WPI

Efficient Factor Graph Fusion for Multi-robot Mapping

Ramkumar Natarajan

Advisor: Prof. Michael A. Gennert

Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

Online Mapping

- Key capability for robot autonomy

Online Mapping

- Key capability for robot autonomy

Challenges:

- Prior maps not available.
  - home, retail store, search & rescue.

Online Mapping

- Key capability for robot autonomy

Challenges:

- Prior maps not available.
  - home, retail store, search & rescue.
- Environments change over time.
- Difficult to automate even with GPS.
  - Google and UBER driverless cars use pre-generated maps.
  - Without GPS even more challenging.

Online Mapping

- Key capability for robot autonomy

Challenges:

- Prior maps not available.
  - home, retail store, search & rescue.
- Environments change over time.
- Difficult to automate even with GPS.
  - Google and UBER driverless cars use pre-generated maps.
  - Without GPS even more challenging.
- High and efficient processing requirements for multiple robots in large scale environment.

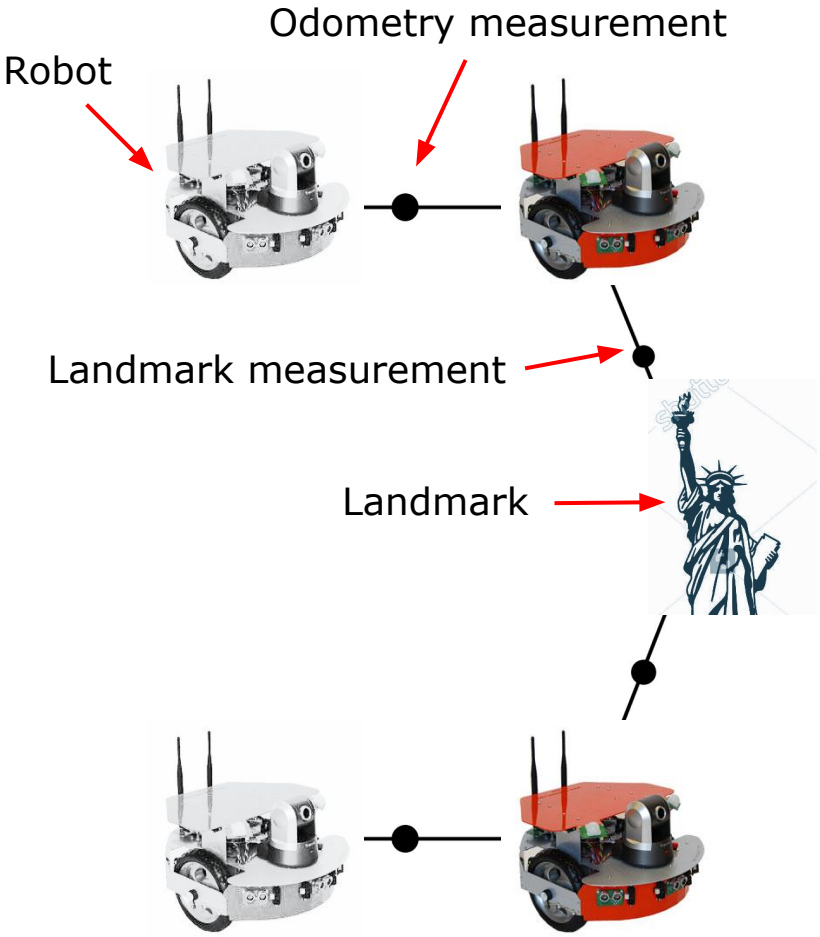
Multi-robot Mapping Problem (Simultaneous Localization and Mapping)

- Onboard sensors:
  - Wheel odometry
  - Inertial measurement unit
  - Laser range finder
  - Sonar
  - Camera
  - RGB-D sensor

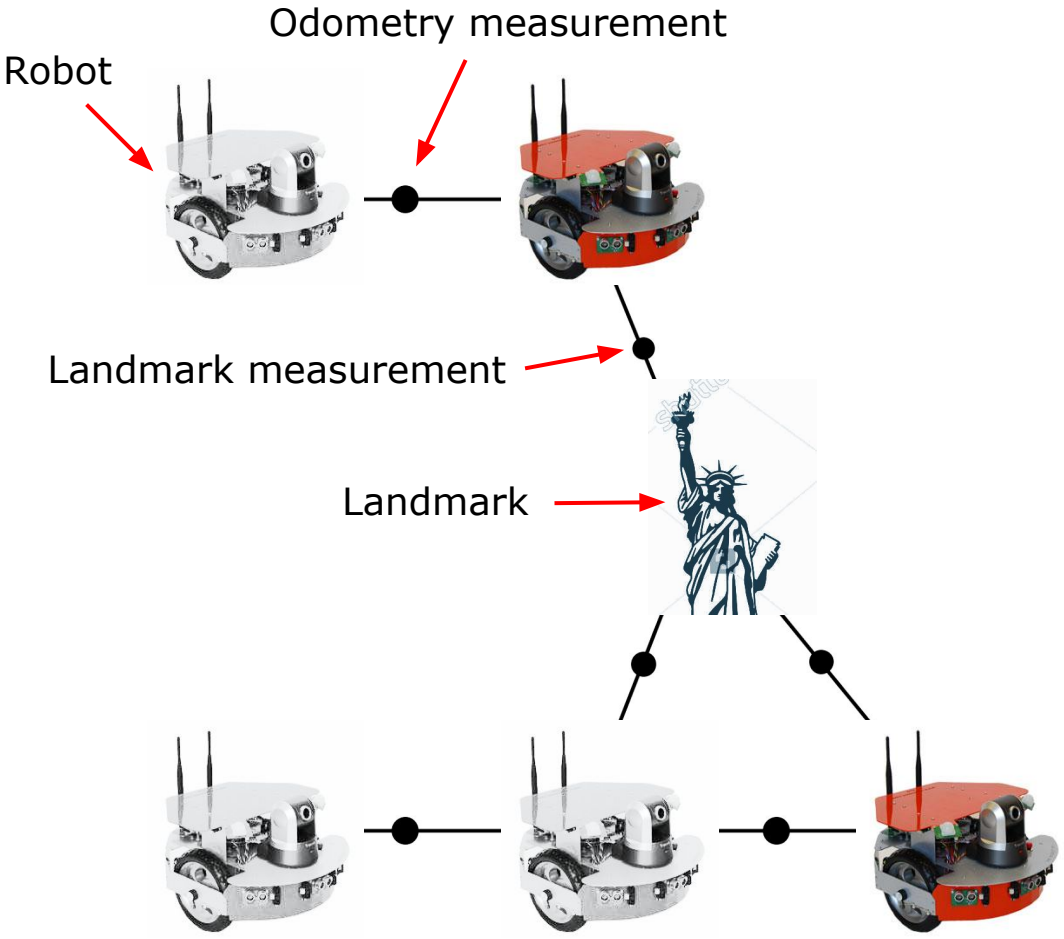
Multi-robot Mapping Problem (t=0)

Robot

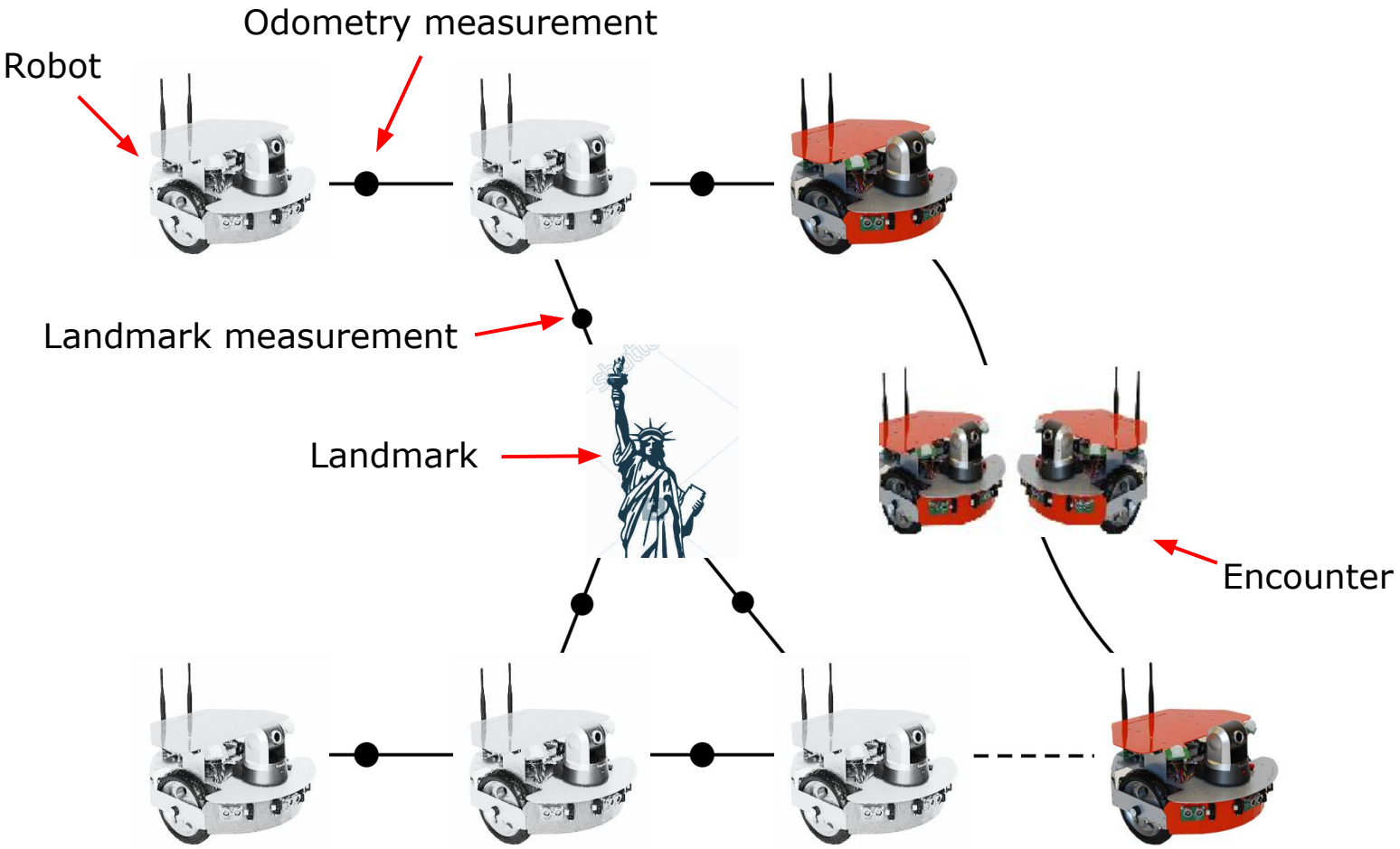
Multi-robot Mapping Problem (t=1)



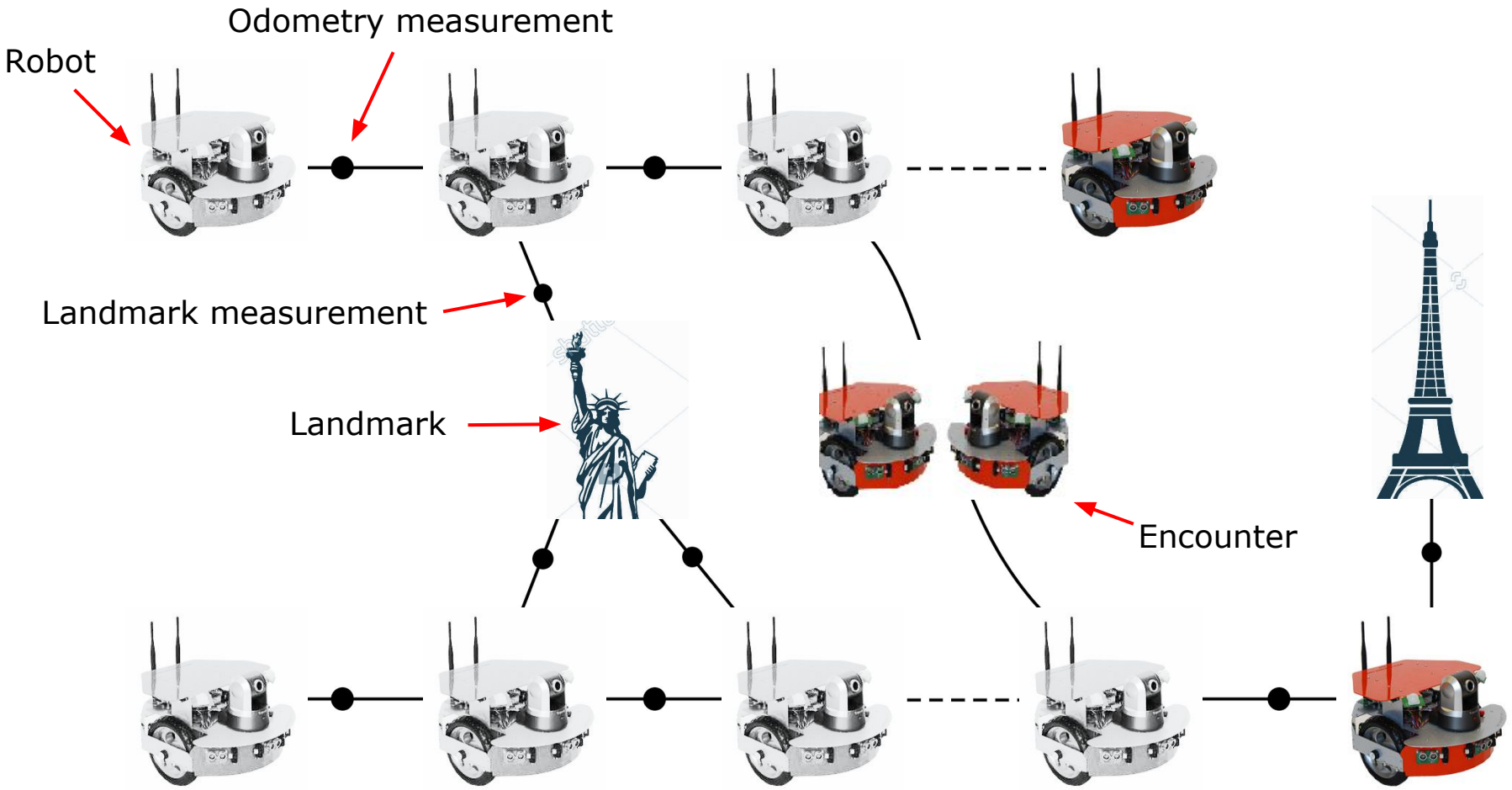
Multi-robot Mapping Problem (t=2)



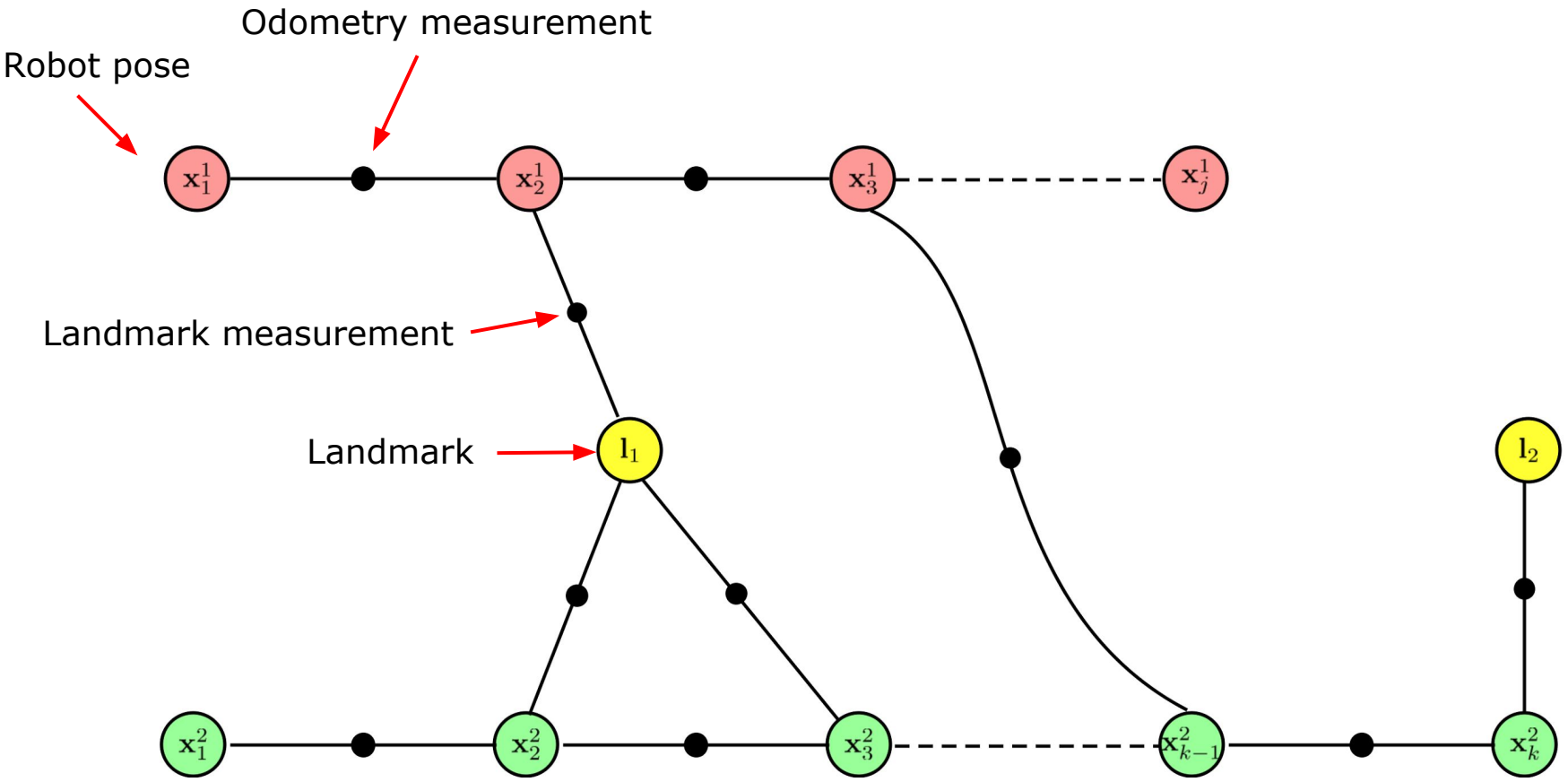
Multi-robot Mapping Problem (t=n-1)



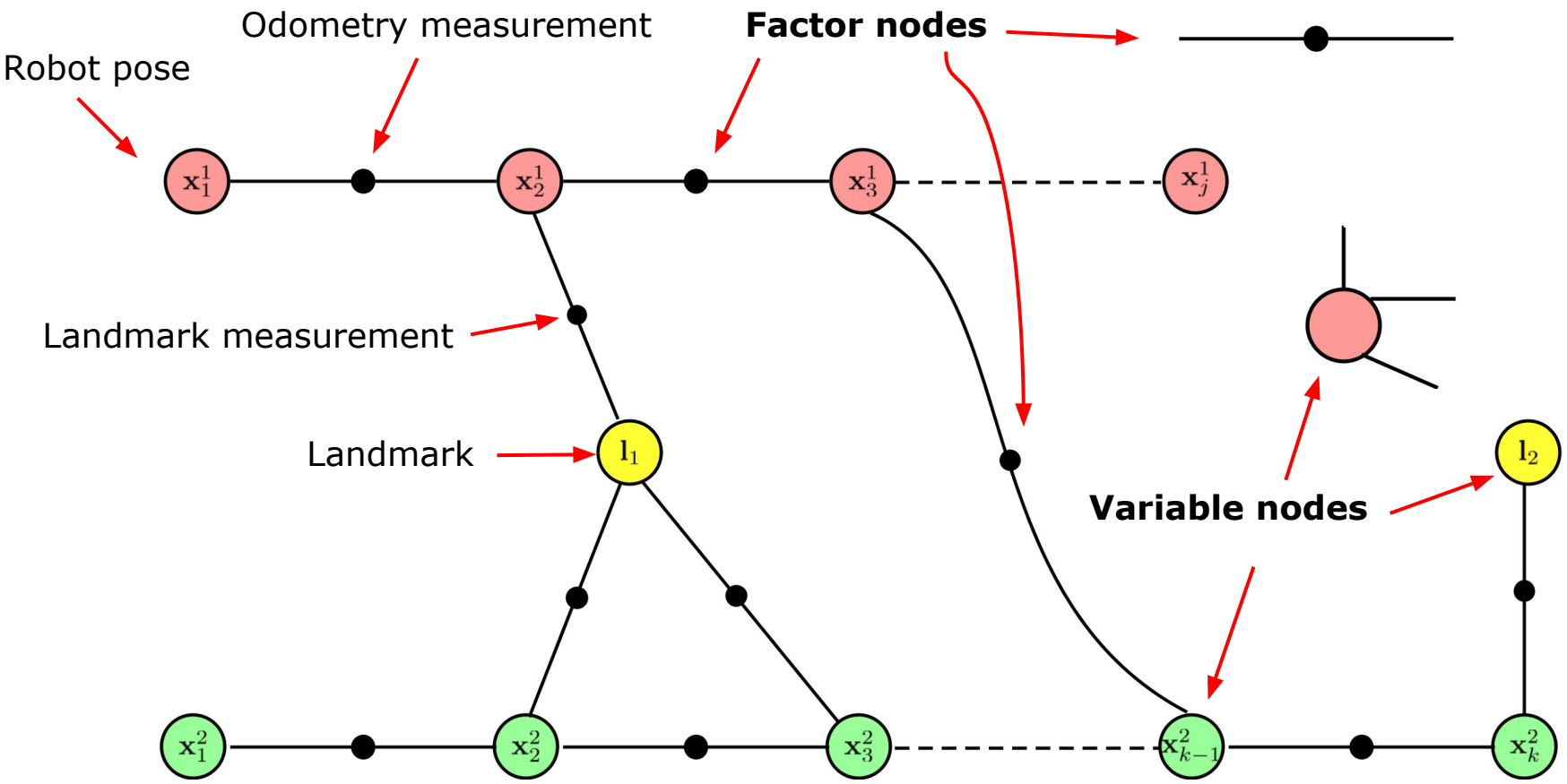
Multi-robot Mapping Problem (t=n)



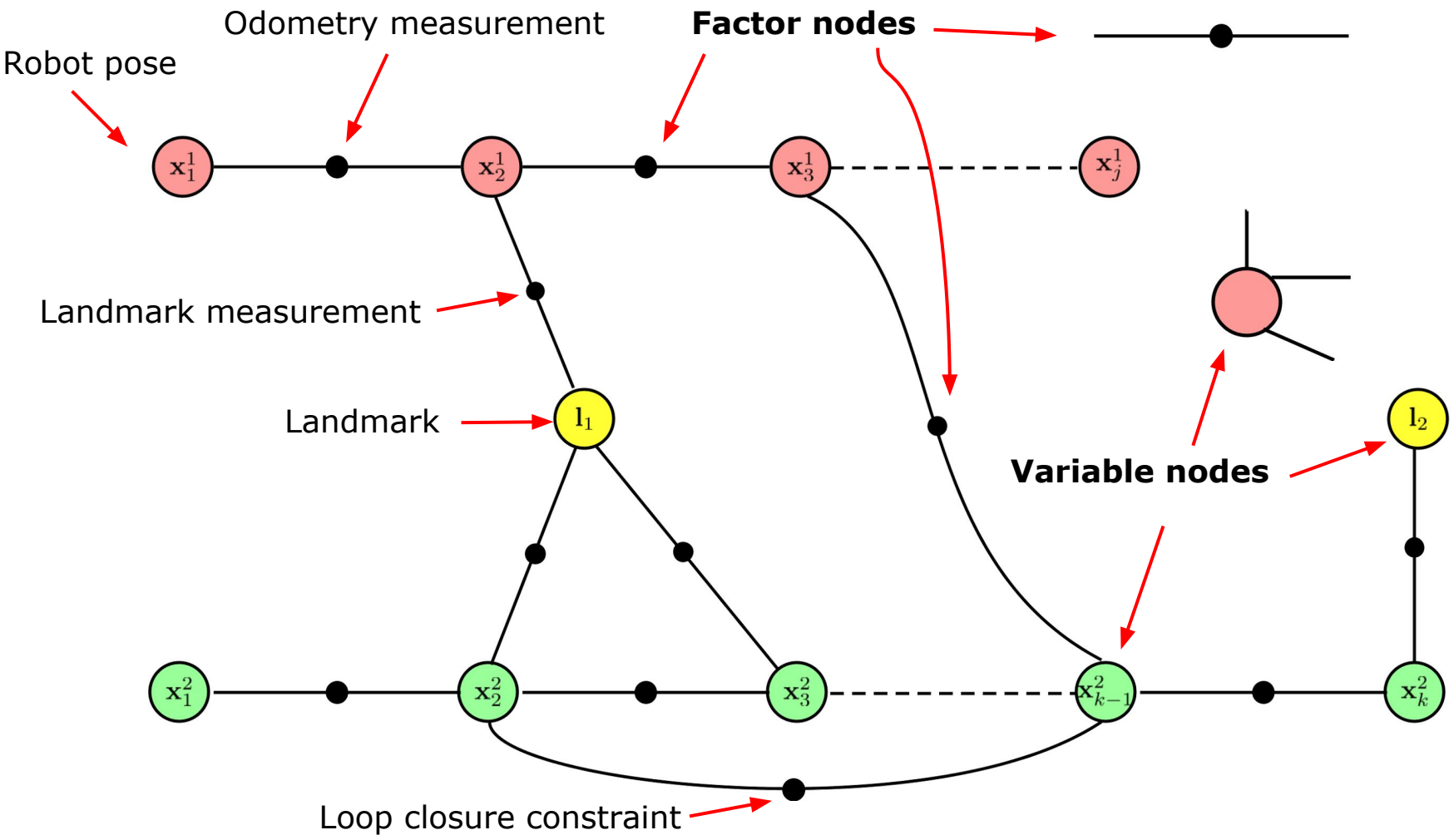
Factor Graph Representation [Lu & Milios 97, Folkesson 06]



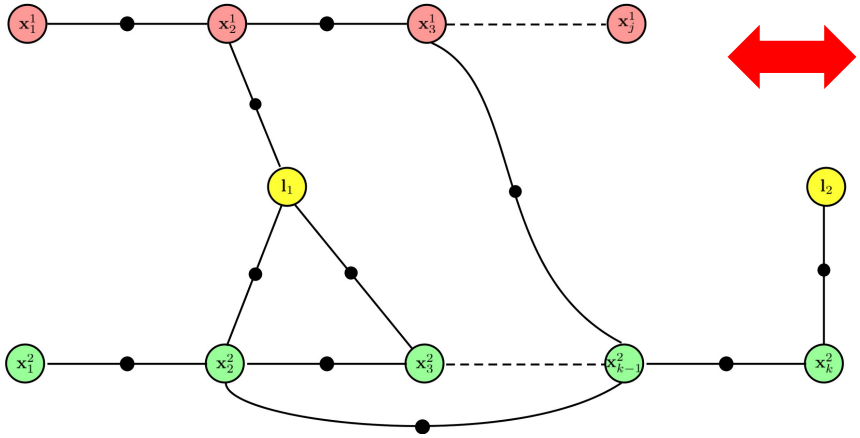
Factor Graph Representation [Lu & Milios 97, Folkesson 06]



Factor Graph Representation [Lu & Milios 97, Folkesson 06]

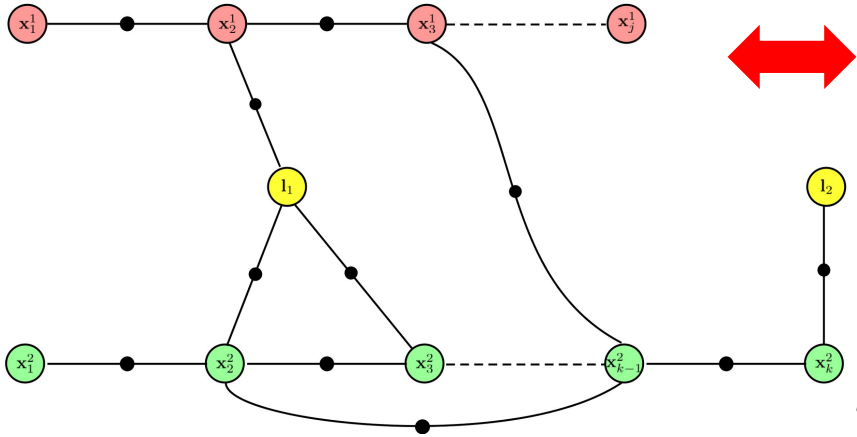


Nonlinear Least Squares



$$\arg\max_{\Theta_r} \prod_i p_i(\Theta_r)$$

Nonlinear Least Squares

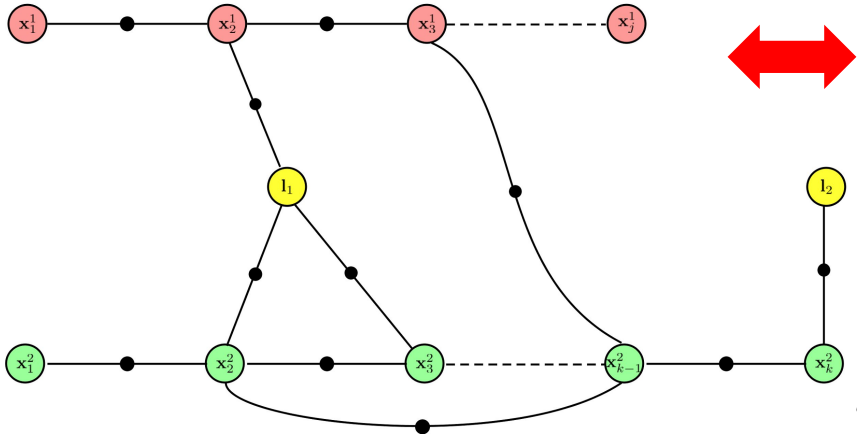


$$\arg\max_{\Theta_r} \prod_i p_i(\Theta_r)$$

Gaussian noise

$$\arg\min_{\Theta_r} \sum_i ||h_i(\Theta_r)||^2_{\Lambda_i}$$

Nonlinear Least Squares



$$\arg\max_{\Theta_r} \prod_i p_i(\Theta_r)$$

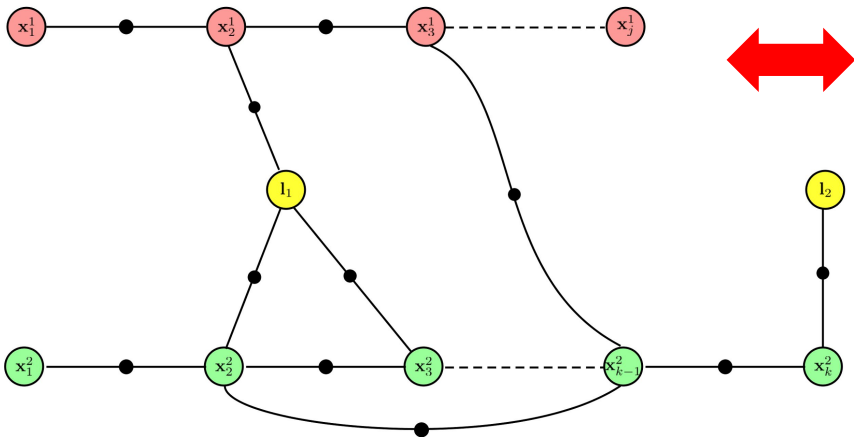
Gaussian noise

$$\arg\min_{\Theta_r} \sum_i ||h_i(\Theta_r)||^2_{\Lambda_i}$$

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} ||A_r\Theta_r - b_r||^2$$

Nonlinear Least Squares



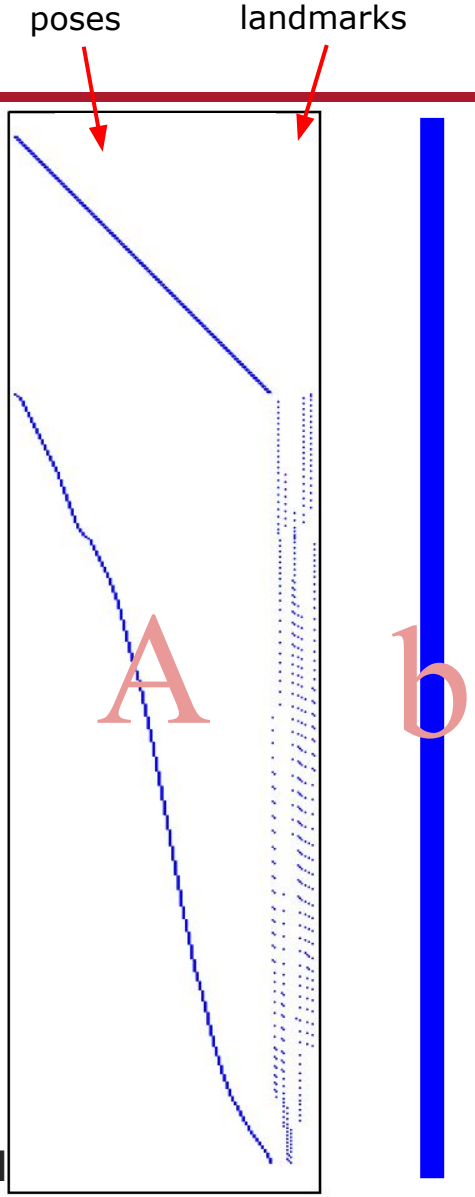
$$\arg\max_{\Theta_r} \prod_i p_i(\Theta_r)$$

