Balanced Graph Matching

Timothee Cour   Praveen Srinivasan   Jianbo Shi

University of Pennsylvania

Many problems in computer vision can be formulated as the matching between two graphs

Contribution 1: bistochastic normalization enhances distinctive matches.
Focus matching on salient points, without explicit saliency detection.

Contribution 2: **SMAC**
Spectral method for graph **M**atching with **A**ffine **C**onstraints

A general graph matching cost:

$$\epsilon_{GM}(M) = \sum_{ii' \in M, jj' \in M} W(i,i',j,j')$$

$x_{ii'} = 1$ for a match $ii' \in M$

Integer Quadratic Programming (IQP) formulation:

$$\max \quad \epsilon(x) = x^T W x \quad \text{s.t.} \quad Cx \le b, \quad x \in \{0,1\}^{nn'}$$

$Cx \le b$: degree constraint (1-1, 1-many,...)

W encodes how well a match (i,i') between 2 graphs G,G' is compatible to another match (j,j') (see figure below)

In image matching, $W(ii',jj')$ is high if
1) feature point i is similar to i', j is similar to j', and
2) Spatial distance dist(i,j) ~= dist(i',j')

$W(ii',jj')$ can be reordered (permuting indexes) into $S(ij,i'j')$ to reflect the similarity between edges (ij) and $(i'j')$

Spectral Matching with Affine Constraints

$$\max_x \frac{x^T W x}{x^T x} \quad \text{s.t.} \quad Cx = b$$

EQUIVALENT to IQP for x binary

Linear Constraint: $Cx = 0 \implies$ Yu and Shi, 2001

Affine Constraint: $Cx = b \iff \sum_{i'} x_{ii'} = 1$ and $\sum_i x_{ii'} = 1$

Inequality Constraint ? $Cx \le b$ NP-HARD (cf AISTATS 07, submitted)

