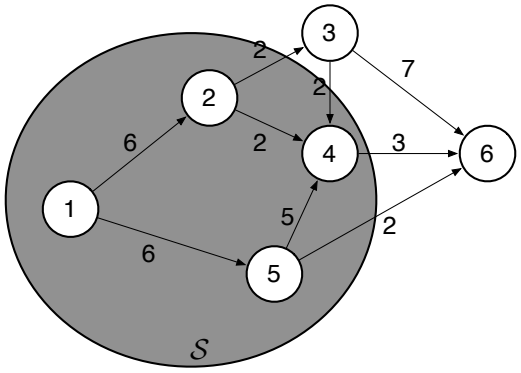
Convex relaxation

The art and science of *convex relaxation* revolves around taking a non-convex problem that you want to solve, and replacing it with a convex problem which you can actually solve – the solution to the convex program gives information about (usually a lower bound) the solution to the original program. Usually this is done by either "convexifying" the constraints or convexifying the functional – we will see examples of both below.

Minimum-Cut

Suppose that we have a directed graph with vertices indexed by $1, \ldots, N$. By convention we will denote vertex 1 as the "source" and vertex N as the "sink". Between each pair of vertices (i, j) there is a capacity $C_{i,j} \geq 0$ — if there is no edge from i to j, we take $C_{i,j} = 0$. A **cut** partitions the vertices into two sets: a $\mathcal{S}$ which contains the source, and a set $\mathcal{S}^c$ which contains the sink. The capacity of the cut is the sum of the capacities of all the edges that originate in $\mathcal{S}$ and terminate in $\mathcal{S}^C$.

In example below, we have $N = 6$ and $\mathcal{S} = \{1, 2, 4, 5\}$:



The capacity of this cut is $2 + 3 + 2 = 7$.

In general, the capacity associated with a cut $\mathcal{S}$ is

$$\sum_{i \in \mathcal{S}, j \notin \mathcal{S}} C_{i,j}.$$

If we take the vector $\boldsymbol{\nu} \in \mathbb{R}^N$ as

$$\nu_i = \begin{cases} 1, & i \in \mathcal{S}, \\ 0, & i \notin \mathcal{S} \end{cases}$$

then we can write the problem of finding the *minimum* cut as

$$\begin{aligned} \underset{\boldsymbol{\nu}}{\text{minimize}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} C_{i,j} \max(\nu_i - \nu_j, 0) \\ \text{subject to} \quad & \nu_i \in \{0, 1\}, \ i = 1, \ldots, N, \\ & \nu_1 = 1, \quad \nu_N = 0. \end{aligned}$$

To make the objective function simpler, we introduce $\lambda_{i,j}$, and the minimum cut program can be rewritten as

$$\begin{aligned} \text{(MINCUT)} \quad \underset{\boldsymbol{\Lambda}, \boldsymbol{\nu}}{\text{minimize}} \quad & \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_{i,j} C_{i,j} \\ \text{subject to} \quad & \lambda_{i,j} = \max(\nu_i - \nu_j, 0), \ i, j = 1, \ldots, N, \\ & \nu_i \in \{0, 1\}, \ i = 1, \ldots, N, \\ & \nu_1 = 1, \quad \nu_N = 0. \end{aligned}$$

We will not do so here, but one reason why the minimum-cut problem is of interest is that its dual is the *maximum-flow* problem (i.e., given a directed graph and capacity constraints, what is the largest flow possible from the source to the sink).

As it is stated, there are two things making this program nonconvex – we have non-affine equality constraints relating $\lambda_{i,j}$ to ν_i and ν_j, and we have binary constraints on ν_i. If we simply drop the integer constraint, and relax

$$\lambda_{i,j} = \max(\nu_i - \nu_i, 0) \quad \text{to} \quad \lambda_{i,j} \geq \nu_i - \nu_j \ \text{ and } \ \lambda_{i,j} \geq 0,$$

we are left with the linear program

$$\begin{aligned} \text{(LP-R)} \qquad & \underset{\boldsymbol{\Lambda},\boldsymbol{\nu}}{\text{minimize}} \ \ \langle \boldsymbol{\Lambda}, \boldsymbol{C} \rangle \\ & \text{subject to} \ \ \lambda_{i,j} \geq \nu_i - \nu_j, \ \ \lambda_{i,j} \geq 0, \ i,j = 1,\ldots,N, \\ & \qquad\qquad \nu_1 = 1, \ \nu_N = 0. \end{aligned}$$

Note that the domain we are optimizing over in the LP relaxation is larger than the domain in the original formulation – this means that every valid cut (feasible $\boldsymbol{\Lambda}, \boldsymbol{\nu}$ for the original program) is feasible in the LP relaxation. So at the very least we know that

$$\text{LP-R}^\star \ \leq \ \text{MINCUT}^\star.$$

But the semi-amazing thing is that the solutions to the two programs turn out to agree.