Gaussian noise

$$\arg\min_{\Theta_r} \sum_i ||h_i(\Theta_r)||^2_{\Lambda_i}$$

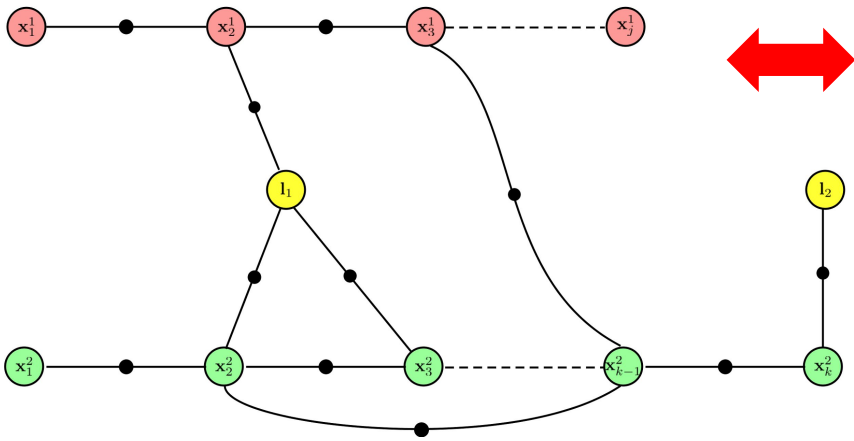
Repeatedly solve the linear system

$$\arg\min_{\Theta_r} ||A_r\Theta_r - b_r||^2$$



[Dellaert et al, 06]

Nonlinear Least Squares



$$\arg\max_{\Theta_r} \prod_i p_i(\Theta_r)$$

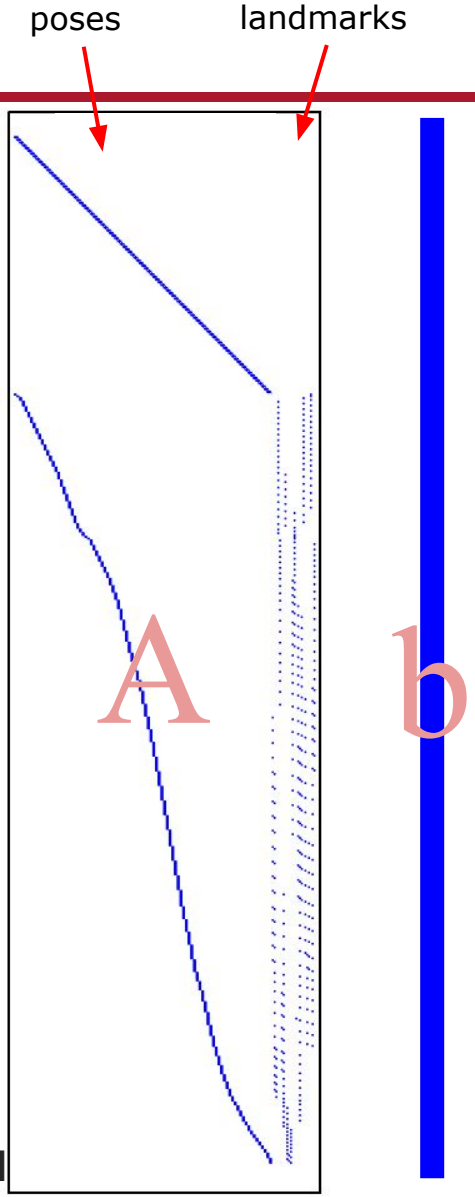
Gaussian noise

$$\arg\min_{\Theta_r} \sum_i ||h_i(\Theta_r)||^2_{\Lambda_i}$$

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} ||A_r\Theta_r - b_r||^2$$

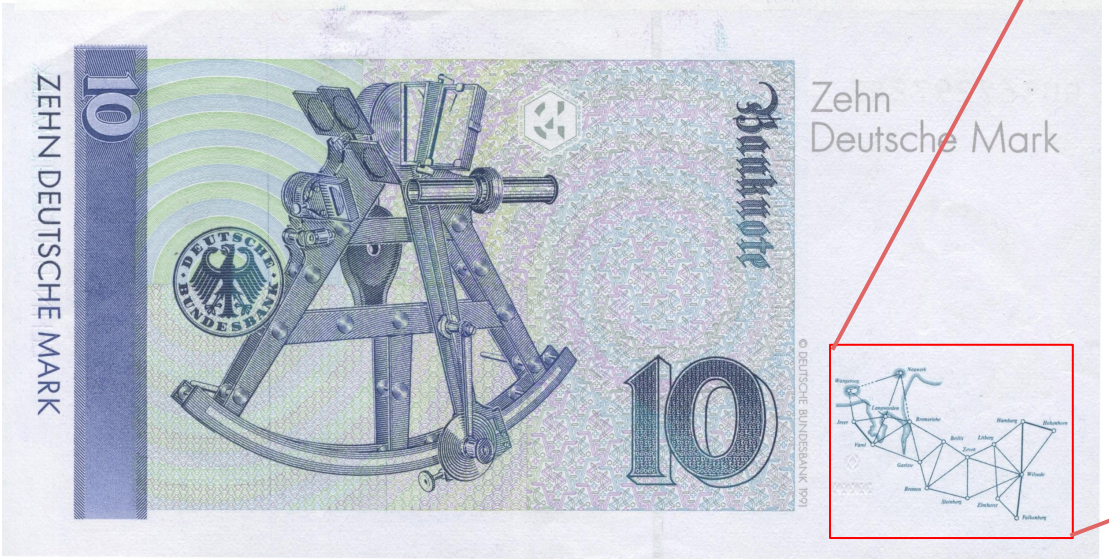
[Lu&Milios, 97, Eustice et al, 05, Dellaert&Kaess 06, Kaess 08]



[Dellaert et al, 06]

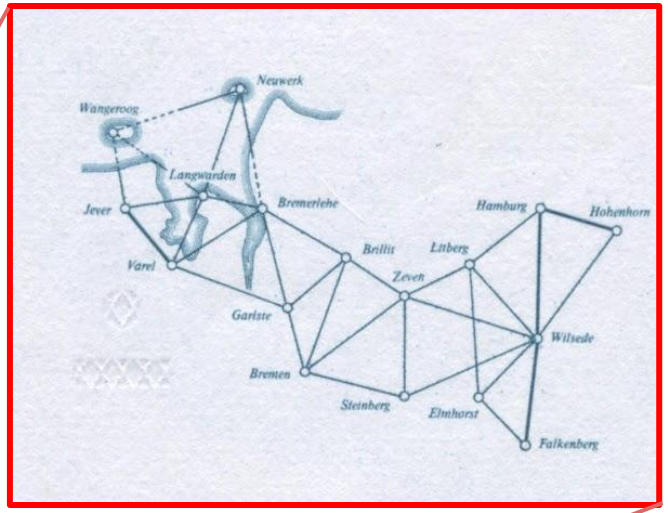
Mapping: History & Motivation

Mapping: History & Motivation

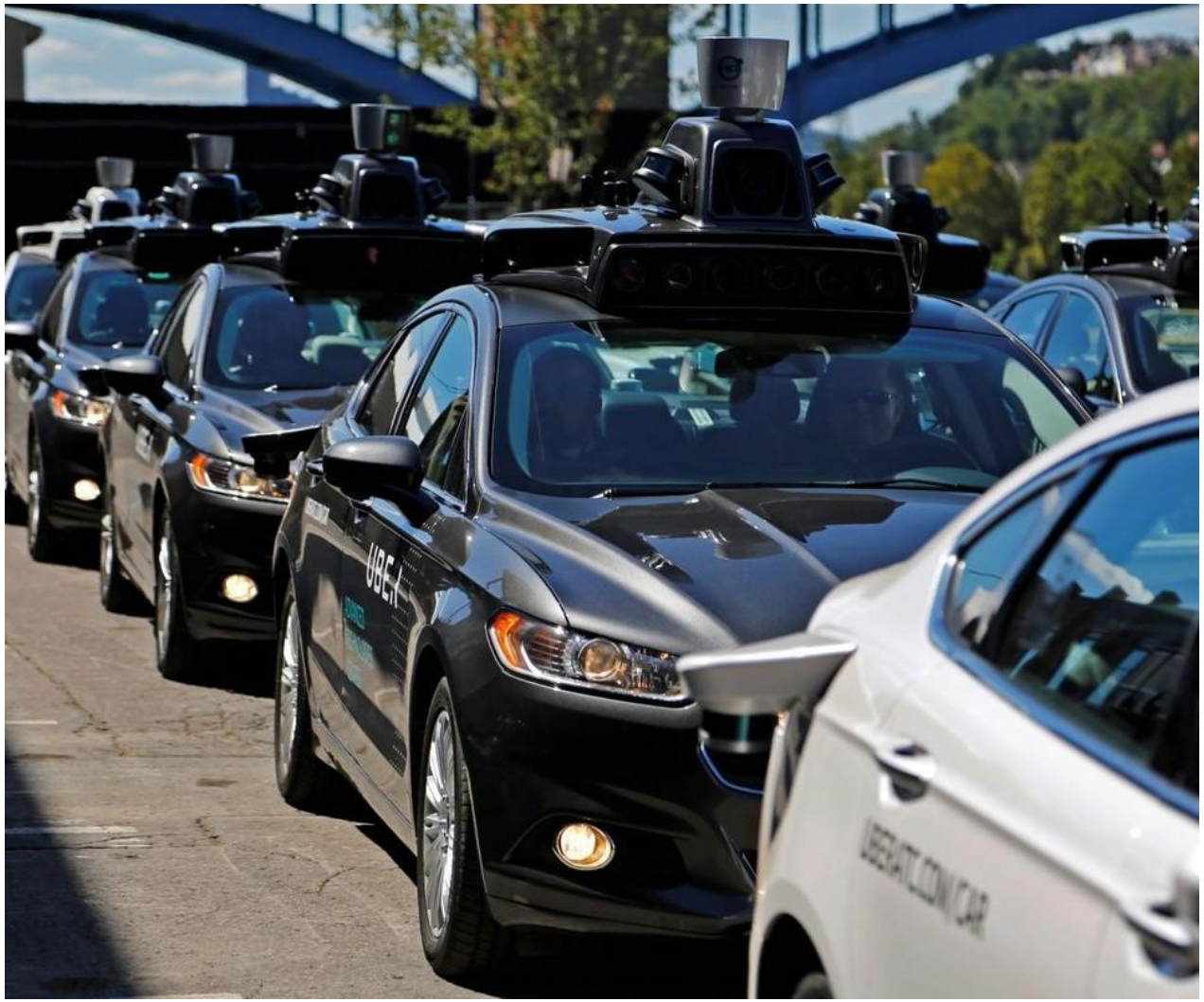


Carl Friedrich Gauss

1828: Triangulation of Hanover



Mapping: History & Motivation



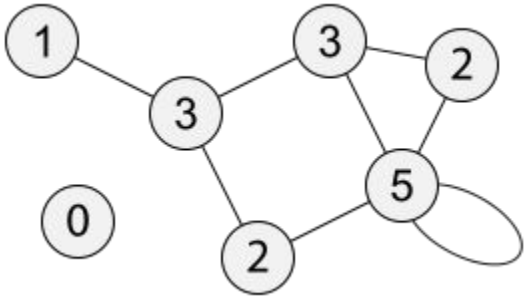
Efficient map sharing by UBER self-driving cars.

Congestion free traffic

Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

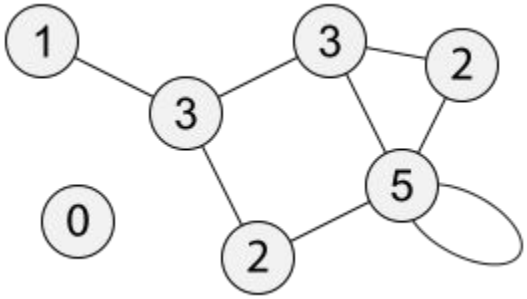
Graph Theory Glossary



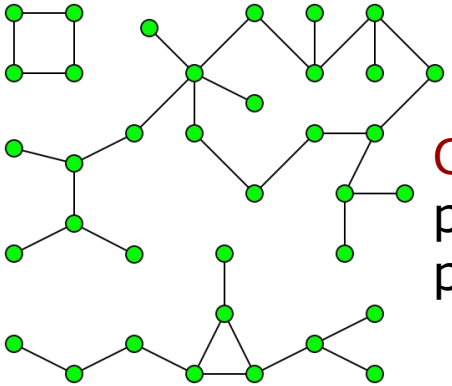
Degree: Number of edges incident to a vertex.

[Image source: Wikipedia]

Graph Theory Glossary



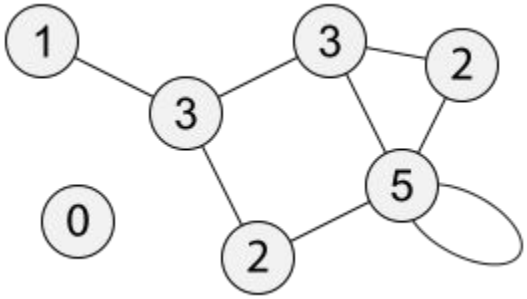
Degree: Number of edges incident to a vertex.



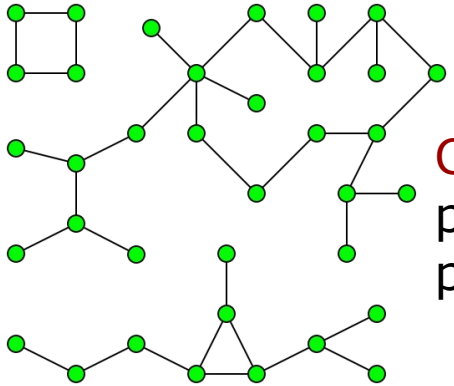
Connected component: A path exists between every pair of distinct vertices.

[Image source: Wikipedia]

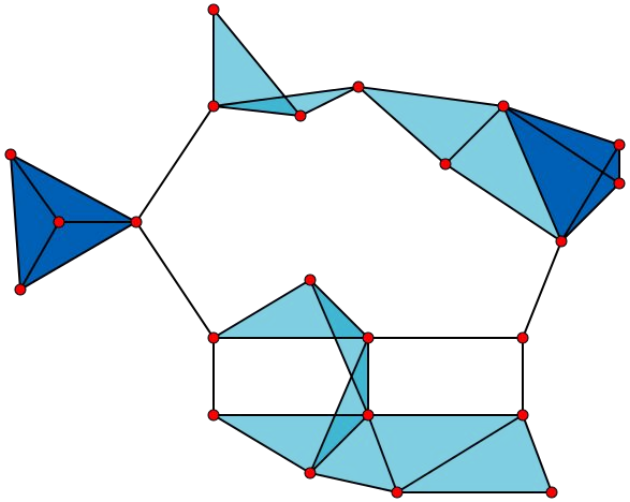
Graph Theory Glossary



Degree: Number of edges incident to a vertex.



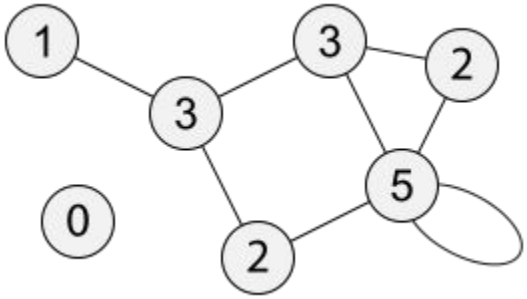
Connected component: A path exists between every pair of distinct vertices.



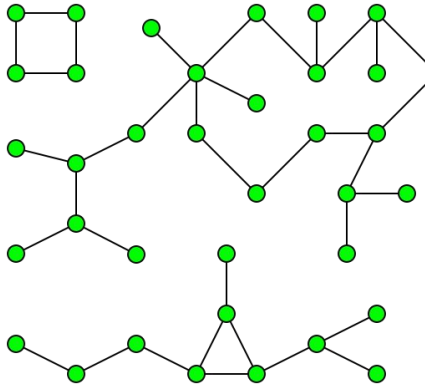
Clique: Fully connected (complete) subgraph.

[Image source: Wikipedia]

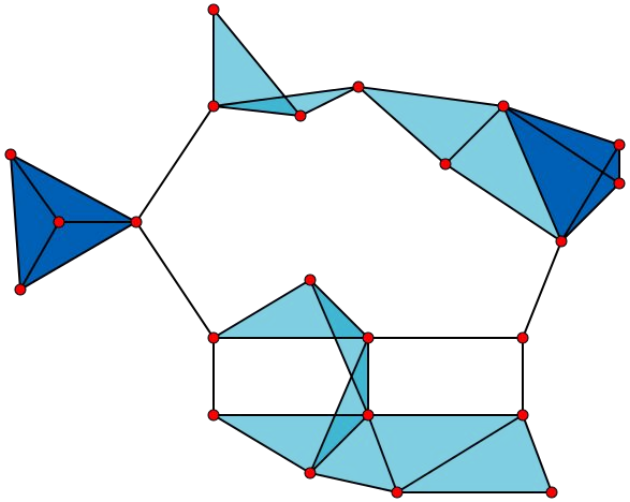
Graph Theory Glossary



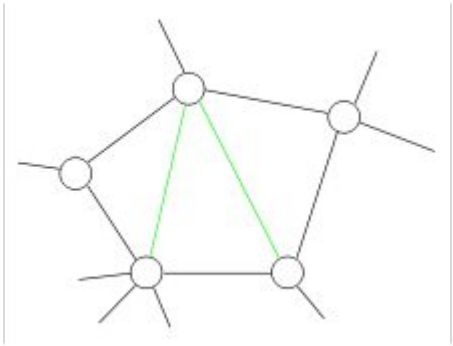
Degree: Number of edges incident to a vertex.



Connected component: A path exists between every pair of distinct vertices.



Clique: Fully connected (complete) subgraph.



Chordal graph: Cycles of 4 or more vertices have a chord.

[Image source: Wikipedia]

Incremental Smoothing and Mapping [Kaess 08]

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} ||A_r \Theta_r - b_r||^2$$

Incremental Smoothing and Mapping [Kaess 08]

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} \left|\left|A_r\Theta_r - b_r\right|\right|^2$$

We use QR decomposition:

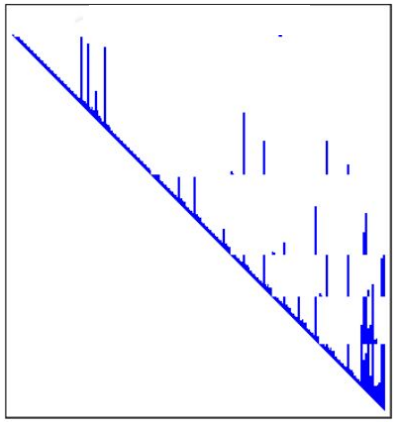
$$\left|\left|A_r\Theta_r - b_r\right|\right|^2 = \left|\left|Q_r^\top Q_r \begin{bmatrix} R_r \\ 0 \end{bmatrix} \Theta_r - Q_r^\top b_r\right|\right|^2$$

$$= \left|\left|R\Theta_r - \mathbf{d}^r\right|\right|^2 + \left|\left|\mathbf{e}^r\right|\right|^2$$

Incremental Smoothing and Mapping [Kaess 08]

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} ||A_r\Theta_r - b_r||^2$$



Square-root factor "R"

We use QR decomposition:

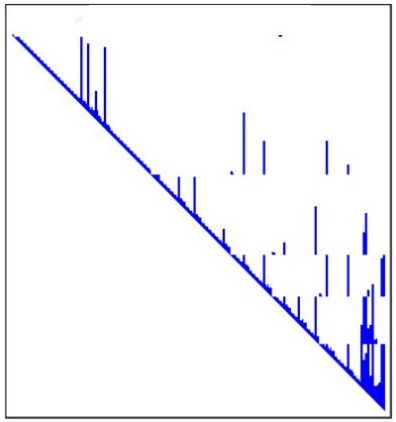
$$||A_r\Theta_r - b_r||^2 = \left|\left|Q_r^\top Q_r \begin{bmatrix} R_r \\ 0 \end{bmatrix} \Theta_r - Q_r^\top b_r\right|\right|^2$$

$$= ||R\Theta_r - \mathbf{d}^r||^2 + ||\mathbf{e}^r||^2$$

Incremental Smoothing and Mapping [Kaess 08]

Repeatedly solve the linear system

$$\arg\min_{\Theta_r} \left|\left|A_r\Theta_r - b_r\right|\right|^2$$



Square-root factor "R"

We use QR decomposition:

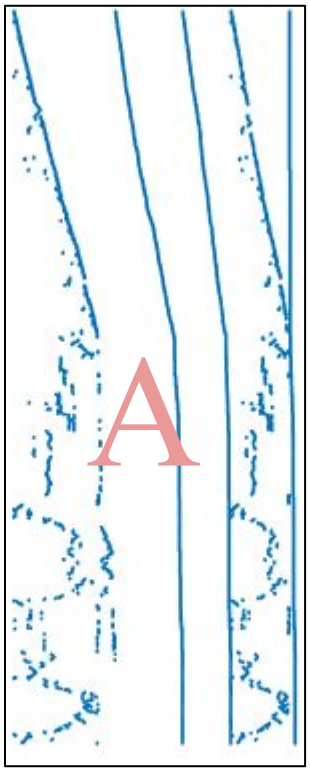
$$\left|\left|A_r\Theta_r - b_r\right|\right|^2 = \left|\left|Q_r^\top Q_r \begin{bmatrix} R_r \\ 0 \end{bmatrix} \Theta_r - Q_r^\top b_r\right|\right|^2$$

$$= \left|\left|R\Theta_r - \mathbf{d}^r\right|\right|^2 + \left|\left|\mathbf{e}^r\right|\right|^2$$

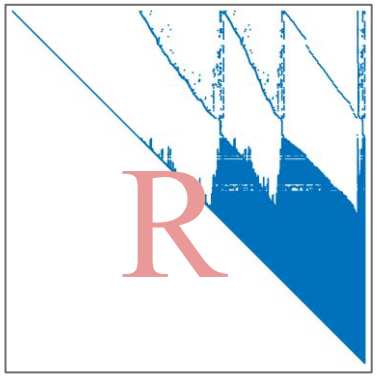
Why QR?

- Incremental update of R matrix using Givens rotation.
- Need not calculate the dense A^TA.
- Numerically stable.

Variable Ordering: Maintaining Sparsity

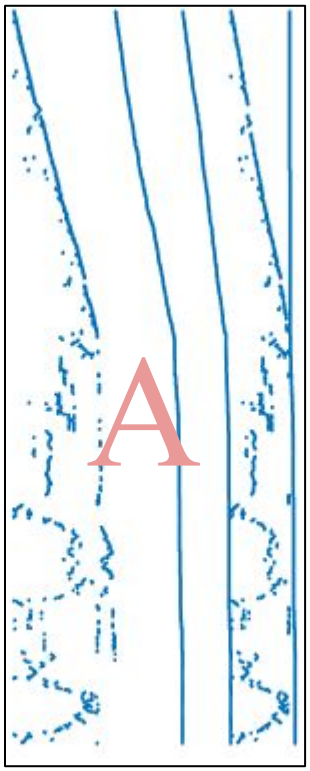


factorize

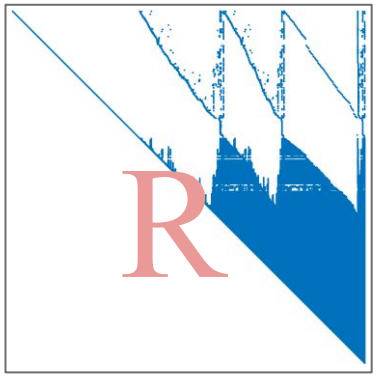


Default ordering as stored in memory (poses, landmarks)

Variable Ordering: Maintaining Sparsity



factorize



Default ordering as stored in memory (poses, landmarks)

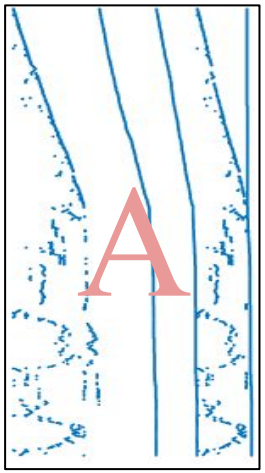
Variable ordering (permute)

factorize

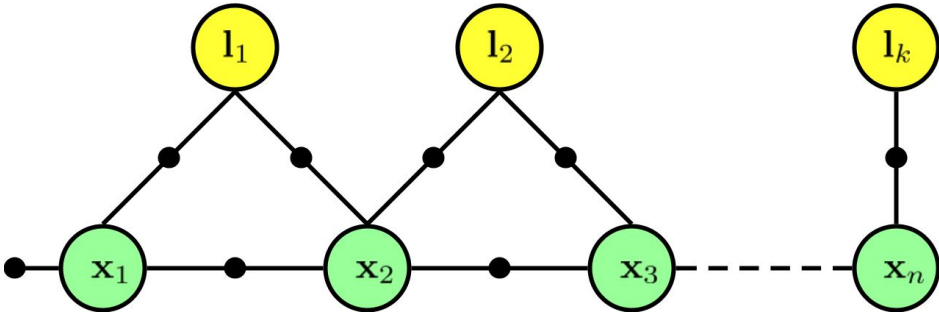
COLAMD ordering
[Davis 04]

Optimal ordering is NP complete
[Arnborg et al, 87]

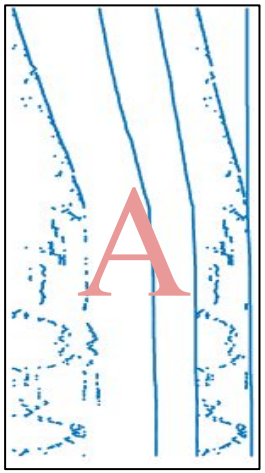
Matrix vs. Graph

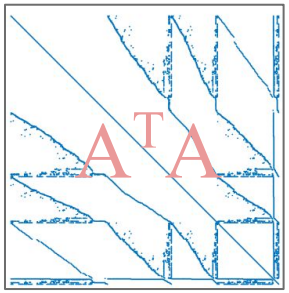


Factor Graph

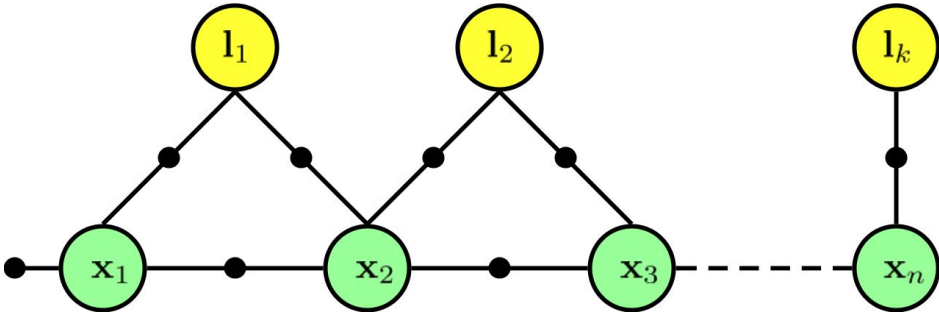


Matrix vs. Graph

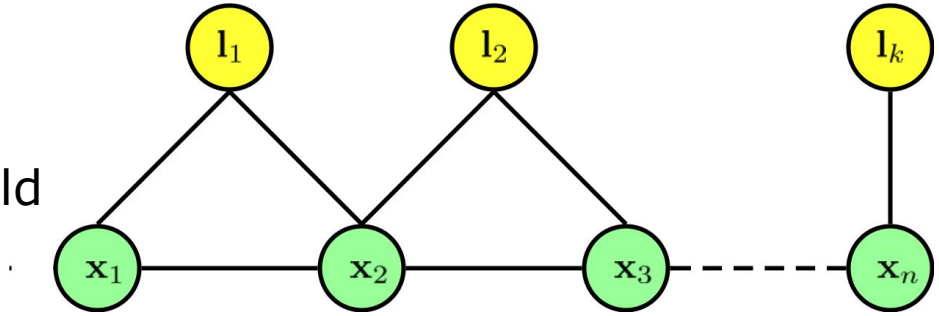




Factor Graph

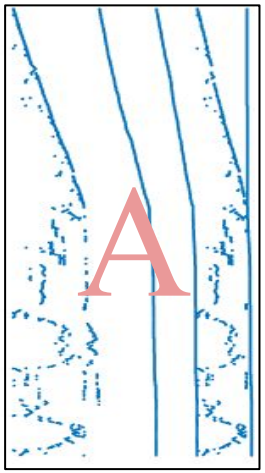


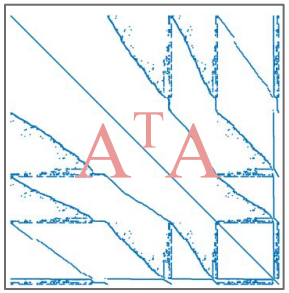
Markov Random Field



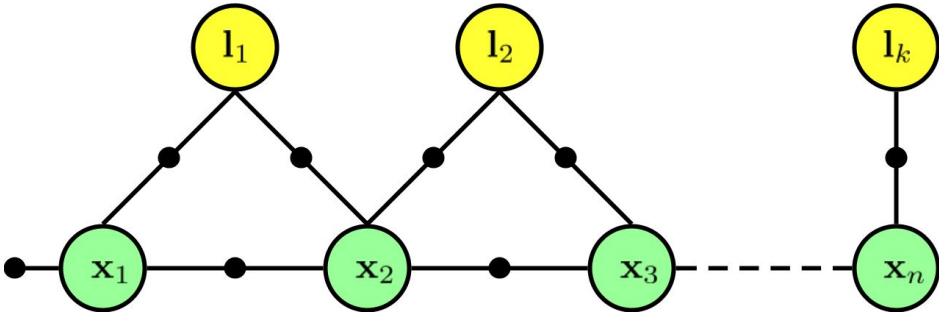
[Dellaert et al, 06]

