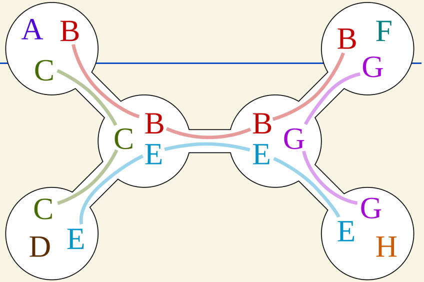
Preprocessing and Kernelization II



DM898: Parameterized Algorithms
Lars Rohwedder

Today's lecture

- Sunflower lemma
- Preprocessing in practice (for linear programming)

Sunflower Lemma

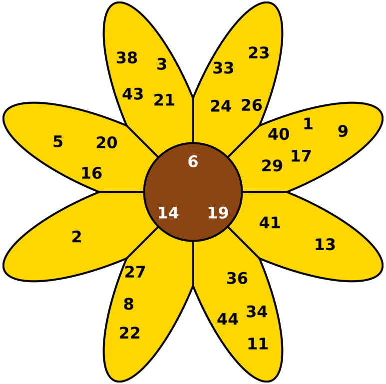
Context

- we have seen some problem-specific kernels (for Vertex Cover and Edge Clique Cover)
- the textbook also contains several examples of **general kernelization techniques** that can be applied to many problems (usually requiring some additional ideas). These are based on finding and exploiting specific structures occuring in the input (often graphs or set systems)
- as a clean example, we look at the **Sunflower Lemma**. Other examples in the textbook are **Crown Decomposition** (Chapter 2.3), **Expansion Lemma** (Chapter 2.4), **Representative Sets** (Chapter 12.3),...

The Sunflower Lemma is relevant to problem that involve set systems, i.e., a family $\mathcal{A}$ of sets over a universe U.

Definition (Sunflower)

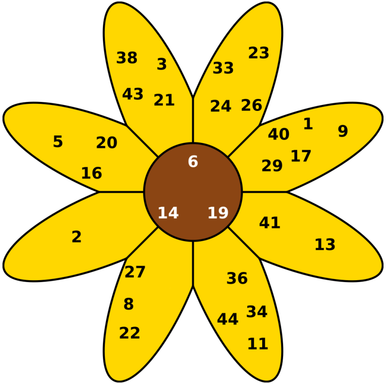
For a **core** set Y, we say that sets $S_1, \dots, S_k \supsetneq Y$ are a **sunflower** if $S_i \cap S_j = Y$ for all $i \neq j$. We call $S_1 \setminus Y, \dots, S_k \setminus Y$ the **petals** of the sunflower.



The Sunflower Lemma is relevant to problem that involve set systems, i.e., a family $\mathcal{A}$ of sets over a universe U.

Definition (Sunflower)

For a **core** set Y, we say that sets $S_1, \dots, S_k \supsetneq Y$ are a **sunflower** if $S_i \cap S_j = Y$ for all $i \neq j$. We call $S_1 \setminus Y, \dots, S_k \setminus Y$ the **petals** of the sunflower.



Sunflower Lemma

Let $\mathcal{A}$ be a family of sets (without duplicates) over a universe U, such that each set in $\mathcal{A}$ has cardinality exactly d. If $|\mathcal{A}| > d!(k-1)^d$ then $\mathcal{A}$ contains a sunflower with k petals and such a sunflower can be computed in time polynomial in $|\mathcal{A}|$, $|U|$, and k.

Proof on blackboard.

Kernel for d-Hitting Set

d-Hitting Set problem

Let $\mathcal{A}$ be a family of sets over a universe U, where each set has cardinality **at most** d and let $k \in \mathbb{N}$. The goal of the d-Hitting Set problem is to find $H \subseteq U$ with $|H| \leq k$ such that $H \cap A \neq \emptyset$ for all $A \in \mathcal{A}$.

Kernel for d-Hitting Set

d-Hitting Set problem

Let $\mathcal{A}$ be a family of sets over a universe U, where each set has cardinality **at most** d and let $k \in \mathbb{N}$. The goal of the d-Hitting Set problem is to find $H \subseteq U$ with $|H| \leq k$ such that $H \cap A \neq \emptyset$ for all $A \in \mathcal{A}$.

If $\mathcal{A}$ is sufficiently large, then it must contain a large sunflower Y, $S_1, \ldots, S_{k+1}$. A hitting set of cardinality k must contain a vertex in Y.

