# Optimization for decoding

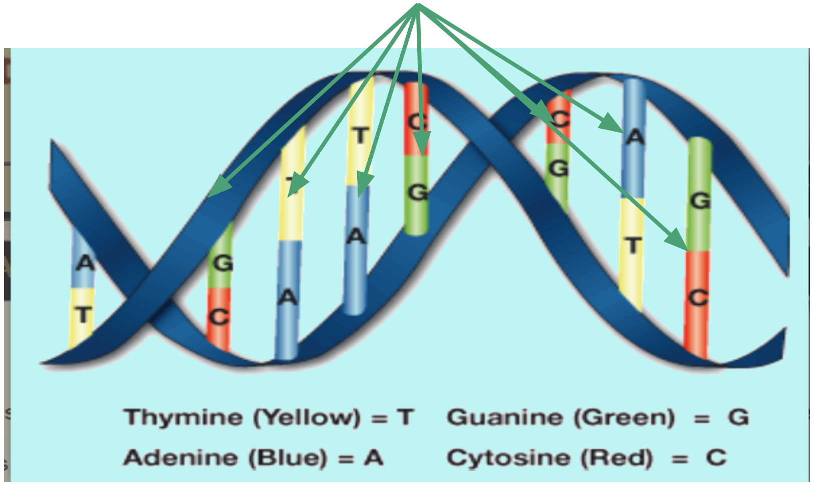
Speaker: Issam Laradji

# Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
    - Determine the assignment of *n* random variables $X_1, \ldots X_n$

$$\chi_i = \{0, 1, \ldots, k-1\}$$

$$X_1, \ldots X_n$$



# Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
  - Determine the assignment of *n* random variables $X_1, \ldots X_n$

$$\chi_i = \{0, 1, \ldots, k-1\}$$

- The maximum a posteriori (MAP) - or the most likely labeling

$$\arg\max_x p_w(x) = \arg\max_x w^T F(x)$$

# Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
  - Determine the assignment of *n* random variables $X_1, \ldots X_n$

$$\chi_i = \{0, 1, \ldots, k-1\}$$

- The maximum a posteriori (MAP) - or the most likely labeling

$$\arg\max_x p_w(x) = \arg\max_x w^T F(x)$$

- No computation of entropy required

# Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
    - Determine the assignment of *n* random variables $X_1, \ldots X_n$

$$\chi_i = \{0, 1, \ldots, k-1\}$$

- The maximum a posteriori (MAP) - or the most likely labeling

$$\arg\max_x p_w(x) = \arg\max_x w^T F(x)$$

- No computation of entropy required

$$\sup_{\mu \in \mathcal{M}} \{w^T \mu + H(p_\mu)\}$$

# Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
  - Determine the assignment of *n* random variables $X_1, \ldots X_n$

$$\chi_i = \{0, 1, \ldots, k-1\}$$

- The maximum a posteriori (MAP) - or the most likely labeling

$$\arg\max_x p_w(x) = \arg\max_x w^T F(x)$$

- No computation of entropy required

$$\sup_{\mu \in \mathcal{M}} \{w^T \mu + H(p_\mu)\}$$

# Problem Statement

- For pairwise UGM the decoding problem is integer-quadratic programming,

$$\arg\max_{x \in \{0,1\}^k} p_w(x) = \arg\max_{x \in \{0,1\}^k} w^T F(x)$$

- Because

$$w^T F(x) = \sum_{s \in V} w_s x_s + \sum_{(s,t) \in E} w_{st} x_s x_t$$

- Integer Programming is NP-hard
- Cast (relax) it to have  linear constraints

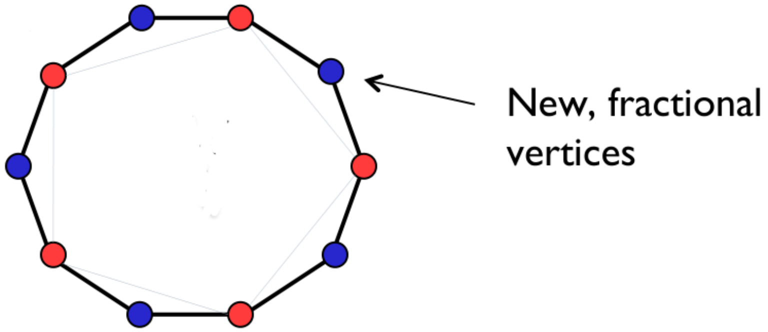
# Problem Statement

- Quadratic unconstrained binary optimization (QUBO)

