# Graph Matching - Algorithms

Wolf-Dieter Vogl, 0626355

Pattern Recognition and Image Processing Group (PRIP)
Institute of Computer Graphics and Algorithms
University Of Technology, Vienna

May 11, 2011

# Overview

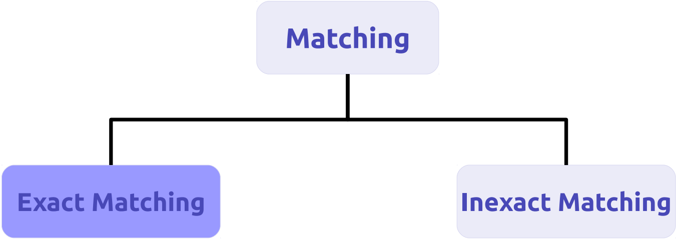
- **Matching**
  - **Exact Matching**
  - **Inexact Matching**

**Goals**

Exact Matching: Find an *edge preserving* mapping.

Inexact Matching: Find a mapping that minimizes matching costs.

# Exact Matching




## Exact Matching Complexity

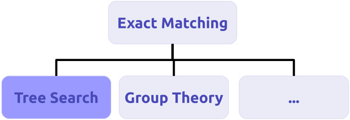
Graph Isomorphism : In NP, but unknown if P or NP complete.

Subgraph Isomorphism, Monomorphism, MCS... : NP complete.

There exists algorithms for special graphs with polynomial runtime.

# Exact Matching

## Tree-Search approach




### Basic Idea

- Iteratively expand partial match by adding new pairs of matched nodes.
- The pair is chosen using some necessary conditions.
- Prune unfruitful search paths.
- If no further vertex pairs may be added due to constraint, undo last additions (backtracking)
- Algorithm stops if match has been found or all matchings that satisfy the constraints has been tried.

# Exact Matching

## Ullmann's Algorithm [J.R. Ullmann 1976]

- Tree-Search algorithm (Depth-Search-First)
- Uses adjacency matrices and additional constraints for matching and pruning.
- Application for graph isomorphism, subgraph isomorphism and monomorphism, also for MCS problem

# Exact Matching

## Ullmann's Algorithm

- Given: Two graphs $G_A(V_A, E_A)$ and $G_B(V_B, E_B)$ and their adjacency matrices: $A$ and $B$
- Idea: $n = |V_a|$, $m = |V_b|$, $n \times m$ permutation matrix $M$ with following form:
  - M contains only '0' and '1'
  - Exact one '1' in each row
  - Not more than one '1' in each column
- Permutate adjacency matrix $B$ by multiplying it with $M$, and compare adjacency.

# Exact Matching

## Ullmann's Algorithm

- $M \times B$: Move row $j$ to row $i$ $\forall M_{ij} = 1$

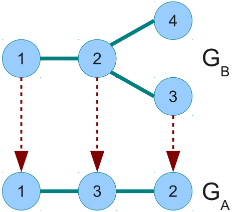
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

$M = M^T$ $\quad$ $B = B^T$


- $(MB)^T$: Move column $j$ to column $i$
- $M(MB)^T$: Move column $j$ to column $i$ and row $j$ to row $i$

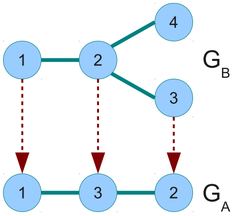
# Exact Matching

## Ullmann's Algorithm



# Exact Matching

## Ullmann's Algorithm




| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |

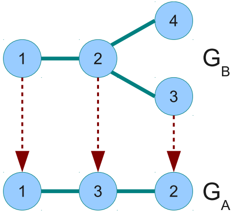
$B=B^T$

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |

M

# Exact Matching

## Ullmann's Algorithm




| | | | |
|---|---|---|---|
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |

$B=B^T$

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |

M

$$M(MB)^T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \left( \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \right)^T$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} = C$$

# Exact Matching

## Ullmann's Algorithm

Creating pairs of nodes by exchanging rows and columns (renaming).

### Adjacency condition

Let $C = M(MB)^T$,
A is a (subgraph-) isomorphism iff

$$A_{ij} = 1 \Rightarrow C_{ij} = 1 \forall i, j$$

How do we get M?