Reduction rule

For each $d' \leq d$ try to apply Sunflower Lemma on sets in $\mathcal{A}$ with cardinality exactly d'. If we find sunflower with $k+1$ petals $Y, S_1, \ldots, S_{k+1}$, then set $\mathcal{A}' = \mathcal{A} \setminus \{S_1, \ldots, S_{k+1}\} \cup Y$ and $U' = \bigcup_{A \in \mathcal{A}'} A$. Return instance $(U', \mathcal{A}')$.

Kernel for d-Hitting Set

d-Hitting Set problem

Let $\mathcal{A}$ be a family of sets over a universe U, where each set has cardinality **at most** d and let $k \in \mathbb{N}$. The goal of the d-Hitting Set problem is to find $H \subseteq U$ with $|H| \leq k$ such that $H \cap A \neq \emptyset$ for all $A \in \mathcal{A}$.

If $\mathcal{A}$ is sufficiently large, then it must contain a large sunflower $Y, S_1, \ldots, S_{k+1}$. A hitting set of cardinality k must contain a vertex in Y.

Reduction rule

For each $d' \leq d$ try to apply Sunflower Lemma on sets in $\mathcal{A}$ with cardinality exactly d'. If we find sunflower with $k+1$ petals $Y, S_1, \ldots, S_{k+1}$, then set $\mathcal{A}' = \mathcal{A} \setminus \{S_1, \ldots, S_{k+1}\} \cup Y$ and $U' = \bigcup_{A \in \mathcal{A}'} A$. Return instance $(U', \mathcal{A}')$.

Kernel with $\leq d!k^d \cdot d$ sets: apply reduction rule until exhaustion. When no more sunflowers with $k+1$ petals are found, then for all $d' \leq d$ the number of sets of cardinality d' in $\mathcal{A}$ must be at most

$$d'!k^{d'} \leq d!k^d.$$

Thus,

$$|\mathcal{A}| \leq d!k^d \cdot d.$$

Preprocessing in practice

Using data reduction routines before applying main algorithm is a promising approach for most problems.

For practical examples, we focus on **integer linear programming**, due to its expressive power, probably the most important problem in Operations Research. Preprocessing is a crucial element in the state-of-the-art:

Progress on ILP solvers during 2000-2020

> *[...] branch-and-cut has replaced branch-and-bound as the basic computational tool [...] The other most significant developments in the solvers are much improved* **preprocessing/presolving** *and many new ideas for primal heuristics. The result has been a speed-up of several orders of magnitude in the codes. The other major change has been the widespread use of decomposition algorithms, in particular column generation (branch-(cut)-and-price) and Benders' decomposition.*
>
> Preface of textbook "Integer Programming" by Wolsey

(Integer) Linear programming

A linear program consists of a set of variables $x_1, \ldots, x_n$ and a mathematical system of the following form:

- Each variable has a **domain** that describes which values are allowed. Allowed domains are upper and/or lower bounds, e.g. $-1 \leq x_1 \leq 2$ and optional integrality requirement. Examples: $x_1 \in \mathbb{R}_{\geq 0}$, $x_2 \in \mathbb{Z}, x_3 \in \{0, 1\}$
- An optional **objective** that describes a linear function in the variables to be optimized and an optimization direction ($\max$ or $\min$). Example: $\max x_1 + 2x_2$
- There are one or more **constraints**. These enforce a relationship ($\leq$, $=$, or $\geq$) between two affine linear functions over the variables. Examples: $2x_1 + 5 \geq 3 - x_2$

If all variables have integer domain, we call the system an **integer linear program (ILP)**. If none of the variables have integer domain, we call the system a **continuous linear program** or just **linear program (LP)**. If both integer and continuous variables appear, we call it a **mixed-integer linear program**.

