OPTIMIZATION TECHNIQUES

Linear and quadratic programming

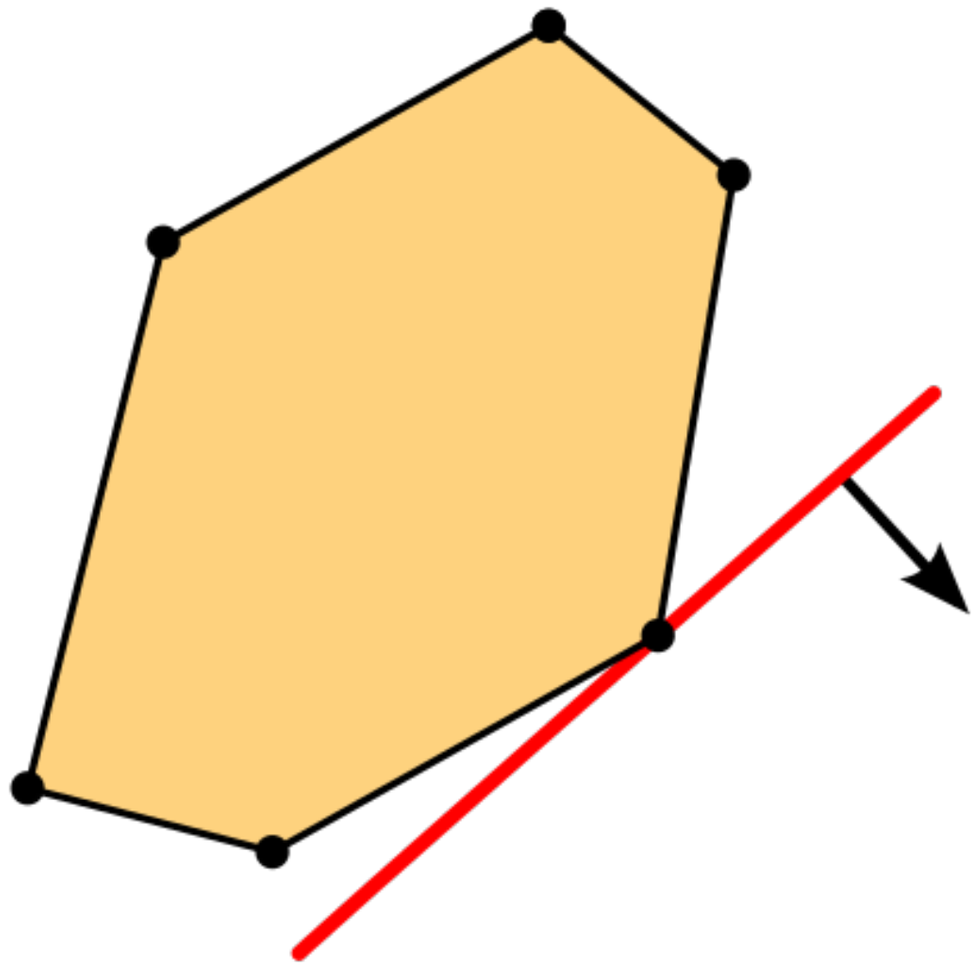
Prof. Giovanni Iacca

giovanni.iacca@unitn.it

UNIVERSITY OF TRENTO - Italy

Information Engineering
and Computer Science Department

Linear programming



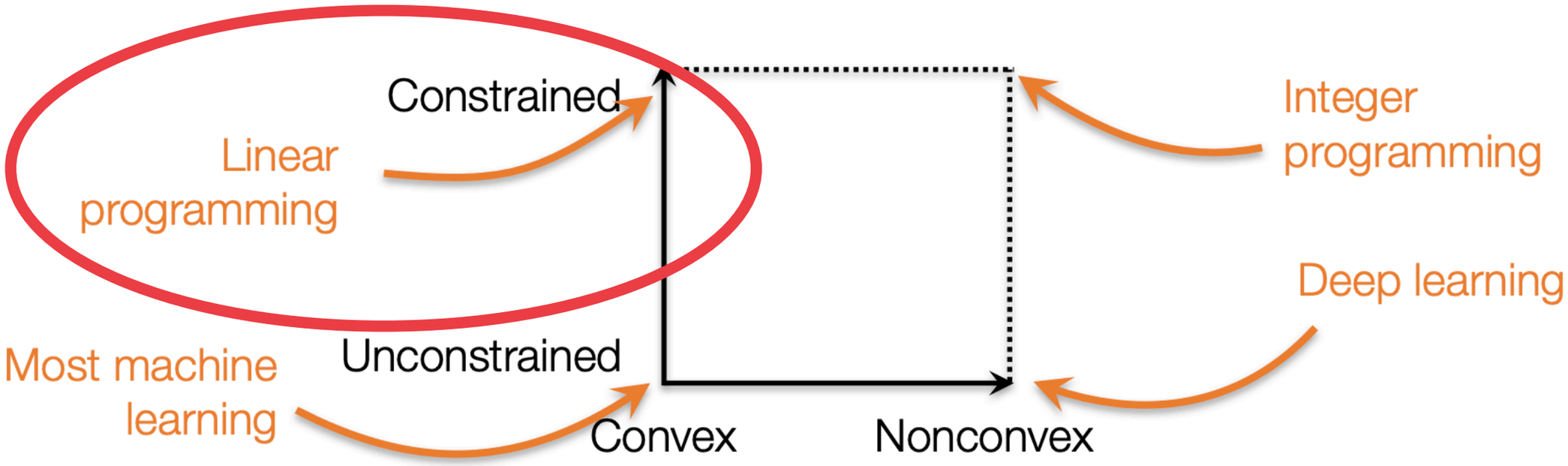
OPTIMIZATION

ONE WORD FOR SEVERAL MEANINGS

Name	Vars	Constraints	Objective
Constraint Programming (CP)	discrete	any	N/A
Linear Programming (LP)	real	linear inequalities	linear function
Integer (Linear) Programming (IP/ILP)	integer	linear inequalities	linear function
Mixed Integer (linear) Programming (MIP/MILP)	integer & real	linear inequalities	linear function
Semidefinite Programming (SDP)	real	linear inequalities + semidefiniteness	linear function
Quadratic Programming (QP)	real	linear inequalities	quadratic function
Quadratically Constrained Quadratic Programming (QCQP)	real	quadratic inequalities	quadratic function
Convex Programming	real	convex region	convex function
Nonlinear Programming (NLP)	real	any	any

OPTIMIZATION

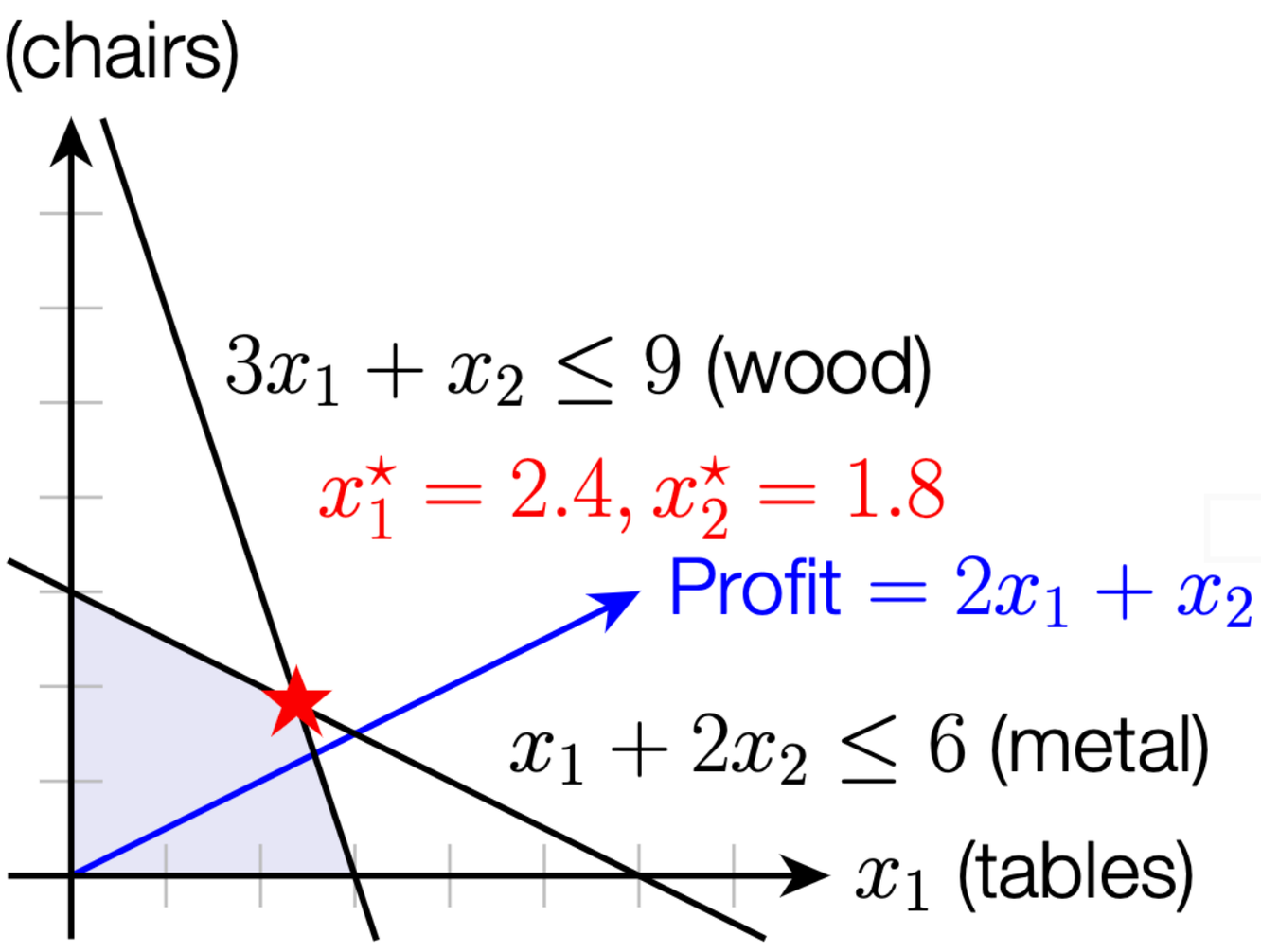
ONE WORD FOR SEVERAL MEANINGS



LINEAR PROGRAMMING

EXAMPLE: MANUFACTURING

A large factory makes tables and chairs. Each table returns a profit of 200 EUR and each chair a profit of 100 EUR. Each table takes 1 unit of metal and 3 units of wood and each chair takes 2 units of metal and 1 unit of wood. The factory has 600 units of metal and 900 units of wood.
How many tables and chairs should the factory make to maximize profit?

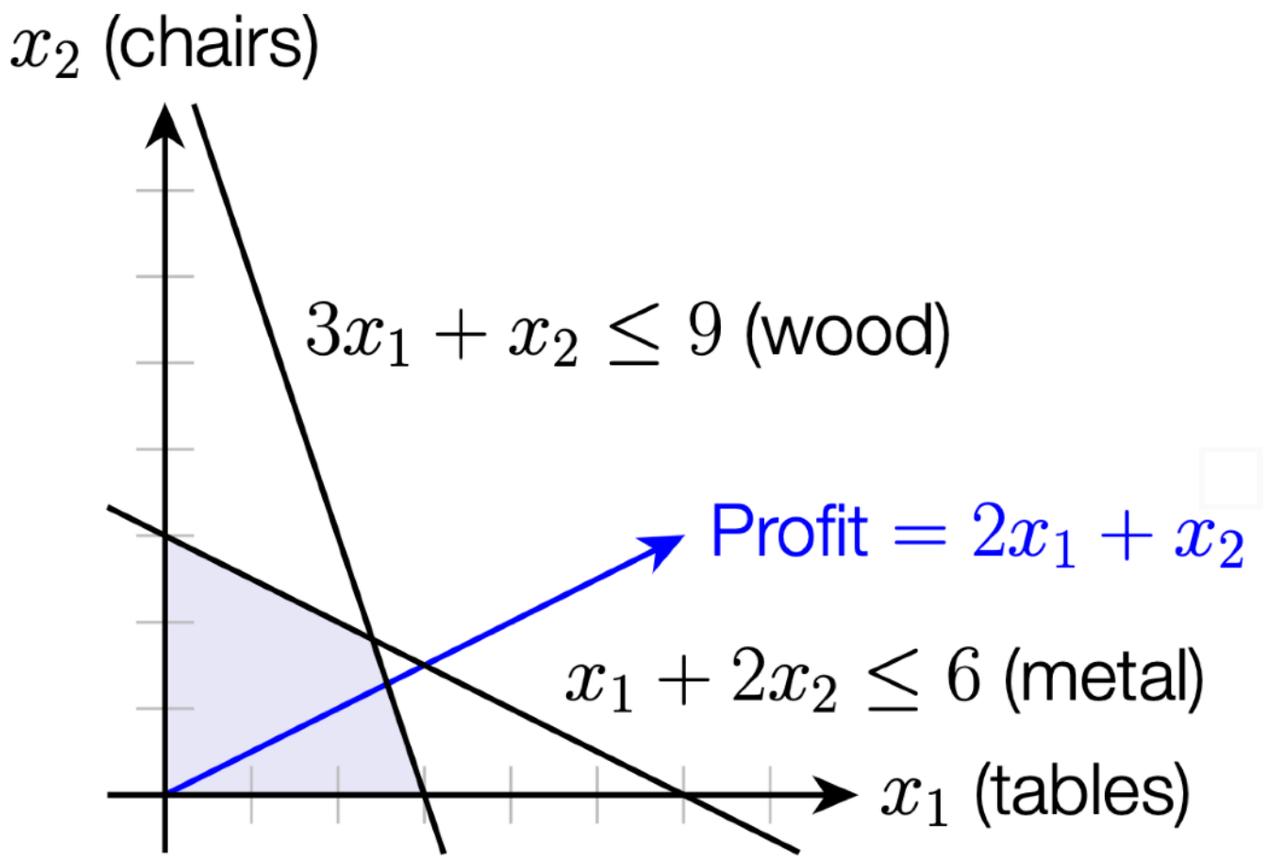


NOTE: x(1) and *x*(2) here are expressed in hundreds

LINEAR PROGRAMMING

EXAMPLE: MANUFACTURING

We can write our manufacturing problem formally as:



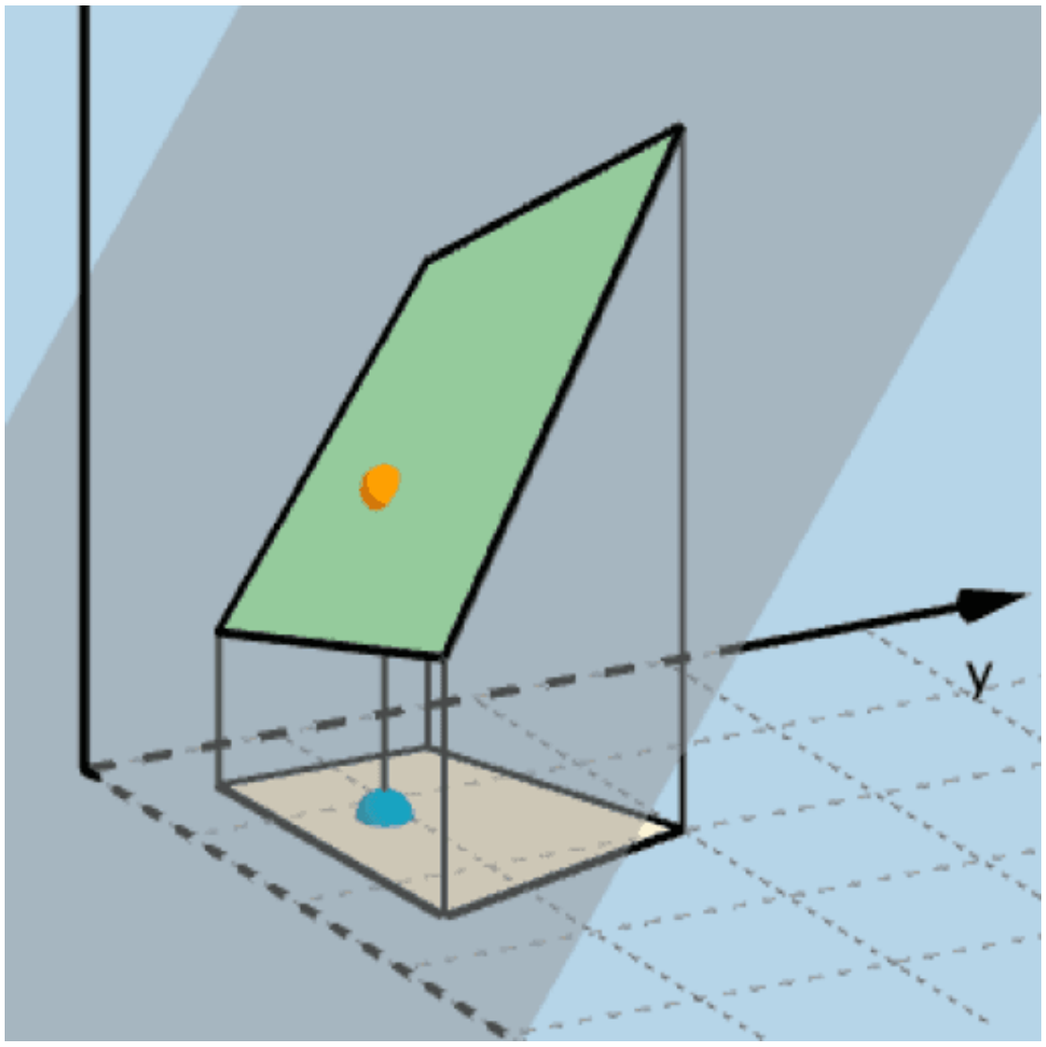
$$\begin{aligned} \underset{x_1, x_2}{\text{maximize}} \quad & 2x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

LINEAR PROGRAMMING

WHY "LINEAR PROGRAMMING"?

Linear: the objective functions and constraints are linear.

Programming: does not refer to computer programming but rather "planning" - planning of activities to obtain an optimal result i.e., it reaches the specified goal in the best possible way (according to the given mathematical model) among all feasible alternatives.



LINEAR PROGRAMMING

INEQUALITY FORM (A.K.A. CANONICAL FORM, OR STANDARD FORM)

Using linear algebra notation, we can write problems compactly as

$$\begin{aligned} \underset{x}{\text{maximize}} \;\; & c^T x \\ \text{subject to} \;\; & Gx \leq h \end{aligned}$$

with optimization variable $x \in \mathbb{R}^n$, problem data $c \in \mathbb{R}^n, G \in \mathbb{R}^{m \times n}$, $h \in \mathbb{R}^m$ and where $\leq$ denotes elementwise inequality

A convex optimization problem (objective is affine, constraints are convex)

Our example:

$$\begin{aligned} \underset{x_1, x_2}{\text{maximize}} \;\; & 2x_1 + x_2 \\ \text{subject to} \;\; & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned} \quad \Longleftrightarrow \quad c = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, G = \begin{bmatrix} 3 & 1 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}, h = \begin{bmatrix} 9 \\ 6 \\ 0 \\ 0 \end{bmatrix}$$

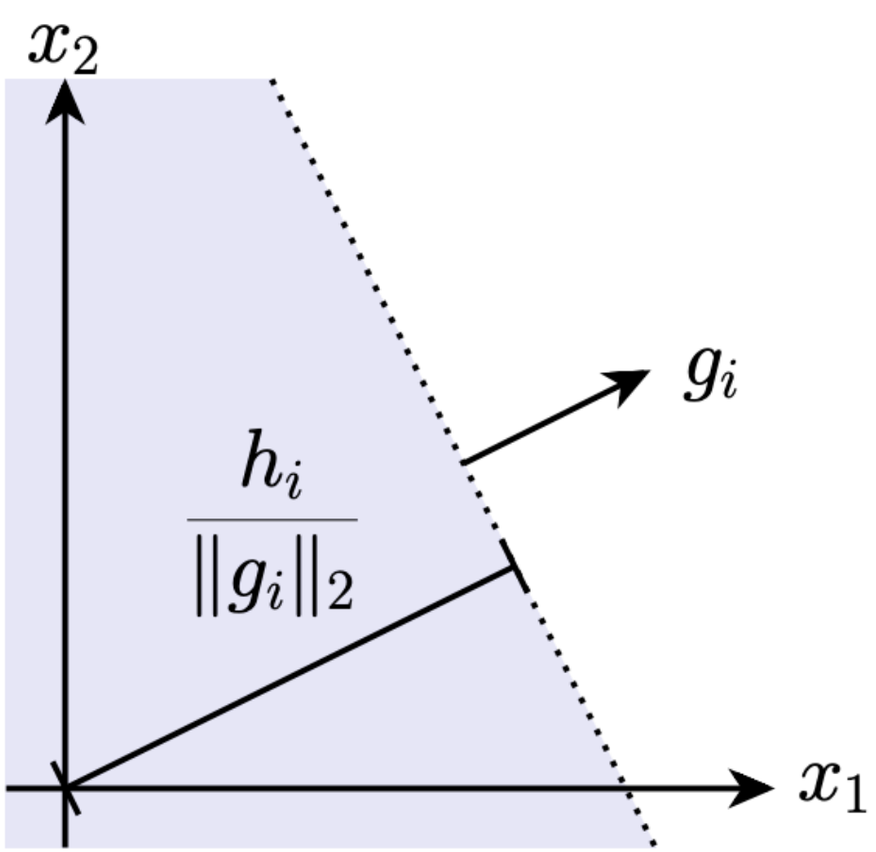
LINEAR PROGRAMMING

GEOMETRY OF LINEAR PROGRAMS

Consider a single row of the constraint matrix:

$$g_i^T x \leq h_i$$

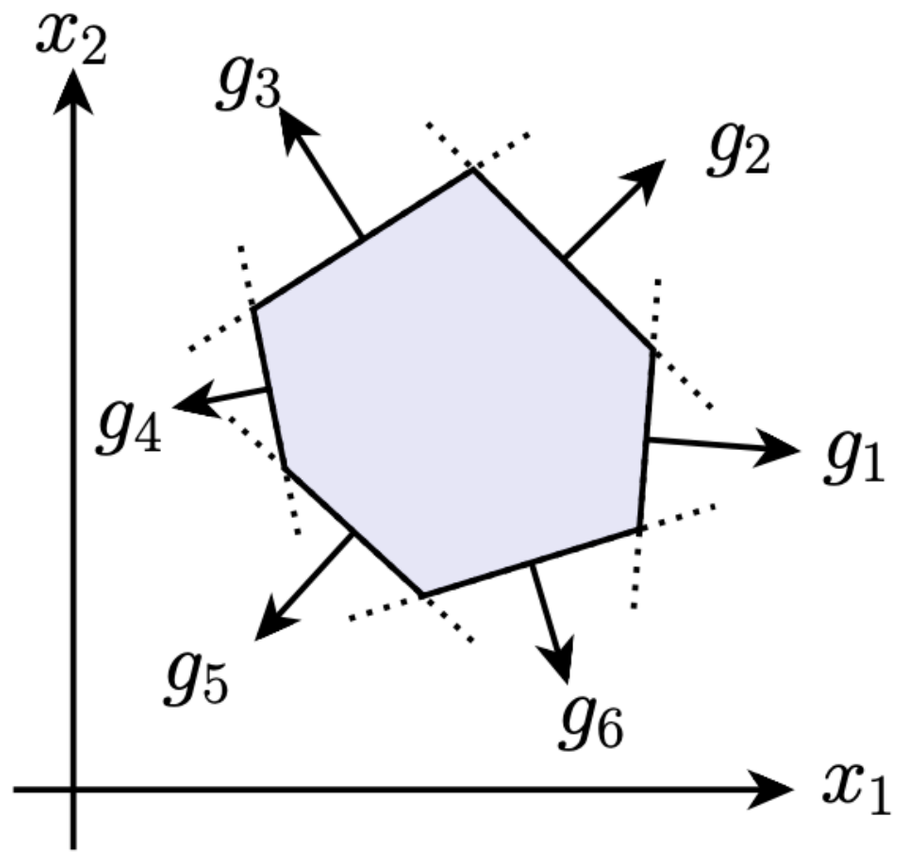
This represents a **halfspace** constraint.