$$\arg\max_{x\in\{0,1\}^k} p_w(x) = \arg\max_{x\in\{0,1\}^k} w^T F(x)$$

- Quadratic programming,

$$\arg\max_{x\in[0,1]^k} p_w(x) = \arg\max_{x\in[0,1]^k} w^T F(x)$$



# Quadratic programming to linear programming

- Quadratic programming,

$$\arg\max_{x \in [0,1]^k} p_w(x) = \arg\max_{x \in [0,1]^k} w^T F(x)$$

- Achieve a tighter bound by converting it to a linear program (best paper award)

$$\arg\max_{x \in \{0,1\}^k} w^T F(x) = \arg\max_{\mu \in M(G)} w^T \mu$$

$$M(G) = \{\mu \in R^d \mid \exists w \in R^d \text{ s.t. } \mu = \mathbb{E}_{P(x,w)}[F(x)]\}$$

# Proof for equivalence

$$\max_p \sum_x p(x) w^T F(x) = \max_x w^T F(x)$$

Original formulation

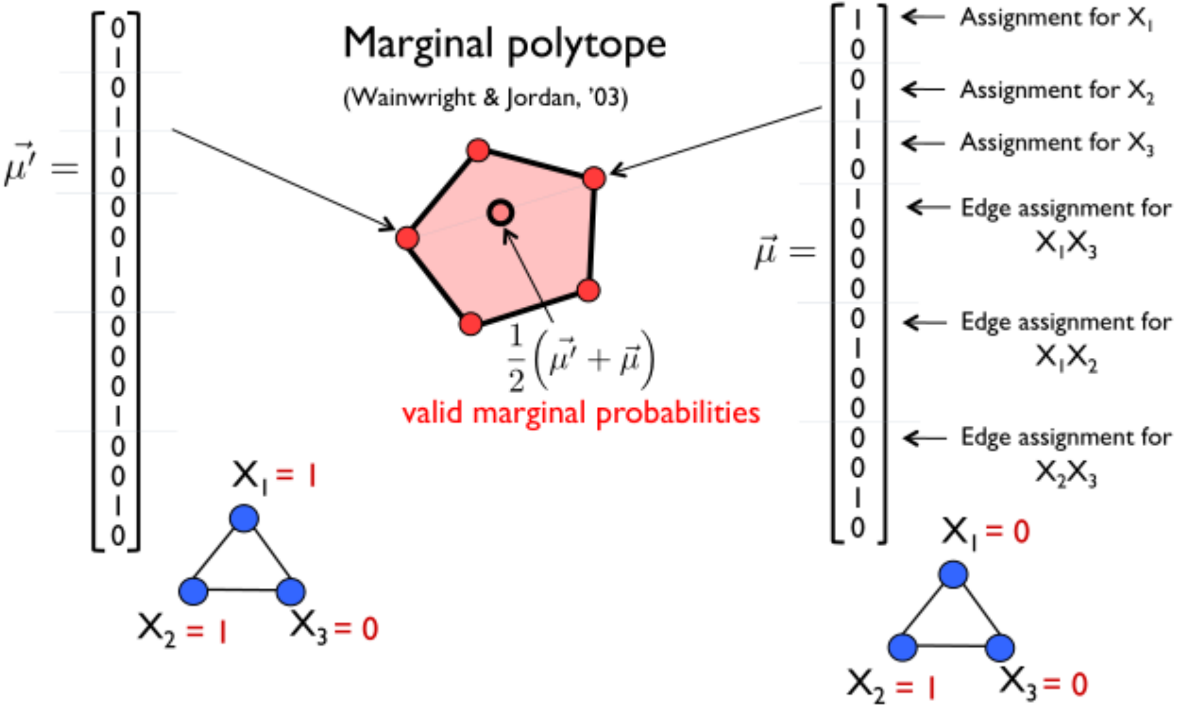
$$\max_p \sum_x p(x) w^T F(x) = \max_p w^T \sum_x p(x) F(x)$$

$$= \max_p w^T \mathbb{E}[F(x)]$$

$$= \max_p w^T \mu_p$$

$$= \max_{\mu \in M(G)} w^T \mu$$

Desired formulation




Marginal Polytope: the set of mean vectors that can arise from some joint distribution $\mu$

