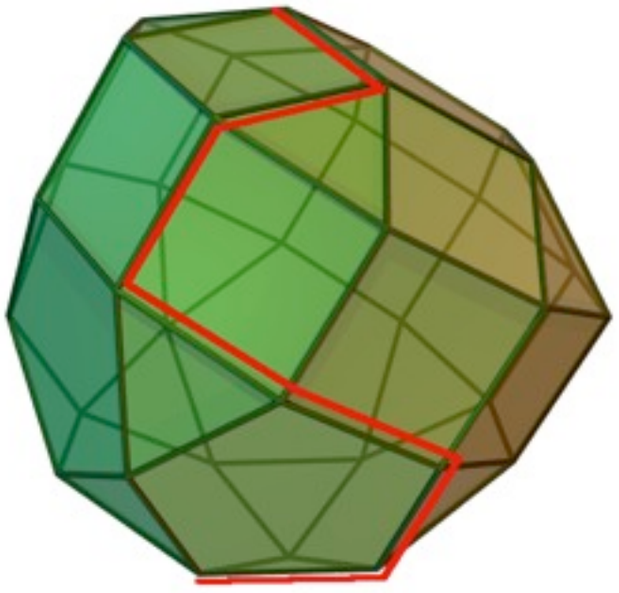
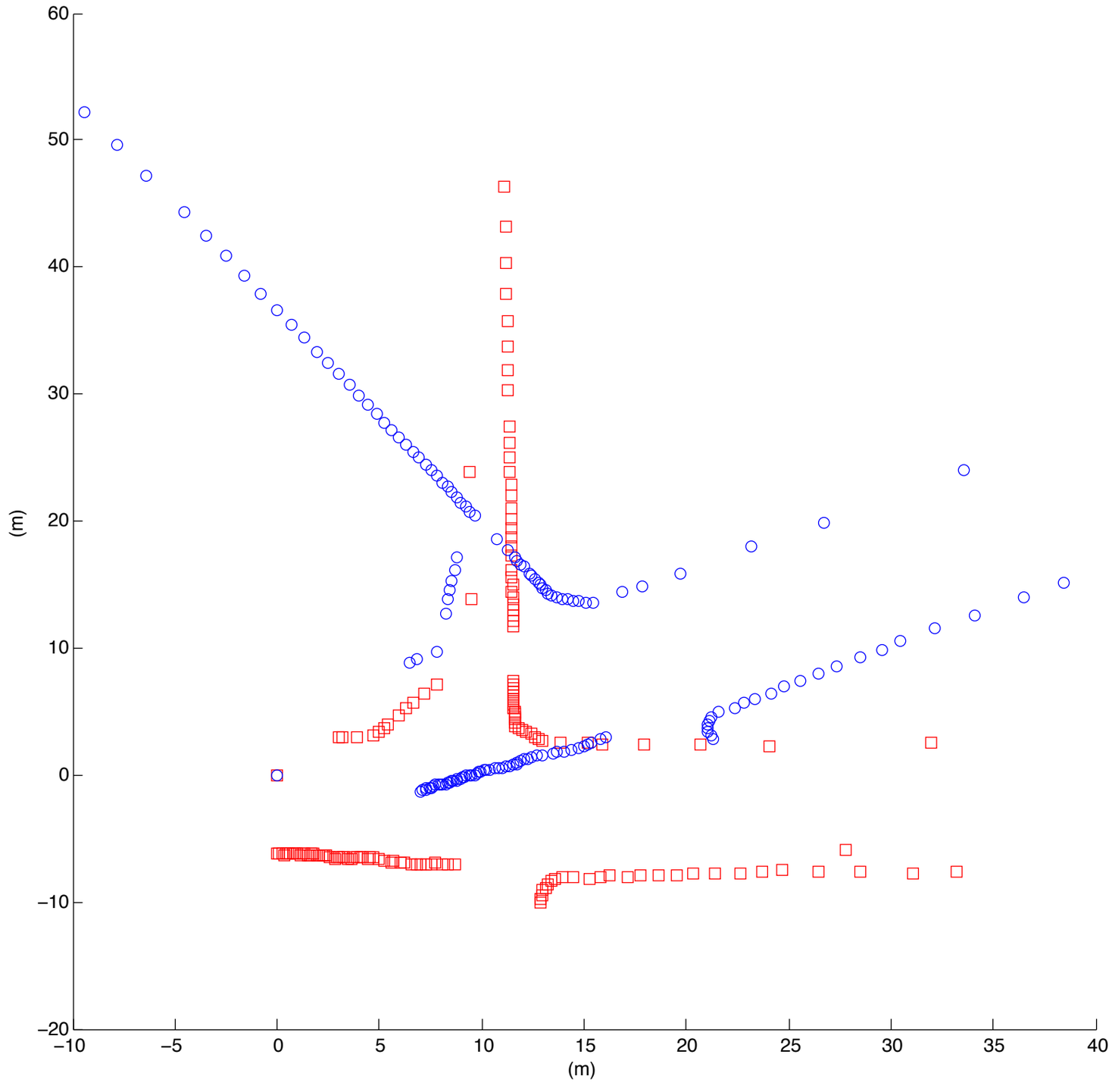


Linear Programming Decompositions for Distributed, Anytime MAP Inference

Joop van de Ven, Fabio Ramos

ACFR
The University of Sydney
October, 2010

Scan Matching



Scan Matching

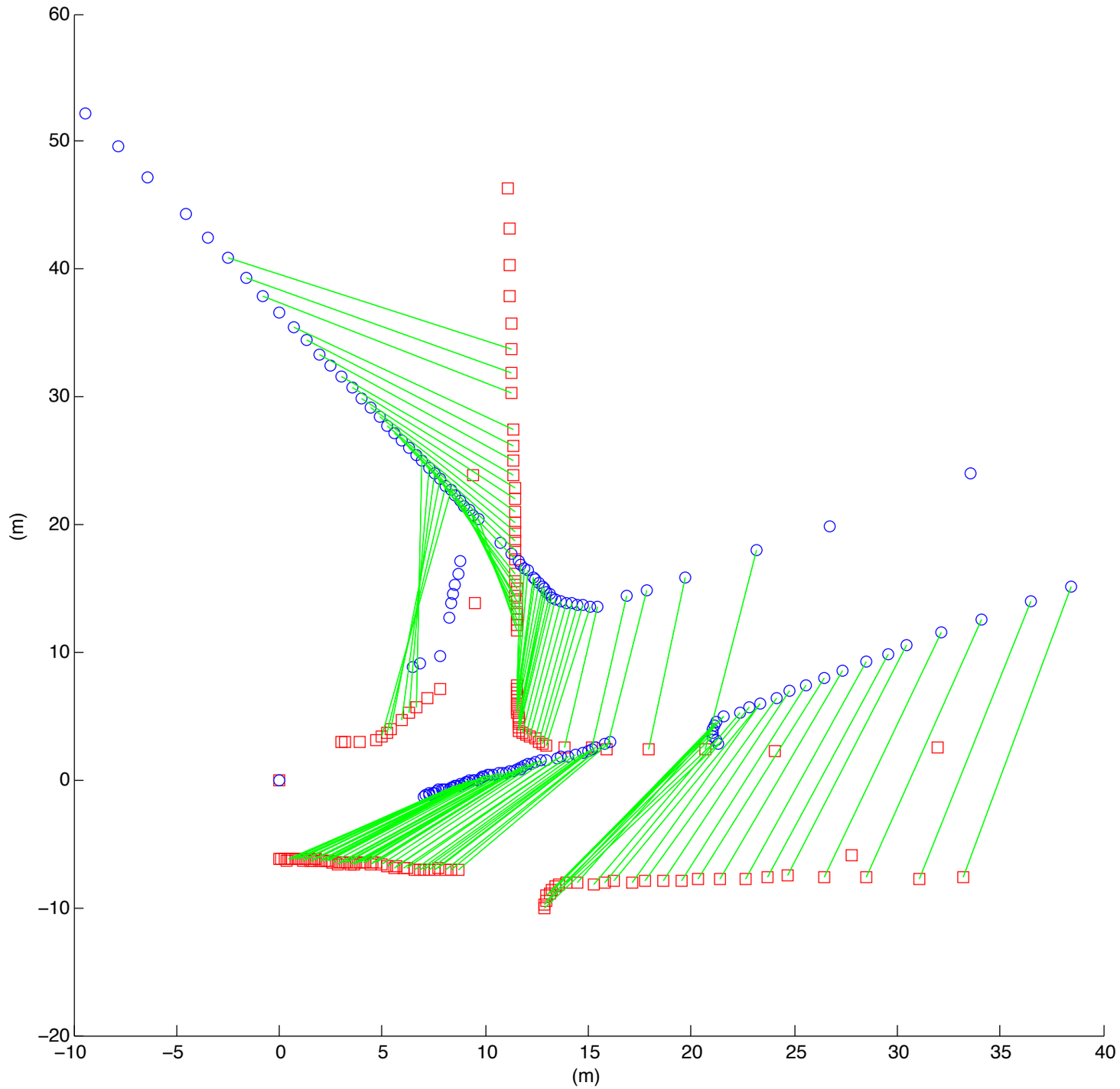


Image Feature Matching

Image Feature Matching

Image Feature Matching

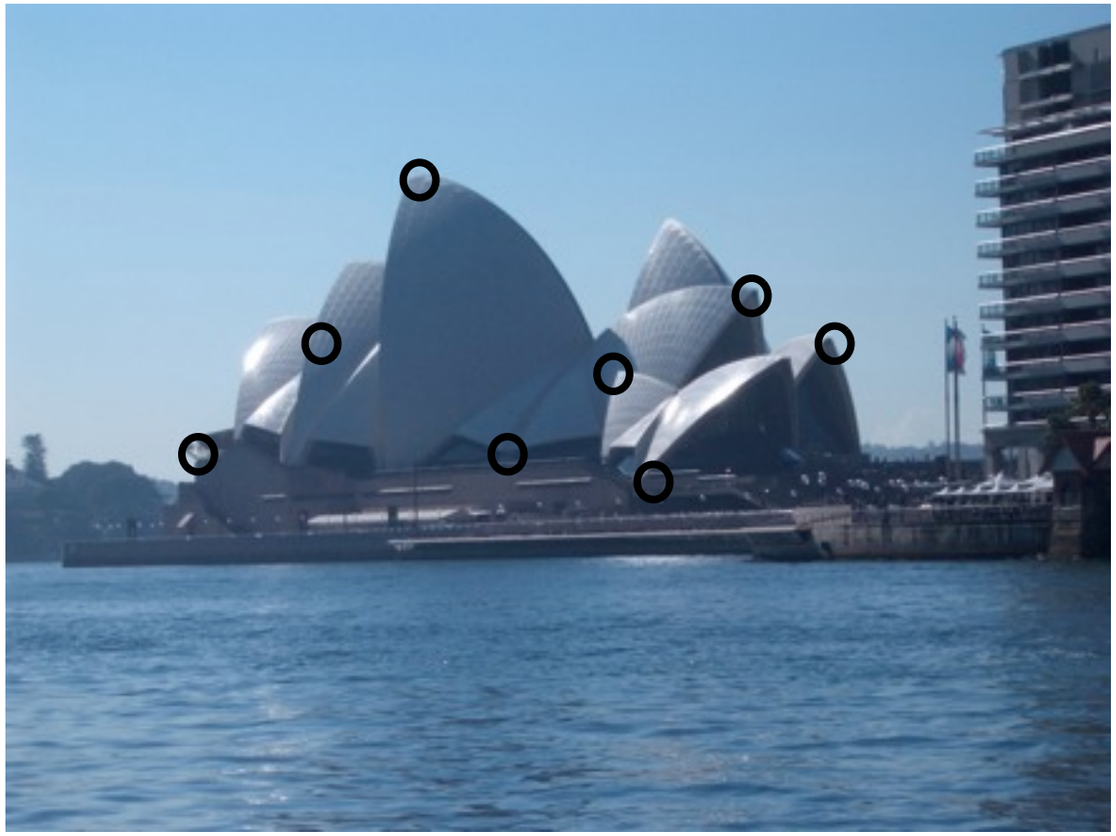


Image Feature Matching

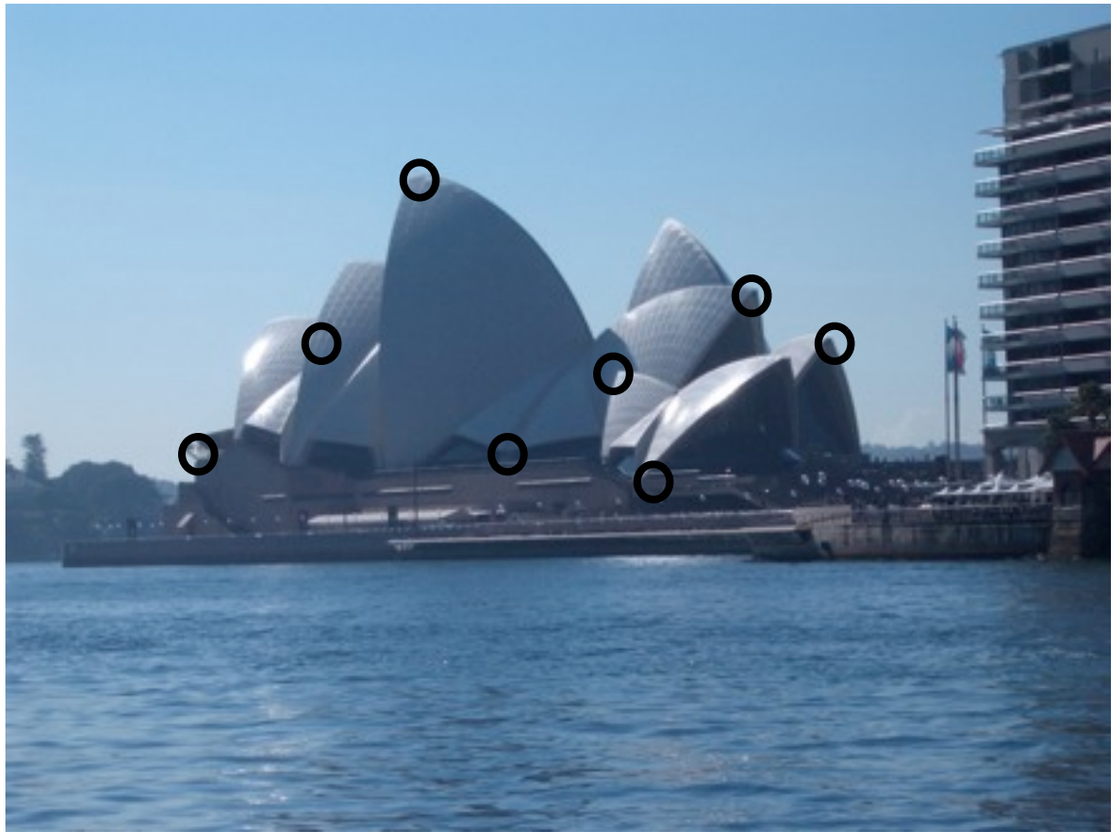
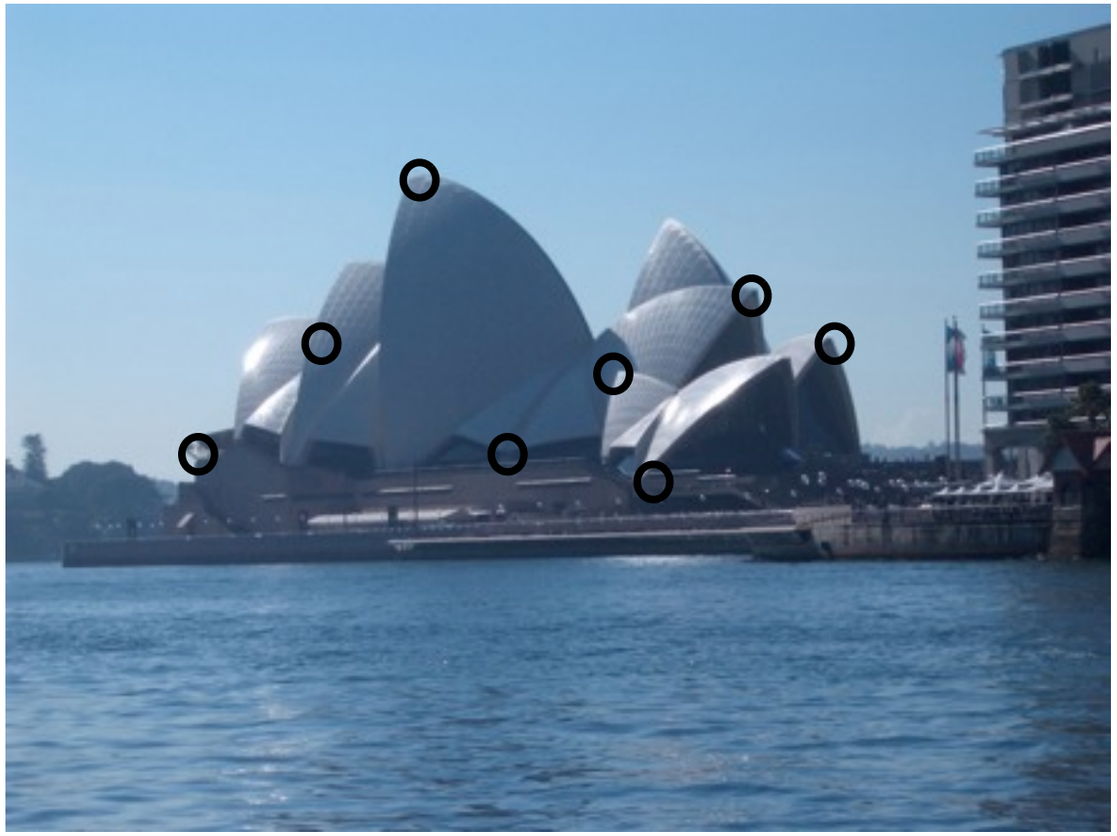


Image Feature Matching



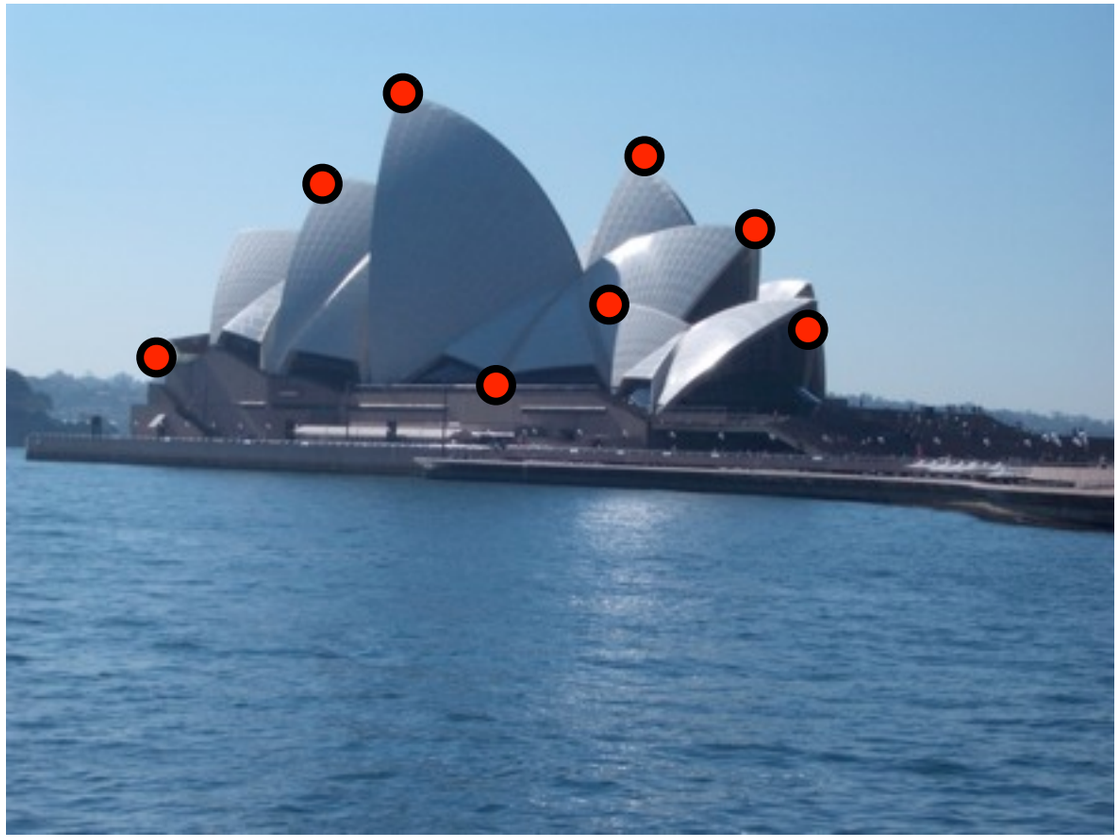
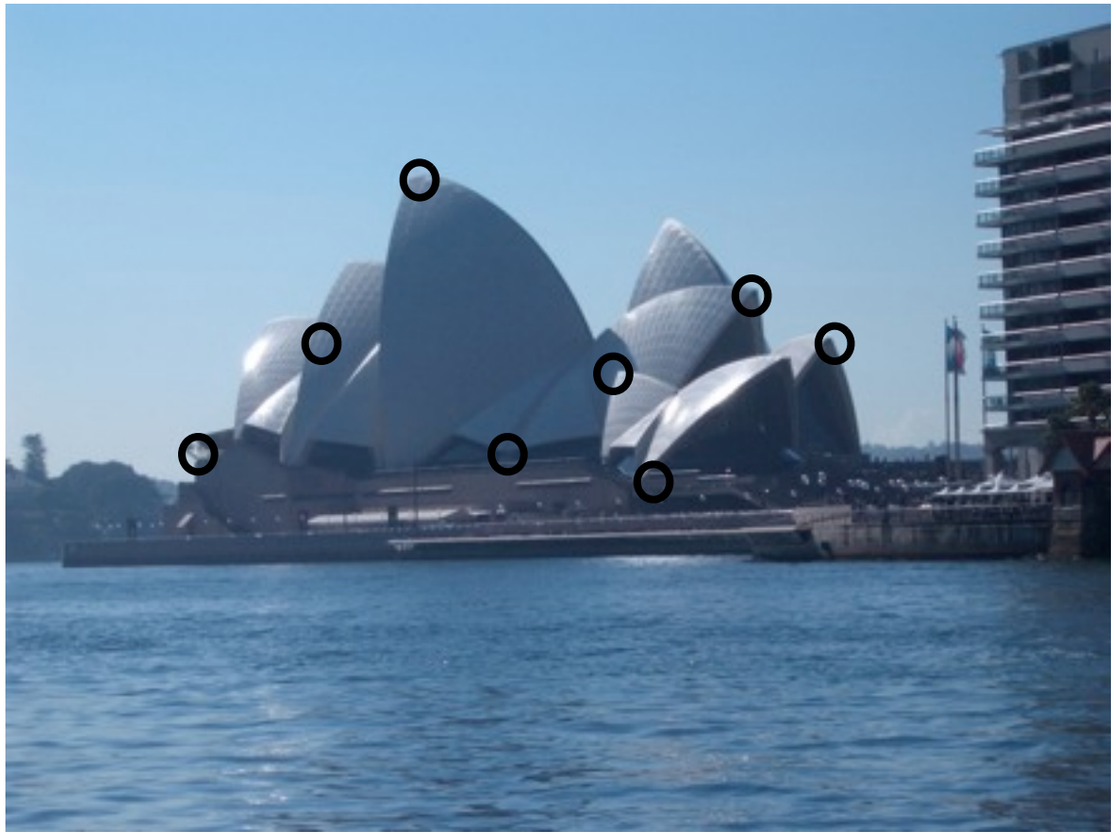


