/2	-1/2	1	0	1/2
1/2	1/2	-1/2	1/2	0	1	3/2

0	0	1	0	0
-1/2	-1/2	1	0	1/2
-1/2	1/2	0	1	3/2

1/2	1/2	0	0	-1/2
-1/2	-1/2	1	0	1/2
-1/2	1/2	0	1	3/2

Is Our Instruction List An Algorithm That Solves LP?

- Termination?
- Precondition: Finding a basic solution is easy (Gaussian elimination) – but finding a basic feasible solution?
- What is the worst-case runtime?

Worst-Case Runtime

- There are at most $\binom{n}{m}$ basic solutions. Consequently, since we never repeat a bfs, we terminate. $\Rightarrow$ We have an algorithm! ☺
- Because of the optimality condition, it is called: The Simplex Algorithm.
- But: the runtime is not polynomial!
- Note: Linear Programming Problems can be solved in polynomial time – but the simplex algorithm does not achieve that.

Average Case Runtime

- What is an “average” instance?
- Under some appropriate distribution assumptions, it can be shown that the simplex algorithm needs $O(n^4)$ steps, each of which takes $O(nm)$.
- In practice, for real-world instances, we measure an average of $\leq 3\ (n-m)$ steps.

Thank you!