We show this by establishing that for every solution of the relaxation, there is at least one cut with value less than or equal to LP-R*. We do this by generating a *random cut* (with the associated probabilities carefully chosen) and show that in expectation, it is less than LP-R*.

Let Z be a uniform random variable on $[0,1]$. Let $\boldsymbol{\Lambda}^\star, \boldsymbol{\nu}^\star$ be solutions to (LP-R). Create a cut $\mathcal{S}$ with the rule:

$$\text{if } \nu_n^\star > Z, \ \text{then take } n \in \mathcal{S}.$$

The probability that a particular edge $i \to j$ is in this cut is

$$\begin{aligned} \mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) &= \mathrm{P}\left(\nu_j^\star \leq Z \leq \nu_i^\star\right) \\ &\leq \max(\nu_i^\star - \nu_j^\star, 0) \\ &\leq \lambda_{i,j}^\star, \end{aligned}$$

where the last inequality follows simply from the constraints in (LP-R). This cut is random, so its capacity is a random variable, and its expectation is

$$\begin{aligned} \mathrm{E}[\text{capacity}(\mathcal{S})] &= \sum_{i,j} C_{i,j} \, \mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) \\ &\leq \sum_{i,j} C_{i,j} \lambda_{i,j}^\star \\ &= \text{LP-R}^\star. \end{aligned}$$

Thus there must be a cut whose capacity is at most LP-R*. This establishes that

$$\text{MINCUT}^\star \leq \text{LP-R}^\star.$$

Of course, combining this with the result above means than

$$\text{MINCUT}^\star = \text{LP-R}^\star.$$

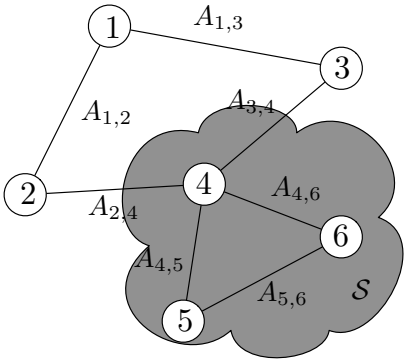
This is an example of a wonderful situation where convex relaxation costs us nothing, but makes solving the program computationally tractable.

Maximum-cut

A good resource for this section and the next is Ben-Tal and Nemirovski [BTN01].

This problem has a very similar setup as the minimum-cut problem, but it is different in subtle ways. We are given a graph; this time the edges are undirected, and have positive weights $A_{i,j}$ associated with them. Since the graph is undirected, $A_{i,j} = A_{j,i}$ and so $\boldsymbol{A}$ is symmetric. We will also assume that $A_{i,i} = 0$ for all i.

As before, a cut partitions the vertices into two sets, $\mathcal{S}$ and $\mathcal{S}^c$ – these sets can be arbitrary; there is no notion of source and sink here. For example, the cut in this example:



has value $\text{cut}(\mathcal{S}) = A_{2,4} + A_{3,4}$. The problem is to find the cut that **maximizes** the weights of the edges going between the two partitions.

We can specify a cut of the graph with a binary valued vector $\boldsymbol{x}$ of length N, where each $x_n \in \{-1, 1\}$. We set $x_n = 1$ if vertex n is in

$\mathcal{S}$ and $x_n = -1$ if vertex n is in $\mathcal{S}^c$. The value of the cut is

$$\text{cut}(\mathcal{S}) = \frac{1}{4}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}(1 - x_i x_j).$$

Note that if $x_i \neq x_j$, then $(1 - x_i x_j) = 2$, while if $x_i = x_j$, then $(1 - x_i x_j) = 0$. The factor of $1/4$ in front comes from the fact that $(1 - x_i x_j) = 2$ for edges in the cut, and that we are counting every edge twice (from i to j and again from j to i). Notice that we can write this value as a quadratic function of $\boldsymbol{x}$:

$$\text{cut}(\mathcal{S}) = \frac{1}{4}\left(\mathbf{1}^{\mathrm{T}}\boldsymbol{A}\mathbf{1} - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}\right)$$

The maximum-cut problem is find the cut with the largest value:

$$\begin{aligned} \text{(MAXCUT)} \quad & \underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{maximize}} \quad \frac{1}{4}\mathbf{1}^{\mathrm{T}}\boldsymbol{A}\mathbf{1} - \frac{1}{4}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} \\ & \text{subject to} \quad x_i \in \{-1, 1\}. \end{aligned}$$