Matrix vs. Graph

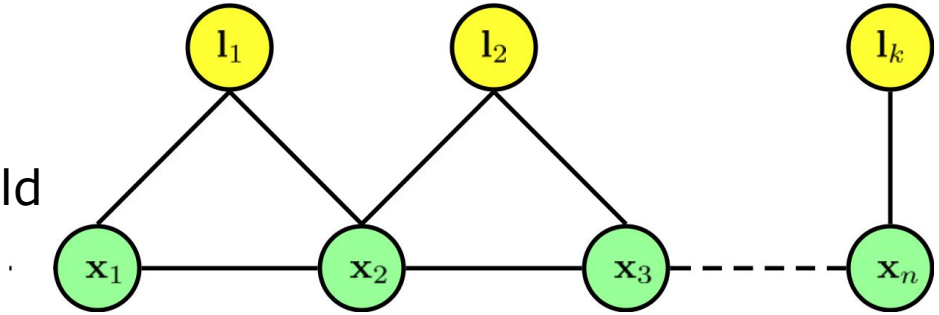




Factor Graph

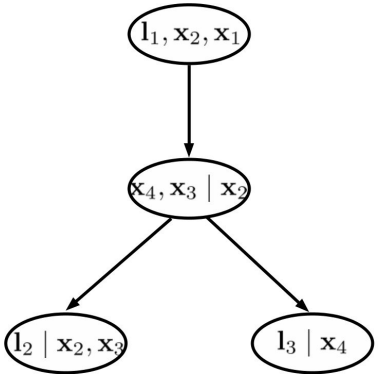


Markov Random Field



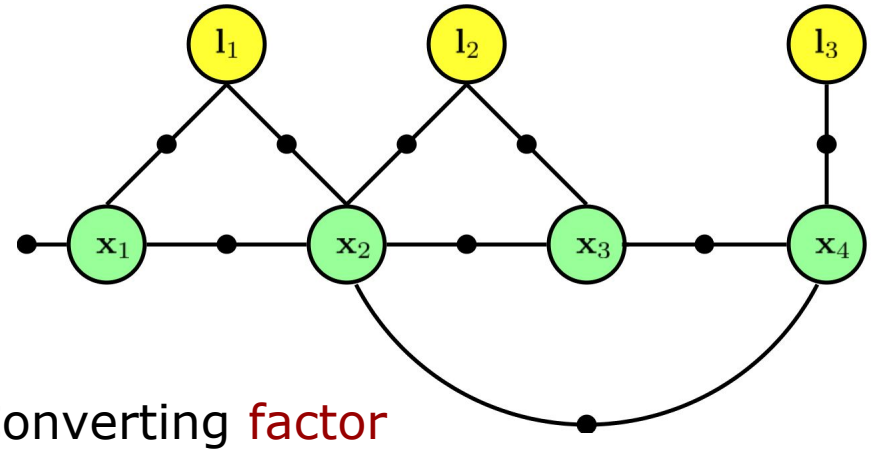
[Dellaert et al, 06]

Bayes Tree



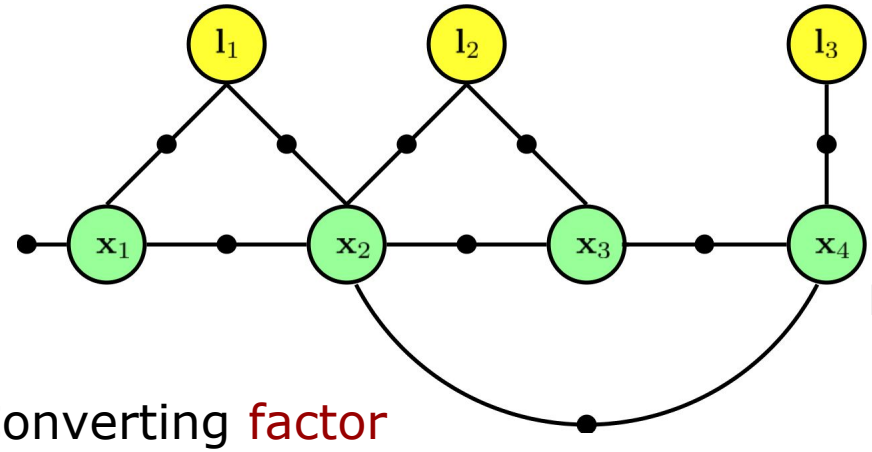
[Kaess et al, 10]

Factor Graph -> Bayes Network -> Bayes Tree



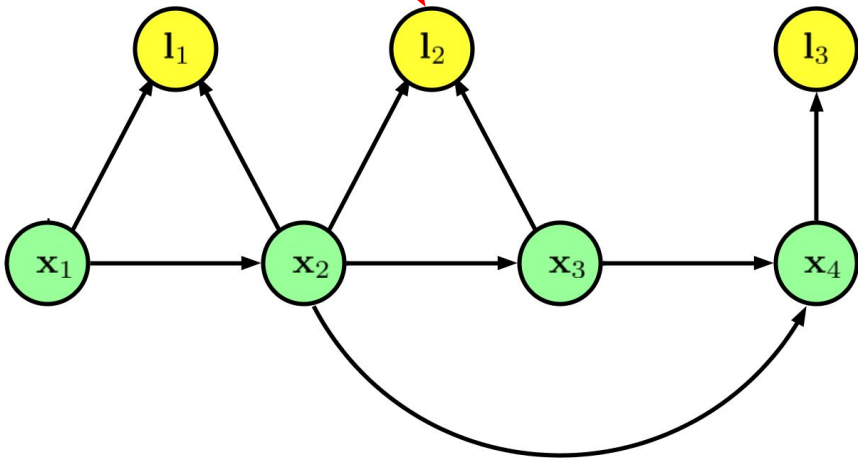
Converting factor graph to Bayes tree is same as QR factorizing A to get R

Factor Graph -> Bayes Network -> Bayes Tree



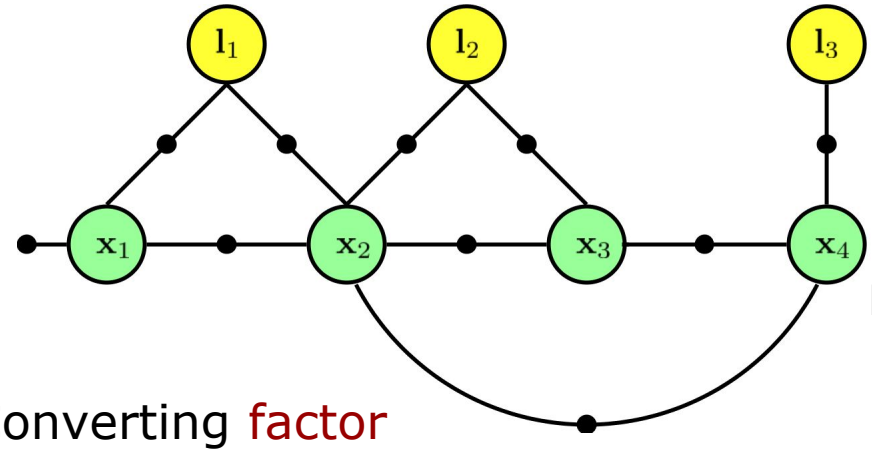
$$O = [l_3, l_2, l_1, x_4, x_3, x_2, x_1]$$

[Heggernes et al, 96]



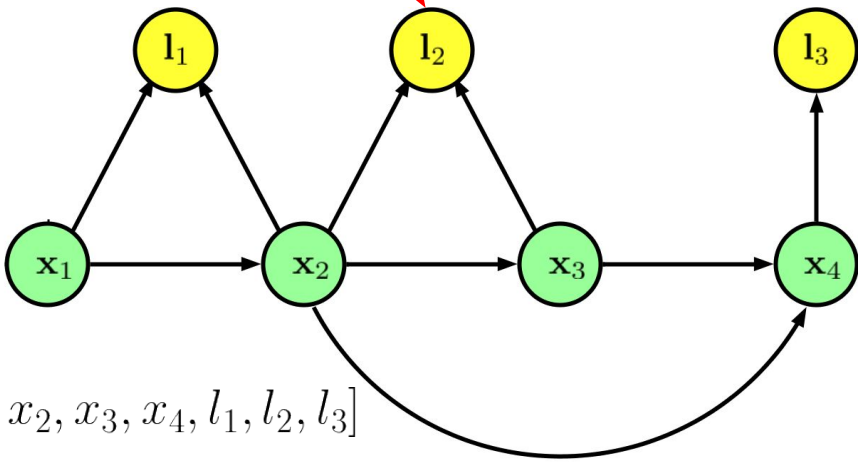
Converting factor graph to Bayes tree is same as QR factorizing A to get R

Factor Graph -> Bayes Network -> Bayes Tree



$$O = [l_3, l_2, l_1, x_4, x_3, x_2, x_1]$$

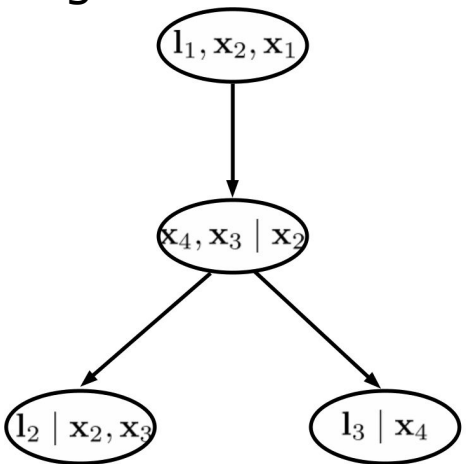
[Heggernes et al, 96]



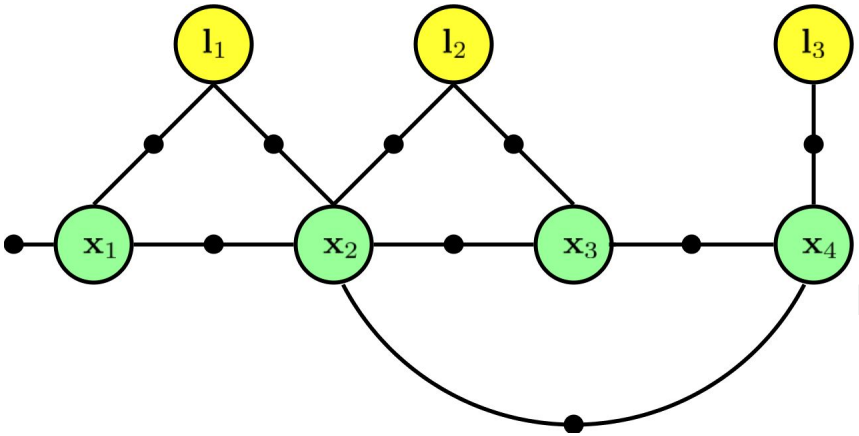
$$O^{reverse} = [x_1, x_2, x_3, x_4, l_1, l_2, l_3]$$

[Tarjan&Yannakakis, 84]
[Kaess, 10]

Converting factor graph to Bayes tree is same as QR factorizing A to get R

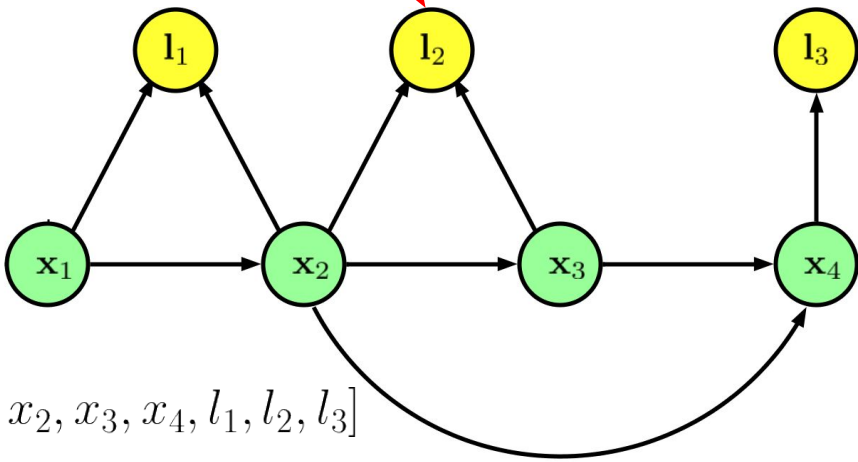


Bayes Tree: Incremental Variable Addition [Kaess 12]



$$O = [l_3, l_2, l_1, x_4, x_3, x_2, x_1]$$

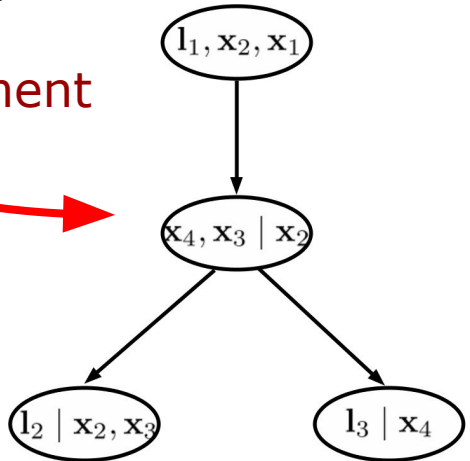
[Heggernes et al, 96]



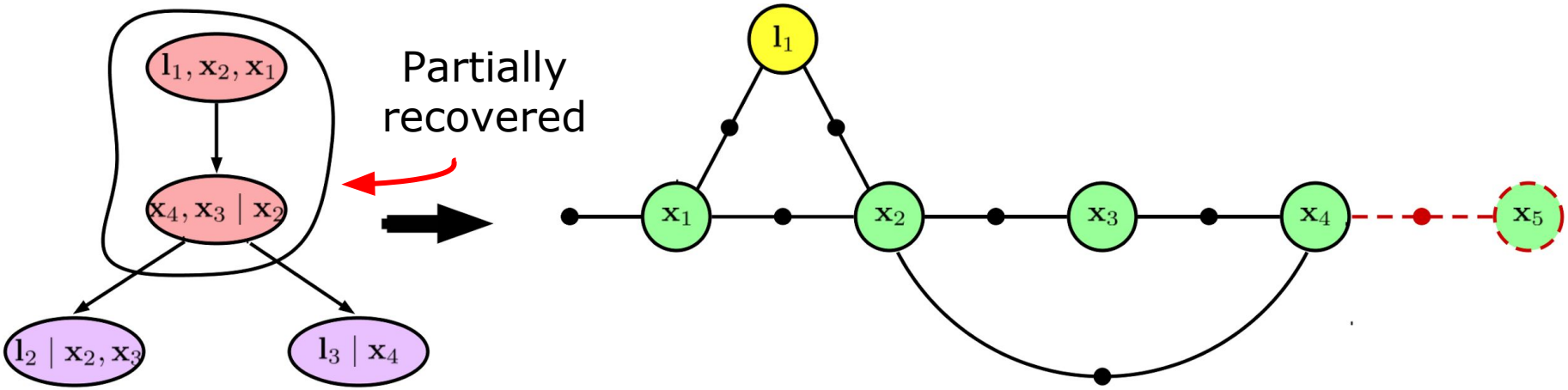
$$O^{reverse} = [x_1, x_2, x_3, x_4, l_1, l_2, l_3]$$

[Tarjan&Yannakakis, 84]
[Kaess, 10]

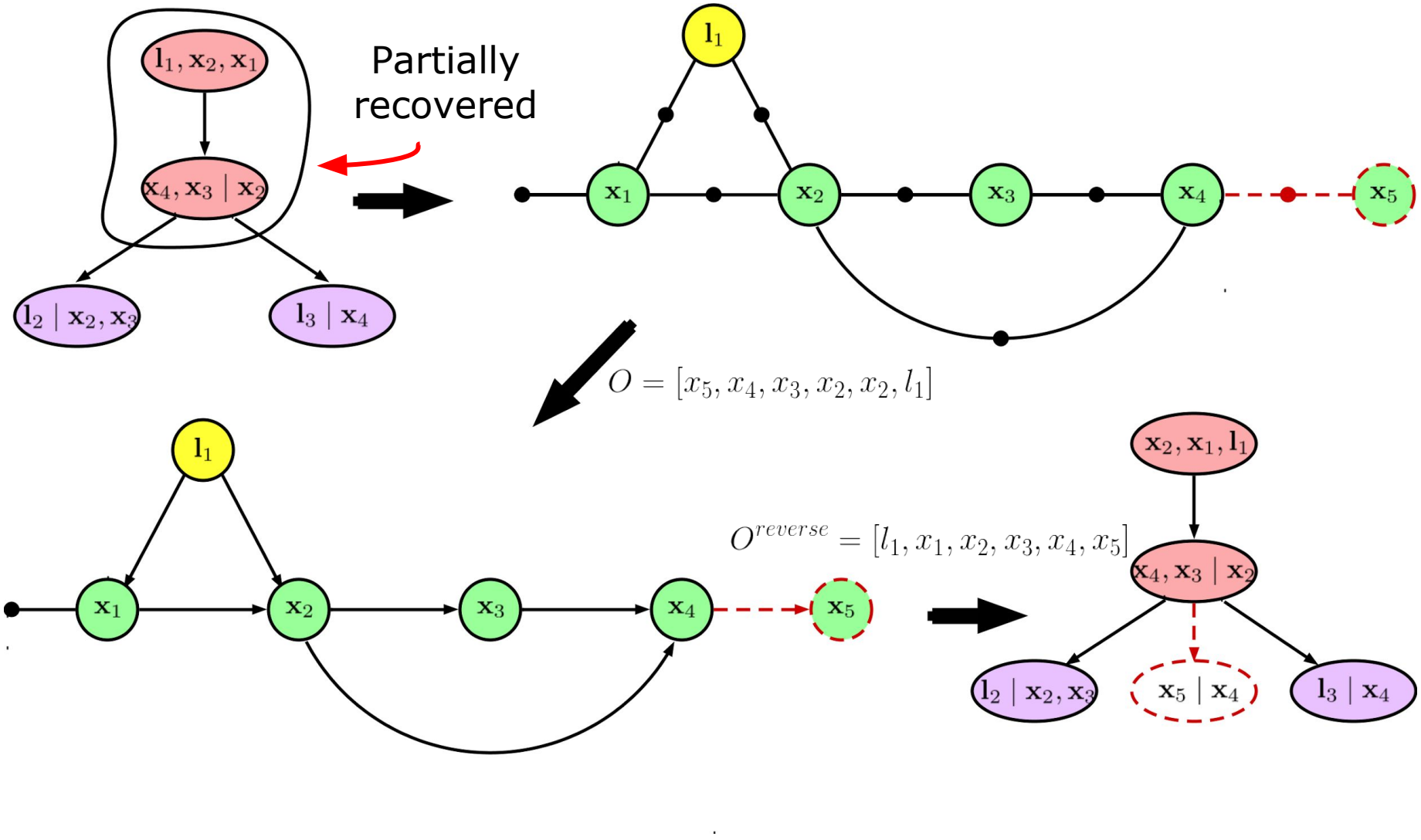
Add a new variable x_5 connecting the node x_4 by an odometry measurement



Bayes Tree: Incremental Update [Kaess 12]



Bayes Tree: Incremental Update [Kaess 12]



Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

Related Work

Smoothing formulation
with Least-Squares
[Lu&Milios, 97]

Related Work

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**

- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**

- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory

- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Optimal ordering is NP-complete **[Arnborg et al, 87]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Multiple robots

Role of Variable Ordering in SLAM **[Agarwal, 12]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Optimal ordering is NP-complete **[Arnborg et al, 87]**

Approximate minimum degree **[Amestoy, 04]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Multiple robots

Role of Variable Ordering in SLAM **[Agarwal, 12]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Optimal ordering is NP-complete **[Arnborg et al, 87]**

Approximate minimum degree **[Amestoy, 04]**

CCOLAMD **[Chen, 08]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

Multiple robots

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Recent Algebraic Graph Theory
- HUND **[Grigori, 10]**
- Exact graph partitioning **[Hager, 14]**

Optimal ordering is NP-complete **[Arnborg et al, 87]**

Approximate minimum degree **[Amestoy, 04]**

CCOLAMD **[Chen, 08]**

Related Work

Improvise Information matrix decomposition

Smoothing formulation with Least-Squares **[Lu&Milios, 97]**

Smoothing based SLAM

Square root SAM **[Dellaert, 05]**
- ISAM **[Kaess, 08]**
- ISAM2 **[Kaess, 12]**

- Conjugate gradient **[Konolige, 04]**
- Gradient descent **[Folkesson et al, 04]**
- Relaxation **[Thrun, 05]**
- Multi-level relaxation **[Frese et al, 05]**

Multiple robots

Role of Variable Ordering in SLAM **[Agarwal, 12]**

- C-SAM **[Andersson, 08]**
- Cooperative ISAM **[Kim, 10]**

- Self correcting graph SLAM **[Folkesson, 04]**
- Treemap SLAM **[Frese, 06]**

Uses Algebraic Graph Theory
- COLAMD **[Davis, 04]**
- Nested dissection **[George, 73]**
- Householder QR **[Kaess, 08]**
- Givens rotations **[Golub, 12]**

Ordering symmetric & unsymmetric matrices **[Markowitz, 57]**

Ordering on the graph **[Rose, 72]**

Nested Dissection **[George, 73]**

Recent Algebraic Graph Theory
- HUND **[Grigori, 10]**
- Exact graph partitioning **[Hager, 14]**

Optimal ordering is NP-complete **[Arnborg et al, 87]**

Approximate minimum degree **[Amestoy, 04]**

CCOLAMD **[Chen, 08]**

Factor Graph Fusion

- Consider
  - Two bipartite graphs: $\mathcal{G}_i = (\mathcal{C}_i, \Theta_i, \mathcal{E}_i); \quad i = 1, 2$
    - $\mathcal{C}$ represents variable nodes
    - $\mathcal{E}$ represents the edges
    - Θ represents the factor nodes.

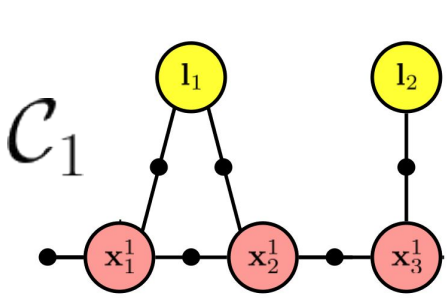
Factor Graph Fusion

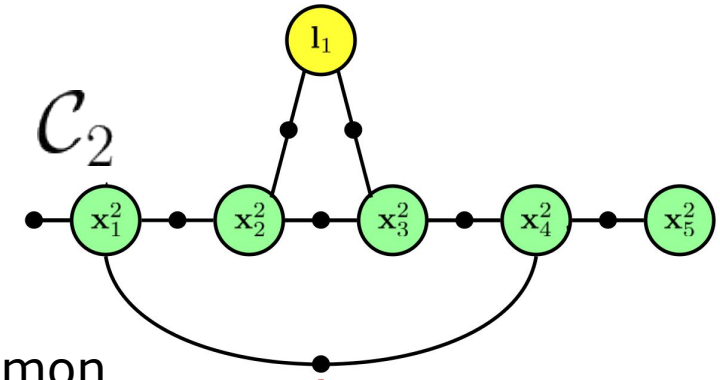
- Consider
  - Two bipartite graphs: $\mathcal{G}_i = (\mathcal{C}_i, \Theta_i, \mathcal{E}_i); \quad i = 1, 2$
    - $\mathcal{C}$ represents variable nodes
    - $\mathcal{E}$ represents the edges
    - Θ represents the factor nodes.
  - Corresponding Bayes tree function: $\mathcal{B}_i(\mathcal{C}_i); \quad i = 1, 2$
    - Returns the set of ancestor variables in the tree for a given variable $\mathcal{C}_i$.

Factor Graph Fusion

- Consider
  - Two bipartite graphs: $\mathcal{G}_i = (\mathcal{C}_i, \Theta_i, \mathcal{E}_i); \quad i = 1, 2$
    - $\mathcal{C}$ represents variable nodes
    - $\mathcal{E}$ represents the edges
    - Θ represents the factor nodes.
  - Corresponding Bayes tree function: $\mathcal{B}_i(\mathcal{C}_i); \quad i = 1, 2$
    - Returns the set of ancestor variables in the tree for a given variable $\mathcal{C}_i$.
- Graph merge condition: $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$

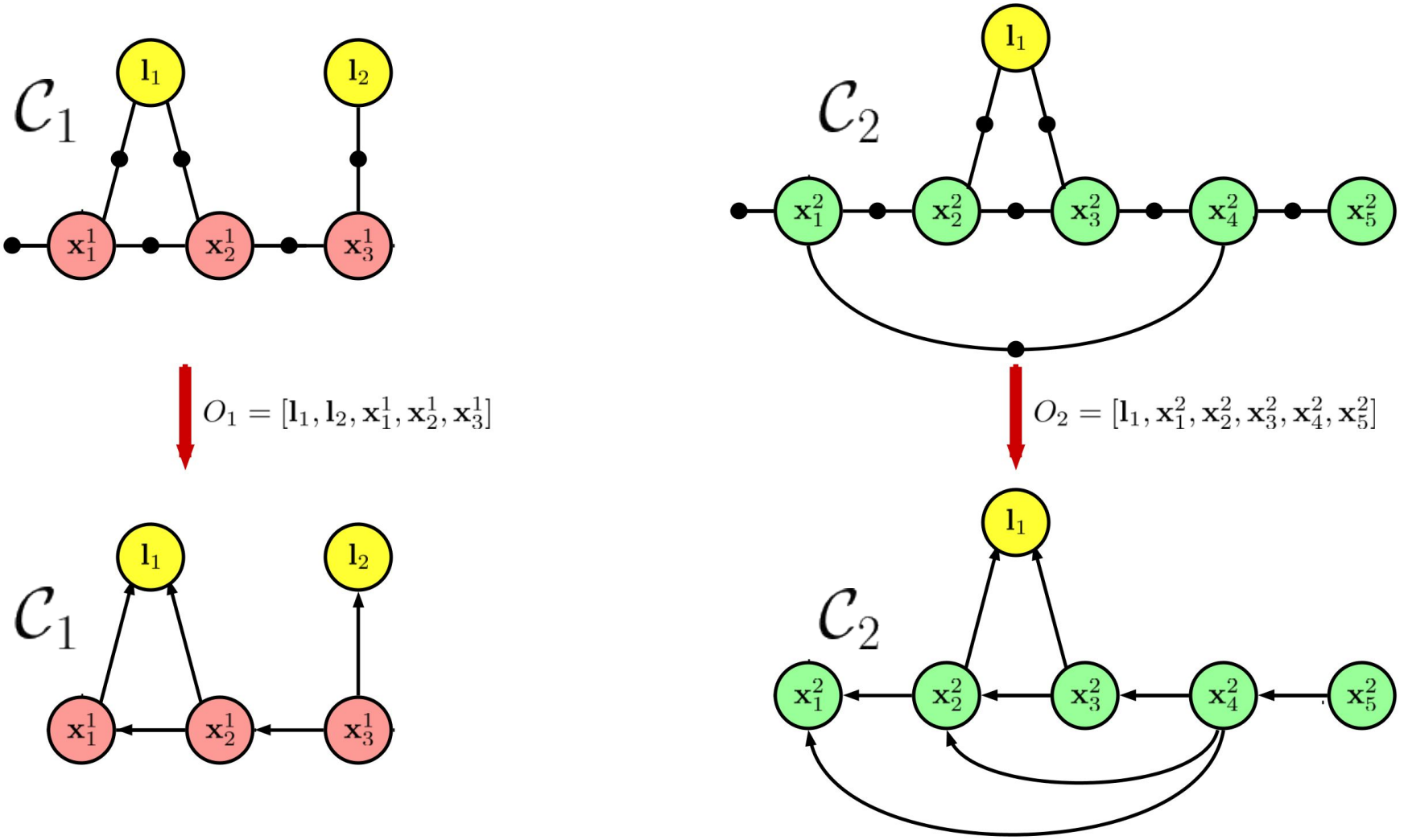
Example:



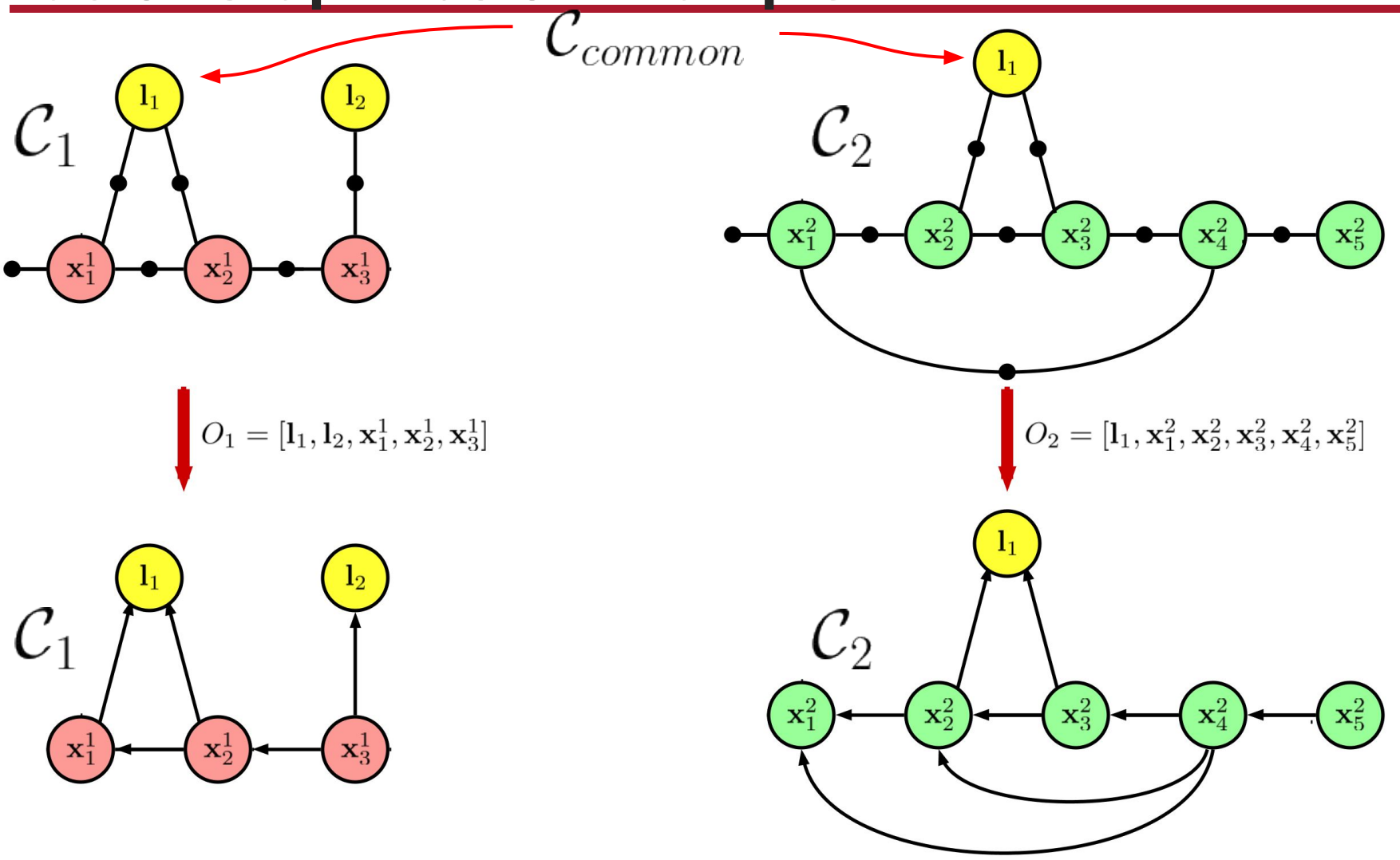


l_1 is the common variable

Factor Graph Fusion Example

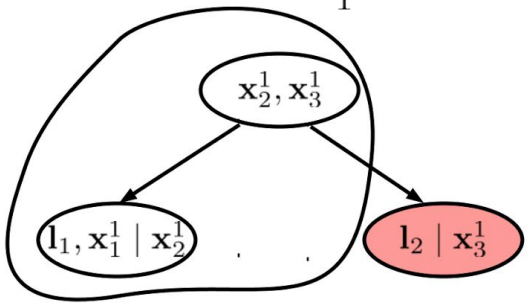


Factor Graph Fusion Example

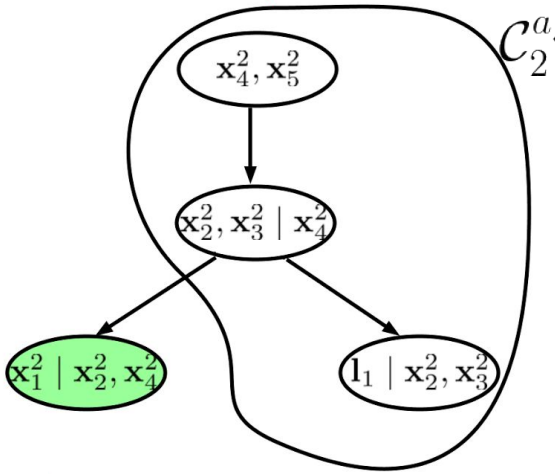


Factor Graph Fusion Example

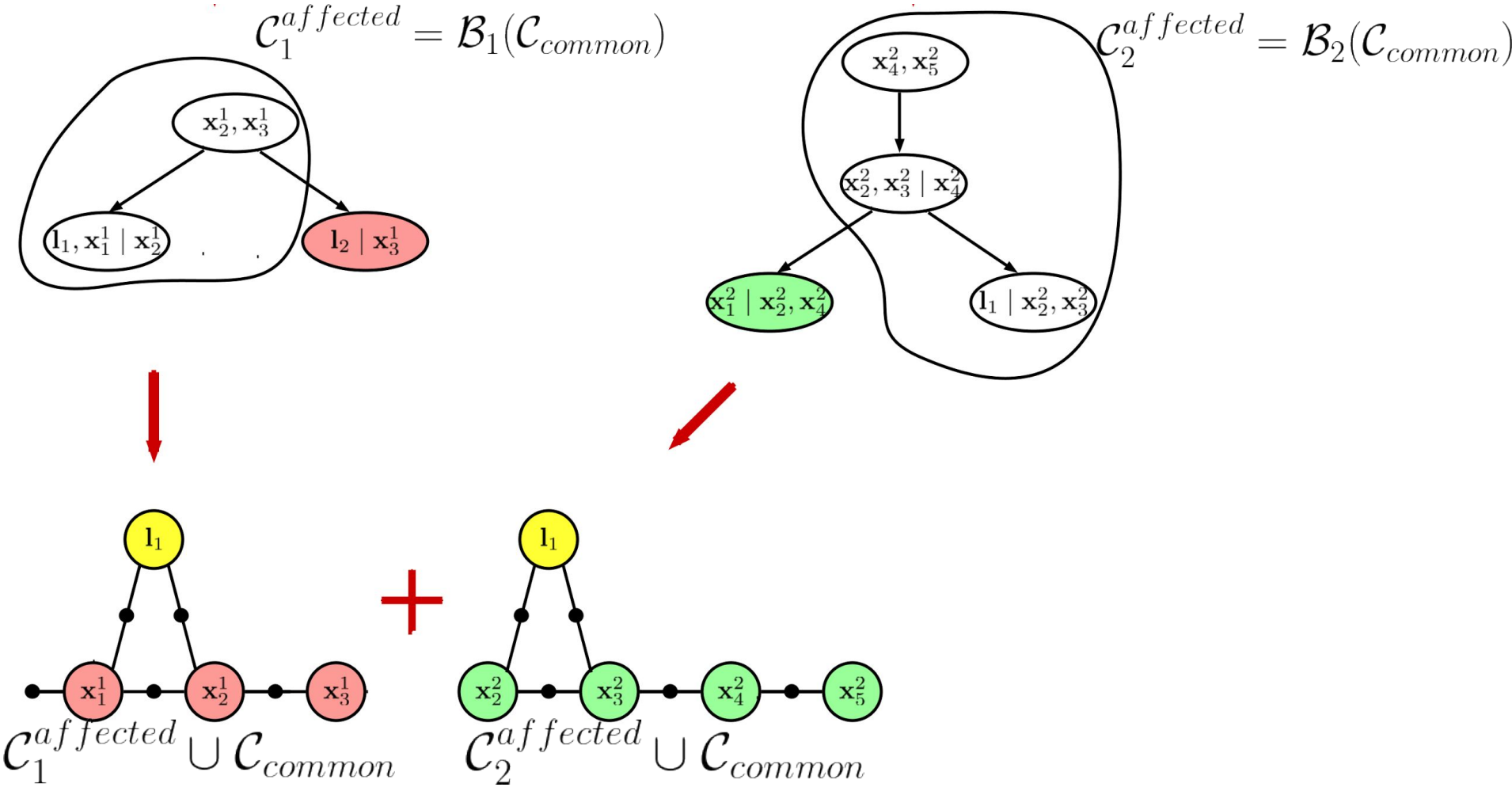
$$\mathcal{C}_1^{affected} = \mathcal{B}_1(\mathcal{C}_{common})$$



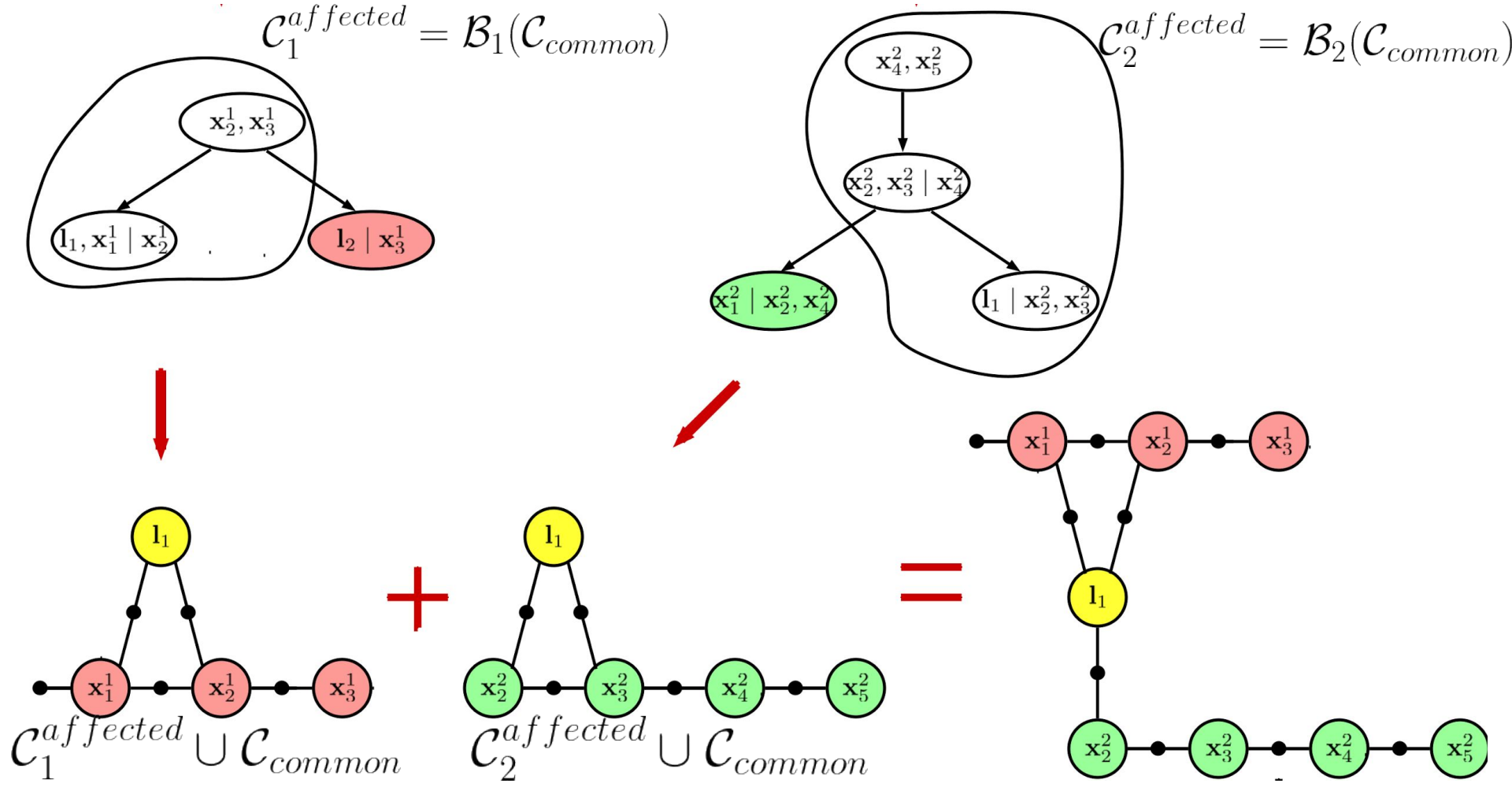
$$\mathcal{C}_2^{affected} = \mathcal{B}_2(\mathcal{C}_{common})$$



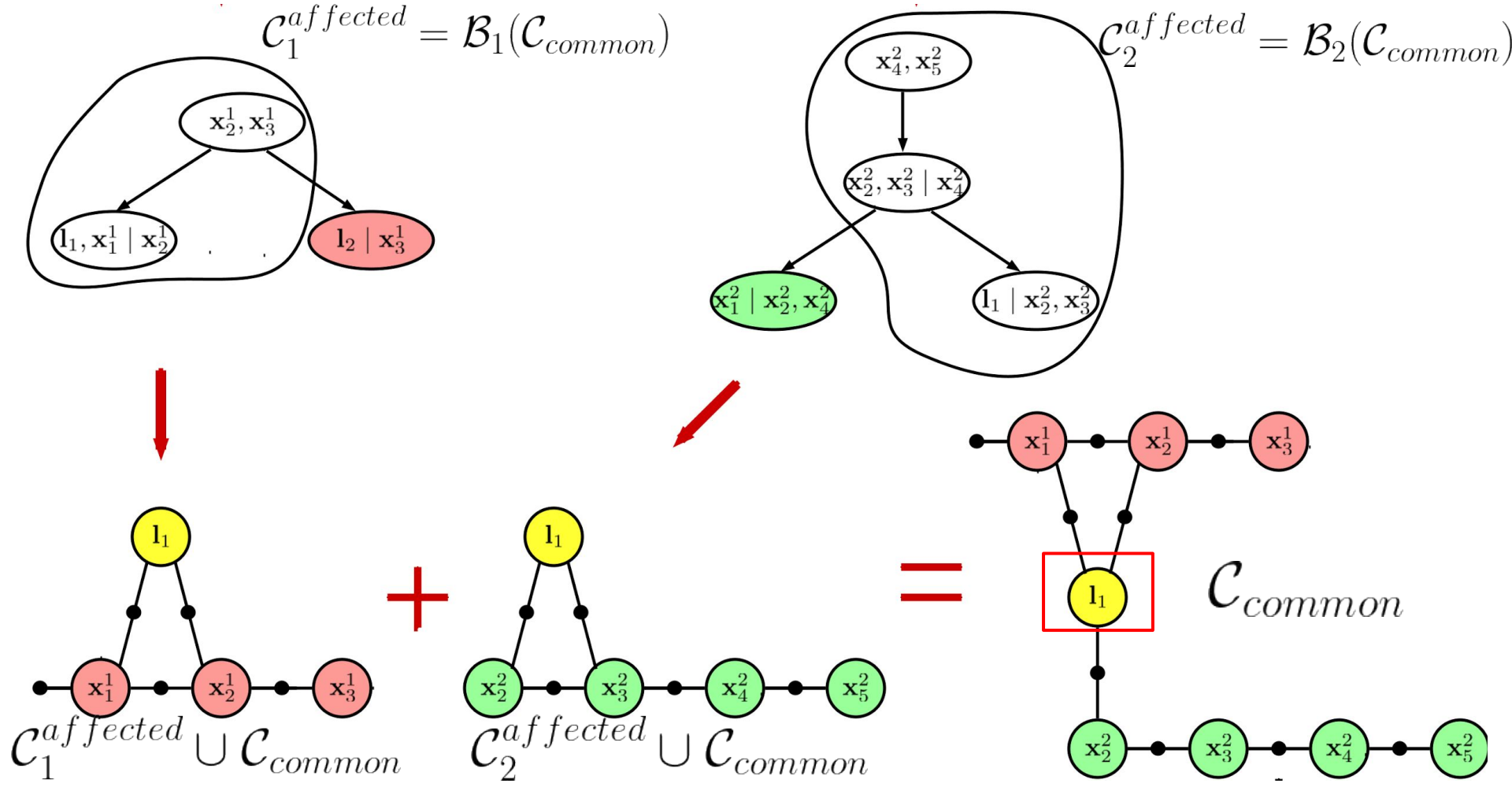
Factor Graph Fusion Example



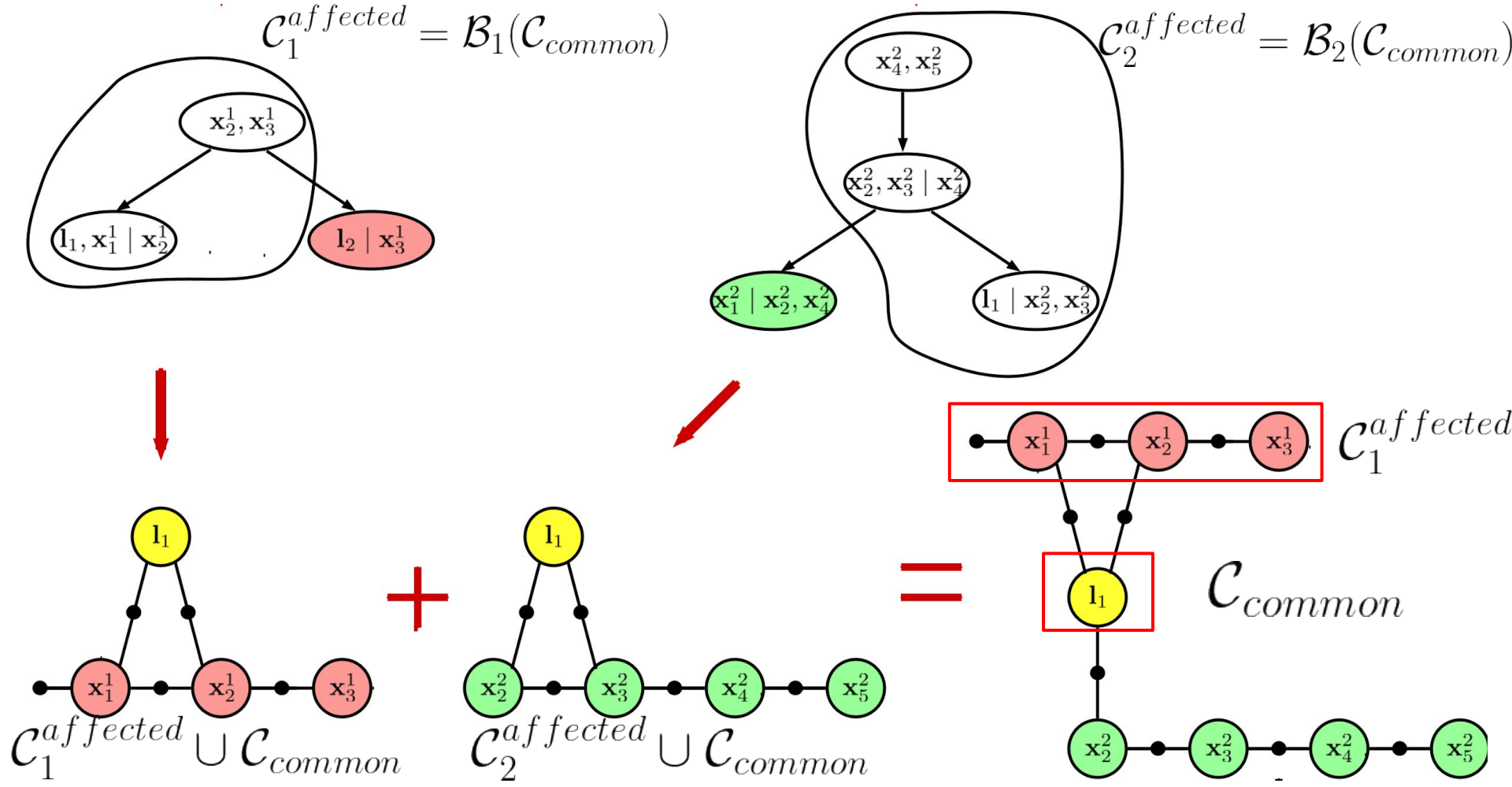
Factor Graph Fusion Example



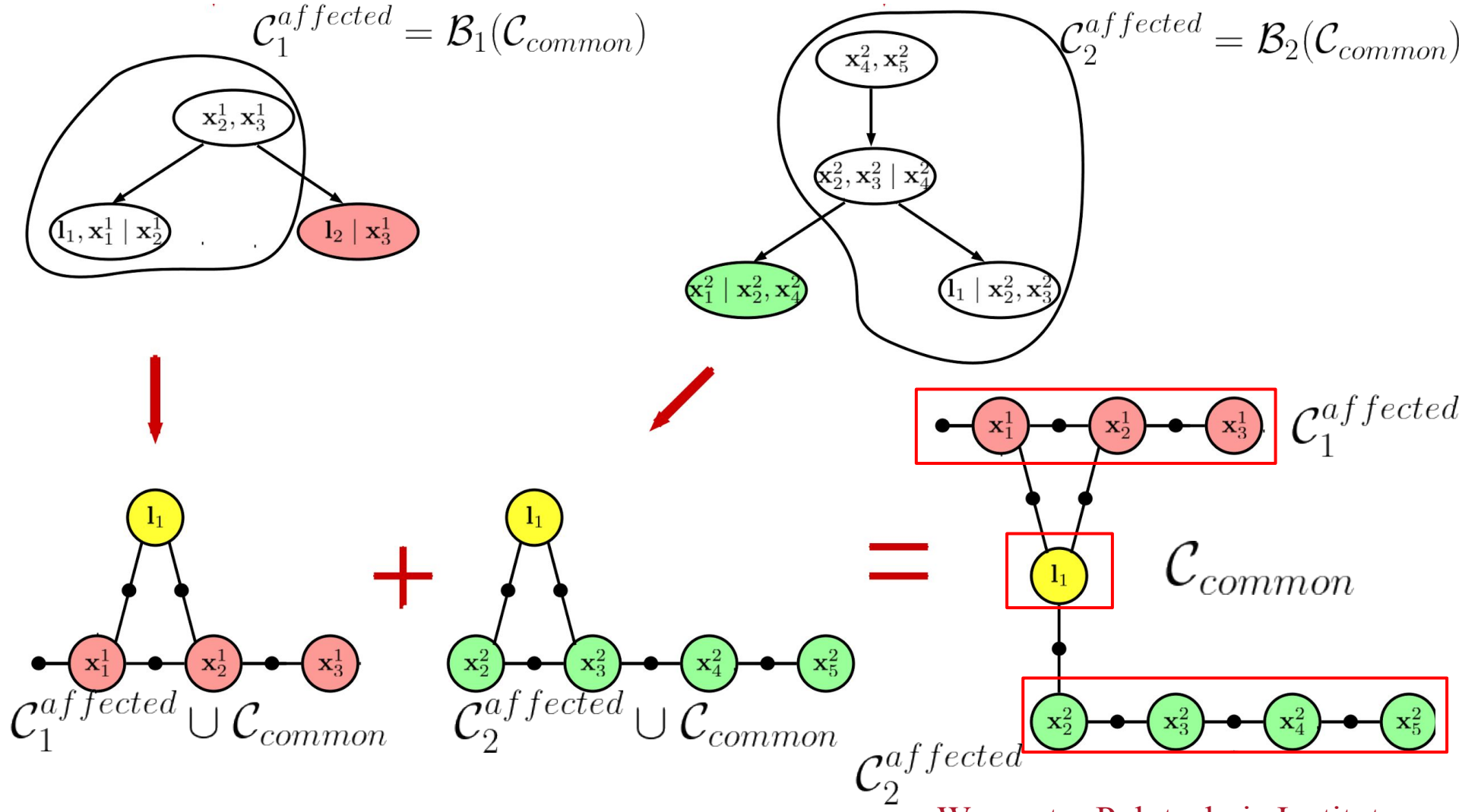
Factor Graph Fusion Example



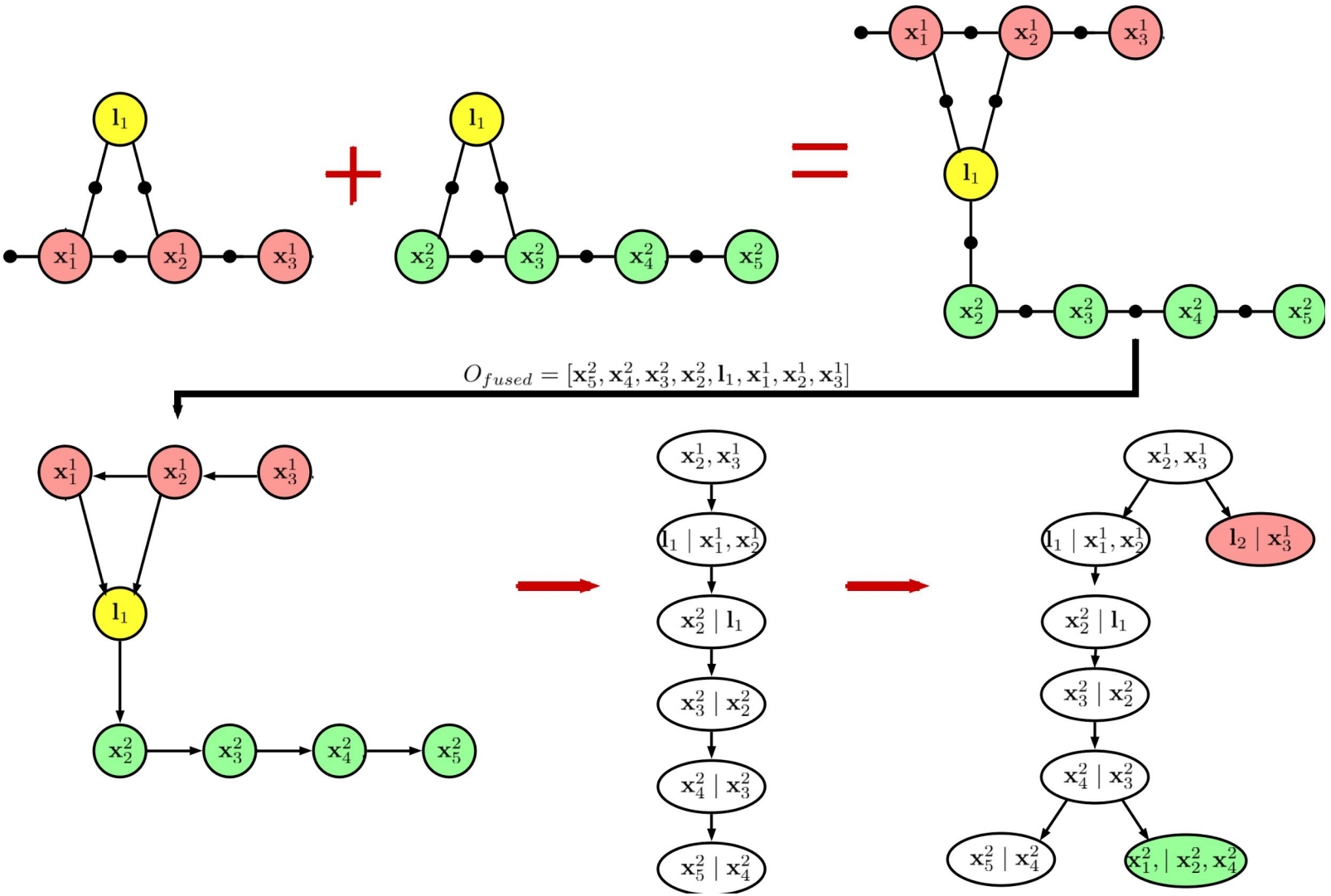
Factor Graph Fusion Example



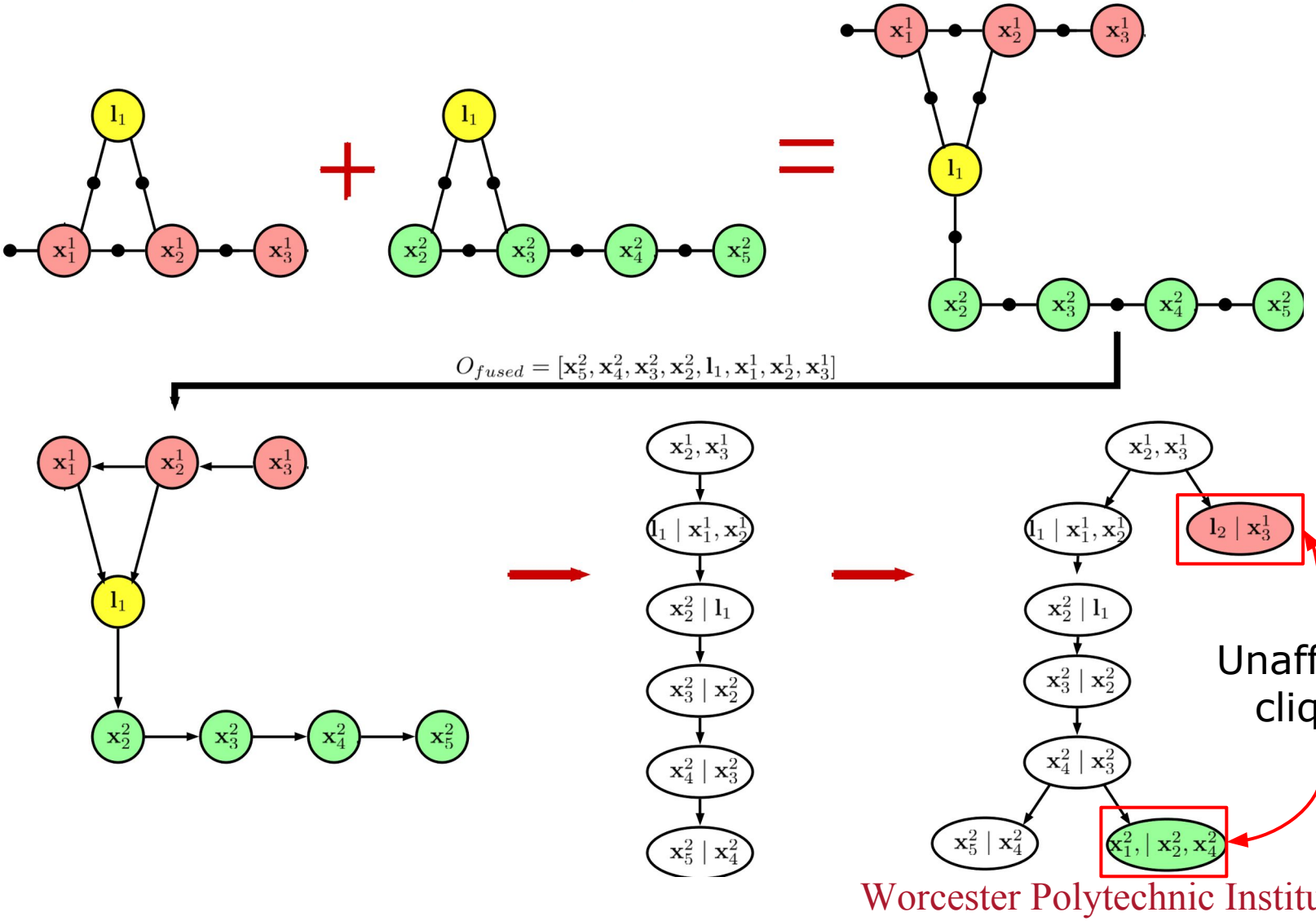
Factor Graph Fusion Example



Factor Graph Fusion Example



Factor Graph Fusion Example

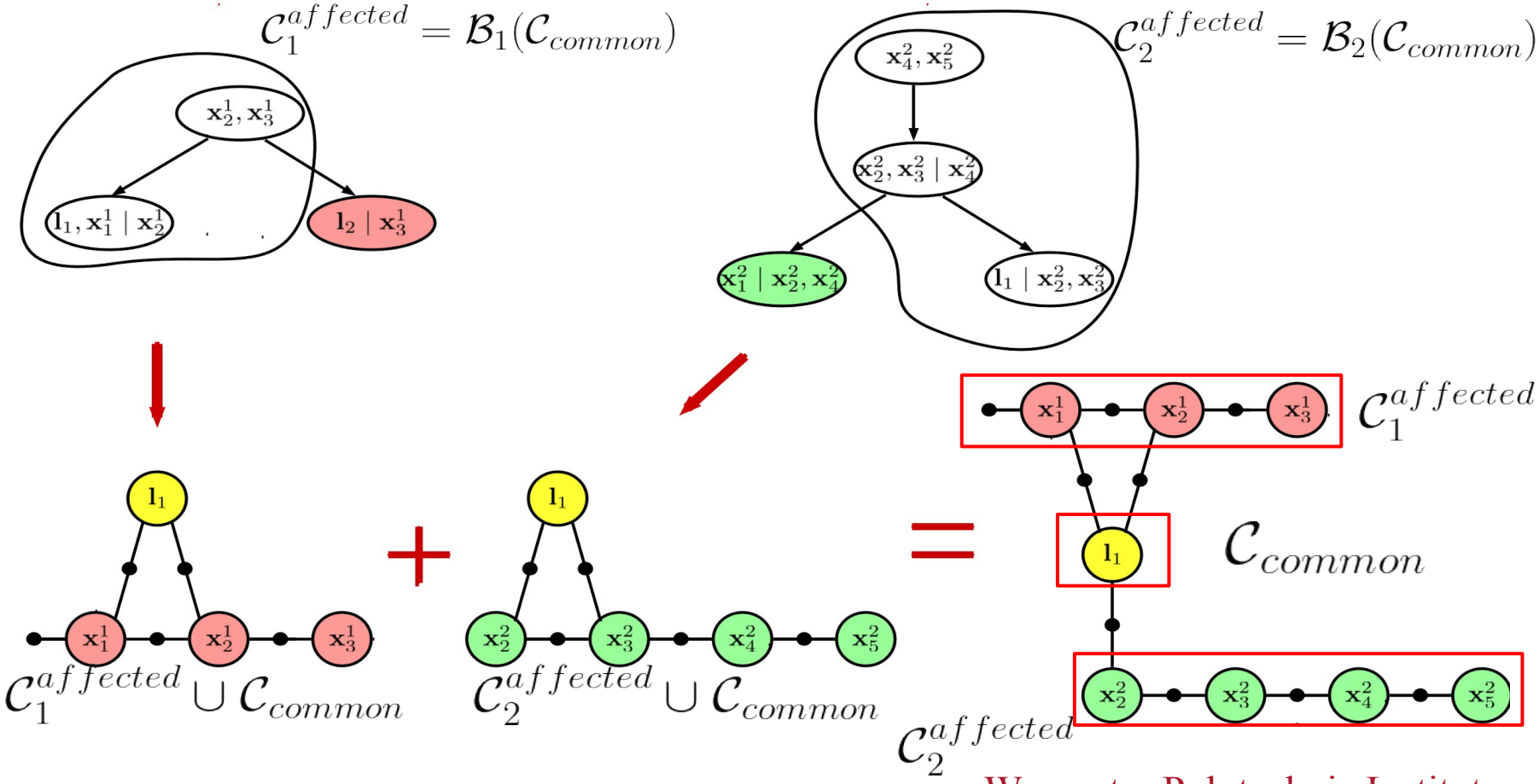


Formal Verification

Theorem 1: The set of affected variables from the Bayes tree do not form a disconnected graph.

