Graph similarity and matching

Many applications involve measuring objects similarity:

- Searching and classifying documents
- Case based Reasoning
- Pattern recognition
- ...

⇒ a soft, quantitative, qualitative and customized measure
Graph similarity and matching

Many applications involve measuring objects similarity:
- Searching and classifying documents
- Case based Reasoning
- Pattern recognition

Match components that may not be identical
- find the best matching!
- allow “multivalent” matchings
- walls ‘e’ and ‘f’ correspond to wall ‘5’

Evaluate the quantity of common features
- ratio w.r.t. the total number of features
- identify the common features

...to explain differences and commonalities

Adapt criteria w.r.t. the considered application
...and the goal of the user

⇒ a soft, quantitative, qualitative and customized measure
Graph similarity and matching

Many applications involve measuring objects similarity:
- Searching and classifying documents
- Case based Reasoning
- Pattern recognition
...

⇒ a soft, quantitative, qualitative and customized measure

- Match components that may not be identical
  ⇝ find the best matching!

- Allow “multivalent” matchings
  ⇝ walls ’e’ and ’f’ correspond to wall ’5’
Graph similarity and matching

Many applications involve measuring objects similarity:
- Searching and classifying documents
- Case based Reasoning
- Pattern recognition

⇒ a soft, quantitative, qualitative and customized measure

- Evaluate the quantity of common features
  ⇔ ratio w.r.t. the total number of features
- Identify the common features
  ...to explain differences and commonalities
Graph similarity and matching

Many applications involve measuring objects similarity:
- Searching and classifying documents
- Case based Reasoning
- Pattern recognition

...to explain differences and commonalities

Match components that may not be identical
→ find the best matching!

Allow "multivalent" matchings
→ walls 'e' and 'f' correspond to wall '5'

Evaluate the quantity of common features
→ ratio w.r.t. the total number of features

Identify the common features

Adapt criteria w.r.t. the considered application
...and the goal of the user

⇒ a soft, quantitative, qualitative and customized measure
## Similarity and graph matchings

### Describing objects by labelled graphs

- Object components $\rightsquigarrow$ Graph vertices
- Relationships between components $\rightsquigarrow$ Graph edges
- Component and relation features $\rightsquigarrow$ Vertex and edge labels

Measuring object similarity $\rightsquigarrow$ Matching graph vertices

### Existing graph matchings and similarity measures

- Graph isomorphism $\rightsquigarrow$ Equivalence
- Subgraph isomorphism $\rightsquigarrow$ Inclusion
- Largest common subgraph $\rightsquigarrow$ Intersection
- Graph edit distance $\rightsquigarrow$ Cost of transformation

### Limitations of these existing matchings

- Univalent matchings
- Hardly customizable
Similarity and graph matchings

Describing objects by labelled graphs

- Object components $\sim$ Graph vertices
- Relationships between components $\sim$ Graph edges
- Component and relation features $\sim$ Vertex and edge labels

Measuring object similarity $\sim$ Matching graph vertices

Existing graph matchings and similarity measures

- Graph isomorphism $\sim$ Equivalence
- Subgraph isomorphism $\sim$ Inclusion
- Largest common subgraph $\sim$ Intersection
- Graph edit distance $\sim$ Cost of transformation

Limitations of these existing matchings

- Univalent matchings
- Hardly customizable
## Similarity and graph matchings

### Describing objects by labelled graphs
- Object components $\leadsto$ Graph vertices
- Relationships between components $\leadsto$ Graph edges
- Component and relation features $\leadsto$ Vertex and edge labels

Measuring object similarity $\leadsto$ Matching graph vertices

### Existing graph matchings and similarity measures
- Graph isomorphism $\leadsto$ Equivalence
- Subgraph isomorphism $\leadsto$ Inclusion
- Largest common subgraph $\leadsto$ Intersection
- Graph edit distance $\leadsto$ Cost of transformation

### Limitations of these existing matchings
- Univalent matchings
- Hardly customizable
Overview of the talk

Description of a similarity measure
- Based on multivalent matchings
- Customizable

Description of two algorithms for computing this measure
- Reactive Tabu Search
- Ant Colony Optimization

Experimental comparison on two different test suites
- Randomly generated multivalent graph matching problem
- Multivalent graph matching problem of [Boeres et al. 2004]
Describing objects with labelled graphs

Let $L_V$ and $L_E$ be sets of vertex and edge labels.
A labelled graph is defined by a triple $\langle V, r_V, r_E \rangle$ s.t.