LINEAR PROGRAMMING

LINEAR POLYTOPE

Multiple halfspace constraints $Gx \leq h$ (i.e., $g_i^T x \leq h_i, i = 1, \ldots, m$) define what is called a **polytope** (i.e., an n-dimensional generalization of a 2D polygon, or a 3D polyhedron):

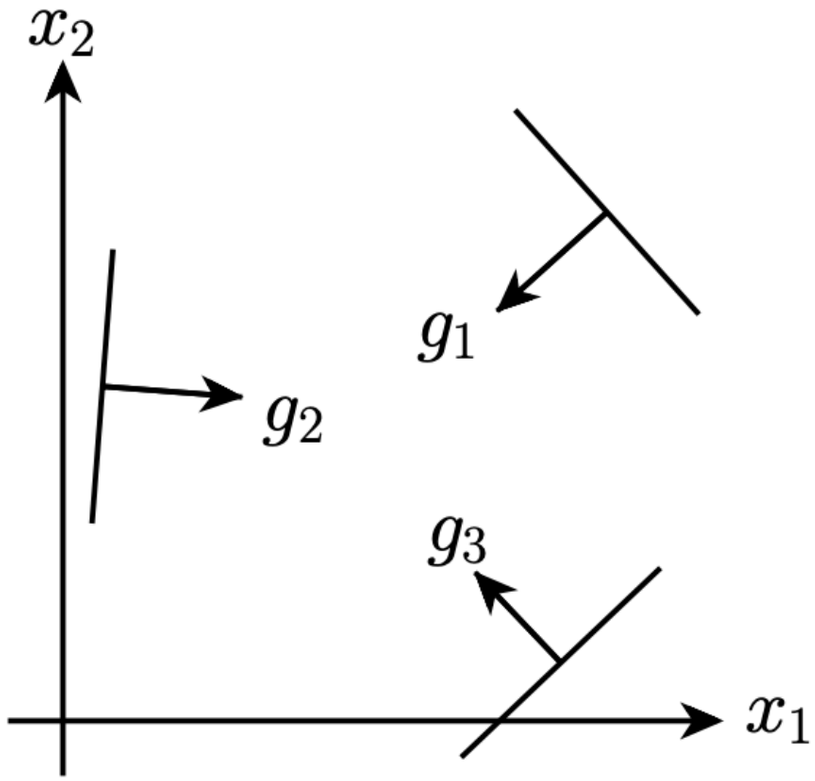


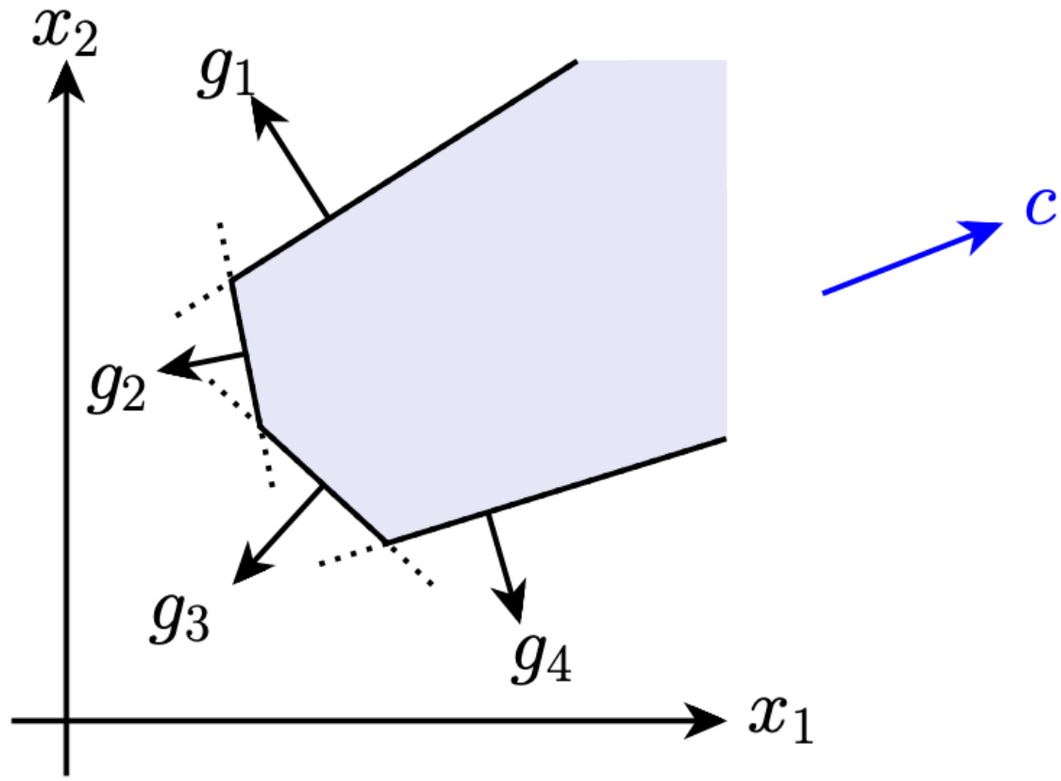
So linear programming in equivalent to **maximizing some direction over this polytope** (note that **optimum will always occur on a corner**).

LINEAR PROGRAMMING

INFEASIBLE OR UNBOUNDED POLYTOPES

Polytopes may be infeasible or unbounded, corresponding to having no solution or potentially an unbounded solution for the linear program.





LINEAR PROGRAMMING

AUGMENTED FORM (A.K.A. SLACK FORM, OR STANDARD FORM)

For the simplex algorithm, we will consider linear programs in an alternative form, known as ***slack form***:

$$\begin{aligned} \underset{x}{\text{minimize}} \;\; & c^T x \\ \text{subject to} \;\; & Ax = b \\ & x \geq 0 \end{aligned}$$

with optimization variable $x \in \mathbb{R}^n$, and problem data $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ (note: m, n are not related to previous sizes)

Looks different, but it is straightforward to convert between inequality form and standard form by adding *slack variables*

$$g_i^T x \leq h_i \implies g_i^T x + s_i = h_i, s_i \geq 0$$

Can also separate non-negative variables in positive/negative part

LINEAR PROGRAMMING

CONVERTING TO AUGMENTED FORM

We can convert our example problem as follows:

$$\begin{aligned} \underset{x_1,x_2}{\text{maximize}} \quad & 2x_1 + x_2 \\ \text{subject to} \quad & 3x_1 + x_2 \leq 9 \\ & x_1 + 2x_2 \leq 6 \\ & x_1, x_2 \geq 0 \end{aligned}$$

$$\begin{aligned} \underset{x_1,x_2,x_3,x_4}{\text{minimize}} \quad & -2x_1 - x_2 \\ \text{subject to} \quad & 3x_1 + x_2 + x_3 = 9 \\ & x_1 + 2x_2 + x_4 = 6 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

$$c = \begin{bmatrix} -2 \\ -1 \end{bmatrix}, A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

LINEAR PROGRAMMING

FINDING CORNERS IN POLYTOPE

In standard form we assume $n > m$ (plus some technical conditions like full row rank), so A is an underdetermined system of linear equations

To find solutions to subsets of these equations, we can select m columns from A, denoted $A_{\mathcal{J}}$ for some set $\mathcal{J} \subset \{1, \ldots, n\}$ with $|\mathcal{J}| = m$, and solve the resulting linear system

$$A_{\mathcal{J}} x_{\mathcal{J}} = b$$

(then set remaining entries of x to zero)

Solutions for which $x \geq 0$, correspond to corners on the polytope

Note: We'll use $A_{\mathcal{J}}$ to denote subselecting columns of A, A_j to denote the jth column of A, and $x_{\mathcal{J}}$ to denote subselecting element of x

LINEAR PROGRAMMING

FINDING CORNERS IN POLYTOPE

Polytope for our example:

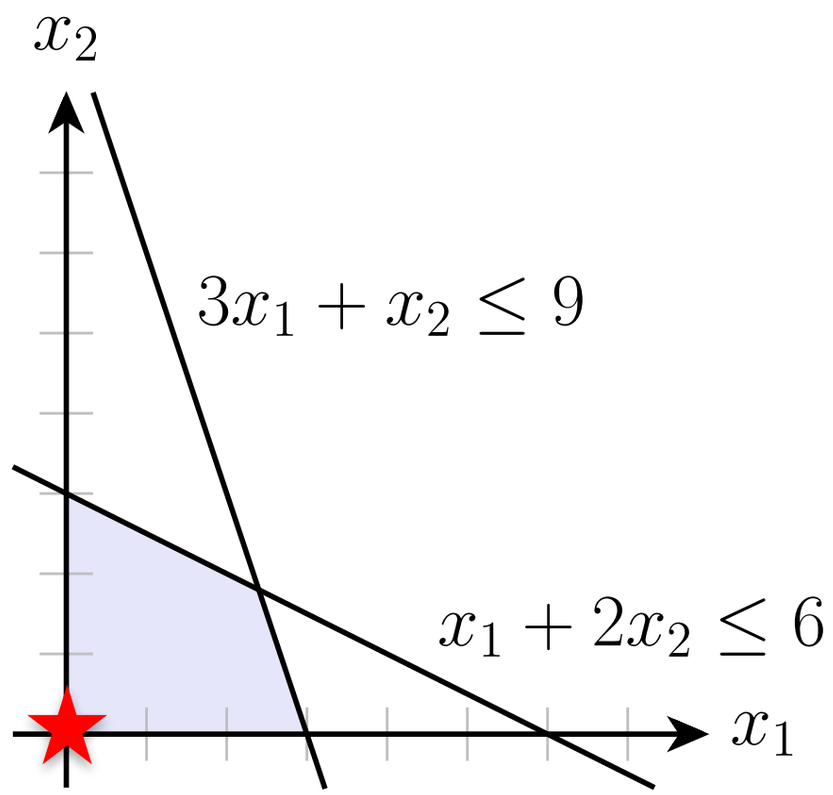
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{3,4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$x = \begin{bmatrix} 0 \\ 0 \\ 9 \\ 6 \end{bmatrix}$$



LINEAR PROGRAMMING

FINDING CORNERS IN POLYTOPE

Polytope for our example:

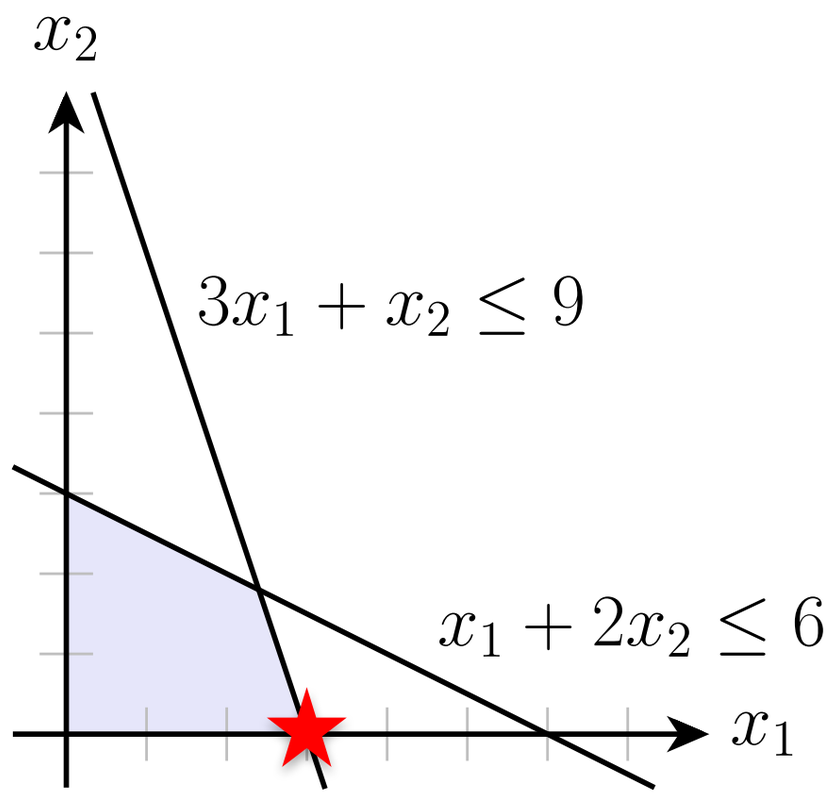
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1,4\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$x = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 3 \end{bmatrix}$$



LINEAR PROGRAMMING

FINDING CORNERS IN POLYTOPE

Polytope for our example:

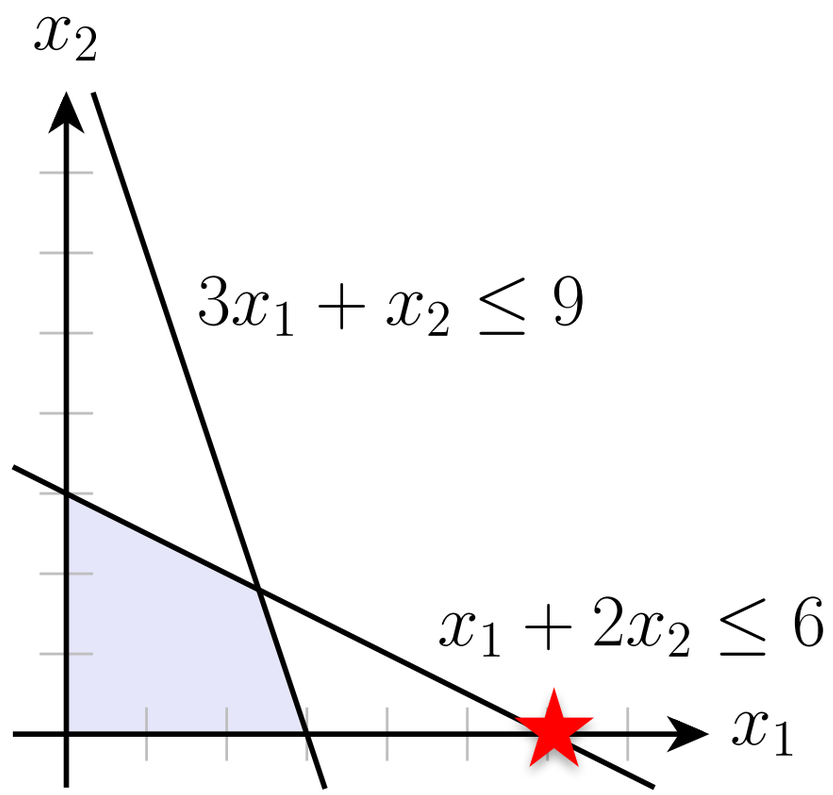
$$A = \begin{bmatrix} 3 & 1 & 1 & 0 \\ 1 & 2 & 0 & 1 \end{bmatrix}, b = \begin{bmatrix} 9 \\ 6 \end{bmatrix}$$

$$\mathcal{J} = \{1,3\}$$

$$x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b$$

$$= \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 9 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ -9 \end{bmatrix}$$