Formal Verification

Theorem 1: The set of affected variables from the Bayes tree do not form a disconnected graph.



Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

$$\mathcal{L}_{j,k} := \begin{cases} \deg(v_j) & \text{if } j = k \\ -1 & \text{if } j \neq k \text{ and } v_j \text{ is adjacent to } v_k \\ 0 & \text{otherwise} \end{cases}$$

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

$$\mathcal{L}_{j,k} := \begin{cases} \deg(v_j) & \text{if } j = k \\ -1 & \text{if } j \neq k \text{ and } v_j \text{ is adjacent to } v_k \\ 0 & \text{otherwise} \end{cases}$$

Row transformation $R_1 = R_1 + R_2 + \ldots + R_n$ will give all zeros in row 1

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

$$\mathcal{L}_{j,k} := \begin{cases} \deg(v_j) & \text{if } j = k \\ -1 & \text{if } j \neq k \text{ and } v_j \text{ is adjacent to } v_k \\ 0 & \text{otherwise} \end{cases}$$

Row transformation $R_1 = R_1 + R_2 + \ldots + R_n$ will give all zeros in row 1

Laplacian matrix is rank deficient by 1

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

$$\mathcal{L}_{j,k} := \begin{cases} \deg(v_j) & \text{if } j = k \\ -1 & \text{if } j \neq k \text{ and } v_j \text{ is adjacent to } v_k \\ 0 & \text{otherwise} \end{cases}$$

Row transformation $R_1 = R_1 + R_2 + \ldots + R_n$ will give all zeros in row 1

Invertible Matrix Theorem

Singular matrix ⟷ 0 is an Eigenvalue

Laplacian matrix is rank deficient by 1

Formal Verification

Number of connected components = Multiplicity of 0 as an Eigenvalue to Laplacian matrix

Laplacian Matrix: $\mathcal{L}^{fused} = \mathcal{D}^{fused} - \mathcal{A}^{fused}$

$$\mathcal{L}_{j,k} := \begin{cases} \deg(v_j) & \text{if } j = k \\ -1 & \text{if } j \neq k \text{ and } v_j \text{ is adjacent to } v_k \\ 0 & \text{otherwise} \end{cases}$$

Row transformation $R_1 = R_1 + R_2 + \ldots + R_n$ will give all zeros in row 1

Laplacian matrix is rank deficient by 1

Invertible Matrix Theorem

Singular matrix ⟷ 0 is an Eigenvalue

Hence one connected component

Formal Verification

Theorem 2: On fusing the graphs, root clique of the Bayes tree will have more than a variable.

Formal Verification

Theorem 2: On fusing the graphs, root clique of the Bayes tree will have more than a variable.

Lemma 1: There is always a path between two nodes in the fused factor graph.

Proof: Follows from previous theorem

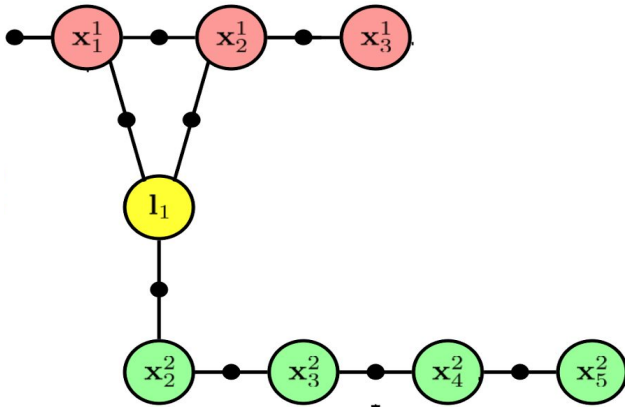
Formal Verification

Theorem 2: On fusing the graphs, root clique of the Bayes tree will have more than a variable.

Lemma 1: There is always a path between two nodes in the fused factor graph.

Proof: Follows from previous theorem

Lemma 2: The edge between any two nodes fill-in if all the variables are eliminated prior to those two nodes.



Formal Verification

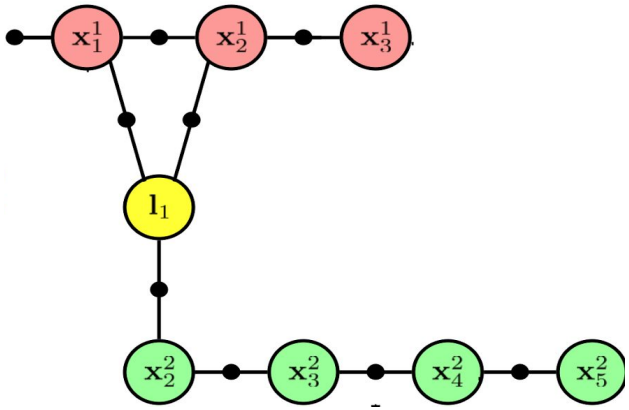
Theorem 2: On fusing the graphs, root clique of the Bayes tree will have more than a variable.

Lemma 1: There is always a path between two nodes in the fused factor graph.

Proof: Follows from previous theorem

Lemma 2: The edge between any two nodes fill-in if all the variables are eliminated prior to those two nodes.

Eliminated column produce changes over the yet to be eliminated columns



Formal Verification

Theorem 2: On fusing the graphs, root clique of the Bayes tree will have more than a variable.

Lemma 1: There is always a path between two nodes in the fused factor graph.

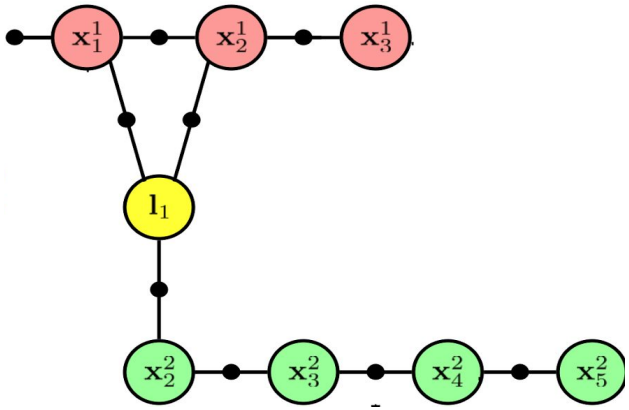
Proof: Follows from previous theorem

Lemma 2: The edge between any two nodes fill-in if all the variables are eliminated prior to those two nodes.

Eliminated column produce changes over the yet to be eliminated columns

There is always a factor of form p(b|a)

b - last but one variable
a - last variable



Fusion Ordering

- Variable ordering of each graph:

$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

Fusion Ordering

- Variable ordering of each graph:

$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

- An ordering query function:

$$o_i(c_i^{n_i}) = o_i^{c_i^{n_i}}; \quad i = 1, 2$$

Fusion Ordering

- Variable ordering of each graph:

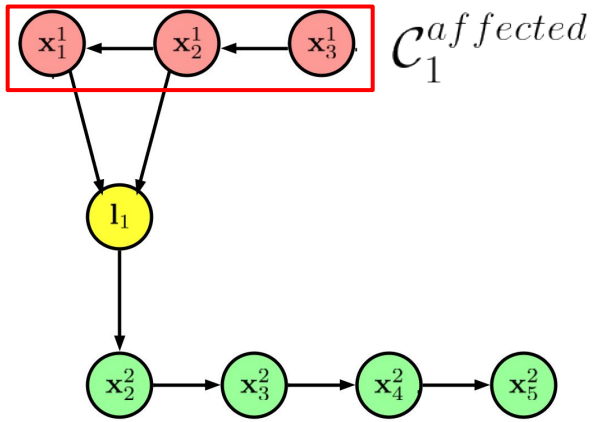
$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

- An ordering query function:

$$o_i(c_i^{n_i}) = o_i^{c_i^{n_i}}; \quad i = 1, 2$$

- Fusion ordering of $\mathcal{C}_1^{affected}$:

$$O_1^{fused} = arg\ \text{sort}\left(\bigcup_{j \in \mathcal{C}_1} o_1(j)\right)$$



Fusion Ordering

- Variable ordering of each graph:

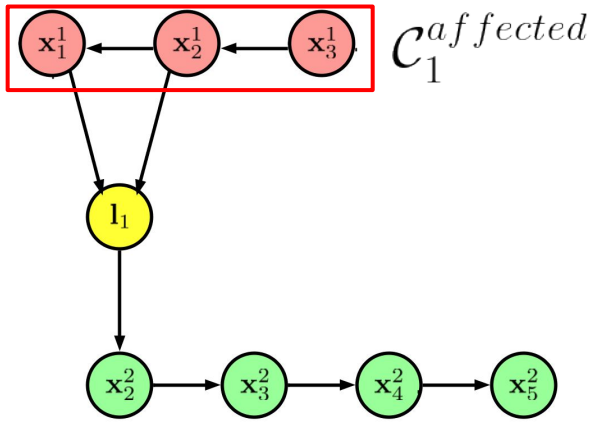
$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

- An ordering query function:

$$o_i(c_i^{n_i}) = o_i^{c_i^{n_i}}; \quad i = 1, 2$$

- Fusion ordering of $\mathcal{C}_1^{affected}$:

$$O_1^{fused} = arg\ \text{sort}\left(\bigcup_{j \in \mathcal{C}_1} o_1(j)\right)$$



arg sort: Original position of each element in the sorted array.

Eg: [41,23,12,8,22] →[4,2,5,2,1]

Fusion Ordering

- Variable ordering of each graph:

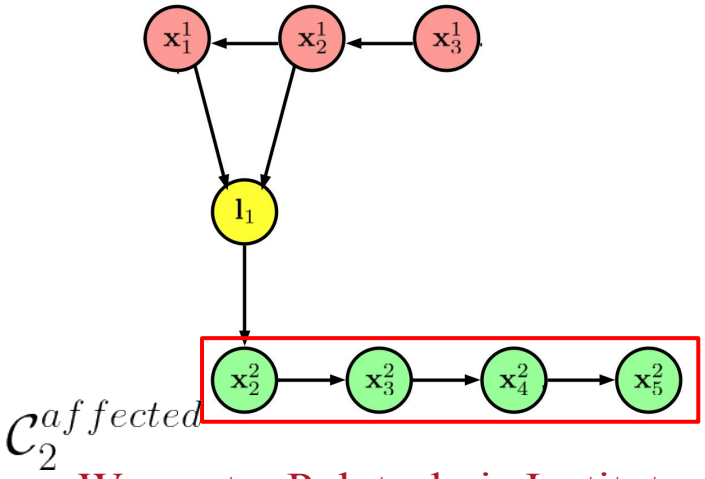
$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

- An ordering query function:

$$o_i(c_i^{n_i}) = o_i^{c_i^{n_i}}; \quad i = 1, 2$$

- Fusion ordering of $\mathcal{C}_2^{affected}$:

$$O_2^{fused} = \mid O_1^{fused} \mid + arg\ \text{sort}\left(\bigcup_{j \in \mathcal{C}_2} o_2(j) \right)$$



arg sort: Original position of each element in the sorted array.

Eg: [41,23,12,8,22] →[4,2,5,2,1]

Fusion Ordering

- Variable ordering of each graph:

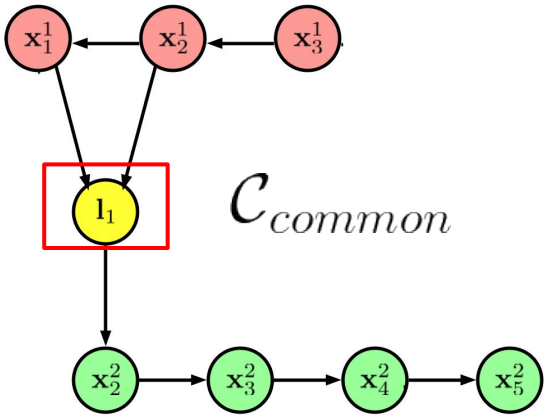
$$O_i = [o_i^{c_i^1}, o_i^{c_i^2}, \ldots, o_i^{c_i^{n_i}}]; \quad i = 1, 2$$

- An ordering query function:

$$o_i(c_i^{n_i}) = o_i^{c_i^{n_i}}; \quad i = 1, 2$$

- Fusion ordering of $\mathcal{C}_{common}$:

$$O_{common}^{fused} = \mid O_1^{fused} \mid + \mid O_2^{fused} \mid + arg\,\text{sort}\left(\bigcup_{j \in \mathcal{C}_{common}} o_{common}(j) \right)$$



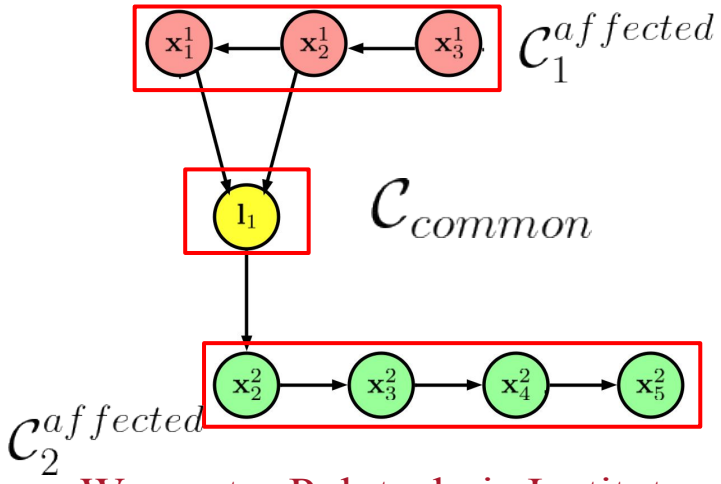
arg sort: Original position of each element in the sorted array.

Eg: [41,23,12,8,22] →[4,2,5,2,1]

Fusion Ordering

- Total Fusion Ordering:

$$O^{fused} = [O_1^{fused}, O_2^{fused}, O_{common}^{fused}]$$



Fusion Ordering

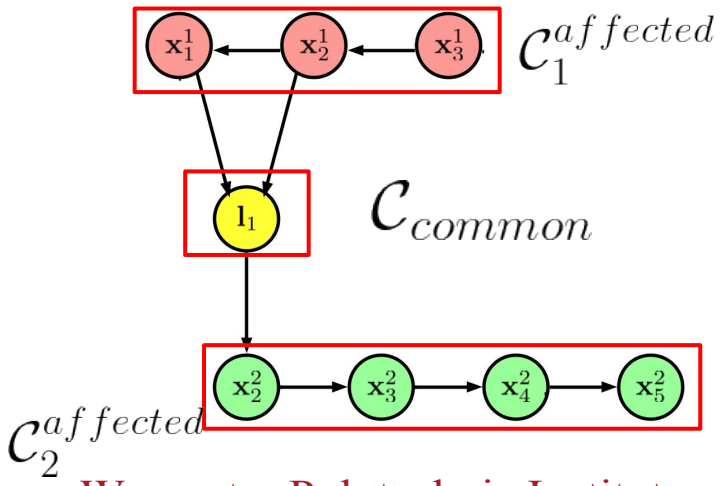
- Total Fusion Ordering:

$$O^{fused} = [O_1^{fused}, O_2^{fused}, O_{common}^{fused}]$$

- The Permutation matrix $P^1 \in [0,1]^{|N| \times |N|}$:

$$P^1_{j,k} := \begin{cases} 1 & \text{if } j = O_k^{fused} \\ 0 & \text{otherwise} \end{cases}$$

where $N = |O^{fused}|$



Why Fusion Ordering Works?

Why Fusion Ordering Works?

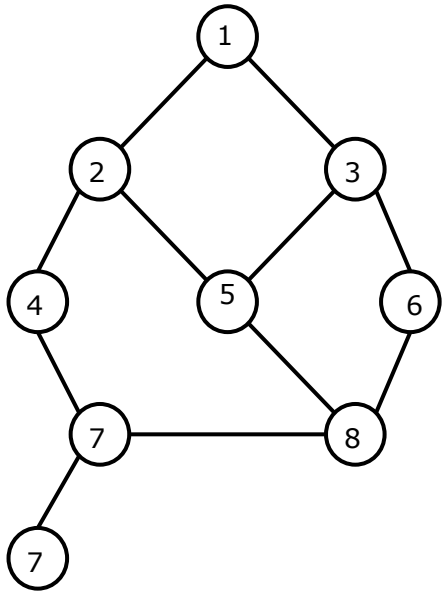
Nested Dissection

Why Fusion Ordering Works?

Nested Dissection:
A fill reducing method based on divide and conquer principle.

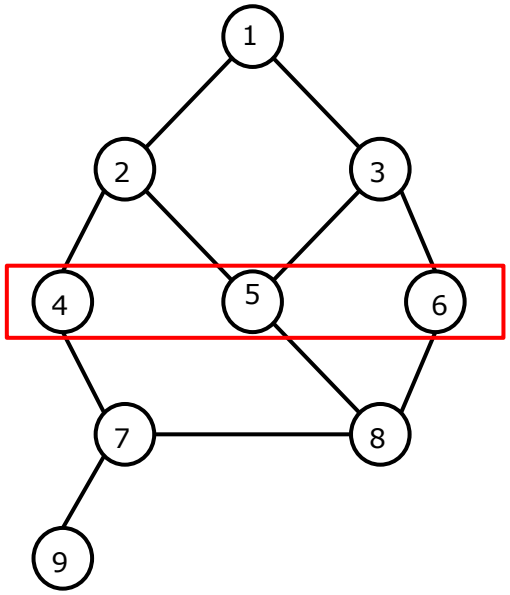
Why Fusion Ordering Works?

Nested Dissection:
A fill reducing method based on divide and conquer principle.



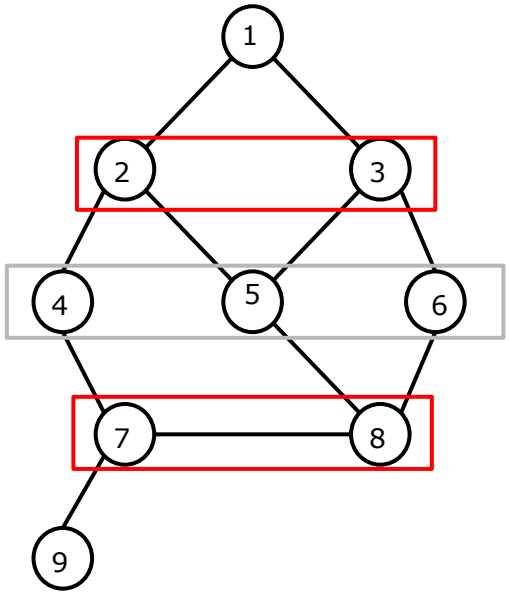
Why Fusion Ordering Works?

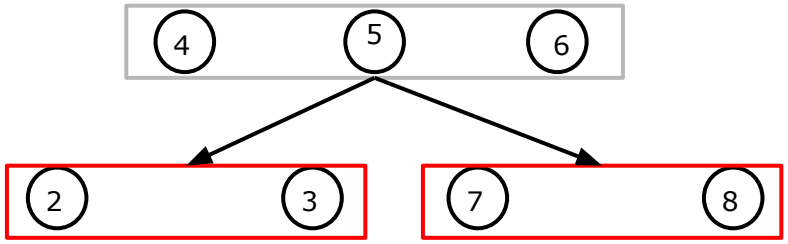
Nested Dissection:
A fill reducing method based on divide and conquer principle.



Why Fusion Ordering Works?

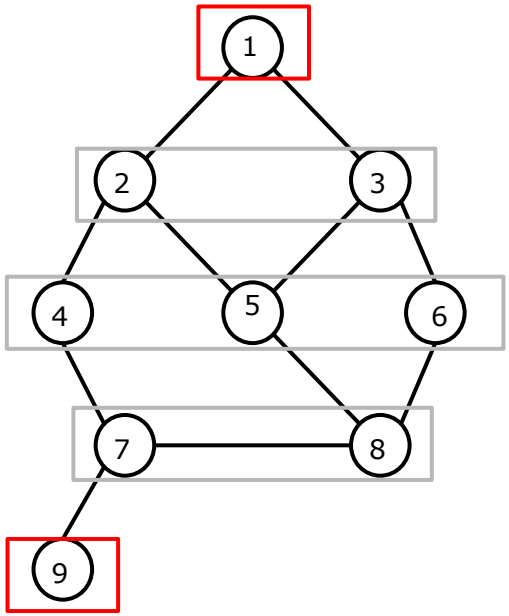
Nested Dissection:
A fill reducing method based on divide and conquer principle.

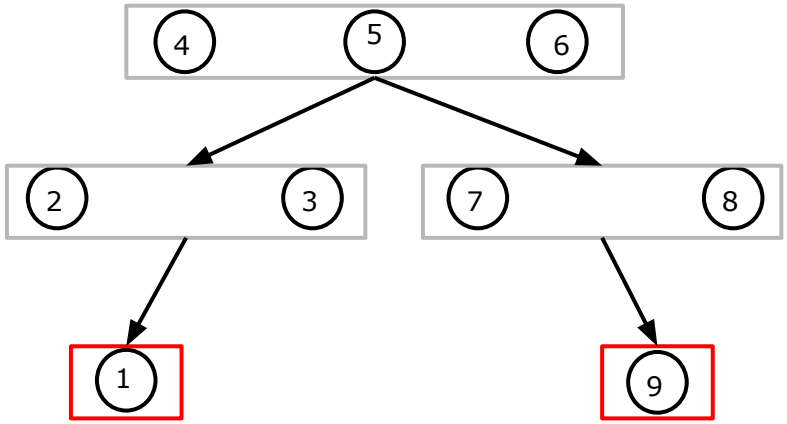




Why Fusion Ordering Works?

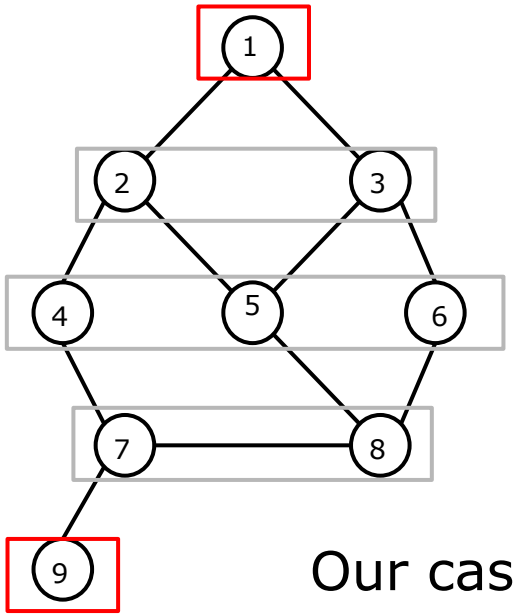
Nested Dissection:
A fill reducing method based on divide and conquer principle.

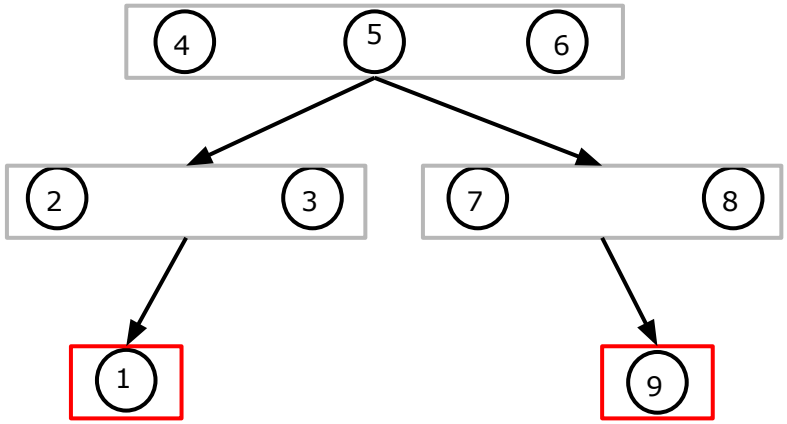




Why Fusion Ordering Works?

Nested Dissection:
A fill reducing method based on divide and conquer principle.





Our case is a depth 1 nested dissection.

Numerical Stabilization

Pivoting: Common technique to add numerical stability.

Numerical Stabilization

Pivoting: Common technique to add numerical stability.

Finding an element to null the remaining elements in its row or column

Numerical Stabilization

Pivoting: Common technique to add numerical stability.

Finding an element to null the remaining elements in its row or column

x	y	z	rhs
*3	2	-4	3
2	3	3	15
5	-3	1	14

$3R_2-2R_1 \rightarrow R_2$ & $3R_3-5R_1 \rightarrow R_3$

x	y	z	rhs
3	2	-4	3
0	5	17	39
0	-19	23	27

Numerical Stabilization

Pivoting: Common technique to add numerical stability.

Finding an element to null the remaining elements in its row or column

x	y	z	rhs
*3	2	-4	3
2	3	3	15
5	-3	1	14

$3R_2-2R_1 \rightarrow R_2$ & $3R_3-5R_1 \rightarrow R_3$

x	y	z	rhs
3	2	-4	3
0	5	17	39
0	-19	23	27

How about finding rows like these?

x	y	z	rhs
5	0	0	15
0	5	0	5
0	0	1	2

Numerical Stabilization

Pivoting: Common technique to add numerical stability.

Finding an element to null the remaining elements in its row or column

x	y	z	rhs
*3	2	-4	3
2	3	3	15
5	-3	1	14

$3R_2-2R_1 \rightarrow R_2$ & $3R_3-5R_1 \rightarrow R_3$

x	y	z	rhs
3	2	-4	3
0	5	17	39
0	-19	23	27

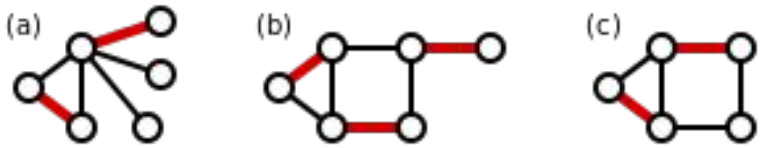
How about finding rows like these?

x	y	z	rhs
5	0	0	15
0	5	0	5
0	0	1	2

Matching problem: Finding pairwise non-adjacent edges

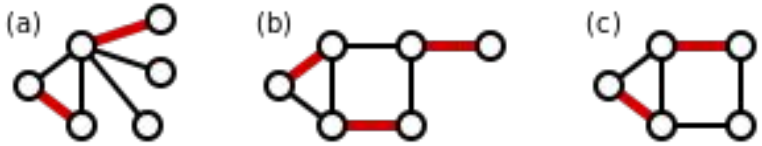
Numerical Stabilization

Matching: Finding an edge set such that no two edges are incident to the same node.



Numerical Stabilization

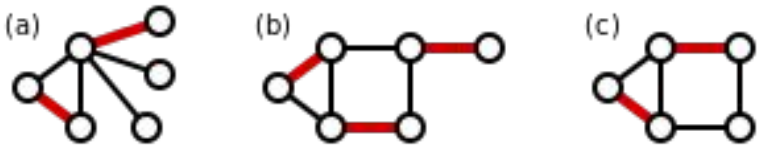
Matching: Finding an edge set such that no two edges are incident to the same node.



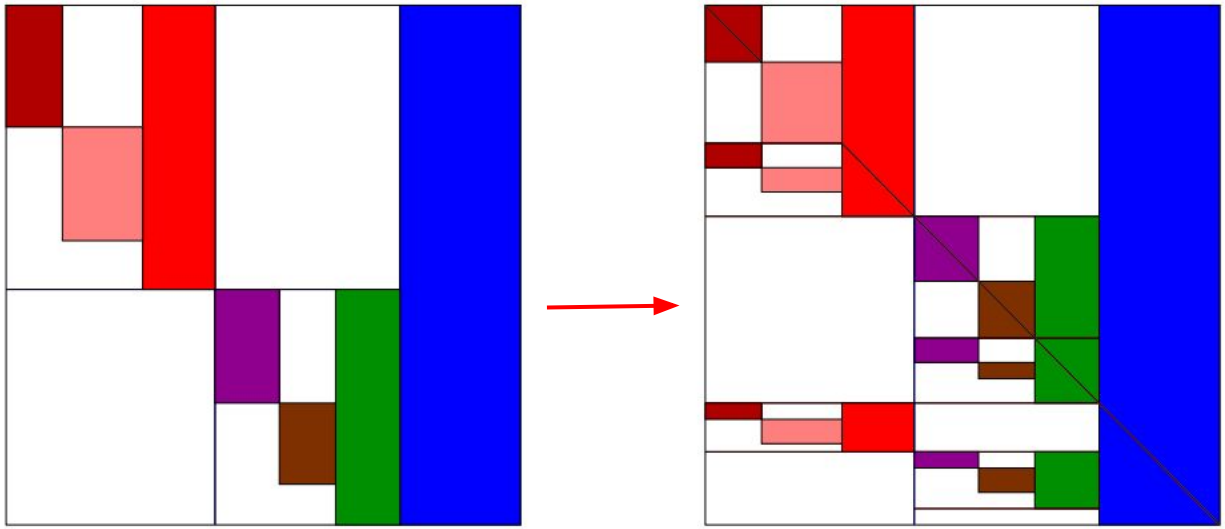
HSL routine MC64: Bipartite matching routine for MATLAB

Numerical Stabilization

Matching: Finding an edge set such that no two edges are incident to the same node.

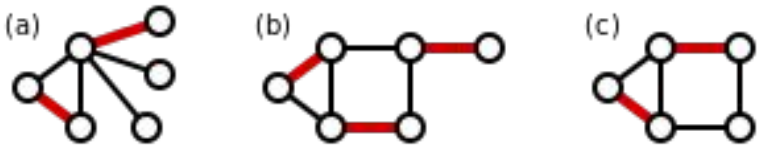


HSL routine MC64: Bipartite matching routine for MATLAB

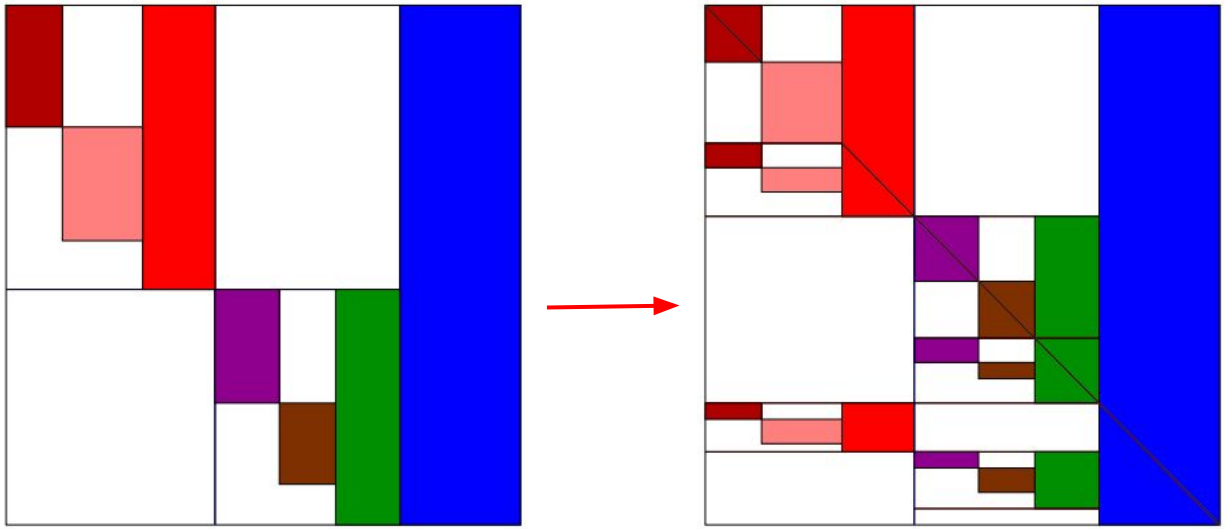


Numerical Stabilization

Matching: Finding an edge set such that no two edges are incident to the same node.