# Exact Matching

## Ullmann's Algorithm

- Build Startmatrix $M^0$ by setting all values to 1 (allow all permutations)
- Set values to 0 for all $M^0_{ij}$ where $deg(B_j) < deg(A_i)$ (remove impossible permutations)

$$M^0_{ij} = \begin{cases} 1 & \textit{if} \quad deg(B_j) \geq deg(A_i) \\ 0 & \quad \text{otherwise} \end{cases} \quad , \forall i, j$$

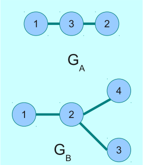
- Generate systematically permutation matrices $M^d$.


$$\underset{A}{\begin{array}{|c|c|c|} \hline 0 & 0 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 1 & 0 \\ \hline \end{array}} \rightarrow \underset{M^0}{\begin{array}{|c|c|c|c|} \hline 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 \\ \hline \end{array}} \leftarrow \begin{array}{|c|c|c|c|} \hline 0 & 1 & 0 & 0 \\ \hline 1 & 0 & 1 & 1 \\ \hline 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 \\ \hline \end{array}$$

# Exact Matching

## Ullmann's Algorithm

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

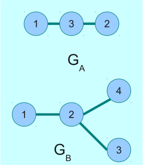


# Exact Matching

## Ullmann's Algorithm

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |



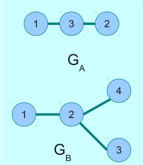
# Exact Matching

## Ullmann's Algorithm

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |



# Exact Matching

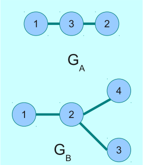
## Ullmann's Algorithm

| | | | |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |

| | | | |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |



# Exact Matching

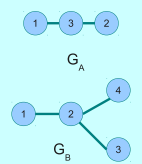
## Ullmann's Algorithm

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

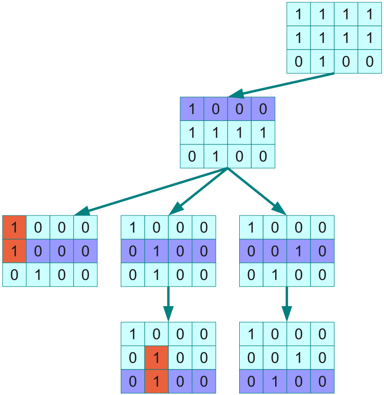
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

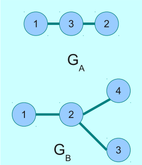
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$