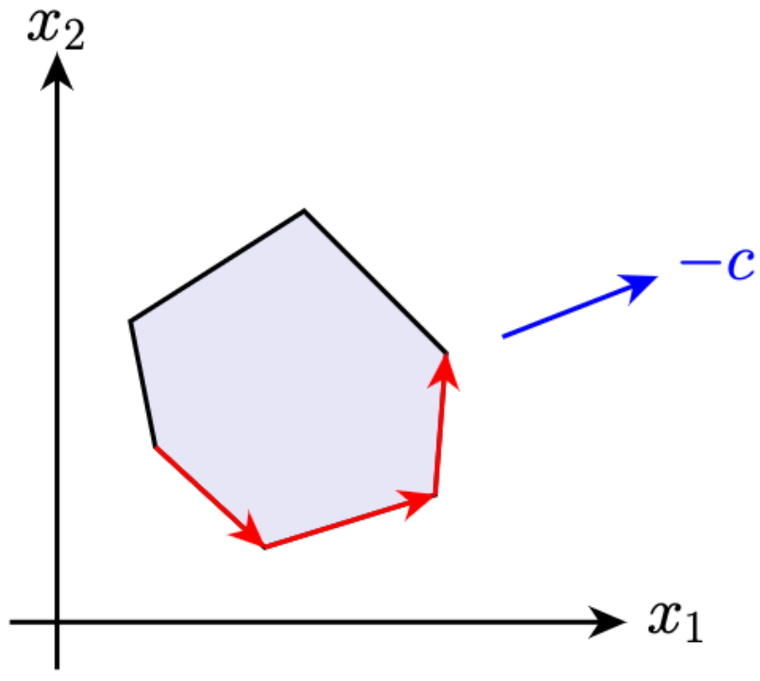
$$x = \begin{bmatrix} 6 \\ 0 \\ -9 \\ 0 \end{bmatrix}$$



LINEAR PROGRAMMING

SIMPLEX ALGORITHM

Basic idea of the simplex algorithm is to move along the edges of the polytope from corner to corner, in directions of decreasing cost.



In worst case, move along an exponentially large number of corners, but typically *much* better in practice (the first "practical" algorithm for linear programming).

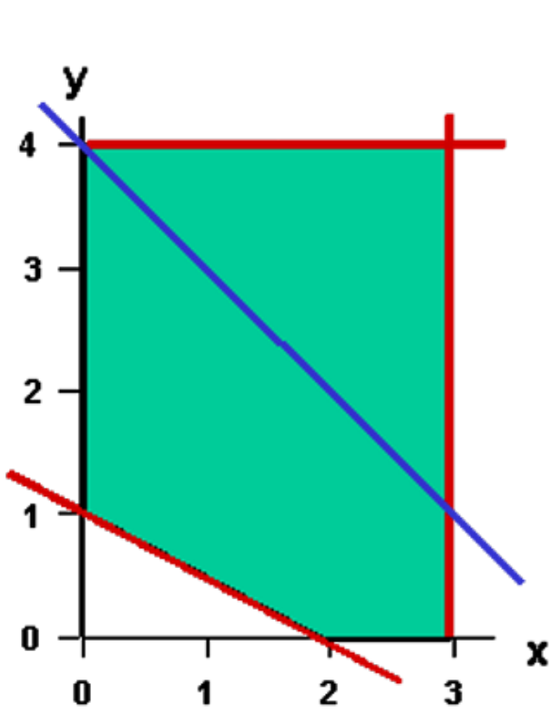
Guaranteed to find a globally optimal solution (ignoring some possible degenerate cases): if simplex returns, then it has found a point that cannot be further improved locally; since **linear programs are convex**, this is a global optimum. Since there are a **finite (but exponential) number of vertices** in the polytope, the algorithm must return after a finite number of steps, improving at each iteration.

LINEAR PROGRAMMING

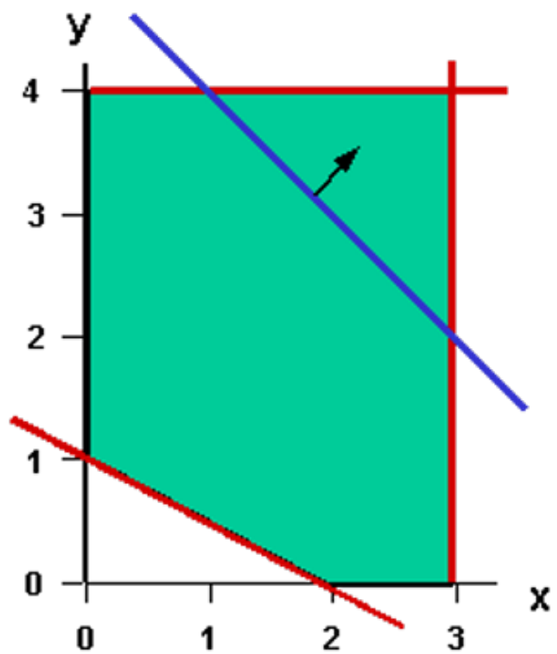
SIMPLEX ALGORITHM

Basic idea of the simplex algorithm is to move along the edges of the polytope from corner to corner, in directions of decreasing cost.

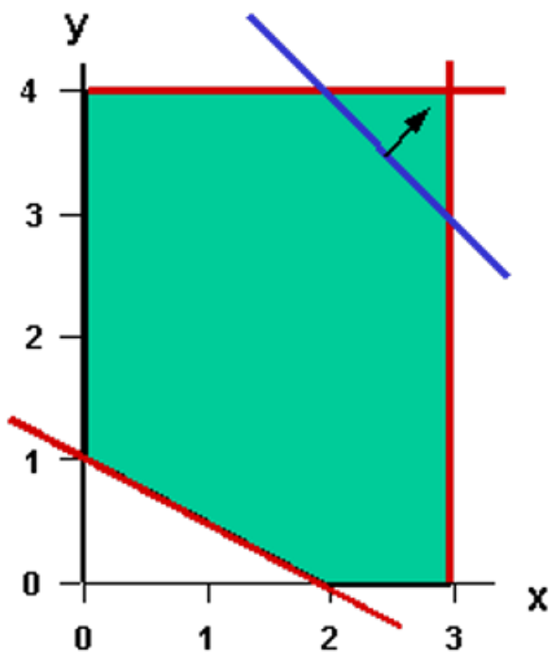
Level sets of the objective f=x+y (i.e., sets where f takes a certain value)