(Integer) Linear programming

A linear program consists of a set of variables $x_1, \ldots, x_n$ and a mathematical system of the following form:

- Each variable has a **domain** that describes which values are allowed. Allowed domains are upper and/or lower bounds, e.g. $-1 \leq x_1 \leq 2$ and optional integrality requirement. Examples: $x_1 \in \mathbb{R}_{\geq 0}$, $x_2 \in \mathbb{Z}, x_3 \in \{0, 1\}$
- An optional **objective** that describes a linear function in the variables to be optimized and an optimization direction ($\max$ or $\min$). Example: $\max x_1 + 2x_2$
- There are one or more **constraints**. These enforce a relationship ($\leq$, $=$, or $\geq$) between two affine linear functions over the variables. Examples: $2x_1 + 5 \geq 3 - x_2$

If all variables have integer domain, we call the system an **integer linear program (ILP)**. If none of the variables have integer domain, we call the system a **continuous linear program** or just **linear program (LP)**. If both integer and continuous variables appear, we call it a **mixed-integer linear program**.

Complexity of linear programming

- ILPs can easily model NP-hard problems $\Rightarrow$ ILP is NP-hard
- Continuous LPs can be solved very efficiently both in practice and theory

(Integer) Linear programming

A linear program consists of a set of variables $x_1, \ldots, x_n$ and a mathematical system of the following form:

- Each variable has a **domain** that describes which values are allowed. Allowed domains are upper and/or lower bounds, e.g. $-1 \leq x_1 \leq 2$ and optional integrality requirement. Examples: $x_1 \in \mathbb{R}_{\geq 0}$, $x_2 \in \mathbb{Z}, x_3 \in \{0, 1\}$
- An optional **objective** that describes a linear function in the variables to be optimized and an optimization direction ($\max$ or $\min$). Example: $\max x_1 + 2x_2$
- There are one or more **constraints**. These enforce a relationship ($\leq$, $=$, or $\geq$) between two affine linear functions over the variables. Examples: $2x_1 + 5 \geq 3 - x_2$

If all variables have integer domain, we call the system an **integer linear program (ILP)**. If none of the variables have integer domain, we call the system a **continuous linear program** or just **linear program (LP)**. If both integer and continuous variables appear, we call it a **mixed-integer linear program**.

Complexity of linear programming

- ILPs can easily model NP-hard problems $\Rightarrow$ ILP is NP-hard
- Continuous LPs can be solved very efficiently both in practice and theory

Examples on blackboard.

Preprocessing for linear programming

Goals:

- Reduce number of variables (columns)
- Reduce number of constraints (rows)
- Tightening variable bounds and constraints

```
Variable types: 4800 continuous, 38160 integer (32880 binary)
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  QMatrix range    [3e-01, 4e+01]
  QLMatrix range   [1e+00, 4e+01]
  Objective range  [1e+00, 2e+03]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 2e+01]
  QRHS range       [4e+01, 8e+03]
Presolve removed 3118 rows and 34593 columns
Presolve time: 0.80s
Presolved: 22954 rows, 14206 columns, 74771 nonzeros
Variable types: 3790 continuous, 10416 integer (10034 binary)

Root relaxation: objective 3.999561e+06, 1970 iterations, 0.12 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 3999560.63    0  269          - 3999560.63      -     -    2s
     0     0 4003616.91    0  413          - 4003616.91      -     -    2s
```

Example output of Gurobi solver including statistics about preprocessing

Example: merging parallel variables

Consider the following linear program

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 - x_4 \\
5x_1 - 2x_2 + 8x_3 + 8x_4 \leq 15 \\
8x_1 + 3x_2 - x_3 - x_4 \geq 9 \\
x_1 + x_2 + x_3 + x_4 \leq 6 \\
0 \leq x_1 \leq 3 \\
0 \leq x_2 \leq 1 \\
1 \leq x_3 \leq 10 \\
0 \leq x_4 \leq 2
\end{aligned}
$$

Variables x_3, x_4 are **parallel** (same coefficients in objective and all constraints)

Example: merging parallel variables

Consider the following linear program

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 - x_4& \\
5x_1 - 2x_2 + 8x_3 + 8x_4 &\leq 15 \\
8x_1 + 3x_2 - x_3 - x_4 &\geq 9 \\
x_1 + x_2 + x_3 + x_4 &\leq 6 \\
0 \leq x_1 &\leq 3 \\
0 \leq x_2 &\leq 1 \\
1 \leq x_3 &\leq 10 \\
0 \leq x_4 &\leq 2
\end{aligned}
$$

Variables x_3, x_4 are **parallel** (same coefficients in objective and all constraints)

$\rightsquigarrow$

Merging x_3, x_4 into single variable x_3:

