Cuts in graphs

Given a non-empty subset $S \subset V$ of nodes in a graph $G$, a cut is the set of edges with one endnode in $S$ and the other in $V \setminus S$.

An $s - t$ cut in a graph $G$ is a cut defined by a non-empty subset $S \subset V$ such that $s \in S$ and $t \in V \setminus S$.

Paths in graphs

In a graph $G$,

- A path is a sequence of nodes $(i_1, i_2, \ldots, i_n)$ such that two consecutive nodes in the sequence form an edge: \( \{i_k, i_{k+1}\} \in E, k = 1, 2 \ldots, n - 1 \).
- A simple path is a path where no two vertices in the sequence are repeated: \( k' \neq k'' \Rightarrow i_{k'} \neq i_{k''} \).
- A cycle is a path where \( i_1 = i_n \).

All paths in this course are simple.

In a digraph $G$,

- A (directed) path is a sequence of nodes $(i_1, i_2, \ldots, i_n)$ such that two consecutive nodes in the sequence form an arc: \( (i_k, i_{k+1}) \in A, k = 1, 2 \ldots, n - 1 \).

Connectivity in graphs

- A graph $G$ is connected if, for any two nodes $i$ and $j$, there exists a path from $i$ to $j$.
- A digraph $G$ is strongly connected if, for any two nodes $i$ and $j$, there exists a directed path from $i$ to $j$.
- A graph $G$ is $k$-connected if, for any two nodes $i$ and $j$, there exist $k$ paths from $i$ to $j$ which share no edge.
**Complexity**

- The *complexity* of an algorithm $A$ solving a problem $P$ is the number of steps performed by $A$ to solve an instance of $P$.
- The complexity of $A$ varies with the instance of $P$ solved, and more simply it depends on the *size* of the instance.
- It is usually defined as a function $f(n_1, n_2, \ldots, n_k)$ of the size parameters $n_1, n_2, \ldots, n_k$.

  e.g. For graph problems, the complexity of an algorithm usually depends on $|V|$ and $|E|$, hence it is $f(|V|, |E|)$.

- The *complexity* of a problem $P$ is the complexity of the most efficient algorithm for solving $P$.

**Complexity**

Two very important classes of problems: $\mathcal{P}$ and $\mathcal{NP}$.

- $\mathcal{P}$ is the class of problems whose complexity is a *polynomial* function of the problem size.
- $\mathcal{NP}$ is the class of problems for which no algorithm is known with polynomial complexity.
  - Roughly speaking, problems in $\mathcal{P}$ are “easy” and those in $\mathcal{NP}$ are “difficult”.
  - Problems in $\mathcal{NP}$ usually have *exponential* complexity.
  - Finding a polynomial-time algorithm for a problem $P$ is a proof that the problem is easy.

**Minimum spanning tree**

One main problem in Network Design is that of connecting all nodes of a graph $G$ by selecting a subset of edges of $G$.

**Problem**

Given a graph $G = (V, E)$ and a function $c : E \rightarrow \mathbb{R}$, find the subset $E'$ of edges of $G$ such that $G' = (V, E')$ is connected.

This problem has *polynomial* complexity and is in the $\mathcal{P}$ subclass. There are two algorithms to solve it: Kruskal’s and Prim’s (p. 126 of the book).