Image Feature Matching



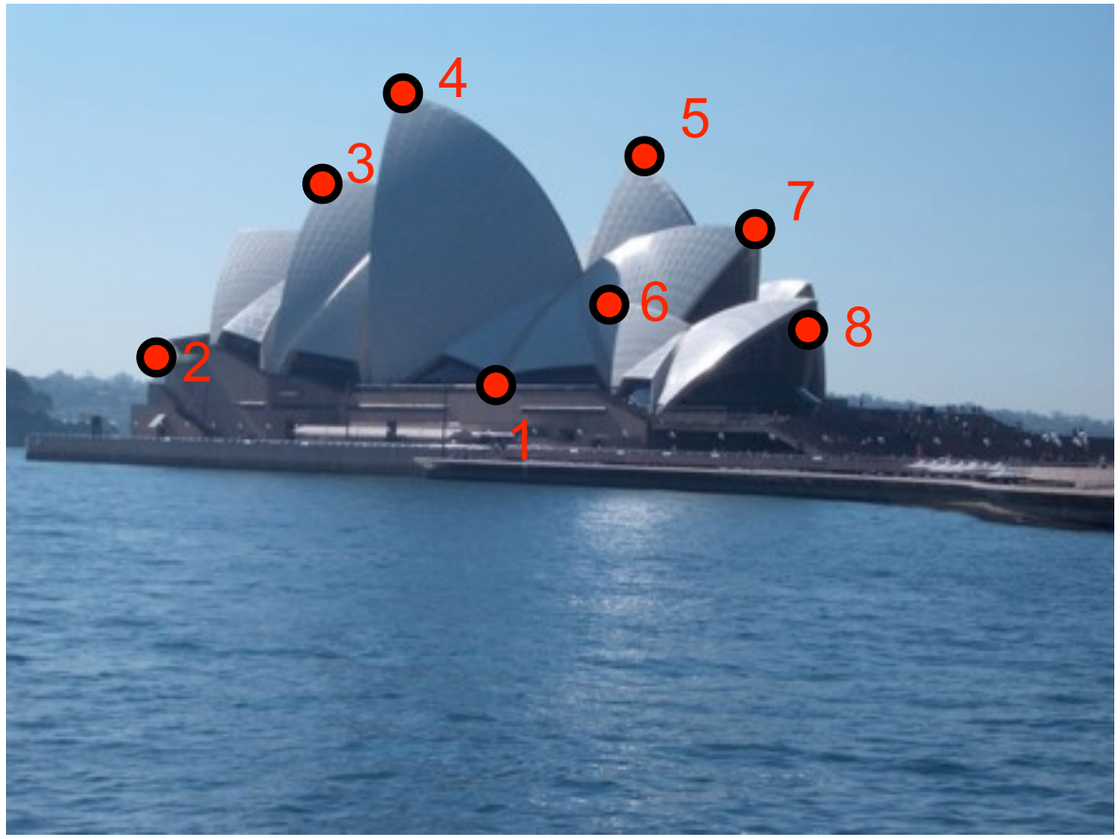
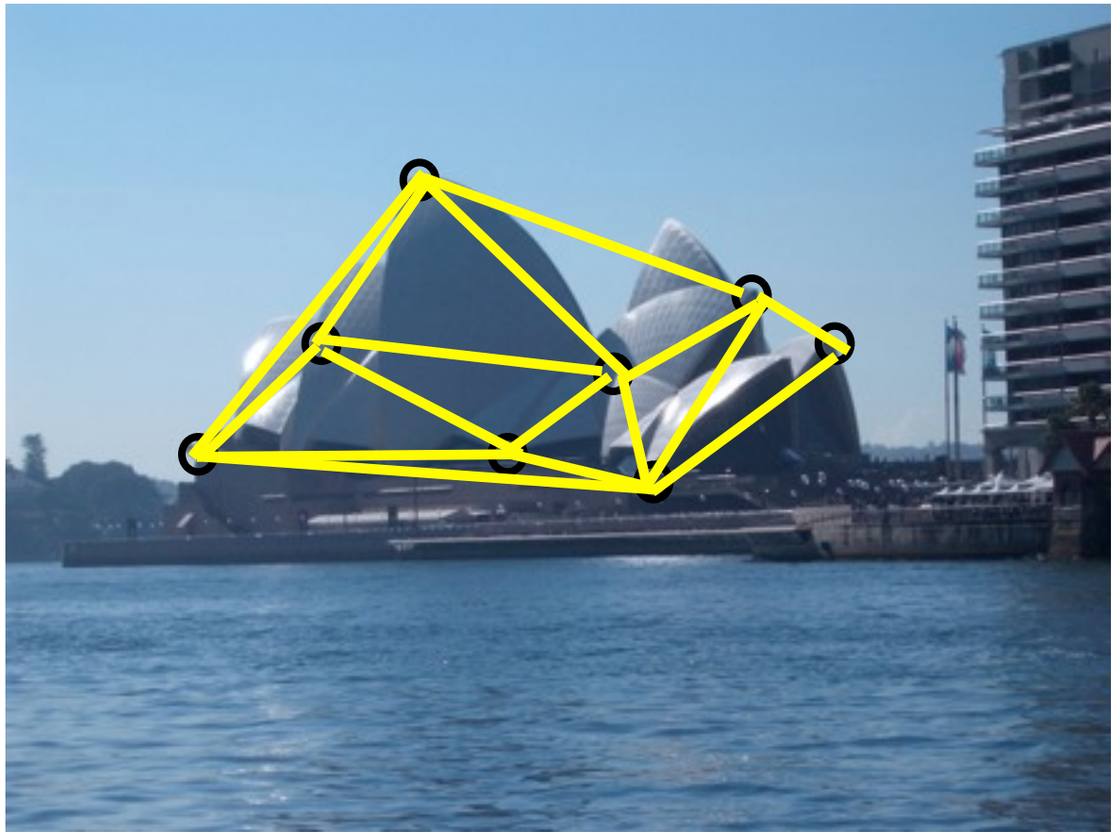


Image Feature Matching



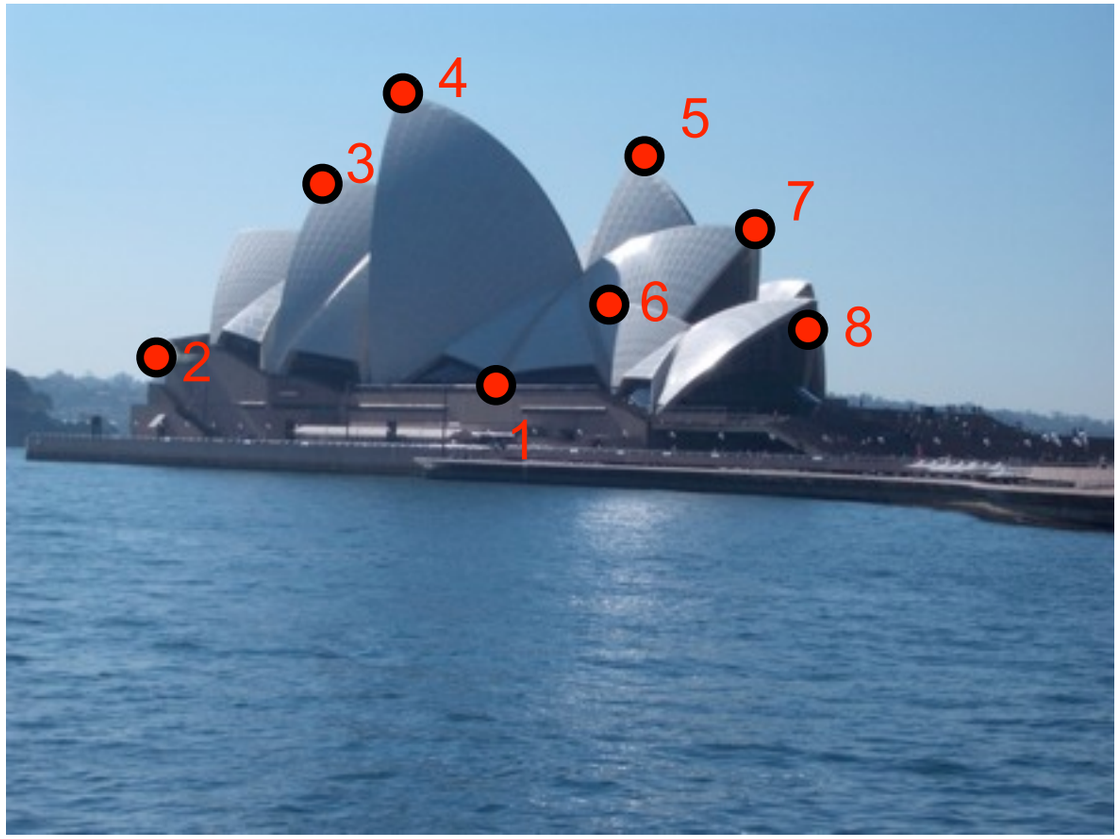


Image Feature Matching

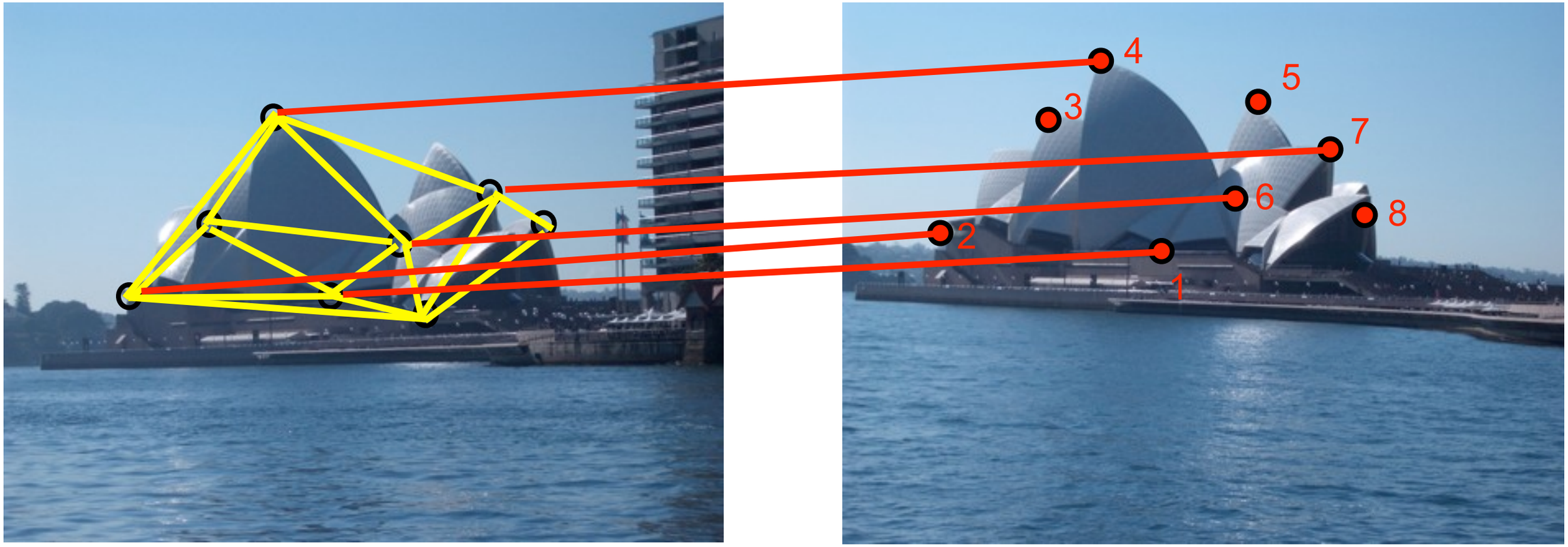
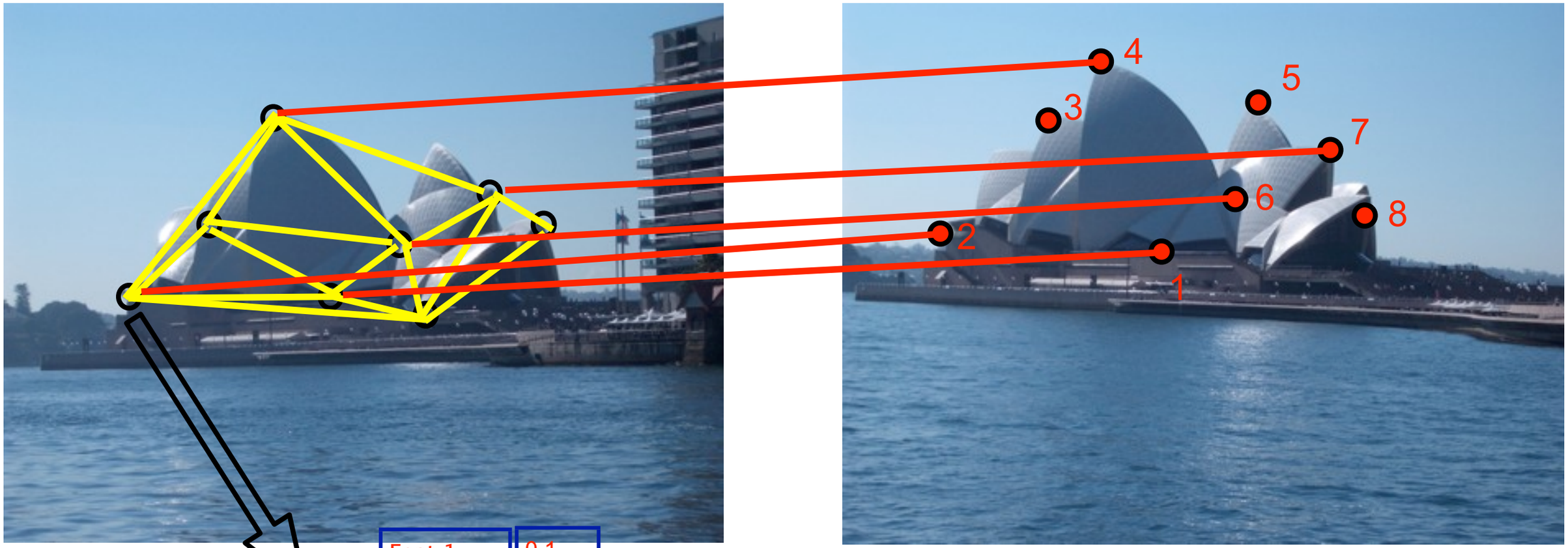


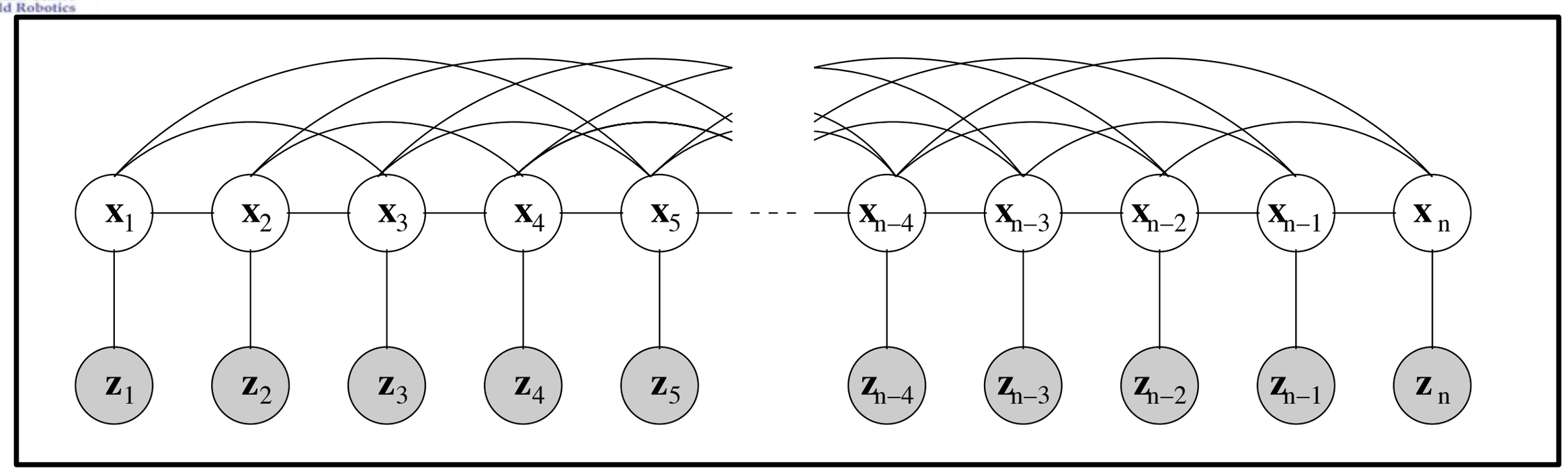
Image Feature Matching



x_1 =

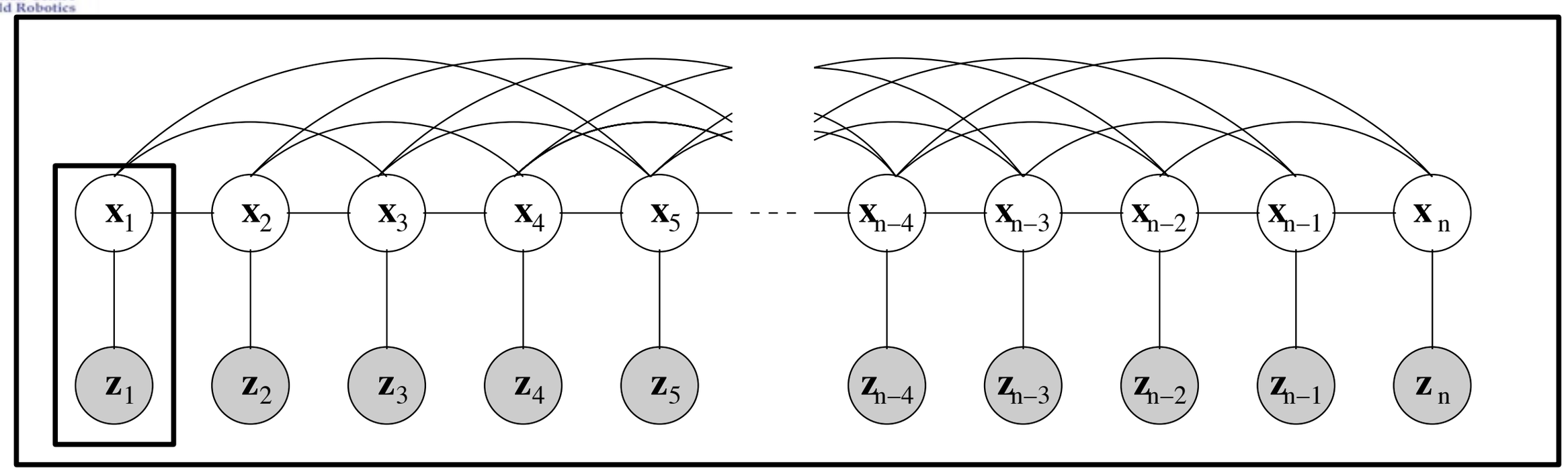
Feat 1	0.1
Feat 2	0.7
Feat 3	0.01
Feat 4	0.01
Feat 5	0.01
Feat 6	0.01
Feat 7	0.01
Feat 8	0.01
Outlier	0.14

CRF-Matching Graphical Model



Each random variable x_i has categorical values representing the association of a laser point in scan A with a laser point in scan B

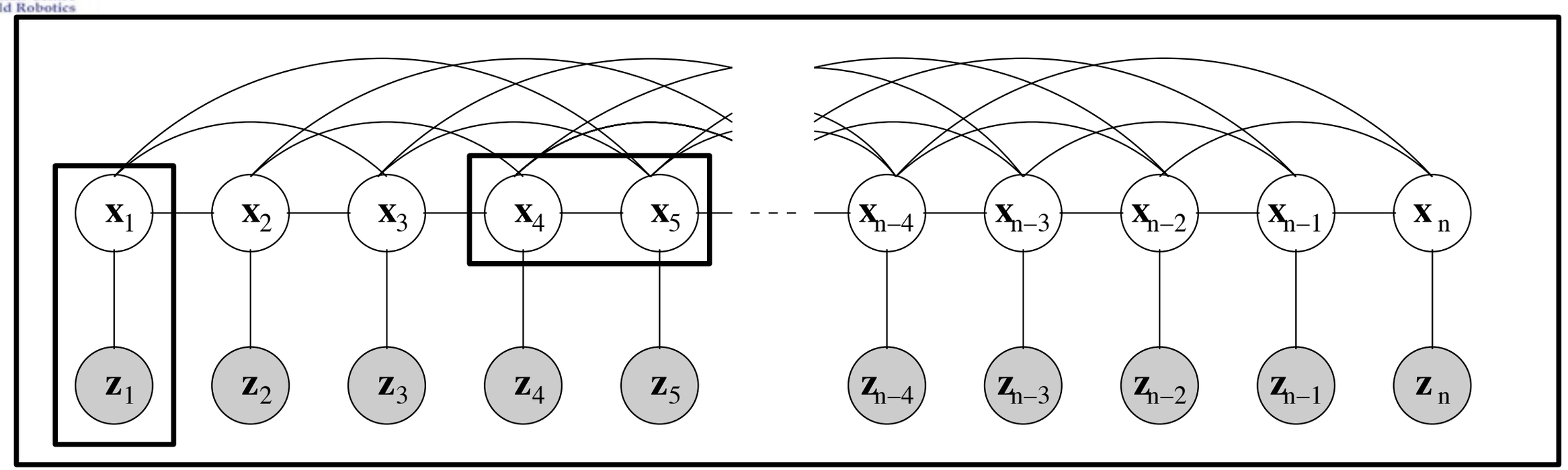
CRF-Matching Graphical Model



Local feature

Each random variable $\mathbf{x}_i$ has categorical values representing the association of a laser point in scan A with a laser point in scan B