- $V \rightsquigarrow$ graph vertices
- $r_V \subseteq V \times L_V \rightsquigarrow$ vertex labelling
- $r_E \subseteq V \times V \times L_E \rightsquigarrow$ edge labelling
Let $L_V$ and $L_E$ be sets of vertex and edge labels. A labelled graph is defined by a triple $\langle V, r_V, r_E \rangle$ s.t.

- $V \rightsquigarrow$ graph vertices
- $r_V \subseteq V \times L_V \rightsquigarrow$ vertex labelling
- $r_E \subseteq V \times V \times L_E \rightsquigarrow$ edge labelling

Vertices: $V = \{a, b, c, d, e, f\}$
Describing objects with labelled graphs

Let $L_V$ and $L_E$ be sets of vertex and edge labels. A labelled graph is defined by a triple $\langle V, r_V, r_E \rangle$ s.t.

- $V \rightsquigarrow$ graph vertices
- $r_V \subseteq V \times L_V \rightsquigarrow$ vertex labelling
- $r_E \subseteq V \times V \times L_E \rightsquigarrow$ edge labelling

Vertex labels: $L_V = \{beam, I, wall\}$

$r_V = \{(a, beam), (b, beam), (c, beam), (d, beam), (a, I), (b, I), (c, I), (d, I), (e, wall), (f, wall)\}$
Let $L_V$ and $L_E$ be sets of vertex and edge labels.
A labelled graph is defined by a triple $\langle V, r_V, r_E \rangle$ s.t.

- $V \leadsto$ graph vertices
- $r_V \subseteq V \times L_V \leadsto$ vertex labelling
- $r_E \subseteq V \times V \times L_E \leadsto$ edge labelling

Edge labels: $L_E = \{next, on\}$

$r_E = \{(a, b, next), (b, c, next), (c, d, next), (a, e, on), (b, e, on), (c, f, on), (d, f, on)\}$
Let $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ and $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$ be 2 graphs such that $V_1 \cap V_2 = \emptyset$

- A matching of $G_1$ and $G_2$ = a relation $m \subseteq V_1 \times V_2$
Let $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ and $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$ be 2 graphs such that $V_1 \cap V_2 = \emptyset$

- A matching of $G_1$ and $G_2$ = a relation $m \subseteq V_1 \times V_2$

$$m = \{ (a, 1), (b, 2), (c, 3), (d, 4), (e, 5), (f, 5) \}$$
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} \mid f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} / f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]

\[ m = \{(a,1), (b,beam), (1,beam), \}

\[ G_1 \cap_m G_2 = \{(a,beam), (1,beam), \} \]
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} / f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} \mid f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]

\[ m = \{(a,1), (b,2), (c,3), \]
\[ G_1 \cap_m G_2 = \{ (a,\text{beam}), (1,\text{beam}), (b,\text{beam}), (2,\text{beam}), (a,b,\text{next}), (1,2,\text{next}), (c,\text{beam}), (3,\text{beam}), (b,c,\text{next}), (2,3,\text{next}), \]
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} / f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]

\[ m = \{(a,1), (b,2), (c,3), (d,4), \]
\[ G_1 \cap_m G_2 = \{(a,beam), (1,beam), (b,beam), (2,beam), (a,b,next), (1,2,next), (c,beam), (3,beam), (b,c,next), (2,3,next), (d,beam), (4,beam), (c,d,next), (3,4,next), \]
Common features w.r.t. a matching

\[ G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} / f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \} \]