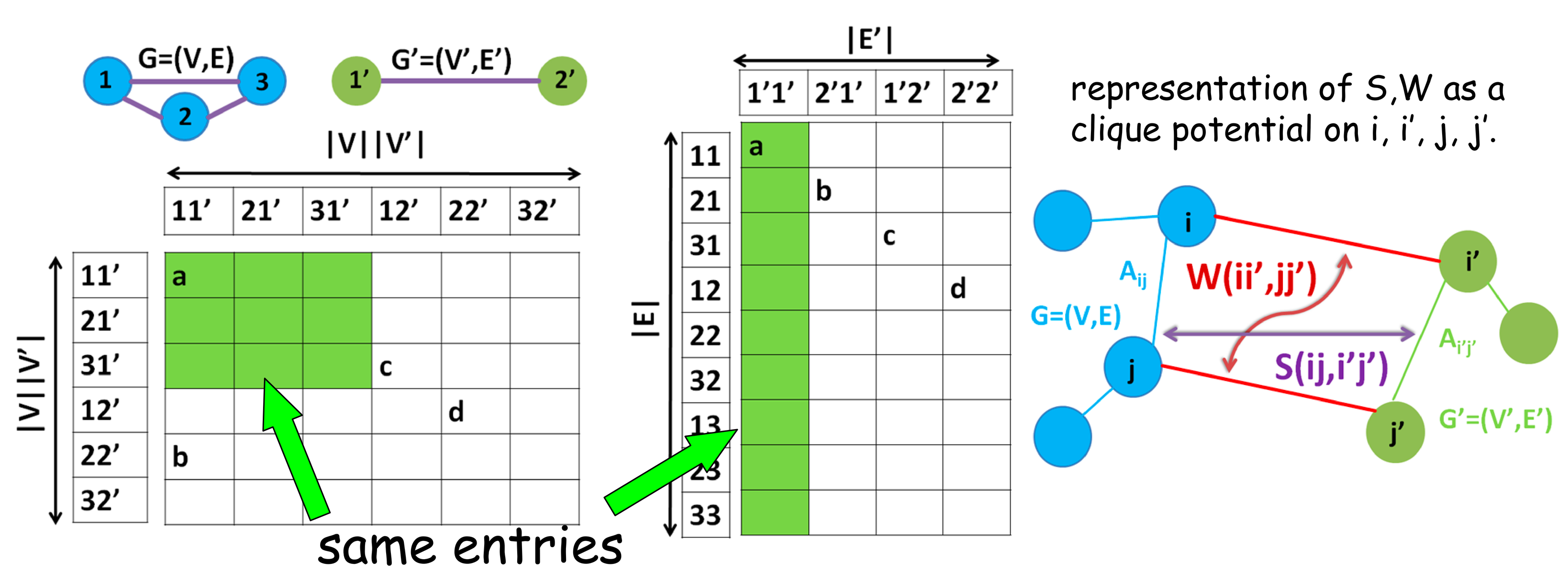
Solution

1. rewrite as $\max_{x,t} \frac{x^T W x}{x^T x}$ s.t. $Cx = tb$ — linear, but ill defined: denominator is not $(x,t)^T(x,t)$
2. introduce $C_{eq} = I_{k-1,k}(C - (1/b_k) b C_k)$ $\quad P = I - C_{eq}^T (C_{eq} C_{eq}^T)^{-1} C_{eq}$
   Efficient computation with Shermann-Morrison formula
3. solve $P \cdot W \cdot P \quad x = \lambda x$

Optimality bounds (cf AISTATS 07, submitted)

Balanced Graph Matching

Dual representation: Matching Compatibility W vs. edge Similarity S



Given matching compatibility **W**, we want to **S** to be bistochastic

Step 1. Convert W to S: $S_{ij,i'j'} = W_{ii',jj'}$

Step 2. repeat until convergence

(a) normalize the rows of S: $S^{t+1}_{ij,i'j'} := S^t_{ij,i'j'} / \sum_{k'l'} S^t_{ij,k'l'}$

(b) normalize the columns of S: $S^{t+2}_{ij,i'j'} := S^{t+1}_{ij,i'j'} / \sum_{kl} S^{t+1}_{kl,i'j'}$

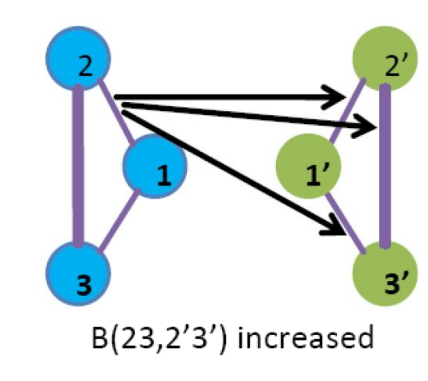
Theorem: iterated row & column normalization converges to unique balancing weights (D,D') s.t. DSD' rectangular bistochastic

Step 3. Convert back S to W

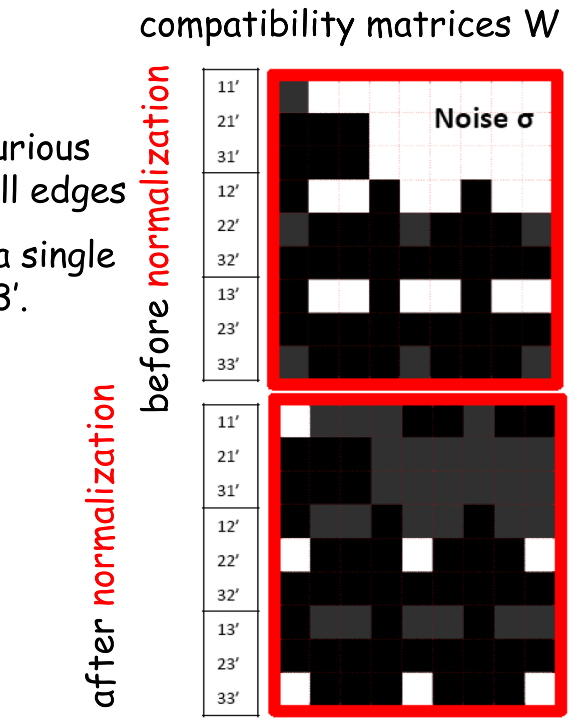
Step 4. apply SMAC (or SDP, GA, or your favorite) to W

edges 12, 13 are uninformative: spurious connections of strength sigma to all edges

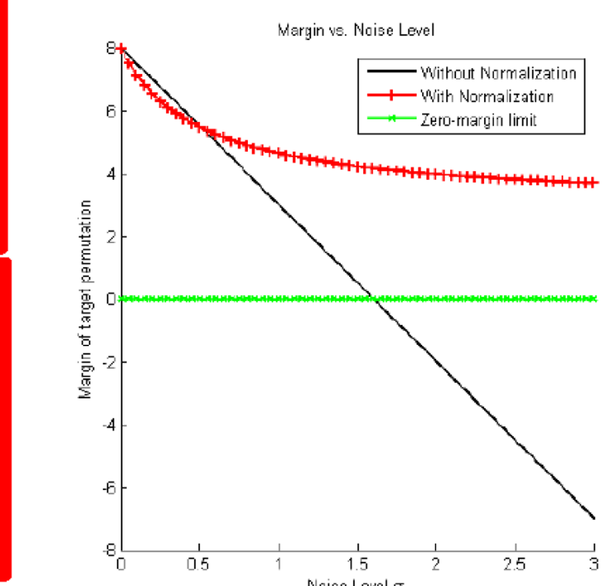
Edge 23 is informative and makes a single connection to the second graph, 2'3'.