Right now, this looks pretty gnarly, as $\boldsymbol{A}$ has no guarantee of being PSD, and we have integer constraints on $\boldsymbol{x}$. We can address the first concern by re-writing this as a search for a matrix $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$. As now $\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x} = \langle \boldsymbol{X}, \boldsymbol{A}\rangle$, we have

$$\begin{aligned} \underset{\boldsymbol{X}\in\mathbb{R}^{N\times N}}{\text{maximize}} \quad & \frac{1}{4}\mathbf{1}^{\mathrm{T}}\boldsymbol{A}\mathbf{1} - \frac{1}{4}\langle \boldsymbol{X}, \boldsymbol{A}\rangle \\ \text{subject to} \quad & \boldsymbol{X} \succeq \mathbf{0} \\ & X_{i,i} = 1, \quad i = 1, \ldots, N \\ & \text{rank}(\boldsymbol{X}) = 1. \end{aligned}$$

You should be able to convince yourself that $\boldsymbol{X}$ is feasible above if and only if it can be written as $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$, where the entries of $\boldsymbol{x}$ are ± 1.

The recast program looks like a SDP, except for the rank constraint. The relaxation, then, is to simply drop it and solve

$$\begin{aligned} \text{(MAXCUT-R)} \quad & \underset{\boldsymbol{X} \in \mathbb{R}^{N \times N}}{\text{maximize}} \quad \frac{1}{4}\mathbf{1}^{\mathrm{T}}\boldsymbol{A}\mathbf{1} - \frac{1}{4}\langle \boldsymbol{X}, \boldsymbol{A} \rangle \\ & \text{subject to} \quad \boldsymbol{X} \succeq \mathbf{0} \\ & \qquad\qquad\quad X_{i,i} = 1, \;\; i = 1, \ldots, N. \end{aligned}$$

As we are optimizing over a larger set, the optimal value of MAXCUT-R will in general be larger than MAXCUT:

$$\text{MAXCUT-R}^\star \;\geq\; \text{MAXCUT}^\star.$$

But there is a classic result [GW95] that shows it will not be too much larger:

$$\text{MAXCUT}^\star \;\geq\; (0.87856) \cdot \text{MAXCUT-R}^\star.$$

The argument again relies on looking at the expected value of a random cut. Let $\boldsymbol{X}^\star$ be a solution to MAXCUT-R. Since $\boldsymbol{X}^*$ is PSD, it can be factored as

$$\boldsymbol{X}^\star = \boldsymbol{V}^{\mathrm{T}}\boldsymbol{V}.$$

With $\boldsymbol{v}_j$ as the j^{th} column of $\boldsymbol{V}$, this means $X^\star_{i,j} = \langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle$. Since along the diagonal we have $X^\star_{i,i} = 1$, this means that $\|\boldsymbol{v}_i\|_2 = 1$ as well. We can associate one column $\boldsymbol{v}_i$ with each vertex in the original problem. To create the cut, we draw a vector $\boldsymbol{z}$ from the unit-sphere (so $\|\boldsymbol{z}\|_2 = 1$) uniformly at random,[1] and set

$$\mathcal{S} = \{i \;:\; \langle \boldsymbol{v}_i, \boldsymbol{z} \rangle \geq 0\}.$$

[1] In practice, you could do this by drawing each entry Normal$(0, 1)$ independently, then normalizing.

It should be clear that the probability that any fixed vertex is in $\mathcal{S}$ is $1/2$. But what is the probability that vertex i and vertex j are on different sides? The probability of this is simply the ratio of the angle between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ to π:

$$\mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) + \mathrm{P}\left(i \notin \mathcal{S}, j \in \mathcal{S}\right) = \frac{\arccos\langle \boldsymbol{v}_i, \boldsymbol{v}_j \rangle}{\pi} = \frac{\arccos X^\star_{i,j}}{\pi}.$$

Thus the expectation of the cut value is

$$\begin{aligned}\mathrm{E}[\mathrm{cut}(\mathcal{S})] &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\left(\mathrm{P}\left(i \in \mathcal{S}, j \notin \mathcal{S}\right) + \mathrm{P}\left(i \notin \mathcal{S}, j \in \mathcal{S}\right)\right) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\frac{\arccos X^\star_{i,j}}{2\pi}.\end{aligned}$$

There must be at least one cut that has a value greater than or equal to the mean, so we know that