\[ m = \{(a,1), (b,2), (c,3), (d,4), (e,5), \]
\[ G_1 \cap_m G_2 = \{(a,\text{beam}), (1,\text{beam}), (b,\text{beam}), (2,\text{beam}), (a,b,\text{next}), (1,2,\text{next}), (c,\text{beam}), (3,\text{beam}), (b,c,\text{next}), (2,3,\text{next}), (d,\text{beam}), (4,\text{beam}), (c,d,\text{next}), (3,4,\text{next}), (e,\text{wall}), (5,\text{wall}), (a,e,\text{on}), (b,e,\text{on}), (1,5,\text{on}), (2,5,\text{on})\} \]
Common features w.r.t. a matching

\[
G_1 \cap_m G_2 = \{ f \in r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} / f \text{ common to } G_1 \text{ and } G_2 \text{ via } m \}
\]

\[
m = \{(a,1), (b,2), (c,3), (d,4), (e,5), (f,5)\}
\]

\[
G_1 \cap_m G_2 = \{(a,\text{beam}), (1,\text{beam}), (b,\text{beam}), (2,\text{beam}), (a,b,\text{next}), (1,2,\text{next}), (c,\text{beam}), (3,\text{beam}), (b,c,\text{next}), (2,3,\text{next}), (d,\text{beam}), (4,\text{beam}), (c,d,\text{next}), (3,4,\text{next}), (e,\text{wall}), (5,\text{wall}), (a,e,\text{on}), (b,e,\text{on}), (1,5,\text{on}), (2,5,\text{on}), (f,\text{wall}), (c,f,\text{on}), (d,f,\text{on}), (3,5,\text{on}), (4,5,\text{on})\}
\]
Similarity of two graphs

Similarity of $G_1$ and $G_2$ w.r.t. a matching $m$

$$\text{sim}_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(\text{splits}(m))}{f(r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2})}$$
Similarity of two graphs

**Similarity of** $G_1$ and $G_2$ **w.r.t. a matching** $m$

$$sim_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(splits(m))}{f(r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2})}$$

$G_1 \cap_m G_2 =$ features common to $G_1$ and $G_2$ via $m$
Similarity of two graphs

**Similarity of $G_1$ and $G_2$ w.r.t. a matching $m$**

$$sim_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(splits(m))}{f(r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2})}$$

$G_1 \cap_m G_2 =$ features common to $G_1$ and $G_2$ via $m$

$splits(m) =$ set of vertices that are matched to several vertices
## Similarity of two graphs

**Similarity of $G_1$ and $G_2$ w.r.t. a matching $m$**

\[
sim_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(splits(m))}{f(r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2})}
\]

$G_1 \cap_m G_2 = \text{features common to } G_1 \text{ and } G_2 \text{ via } m$

$splits(m) = \text{set of vertices that are matched to several vertices}$

$r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2} = \text{set of all features of } G_1 \text{ and } G_2$
Similarity of two graphs

**Similarity of $G_1$ and $G_2$ w.r.t. a matching $m$**

$$sim_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(splits(m))}{f(r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2})}$$

- $G_1 \cap_m G_2$ = features common to $G_1$ and $G_2$ via $m$
- $splits(m)$ = set of vertices that are matched to several vertices
- $r_{V_1} \cup r_{E_1} \cup r_{V_2} \cup r_{E_2}$ = set of all features of $G_1$ and $G_2$
- $f$ and $g$ = similarity functions $\leadsto$ customization
Similarity of two graphs

**Similarity of** $G_1$ **and** $G_2$ **w.r.t. a matching** $m$

$$sim_m(G_1, G_2) = \frac{f(G_1 \cap_m G_2) - g(splits(m))}{f(r_{V_1 \cup E_1} \cup r_{V_2 \cup E_2})}$$

$G_1 \cap_m G_2 =$ features common to $G_1$ and $G_2$ via $m$

$splits(m) =$ set of vertices that are matched to several vertices

$r_{V_1 \cup E_1} \cup r_{V_2 \cup E_2} =$ set of all features of $G_1$ and $G_2$

