EE 583
PATTERN RECOGNITION

SyntPR : Graphical Approaches

- Graph-based Structural Representations
- Graph Isomorphism
- Relational Graphs
- Examples

Introduction

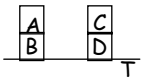
- Higher dimensional grammars are needed to be able to represent complex structures
- Parsing is a good tool for comparing string grammars
- For higher dimensional tree/graph grammars, graph similarity should be defined as a measure of similarity
- Among different trees/graphs (classes), the best match for a given input should be found based on such a measure

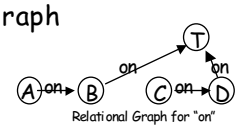
Graph Theory : Definitions (1/2)

- A *graph, G* is represented using a set of nodes (vertices), N and edge set (arcs), R as G={N,R} where R is a subset of NxN
  - In SyntPR applications nodes represent the pattern primitives whereas t edges give the structural information
- A *subgraph* of G is itself a graph G_s={N_s,R_s} where N_s & R_s are subsets of N & R, respectively
- A graph is *connected* if there is a path between all pairs of its nodes
- A graph is *complete* if there is an edge between all pairs of its nodes
- A *directed graph* (digraph) is defined similar to a graph except the pair *(a,b),* which is an element of R, is defined as an edge from *a* to *b*

Graph Theory : Definitions (2/2)

- A *relation* from set A to set B is a subset of AxB
  - e.g. Relation "lies on" : R={(rug,floor),(chair,rug),(person,chair)}
  - Note that relations has a direction: (floor,rug) not element of R
  - Usual notation using functions f: A → B, b=f(a) (function==relation)
  - Relations can be higher dimensional : for (A(xB)xC)xD) → (a,b,c,d)
- A *relational graph* represents one particular relation graphically by using an arrow to show this relation between the elements using a directed graph





- A *semantic net* is a relational graph shows all the relations between its nodes using some labels
- A *tree* is a finite a finiteacyclic (containing no closed loops or paths or cycles) digraph

Comparing Relational Graph Descriptions

- The observed data usually does not match *exactly* to a stored relational representation, hence *similarity* should be measured
- One approach is to check whether the observed data to match a portion of the relational model
  - Case 1 : Any relation not present in both graphs → failure
  - Case 2 : Any single match of a relation → success
  - A realistic strategy is somewhere in between these extremes
- For comparing relations, *adjacency matrix* can be used :

  A digraph G with p nodes can be converted into a matrix by
  - Numbering each node by an index [1,…,p]
  - Representing the existence (absence) of an edge between any nodes in G via Adj(i,j)=1 (Adj(i,j)=0) if G (does not) contains an edge from node j to node i

Graph Isomorphism

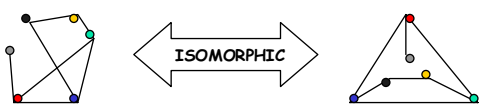
- Consider two graphs, $G_1=\{N_1,R_1\}$ and $G_2=\{N_2,R_2\}$
- *A homomorphism from G_1 to G_2* is a function f from N_1 to N_2

$$(v_1, w_1) \in R_1 \Rightarrow [f(v_1), f(w_1)] \in R_2$$

- An *isomorphism from G_1 to G_2* is a function f from N_1 to N_2 where f is required to be 1:1 and onto

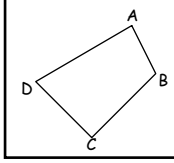
$$(v_1, w_1) \in R_1 \Leftrightarrow [f(v_1), f(w_1)] \in R_2$$

- Isomorphism simply states that relabeling of nodes yields the same graph structure
- Unfortunately, determining graph isomorphism can be computationally expensive

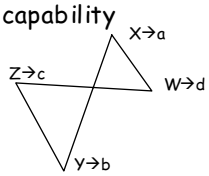


Determining Isomorphism (1/2)