---

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Minimum spanning tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>The complexity of an algorithm $A$ solving a problem $P$ is the number of steps performed by $A$ to solve an instance of $P$.</td>
<td>One main problem in Network Design is that of connecting all nodes of a graph $G$ by selecting a subset of edges of $G$.</td>
</tr>
<tr>
<td>The complexity of $A$ varies with the instance of $P$ solved, and more simply it depends on the size of the instance.</td>
<td>Given a graph $G = (V, E)$ and a function $c : E \rightarrow \mathbb{R}$, find the subset $E'$ of edges of $G$ such that $G' = (V, E')$ is connected.</td>
</tr>
<tr>
<td>It is usually defined as a function $f(n_1, n_2, \ldots, n_k)$ of the size parameters $n_1, n_2, \ldots, n_k$.</td>
<td>This problem has polynomial complexity and is in the $\mathcal{P}$ subclass. There are two algorithms to solve it: Kruskal’s and Prim’s (p. 126 of the book).</td>
</tr>
<tr>
<td>e.g. For graph problems, the complexity of an algorithm usually depends on $</td>
<td>V</td>
</tr>
<tr>
<td>The complexity of a problem $P$ is the complexity of the most efficient algorithm for solving $P$.</td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Minimum spanning tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>There is a subclass, called $\mathcal{NP}$-C, of $\mathcal{NP}$ of problems: the $\mathcal{NP}$-complete problems.</td>
<td></td>
</tr>
<tr>
<td>These problems are all related to one another: if somebody finds a polynomial-time algorithm for a problem of $\mathcal{NP}$-C, all of them will become easy.</td>
<td></td>
</tr>
<tr>
<td>Even more: all problems in $\mathcal{NP}$ would become easy!</td>
<td></td>
</tr>
<tr>
<td>Optimization problems in $\mathcal{NP}$-C are denoted as $\mathcal{NP}$-hard. If you find such an algorithm (and prove $\mathcal{P} = \mathcal{NP}$)</td>
<td></td>
</tr>
<tr>
<td>You will get an A in this class</td>
<td></td>
</tr>
<tr>
<td>You can win a million dollars: <a href="http://www.claymath.org/millennium/P_vs_NP/">http://www.claymath.org/millennium/P_vs_NP/</a></td>
<td></td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Minimum spanning tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Two very important classes of problems: $\mathcal{P}$ and $\mathcal{NP}$.</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{P}$ is the class of problems whose complexity is a polynomial function of the problem size.</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{NP}$ is the class of problems for which no algorithm is known with polynomial complexity. Roughly speaking, problems in $\mathcal{P}$ are “easy” and those in $\mathcal{NP}$ are “difficult”.</td>
<td></td>
</tr>
<tr>
<td>Problems in $\mathcal{NP}$ usually have exponential complexity.</td>
<td></td>
</tr>
<tr>
<td>Finding a polynomial-time algorithm for a problem $P$ is a proof that the problem is easy.</td>
<td></td>
</tr>
</tbody>
</table>
Steiner tree

Another main problem is that of connecting a subset of nodes of a graph $G$ by selecting a subset of edges of $G$.

**Problem**

Given a graph $G = (V, E)$, a subset $S \subseteq V$, and a function $c : E \to \mathbb{R}$, find the subset $E'$ of edges of $G$ such that any two nodes of $S$ are connected by at least one path.

For this problem, no polynomial algorithm is known, and it is actually an $\mathcal{NP}$-hard problem.

Minimum two-connected graph

**Problem**

Given $G = (V, E)$ and a function $c : E \to \mathbb{R}$, find the subset $E'$ of edges of $G$ such that $G' = (V, E')$ is two-connected.

This is also an $\mathcal{NP}$-hard problem.

Many network design problems we will see are Optimization problems. Most of them are $\mathcal{NP}$-hard.

The general optimization problem

Consider a vector $x \in \mathbb{R}^n$ of variables.

An optimization problem can be expressed as:

\[
\begin{align*}
\text{P} : & \quad \text{minimize} \quad f_0(x) \\
& \quad \text{such that} \quad f_1(x) \leq b_1 \\
& \quad \quad f_2(x) \leq b_2 \\
& \quad \quad \vdots \\
& \quad \quad f_m(x) \leq b_m
\end{align*}
\]

Feasible solutions, local and global optima

Define $F = \{x \in \mathbb{R}^n : f_1(x) \leq b_1, f_2(x) \leq b_2, \ldots, f_m(x) \leq b_m\}$, that is, $F$ is the feasible set of an optimization problem.

All points $x \in F$ are called feasible solutions.

A vector $x^l \in \mathbb{R}^n$ is a local optimum if

- $x^l \in F$
- there is a neighbourhood $N$ of $x^l$ with no better point than $x^l$:
  \[
  \exists N : \forall x \in N \cap F, f_0(x) \geq f_0(x^l)
  \]

A vector $x^g \in \mathbb{R}^n$ is a global optimum if

- $x^g \in F$
- there is no $x \in F$ better than $x^g$, i.e.,
  \[
  f_0(x) \geq f_0(x^g) \quad \forall x \in F
  \]
Local optima, global optima

Convex sets

Def.: A set $S \subseteq \mathbb{R}^n$ is convex if any two points $x'$ and $x''$ of $S$ are joined by a segment entirely contained in $S$:

$$\forall x', x'' \in S, \alpha \in [0, 1], \; \alpha x' + (1 - \alpha)x'' \in S$$

The intersection of two convex sets is convex.