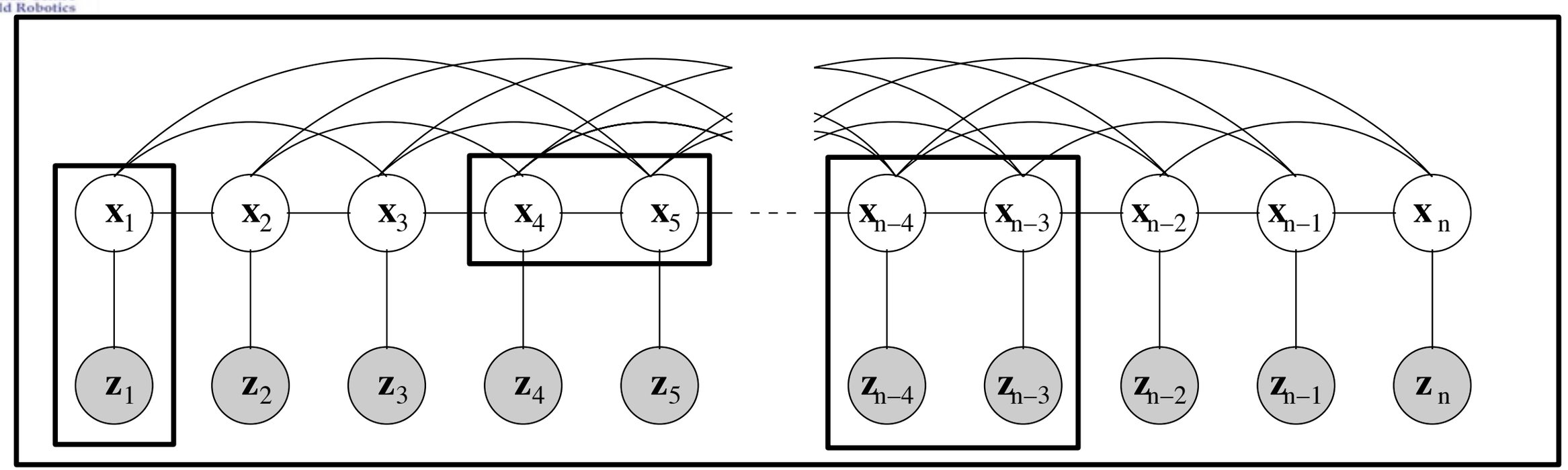
CRF-Matching Graphical Model



Local feature Pairwise feature

Each random variable $\mathbf{x}_i$ has categorical values representing the association of a laser point in scan A with a laser point in scan B

CRF-Matching Graphical Model



Local feature Pairwise feature General feature

Each random variable $\mathbf{x}_i$ has categorical values representing the association of a laser point in scan A with a laser point in scan B

Conditional Random Fields (CRFs)

- CRFs are undirected graphical models developed for labelling sequence data.
- A CRF factorises the conditional distribution into a product of clique potentials
- CRFs can be learnt in a discriminative manner using gradient descent algorithms

$$p_G(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \underbrace{\prod_{s \in V} \Psi_s(x_s|\mathbf{z})}_{\text{local potentials}} \underbrace{\prod_{(s,t) \in E} \Psi_{s,t}(x_s, x_t|\mathbf{z})}_{\text{pairwise potentials}}$$

Most probable explanation (MAP)

- Finds the most likely assignments, that maximise the posterior probability:

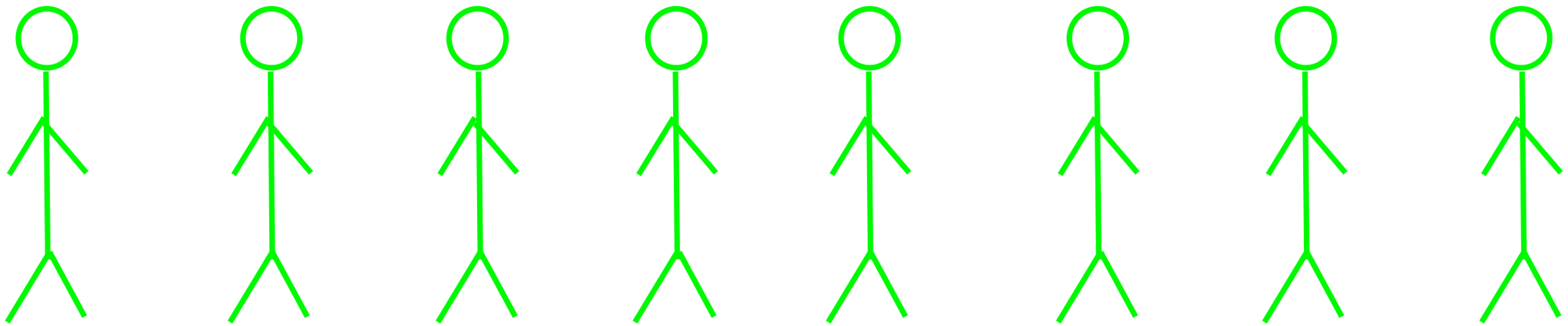
$$X^{MAP} = \underset{X}{\operatorname{argmax}} \log p_G(X;\Phi)$$

Log-linear model

$$\Psi_s(x_s|\mathbf{z}) = e^{\phi_s(x_s|\mathbf{z})} \qquad \Psi_{st}(x_s,x_t|\mathbf{z}) = e^{\phi_{st}(x_s,x_t|\mathbf{z})}$$

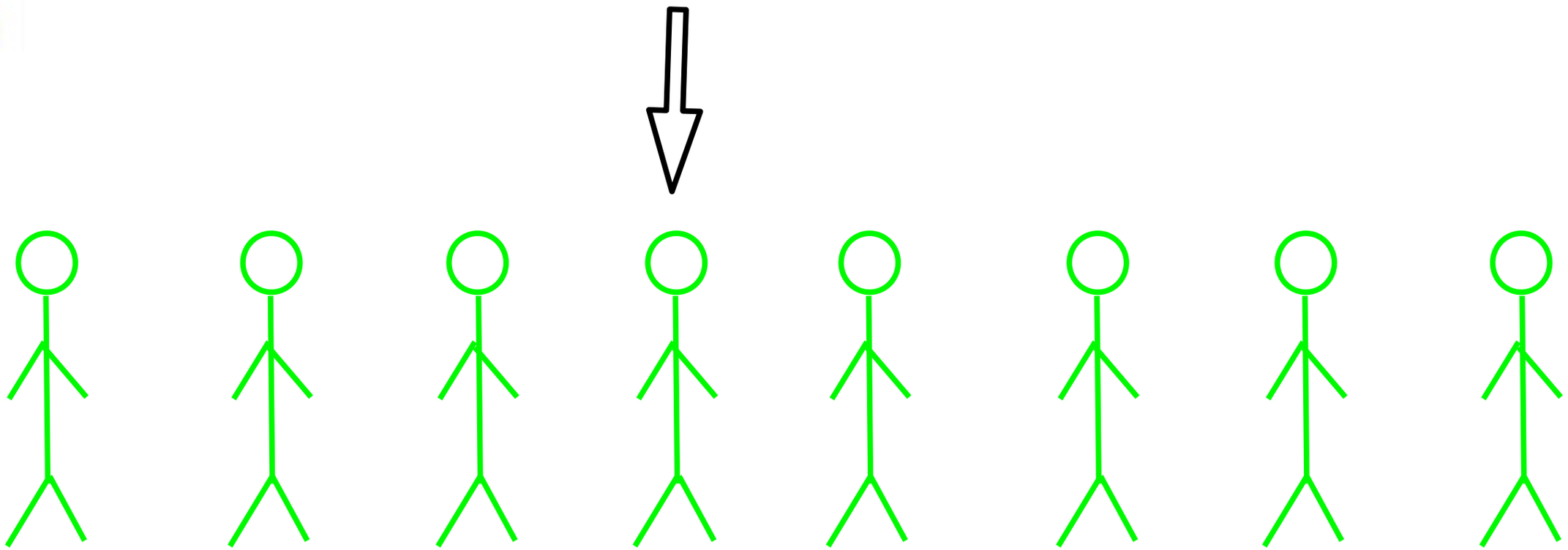
$$\log p_G(X;\Phi) = \sum_{s\in V} \phi_s(x_s) + \sum_{(s,t)\in E} \phi_{st}(x_s,x_t) - C.$$

BP Inference



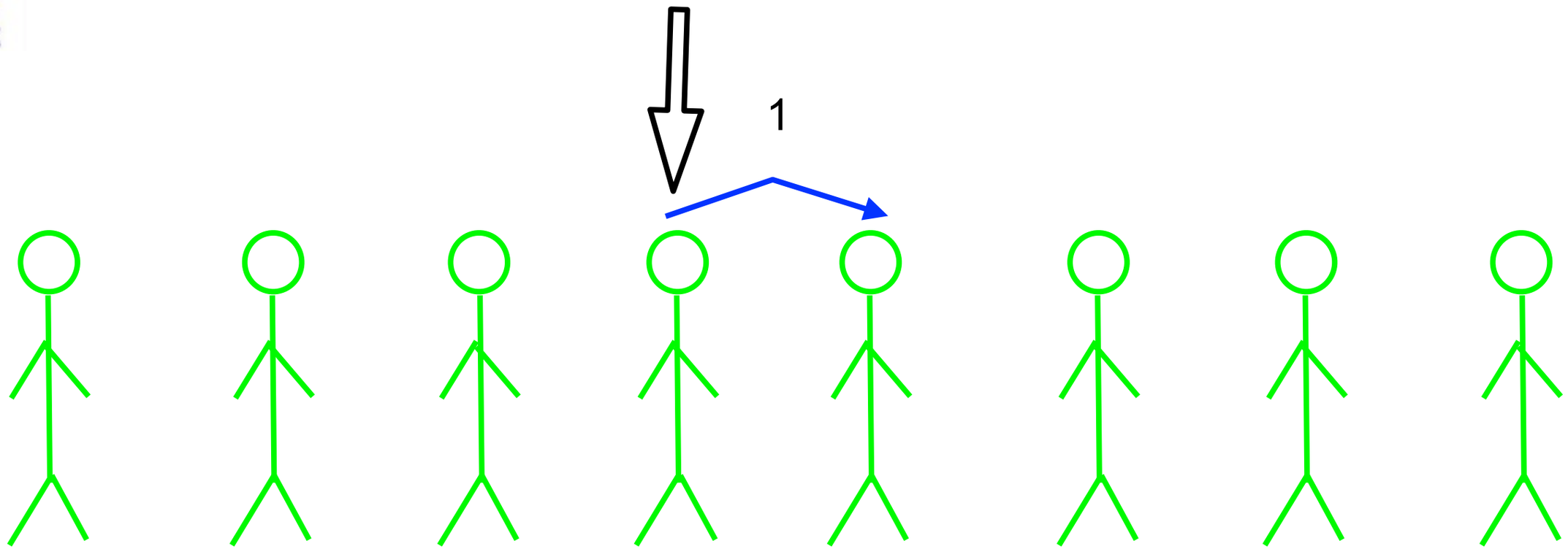
How many soldiers are in the queue?

BP Inference



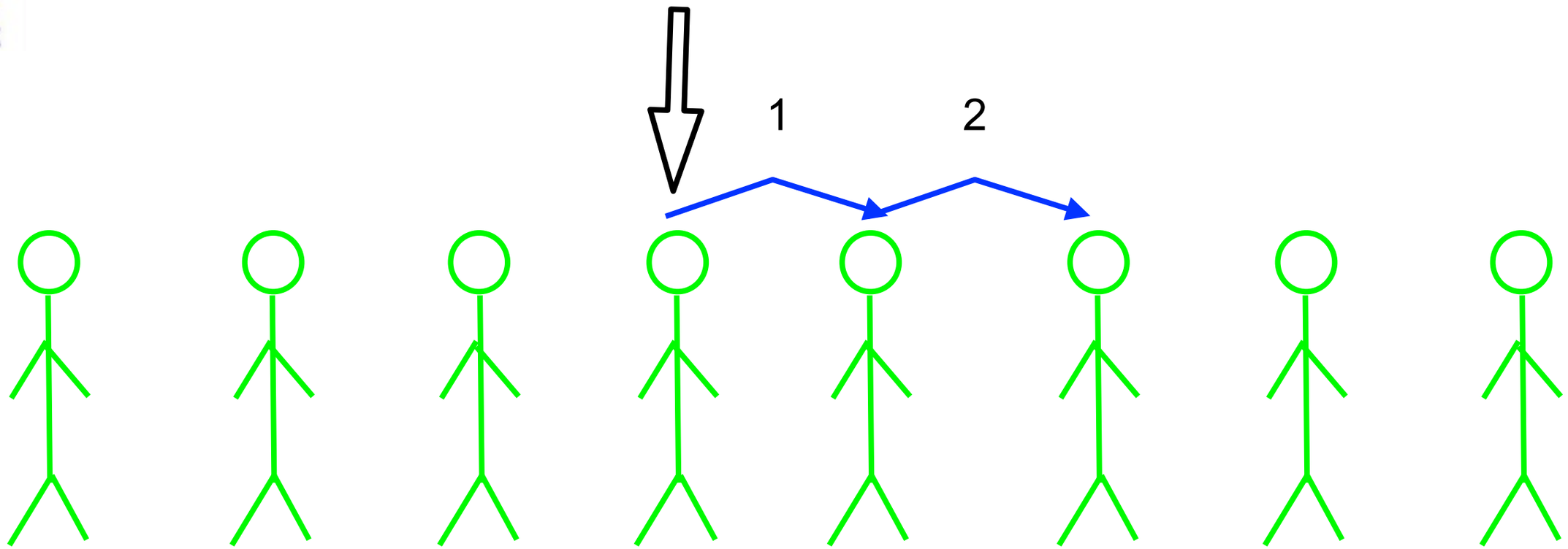
How many soldiers are in the queue?

BP Inference



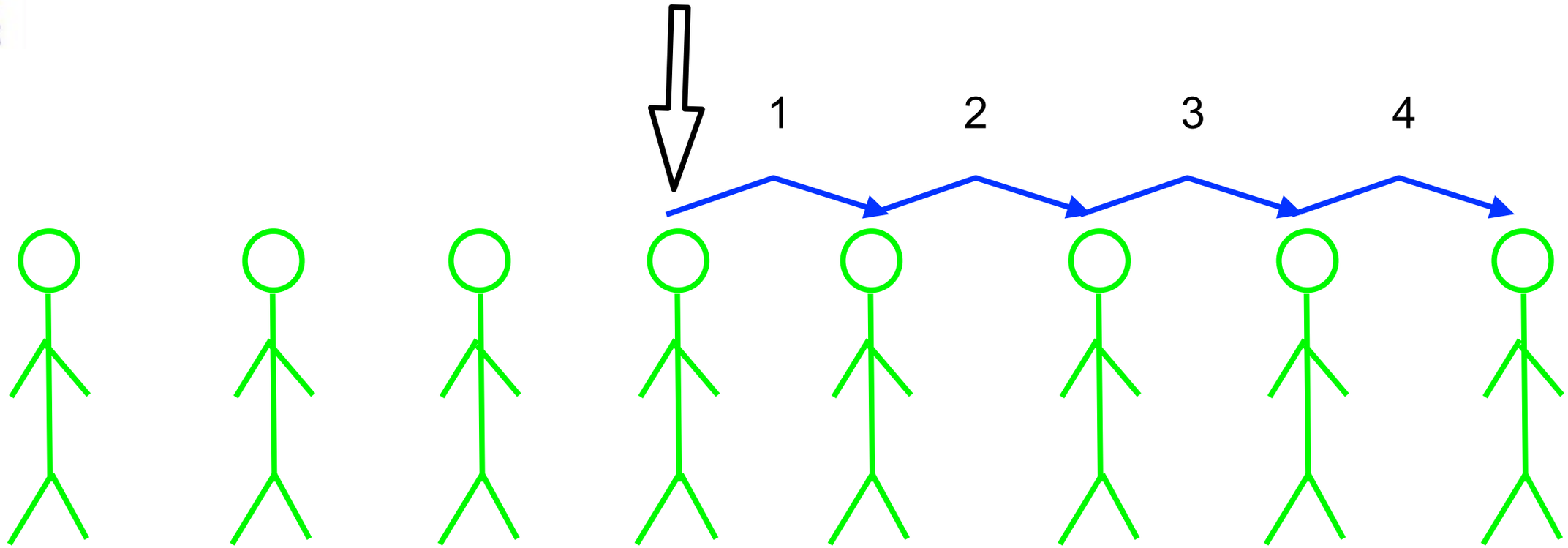
How many soldiers are in the queue?

BP Inference



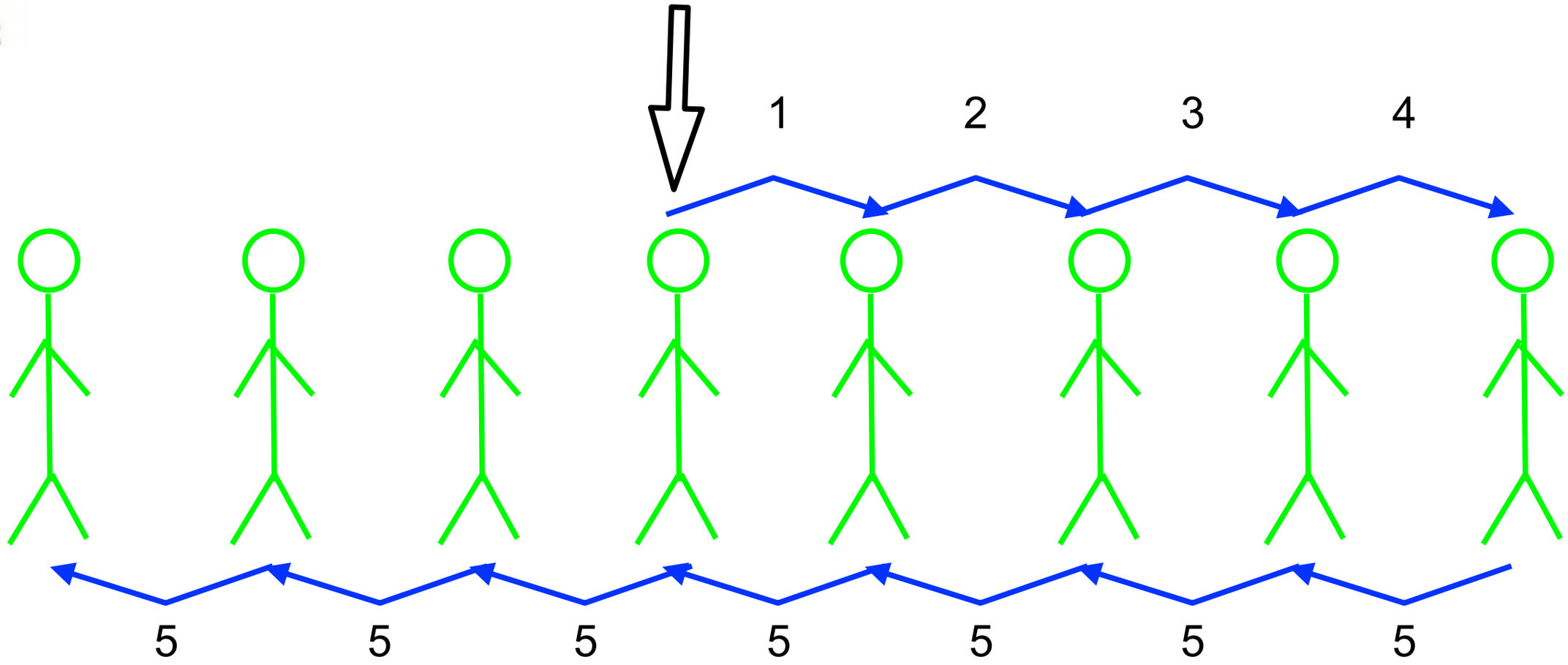
How many soldiers are in the queue?