- Given two graphs, $G_1=\{N_1,R_1\}$ and $G_2=\{N_2,R_2\}$ each with p nodes, an easy method to check isomorphism :
  1) Label the nodes of each graph with labels 1,2,…,p
  2) Form the adjacency graph matrices, M_1 and M_2 , for two graphs
  3) If $M_1=M_2$, then G_1 and G_2 are isomorphic
  4) If M_1 is not equal to M_2 , consider all p! possible labelings on G_2
- Note that
  - Complexity of finding isomorphism is quite high
  - In practice, all the existing relations may not be observed due to effects of noise or structural deformations, hence isomorphism is quite rigorous for a similarity measure without any error correction capability



	A	B	C	D
A	0	1	0	1
B	1	0	1	0
C	0	1	0	1
D	1	0	1	0



	X	W	Y	Z
X	0	1	1	0
W	1	0	0	1
Y	1	0	0	1
Z	0	1	1	0

	a	b	c	d
a	0	1	0	1
b	1	0	1	0
c	0	1	0	1
d	1	0	1	0

Determining Isomorphism (2/2)

- There are some invariant properties which are preserved under graph isomorphism :
  - number of nodes,
  - number of arcs,
  - in-degree of a vertex,
  - out-degree of a vertex,
  - closed path of length l
- An alternative method is to find G_1 and G_2 unisomorphic by finding at least one property (e.g. equal number of vertices) that all isomorphic graphs must share but G_1 and G_2 do not.
- G_1 and G_2 are called *subisomorphic* if a subgraph of G_1 is isomorphic to a subgraph of G_2

Extensions to Graph Matching Approach

- Since simple matching of two graphs is not practical, some extensions are proposed to define a measure between graph similarity
  - Extract some-not pattern but graph-features from G_1 and G_2 to form feature vectors x_1 and x_2, respectively. Then use Stat PR techniques to compare x_1 and x_2.
  - Use a matching metric as the minimum number of transformations required to transform G_1 into G_2 , such as
    - node insertion
    - node deletion
    - node splitting
    - node merging
    - edge insertion
    - edge deletion
- Note that
  - Computational complexity is still high
  - Difficulty in designing distance measure which will label all due structural deformations as similar and label those that represent different classes as dissimilar

Relational Graph Similarity (1/2)

- Given a set of nodes N, corresponding relational descriptions is defined as a set of relations $D_N=\{R_1,R_2,\ldots,R_n\}$ where each relation $R_i \subseteq N\times N$ (in general $R_i \subseteq N\times N\ldots\times N$)
- Given two node sets A and B with $|A|=|B|$, corresponding relational descriptions $D_A=\{R_1,R_2,\ldots,R_n\}$ and $D_B=\{S_1,S_2,\ldots,S_n\}$, we may seek for a measure of similarity
- Based upon the i^{th} relation R_i in D_A and i^{th} relation S_i in D_B, *structural error*, E^i, which is a function of f, is defined as

$$E^i(f) = |R_i \circ f - S_i| + |S_i \circ f^{-1} - R_i| \quad where$$

$$R \circ f = \{(b_1,b_2,\ldots,b_k) \in B^k \mid \exists (a_1,a_2,\ldots,a_k) \in A^k\} \quad with\ f(a_i)=b_i \quad i=1,2,\ldots,k$$

- $E^i(f)$ simply measures the number of elements in R_i that are not in S_i PLUS the number of elements in S_i that are not in R_i
- For a <u>given f</u>, two node sets A and B is compared wrt i^{th} relation using $E^i(f)$

Relational Graph Similarity (2/2)

$$E^i(f) = |R_i \circ f - S_i| + |S_i \circ f^{-1} - R_i| \quad where$$

$$R \circ f = \{(b_1, b_2, \ldots, b_k) \in B^k \mid \exists (a_1, a_2, \ldots, a_k) \in A^k\} \quad with\ f(a_i) = b_i \quad i = 1,2,\ldots,k$$

- *Total structural error*, E(f), simply the sum of all structural error for each relations for a given *f*

$$E(f) = \sum_{i=1}^{n} E^i(f)$$

- *Relational distance,* RD, between D_A and D_B is defined as

$$RD(D_A, D_B) = \min_f E(f)$$

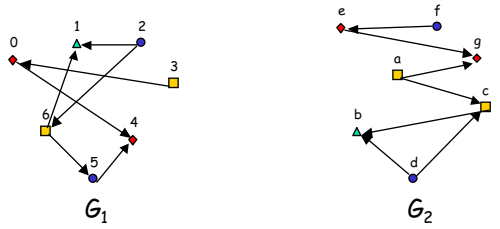
- Note that if any *f* may be found, such that D_A is isomorphic to D_B , then RD is equal to 0
- If they are not isomorphic then RD will give a value according to the "difference" between two graphs