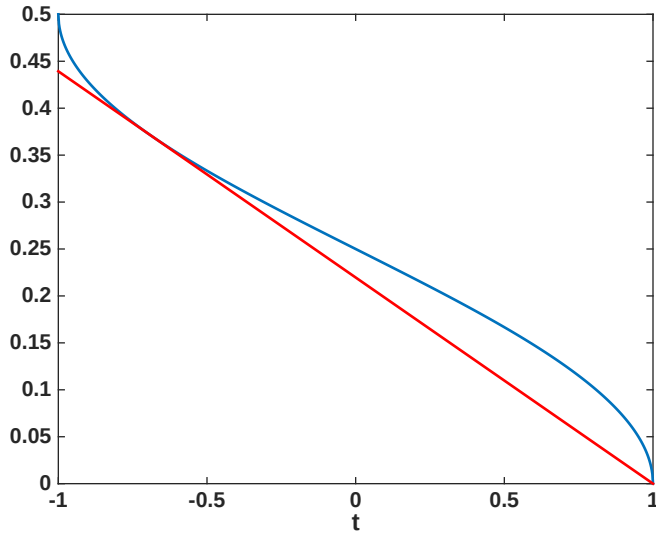
$$\mathrm{MAXCUT} \;\geq\; \mathrm{E}[\mathrm{cut}(\mathcal{S})].$$

Let's compare the terms in this sum to those in the objection function for MAXCUT-R. We know that the entries in $\boldsymbol{X}^\star$ have at most unit magnitude[2] $-1 \leq X^\star_{i,j} \leq 1$, and it is a fact that:

$$\frac{\arccos t}{2\pi} \;\geq\; (0.87856)\frac{1}{4}(1-t), \quad \text{for} \;\; t \in [-1, 1].$$

Here is a little "proof by plot" of this fact:

[2]This follows from $X^\star_{i,j} = \langle \boldsymbol{v}_i, \boldsymbol{v}_k \rangle$, $\|\boldsymbol{v}_i\|_2 = 1$, and Cauchy-Swartz.



$$\text{blue} = \frac{\arccos t}{2\pi}, \text{ red} = (0.878856)\frac{1}{4}(1-t).$$

Thus

$$\begin{aligned}
\text{MAXCUT}^\star &\geq \text{E}[\text{cut}(\mathcal{S})] \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} A_{i,j}\frac{\arccos X^\star_{i,j}}{2\pi} \\
&\geq (0.87856)\sum_{i=1}^{N}\sum_{j=1}^{N}\frac{1}{4}A_{i,j}(1-X^\star_{i,j}) \\
&= (0.87856)\cdot \text{MAXCUT-R}^\star
\end{aligned}$$

Quadratic equality constraints

The integer constraint $x_i \in \{-1, 1\}$ in the example above might also be interpreted as a quadratic *equality* constraint:

$$x_i \in \{-1, 1\} \quad \Leftrightarrow \quad x_i^2 = 1.$$

As we are well aware, quadratic (or any other nonlinear) equality constraints make the feasibility region nonconvex.

We consider general nonconvex quadratic programs of the form

$$\begin{array}{ll} \underset{\boldsymbol{x} \in \mathbb{R}^N}{\text{minimize}} & \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A}_0 \boldsymbol{x} + 2\langle \boldsymbol{x}, \boldsymbol{b}_0 \rangle + c_0 \\ \text{subject to} & \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A}_m \boldsymbol{x} + 2\langle \boldsymbol{x}, \boldsymbol{b}_m \rangle + c_m = 0, \quad m = 1, \ldots, M, \end{array}$$

where the $\boldsymbol{A}_m$ are symmetric, but not necessarily $\succeq \boldsymbol{0}$. We will show how to recast these problems as optimization over the SDP cone with an additional (nonconvex) rank constraint. Then we will have a natural convex relaxation by dropping the rank constraint. This general methodology works for equality or (possibly nonconvex) inequality constraints, but for the sake of simplicity, we will just look at equality constraints.

