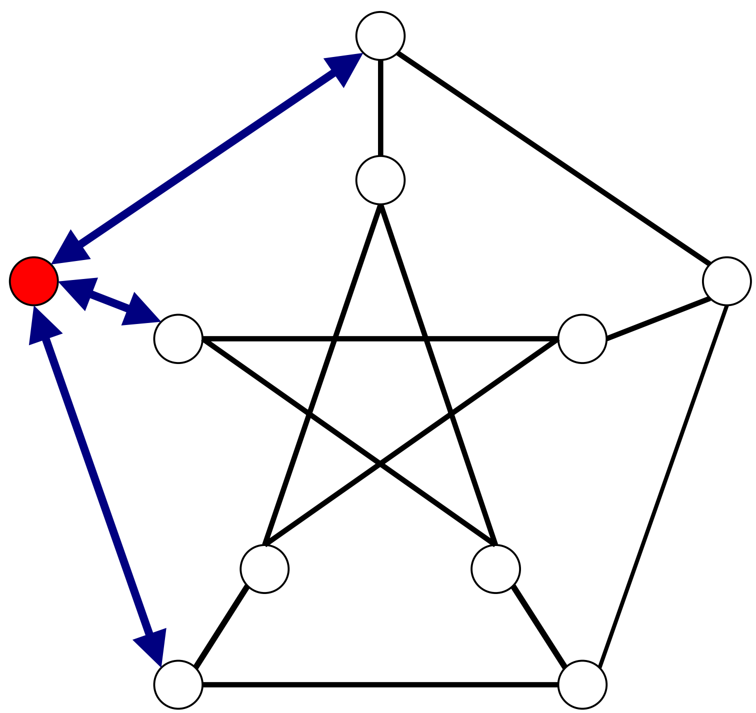
# Distributing semidefinite relaxations

Ben Recht
MIT
November 19, 2003

# Hard problems on networks

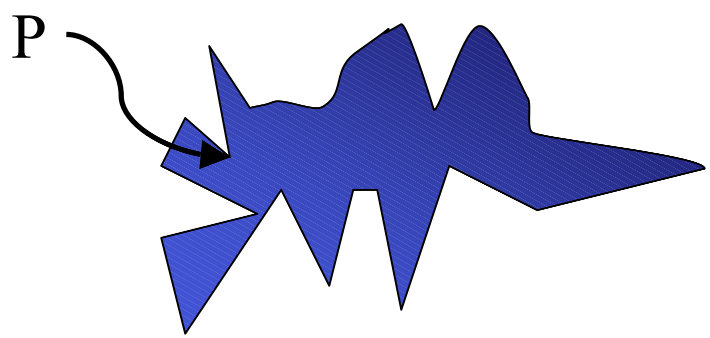


- Communication with neighbors
- Local computation
- Global objective

- **Can we solve the optimization efficiently?**

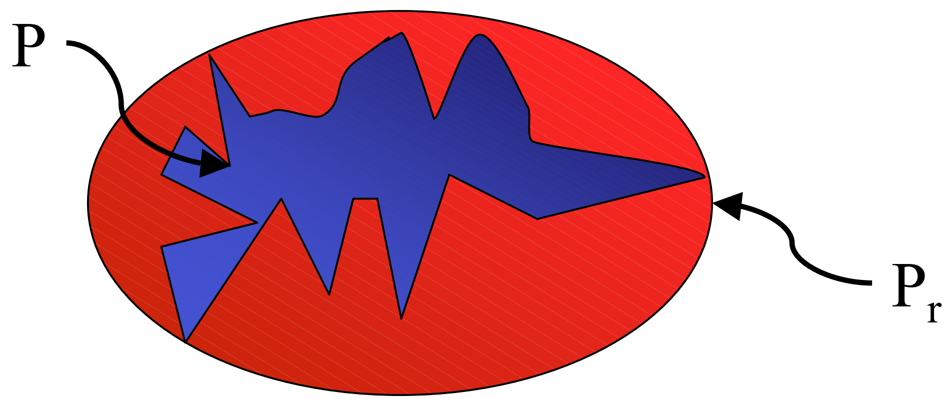
# Relaxations: Settling for Sufficiency

- Given a hard problem P



# Relaxations: Settling for Sufficiency

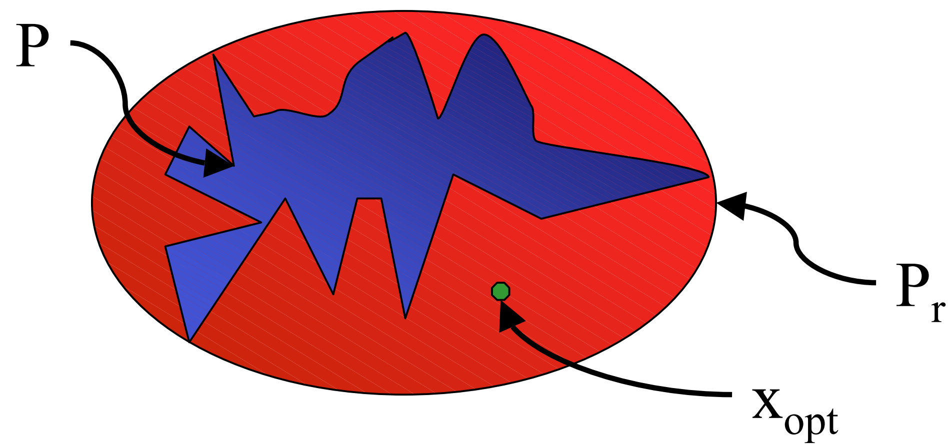
- Given a hard problem P, come up with problem $P_r$ such that




  - We can efficiently, locally find a solution for $P_r$

# Relaxations: Settling for Sufficiency

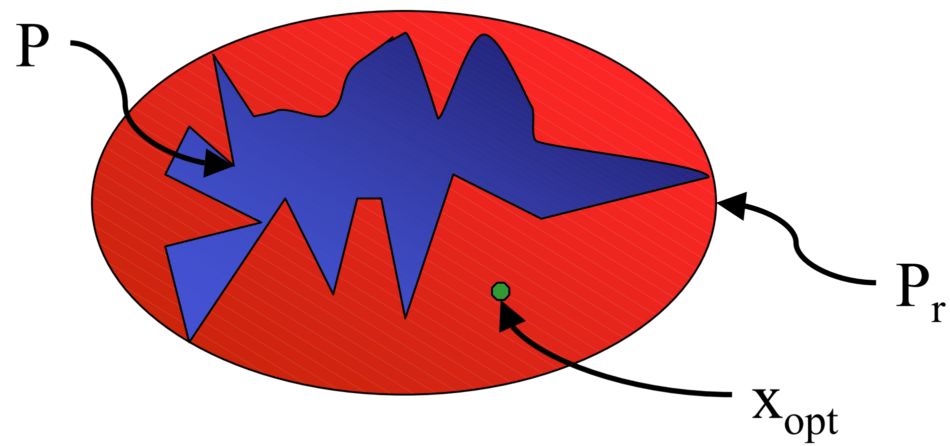
- Given a hard problem P, come up with problem $P_r$ such that




  - We can efficiently, locally find a solution for $P_r$
  - We can efficiently determine if this solution is valid for P

# Relaxations: Settling for Sufficiency

- Given a hard problem P, come up with problem $P_r$ such that




  - We can efficiently, locally find a solution for $P_r$
  - We can efficiently determine if this solution is valid for P
  - If no valid solution exists, we hope to get a bound on the complexity of P

# Distributed approximations

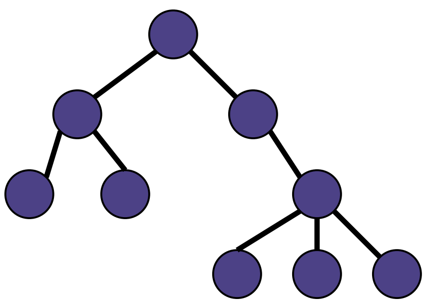
- Hierarchies of approximations
- Linear (Sherali-Adams) vs. Semidefinite (Parillo, Lasserre)