The dimension of $\mu$ is $2|V| + 2^2|E|$

# Linear programming global constraint

- Global constraint: The edge marginals in $\mu$ must arise from a common joint distribution

$$\max_{\mu \in M(G)} w^T \mu$$

- Number of constraints is exponential in the number of edges

- Therefore we relax the linear program to achieve a polynomial time approximation

# First-order relaxation

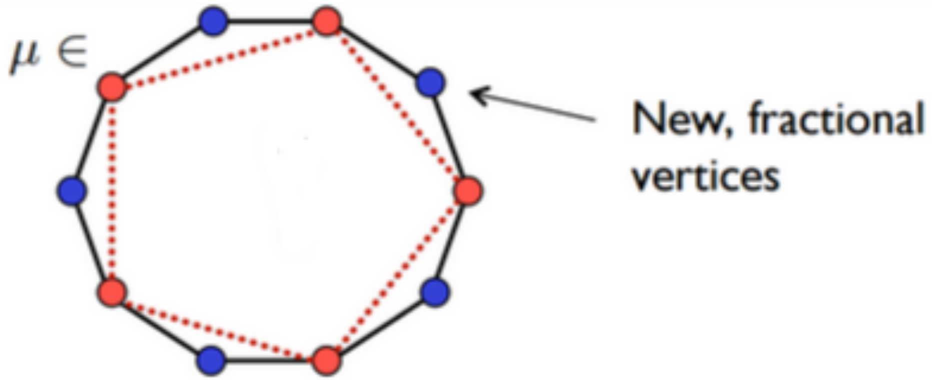
- Pairwise relaxation (Assume dependency between pairs only)

$$\text{LOCAL}(G) = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \mu_i(x_i) \geq 0, \quad \mu_{ij}(x_i, x_j) \geq 0 & \end{array} \right\}$$

$$\max_{\mu \in M(G)} w^T \mu \approx \max_{\mu \in \text{LOCAL}(G)} w^T \mu$$

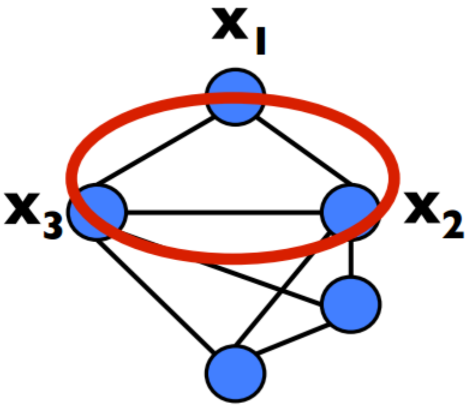
# First-order relaxation

$$\mathrm{LOCAL}(G) = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \mu_i(x_i) \geq 0, \quad \mu_{ij}(x_i, x_j) \geq 0 & \end{array} \right\}$$



# Higher order relaxations

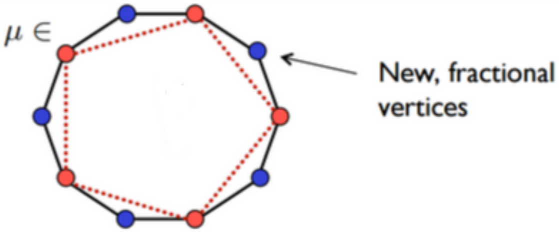
- Edge marginals are consistent on larger subsets