BP Inference



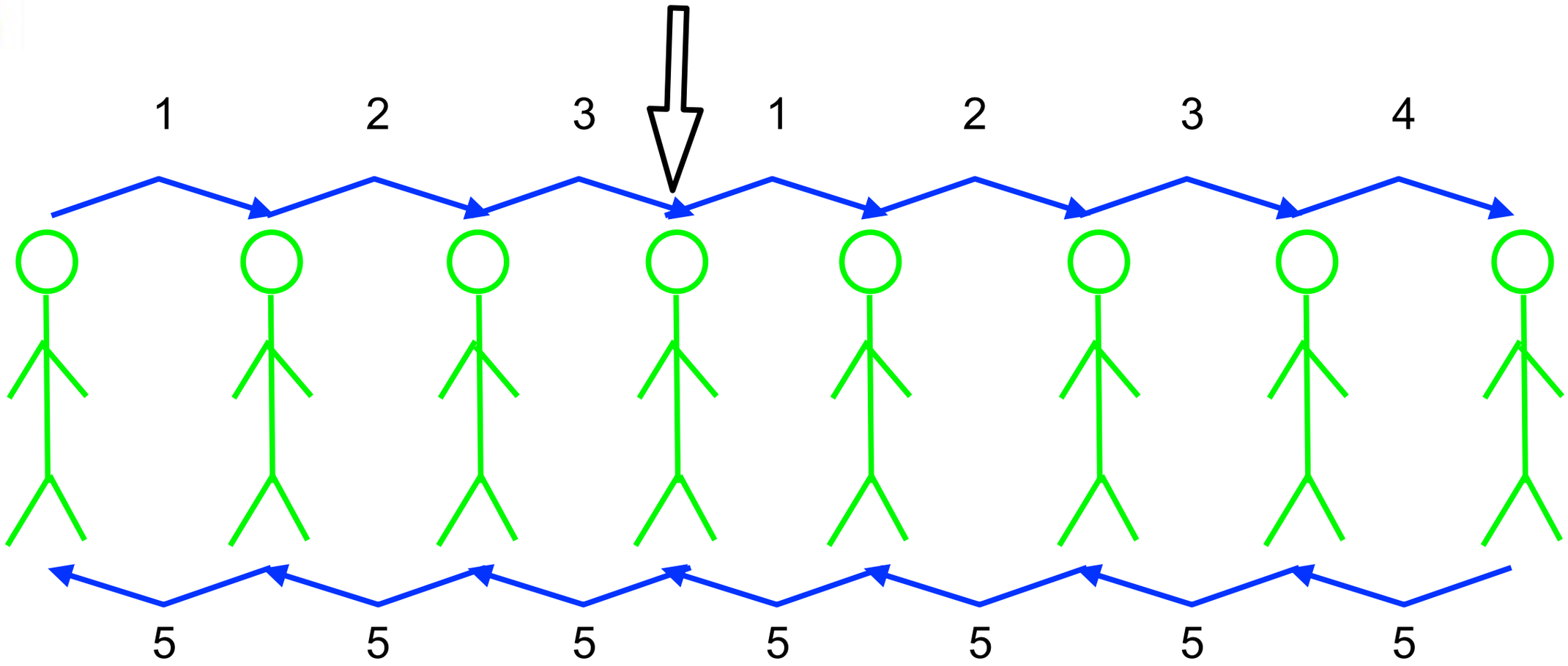
How many soldiers are in the queue?

BP Inference



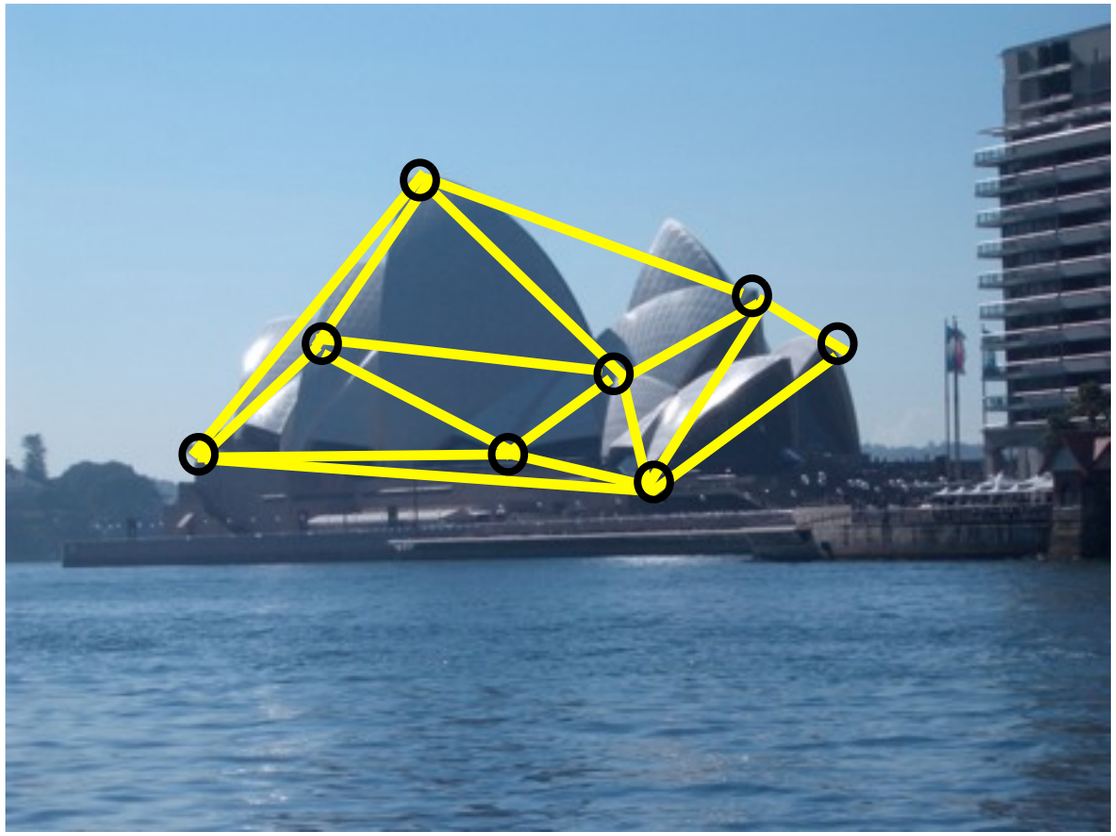
How many soldiers are in the queue?

BP Inference

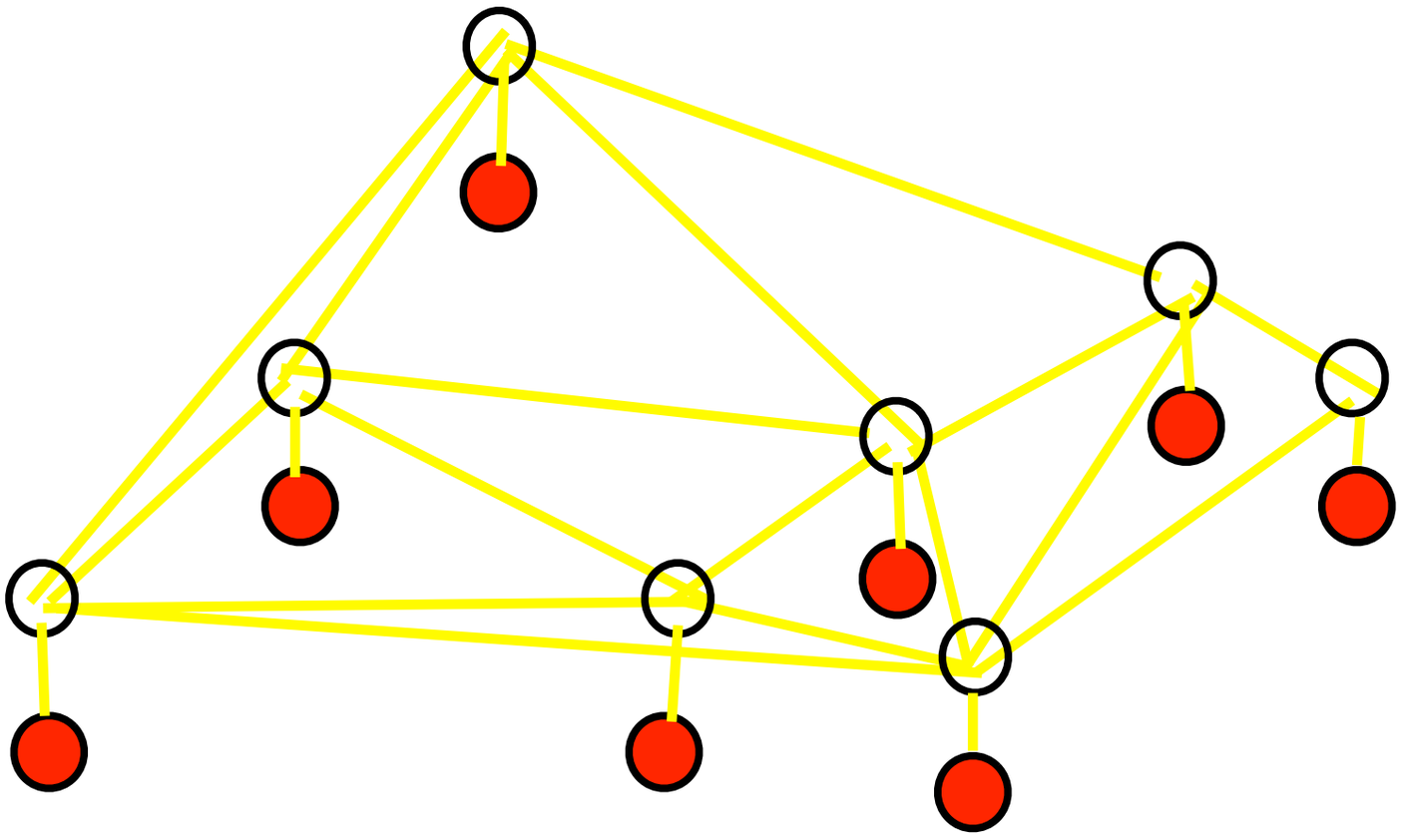


How many soldiers are in the queue?

