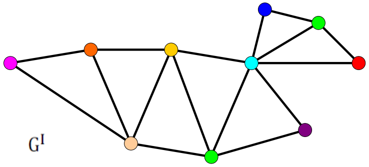
Agenda

Attributed undirected graph

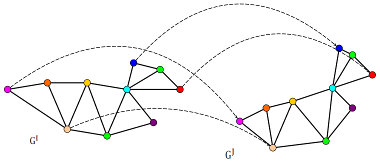
Attributed undirected graph $G = (V, E, D)$

- set of nodes $V = \{v_i\}_{i=1}^{n}$
- set of edges $E \subseteq \{\{u, v\} | u, v \in V\}$
- node attributes $D = \{d_i\}_{i=1}^{n}$, $D \subset \mathbb{R}^r$



Graph matching

Let us consider two undirected attributed graphs $G^I = (V^I, E^I, D^I)$ and $G^J = (V^J, E^J, D^J)$:



A matching function between G^I and G^J is a mapping

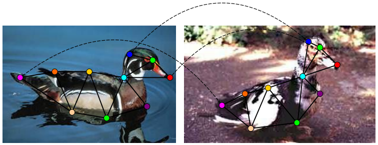
$$m : V^I \rightarrow V^J$$ not unique!

Define a function $S(G^I, G^J, m)$ to measure the quality of matching m that fulfills some constraints

$\Rightarrow$ Graph matching problem between G^I and G^J

$$m = \operatorname*{argmax}_{\hat{m}} S(G^I, G^J, \hat{m})$$

Graph matching in computer vision



- image matching
- shape matching
- object detection
- object tracking
- . . .

Graph matching

A matching function between G^I and G^J is a mapping

$$m : V^I \rightarrow V^J$$

Graph matching problem between G^I and G^J

$$m = \underset{\hat{m}}{\operatorname{argmax}} \, S(G^I, G^J, \hat{m})$$

Depending on the required properties of a matching one distinguishes

- exact graph matching
- inexact graph matching

Exact graph matching I

Edge preserving mapping *m*: $\{v_i, v_{i'}\} \in E^I \Rightarrow \{m(v), m(v_{i'})\} \in E^J$

- mapping *m* is bijective $\rightarrow$ graph isomorphism (GI)

  subgraph isomorphism, maximal common subgraphs

- mapping *m* is injective $\rightarrow$ graph monomorphism

  second graph can contain additional nodes and edges

- mapping *m* is total $\rightarrow$ graph homomorphism

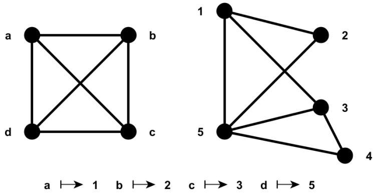
  every node has to be mapped

NP complete (except GI) (?)

Exact graph matching II

Exact graph matching:

- too strict
- time/memory consuming
- cannot handle object deformation



Inexact graph matching I

Introduce similarity measure between nodes/edges in the graphs

$$m = \operatorname*{argmax}_{\hat{m}} S(G^I, G^J, \hat{m})$$

- second-order (edge) similarity $s_E(e_{ii'}, e_{jj'})$, $e_{ii'} \in E^I, e_{jj'} \in E^J$
- first-order (node) similarity $s_V(v_i, v_j)$, $v_i \in V^I, v_j \in V^J$

$$S(G^I, G^J, m) = \sum_{\substack{m(v_i)=v_j \\ m(v_i')=v_j'}} s_E(e_{ii'}, e_{jj'}) + \sum_{m(v_i)=v_j} s_V(v_i, v_j)$$

- assignment matrix $X \in \{0,1\}^{n_1 \times n_2}$, $X_{ij} = 1 \iff m(v_i) = v_j$
  vector form $x = \operatorname{vec} X$

Inexact graph matching II

The most common problem formulation:

Quadratic Assignment Problem (NP complete) (?)