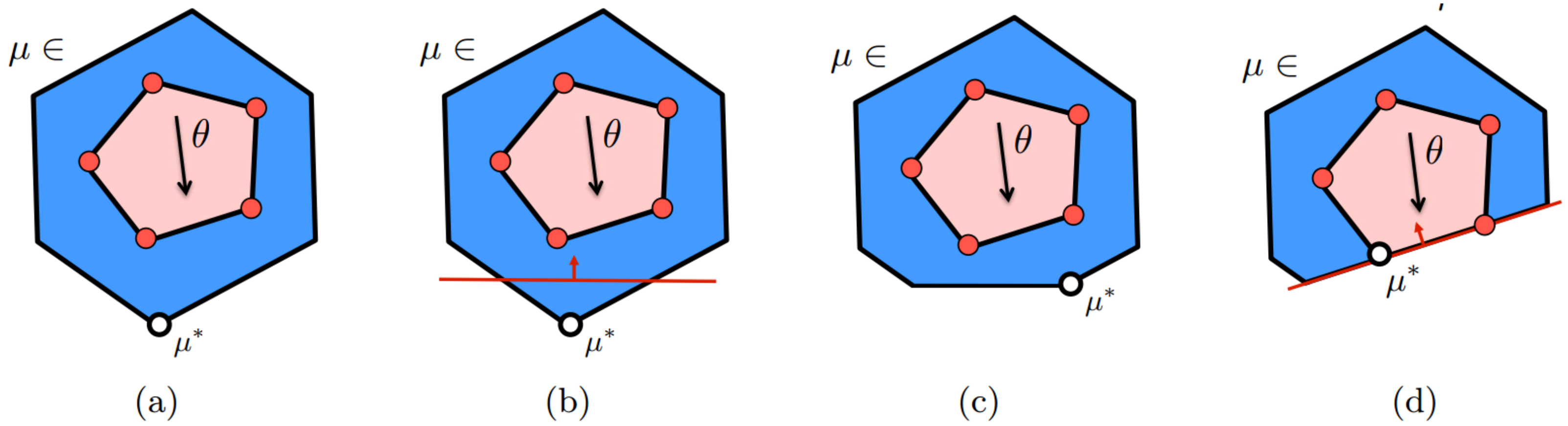

$$\mathrm{TRI}(G) = \left\{ \boldsymbol{\mu} \geq 0 \,\middle|\, \exists \boldsymbol{\tau} \geq 0, \begin{array}{ll} \sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) & \forall ij \in E, x_i \\ \sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) & \forall ij \in E, x_j \\ \sum_{x_i} \mu_i(x_i) = 1 & \forall i \in V \\ \tau_{ij}(x_i, x_j) = \mu_{ij}(x_i, x_j) & \forall ij \in E, x_i, x_j \\ \sum_{x_k} \tau_{ijk}(x_i, x_j, x_k) = \tau_{ij}(x_i, x_j), & \forall i, j, k \end{array} \right\}.$$

- Constraints grow exponentially in the size of the clusters considered

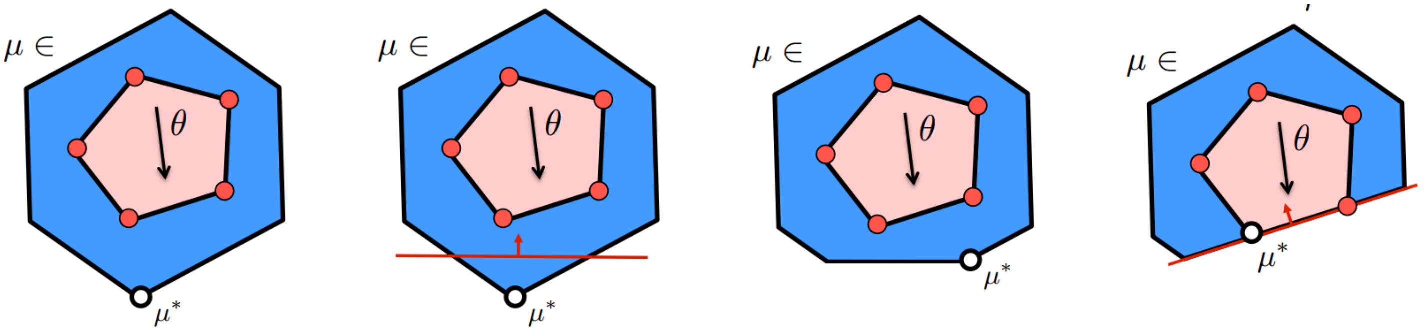
# Cutting-plane algorithm



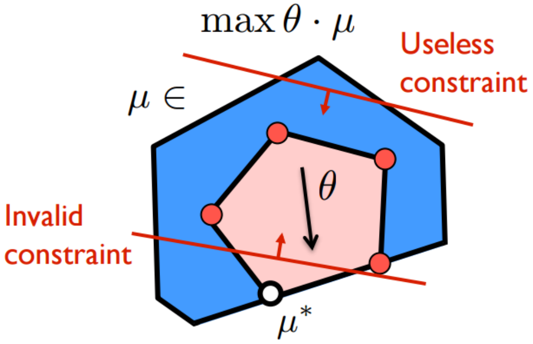

- Few carefully chosen constraints would suffice
  - An integer solution is a MAP assignment
- Solve pairwise LP then find valid constraints to add to the relaxation