Examples: Convex sets

- $\mathbb{R}, \mathbb{R}^2, \mathbb{R}^3$, etc. are convex
- $[a, b]$ is convex
- $\{4\}$ is convex

Examples: Nonconvex sets

- $\{0, 1\}$ is nonconvex
- $\{x \in \mathbb{R} : x \leq 2 \lor x \geq 3\}$ is nonconvex
- $\mathbb{Z}$ is nonconvex
**Convex functions**

**Def.:** A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex if, for any two points $x'$ and $x'' \in \mathbb{R}^n$ and for any $\alpha \in [0, 1]$

$$f(\alpha x' + (1 - \alpha)x'') \leq \alpha f(x') + (1 - \alpha)f(x'')$$

- The sum of convex functions is a convex function
- Multiplying a convex function by a positive scalar gives a convex function
- **Linear** functions $\sum_{i=1}^{k} a_i x_i$ are convex, irrespective of the sign of $a_i$'s.

**Definition**

**Examples**

- The function $f(x) = x$ is convex
- The function $f(x_1, x_2) = x_1 + x_2$ is convex
- The function $f(x_1, x_2) = x_1^2 + x_2$ is convex
- The function $f(x_1, x_2) = 5x_1^2 + 3x_2^2$ is convex
- The function $f(x_1, x_2) = x_1^2 + x_2^2 - x_1x_2$ is convex

- The function $f(x_1, x_2) = x_1^2 + x_2^2 + 5x_1x_2$ is nonconvex
- The function $f(x_1, x_2) = x_1^2 - x_2^2$ is nonconvex
- The function $f(x_1, x_2) = x_1x_2$ is nonconvex
- The function $f(x) = \sin x$, for $x \in [0, 2\pi]$ is nonconvex
- The function $f(x) = -x^2$ is nonconvex
Convex constraints

- A constraint $g(x) \leq b$, with $g : \mathbb{R}^n \rightarrow \mathbb{R}$, defines a subset $S$ of $\mathbb{R}^n$, that is,

  $$S = \{ x \in \mathbb{R}^n : g(x) \leq b \}$$

- the constraint $g(x) \leq b$ is convex if the set $S$ is convex.
- if the function $g(x)$ is convex, the constraint $g(x) \leq b$ is convex.
- linear constraints $\sum_{i=1}^k a_i x_i \begin{cases} \leq b \\ \geq b \end{cases}$ are convex

Convex problems

**Def.:** An optimization problem is **convex** if

- the objective function is convex
- all constraints are convex

Convex optimization problems are *easy*: any local optimum is also a global optimum.

(Hint) When modeling an optimization problem, it would be good if we found a convex problem.

Example of a convex problem

$$P: \begin{array}{l}
\text{minimize} \quad x_1^2 + 2x_2^2 \\
\text{such that} \quad x_1^2 + x_2^2 \leq 1 \\
0 \leq x_1 \leq 2 \\
1 \leq x_2 \leq 5 \\
x_2 \in \mathbb{Z}
\end{array}$$
Example of a convex problem

\[ P : \begin{align*}
\text{minimize} & \quad x_1 - 2x_2^2 \\
\text{such that} & \quad x_2^2 + x_2^2 \leq 1 \\
& \quad x_2 = 0 \\
& \quad 0 \leq x_1 \leq 5
\end{align*} \]

It is convex because \( x_2 = 0 \) can be eliminated.

Relaxation of an Optimization problem

Consider an optimization problem

\[ \begin{align*}
P : \quad \text{minimize} & \quad f_0(x) \\
\text{such that} & \quad f_1(x) \leq b_1 \\
& \quad f_2(x) \leq b_2 \\
& \quad \vdots \\
& \quad f_m(x) \leq b_m,
\end{align*} \]

or \( P : \min \{ f_0(x) : x \in F \} \) for short.

A problem \( P' : \min \{ f'_0(x) : x \in F' \} \) is a relaxation of \( P \) if:

\( F' \supseteq F \)

\( f'_0(x) \leq f_0(x) \) for all \( x \in F \).

If \( P' \) is a relaxation of a problem \( P \), then the global optimum of \( P' \) is \( \leq \) the global optimum of \( P \).

\(^1\)We don’t care what \( f'_0(x) \) is outside of \( F \).