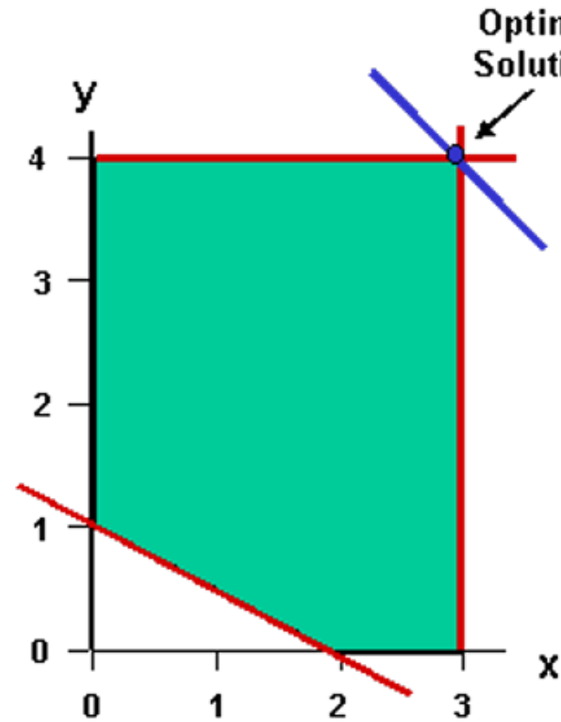
x+y = 4



x+y = 5



x+y = 6

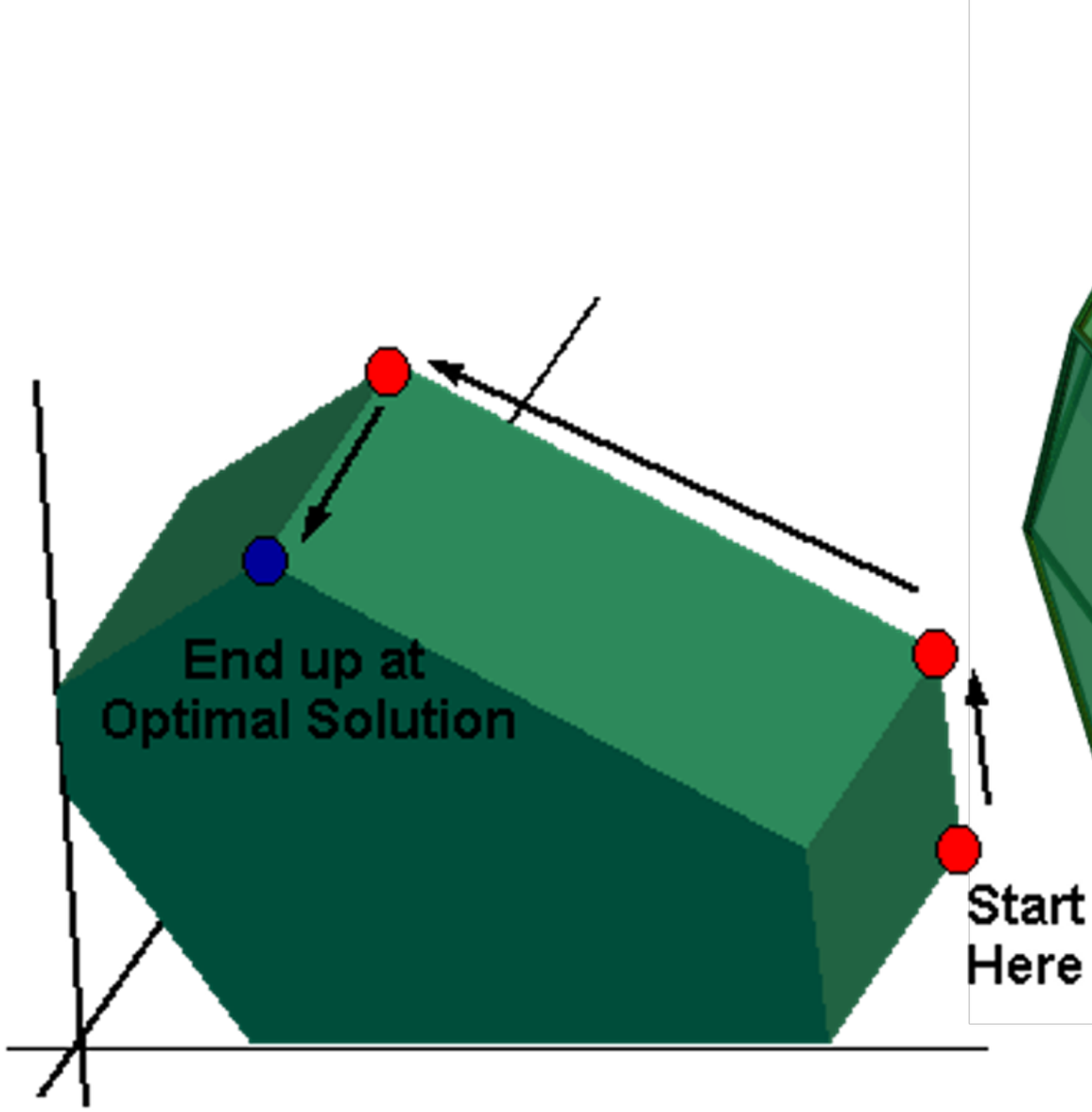


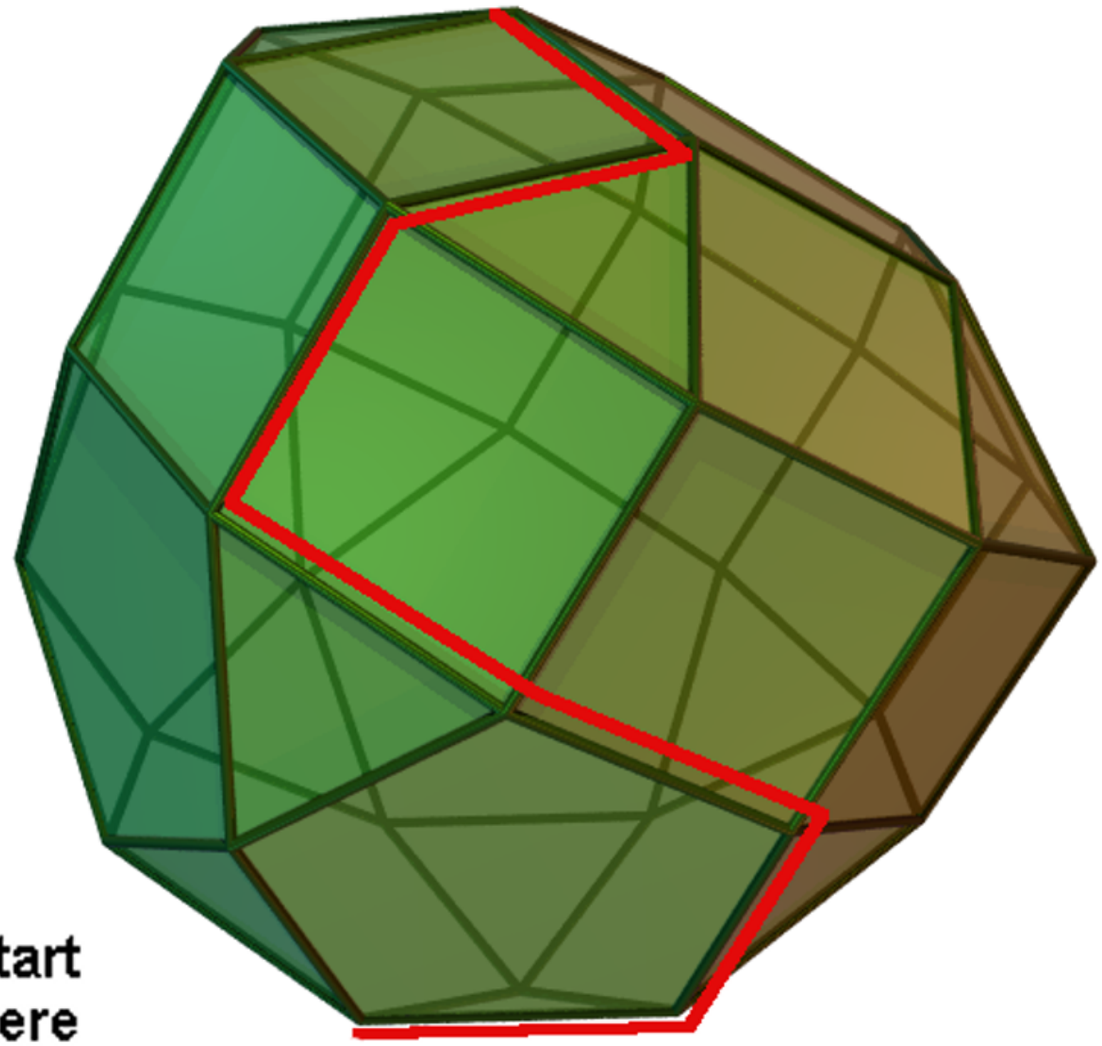
x+y = 7

LINEAR PROGRAMMING

SIMPLEX ALGORITHM

Basic idea of the simplex algorithm is to move along the edges of the polytope from corner to corner, in directions of decreasing cost.

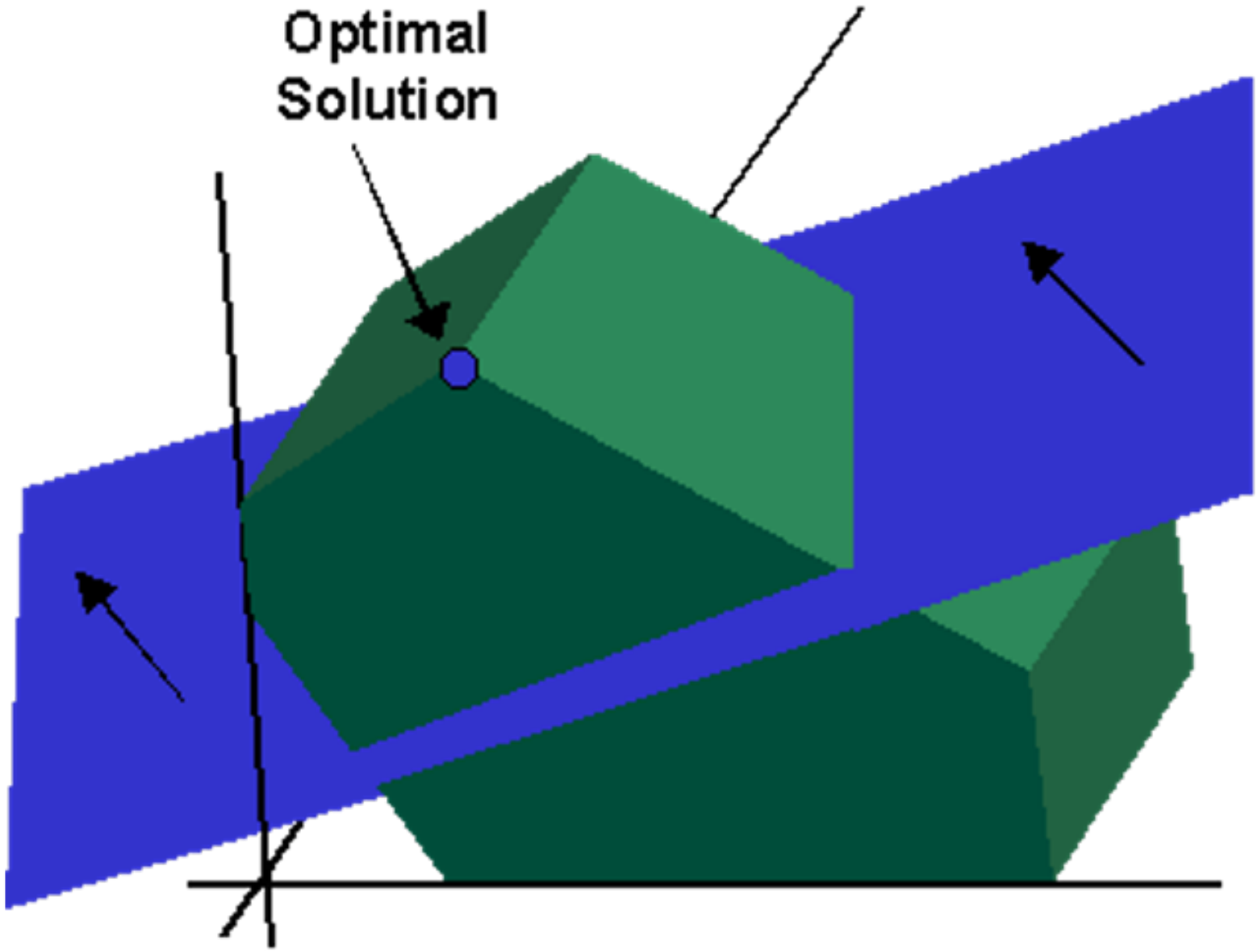




LINEAR PROGRAMMING

SIMPLEX ALGORITHM

Basic idea of the simplex algorithm is to move along the edges of the polytope from corner to corner, in directions of decreasing cost.



LINEAR PROGRAMMING

A SINGLE STEP OF THE SIMPLEX ALGORITHM

Suppose we are optimizing :

$$\begin{aligned} \underset{x}{\text{minimize}} \;\; & c^T x \\ \text{subject to} \;\; & Ax = b \\ & x \geq 0 \end{aligned}$$

and suppose we have some initial feasible corner point x (i.e., we have a basis $\mathcal{J}$ such that $x_{\mathcal{J}} = A_{\mathcal{J}}^{-1} b \geq 0$)

(Seems like a big assumption, but we can relax this)

We would like to *adjust* this point so as to further decrease the cost, but how do we go about adjusting it?

LINEAR PROGRAMMING

A SINGLE STEP OF THE SIMPLEX ALGORITHM (CONT'D)

Suppose we want to adjust x by setting $x_j \leftarrow \alpha$ for some $j \notin \mathcal{I}$ (i.e., so $x_j = 0$ initially), in hopes of decreasing the objective

We cannot *only* change $x_j \leftarrow \alpha$, because this new point would not satisfy the linear equalities

$$A_{\mathcal{I}} x_{\mathcal{I}} = b \implies A_{\mathcal{I}} x_{\mathcal{I}} + \alpha A_j \neq b$$

Instead, we need to adjust $x_{\mathcal{I}}$ by some $\alpha d_{\mathcal{I}}$ to ensure that the equality constraint still holds

$$\begin{aligned}
&A_{\mathcal{I}}(x_{\mathcal{I}} + \alpha d_{\mathcal{I}}) + \alpha A_j = b \\
&\implies \alpha A_{\mathcal{I}} d_{\mathcal{I}} = b - A_{\mathcal{I}} x_{\mathcal{I}} - \alpha A_j \\
&\implies \alpha A_{\mathcal{I}} d_{\mathcal{I}} = -\alpha A_j \qquad (\text{because } A_{\mathcal{I}} x_{\mathcal{I}} = b) \\
&\implies d_{\mathcal{I}} = -A_{\mathcal{I}}^{-1} A_j
\end{aligned}$$

LINEAR PROGRAMMING

A SINGLE STEP OF THE SIMPLEX ALGORITHM (CONT'D)

Now suppose we adjust $x_{\mathcal{J}} \leftarrow x_{\mathcal{J}} + \alpha d_{\mathcal{J}}$ and $x_j \leftarrow \alpha$, how does this change the objective of our optimization problem?

$$c^T x \leftarrow c^T x + \alpha(c_j + c_{\mathcal{J}}^T d_{\mathcal{J}})$$

In other words, setting x_j to be α will increase objective by $\alpha \bar{c}_j$ where

$$\bar{c}_j = c_j - c_{\mathcal{J}}^T A_{\mathcal{J}}^{-1} A_j$$

Thus, as long as $\bar{c}_j$ is negative, it is a "good idea" to adjust x_j in this manner (if more than one $\bar{c}_j$ is negative, we could pick any)

If no $\bar{c}_j < 0$, we have found a solution!

LINEAR PROGRAMMING

A SINGLE STEP OF THE SIMPLEX ALGORITHM (CONT'D)

Final question: how big of a step $x_j \leftarrow \alpha$ should we take?

If all $d_{\mathcal{J}} \geq 0$, we are in "luck", we can decrease the optimization objective arbitrarily without leaving the feasible region (i.e., an unbounded problem)

But if some element $d_i < 0$, for $i \in \mathcal{J}$, we can take at most a step of size:

$$x_i + \alpha d_i = 0 \implies \alpha = -x_i / d_i$$

or we would leave the feasible set

So, take the biggest step we can while keeping x positive, i.e., find:

$$i^\star = \underset{i \in \mathcal{J}: d_i < 0}{\operatorname{argmin}} -x_i / d_i$$

and take step such that $x_{i^\star} = 0$ (at this point, j enters $\mathcal{J}$ and $i^\star$ leaves)

LINEAR PROGRAMMING

SIMPLEX ALGORITHM (COMPLETE)

Repeat:

1. Given index set $\mathcal{I}$ such that $x_{\mathcal{I}} = A_{\mathcal{I}}^{-1} b \geq 0$

2. Find j for which $\bar{c}_j = c_j - c_{\mathcal{I}}^T A_{\mathcal{I}}^{-1} A_j < 0$ (if none exists, return x)

3. Compute step direction $d_{\mathcal{I}} = -A_{\mathcal{I}}^{-1} A_j$ and determine index to remove (or return bounded if $d_{\mathcal{I}} \geq 0$)