# Exact Matching

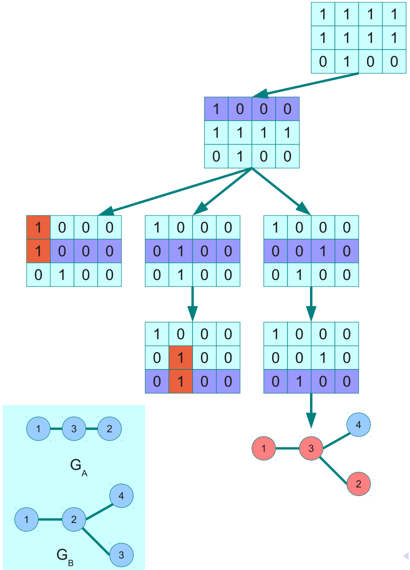
## Ullmann's Algorithm



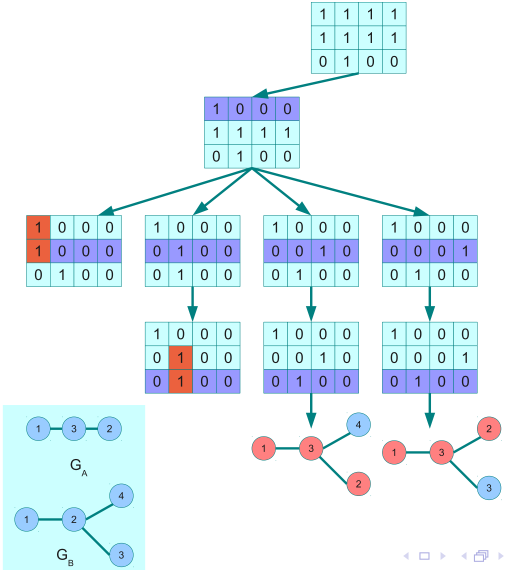
# Exact Matching

## Ullmann's Algorithm



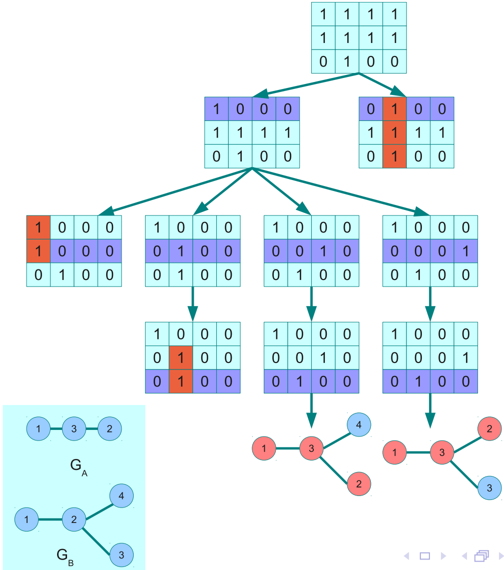
# Exact Matching

## Ullmann's Algorithm



# Exact Matching

## Ullmann's Algorithm



# Exact Matching

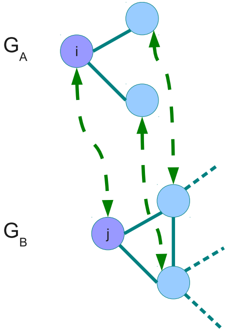
## Ullmann's Algorithm V2

*Refinement Procedure:*

- For all neighbours in A there must be proper neighbours in B.
- Formally:

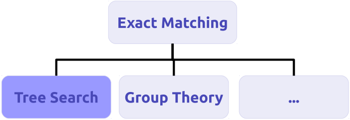
$$\forall k(A_{ik} = 1 \Rightarrow \exists p(M_{kp}B_{pj} = 1))$$

- Set $M^d_{ij} = 0$ where conditions are not complied.



# Exact Matching

## Tree-Search approaches

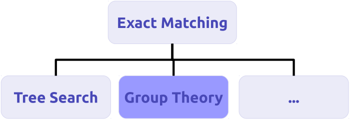



### VF and VF2 algorithm

- Application for isomorphism and subgraph isomorphism
- VF algorithm defines a heuristic based on the analysis of the sets of nodes adjacent to the ones already considered in the partial mapping.

# Exact Matching

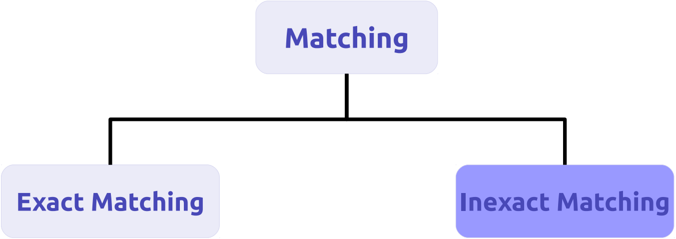
## Group theory approach




### McKay's Nauty

- Nauty - No automorphisms, yes?
- Application for isomorphism only
- It uses the property that the canonical labeling for isomorph graphs is identical.
- It constructs the automorphism group of each of the input graphs and derives a canonical labeling

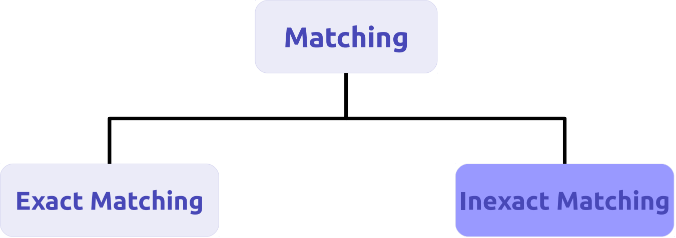
# Inexact Matching




## Reasons for using inexact matching

- Deformations in graphs (eg. noise, variability of patterns, nondeterministic elements...)
- Exact matching too expensive

# Inexact Matching

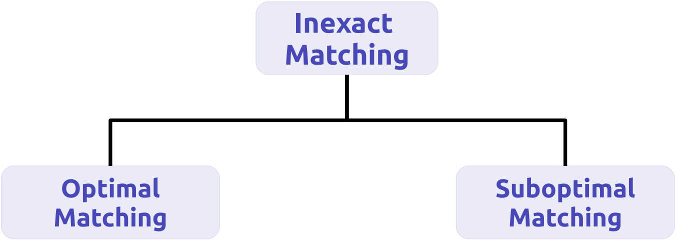



## How

- Relax constraints - no edge preserving.
- Penalize graph differences.
- Find a mapping that minimizes matching costs.