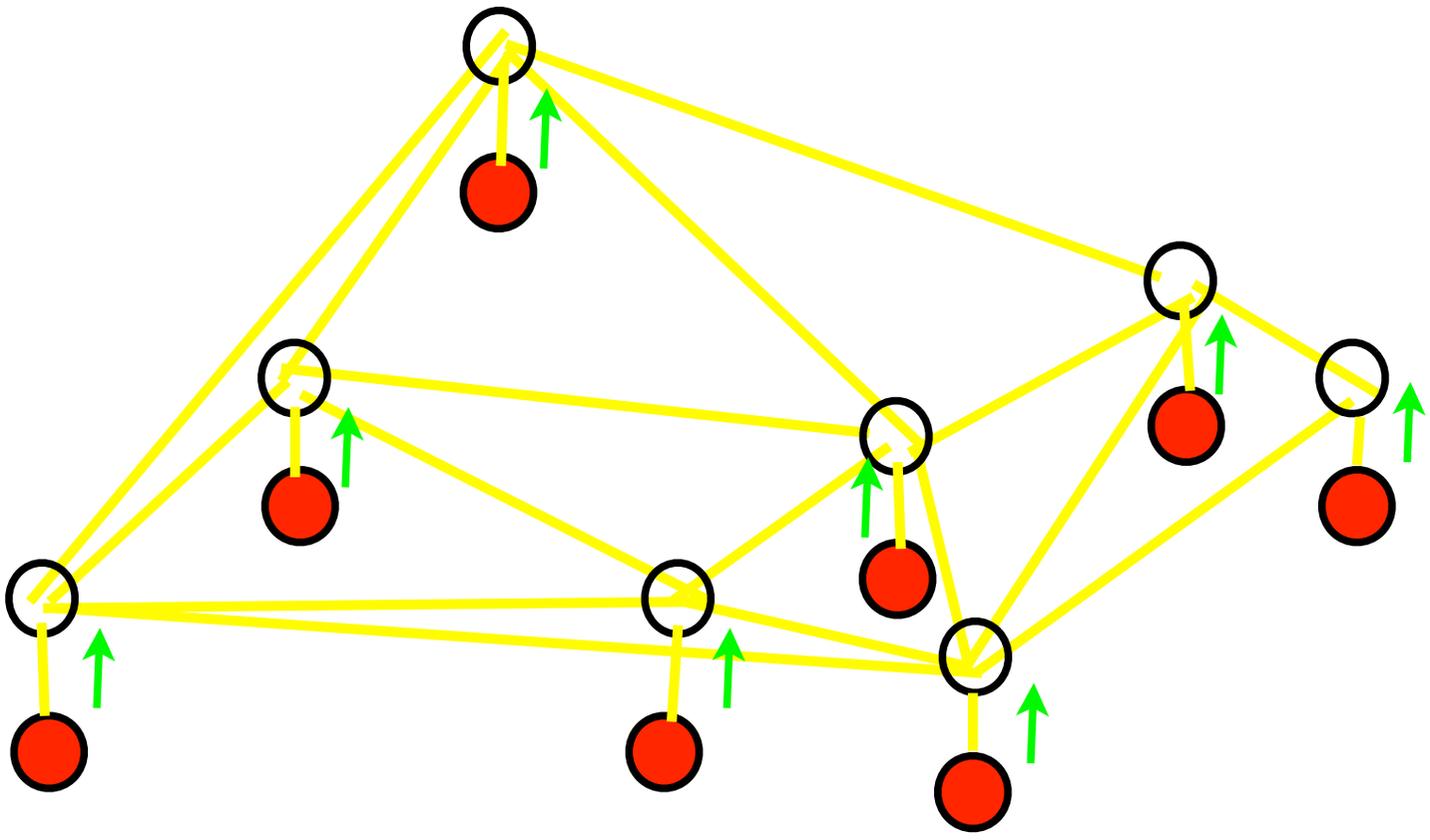
Loopy Belief Propagation



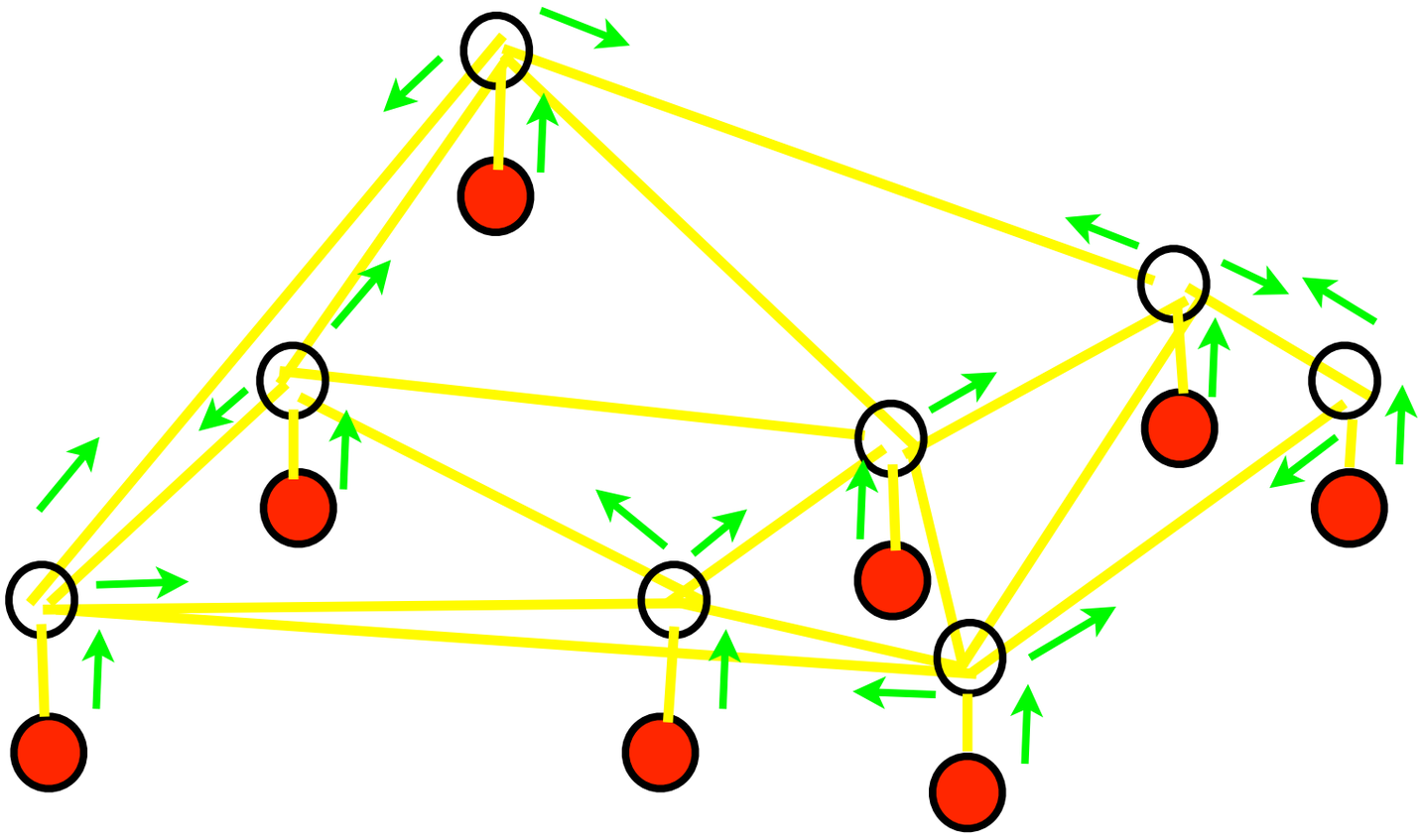
Loopy Belief Propagation



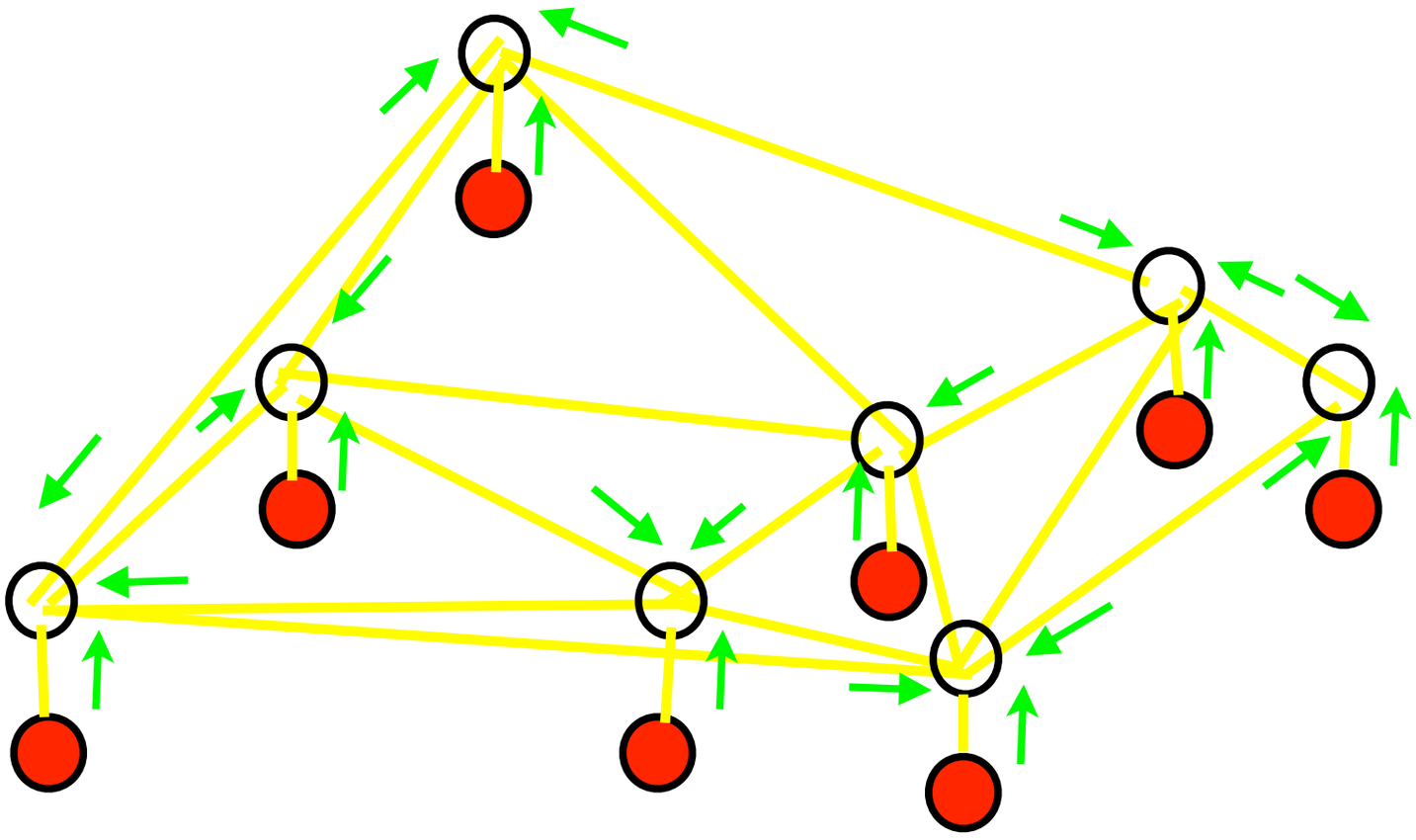
Loopy Belief Propagation



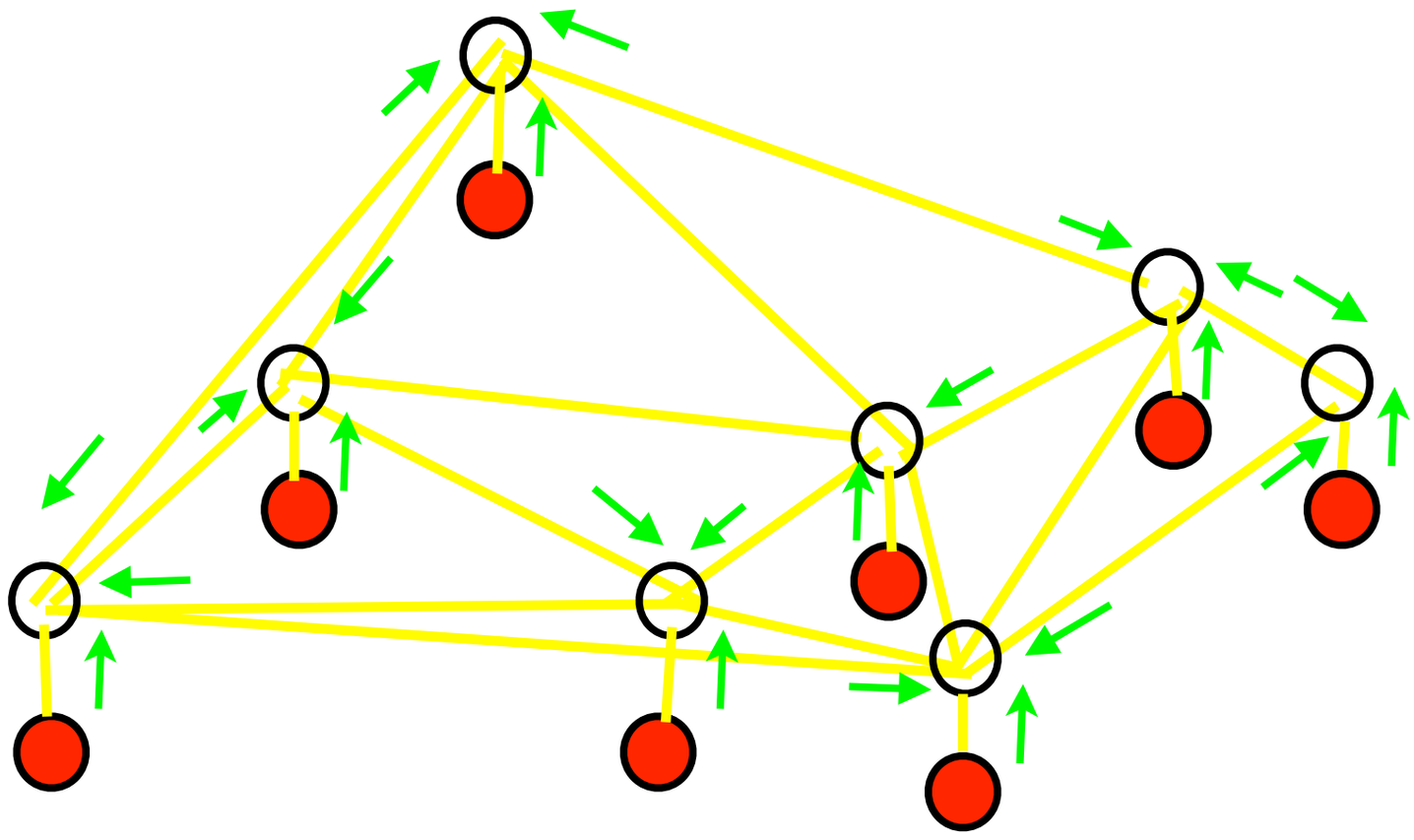
Loopy Belief Propagation



Loopy Belief Propagation

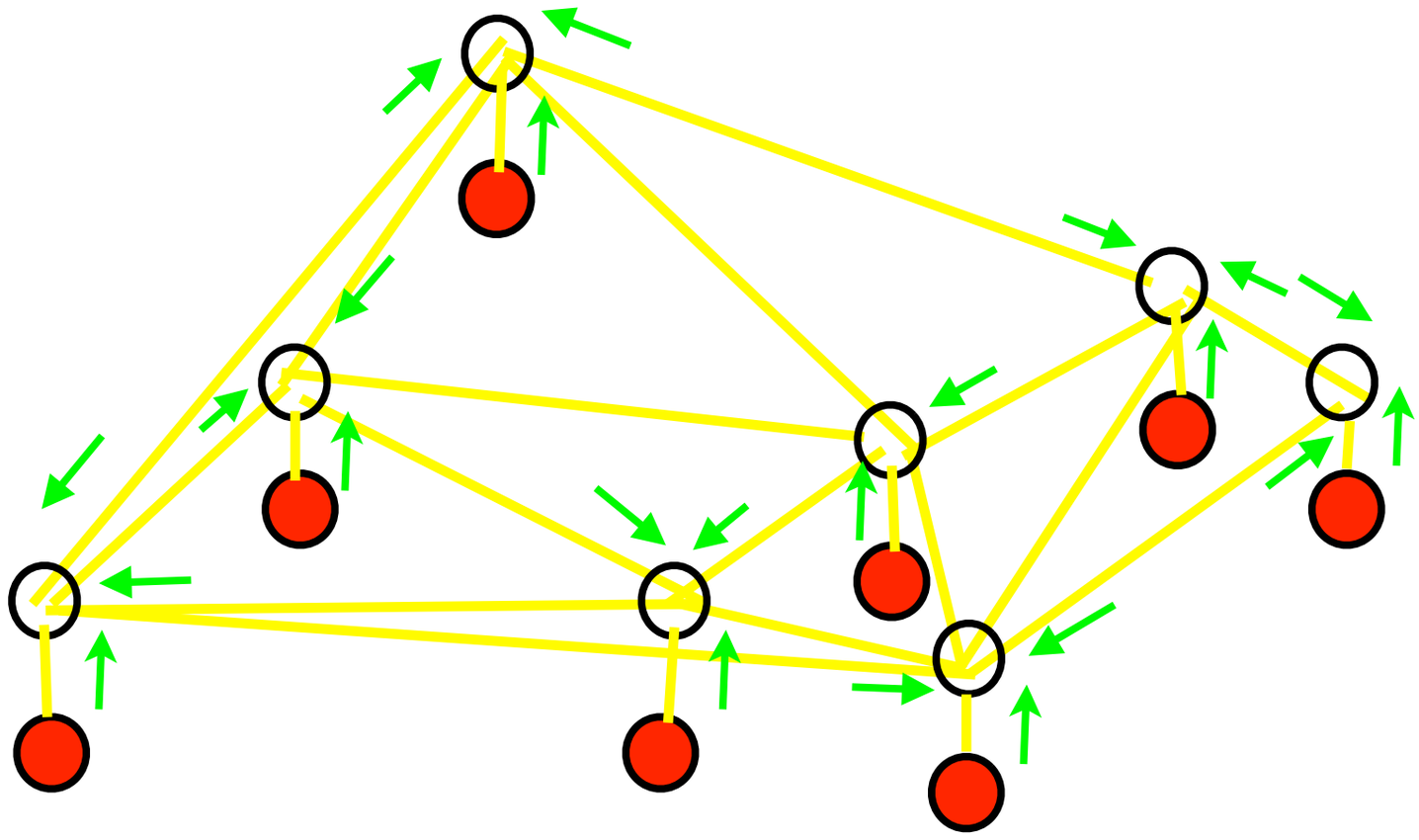


Loopy Belief Propagation



$$m_{st}(x_t) \leftarrow \alpha_{st} \max \phi_s(x_s)\phi_{st}(x_s, x_t) \prod_{u \in \mathcal{N}_s \setminus t} m_{us}(x_s)$$

Loopy Belief Propagation

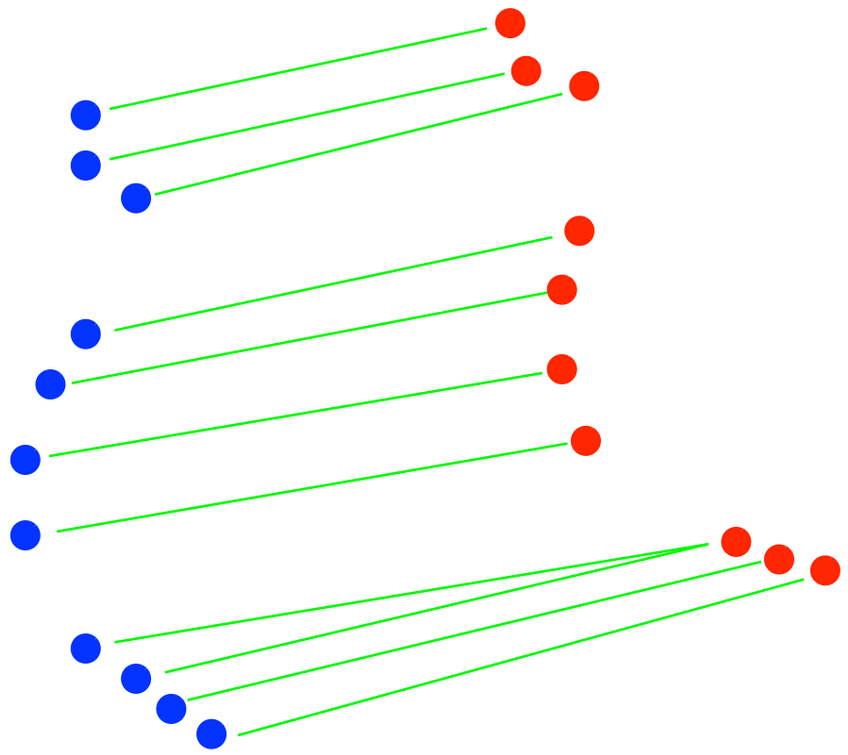


$$m_{st}(x_t) \leftarrow \alpha_{st} \max \phi_s(x_s)\phi_{st}(x_s, x_t) \prod_{u \in \mathcal{N}_s \setminus t} m_{us}(x_s)$$

$$b_s(x_s) \propto \phi_s(x_s) \prod_{t \in \mathcal{N}_s} m_{ts}(x_s)$$

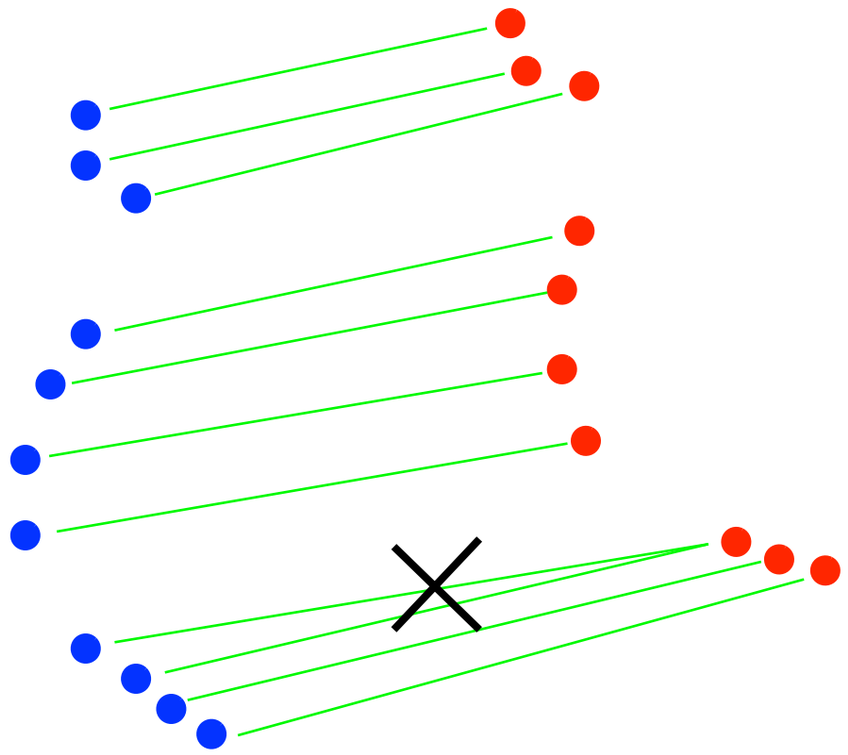
Problems with LBP

- No convergence guarantee
- No optimality guarantee
- Difficult to parallelise
- Difficult to enforce global constraints
  - For example, uniqueness (1 to 1) associations



Problems with LBP

- No convergence guarantee
- No optimality guarantee
- Difficult to parallelise
- Difficult to enforce global constraints
  - For example, uniqueness (1 to 1) associations

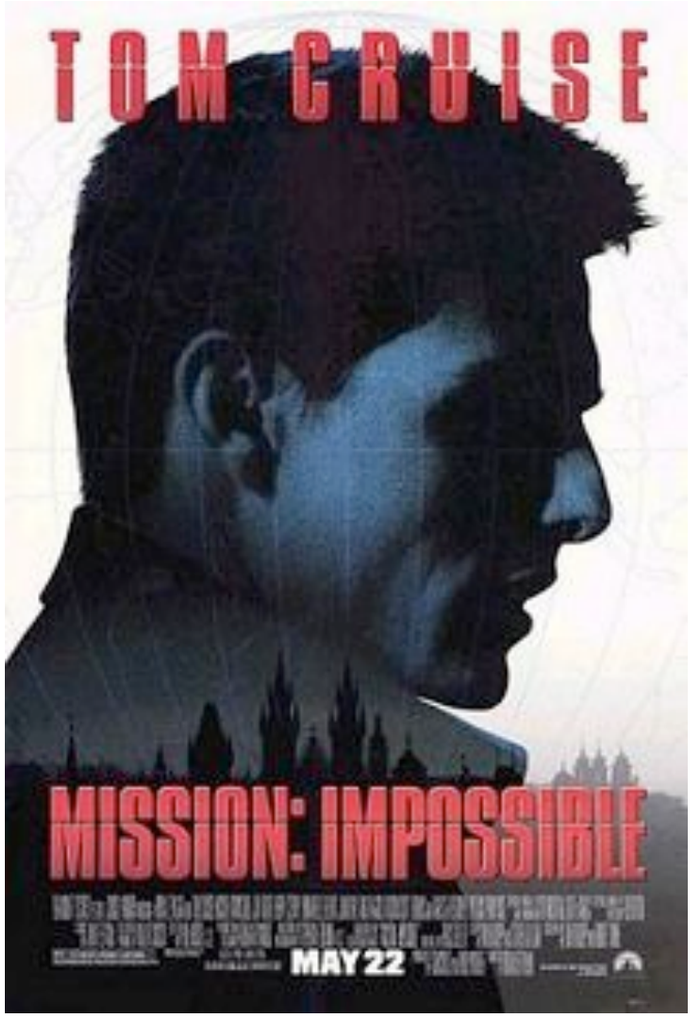


MAP Inference algorithms

- Loopy Belief Propagation
- Tree-Reweighted Belief Propagation
- Graph Cuts
- Iterative conditional modes (ICM)
- Linear programming relaxations
- Quadratic programming relaxations
- Linear Programming Message Passing

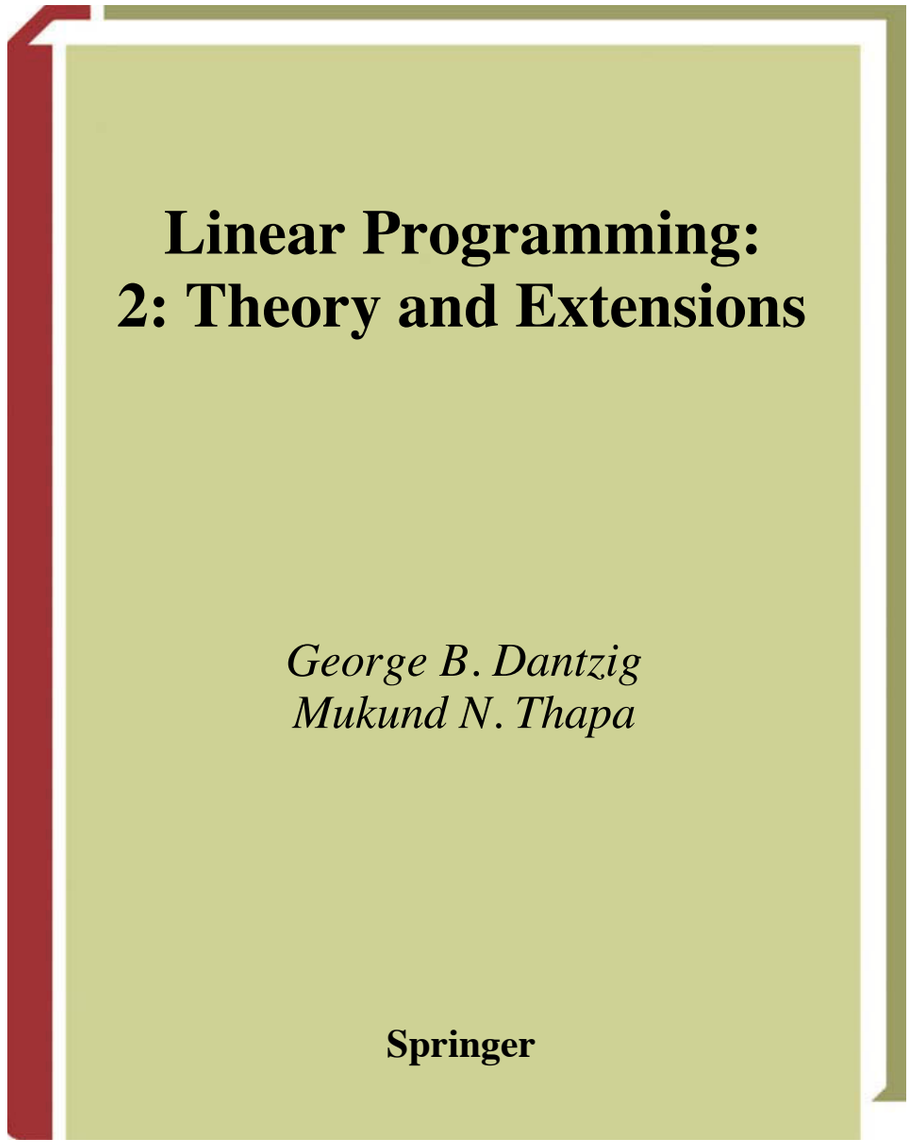
What we want

- Anytime algorithm (monotonic convergence)
- Theoretical guarantees
- Parallel, distributed
- Can handle global constraints
- Can solve really difficult problems



1996

1966


Linear Programming:
2: Theory and Extensions
George B. Dantzig
Mukund N. Thapa
Springer

1963

Linear Programming

Primal

$$\begin{aligned} \text{Minimize} \quad & c^T x = z \\ \text{subject to} \quad & Ax \geq b, \qquad A: \ m \times n, \\ & x \geq 0, \end{aligned}$$

Dual

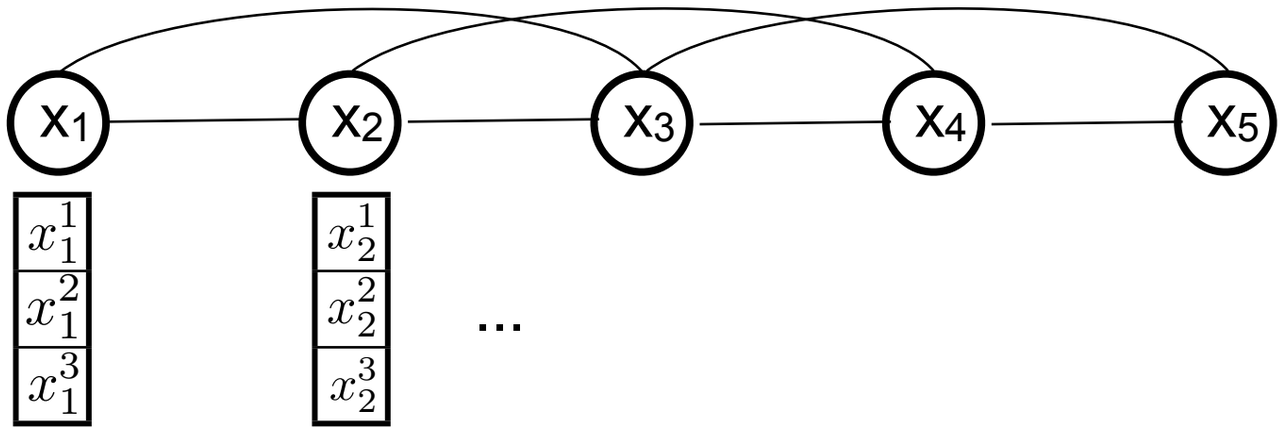
$$\begin{aligned} \text{Maximize} \quad & b^T y = v \\ \text{subject to} \quad & A^T y \leq c, \qquad A: \ m \times n, \\ & y \geq 0. \end{aligned}$$

$$b^T y^* = \max v = \min z = c^T x^*.$$

- A local maximum is also a global
- Very efficient algorithms (Simplex, Interior point)
- Fast implementations (CPLEX, GKLP, etc)
- Very well understood theory

Quadratic Programming Formulation

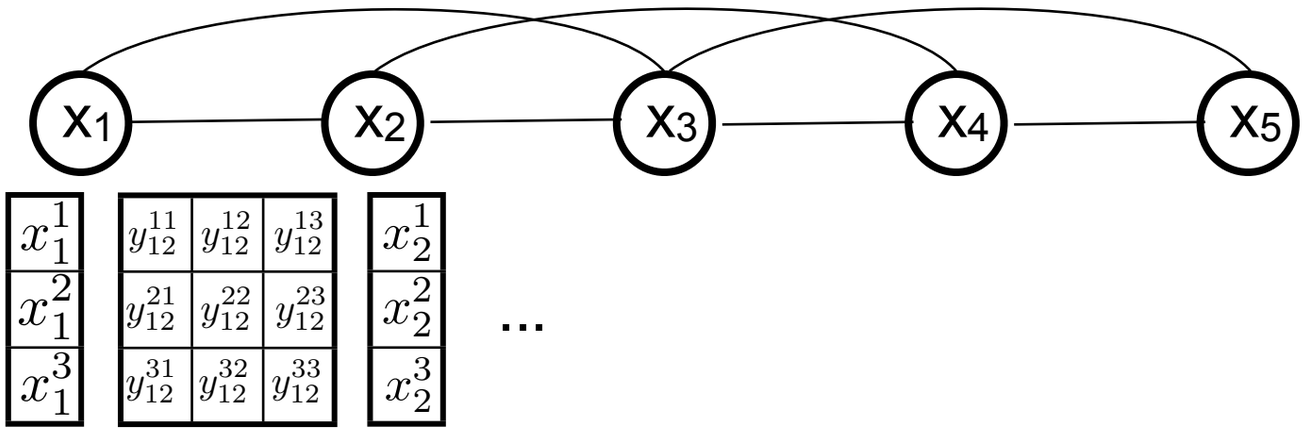
How do we formulate MAP inference in a graphical model as a linear program?



$$X^{MAP} = \underset{X}{\operatorname{argmax}} \log p_G(X;\Phi) = \underset{X}{\operatorname{argmax}} \sum_{(s,t)\in E} x_s^T Q_{st} x_t$$

$$Q_{st} = \phi_{st} + (e\phi_s^T)/|\mathcal{N}(s)| + (\phi_t e^T)/|\mathcal{N}(t)|.$$