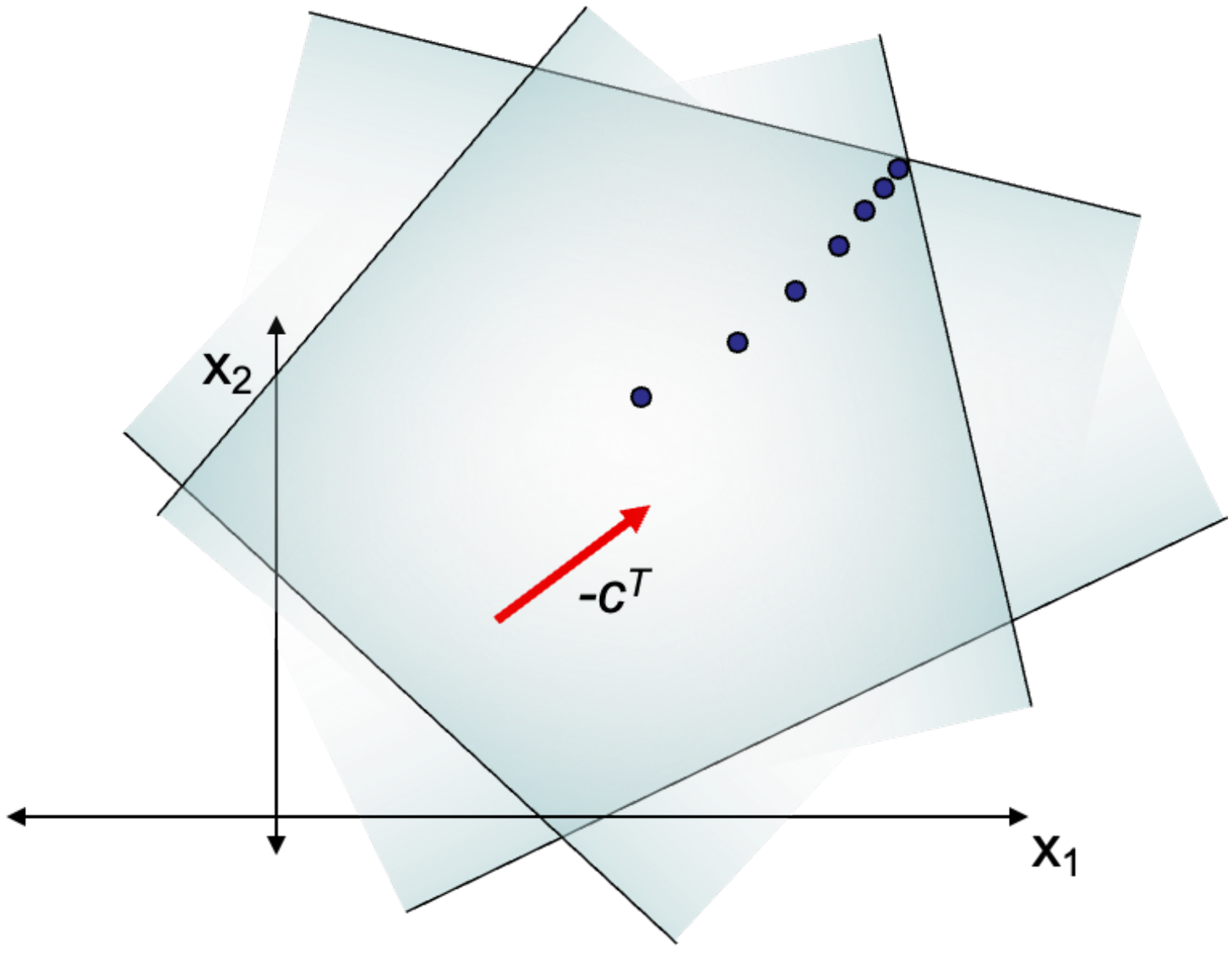
$$i^\star = \underset{i \in \mathcal{I} : d_i < 0}{\operatorname{argmin}} -x_i / d_i$$

4. Update index set: $\mathcal{I} \leftarrow \mathcal{I} - \{i^\star\} \cup \{j\}$

LINEAR PROGRAMMING

ALTERNATIVES TO SIMPLEX

Interior point methods (they scale better than simplex)



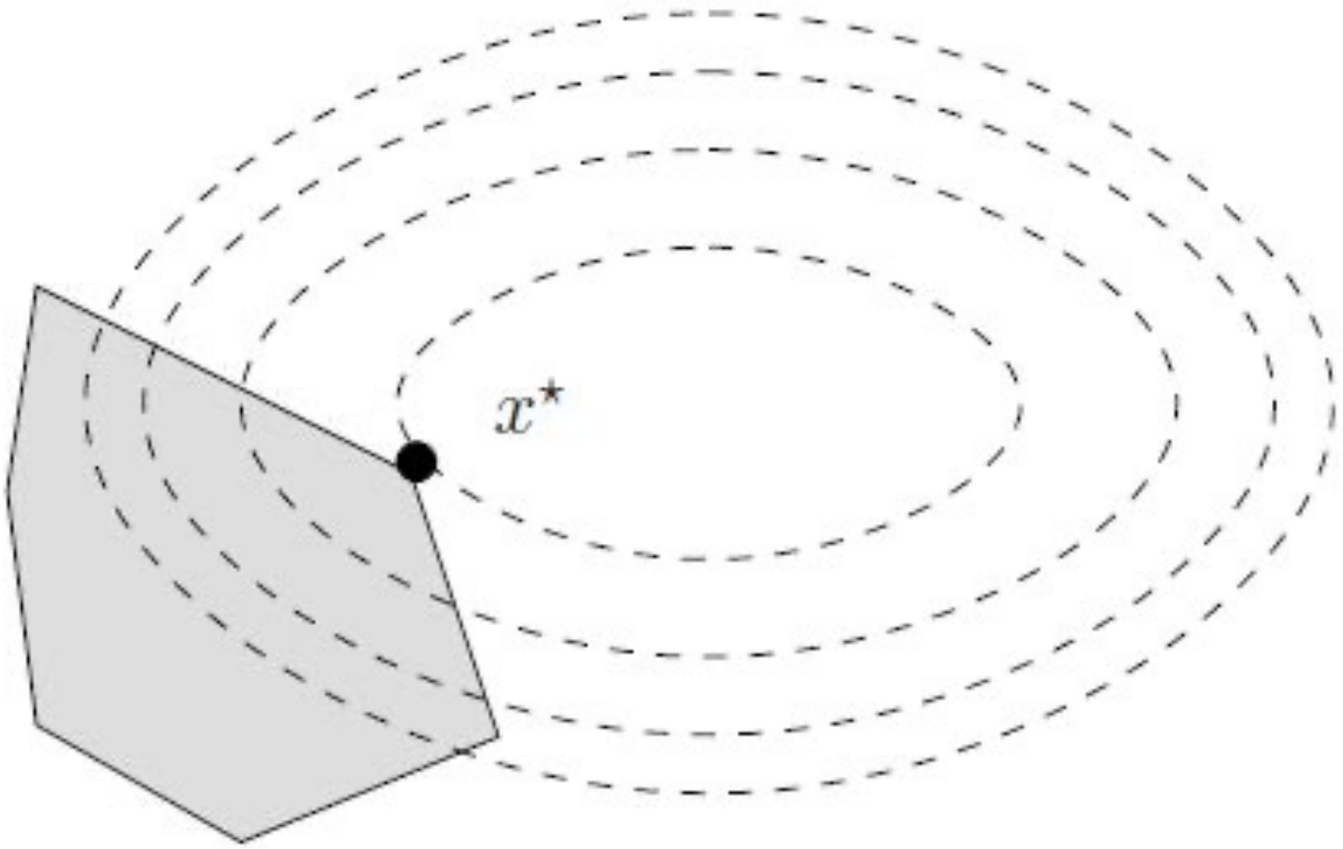
LINEAR PROGRAMMING

MANY PROBLEMS AND APPLICATIONS CAN BE TACKLED BY LP

- Min-cut / max-flow network problems
- Resource allocation problems
- Cost-benefit trade-off problems
- Distribution network problems
- Economic portfolio optimization
- Robotic control
- Scheduling generation
- … and much more!

Many commercial and open-source solvers available, see
https://en.wikipedia.org/wiki/Linear_programming
https://docs.scipy.org/doc/scipy/reference/optimize.html#global-optimization

Quadratic programming



QUADRATIC PROGRAMMING

GENERAL DEFINITION

- A quadratic programming problem minimizes a quadratic function of *n* variables subject to *m* linear inequality or equality constraints. In compact notation, a QP problem is:

$$\textbf{Minimize } f(\boldsymbol{x}) = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{Q}\boldsymbol{x}$$

$$\textbf{Subject to } \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

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

  where c is a vector (n x 1), A is an m x n matrix, and Q is a **symmetric** n x n matrix.

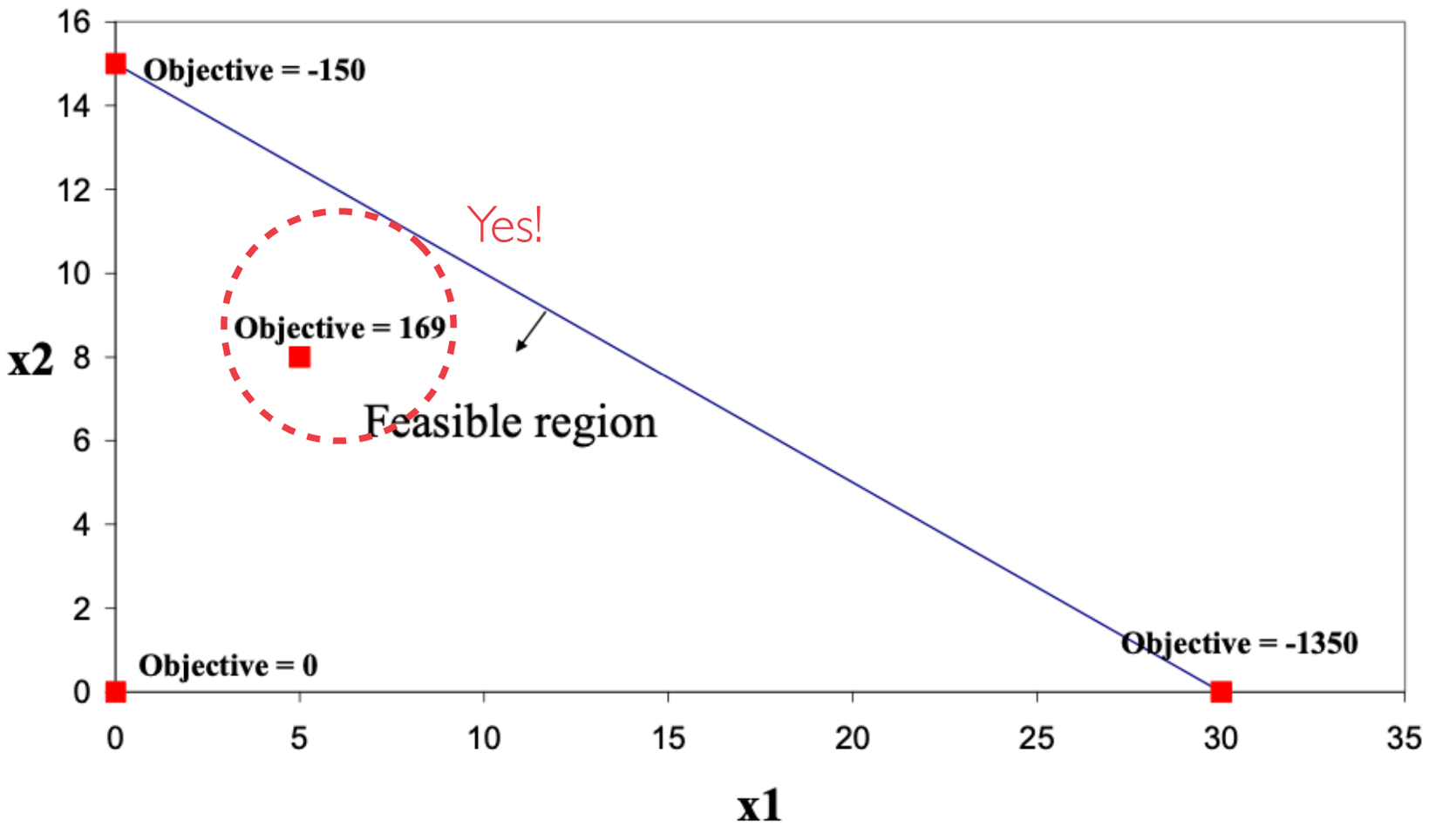
- As for LP, it is also possible to use an inequality form with Ax ≤ b with maximization of f(x).

QUADRATIC PROGRAMMING

EXAMPLE

- Now, if we had a linear objective, the maximum value of the objective would be achieved on the boundary of the feasible region, indeed at a vertex (*as seen earlier in this lecture*).

- But… we actually have a quadratic objective. So does this make a difference?