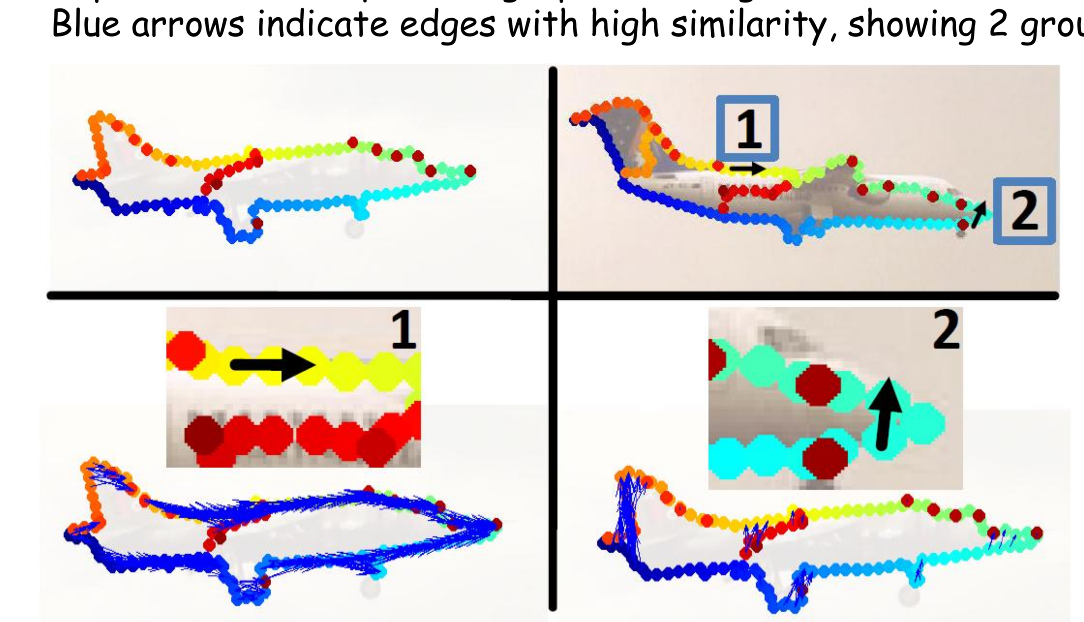
compatibility matrices W



margin as a function of noise (difference between correct matching score and best runner-up score).



Representative cliques for graph matching.
Blue arrows indicate edges with high similarity, showing 2 groups:

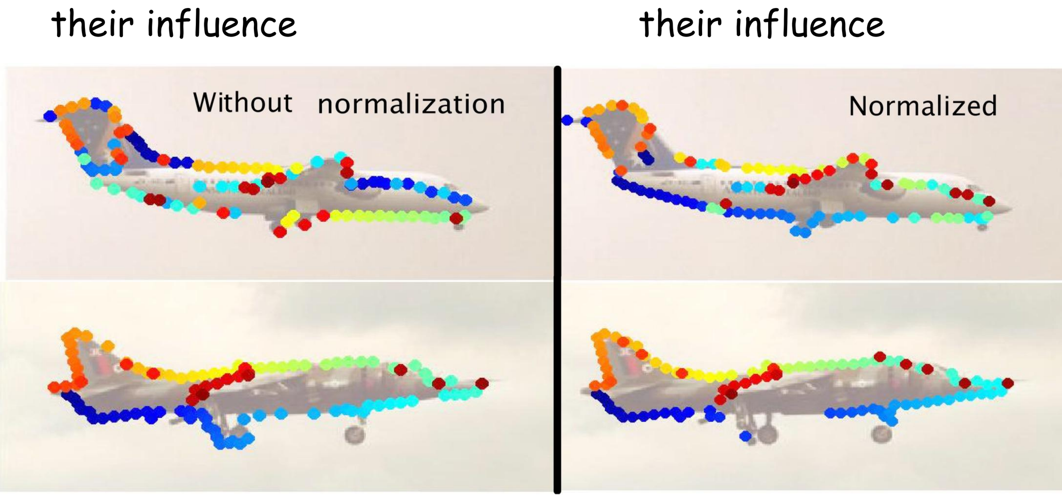


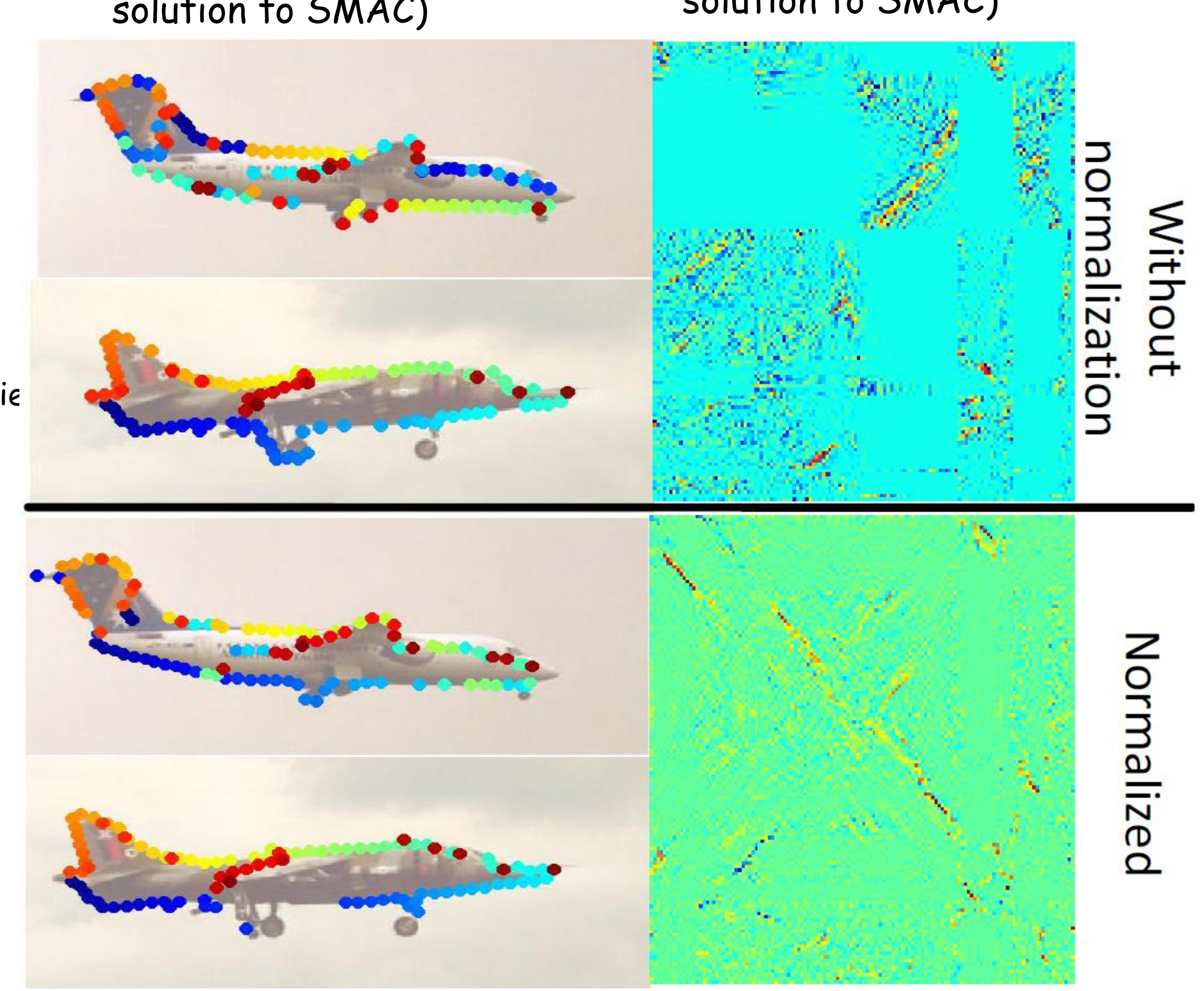
cliques of type 1 (pairing common edges in the 2 images) are uninformative