# Inexact Matching

## Optimal and suboptimal inexact matching algorithms



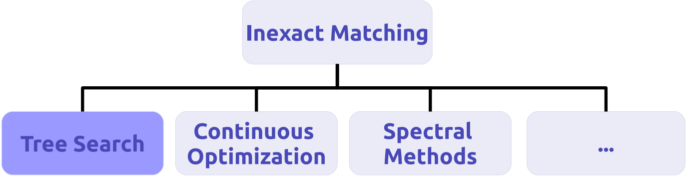

**Runtime**

Optimal Inexact Matching : More expensive than exact algorithms.

Suboptimal Inexact Matching : Usually polynomial matching time.

# Inexact Matching

## Tree-Search approaches

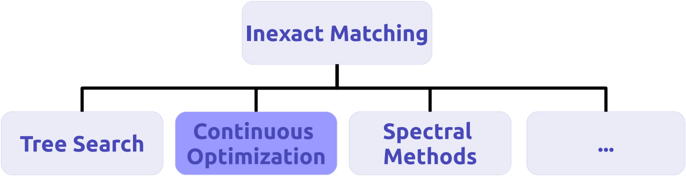



### Tree Search with Backtracking

- Search guided by cost function.
- Heuristic estimate matching cost for remaining nodes.
- Prune unfruitful paths.

# Inexact Matching

## Continuous optimization approaches

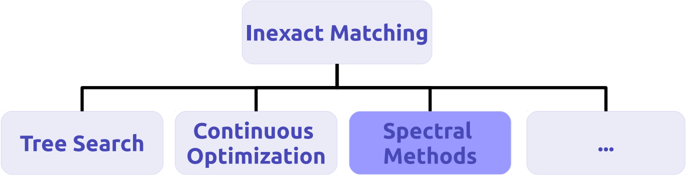



### Continuous Optimization

- Idea:
  - Convert discrete optimization problem to a continuous, nonlinear optimization problem.
  - Use a nonlinear optimization algorithm.
  - Convert back to graph matching domain.
- Suboptimal Matching
- Polynomial Runtime

# Inexact Matching

## Spectral methods

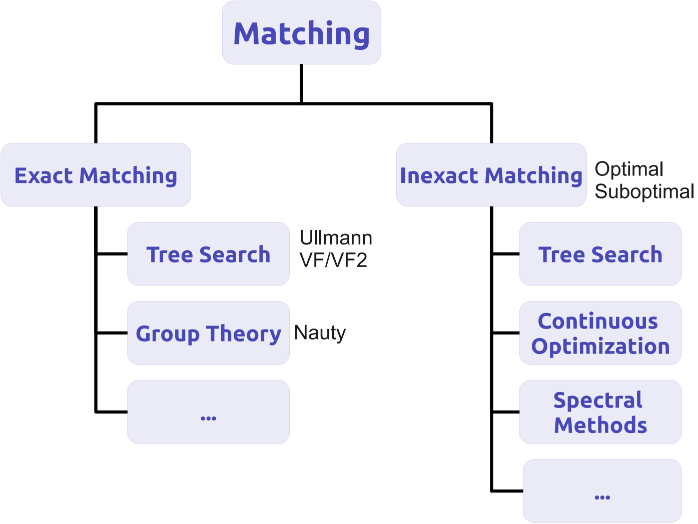



### Spectral Methods

Uses property, that

- Eigenvalues and eigenvectors of adjacency matrix of a graph are invariant to node permutations.
- $G_1, G_2 isomorph \Rightarrow$
  $EW(Adj(G_1)) = EW(Adj(G_2)) \wedge EV(Adj(G_1)) = EV(Adj(G_2))$

# Recall


Matching
Exact Matching
Tree Search
Ullmann
VF/VF2
Group Theory
Nauty
...
Inexact Matching
Optimal
Suboptimal
Tree Search
Continuous Optimization
Spectral Methods
...

# Literature


- D.Conte, *Thirty Years Of Graph Matching in Pattern Recognition*, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 18, No. 3, p.265-298, (2004)
- Riesen K. Jiang X., Bunke H., *Exact and Inexact Graph Matching: Methodology And Applications*, Managing and Mining Graph Data, p.217-247, Springer Verlag (2010)

# Thank you for your attention!

## Questions?