QUADRATIC PROGRAMMING

EXAMPLE

- Let's suppose we have a QP *maximization* problem formulated as:

maximise

$$15x_1 + 30x_2 + 4x_1x_2 - 2(x_1)^2 - 4(x_2)^2$$

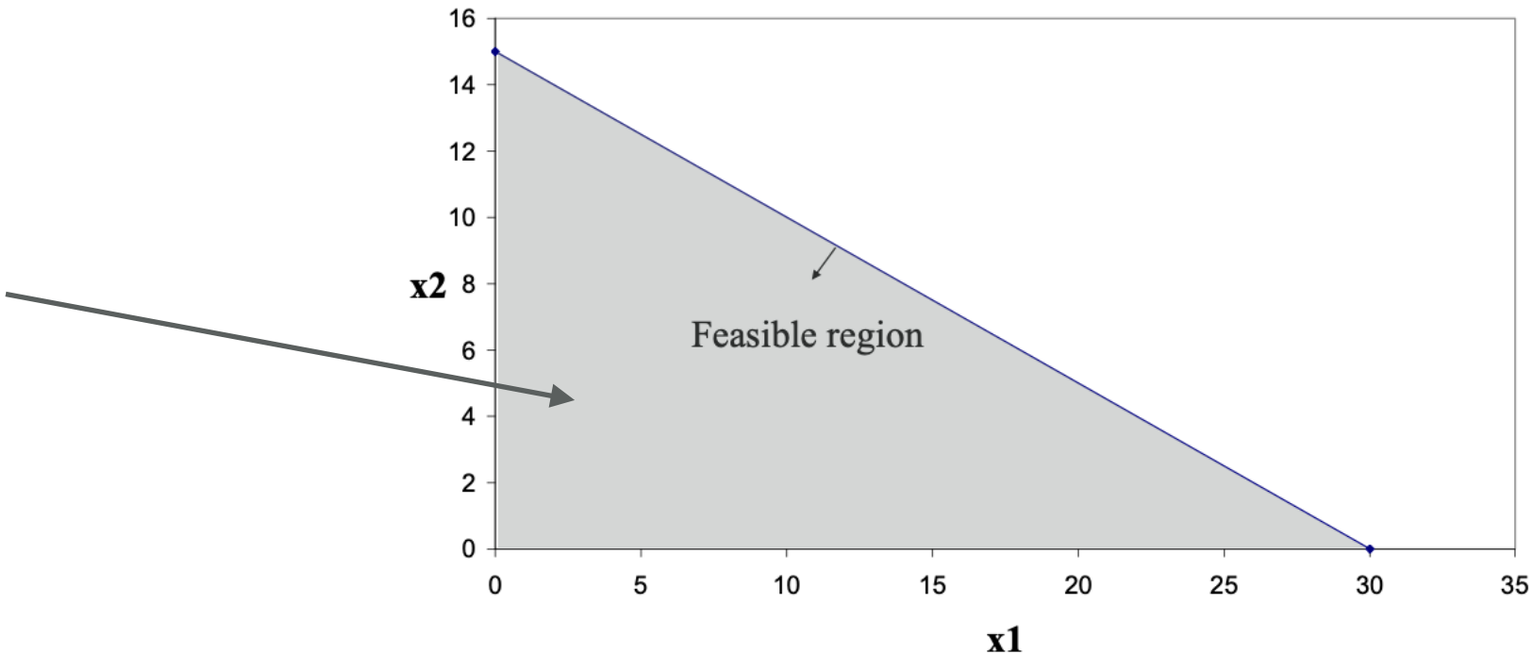
subject to

$$x_1 + 2x_2 \leq 30$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

Note that the feasible region is clearly a convex region.



QUADRATIC PROGRAMMING

EXAMPLE

- To put the constraints in the inequality form $Ax \leq b$ we have:

$$\begin{aligned} \textbf{maximise} \quad & cx - \tfrac{1}{2}x^{\mathrm{T}}Qx \\ \textbf{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

where $c = [15, 30]$, $A = [1,2]$, $b = [30]$, and $Q = [4, -4; -4, 8]$ (note that to get Q we double the coefficients on the nonlinear terms and change the signs, note the -1/2 factor).

- Once the QP problem is in the standard/inequality form, we can solve it in various ways (depending on the properties of Q) - Recall we saw above that the solution is not at a vertex of the feasible region!
  - If Q is a positive semidefinite matrix ($x^{\top}Qx \geq 0 \ \forall x$) → Specific algorithms with theoretical **guarantee** to find the optimal solution.
  - If Q is positive definite ($x^{\top}Qx > 0 \ \forall x$) → Special case of **convex optimization** (no local minima!).
  - General methods for QP: interior point, active set, augmented Lagrangian, conjugate gradient, gradient projection, extensions of the simplex algorithm

QUADRATIC PROGRAMMING

A MORE CONCRETE EXAMPLE - MEAN-VARIANCE PORTFOLIO THEORY

- The archetypical example of QP problems is the portfolio optimization problem, as per the original mean-variance formulation proposed by Markowitz, 1952. The problem is essentially the following: *How can we split our investment between these assets in an appropriate way?*

- To proceed with Markowitz mean-variance portfolio optimization we need some notation, let:
  - N be the number of assets (e.g. stocks) available
  - μ_i be the expected return of asset i
  - ρ_{ij} be the correlation between the returns for assets i and j ($-1 \leq \rho_{ij} \leq +1$)
  - s_i be the standard deviation in return for asset i

- Then the decision variables are:
  w_i the proportion of the total investment associated with (invested in) asset i ($0 \leq w_i \leq 1$)

- Note: this formulation is completely general – provided we have a price history for an "asset" that can be included. So, we could consider making up a portfolio from stocks, commodities (e.g. oil, metals), bonds, etc.

QUADRATIC PROGRAMMING

A MORE CONCRETE EXAMPLE - MEAN-VARIANCE PORTFOLIO THEORY

- Suppose N=2, so we have two assets available in which we can invest. Then the Markowitz approach says that the return we get from investing a proportion w_1 of our wealth in asset 1 and a proportion w_2 of our wealth in asset 2 is:

$$\sum_{i=1}^{N} \mathbf{w_i \mu_i} = \mathbf{w_1 \mu_1 + w_2 \mu_2}$$

- where it must be true that:

$$\sum_{i=1}^{N} \mathbf{w_i} = \mathbf{w_1 + w_2} = 1$$

which states that we invest all of the money we have available.

- The risk (variance) associated with this investment is given by (note that all the terms are so here all the terms are **quadratic** in the decision variables w_i):

$$\begin{aligned}
& \sum_{i=1}^{N} \sum_{j=1}^{N} \mathbf{w_i w_j \rho_{ij} s_i s_j} \\
= \quad & \mathbf{w_1 w_1 \rho_{11} s_1 s_1 + w_1 w_2 \rho_{12} s_1 s_2 +} \\
& \mathbf{w_2 w_1 \rho_{21} s_2 s_1 + w_2 w_2 \rho_{22} s_2 s_2} \\
= \quad & \mathbf{w_1 w_1 s_1 s_1 + 2 w_1 w_2 \rho_{12} s_1 s_2 + w_2 w_2 s_2 s_2} \\
= \quad & \mathbf{(w_1)^2 (s_1)^2 + 2 w_1 w_2 \rho_{12} s_1 s_2 + (w_2)^2 (s_2)^2}
\end{aligned}$$

QUADRATIC PROGRAMMING

A MORE CONCRETE EXAMPLE - MEAN-VARIANCE PORTFOLIO THEORY

- If we had just N=2 assets as above, then it is just a matter of considering all possible investment portfolios by enumerating all possible choices for w_1 and w_2 (where $w_2 = w_1 - 1$ in the 2-asset case).

- Of course we almost always have *many more than two assets* in which we could invest, and so the approach considered above becomes infeasible. Obviously, we need to move from enumerating choices to making a choice via numerical optimization.

- Let R be the desired expected return from the portfolio chosen. Then, using the standard Markowitz mean-variance approach we can define the *portfolio optimization problem* as follows:

$$\text{minimise} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \rho_{ij} s_i s_j$$

$$\text{subject to}$$

$$\sum_{i=1}^{N} w_i \mu_i = R$$

$$\sum_{i=1}^{N} w_i = 1$$

$$w_i \geq 0 \qquad i=1,\ldots,N$$

QUADRATIC PROGRAMMING

A MORE CONCRETE EXAMPLE - MEAN-VARIANCE PORTFOLIO THEORY

$$\text{minimise} \quad \sum_{i=1}^{N} \sum_{j=1}^{N} w_i w_j \rho_{ij} s_i s_j$$

$$\text{subject to}$$

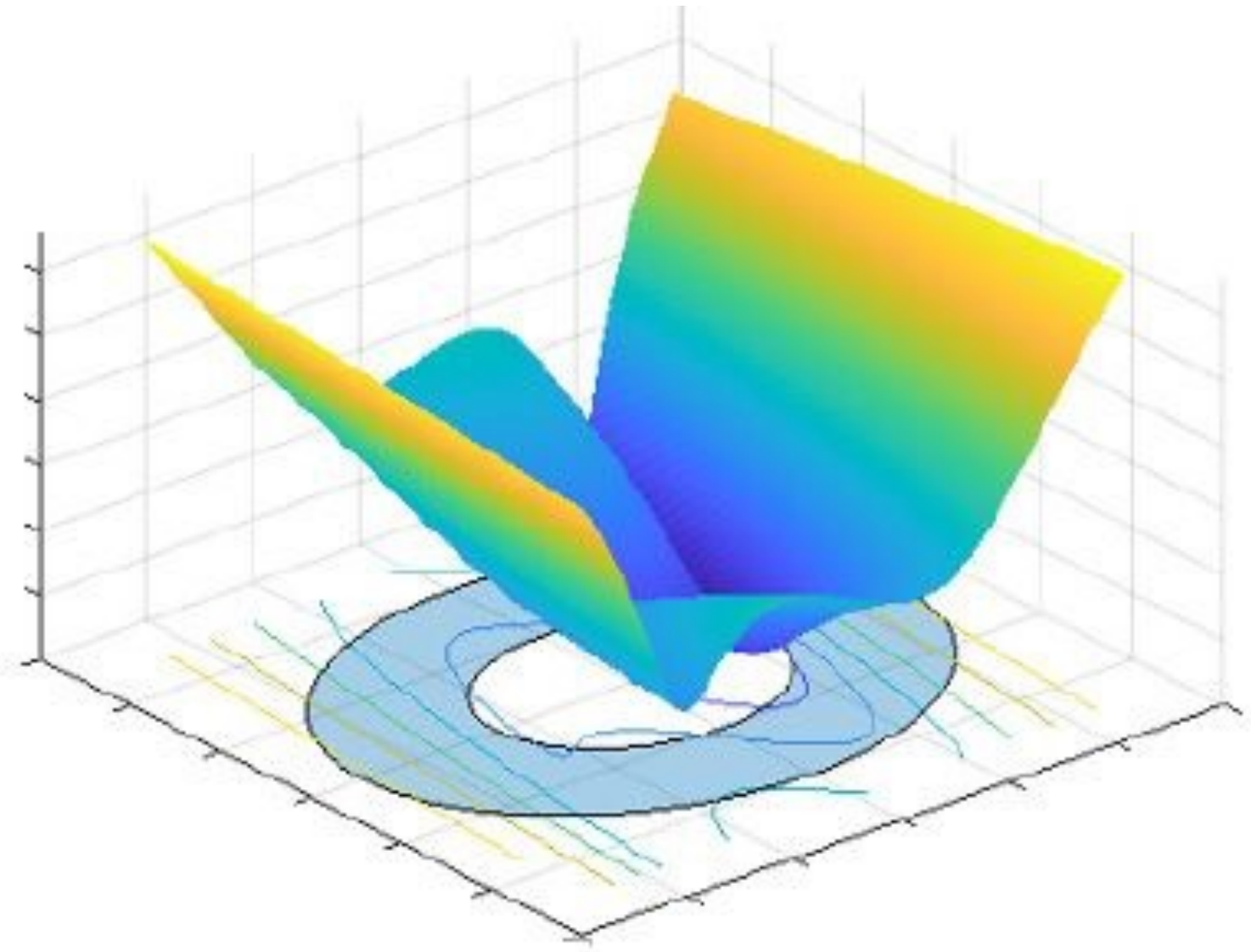
$$\sum_{i=1}^{N} w_i \mu_i = R$$

$$\sum_{i=1}^{N} w_i = 1$$

$$w_i \geq 0 \qquad i=1,\ldots,N$$

- The objective function minimizes the total variance (risk) associated with the portfolio.
- The first constraint ensures that the portfolio has an expected return of *R* (note that some formulation replace equality with inequality ≥ R).
- The second constraint ensures that the proportions add to one.
- The third constraint is simply the non-negativity constraint.

Other cases



OTHER CASES

QUADRATIC CONSTRAINED QUADRATIC PROGRAMMING (QCQP)

- In this case, quadratic objective function and quadratic constraints:

$$\begin{aligned} \textbf{minimize} \quad & \tfrac{1}{2}x^{\mathrm{T}}P_0x + q_0^{\mathrm{T}}x \\ \textbf{subject to} \quad & \tfrac{1}{2}x^{\mathrm{T}}P_ix + q_i^{\mathrm{T}}x + r_i \leq 0 \quad \text{for } i = 1, \ldots, m, \\ & Ax = b, \end{aligned}$$

- Solving the general case is an NP-hard problem. To see this, note that the two constraints $x1(x1 - 1) \leq 0$ and $x1(x1 - 1) \geq 0$ are equivalent to the constraint $x1(x1 - 1) = 0$, which is in turn equivalent to the constraint $x1 \in \{0, 1\}$. Hence, any 0–1 integer program (in which all variables have to be either 0 or 1) can be formulated as a quadratically constrained quadratic program. Since 0–1 integer programming is NP-hard in general, QCQP is also NP-hard.