$f$ and $g =$ similarity functions $\rightsquigarrow$ customization

**Similarity of** $G_1$ **and** $G_2$

$$sim(G_1, G_2) = \max_{m \subseteq V_1 \times V_2} sim_m(G_1, G_2)$$

Measuring the similarity of $G_1$ and $G_2$

$\rightsquigarrow$ find $m \subseteq V_1 \times V_2$ that maximizes $f(G_1 \cap_m G_2) - g(splits(m))$
## A generic and customizable measure

### Similarity functions $f$ and $g$

- Allow one to express similarity knowledge
- May be defined by weighted sums

### Solving univalent matching problems...

- Graph isomorphism
- (Partial) Subgraph isomorphism
- Largest (partial) common subgraph
- Graph edit distance

### ...and multivalent graph matching problems

- Extended graph edit distance [Ambauen & al. 2003]
- Non bijective matchings [Boeres & al. 2004]
### A generic and customizable measure

**Similarity functions $f$ and $g$**
- Allow one to express similarity knowledge
- May be defined by weighted sums

**Solving univalent matching problems...**
- Graph isomorphism
- (Partial) Subgraph isomorphism
- Largest (partial) common subgraph
- Graph edit distance

...and multivalent graph matching problems
- Extended graph edit distance [Ambauen & al. 2003]
- Non bijective matchings [Boeres & al. 2004]
A generic and customizable measure

**Similarity functions \( f \) and \( g \)**
- Allow one to express similarity knowledge
- May be defined by weighted sums

**Solving univalent matching problems...**
- Graph isomorphism
- (Partial) Subgraph isomorphism
- Largest (partial) common subgraph
- Graph edit distance

**...and multivalent graph matching problems**
- Extended graph edit distance [Ambauen & al. 2003]
- Non bijective matchings [Boeres & al. 2004]
## Algorithms for measuring graph similarity

### A new combinatorial problem

- Goal = find \( m \subseteq V_1 \times V_2 \) that maximizes
  \[
  \text{score}(m) = f(G_1 \cap_m G_2) - g(\text{splits}(m))
  \]
- \( \mathcal{NP} \)-hard problem \( \leadsto 2^{|V_1| \cdot |V_2|} \) states to explore

### Solving with a complete approach

- Structure the search space with a lattice...
- ...but the score function is not monotonic w.r.t. inclusion
  \( \leadsto \) limited to very small graphs

### Solving with incomplete approaches

- Greedy: incremental construction of a matching
- Tabu: improvement of a matching by exploring its neighborhood
- ACO: using ants to guide greedy constructions
Algorithms for measuring graph similarity

A new combinatorial problem

- Goal = find \( m \subseteq V_1 \times V_2 \) that maximizes
  \[
  \text{score}(m) = f(G_1 \cap_m G_2) - g(\text{splits}(m))
  \]

- \( \mathcal{NP} \)-hard problem \( \leadsto 2^{\mid V_1 \mid \cdot \mid V_2 \mid} \) states to explore

Solving with a complete approach

- Structure the search space with a lattice...
- ...but the score function is not monotonic w.r.t. inclusion

  \( \leadsto \) limited to very small graphs

Solving with incomplete approaches

- Greedy: incremental construction of a matching
- Tabu: improvement of a matching by exploring its neighborhood
- ACO: using ants to guide greedy constructions
A new combinatorial problem

- Goal = find \( m \subseteq V_1 \times V_2 \) that maximizes

\[
\text{score}(m) = f(G_1 \cap_m G_2) - g(\text{splits}(m))
\]

- \( \mathcal{NP} \)-hard problem \( \sim \) \( 2^{|V_1| \cdot |V_2|} \) states to explore

Solving with a complete approach

- Structure the search space with a lattice...
- ...but the score function is not monotonic w.r.t. inclusion