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3& \\
5x_1 - 2x_2 + 8x_3 &\leq 15 \\
8x_1 + 3x_2 - x_3 &\geq 9 \\
x_1 + x_2 + x_3 &\leq 6 \\
0 \leq x_1 &\leq 3 \\
0 \leq x_2 &\leq 1 \\
1 \leq x_3 &\leq 12
\end{aligned}
$$

Example: tightening bounds

Continuing with the same linear program

$$\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 \leq 15 \\
8x_1 + 3x_2 - x_3 \geq 9 \\
x_1 + x_2 + x_3 \leq 6 \\
0 \leq x_1 \leq 3 \\
0 \leq x_2 \leq 1 \\
1 \leq x_3
\end{aligned}$$

Constraint 1 implies

$$5x_1 \leq 15 + 2x_2 - 8x_3 \leq 9$$

Similarly,

$$8x_3 \leq 15 + 2x_2 - 5x_1 \leq 17.$$

Example: tightening bounds

Continuing with the same linear program

$$\max 2x_1 + x_2 - x_3$$
$$5x_1 - 2x_2 + 8x_3 \leq 15$$
$$8x_1 + 3x_2 - x_3 \geq 9$$
$$x_1 + x_2 + x_3 \leq 6$$
$$0 \leq x_1 \leq 3$$
$$0 \leq x_2 \leq 1$$
$$1 \leq x_3$$

Constraint 1 implies

$$5x_1 \leq 15 + 2x_2 - 8x_3 \leq 9$$

Similarly,

$$8x_3 \leq 15 + 2x_2 - 5x_1 \leq 17.$$

$$\rightsquigarrow$$

Tightening the bounds:

$$\max 2x_1 + x_2 - x_3$$
$$5x_1 - 2x_2 + 8x_3 \leq 15$$
$$8x_1 + 3x_2 - x_3 \geq 9$$
$$x_1 + x_2 + x_3 \leq 6$$
$$0 \leq x_1 \leq \frac{9}{5}$$
$$0 \leq x_2 \leq 1$$
$$1 \leq x_3 \leq \frac{17}{8}$$

Example: removing redundant constraints

Continuing with the same linear program

$$\max 2x_1 + x_2 - x_3$$

$$5x_1 - 2x_2 + 8x_3 \leq 15$$

$$8x_1 + 3x_2 - x_3 \geq 9$$

$$x_1 + x_2 + x_3 \leq 6$$

$$0 \leq x_1 \leq \frac{9}{5}$$

$$0 \leq x_2 \leq 1$$

$$1 \leq x_3 \leq \frac{17}{8}$$

3rd constraint is **redundant** due to variable bounds:

$$x_1 + x_2 + x_3 \leq \frac{9}{5} + 1 + \frac{17}{8} = \frac{197}{40} < 6$$

Example: removing redundant constraints

Continuing with the same linear program

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 \leq 15 \\
8x_1 + 3x_2 - x_3 \geq 9 \\
x_1 + x_2 + x_3 \leq 6 \\
0 \leq x_1 \leq \frac{9}{5} \\
0 \leq x_2 \leq 1 \\
1 \leq x_3 \leq \frac{17}{8}
\end{aligned}
$$

3rd constraint is **redundant** due to variable bounds:

$$
x_1 + x_2 + x_3 \leq \frac{9}{5} + 1 + \frac{17}{8} = \frac{197}{40} < 6
$$

$\rightsquigarrow$

Removing redundant constraint:

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 \leq 15 \\
8x_1 + 3x_2 - x_3 \geq 9 \\
0 \leq x_1 \leq \frac{9}{5} \\
0 \leq x_2 \leq 1 \\
1 \leq x_3 \leq \frac{17}{8}
\end{aligned}
$$

Example: fixing variables

Continuing with the same linear program

$$\max 2x_1 + x_2 - x_3$$
$$5x_1 - 2x_2 + 8x_3 \leq 15$$
$$8x_1 + 3x_2 - x_3 \geq 9$$
$$0 \leq x_1 \leq \frac{9}{5}$$
$$0 \leq x_2 \leq 1$$
$$1 \leq x_3 \leq \frac{17}{8}$$

Increasing x_2 improves objective and makes all constraints less tight

Example: fixing variables

Continuing with the same linear program

$$\begin{aligned} \max 2x_1 + x_2 - x_3 \\ 5x_1 - 2x_2 + 8x_3 \leq 15 \\ 8x_1 + 3x_2 - x_3 \geq 9 \\ 0 \leq x_1 \leq \frac{9}{5} \\ 0 \leq x_2 \leq 1 \\ 1 \leq x_3 \leq \frac{17}{8} \end{aligned}$$