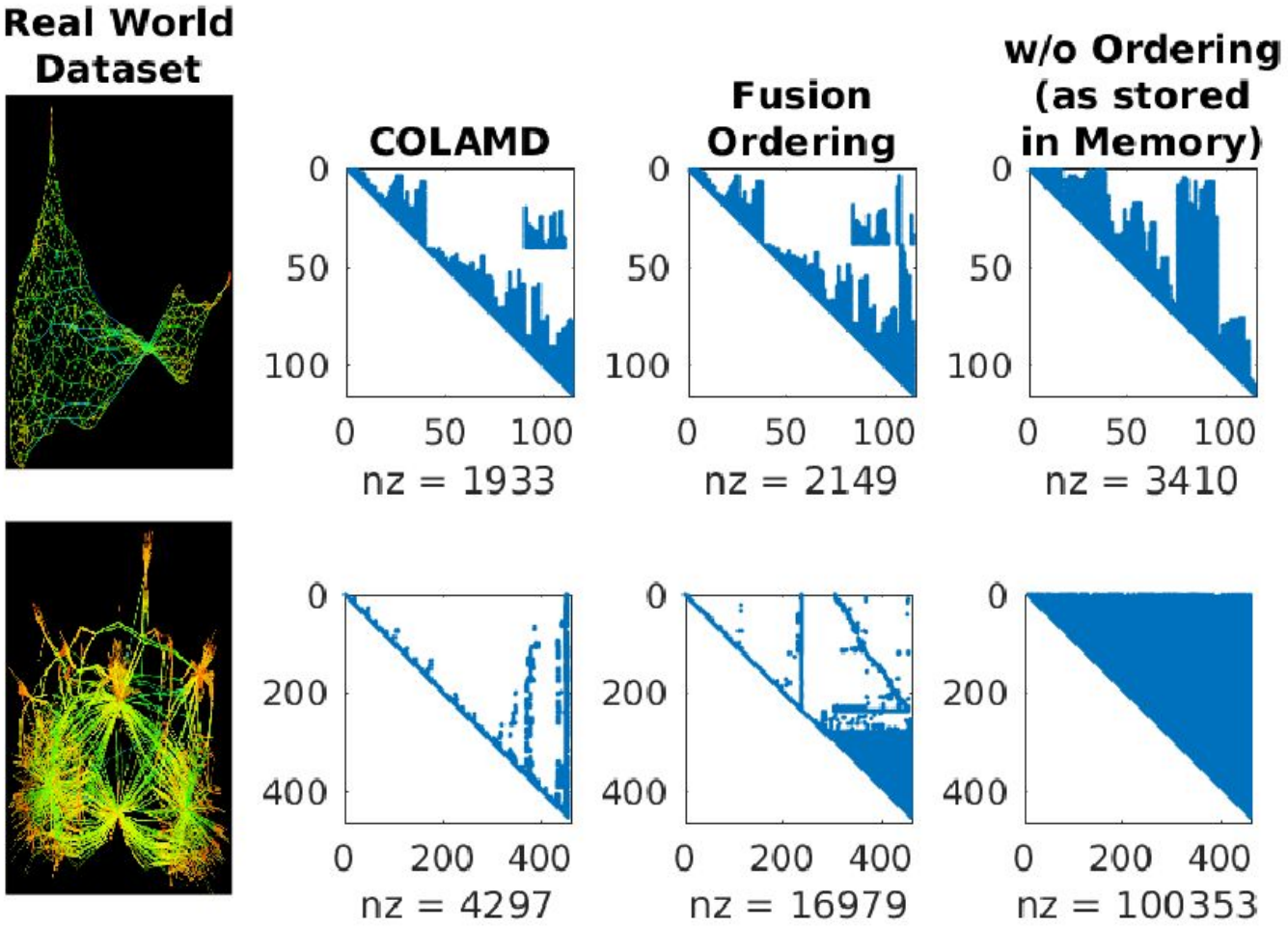


HSL routine MC64: Bipartite matching routine for MATLAB

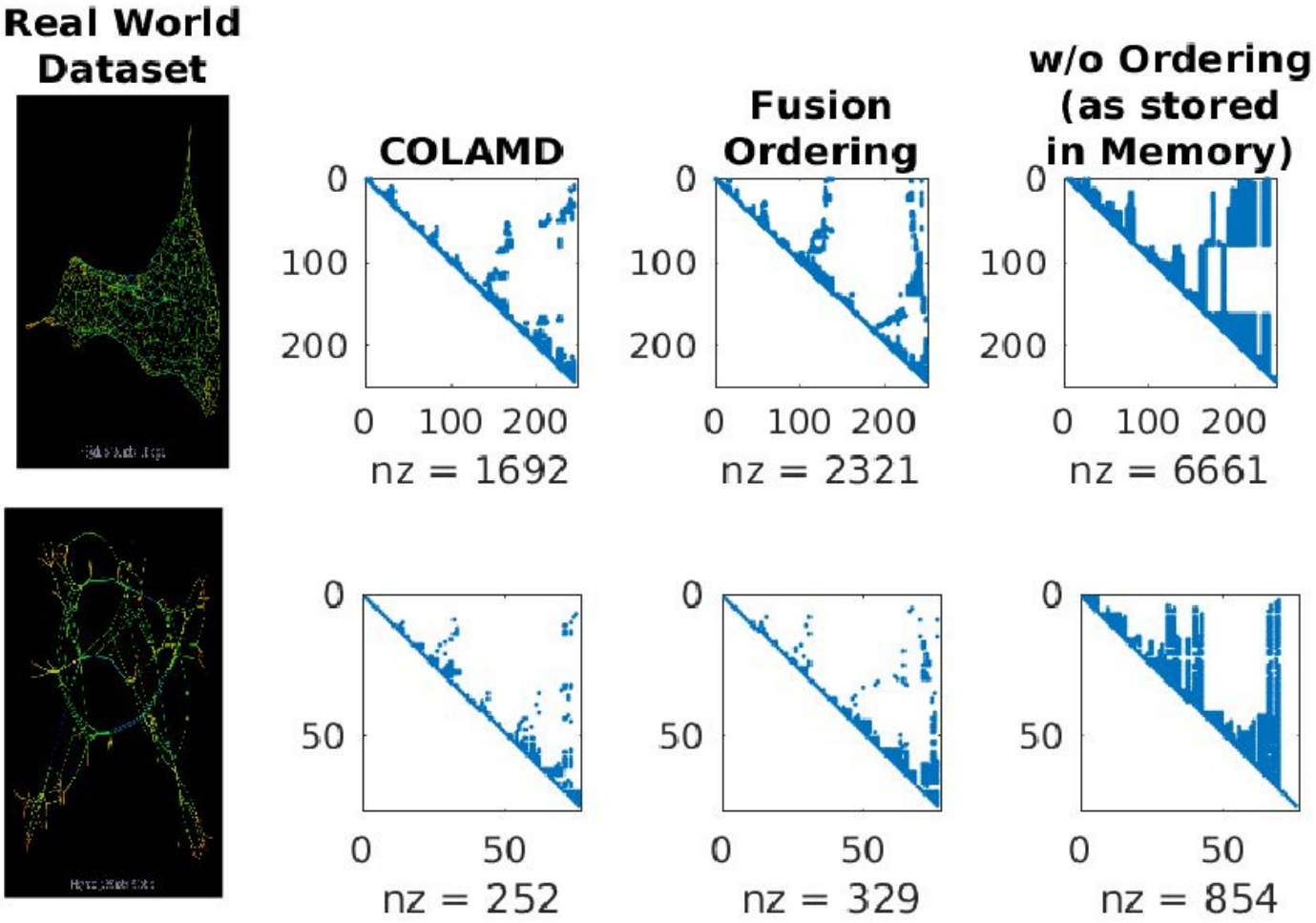


$$A_r^{ordered} = P^{matching} A_r P^1$$

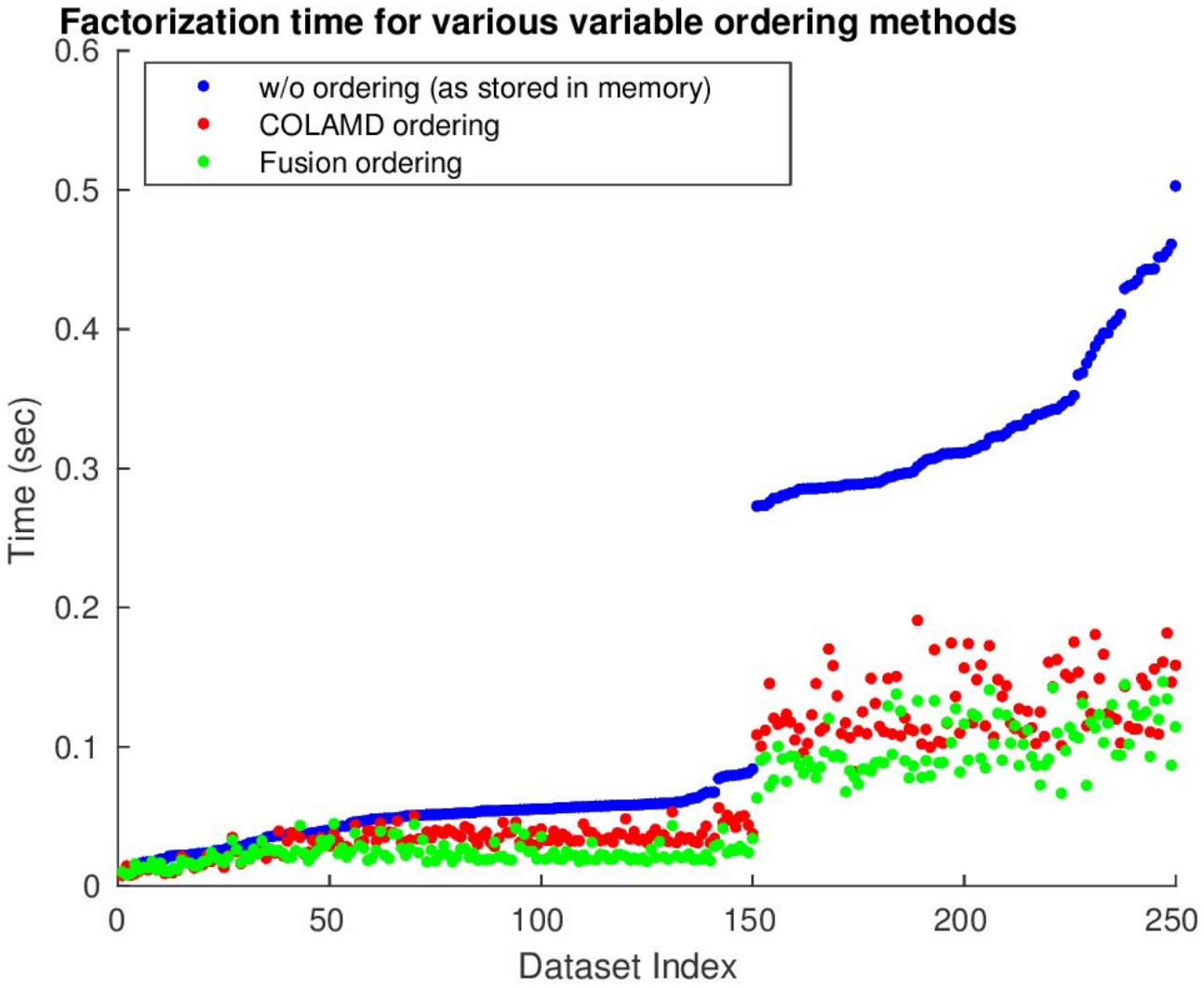
SuiteSparse Results



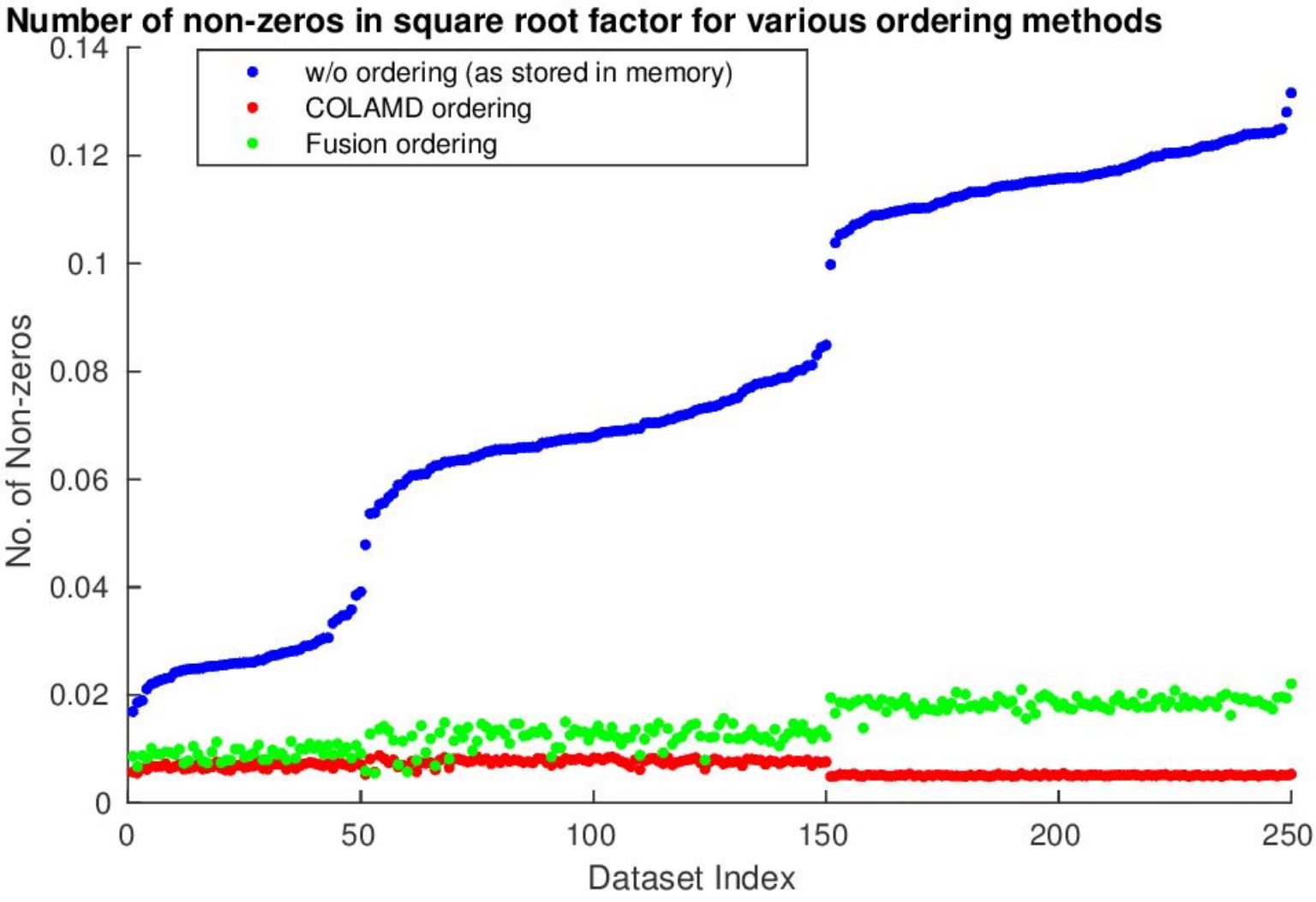
SuiteSparse Results



SuiteSparse Results



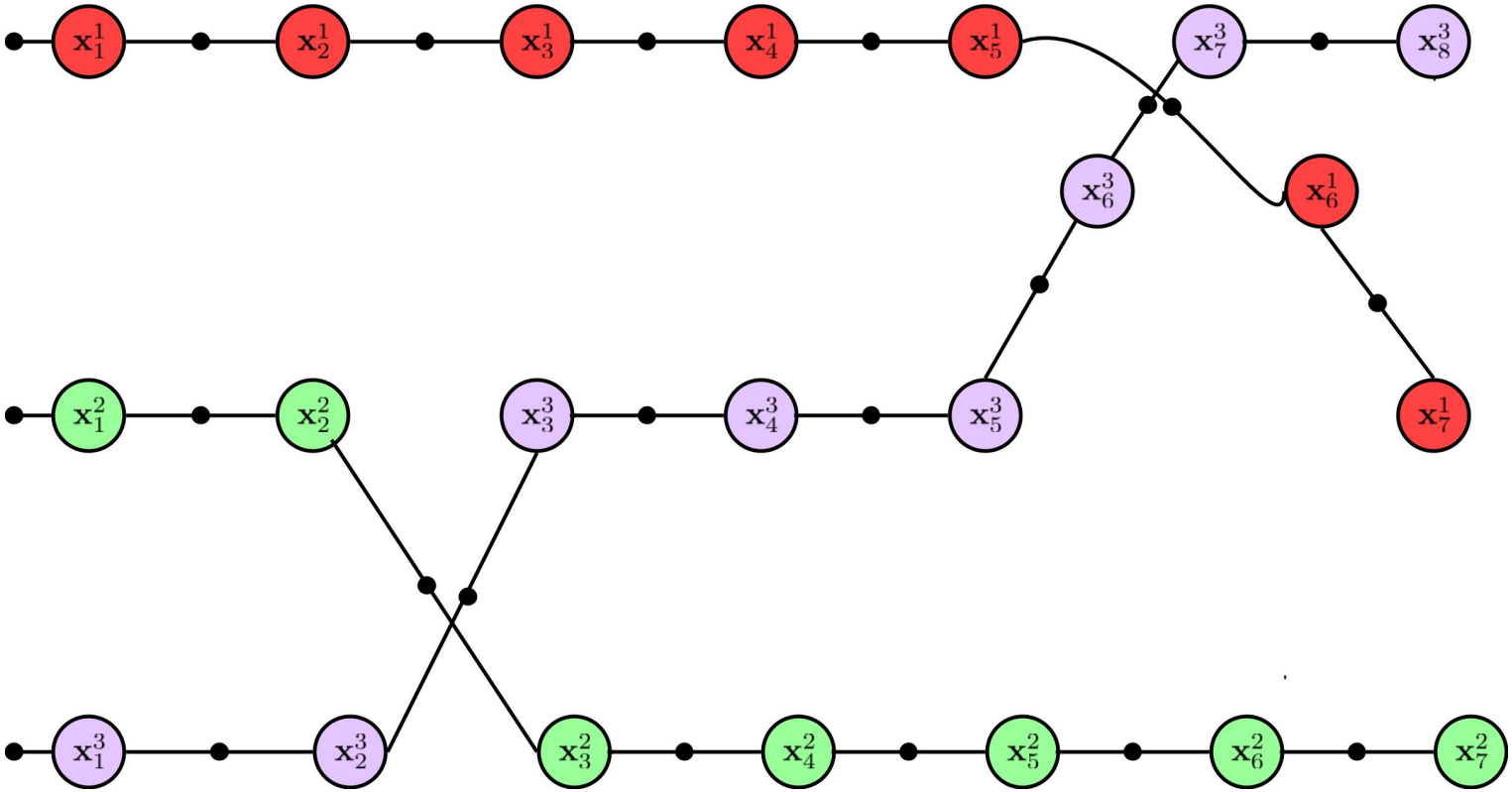
SuiteSparse Results



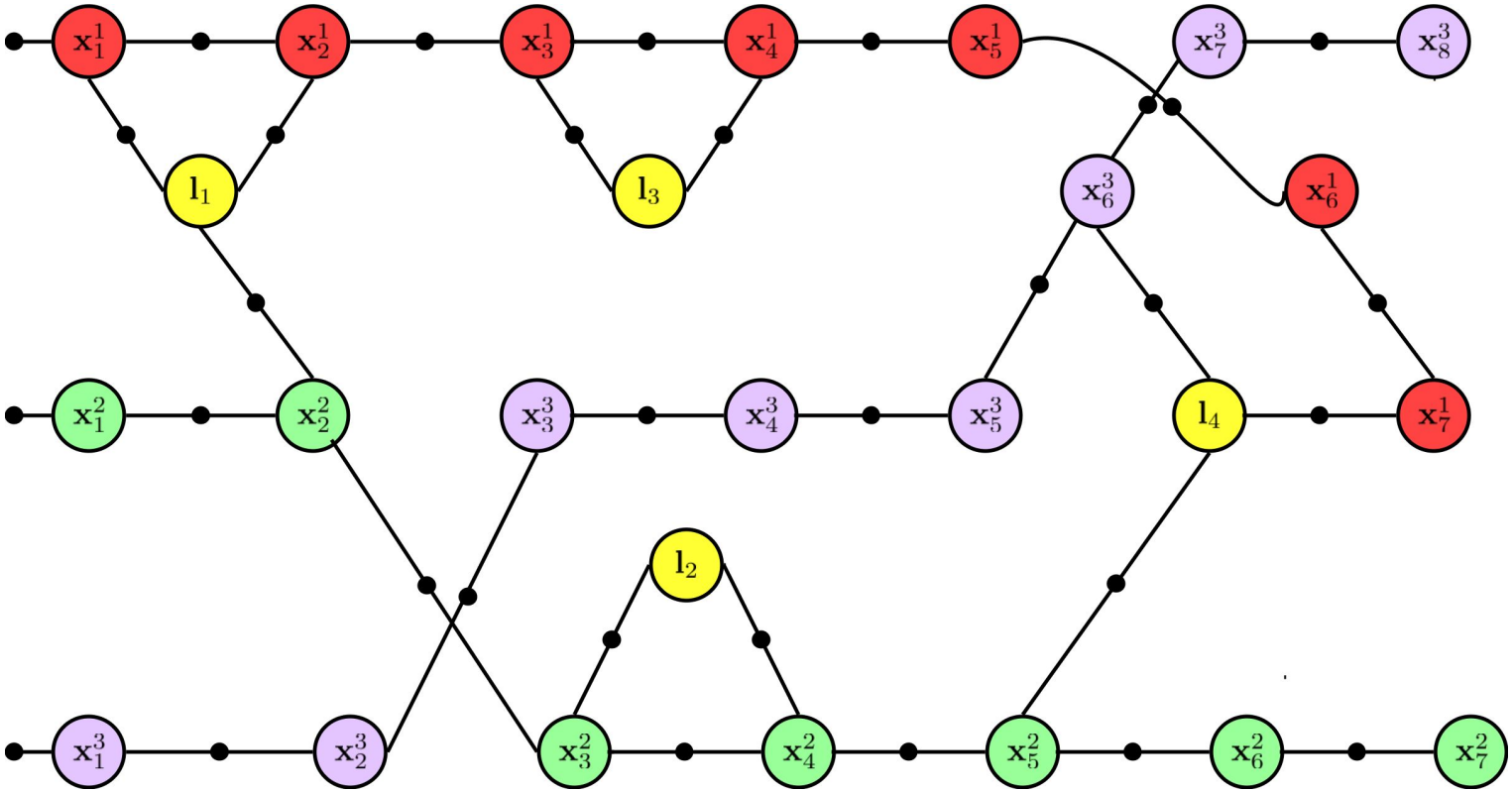
Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