Attributed Graphs (1/2)

- The representation of pattern structure can be improved by including numerical/symbolic attributes of pattern primitives as a graph
- An *attributed graph*, G_i, is a 3-tuple : $G_i = \{N_i, P_i, R_i\}$ where
  - N_i: a set of nodes
  - P_i : a set of properties of these nodes
  - R_i : a set of relations between nodes
- Let $p^i_q(n)$ be the value of q^{th} property of node n of graph G_i
- Nodes n_1 and n_2 (on G_1 and G_2) are said to form an *assignment* if $p^1_q(n_1) \sim p^2_q(n_2)$ (~ denotes similarity)
- Let $r^i_j(n_x, n_y)$ be the j^{th} relation between nodes n_x and n_y of graph G_i
- While comparing G_1 and G_2, two assignments (n_1, n_2) and (n'_1, n'_2) are considered *compatible* if $r^1_j(n_1, n'_1) \sim r^2_j(n_2, n'_2)$ for all j
- Two attributed graphs, G_1 and G_2, are isomorphic if there exists a set of 1:1 assignments of nodes in G_1 to nodes in G_2, such that all assignments are compatible

Attributed Graphs (2/2)

- Example: For the given 2 attributed graphs, G_1 and G_2,
  - Node properties denoted by different shapes and colors
  - The single relation is shown by the arcs connecting the nodes



Assignments = { (0~e), (0~g), (1~b), (2~f), (2~d), (3~a), (3~c), (4~e), (4~g), (5~d), (5~f), (6~a), (6~c) }
Compatibility= { [(6,1)~(c,b)], [(2,6)~(d,c)], [(5,4)~(f,e)], [(0,4)~(e,g)], [(3,0)~(a,g)], [(2,1)~(d,b)]
[(0,1)~(e,b)], [(0,1)~(g,b)], ...(all the compatible unconnected assignments).....}

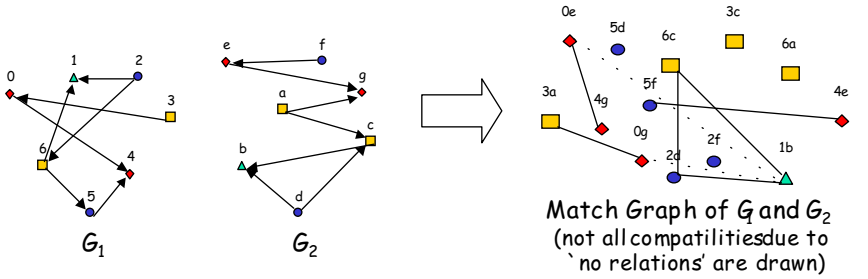
Two graphs are not isomorphic since there is no set of 1 -1 assignments between the nodes of G_1 and G_2, such that all assignments are compatible

Comparing Attributed Graphs (1/3)

- A less rigorous test of similarity is also required for attributed graphs, as in the case of relational graphs
  - Extract some-not pattern but graph features from G_1 and G_2 to form feature vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Then, use StatPR techniques to compare $\mathbf{x}_1$ and $\mathbf{x}_2$.
  - Use a matching metric as the minimum number of transformations required to transform G_1 into G_2
- A better way to check similarity of two attributed graphs is obtained by beginning from (not all the nodes but) the nodes belonging to the maximal cliques of the match graph
  - A *clique* of a graph G is a totally connected subgraph
  - A *maximal clique* is not included in any other clique
  - A *match graph* (MG) is formed from two graphs G_1 and G_2 by
    - Nodes of MG are *assignments* from G_1 and G_2 , respectively
    - An edge in the MG exists between two nodes if the corresponding *assignments* are *compatible*

Comparing Attributed Graphs (2/3)