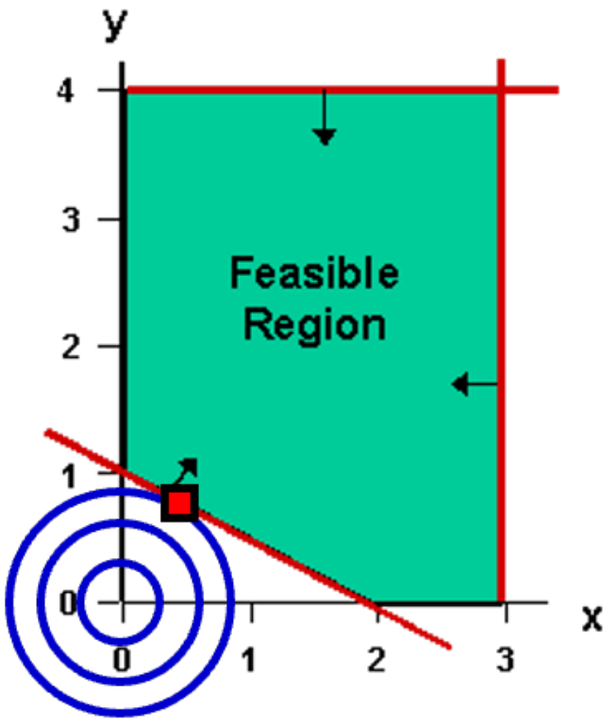
- Example: Max Cut problem. Given a graph, the problem is to divide the vertices in two sets, so that as many edges as possible go from one set to the other. Max Cut can be formulated as a QCQP.

- This kind of problems can be solved e.g. by means of SDP relaxations.

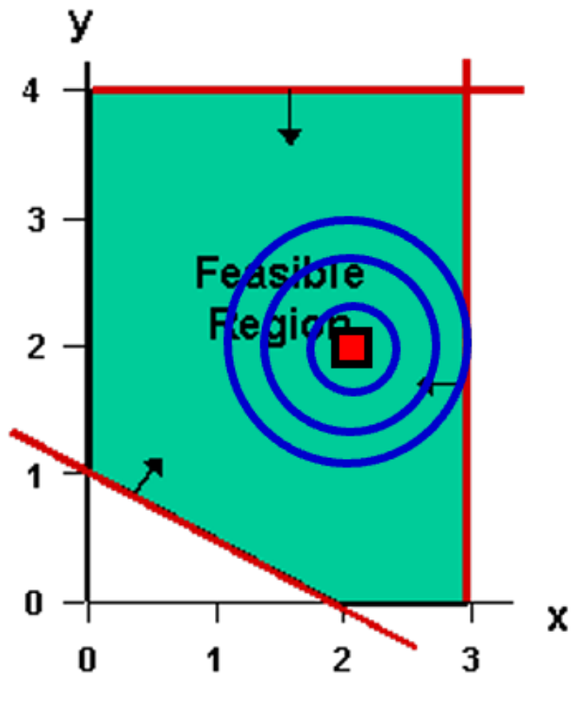
OTHER CASES

QUADRATIC CONSTRAINED QUADRATIC PROGRAMMING (QCQP)

Solution no longer at a vertex!

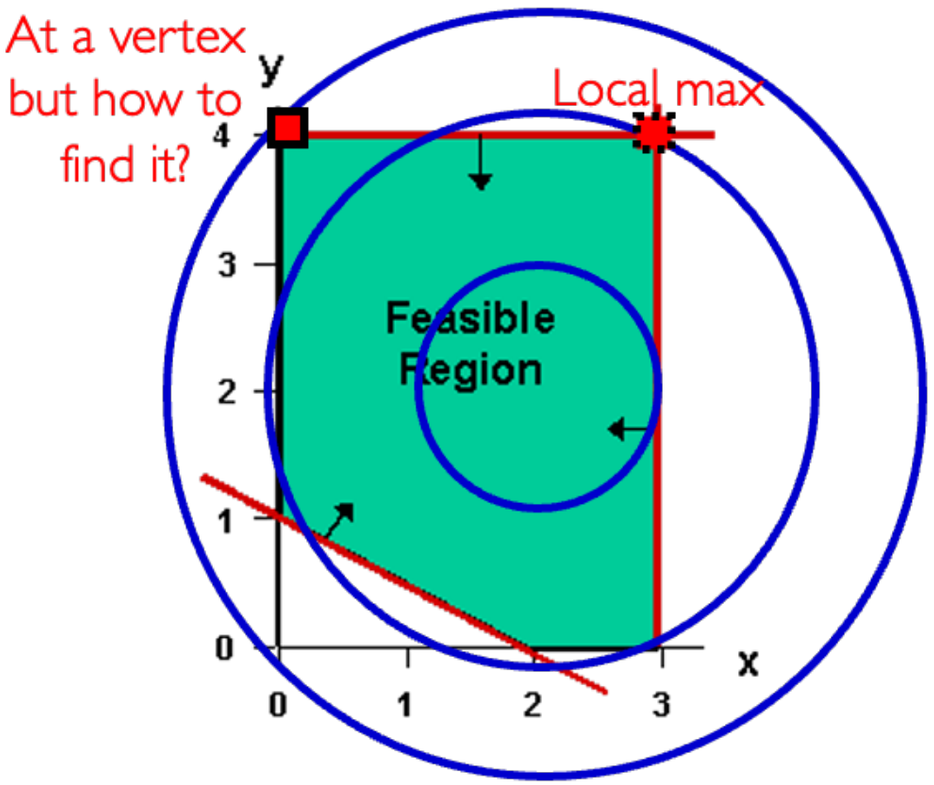


Level sets of x^2+y^2
(try to minimize)



Level sets of $(x-2)^2+(y-2)^2$
(try to minimize)

At a vertex but how to find it?

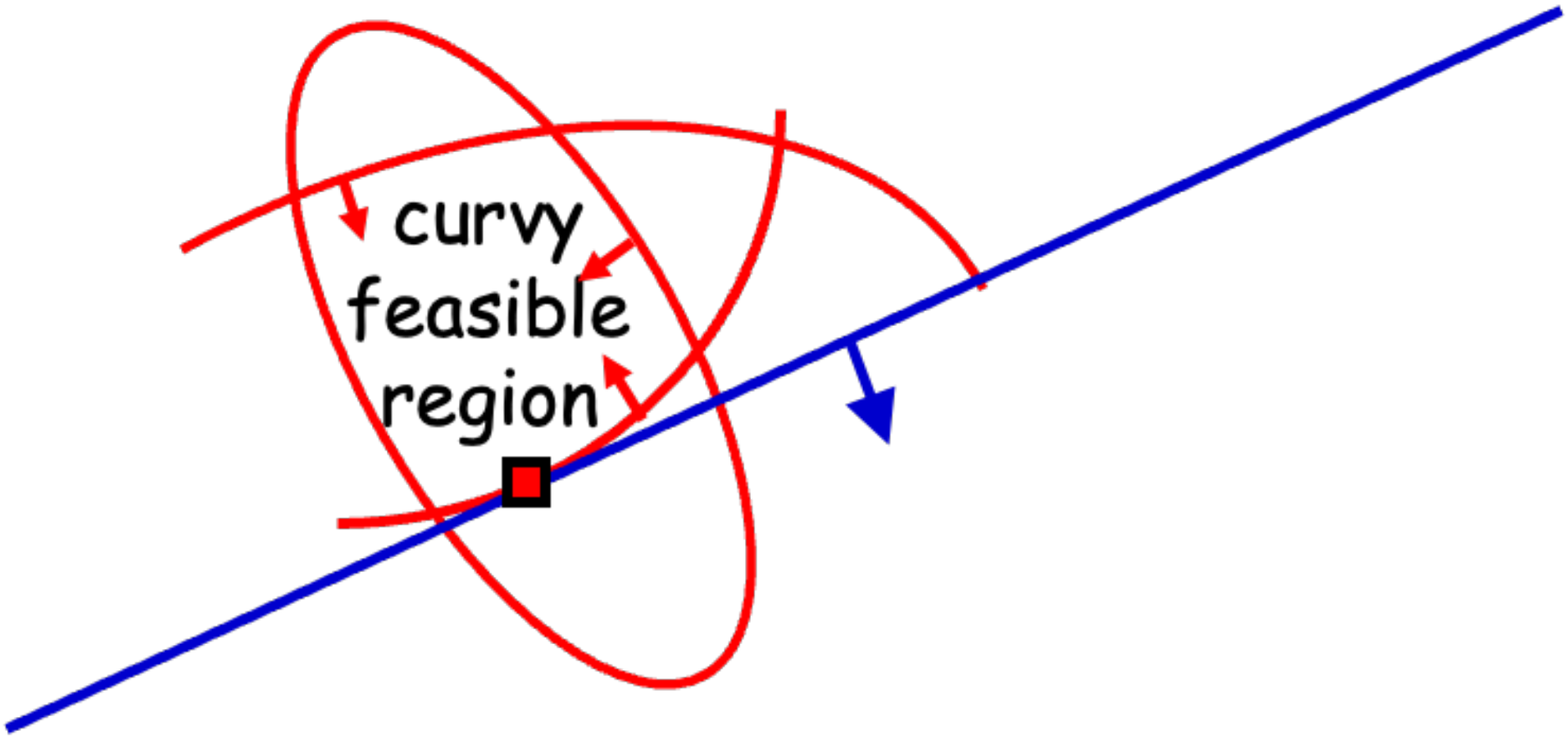


Same, but maximize
(no longer convex)

OTHER CASES

QUADRATIC CONSTRAINED QUADRATIC PROGRAMMING (QCQP)

NOTE: A special case of QCQP problems occurs where the objective function is linear (P_0=0). The level sets of f(x) are hyperplanes (as in LP), but in this case the optimum is not at a vertex!



OTHER CASES

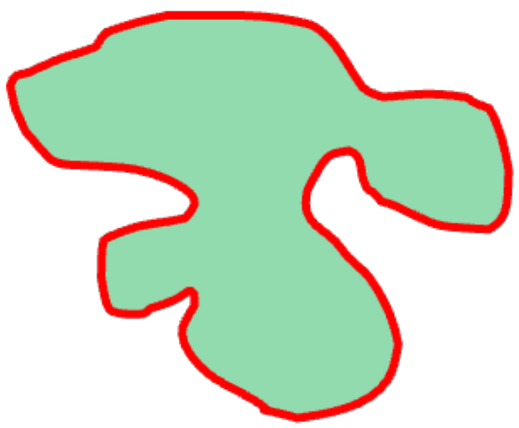
NONLINEAR PROGRAMMING

- The most general optimization problem occurs when both the objective function and constraints are nonlinear/non-convex, a case referred to as nonlinear programming (NLP).

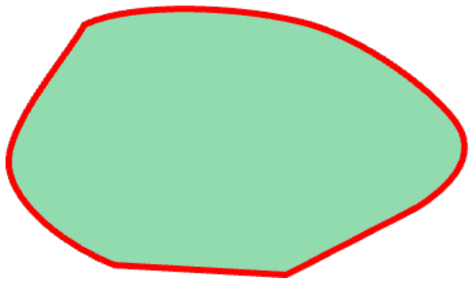
- Remember: what makes problem really hard is non-convexity (rather than nonlinearity)!

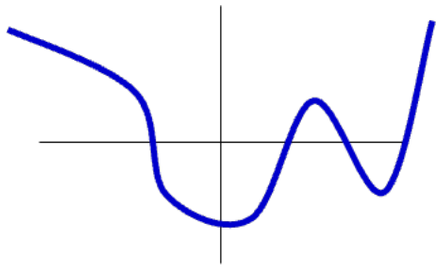
- No specific methods for this class of problems: in practice, one often falls back on metaheuristics (e.g., Simulated Annealing, evolutionary algorithms, etc.), Bayesian optimization, etc.

- To get an exact solution, "backtracking" search methods can be used, that recursively divide up the space into regions can be used (similar to DIRECT for continuous optimization, or branch and bound for combinatorial problems). These are applicable if it is possible to compute informative "optimistic" bounds on the best solution within a region, e.g., by linear approximations.

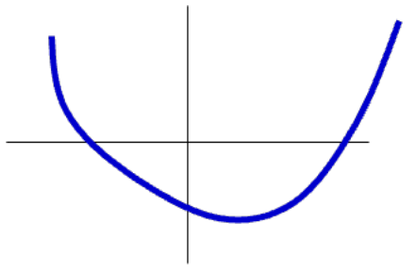


but not





but not



Questions?