# Cutting-plane algorithm




(a) (b) (c) (d)



# Cutting-plane algorithm

- Add valid cycle constraints - cycle of the graph should be consistent with some joint distribution

$$\text{CYCLE}(C) = \left\{ \boldsymbol{\mu} \in \mathbb{R}^d \;\middle|\; \exists \boldsymbol{\tau}_C \geq 0, \begin{array}{ll} \sum_{\mathbf{x}_{C \setminus i,j}} \tau_C(\mathbf{x}_C) = \mu_{ij}(x_i, x_j) & \forall ij \in C, x_i, x_j \\ \sum_{\mathbf{x}_C} \tau_C(\mathbf{x}_C) = 1 & \end{array} \right\}.$$

[1] We use $C$ to refer to both a set of edges (e.g., with notation $ij \in C$), and the variables involved in these edges. The notation $\mathbf{x}_C$ refers to an assignment to all of the variables in the cycle $C$, and $C \backslash \{i, j\}$ refers to the set of variables in $C$ except for $i$ or $j$. Also, $\sum_{\mathbf{x}_{C \setminus i,j}} \tau_C(\mathbf{x}_C)$ means the sum over all assignments $\mathbf{x}_{C \setminus i,j}$ to the variables in $C \backslash \{i, j\}$ of $\tau_C(\mathbf{x}_{C \setminus i,j}, x_i, x_j)$, where $x_i$ and $x_j$ are instantiated outside of the sum.

# Cutting-plane algorithm

1. Solve the LP relaxation (in Iteration 1, use the pairwise relaxation).
2. Construct the projection graph $G_\pi$ and pseudomarginals $\boldsymbol{\mu}_\pi$ using Eqs. 3.9-3.11.
3. Run **SeperateCycles**$(G_\pi, \boldsymbol{\mu}_\pi)$ to see if there are any violated cycle inequalities.
4. Add all cycle inequalities returned by Step 3 to the LP relaxation.
5. Return to Step 1, but now solve using the tighter relaxation.

2. Construct the projection graph $G_\pi$ and pseudomarginals $\boldsymbol{\mu}_\pi$ using Eqs. 3.9-3.11.

$$V_\pi = \bigcup_{i \in V} \pi_i, \qquad E_\pi \subseteq \{(\pi_i^q, \pi_j^r) \mid (i,j) \in E, q \leq |\pi_i|, r \leq |\pi_j|\}. \quad \pi_i^q : \chi_i \to \{0,1\}$$

$$\mu_m^\pi(x_m) = \sum_{s_i \in \chi_i \,:\, \pi_i^q(s_i) = x_m} \mu_i(s_i) \qquad \forall m = \pi_i^q \in V_\pi$$

$$\mu_{mn}^\pi(x_m, x_n) = \sum_{\substack{s_i \in \chi_i \,:\, \pi_i^q(s_i) = x_m, \\ s_j \in \chi_j \,:\, \pi_j^r(s_j) = x_n}} \mu_{ij}(s_i, s_j) \qquad \forall mn = (\pi_i^q, \pi_j^r) \in E_\pi.$$

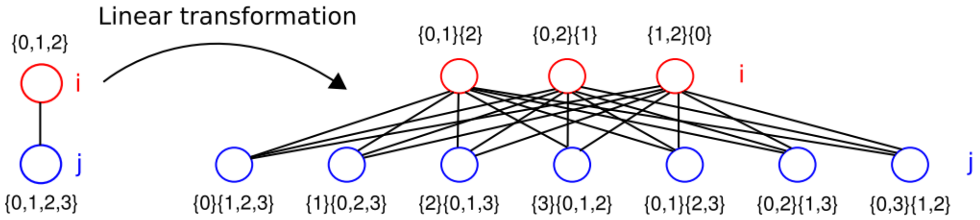



Figure 3-2: Illustration of the projection for one edge $ij \in E$ where $\chi_i = \{0,1,2\}$ and $\chi_j = \{0,1,2,3\}$. The projection graph (shown on right), has 3 partitions for $i$ and 7 for $j$.