- Example: For the previously given 2 attributed graphs G_1 and G_2,



Assignments = { (0~e), (0~g), (1~b), (2~f), (2~d), (3~a), (3~c), (4~e), (4~g), (5~d), (5~f), (6~a), (6~c) }
Compatibility= { [(6,1)~(c,b)], [(2,6)~(d,c)], [(5,4)~(f,e)], [(0,4)~(e,g)], [(3,0)~(a,g)], [(2,1)~(d,b)]
[(0,1)~(e,b)], [(0,1)~(g,b)], ...(all the compatible unconnected assignments).....} }

Comparing Attributed Graphs (3/3)

- Measuring the transformation difference between graphs
- In order to transform graph G_i to graph G_j, a similarity measure $D(G_i, G_j)$ must be defined with these properties :
  - $D(G_i, G_i) = 0$
  - $D(G_i, G_j) > 0$ for i not equal to j
  - $D(G_i, G_j) = D(G_j, G_i)$ (equal to costs for insertion/deletion)
  - $D(G_i, G_j) < D(G_i, G_k) + D(G_k, G_j)$ (triangle inequality)
- Distance measure can be defined as

$$D = \min_x \{D_s(x)\} \quad where\ D_s(x) = w_{ni}c_{ni} + w_{nd}c_{nd} + w_{ei}c_{ei} + w_{ed}c_{ed} + w_n c_n(x)$$

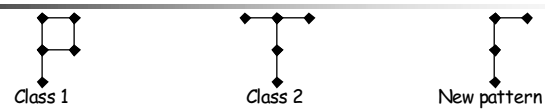
w : weight for corresponding transformation ni : node insert, nd : node delete

c : cost for corresponding transformation ei : edge insert, ed : edge delete

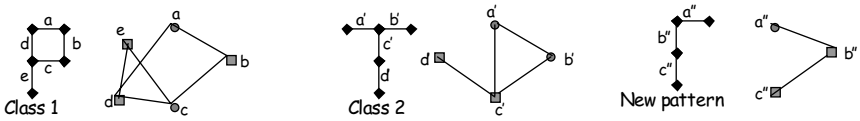
$$c_n(x) = \sum f_n(p_i, q_j) \quad where\ f_n(p_i, q_j)\ \text{similarity measure between}\ p_i\ \text{of}\ G_i\ \text{and}\ q_j\ \text{of}\ G_j$$

x : denotes a node mapping (configuration) between two graphs

Example : Classification by Attributed Graphs

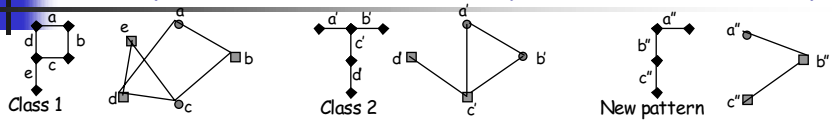


- For each pattern, a label is placed on each line segment, thus specifying an attribute indicating segment orientation
- Attributed segments (horizontal & vertical) form graph node
- Symmetric/undirected relation of "attached" is used

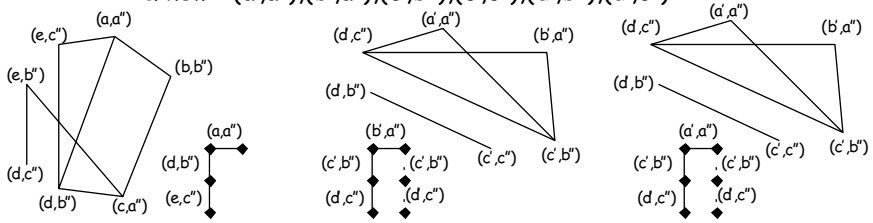


- Some of the previously mentioned matching methods can be used to compare the graphs above
  - Determinesubisomorphism
  - Use features extracted from these graphs, such as # of nodes

Example : Classification by Attributed Graphs



- In order to compare these graphs via maximum cliques, we have to develop matching graphs
- "P" & new : (a,a"),(b,b"),(b,c"),(c,a"),(d,b"),(d,c"), (e,b"),(e,c")
- "T" & new : (a',a"),(b',a"),(c',b"),(c',c"),(d',b"),(d',c")



- The maximum cliques of both Match Graphs have a cardinality = 3

➔ Choose one of the two arbitrarily