We can turn a quadratic form into a trace inner product with a rank 1 matrix as follows. It is clear that

$$\begin{aligned} \boldsymbol{x}^{\mathrm{T}} \boldsymbol{A} \boldsymbol{x} + 2\boldsymbol{b}^{\mathrm{T}} \boldsymbol{x} + c &= \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^{\mathrm{T}} & c \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \\ &= \operatorname{trace}\left(\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{b}^{\mathrm{T}} & c \end{bmatrix} \boldsymbol{X}_x \right), \quad \boldsymbol{X}_x = \begin{bmatrix} \boldsymbol{x} \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}} & 1 \end{bmatrix}. \end{aligned}$$

This means we can write the nonconvex quadratic program as

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^N}{\text{minimize}} \quad & \operatorname{trace}\left(\begin{bmatrix} \boldsymbol{A}_0 & \boldsymbol{b}_0 \\ \boldsymbol{b}_0^{\mathrm{T}} & c_0 \end{bmatrix} \boldsymbol{X}_x\right) \\ \text{subject to} \quad & \operatorname{trace}\left(\begin{bmatrix} \boldsymbol{A}_m & \boldsymbol{b}_m \\ \boldsymbol{b}_m^{\mathrm{T}} & c_m \end{bmatrix} \boldsymbol{X}_x\right) = 0, \quad m = 1, \ldots, M. \end{aligned}$$

With

$$\boldsymbol{F}_m = \begin{bmatrix} \boldsymbol{A}_m & \boldsymbol{b}_m \\ \boldsymbol{b}_m^{\mathrm{T}} & c_m \end{bmatrix},$$

we see that this program is equivalent to

$$\begin{aligned} \underset{\boldsymbol{X}\in\mathbb{R}^{N\times N}}{\text{minimize}} \quad & \langle \boldsymbol{X}, \boldsymbol{F}_0 \rangle \\ \text{subject to} \quad & \langle \boldsymbol{X}, \boldsymbol{F}_m \rangle = 0, \quad m = 1, \ldots, M \\ & \boldsymbol{X} \succeq \boldsymbol{0} \\ & \operatorname{rank}(\boldsymbol{X}) = 1. \end{aligned}$$

Again, we can get a convex relaxation simply by dropping the rank constraint. How well this works depends on the particulars of the problem. There are certain situations where it is exact; one of these is when there is a single non-convex inequality constraint.There are other situations where it is not exact but is provably good – one example is maximum-cut. There are other situations where it is arbitrarily bad.

Example: Phase retrieval

In coherent imaging applications, a not uncommon problem is to reconstruct an unknown vector $\boldsymbol{x}$ from measurements of the *magnitude* of a series of linear functionals. We observe

$$y_m = |\langle \boldsymbol{x}, \boldsymbol{a}_m \rangle|^2 \ \ (+ \text{ noise}), \quad m = 1, \ldots, M.$$

For instance, if $\boldsymbol{a}_m$ are Fourier vectors, we are observing samples of the magnitude of the Fourier transform of $\boldsymbol{x}$. If we also measured the phase, then recovering $\boldsymbol{x}$ is a standard linear inverse problem (and if we have a complete set of samples in the Fourier domain, you can just take an inverse Fourier transform). But since we do not get to see the phase, we have to estimate it along with the underlying $\boldsymbol{x}$ – this problem is often referred to as **phase retrieval**.

We can rewrite the measurements as

$$\begin{aligned} y_m = \langle \boldsymbol{a}_m, \boldsymbol{x} \rangle \langle \boldsymbol{x}, \boldsymbol{a}_m \rangle = \boldsymbol{x}^{\mathrm{H}} \boldsymbol{a}_m \boldsymbol{a}_m^{\mathrm{H}} \boldsymbol{x} &= \operatorname{trace}(\boldsymbol{a}_m \boldsymbol{a}_m^{\mathrm{H}} \boldsymbol{x} \boldsymbol{x}^{\mathrm{H}}) \\ &= \langle \boldsymbol{X}, \boldsymbol{A}_m \rangle_F, \end{aligned}$$

where $\boldsymbol{A}_m = \boldsymbol{a}_m \boldsymbol{a}_m^{\mathrm{H}}$ and $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^{\mathrm{H}}$. So solving the phase retrieval problem is the same as finding an $N \times N$ matrix with the following properties:

$$\langle \boldsymbol{X}, \boldsymbol{A}_m \rangle_F = y_m, \quad m = 1, \ldots, M, \quad \boldsymbol{X} \succeq \boldsymbol{0}, \quad \operatorname{rank}(\boldsymbol{X}) = 1.$$

The first condition is just that $\boldsymbol{X}$ obeys a certain set of linear equality constraints; the second is that $\boldsymbol{X}$ is in the SDP cone; the third is a nonconvex constraint.

One convex relaxation for this problem simply drops the rank constraint and finds a feasible point that obeys the first two conditions. Under certain conditions on the $\boldsymbol{a}_m$, there will only be one point in this intersection once M is mildly larger than N.

References

[BTN01] A. Ben-Tal and A. Nemirovski. *Lectures on Modern Covex Optimization*. SIAM, 2001.

[GW95] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comp. Mach.*, 42(6):1115–1145, November 1995.