3. Run **SeperateCycles**($G_\pi$, $\boldsymbol{\mu}_\pi$) to see if there are any violated cycle inequalities.

**Algorithm SeparateCycles**($G_\pi$, $\boldsymbol{\mu}_\pi$)

// Initialize the auxilliary graph used in the shortest path computation.
**let** $G' = (V', E')$, **where**
    $V' = \cup_{i \in V_\pi} \ \{i_1, i_2\}$
    $E' = \cup_{(i,j) \in E_\pi} \ \{(i_1, i_2), (i_1, j_2), (i_2, j_1), (j_2, j_2)\}$

// Setup the edge weights.
**for** each edge $ij \in E_\pi$
    $w(i_1, j_2) = \mu^\pi_{ij}(0,0) + \mu^\pi_{ij}(1,1)$ // Cut
    $w(i_2, j_1) = \mu^\pi_{ij}(0,0) + \mu^\pi_{ij}(1,1)$ // Cut
    $w(i_1, j_1) = \mu^\pi_{ij}(0,1) + \mu^\pi_{ij}(1,0)$ // Not cut
    $w(i_2, j_2) = \mu^\pi_{ij}(0,1) + \mu^\pi_{ij}(1,0)$ // Not cut

// Run the shortest path algorithm, once for each node.
**for** each node $i \in V_\pi$
    // Find shortest path $P_i$ from $i_1$ to $i_2$ on graph $G'$ with weights $w$
    $P_i$ = ShortestPath($i_1$, $i_2$, $G'$, $w$)

    // Make sure that this is a simple cycle in $G_\pi$.
    **if** $\exists j \neq i$ **such that** $j_1, j_2 \in P_i$
        Discard $P_i$.

**return** $\{P_i : w(P_i) < 1\}$

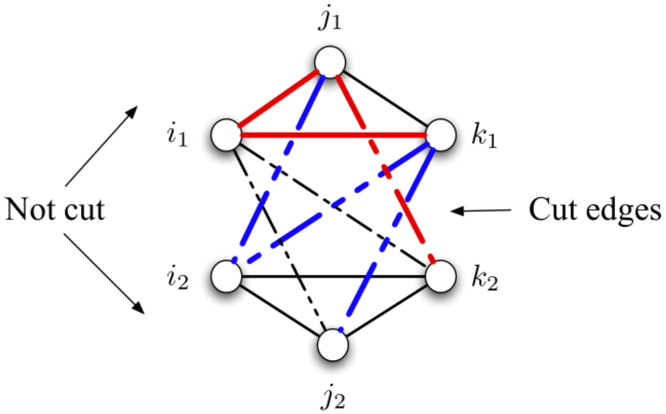



Figure 3-4: Given the projection graph $G_\pi = (V_\pi, E_\pi)$ and edge pseudomarginals $\boldsymbol{\mu}_\pi$, find the most violated cycle inequality.