\( \sim \) limited to very small graphs

Solving with incomplete approaches

- Greedy: incremental construction of a matching
- Tabu: improvement of a matching by exploring its neighborhood
- ACO: using ants to guide greedy constructions
Greedy algorithm

**Greedy construction of a matching** \( m \)

- \( m \leftarrow \emptyset \)
- **Iterate**
  - \( \text{Cand} \leftarrow V_1 \times V_2 - m \)
  - Choose \((u_1, u_2) \in \text{Cand}\) that maximizes the \( \text{score} \) function
    \( \leadsto \) break ties randomly
  - **Exit when** \( \text{score}(m \cup \{(u_1, u_2)\}) < \text{score}(m) \)
    - \( m \leftarrow m \cup \{(u_1, u_2)\} \)
- **End iterate**

**Properties**

- Polynomial complexity \( \mathcal{O}(\left| V_1 \right| \cdot \left| V_2 \right|)^2 \)
- Non optimal
- Non deterministic \( \leadsto \) may be iterated
Reactive Tabu Search

**Exploration of the neighborhood of a matching** $m$

- $m \leftarrow \text{Greedy}(G_1, G_2)$
- **While** $\text{score}(m) < Q$ **and** time limit not reached
  - Choose $m' \in \text{Neighborhood}(m)$ so that
    - the move $m \leadsto m'$ is not “Tabu”
    - $m'$ maximizes the score function
  - $m \leftarrow m'$
  - Record the move $m' \leadsto m$ as “Tabu”
- **End while**
### Reactive Tabu Search

**Exploration of the neighborhood of a matching** \( m \)

1. \( m \leftarrow \text{Greedy}(G_1, G_2) \)
2. **While** \( \text{score}(m) < Q \text{ and } \) time limit not reached
   - Choose \( m' \in \text{Neighborhood}(m) \) so that
     - the move \( m \rightsquigarrow m' \) is not “Tabu”
     - \( m' \) maximizes the score function
   - \( m \leftarrow m' \)
   - Record the move \( m' \rightsquigarrow m \) as “Tabu”
3. **End while**

**Greedy:** Local search is started from a matching built by Greedy

⇒ May be iterated from \( \neq \) starting points computed by Greedy
Reactive Tabu Search

**Exploration of the neighborhood of a matching** $m$

- $m \leftarrow Greedy(G_1, G_2)$
- While $score(m) < Q$ and time limit not reached
  - Choose $m' \in Neighborhood(m)$ so that
    - the move $m \sim m'$ is not “Tabu”
    - $m'$ maximizes the score function
  - $m \leftarrow m'$
  - Record the move $m' \sim m$ as “Tabu”
- End while

**Neighborhood(m)** = set of matchings obtained by removing or adding a couple of vertices from $m$
Reactive Tabu Search

Exploration of the neighborhood of a matching $m$

- $m \leftarrow Greedy(G_1, G_2)$
- **While** $\text{score}(m) < Q$ and time limit not reached
  - Choose $m' \in \text{Neighborhood}(m)$ so that
    - the move $m \leadsto m'$ is not "Tabu"
    - $m'$ maximizes the score function
  - $m \leftarrow m'$
  - Record the move $m' \leadsto m$ as "Tabu"
- **End while**

Tabu principle $\leadsto$ prevent the search from cycling

- Memorize the $k$ last moves in a tabu list
- $k$ determines intensification/diversification of the search
  - decreasing $k \leadsto$ intensifying
  - increasing $k \leadsto$ diversifying
- Reactive search $\leadsto$ dynamically adapt $k$
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{max}$
- **repeat**
  - 1. Each ant builds a matching
  - 2. Perform greedy local search on the best matching
  - 3. Update pheromone trails
- **until** optimal matching found or stagnation
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{max}$
- repeat
  1. Each ant builds a matching
  2. Perform greedy local search on the best matching
  3. Update pheromone trails
- until optimal matching found or stagnation