Examples

\( \begin{align*}
\text{min} \{ f(x) : -1 \leq x \leq 1 \} \text{ is a relaxation of } \min \{ f(x) : x = 0 \} \\
\text{min} \{ f(x) : -1 \leq x \leq 1 \} \text{ is a r. of } \min \{ f(x) : 0 \leq x \leq 1 \} \\
\text{min} \{ f(x) : -1 \leq x \leq 1 \} \text{ is not a r. of } \min \{ f(x) : -2 \leq x \leq 1 \} \\
\text{min} \{ f(x) : g(x) \leq b \} \text{ is a r. of } \min \{ f(x) : g(x) \leq b - 1 \} \\
\text{min} \{ f(x) - 1 : g(x) \leq b \} \text{ is a r. of } \min \{ f(x) : g(x) \leq b \}
\end{align*} \)
Relaxation of an Optimization problem

Consider again a problem

\[ P : \min \{ f_0(x) : f_1(x) \leq b_1, f_2(x) \leq b_2, \ldots, f_m(x) \leq b_m \}, \]

or

\[ P : \min \{ f_0(x) : x \in F \} \] for short.

◮ **Deleting** a constraint from \( P \) provides a relaxation of \( P \).

◮ **Adding** a constraint \( f_{m+1}(x) \leq b_{m+1} \) to a problem \( P \) provides a restriction of \( P \), i.e. it does the opposite:

\[ F'' = \{ x \in \mathbb{R}^n : f_1(x) \leq b_1, f_2(x) \leq b_2, \ldots, f_m(x) \leq b_m, f_{m+1}(x) \leq b_{m+1} \} \subseteq F \]

and therefore

\[ \min \{ f_0(x) : x \in F'' \} \geq \min \{ f_0(x) : x \in F \} \]

Lower and upper bounds

Consider an optimization problem \( P : \min \{ f_0(x) : x \in F \} \):

◮ for any feasible solution \( x \in F \), the corresponding objective function value \( f_0(x) \) is an upper bound.

◮ the most interesting upper bounds are local optima.

◮ a lower bound of \( P \) is instead a value \( z \) such that

\[ z \leq \min \{ f_0(x) : x \in F \}. \]

What relaxations are for

◮ If \( P' \) is a relaxation of a problem \( P \), then the global optimum of \( P' \) is \( \leq \) the global optimum of \( P \).

◮ Hence, any relaxation \( P' \) of \( P \) provides a lower bound on \( P \).

⇒ If a problem \( P \) is difficult but a relaxation \( P' \) of \( P \) is easier to solve than \( P \) itself, we can still try and solve \( P' \): (i) we get a lower bound and (ii) the solution of \( P' \) may help solve \( P \).

To recap: the Knapsack problem

At a flea market in Rome, you spot \( n \) objects (old pictures, a vessel, rusty medals . . . ) that you could re-sell in your antique shop for about double the price.

◮ You want these objects to pay for your flight ticket to Rome, which cost \( C \).

◮ Also, your backpack can carry all of them, but you don’t want it heavy, so you want to buy the objects that will load your backpack as little as possible.

How do you solve this problem?
The Knapsack problem

Each object \(i = 1, 2, \ldots, n\) has a price \(p_i\) and a weight \(w_i\).

- Variables: one variable \(x_i\) for each \(i = 1, 2, \ldots, n\). This is a “yes/no” variable, i.e., either you take the \(i\)-th object or not.
- Constraint: total revenue must be at least \(C\)
  (As you’ll double the price when selling them at your store, the revenue for each object is exactly \(p_i\))
- Objective function: the total weight

Our first (non-trivial) optimization model

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} w_i x_i \\
\text{s.t.} & \quad \sum_{i=1}^{n} p_i x_i \geq C \\
& \quad x_i \in \{0, 1\} \quad \forall i = 1, 2, \ldots, n
\end{align*}
\]

Is it convex? No!

Relaxing the Knapsack problem

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} w_i x_i \\
& \quad x_i \in \{0, 1\} \quad \forall i = 1, 2, \ldots, n
\end{align*}
\]

This relaxation would give us \(x_i = 0\) for all \(i = 1, 2, \ldots, n\), and a lower bound of \(\sum_{i=1}^{n} w_i x_i = 0\). Not so great . . .

\[
\begin{align*}
\min & \quad \sum_{i=1}^{n} w_i x_i \\
& \quad \sum_{i=1}^{n} p_i x_i \geq C \\
& \quad 0 \leq x_i \leq 1 \quad \forall i = 1, 2, \ldots, n
\end{align*}
\]

Relaxing integrality of the variables gives a relaxation where we admit fractions of objects.
\(\approx\) pulverize one of the objects and take some spoonfuls of it.
It’s a relaxation, and it does give us a better lower bound.