3. Run **SeperateCycles**$(G_\pi, \boldsymbol{\mu}_\pi)$ to see if there are any violated cycle inequalities.

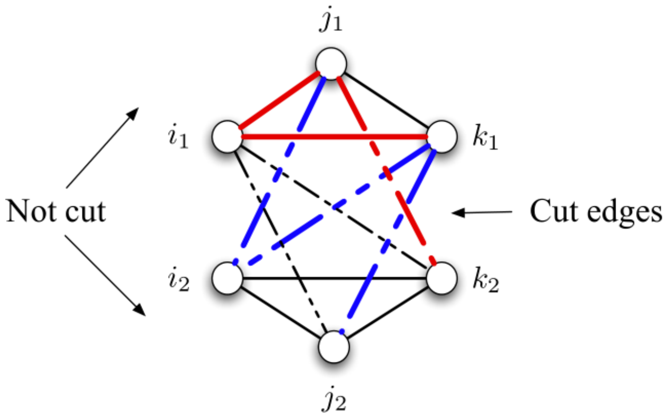



Figure 3-5: Illustration of graph used in shortest path algorithm for finding the most violated cycle inequality. The dashed edges denote *cut* edges (i.e., if used in the shortest path, they are assigned to $F$), while the solid edges denote edges that are *not cut*. The algorithm is as follows: To find the most violated cycle inequality on a cycle involving node $j$, find the shortest path from $j_1$ to $j_2$ in the graph (edge weights are discussed in Section 3.4.2). To find the most violated cycle inequality overall, considering *all* cycles, repeat this for every node (e.g., also look for the shortest path from $k_1$ to $k_2$). The red and blue paths demonstrate two different cycle inequalities. The red path, from $k_1$ to $k_2$, denotes the cycle inequality $C = \{ki, ij, jk\}$, $F = \{jk\}$. The blue path, from $j_1$ to $j_2$, denotes the cycle inequality $C = \{ji, ik, kj\}$, $F = C$, i.e. all three edges are cut. Since the paths begin in the top component and end in the bottom component, each path must have an odd number of cut edges. Thus, $|F|$ is always odd, as required to obtain a valid inequality.

# Cutting-plane algorithm

1. Solve the LP relaxation (in Iteration 1, use the pairwise relaxation).
2. Construct the projection graph $G_\pi$ and pseudomarginals $\boldsymbol{\mu}_\pi$ using Eqs. 3.9-3.11.
3. Run **SeperateCycles**$(G_\pi, \boldsymbol{\mu}_\pi)$ to see if there are any violated cycle inequalities.
4. Add all cycle inequalities returned by Step 3 to the LP relaxation.
5. Return to Step 1, but now solve using the tighter relaxation.

# Semidefinite programming

- Some NP−hard combinatorial optimization problems have convex relaxations that are semidefinite programs.

- SDP relaxation is very tight in practice

$$
\begin{aligned}
SDP: \quad \text{minimize} \quad & C \bullet X \\
\text{s.t.} \quad & A_i \bullet X = b_i \quad, i = 1, \ldots, m, \\
& X \succeq 0,
\end{aligned}
$$

# Semidefinite programming

$$SDP: \quad \text{minimize} \quad C \bullet X$$
$$\text{s.t.} \quad A_i \bullet X = b_i \quad , i = 1, \ldots, m,$$
$$X \succeq 0,$$

$$LP: \quad \text{minimize} \quad c \cdot x$$
$$\text{s.t.} \quad a_i \cdot x = b_i, \quad i = 1, \ldots, m$$
$$x \in \Re^n_+.$$

$$A_i = \begin{pmatrix} a_{i1} & 0 & \ldots & 0 \\ 0 & a_{i2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{in} \end{pmatrix}, \ i = 1, \ldots, m, \quad \text{and} \quad C = \begin{pmatrix} c_1 & 0 & \ldots & 0 \\ 0 & c_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & c_n \end{pmatrix}.$$

Linear Programming

# Maxcut problem

- Determine a subset S of the nodes N for which the sum of the weights of the edges that cross from S to its complement S¯ is maximized

$$MAXCUT: \quad \text{maximize}_x \quad \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(1 - x_i x_j)$$

$$\text{s.t.} \quad x_j \in \{-1, 1\}, \quad j = 1, \ldots, n.$$

↓

$$MAXCUT: \quad \text{maximize}_{Y,x} \quad \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - W \bullet Y$$

$$\text{s.t.} \quad Y_{jj} = 1, \quad j = 1, \ldots, n$$

$$Y = xx^T.$$

# Maxcut problem

- Remove Rank-1 restriction $Y = xx^T$

$$
\begin{aligned}
MAXCUT: \quad & \text{maximize}_{Y,x} \quad \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - W \bullet Y \\
\text{s.t.} \quad & Y_{jj} = 1, \quad j = 1, \ldots, n \\
& Y = xx^T.
\end{aligned}
$$

↓ To SDP formulation

$$
\begin{aligned}
RELAX: \quad & \text{maximize}_{Y} \quad \frac{1}{4}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij} - W \bullet Y \\
\text{s.t.} \quad & Y_{jj} = 1, \quad j = 1, \ldots, n \\
& Y \succeq 0.
\end{aligned}
$$

# Summary

- Classic decoding using Quadratic programming with integer constraints
  - Can be relaxed to Quadratic programming with linear constraints (polynomial time for pairwise)
- Quadratic programming to linear programming (tighter)
  - Relaxation based on local consistency of mean vectors
  - Higher-order relaxation leads to an NP-hard optimization problem

- Cutting-plane algorithm for tight solutions that is computationally tractable
  - Find and add violated constraints to the optimization

- NP−hard optimization problems like MAXCUT have convex relaxations that are semidefinite programs, which are very tight in practice.

# Thank you!