Pheromone trails

Let $G_1 = \langle V_1, r_{V_1}, r_{E_1} \rangle$ and $G_2 = \langle V_2, r_{V_2}, r_{E_2} \rangle$ be 2 graphs
Associate a pheromone trail with each couple $(u, v) \in V_1 \times V_2$
$\tau(u, v) = $ learnt desirability of matching $u$ with $v$
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{\text{max}}$
- \textbf{repeat}
  - \textbf{1} Each ant builds a matching
  - \textbf{2} Perform greedy local search on the best matching
  - \textbf{3} Update pheromone trails
- \textbf{until} optimal matching found or stagnation

Greedy randomized construction of a matching by an ant

- Let $m =$ current matching and $\text{cand} = V_1 \times V_2 - m$
- Choose $(u, v) \in \text{cand}$ w.r.t. the probability

\[
p(u, v) = \frac{\left[ \tau(u, v) \right]^\alpha \cdot \left[ \eta_m(u, v) \right]^\beta}{\sum_{(u', v') \in \text{cand}} \left[ \tau(u', v') \right]^\alpha \cdot \left[ \eta_m(u', v') \right]^\beta}
\]
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{max}$
- repeat
  1. **Each ant builds a matching**
  2. Perform greedy local search on the best matching
  3. Update pheromone trails
- until optimal matching found or stagnation

---

Greedy randomized construction of a matching by an ant

- Let $m =$ current matching and $\text{cand} = V_1 \times V_2 - m$
- Choose $(u, v) \in \text{cand}$ w.r.t. the probability

$$
p(u, v) = \frac{[\tau(u, v)]^\alpha \cdot [\eta_m(u, v)]^\beta}{\sum_{(u', v') \in \text{cand}} [\tau(u', v')]^\alpha \cdot [\eta_m(u', v')]^\beta}
$$

- $\tau(u, v) \sim$ past experience of the colony w.r.t. choosing $(u, v)$
### ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{\text{max}}$

**repeat**

1. **Each ant builds a matching**
2. Perform greedy local search on the best matching
3. Update pheromone trails

**until** optimal matching found or stagnation

---

### Greedy randomized construction of a matching by an ant

- Let $m =$ current matching and $\text{cand} = V_1 \times V_2 - m$
- Choose $(u, v) \in \text{cand}$ w.r.t. the probability

\[
p(u, v) = \frac{[\tau(u, v)]^{\alpha} \cdot [\eta_m(u, v)]^{\beta}}{\sum_{(u', v') \in \text{cand}} [\tau(u', v')]^{\alpha} \cdot [\eta_m(u', v')]^{\beta}}
\]

$\eta_m(u, v) \sim$ heuristic factor, proportional to the score function
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $\tau_{max}$
- repeat
  1. Each ant builds a matching
  2. Perform greedy local search on the best matching
  3. Update pheromone trails
- until optimal matching found or stagnation

Greedy randomized construction of a matching by an ant

- Let $m =$ current matching and $cand = V_1 \times V_2 - m$
- Choose $(u, v) \in cand$ w.r.t. the probability

$$p(u, v) = \frac{[\tau(u, v)]^\alpha \cdot [\eta_m(u, v)]^\beta}{\sum_{(u', v') \in cand} [\tau(u', v')]^\alpha \cdot [\eta_m(u', v')]^\beta}$$

$\alpha, \beta \sim$ factor weights (parameters)
ACO algorithm for the graph matching problem

<table>
<thead>
<tr>
<th>Define pheromone trails and initialize them to $\tau_{max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>repeat</td>
</tr>
<tr>
<td>1. Each ant builds a matching</td>
</tr>
<tr>
<td>2. Perform greedy local search on the best matching</td>
</tr>
<tr>
<td>3. Update pheromone trails</td>
</tr>
<tr>
<td>until optimal matching found or stagnation</td>
</tr>
</tbody>
</table>

Greedy Local Search from the best matching $m$

<table>
<thead>
<tr>
<th>Iterate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remove the 3 worse couples of $m$</td>
</tr>
<tr>
<td>Add new couples to $m$ in a greedy way</td>
</tr>
<tr>
<td>Until $m$ is locally optimal</td>
</tr>
</tbody>
</table>