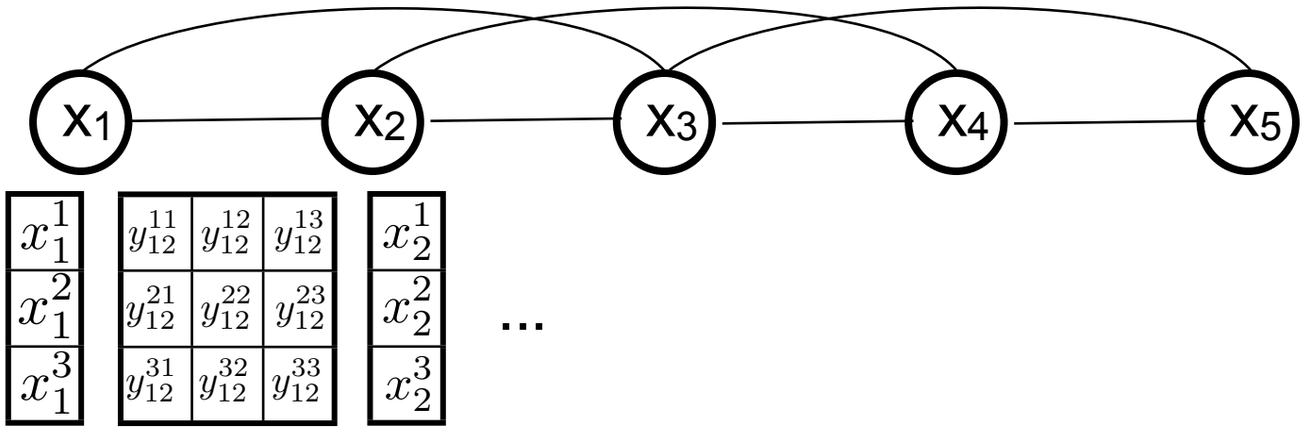
Linear Programming Relaxation



$$y_{st} = (x_s^i x_t^j | i = 1, \ldots, |\mathcal{X}_s|, j = 1, \ldots, |\mathcal{X}_t|, (s,t) \in E)$$

$$\begin{array}{lrll} \text{Maximise} & \sum_{(s,t)\in E} c_{st}^T y_{st} & & \\ \text{Subject to} & A_{st} y_{st} - A_{su} y_{su} = 0 & \forall s \in V, t \in \mathcal{N}(s), \forall u \in \mathcal{N}(s), u \neq t \\ & \sum y_{st} = 1 & \forall (s,t) \in E \\ & y_{st} \in \{0,1\} & \forall (s,t) \in E. \end{array}$$

Linear Programming Relaxation

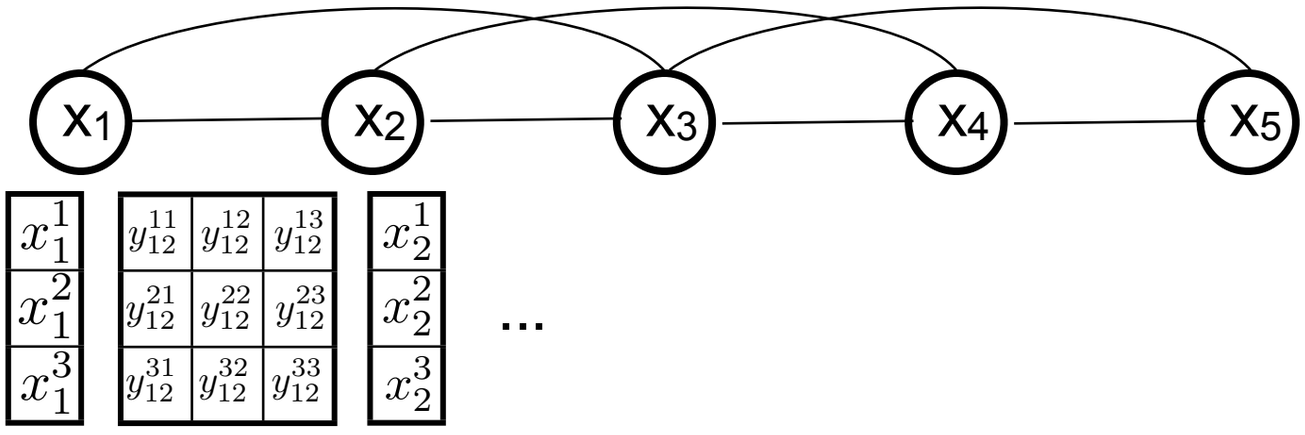


$$y_{st} = (x_s^i x_t^j | i = 1, \ldots, |\mathcal{X}_s|, j = 1, \ldots, |\mathcal{X}_t|, (s,t) \in E)$$

c_{st} is a vector constructed from Q_{st}

$$\begin{array}{lll} \text{Maximise} & \sum_{(s,t)\in E} c_{st}^T y_{st} & \\ \text{Subject to} & A_{st} y_{st} - A_{su} y_{su} = 0 & \forall s \in V, t \in \mathcal{N}(s), \forall u \in \mathcal{N}(s), u \neq t \\ & \sum y_{st} = 1 & \forall (s,t) \in E \\ & y_{st} \in \{0,1\} & \forall (s,t) \in E. \end{array}$$

Linear Programming Relaxation



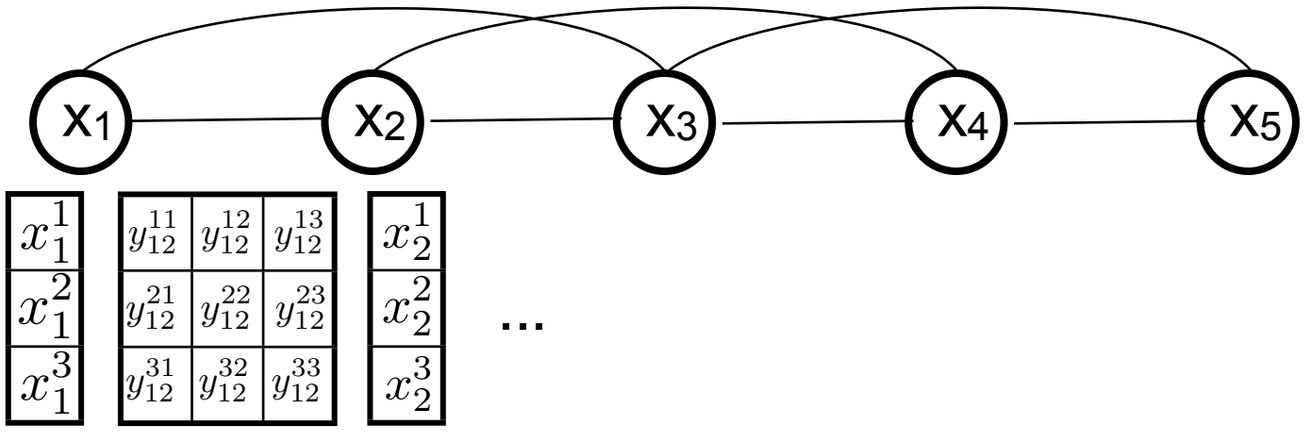
$$y_{st} = (x_s^i x_t^j | i = 1, \ldots, |\mathcal{X}_s|, j = 1, \ldots, |\mathcal{X}_t|, (s,t) \in E)$$

c_{st} is a vector constructed from Q_{st}

$$\begin{array}{lll} \text{Maximise} & \sum_{(s,t)\in E} c_{st}^T y_{st} & \\ \text{Subject to} & A_{st} y_{st} - A_{su} y_{su} = 0 & \forall s \in V, t \in \mathcal{N}(s), \forall u \in \mathcal{N}(s), u \neq t \\ & \sum y_{st} = 1 & \forall (s,t) \in E \\ & y_{st} \in \{0,1\} & \forall (s,t) \in E. \end{array}$$

Consistency constraints

Linear Programming Relaxation



$$y_{st} = (x_s^i x_t^j | i = 1, \ldots, |\mathcal{X}_s|, j = 1, \ldots, |\mathcal{X}_t|, (s,t) \in E)$$

c_{st} is a vector constructed from Q_{st}

$$\begin{array}{lll} \text{Maximise} & \sum_{(s,t)\in E} c_{st}^T y_{st} & \\ \text{Subject to} & A_{st} y_{st} - A_{su} y_{su} = 0 & \forall s \in V, t \in \mathcal{N}(s), \forall u \in \mathcal{N}(s), u \neq t \\ & \sum y_{st} = 1 & \forall (s,t) \in E \\ & 0 \leq y_{st} \leq 1 & \forall (s,t) \in E. \end{array}$$

Consistency constraints

Relaxation

Stop! How many variables?

- For a 360 scan matching problem in a chain graph:
  - 360 x 360 x 360 = 47M variables!

Conventional LP inference does not scale well!

Linear Programming Decompositions

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{n_1} c_j x_j + \sum_{j=n_1+1}^{n} c_j x_j = z \\
\text{subject to} \quad & \sum_{j=1}^{n_1} A_{ij} x_j = b_i, \quad i = 1, \ldots, m_1 \\
& \sum_{j=n_1+1}^{n} A_{ij} x_j = b_i, \quad i = m_1 + 1, \ldots, m \\
& x_j \geq 0, \quad j = 1, \ldots, n.
\end{aligned}
$$

Linear Programming Decompositions

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{n_1} c_j x_j + \sum_{j=n_1+1}^{n} c_j x_j = z \\
\text{subject to} \quad & \sum_{j=1}^{n_1} A_{ij} x_j = b_i, \quad i = 1, \ldots, m_1 \\
& \sum_{j=n_1+1}^{n} A_{ij} x_j = b_i, \quad i = m_1 + 1, \ldots, m \\
& x_j \geq 0, \quad j = 1, \ldots, n.
\end{aligned}
$$

Block Angular System

$$
\begin{array}{lccccccc}
\text{Maximise} & c_1^T x_1 & + & \ldots & + & c_K^T x_K & & \\
\text{Subject to} & B_1 x_1 & + & \ldots & + & B_K x_K & = & b \\
& F_1 x_1 & & & & & = & f_1 \\
& & & \ddots & & & \vdots & \\
& & & & & F_K x_K & = & f_K \\
& & & x_i \geq 0 & & i & = & 1, \ldots, K,
\end{array}
$$

Linear Programming Decompositions

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{n_1} c_j x_j + \sum_{j=n_1+1}^{n} c_j x_j = z \\
\text{subject to} \quad & \sum_{j=1}^{n_1} A_{ij} x_j = b_i, \quad i = 1, \ldots, m_1 \\
& \sum_{j=n_1+1}^{n} A_{ij} x_j = b_i, \quad i = m_1 + 1, \ldots, m \\
& x_j \geq 0, \quad j = 1, \ldots, n.
\end{aligned}
$$

Block Angular System

$$
\begin{array}{lccccccc}
\text{Maximise} & c_1^T x_1 & + & \ldots & + & c_K^T x_K & & \\
\text{Subject to} & B_1 x_1 & + & \ldots & + & B_K x_K & = & b \\
& F_1 x_1 & & & & & = & f_1 \\
& & & \ddots & & & \vdots & \\
& & & & & F_K x_K & = & f_K \\
& & & x_i \geq 0 & & i & = & 1, \ldots, K,
\end{array}
$$

Similar structure to our problem

Linear Programming Decompositions

$$
\begin{aligned}
\text{Minimize} \quad & \sum_{j=1}^{n_1} c_j x_j + \sum_{j=n_1+1}^{n} c_j x_j = z \\
\text{subject to} \quad & \sum_{j=1}^{n_1} A_{ij} x_j = b_i, \quad i = 1, \ldots, m_1 \\
& \sum_{j=n_1+1}^{n} A_{ij} x_j = b_i, \quad i = m_1 + 1, \ldots, m \\
& x_j \geq 0, \quad j = 1, \ldots, n.
\end{aligned}
$$

Block Angular System

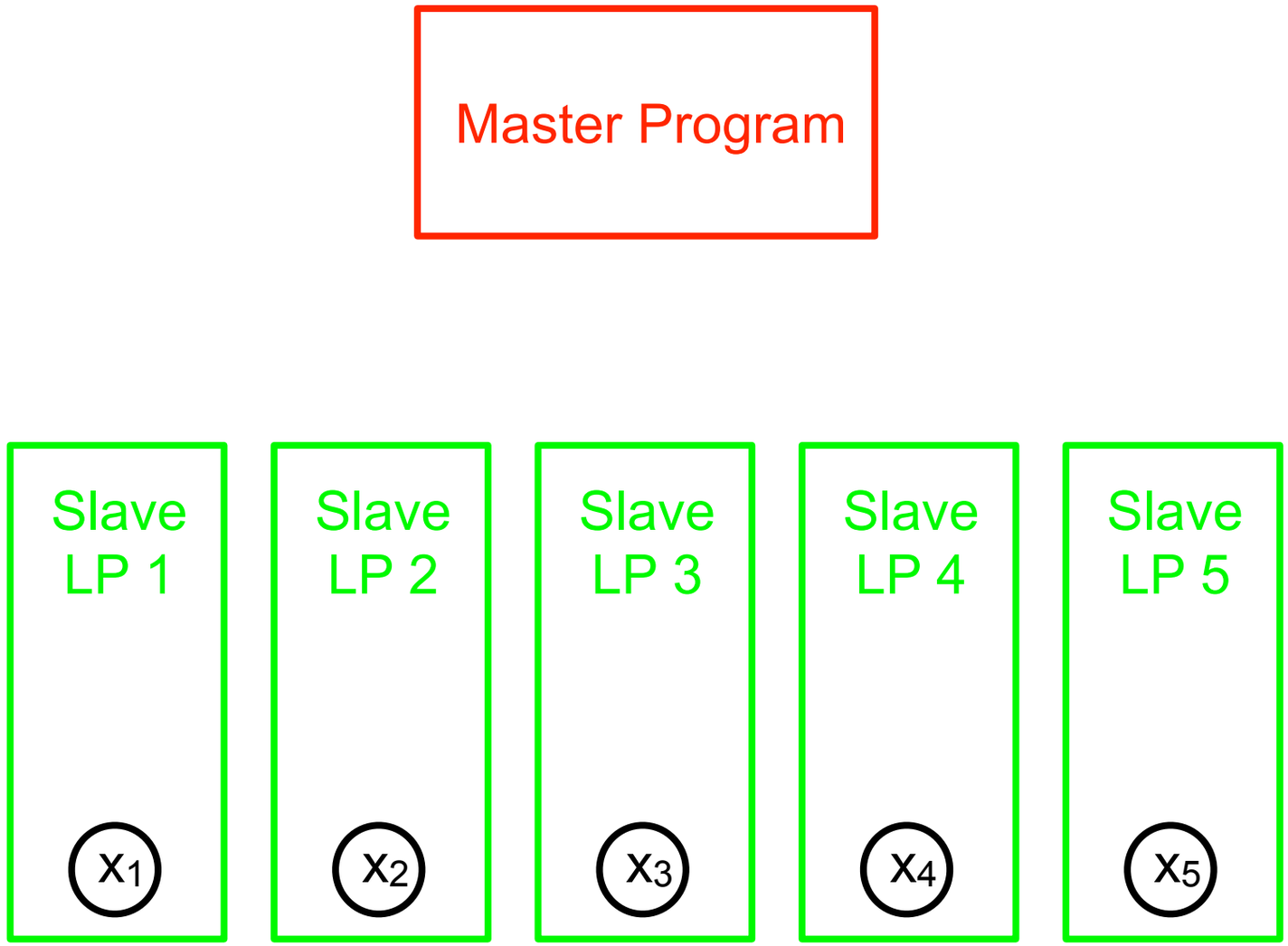
$$
\begin{array}{lccccccc}
\text{Maximise} & c_1^T x_1 & + & \ldots & + & c_K^T x_K & & \\
\text{Subject to} & B_1 x_1 & + & \ldots & + & B_K x_K & = & b \\
& F_1 x_1 & & & & & = & f_1 \\
& & & \ddots & & & \vdots & \\
& & & & & F_K x_K & = & f_K \\
& & & x_i \geq 0 & & i & = & 1, \ldots, K,
\end{array}
$$