$$x^* = \arg\max \sum_{\substack{x_{ij}=1 \\ x_{i'j'}=1}} s_E(e_{ii'}, e_{jj'}) + \sum_{x_{ij}=1} s_V(v_i, v_j)$$

$$s.t. \begin{cases} x \in \{0,1\}^{n_1 n_2} \\ \sum_{i=1}^{n_1} x_{ij} \leq 1 \\ \sum_{j=1}^{n_2} x_{ij} \leq 1 \end{cases}$$

Using matrix notation : $\arg\max_x x^T S x$, S is a similarity (or affinity) matrix

Inexact graph matching III

Solution techniques

- discrete optimization
  - tree search tree search with backtracking
  - simulated annealing
- continuous optimization
  - constraint relaxation Frank-Wolfe Algorithm, search direction by linear assignment problem
  - spectral methods eigenvalue decomposition of adjacency matrices, only graph geometry
  - probabilistic frameworks optimal assignment between set of labels and set of objects; EM: nodes of the first graph are considered as an observed data and the nodes of the second graph as hidden random variables
  - clustering agglomerative clustering+threshold the cluster size; additional constraints, complexity reduction, set of independent problems

Drawback of the existing algorithms

- most of the algorithms are developed for matching relatively small graphs ($\sim$ 150 nodes)
- scale badly due to the polynomial increase of time and storage demand
- algorithms for the big graphs use another formulation of the graph matching optimization problem

$$x^* = \operatorname{argmin}_X \|A^I - XA^J X^T\|^2 + \|D^I - XD^J\|_2^2$$

Aim of the master's thesis

- a novel framework that should help to extend the usability of existing graph matching algorithms to bigger graphs
- a variant of the well-known divide and conquer paradigm: subdividing initial problem into a set of smaller problems, which can be easily handled with existing algorithms
- iterative algorithm that allows to improve initial subdivision into subproblems

Complexity reduction

$$x^* = \arg\max x^T S x$$

$$s.t. \begin{cases} x \in \{0,1\}^{n_1 n_2} \\ \sum_{i=1}^{n_1} x_{ij} \leq 1 \\ \sum_{j=1}^{n_2} x_{ij} \leq 1 \end{cases}$$

- set of candidate correspondences replace initial problem with one smaller problem
- sparse affinity matrix
- subdivide problem into a set of smaller subproblems $\leftarrow$
  Similar works:
  - semisupervised case spectral co-clustering
  - another objective function minimization, relaxation labeling
  - special kind of subproblem cluster = direct neighborhood of a node

Two level graph matching framework

Lower level: initial graphs G^I, G^J
Higher level: simplified graphs (anchor graphs A^I, A^J)

1. find correspondences between nodes of the anchors graphs (higher level);
2. find for each pair of matched anchors a mapping between the nodes of the underlying subgraphs (lower level);
3. updates subgraphs;
4. runs further through steps 1 and 3 until no improvement in the overall solution is achieved in several successive iterations.

Anchor graph construction

Goal: $G^l = (V^l, E^l, D^l) \rightarrow A^l = (V^{la}, E^{la}, U^{la})$
Equivalent: partitioning of $G^l \supset (G^l_1 \cup \dots \cup G^l_{|V^{la}|})$
Done by:

- grid with r rows and c columns
- multi-level graph partitioning algorithms, 3 phases

Anchor graph and subgraph matching

Goal: find correspondences between two anchor graphs $A^I = (V^{Ia}, E^{Ia}, U^{Ia})$ and $A^J = (V^{Ja}, E^{Ja}, U^{Ja})$

- edge similarity: compare length of the edges beween anchors
- node similarity:
  - score of the matching of G_k^I and G_p^J
  - bag of features model
  - each histogram represents a distribution of the length of the subgraph edges inside a small circle region around a node
  - χ^2 statistic test

Match anchor graphs and subgraph using some existing algorithm (e.g. RRWM)