Use a tabu list to prevent the local search from cycling
ACO algorithm for the graph matching problem

- Define pheromone trails and initialize them to $T_{max}$
- repeat
  1. Each ant builds a matching
  2. Perform greedy local search on the best matching
  3. Update pheromone trails
- until optimal matching found or stagnation

Update pheromone trails

- Evaporate: multiply pheromone trails by $(1 - \rho)$
  $\Rightarrow \rho = $ evaporation factor such that $0 \leq \rho \leq 1$
- Reward: add pheromone trails on the couples of the best matching in a quantity proportional to the score
- Bound pheromone trails within $[T_{min}; T_{max}]$
  $\Rightarrow$ prevent stagnation
A first version of ANT-GM has been presented at EvoCOP’05

**Improvements introduced in ANT-GM’06**

1. New pheromonal components
   - ANT-GM’05 lays pheromone on edges: $\tau((u, u'), (v, v'))$
     $\leadsto$ desirability of matching together $u$ with $u'$ and $v$ with $v'$
   - ANT-GM’06 lays pheromone on vertices: $\tau(u, u')$
     $\leadsto$ desirability of matching $u$ with $u'$
   $\Rightarrow$ Better results obtained quicker

2. New ACO strategy
   - ANT-GM’05 based on Ant System
   - ANT-GM’06 based on MAX-MIN Ant System
   $\Rightarrow$ favor exploration $\Rightarrow$ better results

3. ANT-GM’06 integrates local search to improve ant solutions
Experimental Comparison

**Test Suites**
- Test suite 1: Randomly generated instances
- Test suite 2: Instances of [Boeres et al. 2004]

**Considered algorithms**
- ANT-GM’06 $\rightarrow$ ACO without local search
- ANT-GM’06 + LS $\rightarrow$ ACO combined with local search
- RTS $\rightarrow$ Reactive Tabu Search

$\rightarrow$ 20 runs of each algorithm on each instance
Test suite 1: Randomly generated instances

Random generation of couples of similar graphs

- Generation of a couple of similar graphs \((G_1, G_2)\):
  - Random generation of \(G_1\)
  - Apply 15 modifications on \(G_1\) to obtain \(G_2\)

- Non-labelled graphs \(\rightarrow\) harder instances

- Generate 100 couples... and keep the 13th hardest ones

Experimental settings

- ACO: \(\alpha=1, \beta=10, \rho=0.98, nbAnts=20, nbCycles=1000\)

- RTS:
  - \(10 \leq \text{tabuLength} \leq 50, \text{freqUpdate}=1000, \Delta_{\text{tabuLength}}=15\)
  - \(nbMaxMoves=50000\)
  - Iteration on \(\neq\) starting points
  - Until CPU time = time spent by ACO
## Test suite 1: Experimental results

| Problem (|V₁|, |E₁|) | RTS |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   | Best | Avg | Time | Best | Avg | Time | Best | Avg | Time | Best | Avg | Time |
| (80, 200) | 511 | 511.00 | 57 | 511 | 511.00 | 131 | 512 | 511.10 | 140 |
| (80, 240) | 644 | 644.00 | 60 | 644 | 644.00 | 266 | 644 | 644.00 | 230 |
| (80, 320) | 821 | 820.97 | 279 | 821 | 820.50 | 498 | 822 | 821.20 | 660 |
| (80, 340) | 753 | 753.00 | 55 | 753 | 753.00 | 111 | 753 | 753.00 | 130 |
| (80, 360) | 856 | 855.97 | 187 | 855 | 855.00 | 321 | 855 | 855.00 | 249 |
| (80, 360) | 863 | 863.00 | 21 | 863 | 863.00 | 187 | 864 | 863.94 | 565 |
| (90, 300) | 762 | 762.00 | 98 | 762 | 762.00 | 326 | 762 | 762.00 | 213 |
| (90, 320) | 780 | 780.00 | 51 | 780 | 780.00 | 572 | 780 | 780.00 | 409 |
| (90, 320) | 816 | 816.00 | 69 | 816 | 815.45 | 546 | 816 | 815.45 | 602 |
| (100, 260) | 697 | 696.63 | 628 | 697 | 696.90 | 976 | 697 | 697.00 | 812 |
| (100, 300) | 780 | 780.00 | 148 | 780 | 780.00 | 278 | 780 | 780.00 | 279 |
| (100, 320) | 828 | 828.00 | 46 | 828 | 828.00 | 286 | 828 | 828.00 | 218 |
| (100, 360) | 915 | 915.00 | 90 | 915 | 915.00 | 267 | 915 | 915.00 | 152 |