Dantzig-Wolfe Algorithm

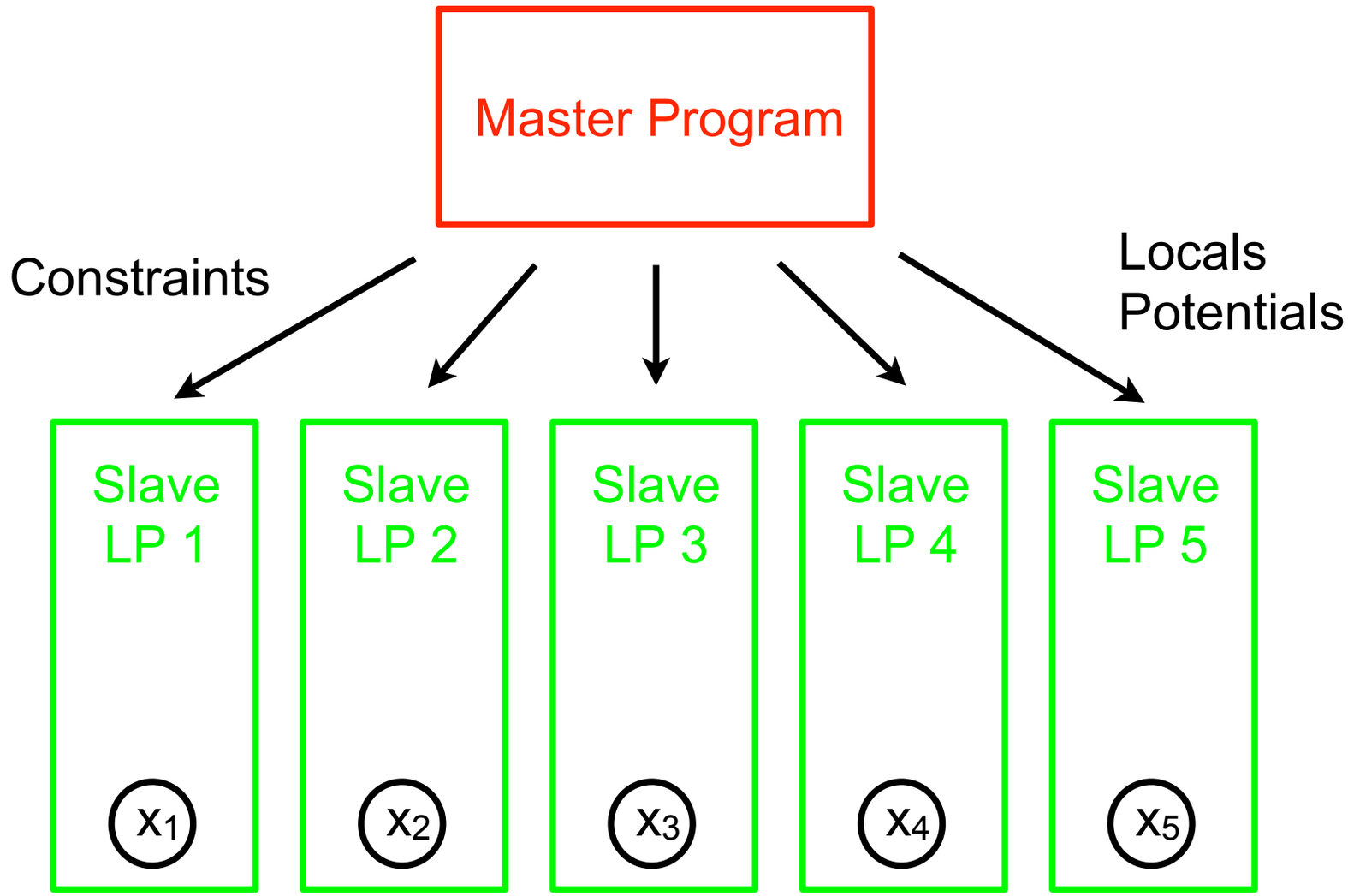
↑

Similar structure to our problem

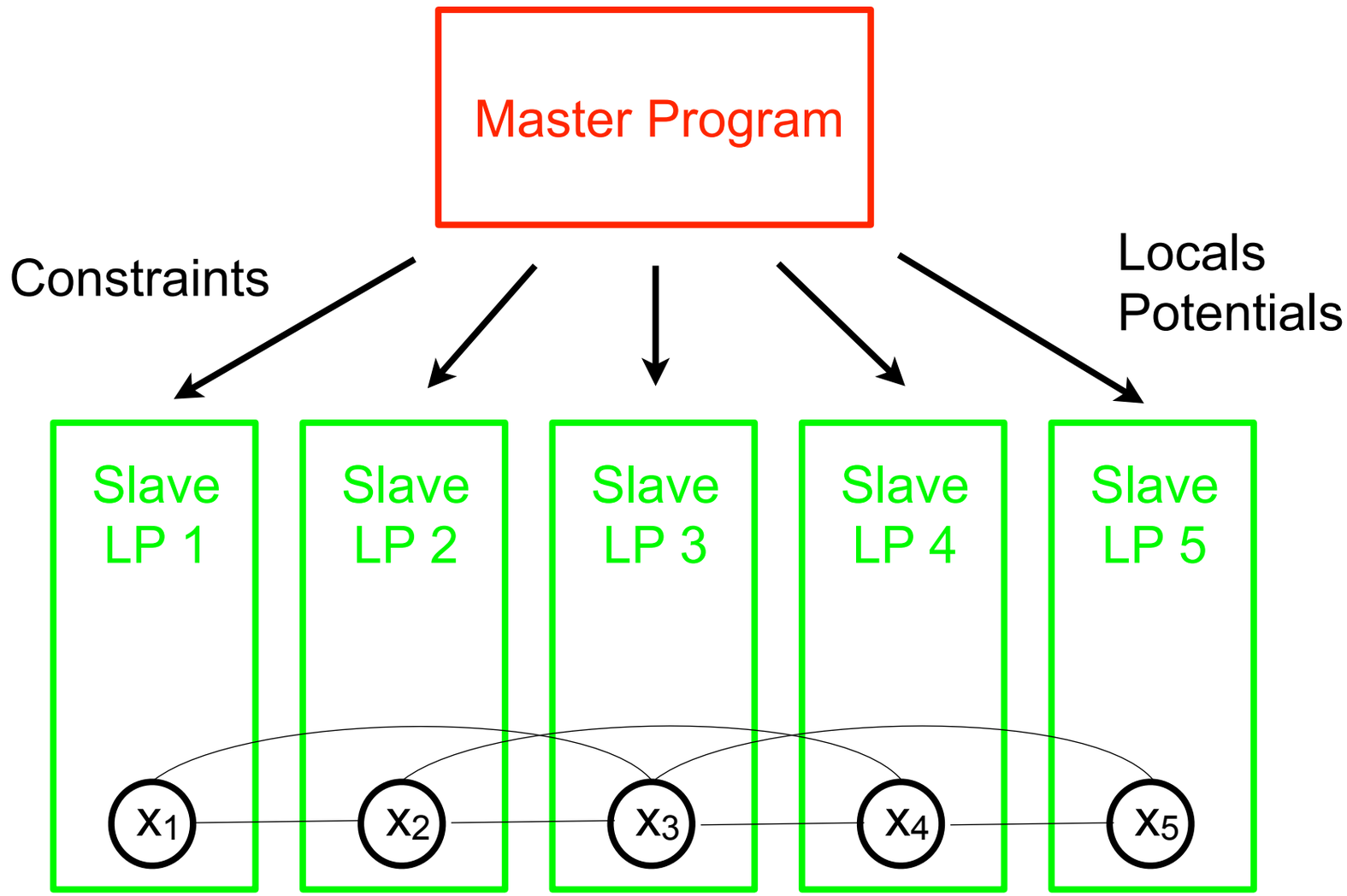
Distributed LP Inference



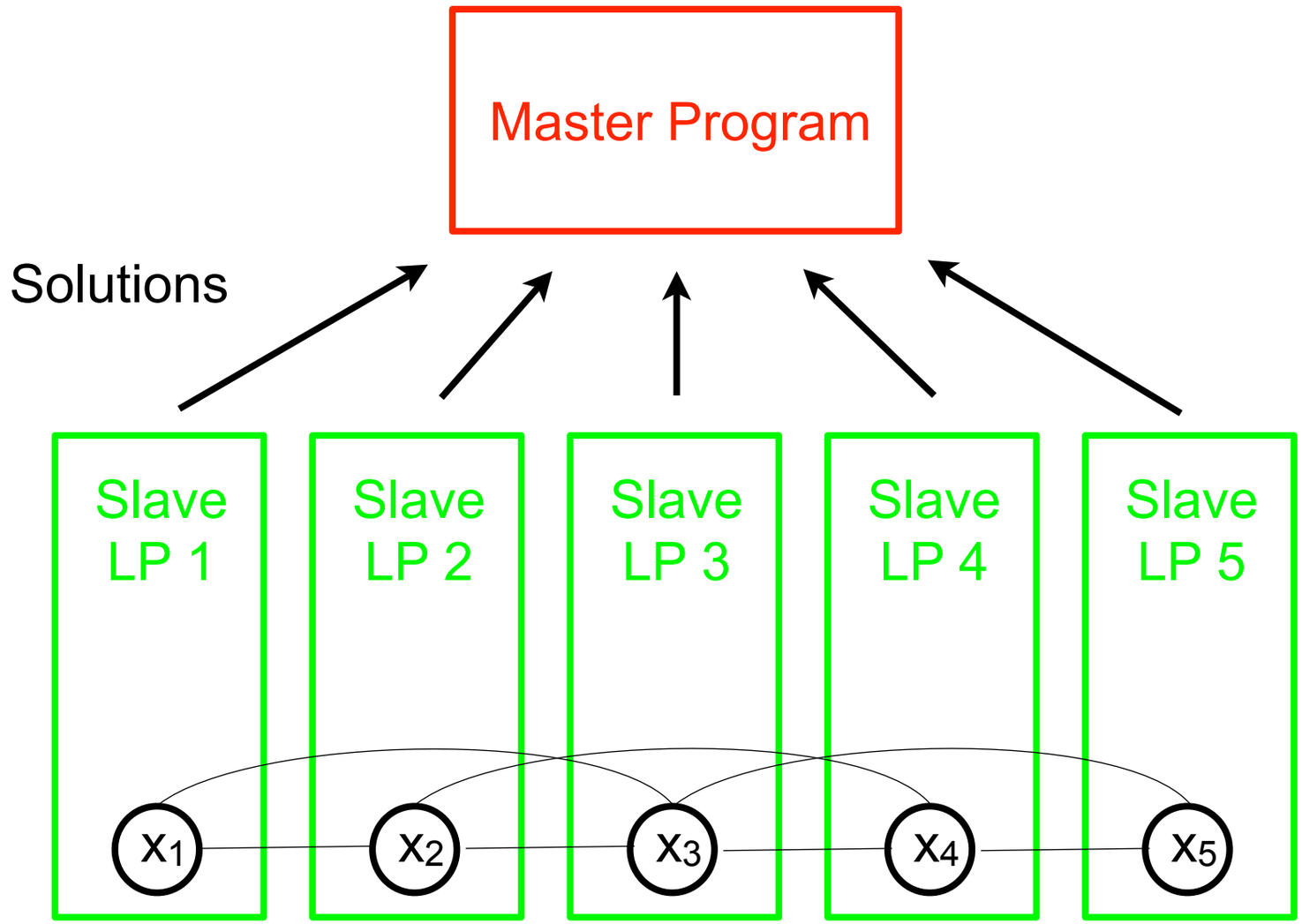
Distributed LP Inference



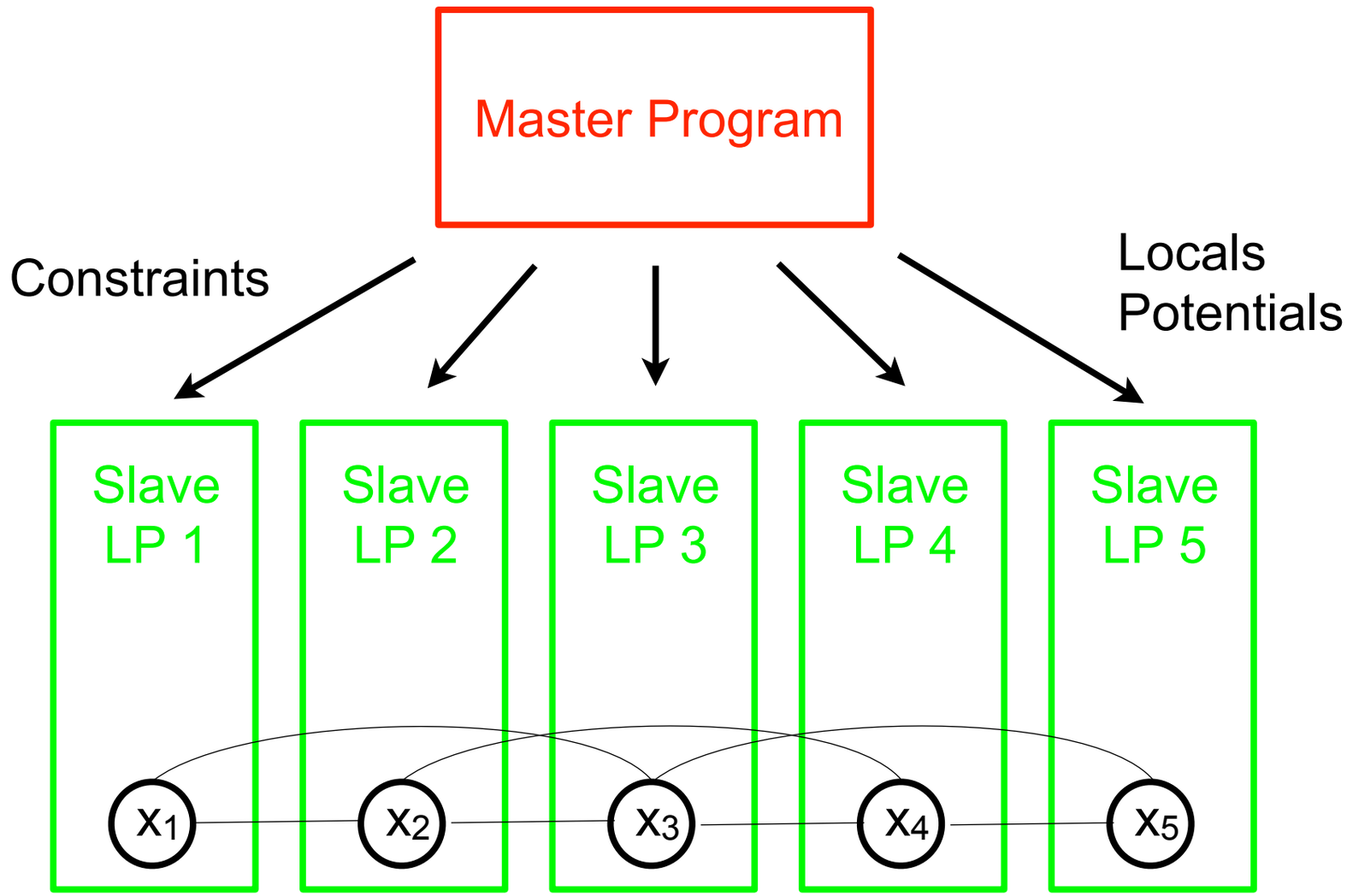
Distributed LP Inference



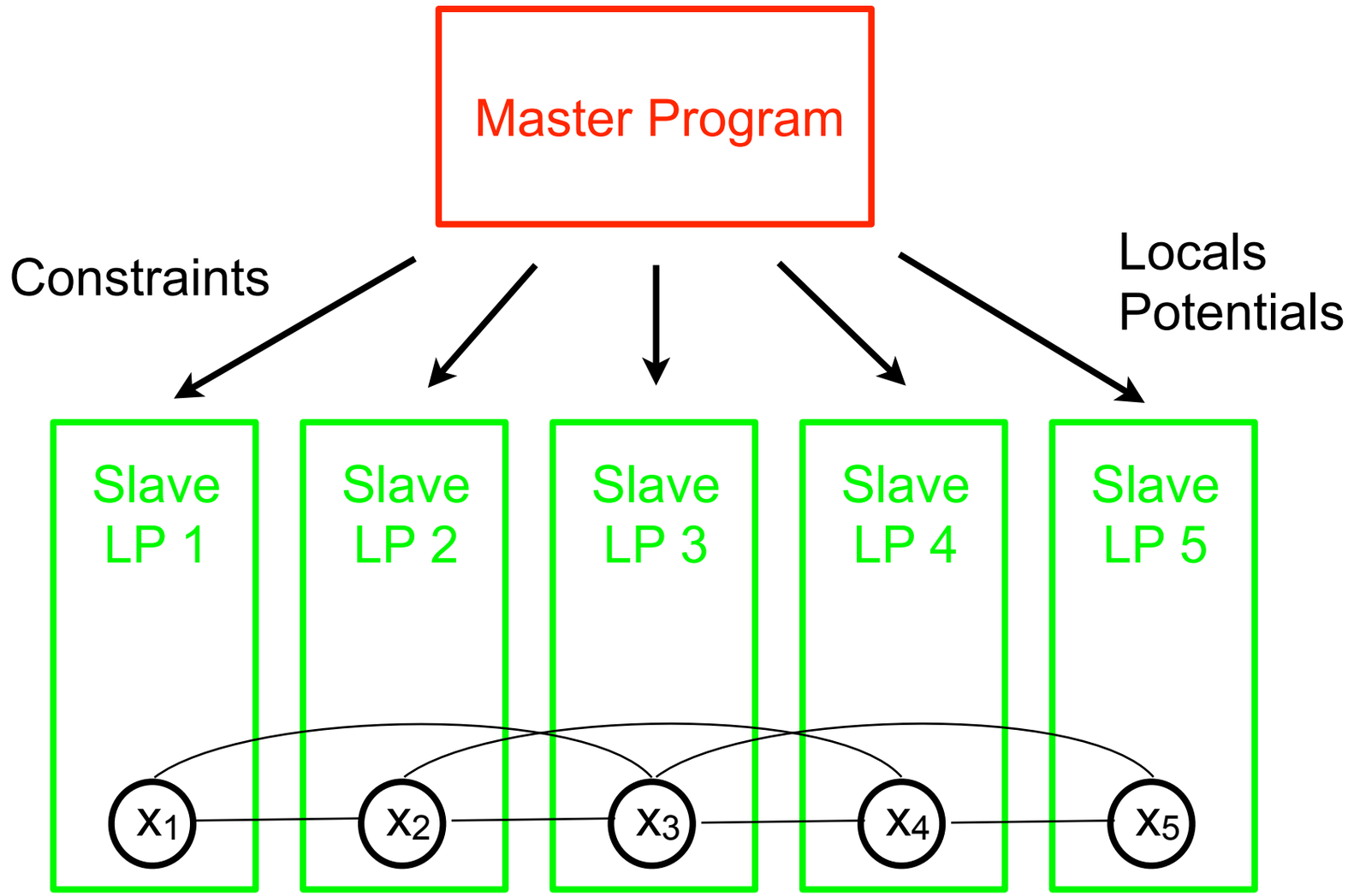
Distributed LP Inference



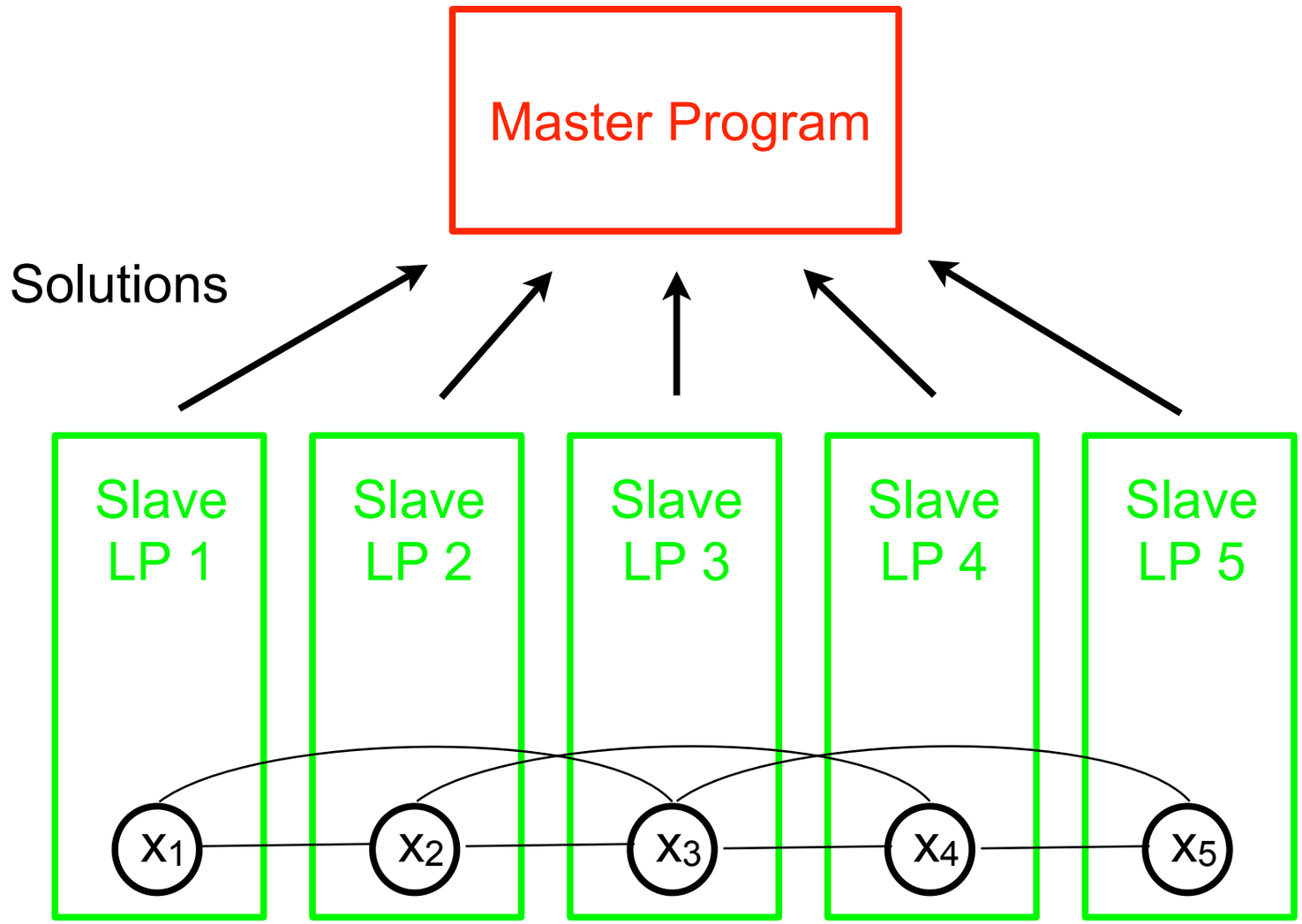
Distributed LP Inference



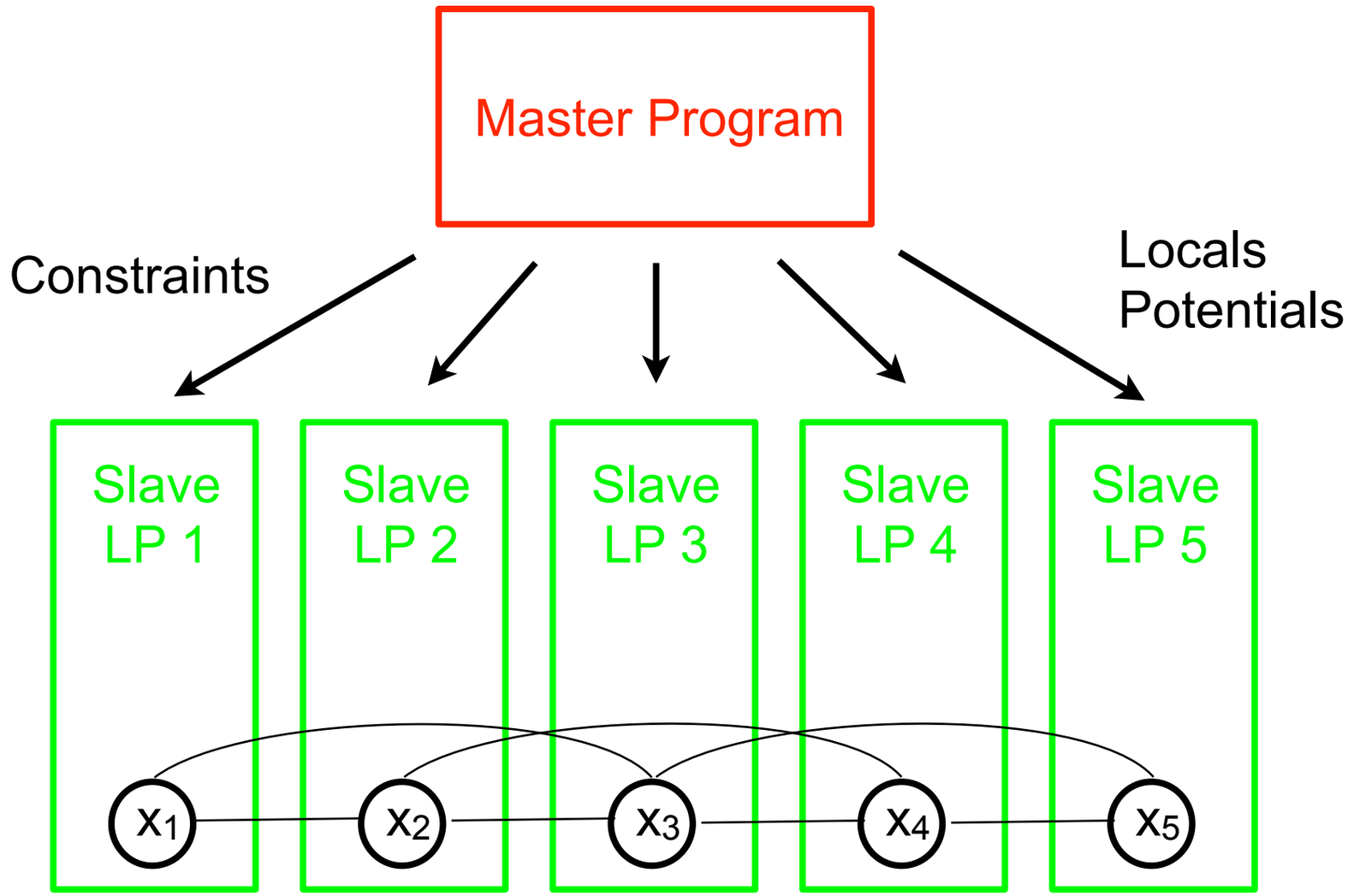
Distributed LP Inference



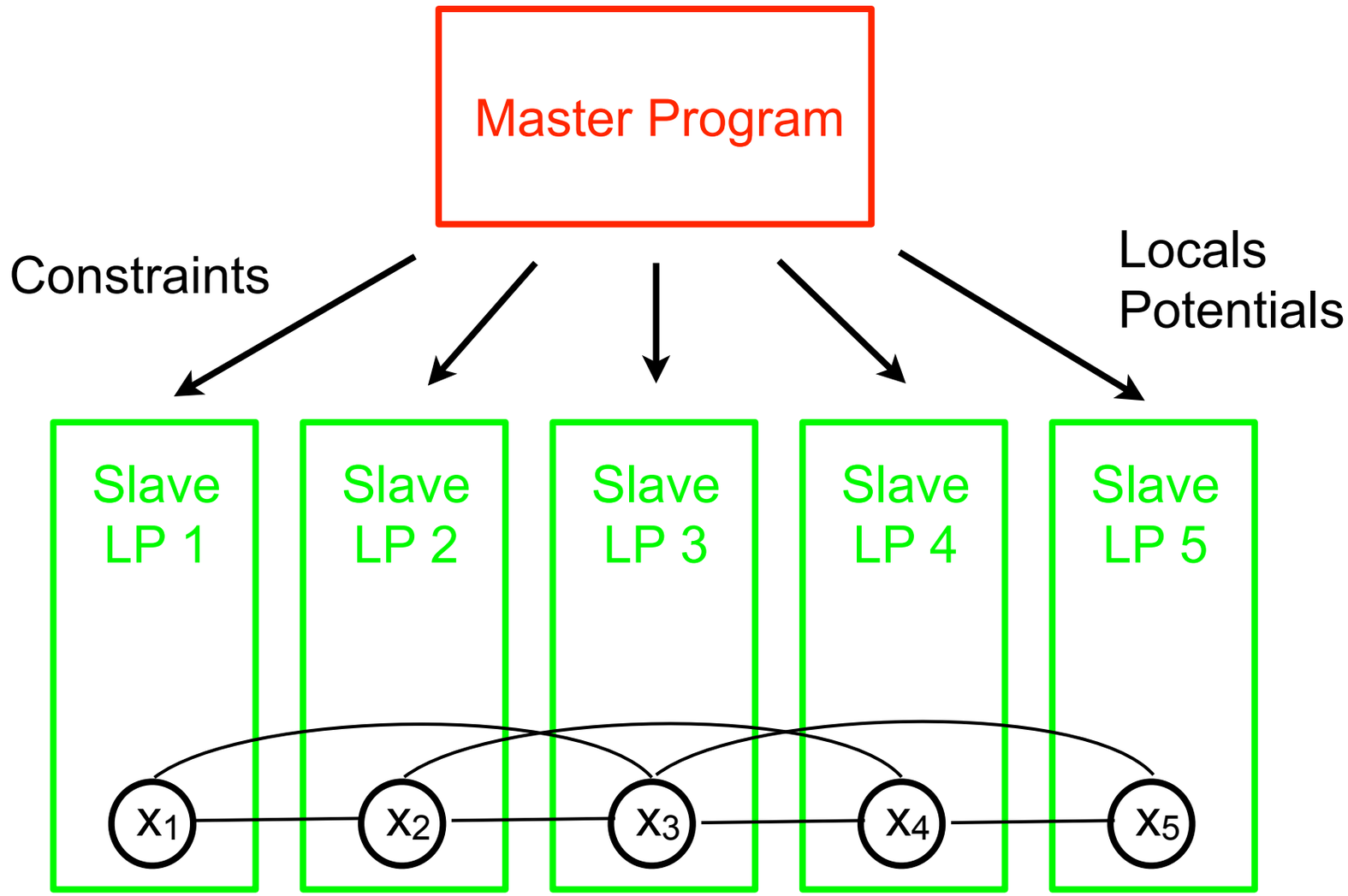
Distributed LP Inference



Distributed LP Inference



Distributed LP Inference



Distributed LP Inference

Maximise $\sum_{(s,t)\in E} \sum_{i\in L_{st}} g^i_{st}\alpha^i_{st} = z$

Subject to $\sum_{(s,t)\in E} \sum_{i\in L_{st}} G^i_{st}\alpha^i_{st} = 0 \quad : \pi$