$$
\begin{array}{ccccccc}
\mathcal{L}_1 & \subseteq & \mathcal{L}_2 & \subseteq & \ldots & \subseteq & \mathcal{L}_V \\
\rotatebox{90}{$\supseteq$} & & \rotatebox{90}{$\supseteq$} & & & & \| \\
\mathcal{S}_1 & \subseteq & \mathcal{S}_2 & \subseteq & \ldots & \subseteq & \mathcal{S}_V
\end{array}
$$

fast/easy hard slog

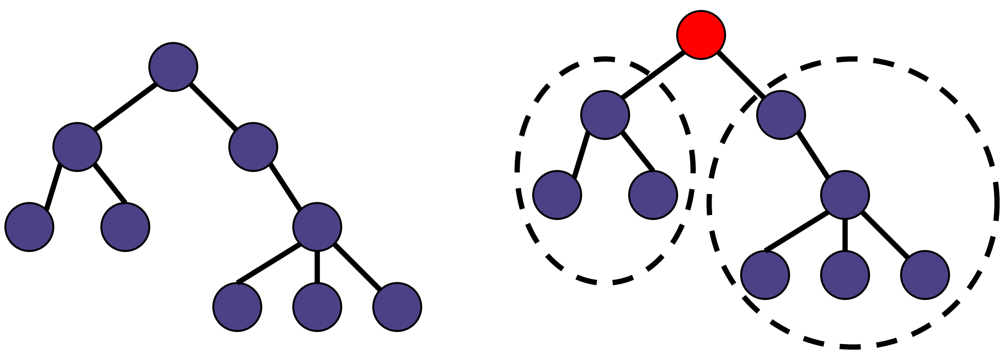
# Linear = Belief Propagation

- Key magic: Dynamic Programming



# Linear = Belief Propagation

- Key magic: Dynamic Programming

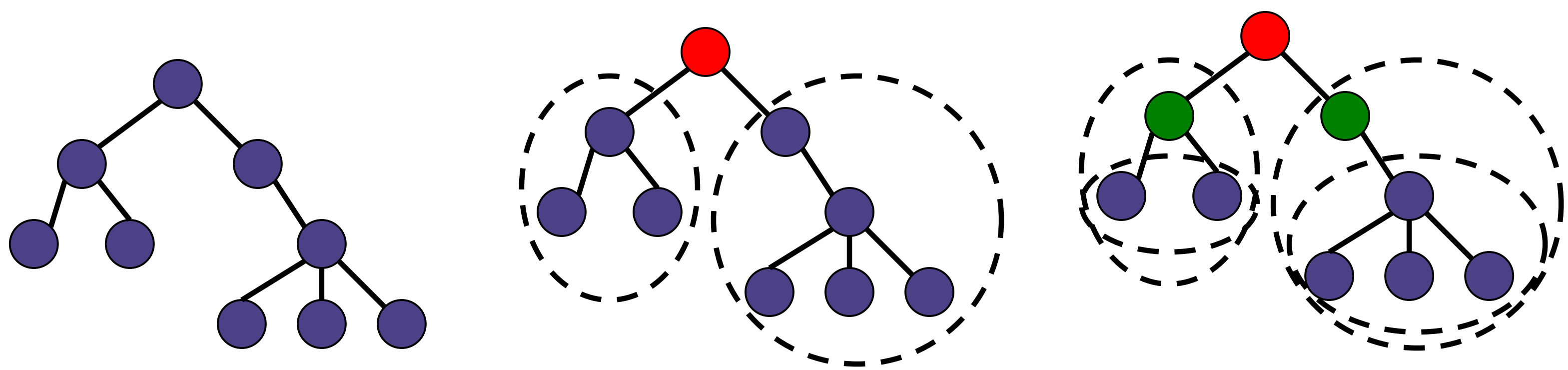


- Recursive attack
- Don't repeat calculations

$$\boxed{\begin{aligned}&\min J[i,j] = \\ &\quad \min\Big(\min J^k[i,k] \\ &\qquad\qquad + \min J_k[k,j]\Big)\end{aligned}}$$