Increasing x_2 improves objective and makes all constraints less tight

$\rightsquigarrow$

Fix x_2 to its upper bound

$$\begin{aligned} \max 2x_1 + 1 - x_3 \\ 5x_1 + 8x_3 \leq 17 \\ 8x_1 - x_3 \geq 6 \\ 0 \leq x_1 \leq \frac{9}{5} \\ 1 \leq x_3 \leq \frac{17}{8} \end{aligned}$$

Preprocessing specific for integer linear programming

Consider now an **integer** linear program

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 \leq 15 \\
8x_1 + 3x_2 - x_3 \geq \frac{40}{7} \\
0 \leq x_1 \leq \frac{9}{5} \\
0 \leq x_2 \leq 1 \\
1 \leq x_3 \leq \frac{17}{8} \\
x_1, x_2, x_3 \in \mathbb{Z}
\end{aligned}
$$

We can replace upper bounds $9/5$ and $17/8$ by $\lfloor 9/5 \rfloor$ and $\lfloor 17/8 \rfloor$.
Similarly, replace right-hand side of Constraint 2 by $\lceil 40/7 \rceil$ (because coefficients are integral).

Preprocessing specific for integer linear programming

Consider now an **integer** linear program

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 &\leq 15 \\
8x_1 + 3x_2 - x_3 &\geq \frac{40}{7} \\
0 \leq x_1 &\leq \frac{9}{5} \\
0 \leq x_2 &\leq 1 \\
1 \leq x_3 &\leq \frac{17}{8} \\
x_1, x_2, x_3 &\in \mathbb{Z}
\end{aligned}
$$

$\rightsquigarrow$

By rounding bounds:

$$
\begin{aligned}
\max 2x_1 + x_2 - x_3 \\
5x_1 - 2x_2 + 8x_3 &\leq 15 \\
8x_1 + 3x_2 - x_3 &\geq 7 \\
0 \leq x_1 &\leq 1 \\
0 \leq x_2 &\leq 1 \\
1 \leq x_3 &\leq 2 \\
x_1, x_2, x_3 &\in \mathbb{Z}
\end{aligned}
$$

We can replace upper bounds $9/5$ and $17/8$ by $\lfloor 9/5 \rfloor$ and $\lfloor 17/8 \rfloor$.
Similarly, replace right-hand side of Constraint 2 by $\lceil 40/7 \rceil$ (because coefficients are integral).

Preprocessing specific for integer linear programming

Consider now an **integer** linear program

$$\begin{aligned} \max 2x_1 + x_2 - x_3 \\ 5x_1 - 2x_2 + 8x_3 \leq 15 \\ 8x_1 + 3x_2 - x_3 \geq \frac{40}{7} \\ 0 \leq x_1 \leq \frac{9}{5} \\ 0 \leq x_2 \leq 1 \\ 1 \leq x_3 \leq \frac{17}{8} \\ x_1, x_2, x_3 \in \mathbb{Z} \end{aligned}$$

$\rightsquigarrow$

By rounding bounds:

$$\begin{aligned} \max 2x_1 + x_2 - x_3 \\ 5x_1 - 2x_2 + 8x_3 \leq 15 \\ 8x_1 + 3x_2 - x_3 \geq 7 \\ 0 \leq x_1 \leq 1 \\ 0 \leq x_2 \leq 1 \\ 1 \leq x_3 \leq 2 \\ x_1, x_2, x_3 \in \mathbb{Z} \end{aligned}$$

We can replace upper bounds $9/5$ and $17/8$ by $\lfloor 9/5 \rfloor$ and $\lfloor 17/8 \rfloor$.
Similarly, replace right-hand side of Constraint 2 by $\lceil 40/7 \rceil$ (because coefficients are integral).

Why make bounds tighter

- Can lead to further simplifications (see e.g. redundant constraint example)
- Strengthens LP relaxation $\rightsquigarrow$ faster Branch-and-Bound (see later lectures)

Summary of linear programming preprocessing

- We have seen examples of how a linear program can be simplified by simple arguments
- These arguments can easily be turned into generic rules
- Many other and more sophisticated preprocessing rules for linear programming exist, see for example Wolsey, Section 7.6