$\sum \alpha_{st} = 1 \quad \forall (s,t) \in E \quad : \gamma$

$\alpha_{st} \geq 0 \quad \forall (s,t) \in E.$

Master Program

$G^i_{st} = B_{st}\hat{y}_{st,i}$

$g^i_{st} = c^T_{st}\hat{y}_{st,i}$

π

γ

Maximise $c^T_{st} y_{st} - B^T_{st}\pi$

Subject to $\sum y_{st} = 1$

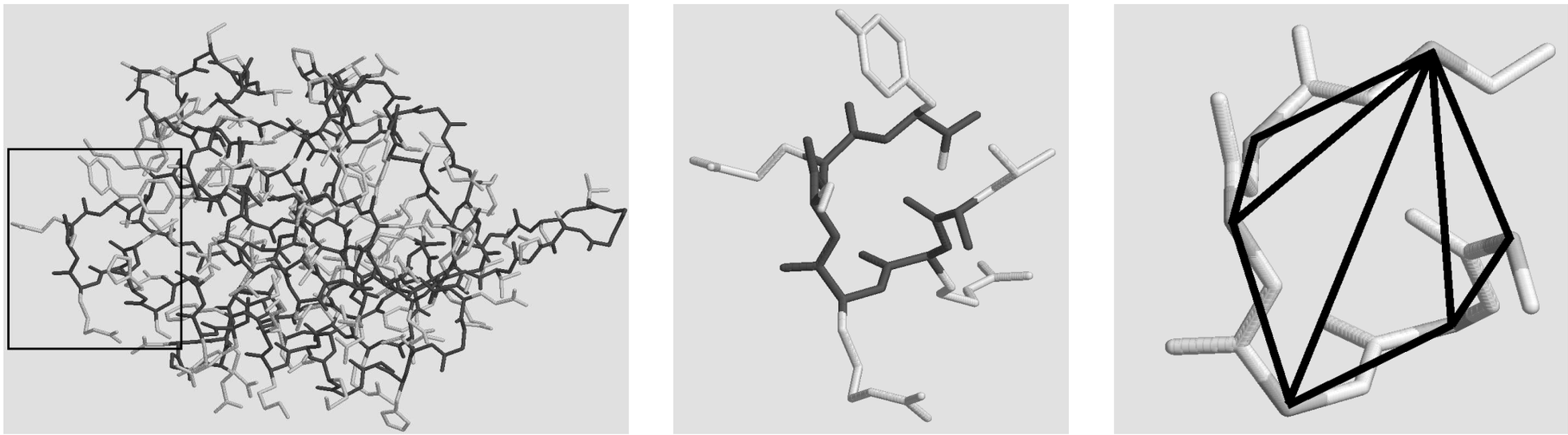
$0 \leq y_{st} \leq 1.$

Slave Program

$$\check{y}_{st} = \sum_{i\in L_{st}} \alpha^i_{st}\hat{y}_{st,i}.$$

Optimal Solution

Protein Design



(a) **(b)** **(c)**

Only “Tightening LPMP” (Sontag,2008) was able to solve these problems.

Results

		min	max	mean	std
Execution Time	(s)	7.735	1404.005	176.645	299.579
Master Program	(%)	42.410	71.594	58.118	8.335
Sub Program	(%)	23.734	51.906	37.293	7.574
Rounding & Overhead	(%)	1.177	7.746	4.589	1.802

Table 1: Timing Statistics.

		min	max	mean	std
MAP Value Approximation	(%)	0.026	38.210	3.313	7.879
MAP Configuration Approximation	(%)	2.033	8.276	4.984	1.825
Fractional Solution	(%)	0.434	10.435	3.088	2.094
Fractional Approximations	(%)	0	66.667	33.581	16.171

Table 2: Solution Quality Statistics.

We are able to solve to within 3.3% of their value without rounding.

Results

		min	max	mean	std
Execution Time	(s)	7.735	1404.005	176.645	299.579
Master Program	(%)	42.410	71.594	58.118	8.335
Sub Program	(%)	23.734	51.906	37.293	7.574
Rounding & Overhead	(%)	1.177	7.746	4.589	1.802

Table 1: Timing Statistics.

		min	max	mean	std
MAP Value Approximation	(%)	0.026	38.210	3.313	7.879
MAP Configuration Approximation	(%)	2.033	8.276	4.984	1.825
Fractional Solution	(%)	0.434	10.435	3.088	2.094
Fractional Approximations	(%)	0	66.667	33.581	16.171

Table 2: Solution Quality Statistics.

We are able to solve to within 3.3% of their value without rounding.

Sontag's average time: 9 minutes

Results

		min	max	mean	std
Execution Time	(s)	7.735	1404.005	176.645	299.579
Master Program	(%)	42.410	71.594	58.118	8.335
Sub Program	(%)	23.734	51.906	37.293	7.574
Rounding & Overhead	(%)	1.177	7.746	4.589	1.802

Table 1: Timing Statistics.

		min	max	mean	std
MAP Value Approximation	(%)	0.026	38.210	3.313	7.879
MAP Configuration Approximation	(%)	2.033	8.276	4.984	1.825
Fractional Solution	(%)	0.434	10.435	3.088	2.094
Fractional Approximations	(%)	0	66.667	33.581	16.171

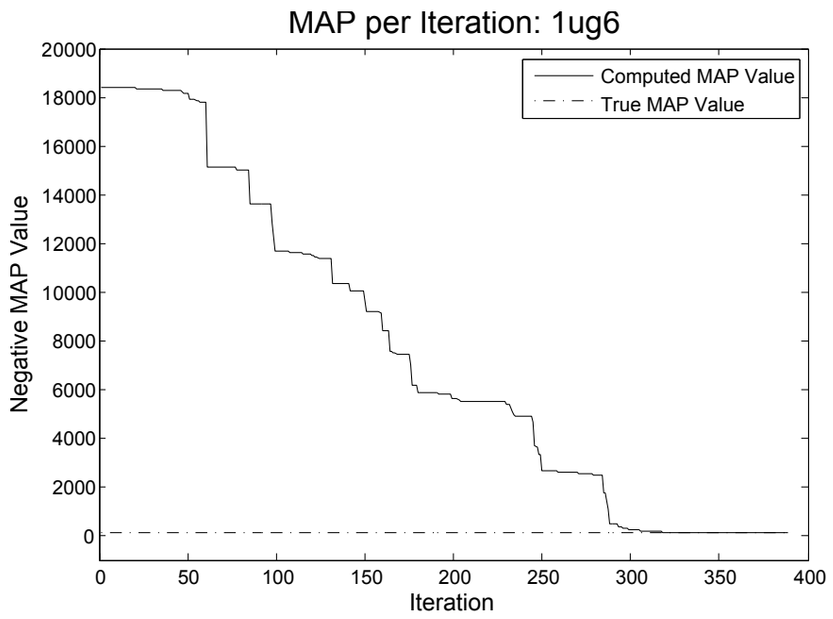
Table 2: Solution Quality Statistics.

We are able to solve to within 3.3% of their value without rounding.

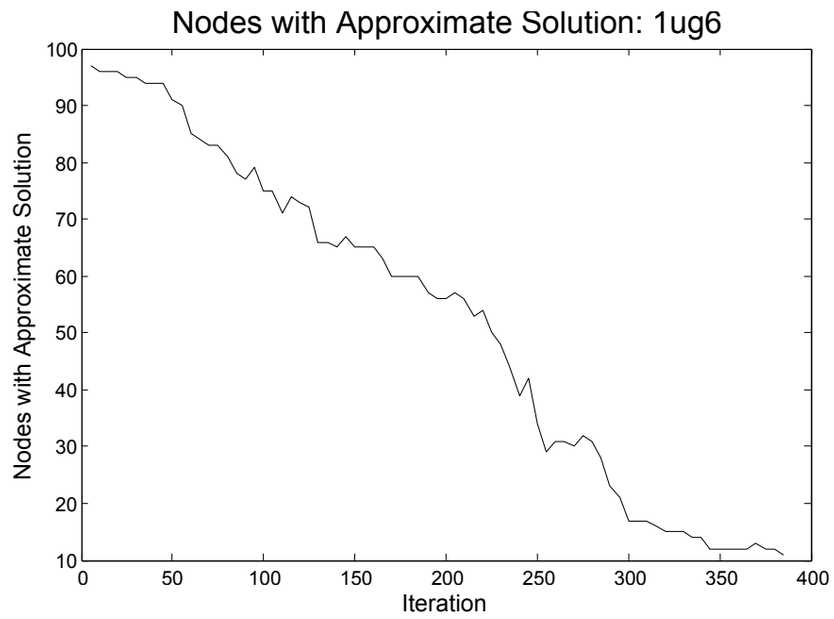
Sontag's average time: 9 minutes

Ours: 53 seconds

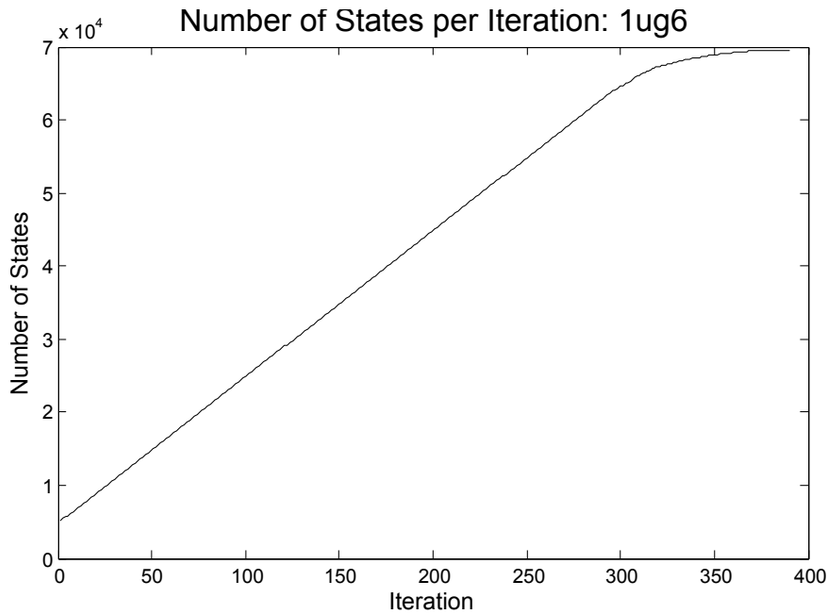
MAP Evolution



(a)



(b)



(c)

(d)

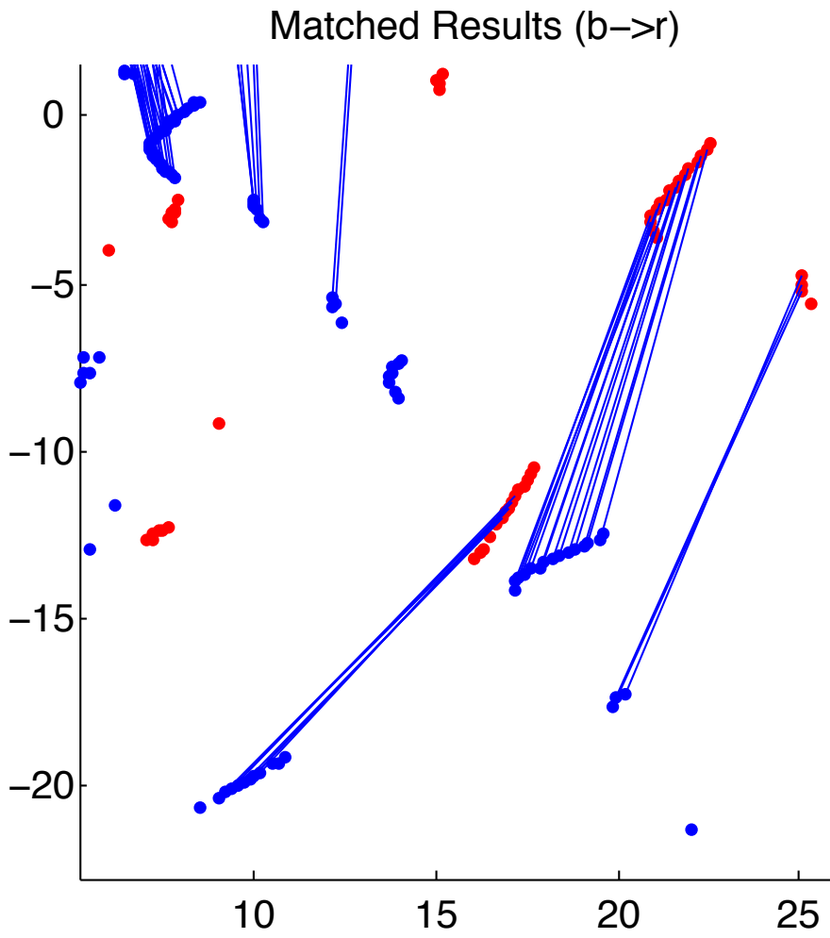
Used Variables

		min	max	mean	std
LP Variables		92151	1569834	460245	379522
Master Program Variables		6322	183082	41608	43069
LP Variables Used	(%)	3.987	13.081	8.164	2.447

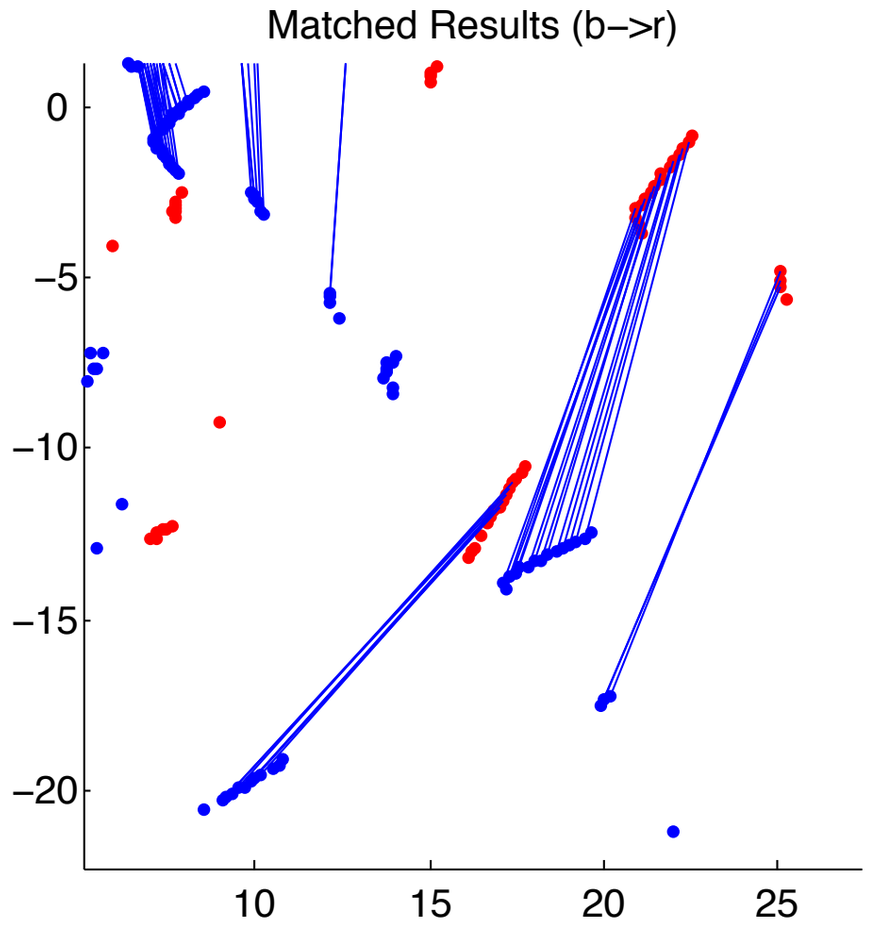
Table 3: Variable Usage Statistics.

We solve the problem with only 8% of the original LP variables.

Scan Matching

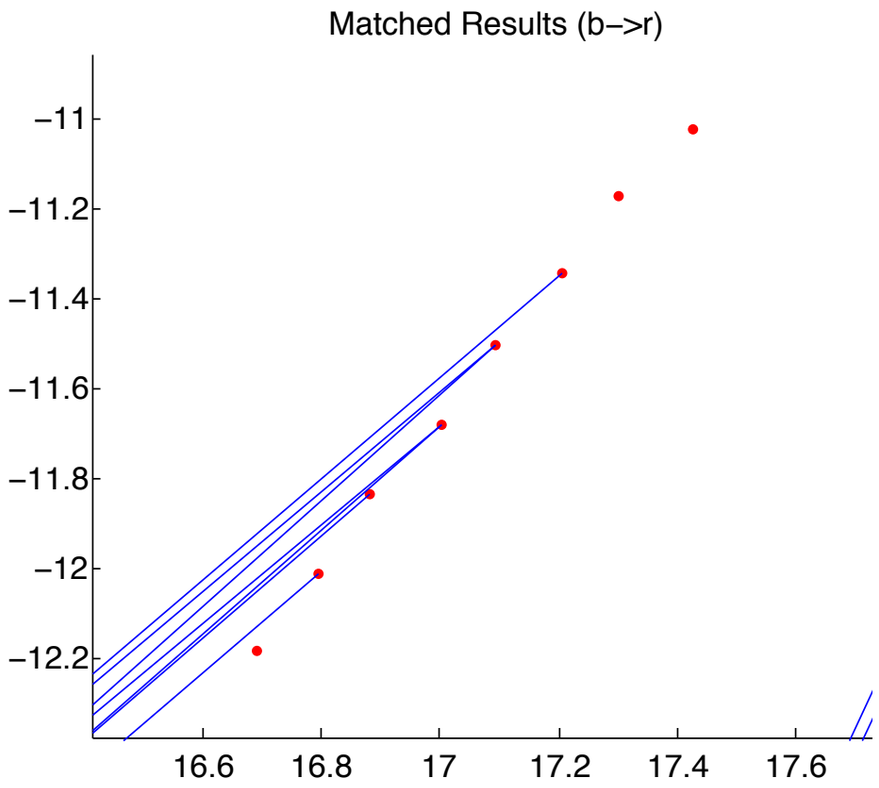


LBP

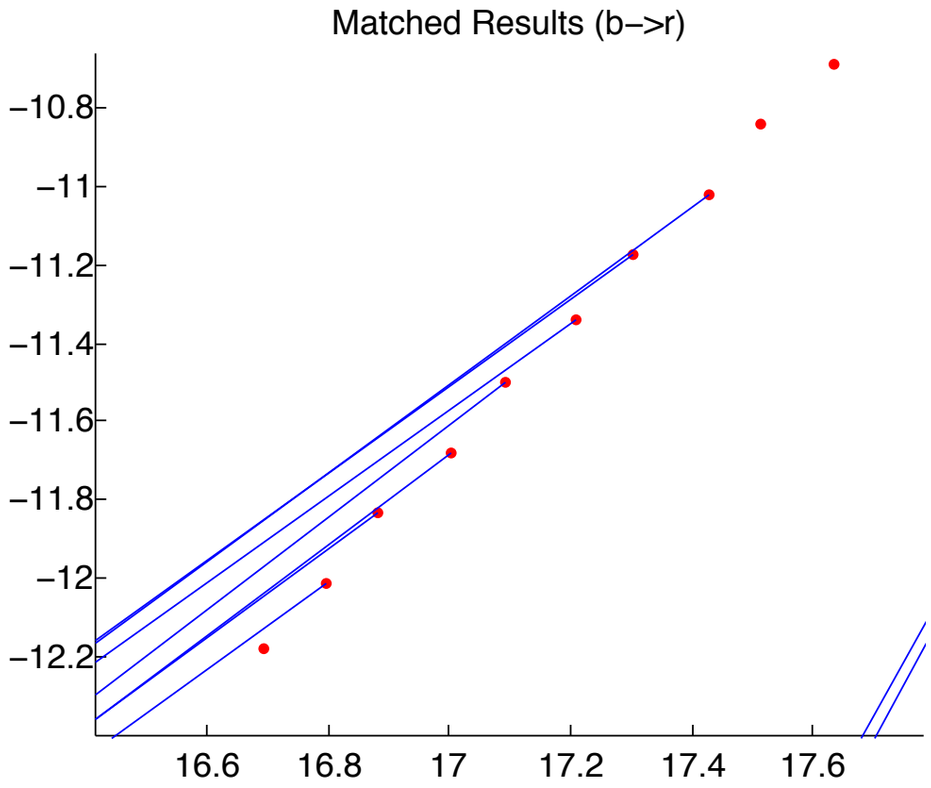


DWLP

Scan Matching



LBP



DWLP

Summary

- New LP based MAP inference method
- Distributed
- Handles global constraints
- Reduces the number of variables
- Solves LP problems that do not fit in memory
- As efficient as state-of-the-art MP algorithms

Questions:

Joop van de Ven

j.vandeven@acfr.usyd.edu.au

Tech Report:

http://www.acfr.usyd.edu.au/techreports/2010-001.shtml

Thanks!