cliques of type 2 (pairing salient edges) are distinctive

normalization decreases their influence

normalization increases their influence

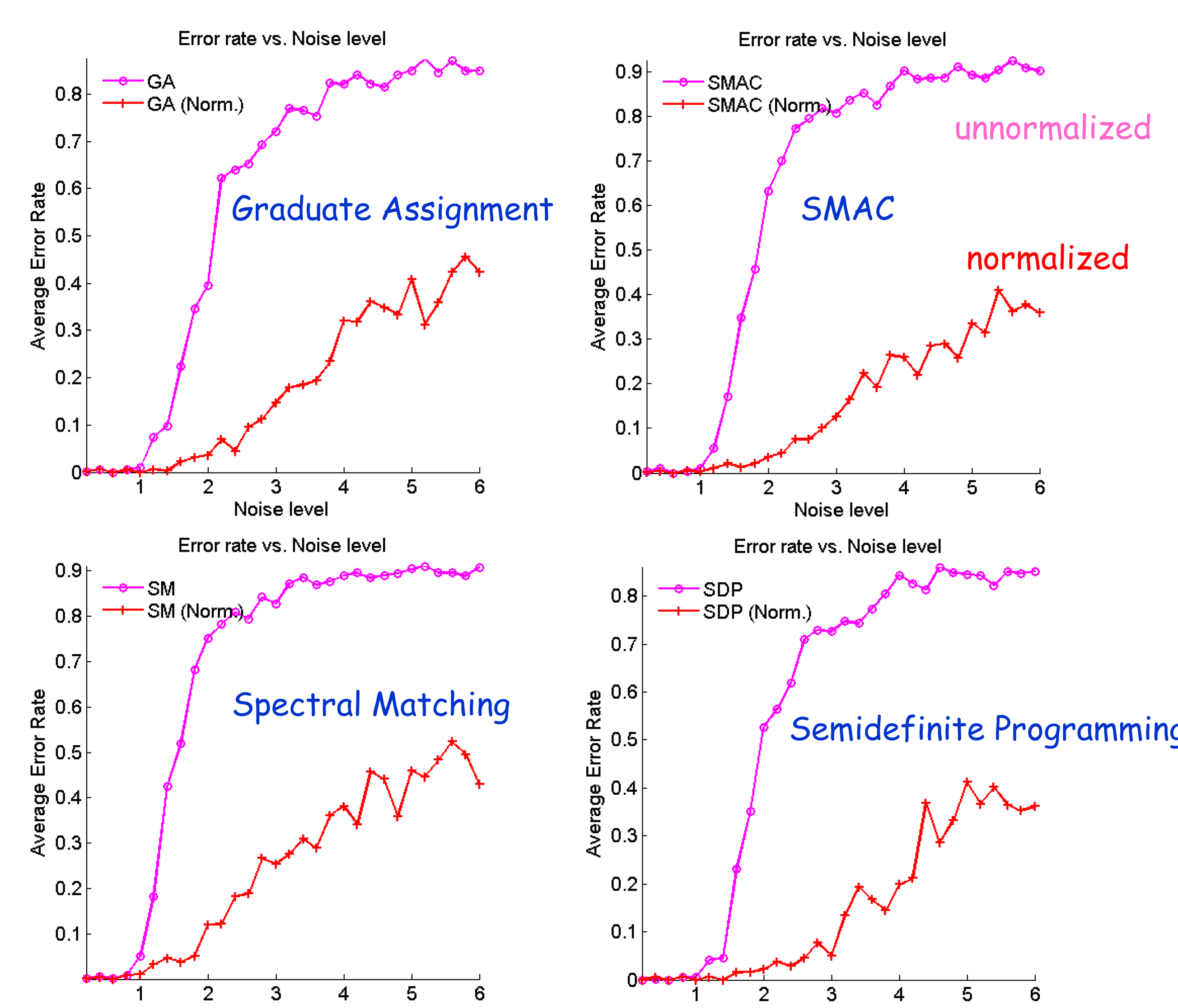




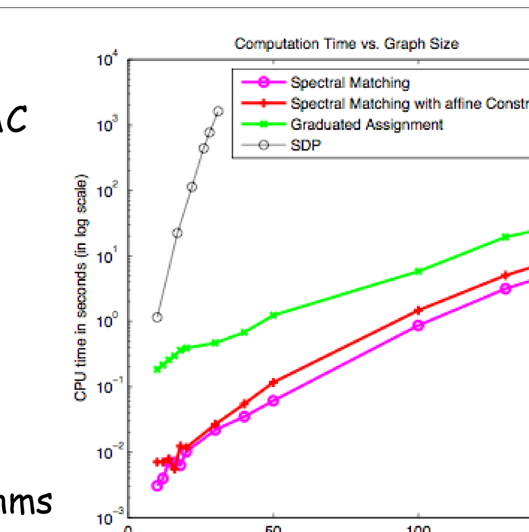
Experiments on 1-1 matchings with random graphs

Comparison of matching performance with normalized and unnormalized **W**

Axes are error rate vs. noise level



Running on GA, SDP, SM, SMAC



error rate across algorithms