# Linear = Belief Propagation

- Key magic: Dynamic Programming

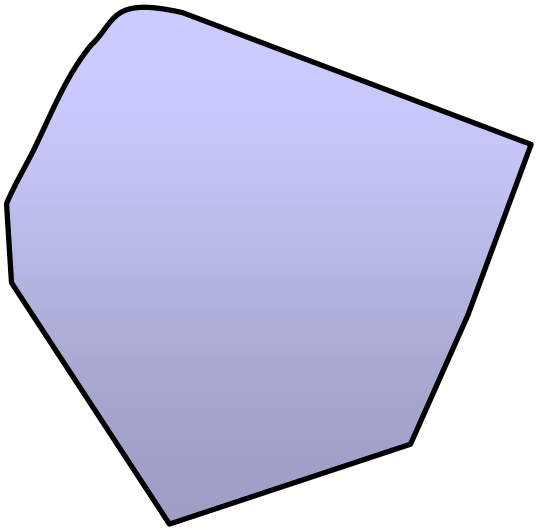


- Recursive attack
- Don't repeat calculations
- **Linear time in network size**

$$\min J[i,j] = \min\Big(\min J^k[i,k] + \min J_k[k,j]\Big)$$

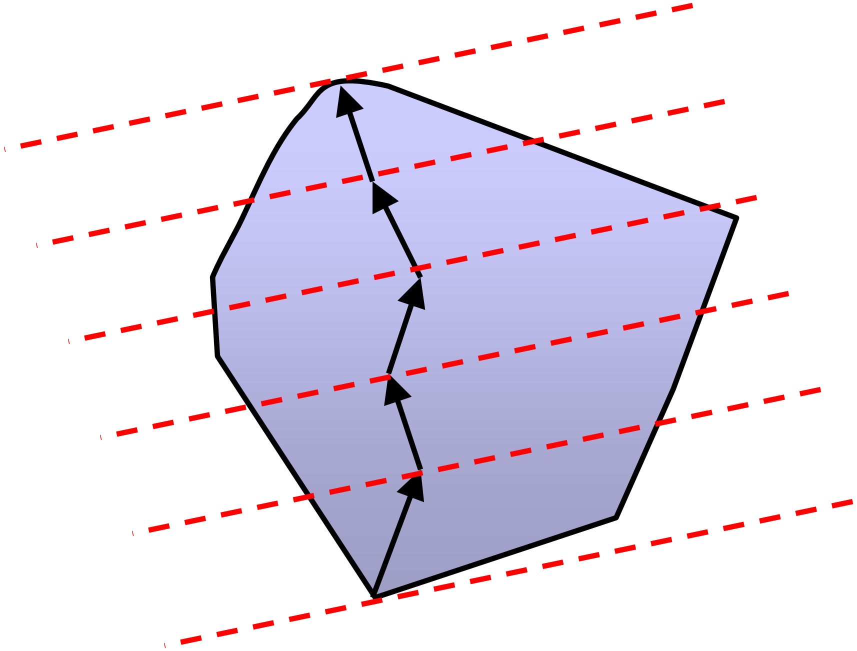
# Semidefinite Programs

- More complicated constraints



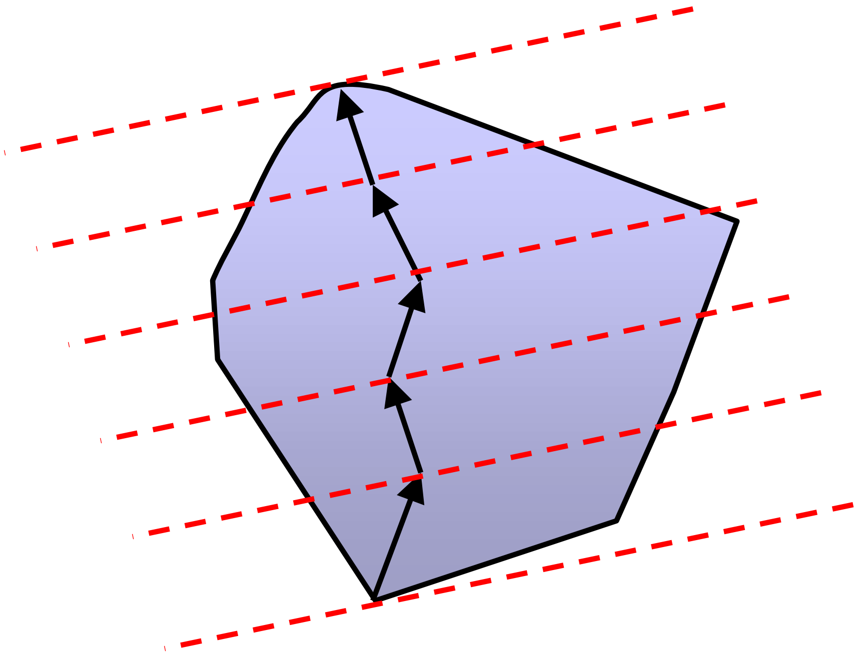
# Semidefinite Programs

- More complicated constraints
- Solve with Interior Point Methods



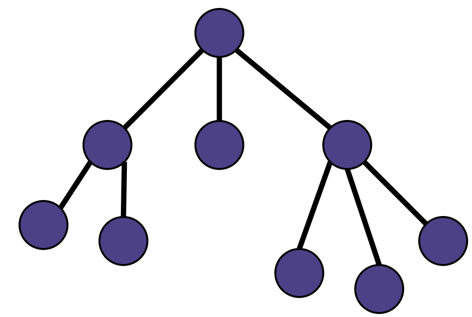
# Semidefinite Programs

- More complicated constraints
- Solve with Interior Point Methods
- Can be **very** slow
- Need **global info**



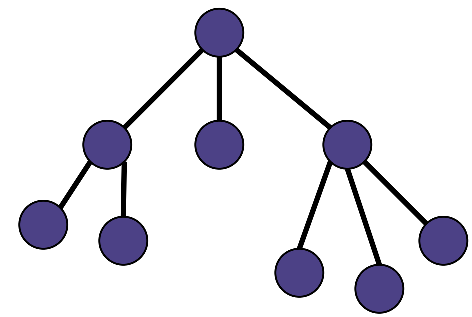
# Dynamic Programming for Semidefinite Constraints

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{B}12 & \mathbf{B}_{13} & \mathbf{B}_{14} \\ \mathbf{B}_{12}^\top & \mathbf{A}_2 & 0 & 0 \\ \mathbf{B}_{13}^\top & 0 & \mathbf{A}_3 & 0 \\ \mathbf{B}_{14}^\top & 0 & 0 & \mathbf{A}_4 \end{bmatrix} \succeq 0$$

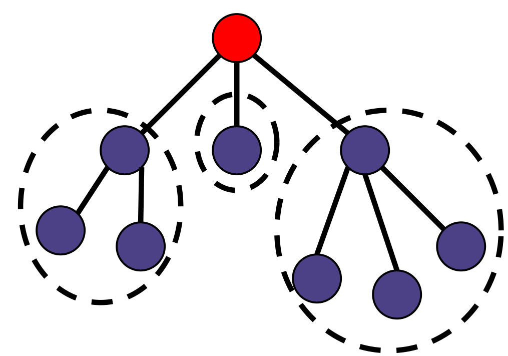


# Dynamic Programming for Semidefinite Constraints

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{B}12 & \mathbf{B}_{13} & \mathbf{B}_{14} \\ \mathbf{B}_{12}^\top & \mathbf{A}_2 & 0 & 0 \\ \mathbf{B}_{13}^\top & 0 & \mathbf{A}_3 & 0 \\ \mathbf{B}_{14}^\top & 0 & 0 & \mathbf{A}_4 \end{bmatrix} \succeq 0$$



$$\Updownarrow$$



$$\mathbf{A}_2 \succeq 0,\ \mathbf{A}_3 \succeq 0\ \mathbf{A}_4 \succeq 0$$
$$\mathbf{A}_1 - \textstyle\sum_{j=2}^{4} \mathbf{B}_{1j}^\top \mathbf{A}_j^{-1} \mathbf{B}_{1j} \succeq 0$$

# Tree based approximations

- Use tree-parameterization or "loopy" approximations
- Each successive relaxation groups nodes
- Revisit treewidth to solve problems exactly [Bodlander 87]