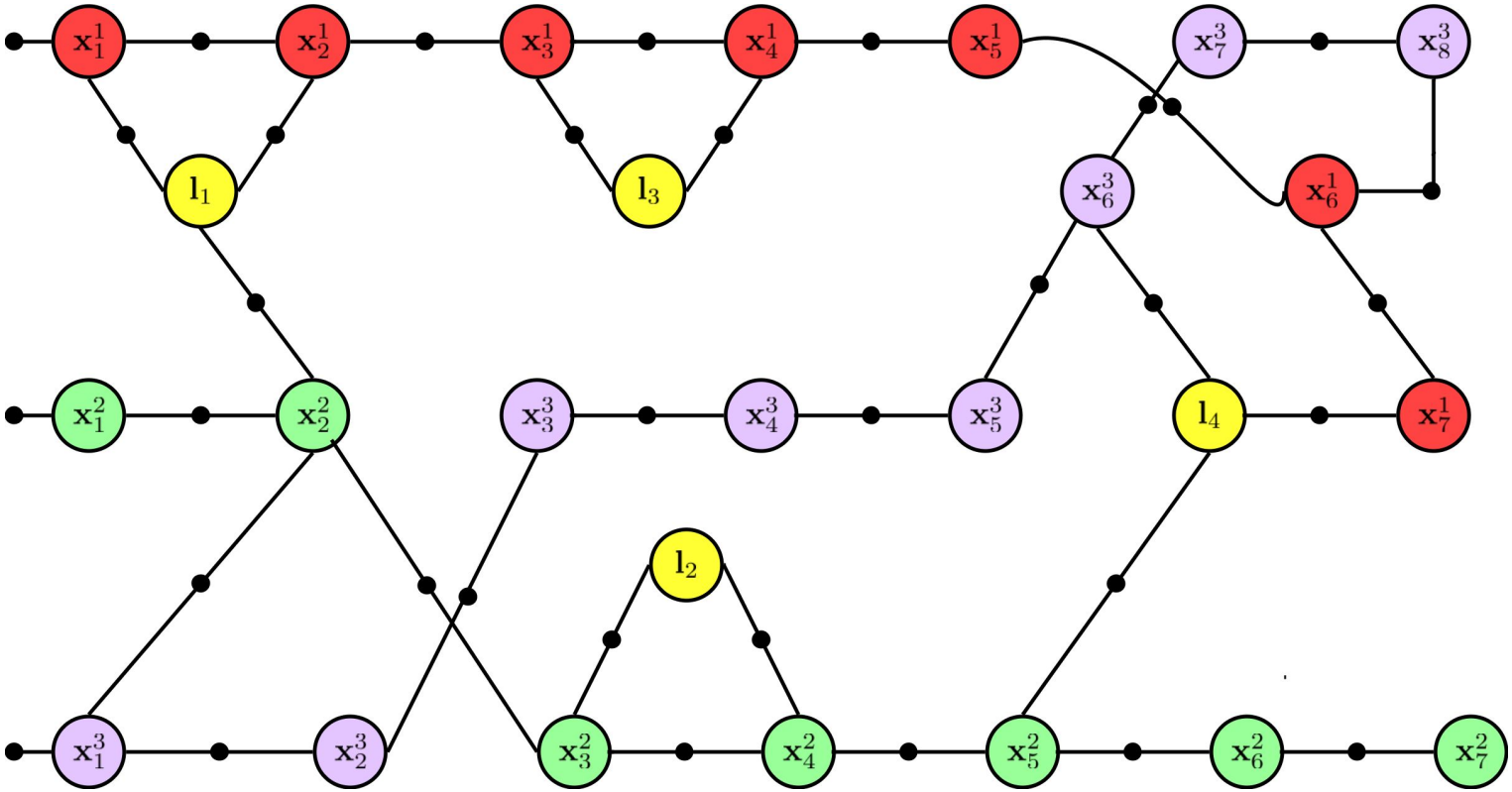
Relative Pose Graph Initialization



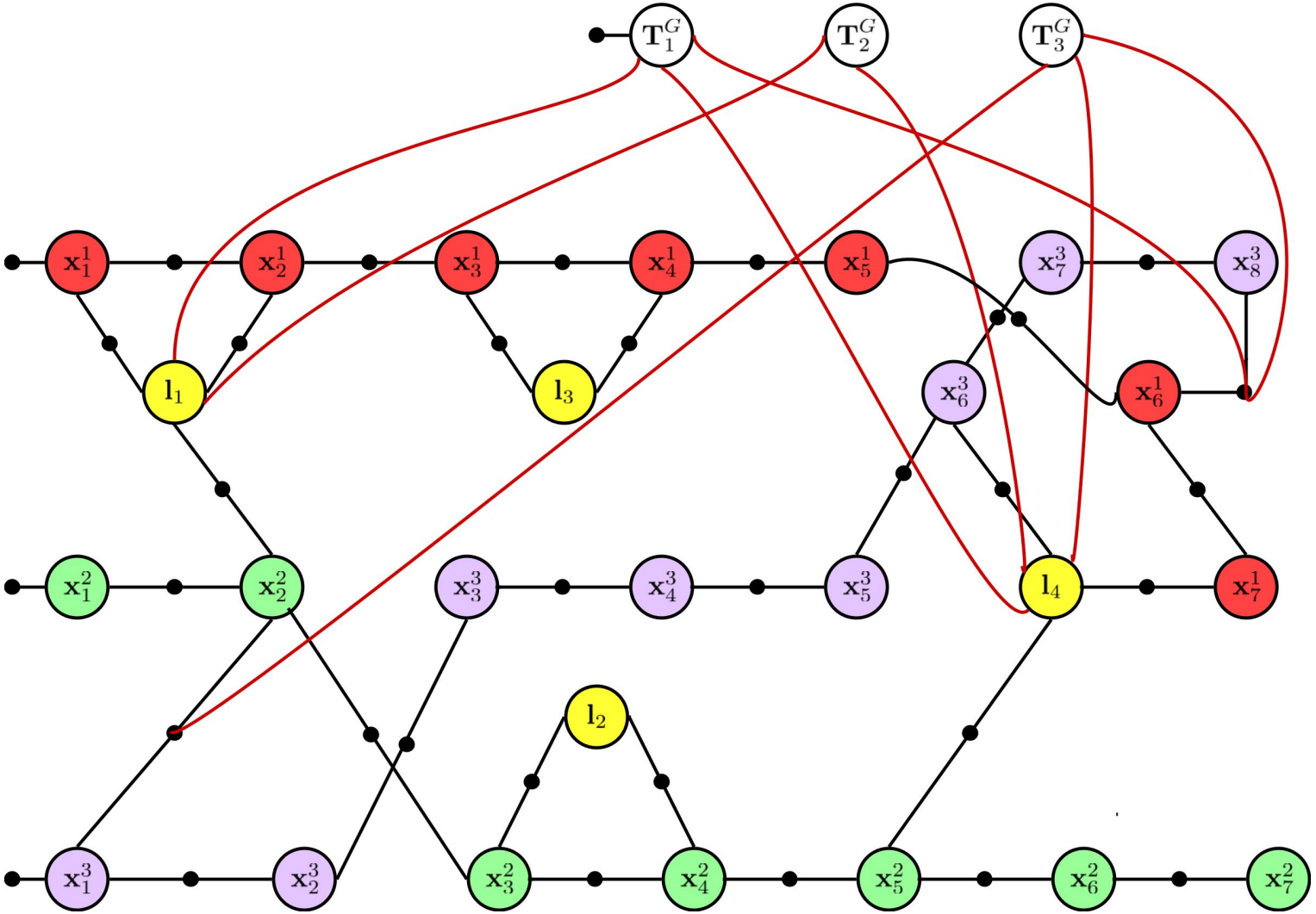
Relative Pose Graph Initialization



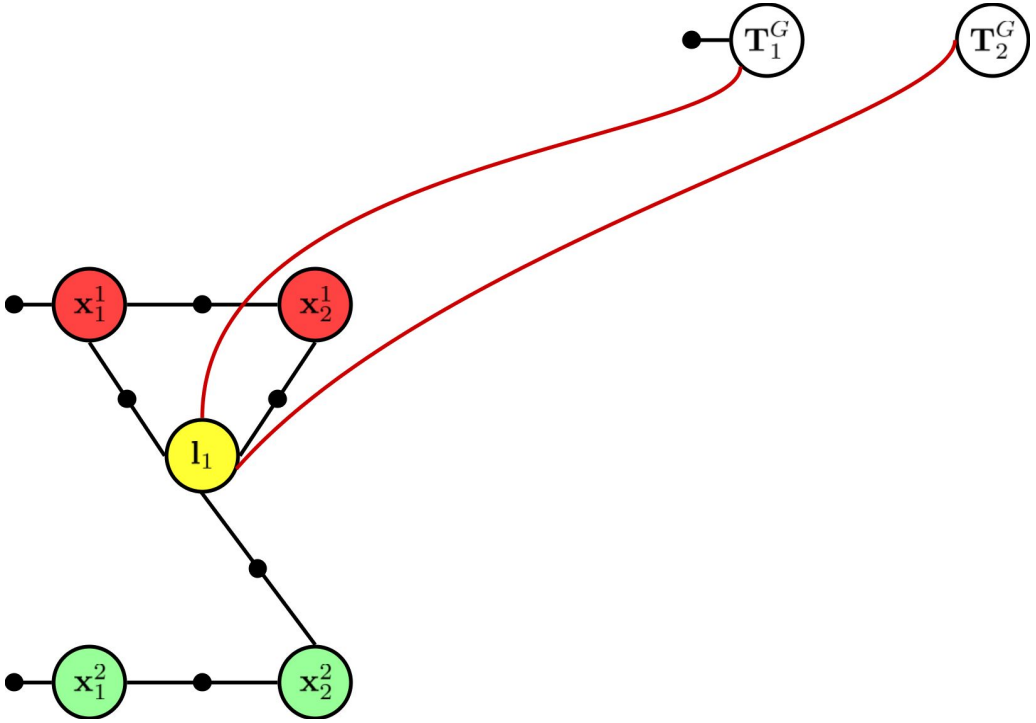
Relative Pose Graph Initialization



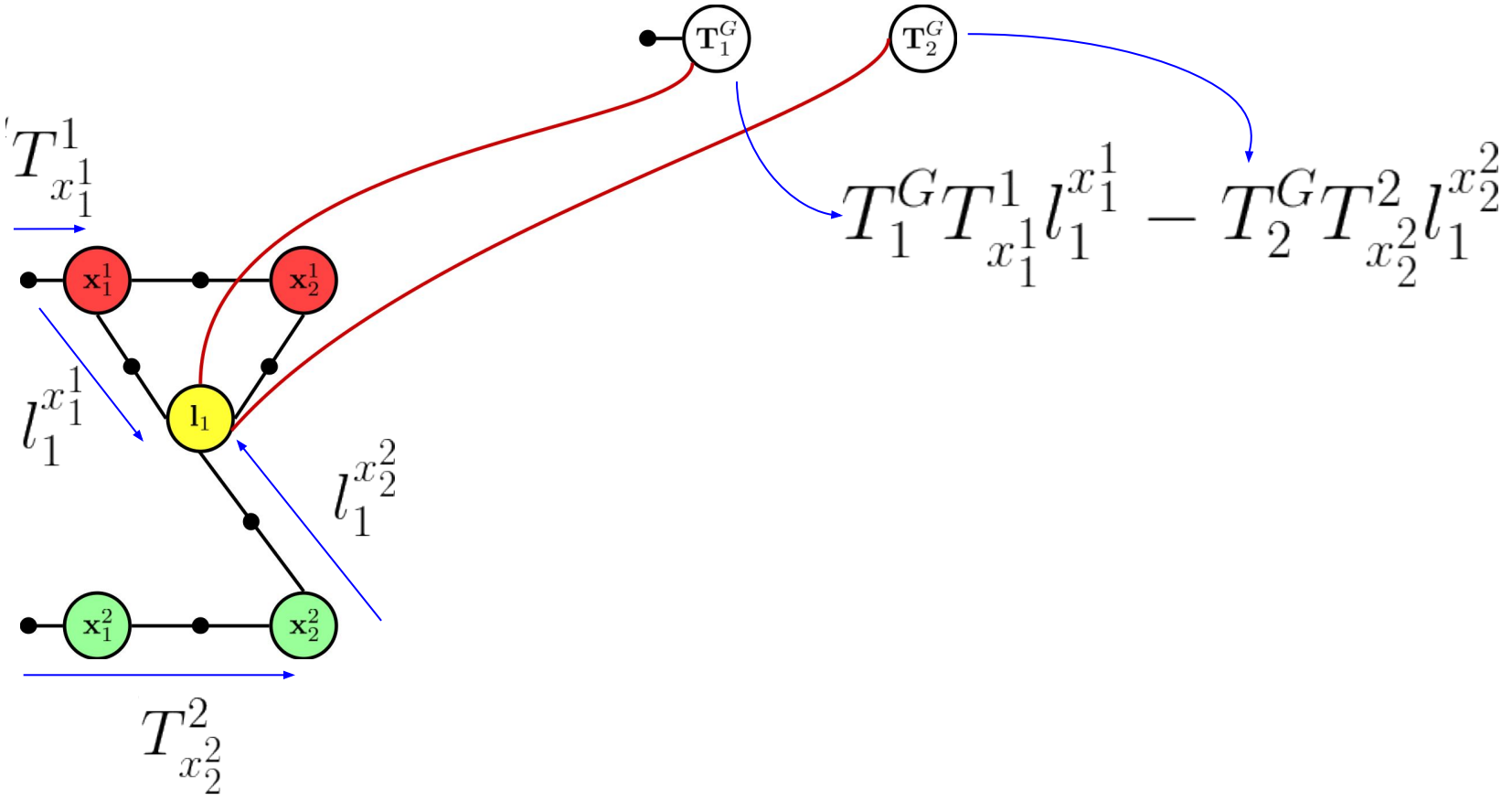
Relative Pose Graph Initialization



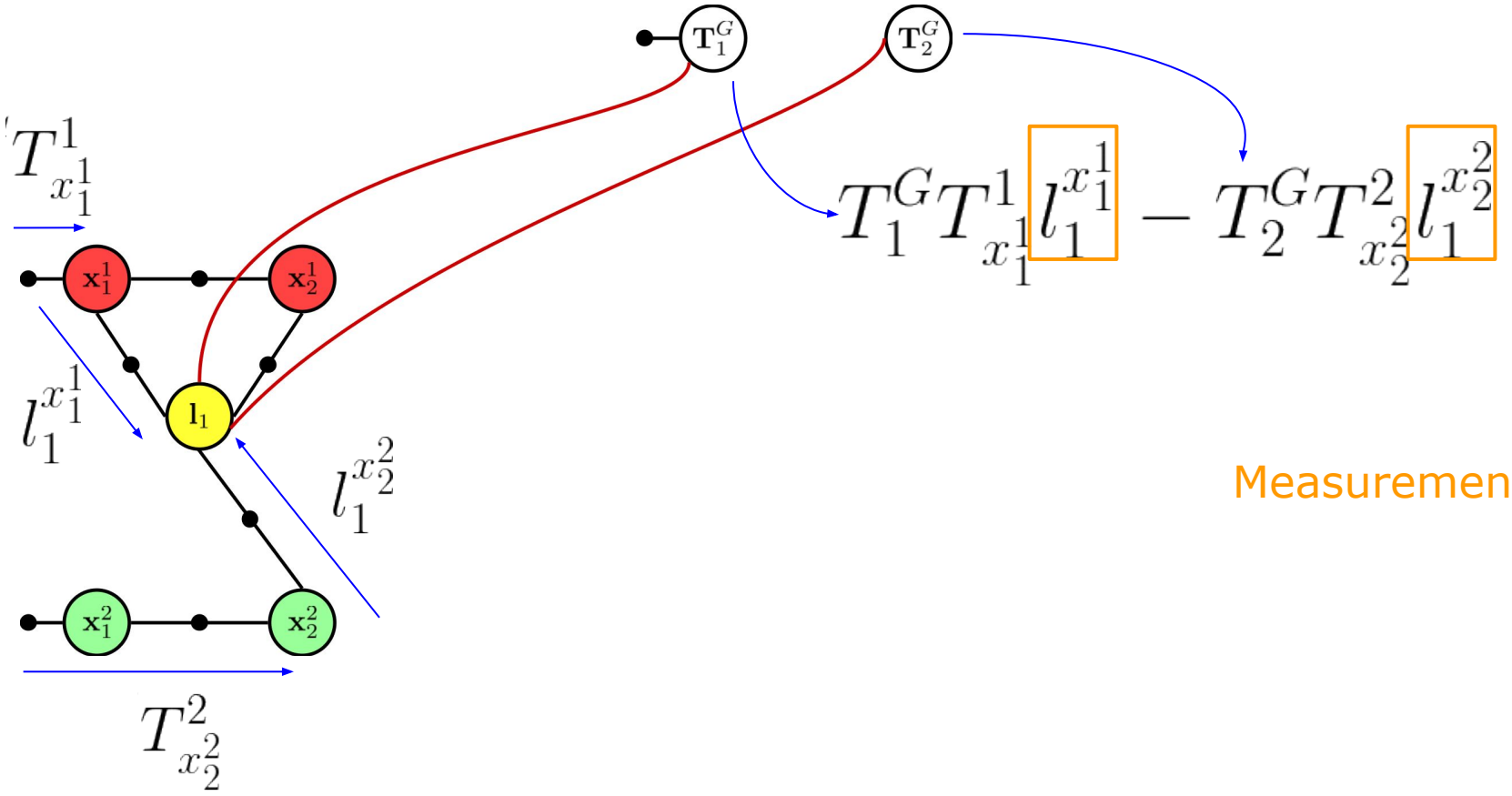
Relative Pose Graph Initialization



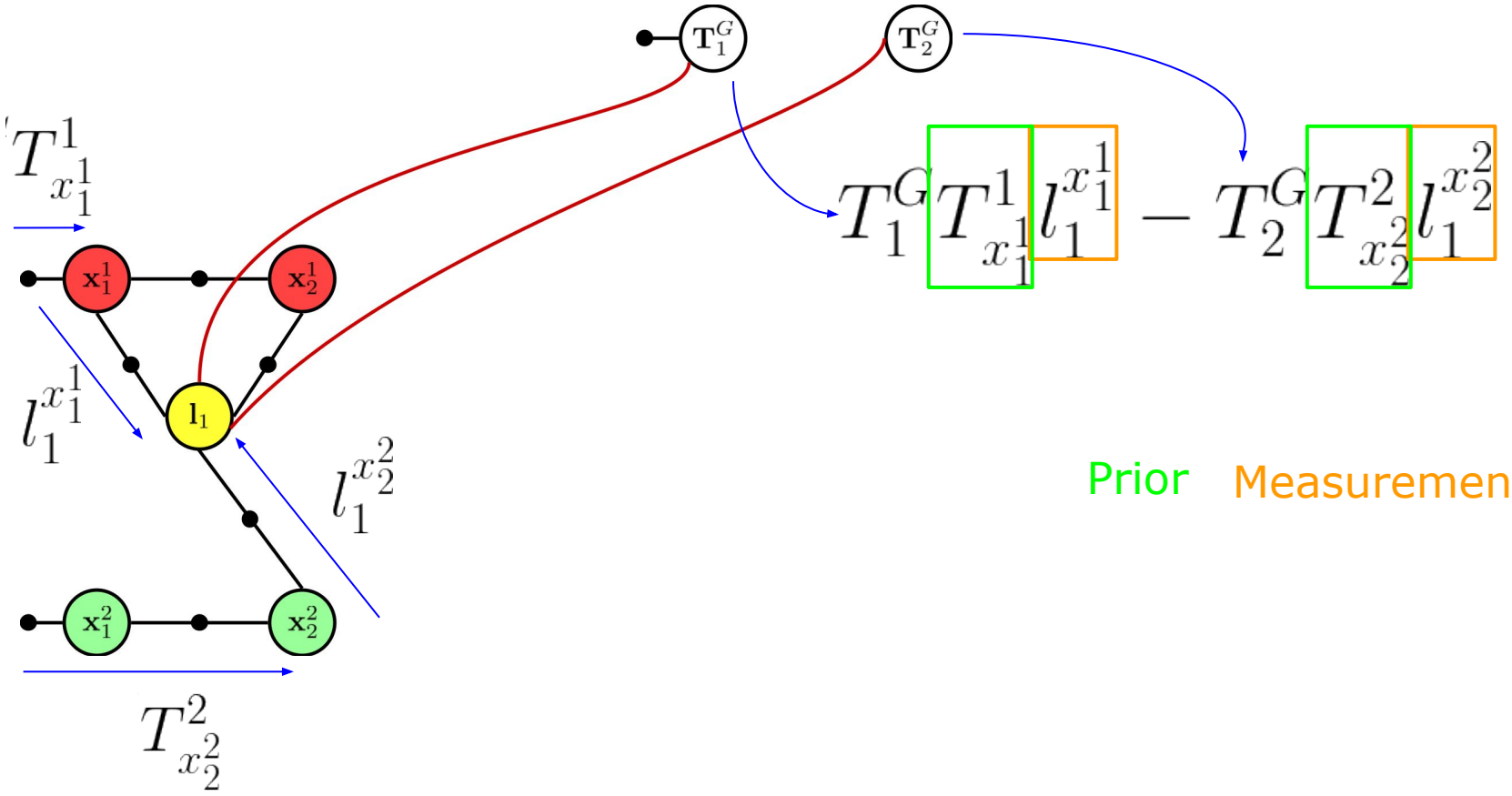
Relative Pose Graph Initialization



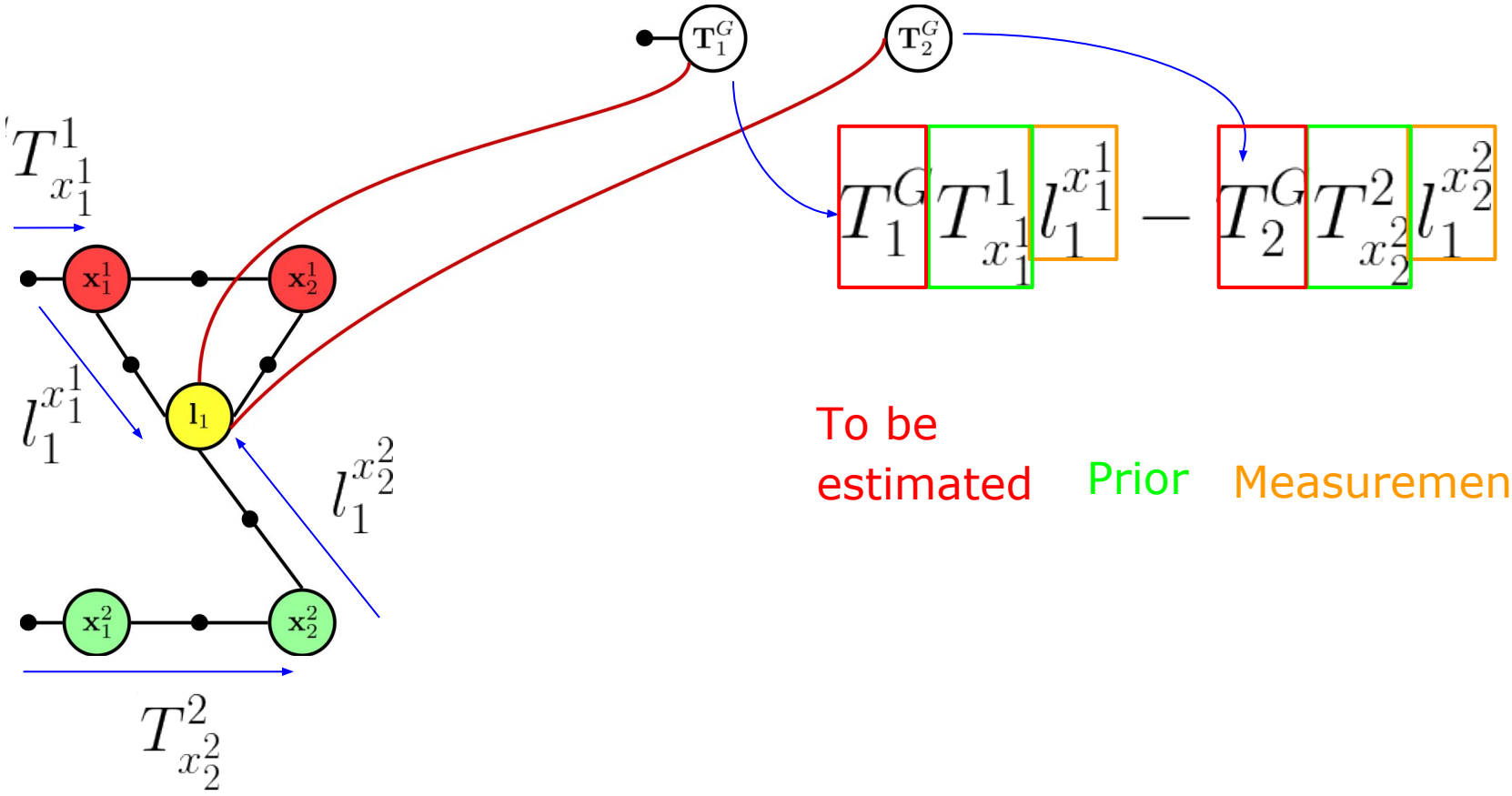
Relative Pose Graph Initialization



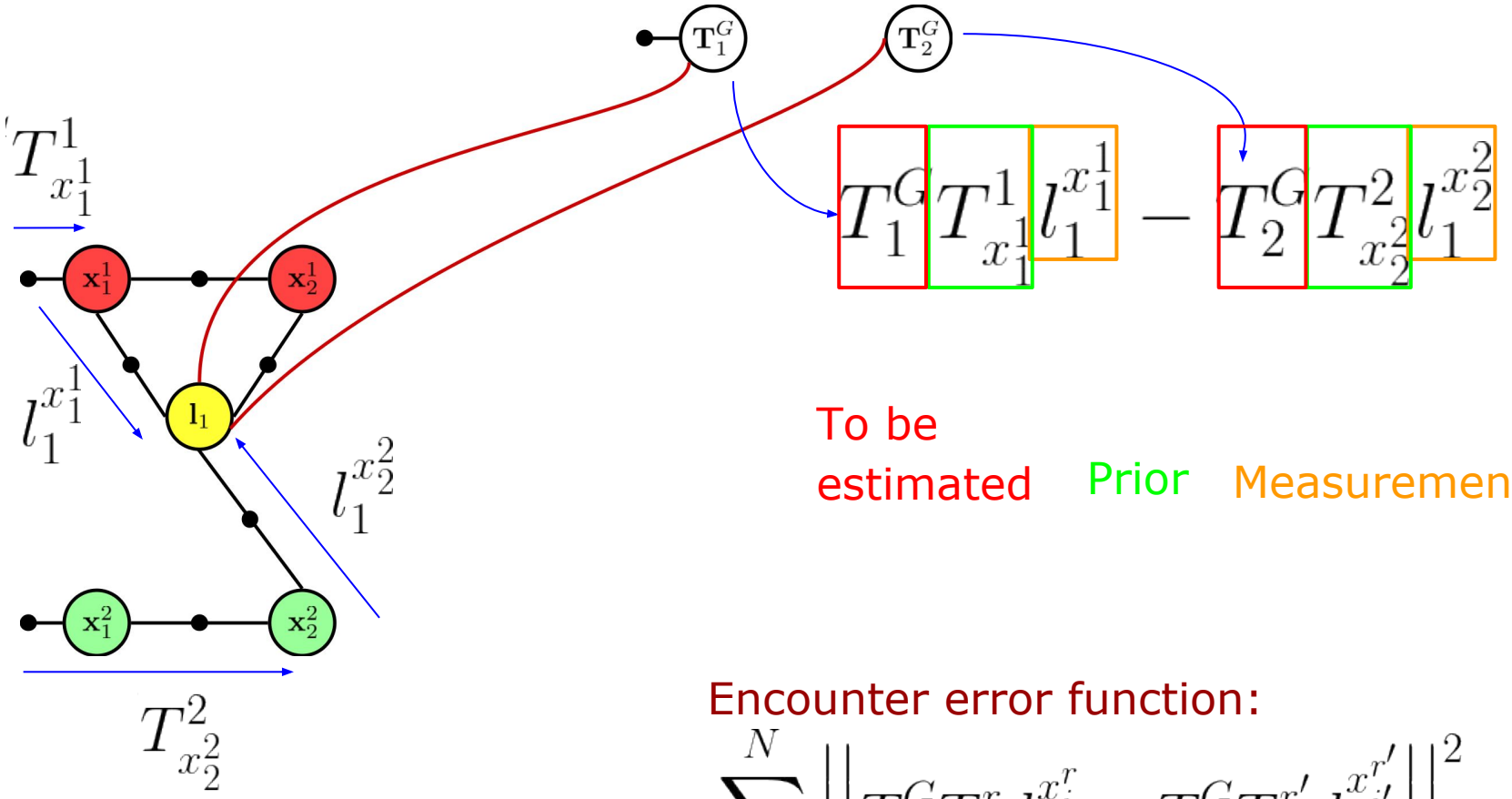
Relative Pose Graph Initialization



Relative Pose Graph Initialization



Relative Pose Graph Initialization



Encounter error function:

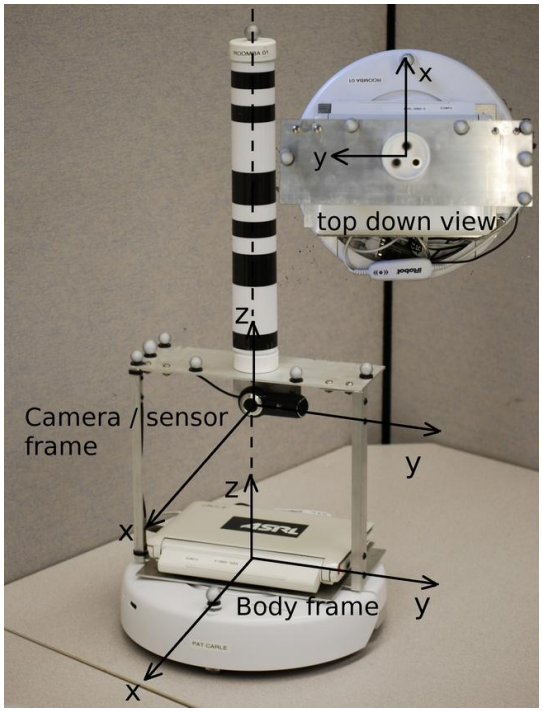
$$\sum_{n=1}^{N} \left|\left| T_r^G T_{x_i^r}^r l_n^{x_i^r} - T_{r'}^G T_{x_{i'}^{r'}}^{r'} l_1^{x_{i'}^{r'}} \right|\right|_{\Gamma_n}^2$$

Outline

- Motivation
  - Factor Graphs
  - Mapping History
- Factor Graph SLAM
  - Variable Ordering
  - Bayes Trees
- Efficient Factor Graph Fusion
  - Fusion Ordering
  - Formal Verification
- Relative Pose Graphs
- Results

Experimental Results

- AP Hill multi-robot dataset
- 4 robots
  - Odometry (0.40 m/sec)
  - LIDAR (20m range with 1080 indices)
  - Camera
  - IMU
- Fiducial on top of each robot



Experimental Results

- Mapping is an *engineering* problem

Experimental Results

- Mapping is an *engineering* problem
- Variables to be tuned
  - Odometry model parameters - 3
  - Odometry velocity model parameters - 2
  - Scan matching parameters - ~20
  - Fiducial detection classifier parameters - ~5
  - Odometry covariance - 3
  - Odometry velocity model covariance - 3
  - Fiducial detection covariance - 3
  - Encounter covariance - 3
  - Scan matching keyframe size - 1
  - Keyframe type - 3
  - Loop closure covariance - 3
  - Optimizer update rate - 1

Experimental Results

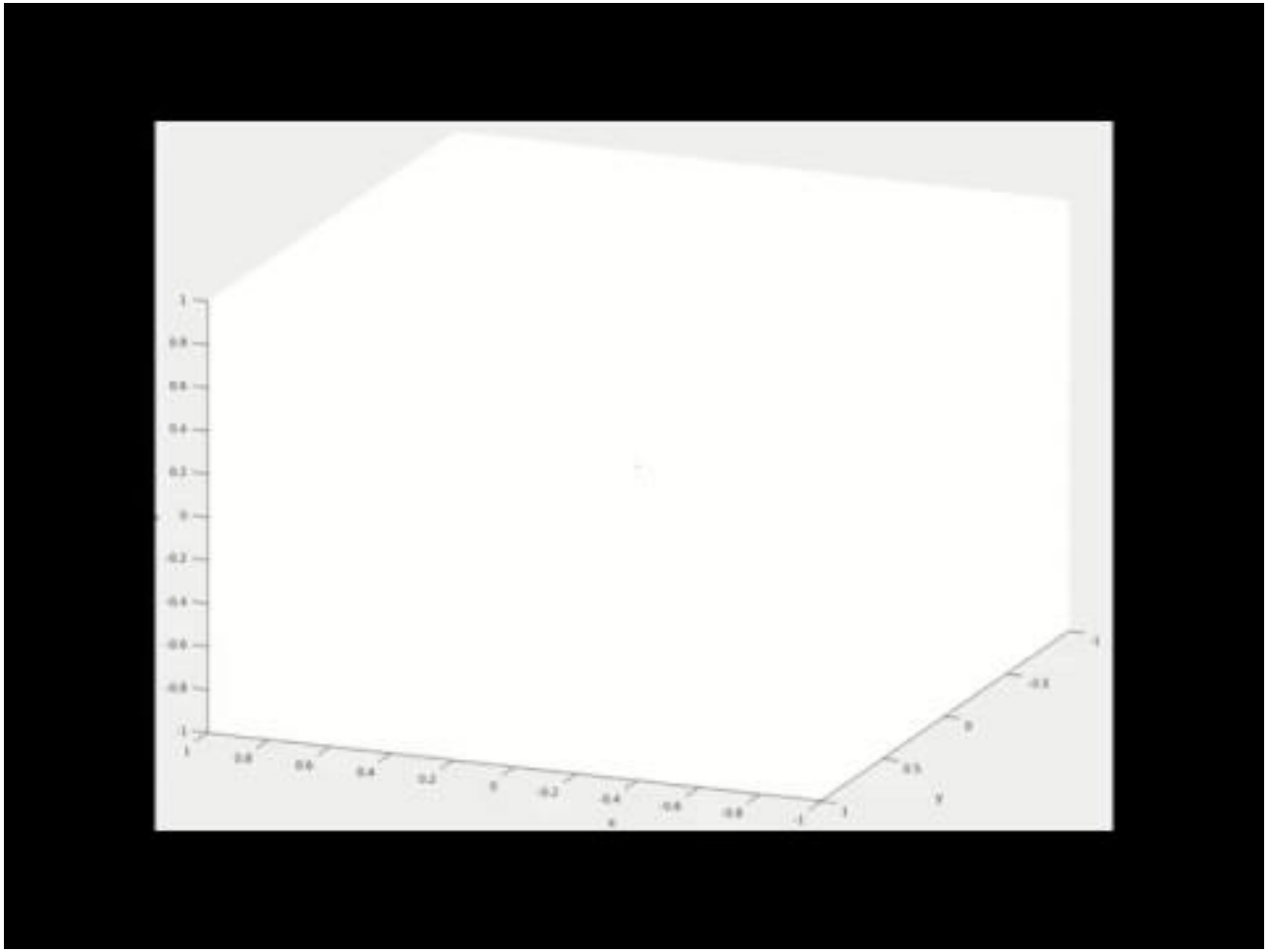
- Mapping is an *engineering* problem
- Variables to be tuned
  - Odometry model parameters - 3
  - Odometry velocity model parameters - 2
  - Scan matching parameters - ~20
  - Fiducial detection classifier parameters - ~5
  - Odometry covariance - 3
  - Odometry velocity model covariance - 3
  - Fiducial detection covariance - 3
  - Encounter covariance - 3
  - Scan matching keyframe size - 1
  - Keyframe type - 3
  - Loop closure covariance - 3
  - Optimizer update rate - 1