Graph partition update

1. two affine transformations $T_{kp} : V_k^I \rightarrow V_p^J$ and $T_{pk} : V_p^J \rightarrow V_k^I$(point set registration problem)
2. transformation T_{pk} is casted as $T_{pk} : V_p^J \rightarrow V^I$
   projected points $T_{pk}(v_j) \rightarrow$ nearest neighbors
   $\bar{v}_i = NN(T_{pk}(v_j))$
   a new matrix $\bar{U}^{Ia} \in \mathbb{R}^{|V^I| \times m_1}$
   $\bar{U}^{Ia}_{\bar{v}_i, a_k} = \| T_{pk}(v_j) - \bar{v}_i \|_2$ the distance between the projections and their nearest neighbor.

Evaluation

Two ways of evaluation have been used

- synthetic data
- real data

Accuracy -> recall: number of correct detected matches divided by the number of all correct matches

Synthetic data I

Performance of the 2LevelGM with non-attributed anchor graphs

To our best knowledge, those algorithms have not been applied directly to graphs with more than 150 nodes each without additional problem simplifications

1. the SM algorithm has the worst matching quality in all cases
2. 2LevelGM performs a little bit unstable. The average performance of 2LevelGM is a little bit lower than the one of IPFP and RRWM, although in individual runs 2LevelGM is not worse.
3. The running time of 2LevelGM is slower as of IPFP and RRWM.
4. The slowest algorithm in this comparison is MPM.

Synthetic data II

5. Nevertheless MPM achieved best accuracy and objective score in three of four tests. It is outperformed by 2LevelGM in the first test with significantly smaller time demand

Synthetic data III

Performance of the 2LevelGM with attributed anchor graphs
geometrical structure of the subgraphs
the size of histograms: 35 bins
radius R of the circle region around each node: 2

1. direct improved performance time of the 2LevelGM algorithm in all tests.
2. a more stable performance in the second test.
3. be more susceptible to graph deformations: selected anchor attributes are not sufficiently robust against deformations in the length of edges.

Synthetic data IV

Comparison of 2LevelGM, GLAG and PATH on bigger graphs

1. GLAG, PATH another optimization problem => can be directly applied to bigger graphs without the necessity to reduce the set of possible candidate matches; initialized with the weighted adjacency matrices
2. do not provide averaged results for this group of tests due to their high computational demand
3. all tests for bigger graphs can be considered as instances of the graph isomorphism problem in exact (1 line) and inexact (2,3 line).
4. 2LevelGM outperforms both GLAG and PATH in objective score and especially in running time with one exception
5. Overall 2LevelGM shows a high matching accuracy in all three tests.

Synthetic data V

6. Additionally we believe it is possible to improve the framework by solving instability issues, which we saw in previous cases.

Image affine transformation I

152 MSER keypoints

1. 2LevelGM is able to find the absolutely correct matching for the whole set except one image pair and therefore shows better results in objective score and accuracy than ProgGM, PATH and GLAG.
2. The result of matching of the last image pair is roughly the same for 2LevelGM and ProgGM. We notice that 2LevelGM is not able to improve graph partitioning further in this case,
3. ProgGM is the fastest algorithm in this comparison. 2LevelGM is a little bit slower than ProgGM. However ProgGM directly stops as soon as the matching score does not increase any more. In contrast to that, our 2LevelGM has to make some additional iterations at the end to ensure that a local minimum is found. This is done intentionally, as we cannot make certain statements about the convergence of our framework.

House data set

1. 111 images of a toy house taken from different viewpoints
2. around 250 MSER keypoints on each image
3. provided ground truth is extrapolated (extrapolation radius 10)
4. 2LevelGM: grid initialization with 2×2 cells $+$ codebook of features
5. ProgGM shows better matching accuracy
6. simple feature matching and GLAG have the lowest accuracy
7. PATH shows the third best result. more robust against deformations
8. running time of 2LevelGM and ProgGM lie in the same range
9. 2LevelGM is comparable with ProgGM for small values of the sequence gap
10. Reasons: RRWM and ineffectiveness of our update rule to improve graph partitions

The end

Thank you for your attention!