Variables/robot ~ 50

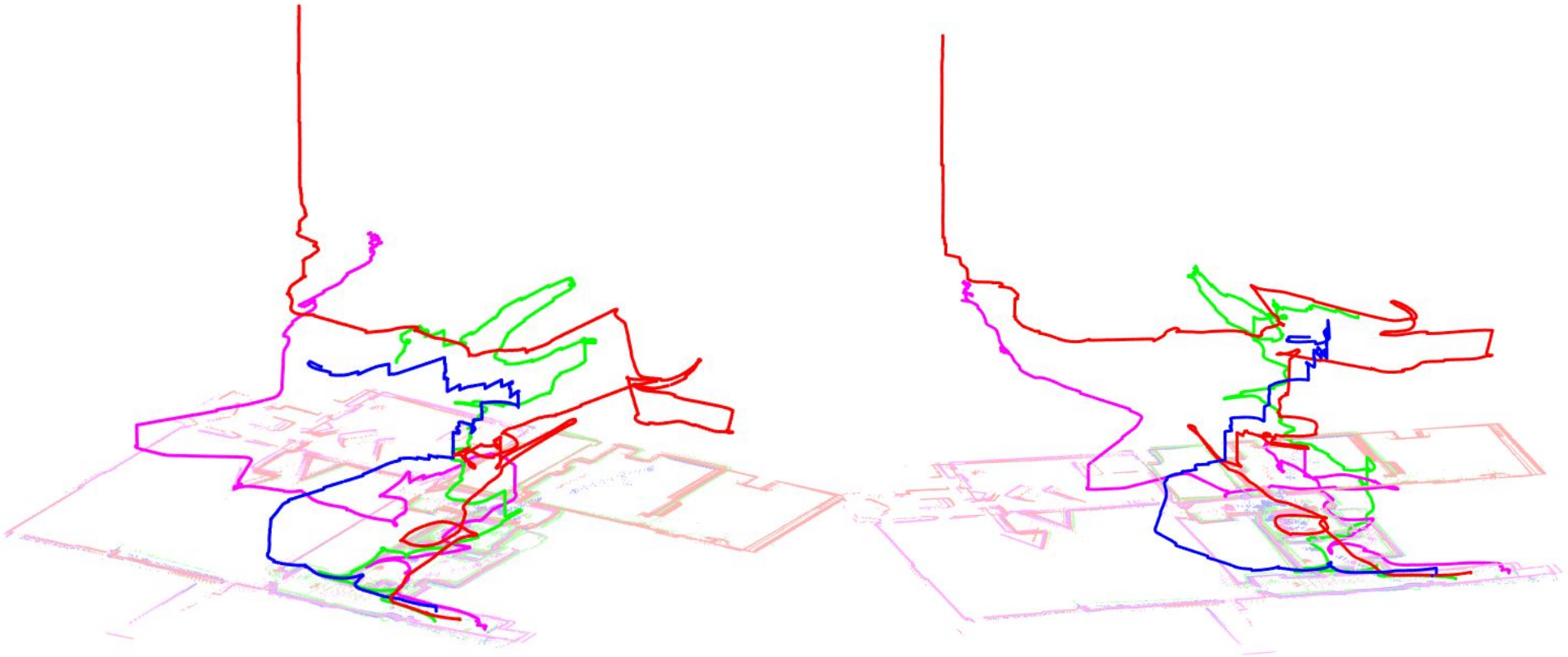
Total no. of variables

~200

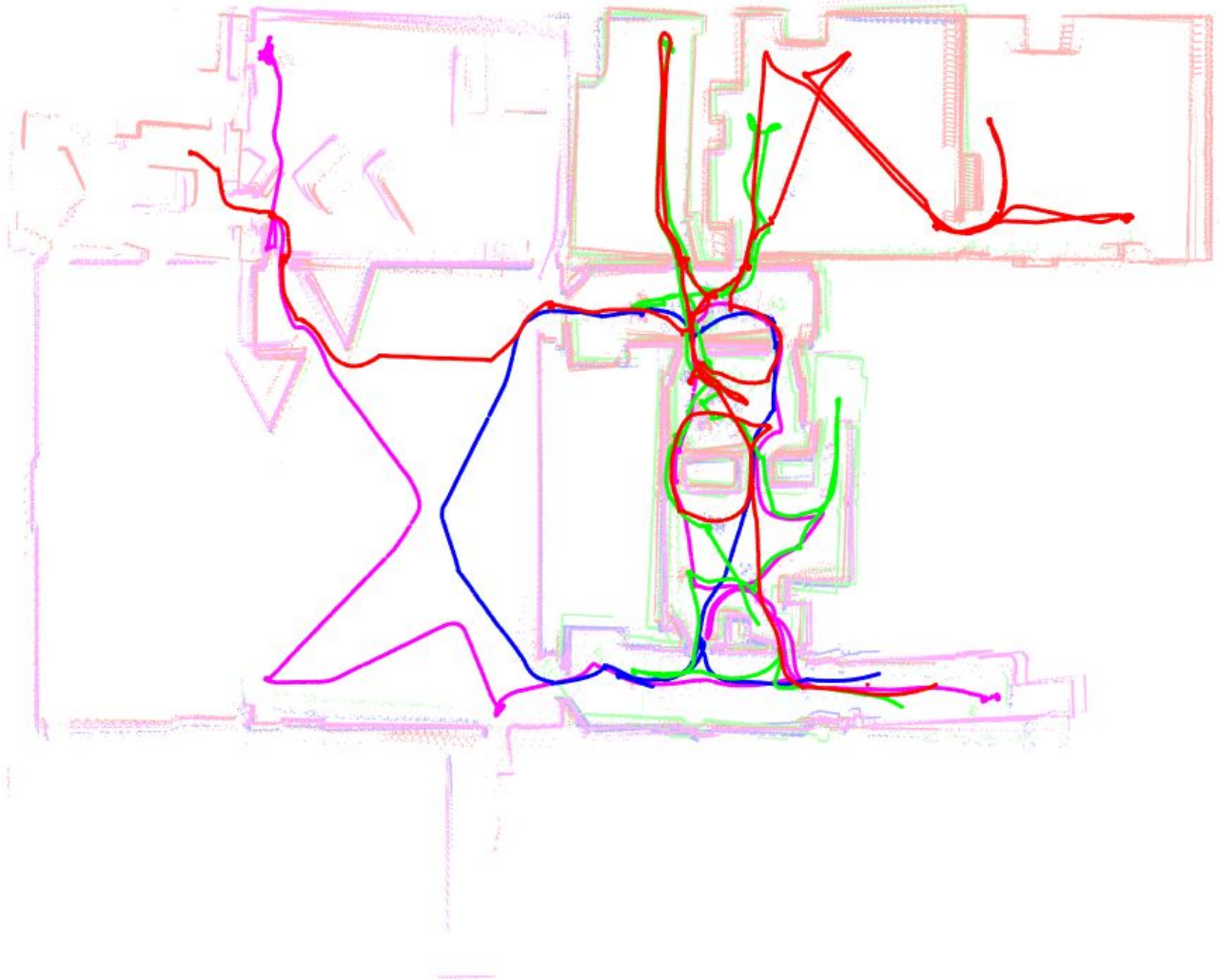
Experimental Results



Experimental Results

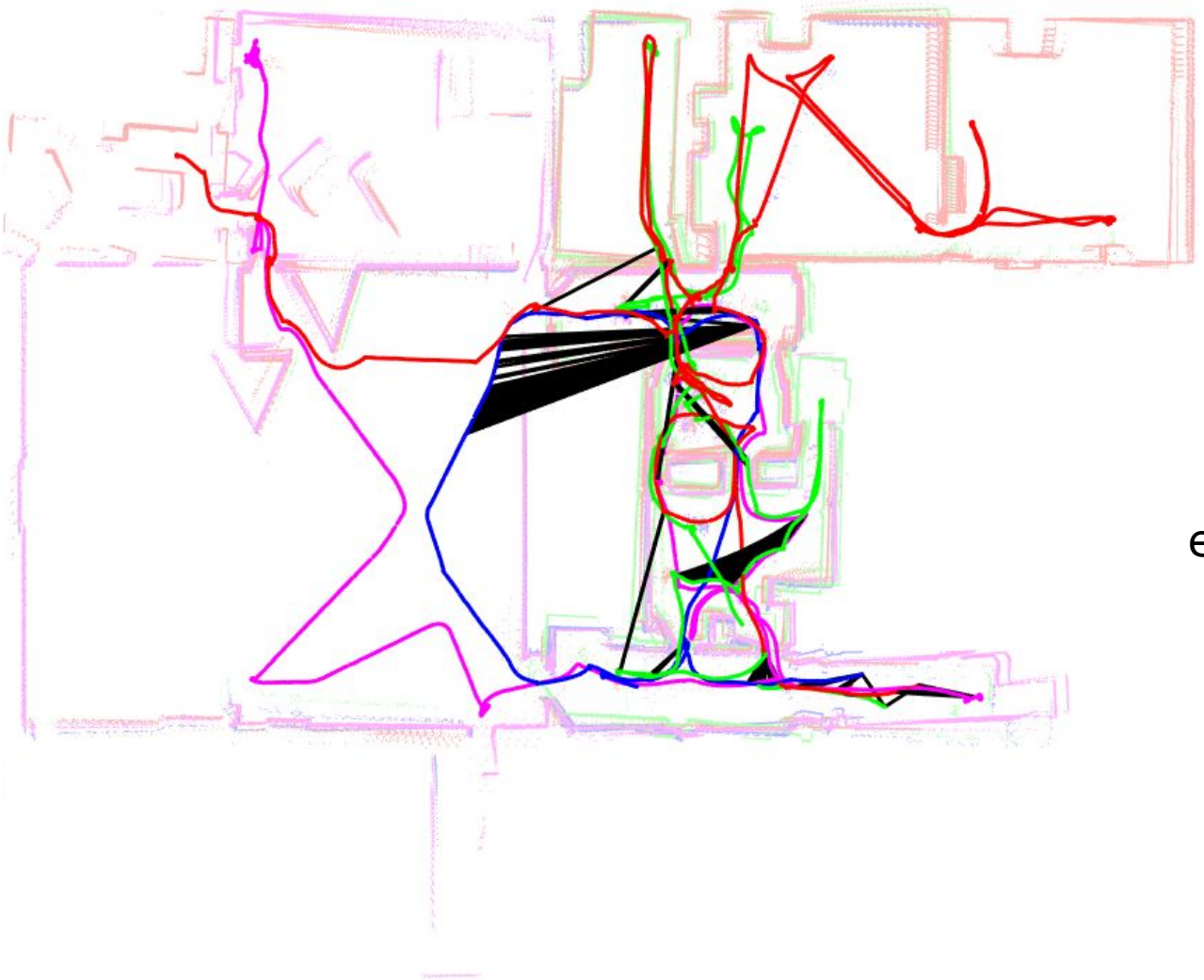


Experimental Results



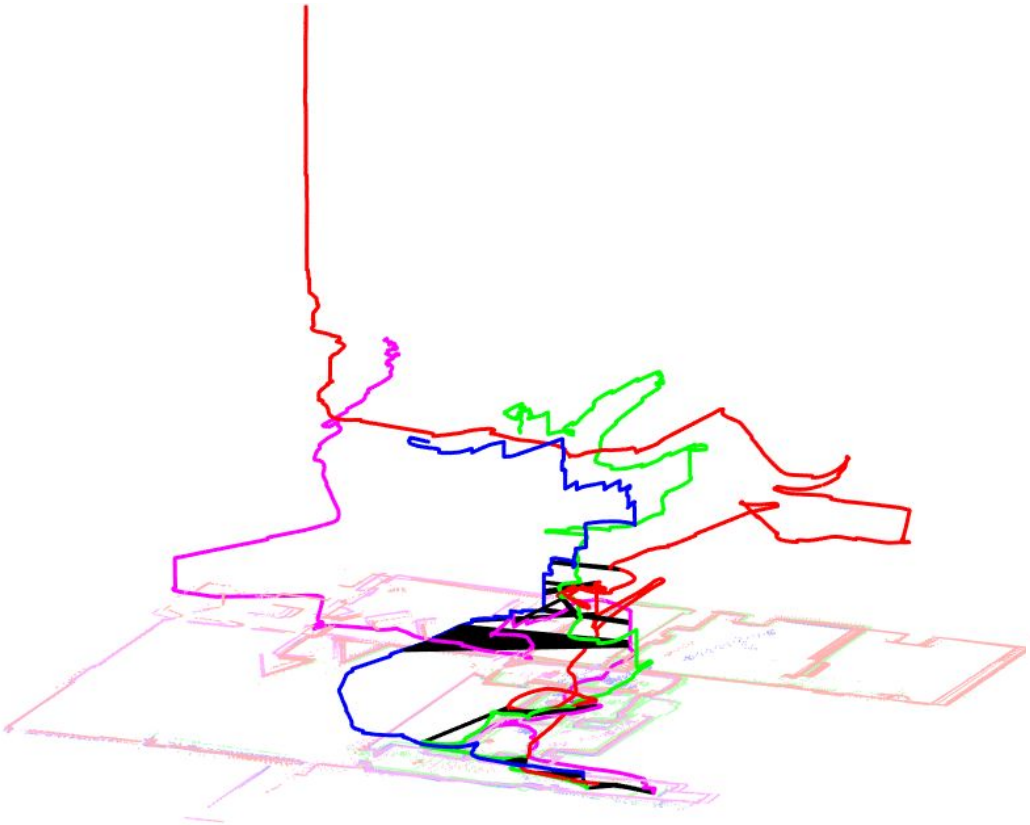
Top view

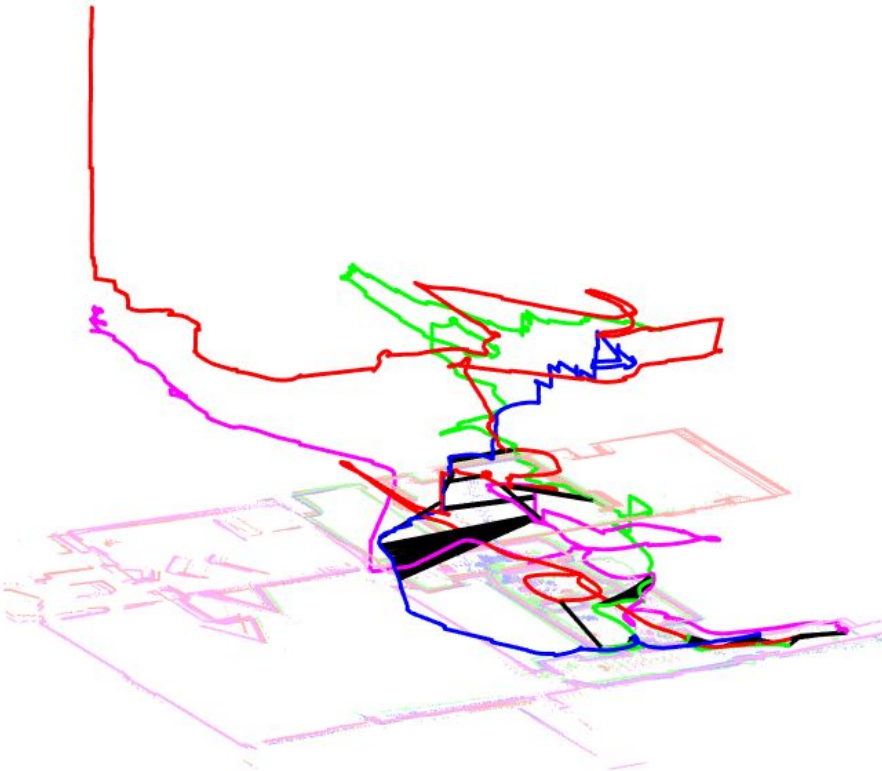
Experimental Results



Top view with encounters

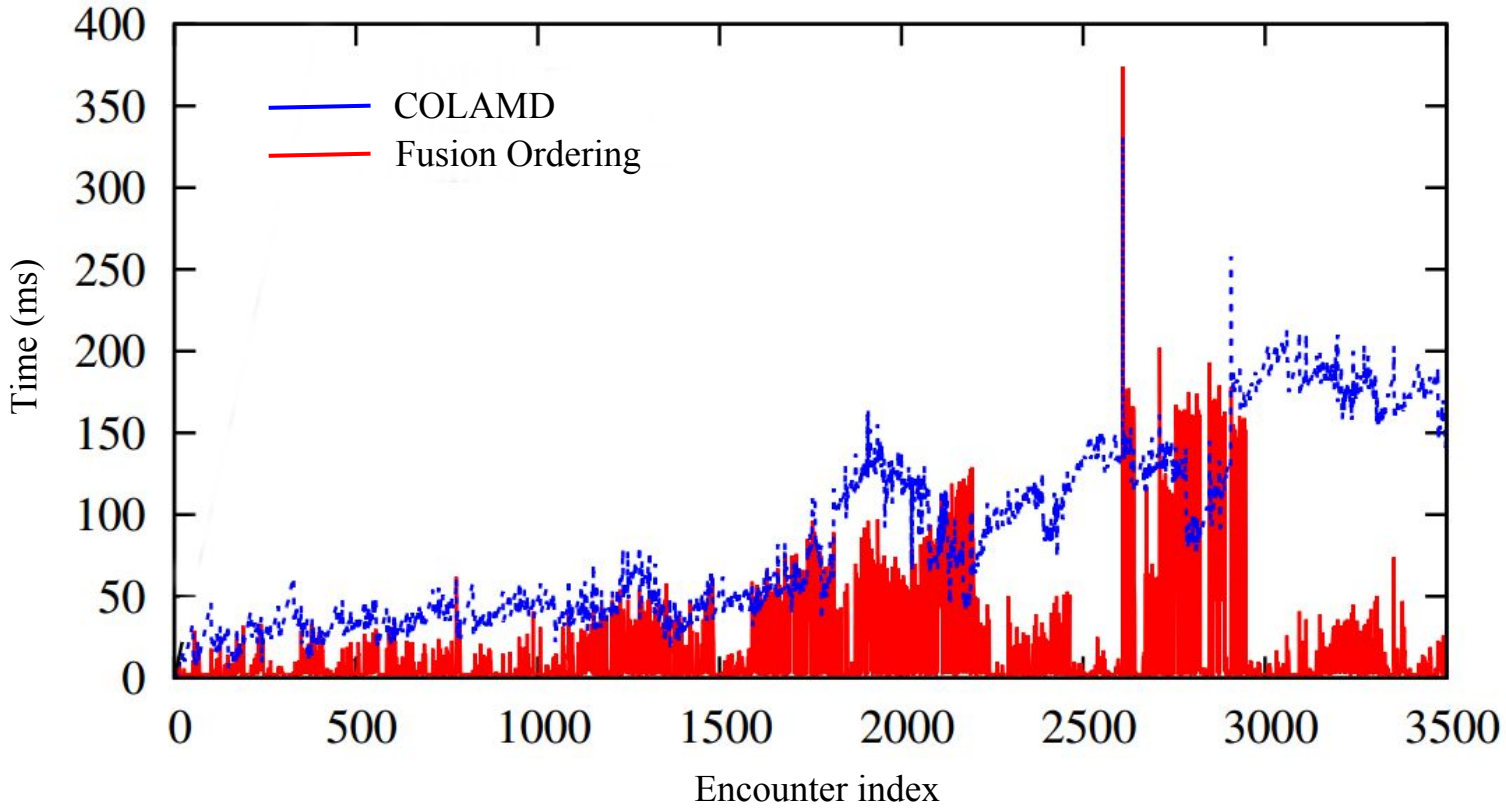
Experimental Results





Experimental Results

Time comparison between the ordering strategy



Summary

- Efficient Fusion Ordering scheme for combined graph
  - Relation with nested dissection
  - Formal verification to comply with relevant standards

Summary

- Efficient Fusion Ordering scheme for combined graph
  - Relation with nested dissection
  - Formal verification to comply with relevant standards
- Works outside SLAM. Possibility of interesting applications in:
  - Finite Element Analysis
  - Power network problem
  - Computational fluid dynamics

Summary

- Efficient Fusion Ordering scheme for combined graph
  - Relation with nested dissection
  - Formal verification to comply with relevant standards
- Works outside SLAM. Possibility of interesting applications in:
  - Finite Element Analysis
  - Power network problem
  - Computational fluid dynamics
- Relative pose graph initialization

Summary

- Efficient Fusion Ordering scheme for combined graph
  - Relation with nested dissection
  - Formal verification to comply with relevant standards
- Works outside SLAM. Possibility of interesting applications in:
  - Finite Element Analysis
  - Power network problem
  - Computational fluid dynamics
- Relative pose graph initialization
- Results on standard real-world datasets
  - SuiteSparse
  - AP hill multi-robot dataset

Future Work

- Upper bound on additional fill-in in using Fusion Ordering.
- Use Fusion Ordering for Concurrent Filtering and Mapping (CFS) **[Williams, 2014]** to extend to multiple robots.
- Investigate communication bandwidth constraints.
- Incorporate partial encounter constraints like range-only or bearing-only.

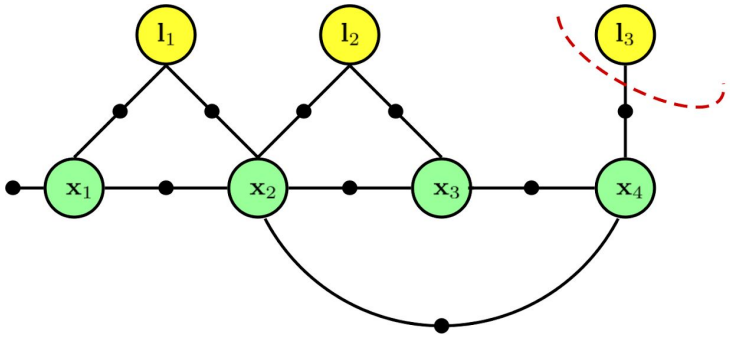
Thank you

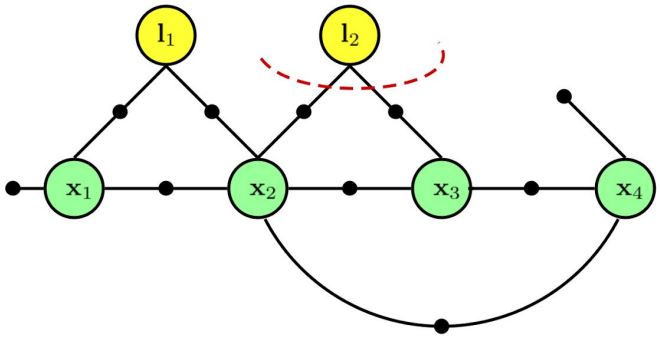
Questions?

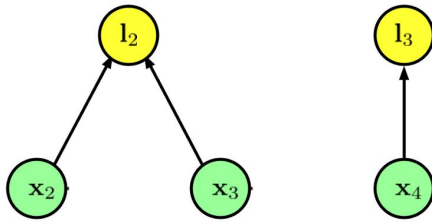
csm_params

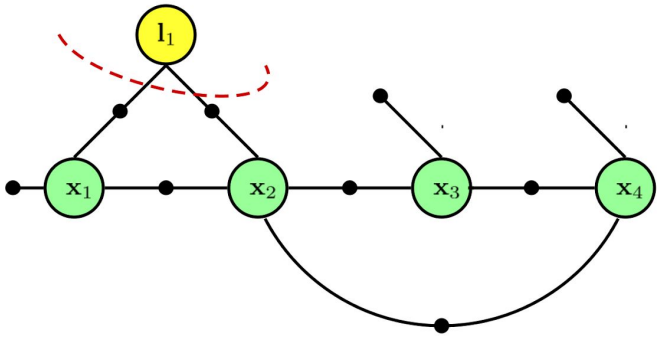
```
// Create a CSM parameters structure from the supplied configuration

sm_params csm_params;

csm_params.use_point_to_line_distance = true;

csm_params.use_corr_tricks = true;

csm_params.max_iterations = 1000;

csm_params.max_angular_correction_deg = 1.57 * 180.0 / M_PI;

csm_params.max_linear_correction = 2.0;

csm_params.max_correspondence_dist = 2.0;

csm_params.sigma = 0.05;

csm_params.epsilon_xy = 0.0001;

csm_params.epsilon_theta = 0.0001;

csm_params.outliers_maxPerc = 0.95;

csm_params.outliers_adaptive_order = 0.70;

csm_params.outliers_adaptive_mult = 2.0;

csm_params.outliers_remove_doubles = true;

csm_params.do_visibility_test = false;

csm_params.do_alpha_test = false;

csm_params.do_alpha_test_thresholdDeg = 0.35 * 180.0 / M_PI;

csm_params.clustering_threshold = 0.05;

csm_params.orientation_neighbourhood = 3;
```

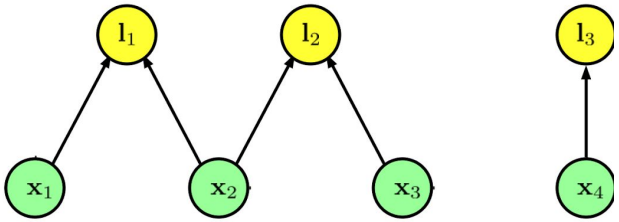
Factor Graph -> Bayes Network -> Bayes Tree



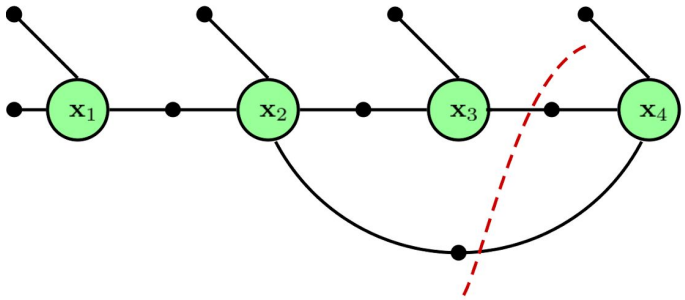


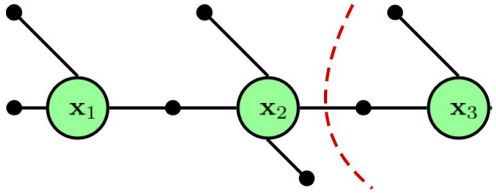


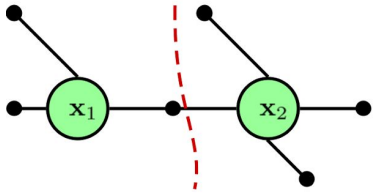


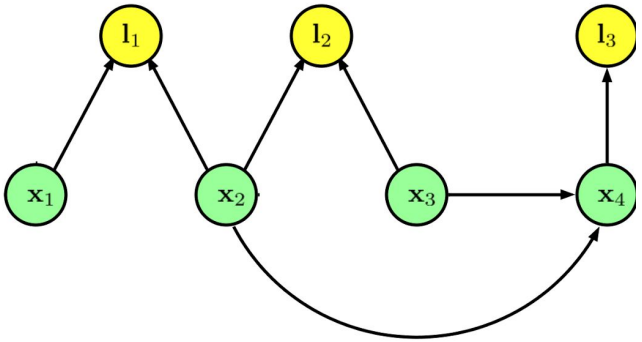


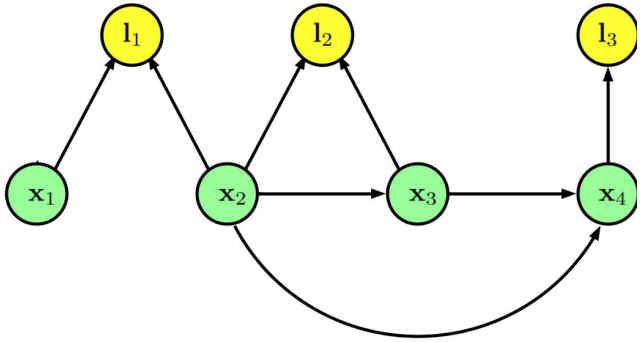
Factor Graph -> Bayes Network -> Bayes Tree

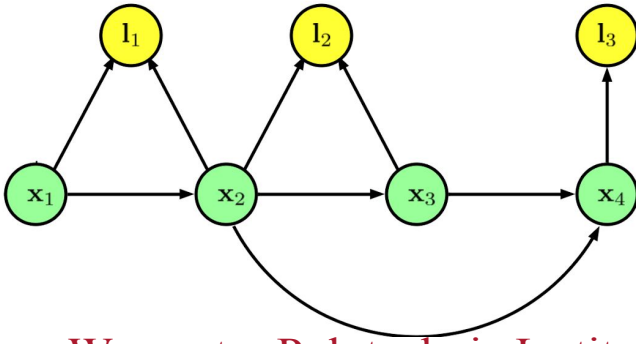




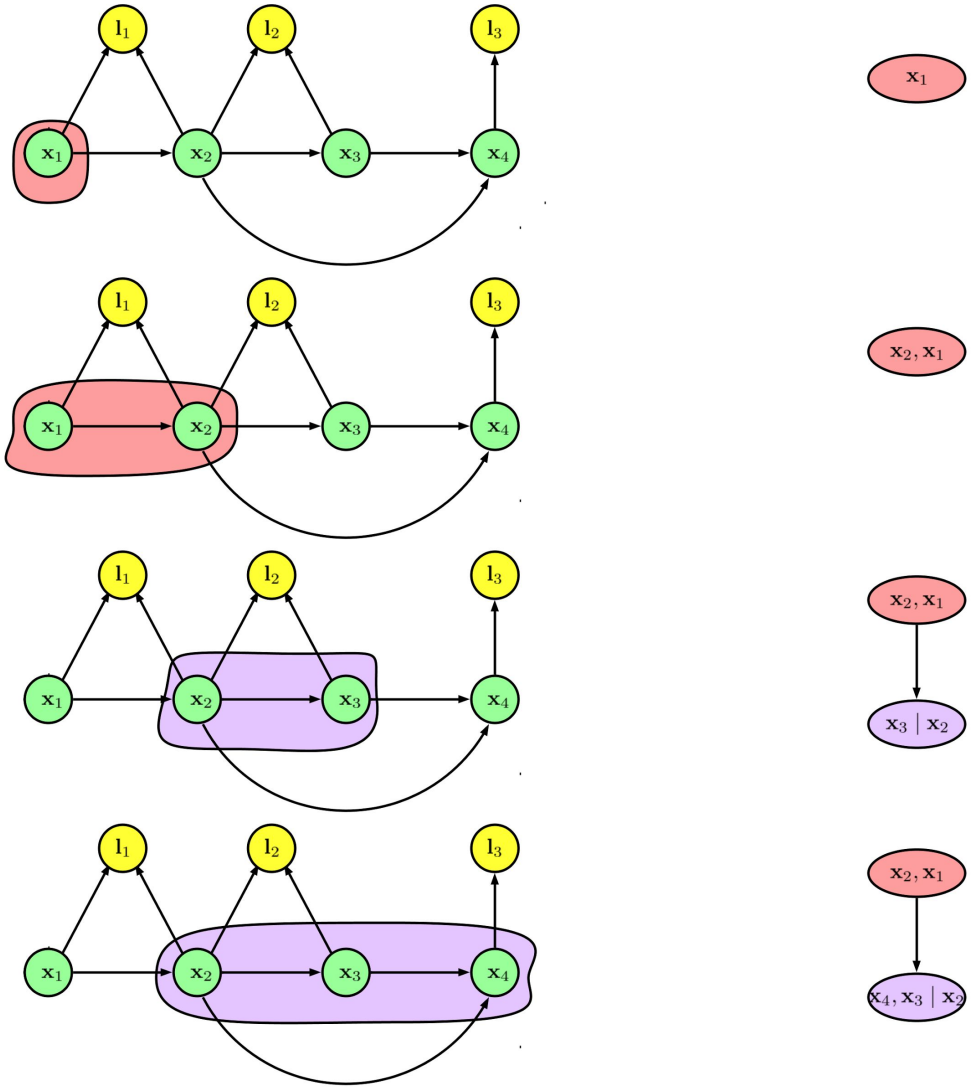




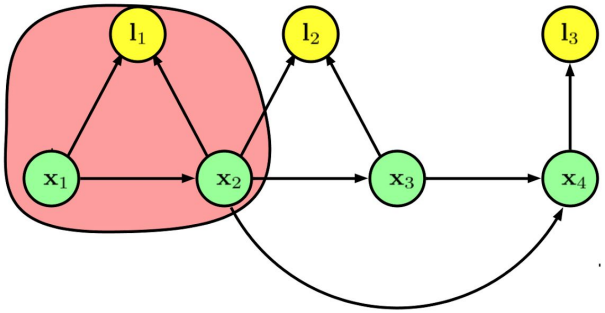


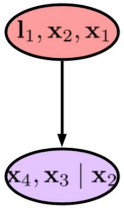


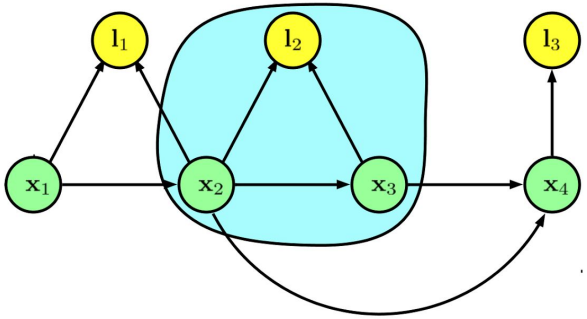
Factor Graph -> Bayes Network -> Bayes Tree

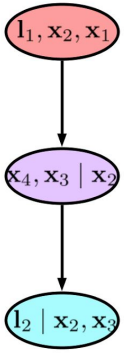


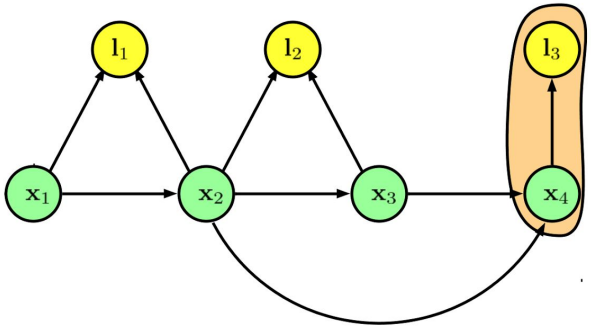
Factor Graph -> Bayes Network -> Bayes Tree

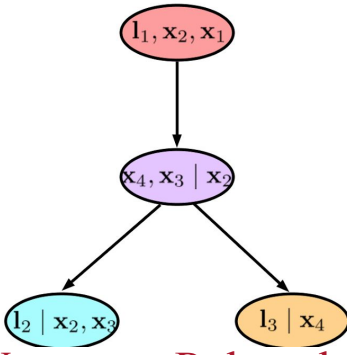












Ordering matters! - Intuition

Mapping: History & Motivation