For each algo: **Best** = best score over 20 runs  
**Avg** = average score over 20 runs  
**Time** = average time to find the best score of each run
## Test suite 1: Experimental results

| Problem $(|V_1|, |E_1|)$ | RTS | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &nbsp; | &n
### Test suite 1: Experimental results

| Problem (|V₁|, |E₁|) | RTS | ANT-GM’06 | ANT-GM’06+LS |
|----------|-----|---------|------------|
|          | Best | Avg     | Time       | Best     | Avg     | Time       | Best     | Avg     | Time       |
| (80, 200)| 511  | 511.00  | 57         | 511      | 511.00  | 131        | 512      | 511.10  | 140        |
| (80, 240)| 644  | 644.00  | 60         | 644      | 644.00  | 266        | 644      | 644.00  | 239        |
| (80, 320)| 821  | 820.97  | 279        | 821      | 820.50  | 498        | 822      | 821.20  | 660        |
| (80, 340)| 753  | 753.00  | 55         | 753      | 753.00  | 111        | 753      | 753.00  | 130        |
| (80, 360)| 856  | 855.97  | 187        | 855      | 855.00  | 321        | 855      | 855.00  | 249        |
| (80, 360)| 863  | 863.00  | 21         | 863      | 863.00  | 187        | 864      | 863.94  | 565        |
| (90, 300)| 762  | 762.00  | 98         | 762      | 762.00  | 326        | 762      | 762.00  | 213        |
| (90, 320)| 780  | 780.00  | 51         | 780      | 780.00  | 572        | 780      | 780.00  | 409        |
| (90, 320)| 816  | 816.00  | 69         | 816      | 815.45  | 546        | 816      | 815.45  | 602        |
| (100, 260)| 697  | 696.63  | 628        | 697      | 696.90  | 976        | 697      | 697.00  | 812        |
| (100, 300)| 780  | 780.00  | 148        | 780      | 780.00  | 278        | 780      | 780.00  | 279        |
| (100, 320)| 828  | 828.00  | 46         | 828      | 828.00  | 286        | 828      | 828.00  | 218        |
| (100, 360)| 915  | 915.00  | 90         | 915      | 915.00  | 267        | 915      | 915.00  | 152        |

#### Comparison of RTS with ANT-GM’06 without LS

RTS obtains slightly better results... much quicker
## Test suite 1: Experimental results

| Problem $(|V_1|,|E_1|)$ | RTS | ANT-GM’06 | ANT-GM’06+LS |
|--------------------------|-----|-----------|--------------|
|                          | Best | Avg       | Time | Best | Avg       | Time | Best | Avg       | Time |
| (80, 200)                | 511  | 511.00    | 57   | 511  | 511.00    | 131  | 512  | 511.10    | 140  |
| (80, 240)                | 644  | 644.00    | 60   | 644  | 644.00    | 266  | 644  | 644.00    | 239  |
| (80, 320)                | 821  | 820.97    | 279  | 821  | 820.50    | 498  | 822  | 821.20    | 660  |
| (80, 340)                | 753  | 753.00    | 55   | 753  | 753.00    | 111  | 753  | 753.00    | 130  |
| (80, 360)                | 856  | 855.97    | 187  | 855  | 855.00    | 321  | 855  | 855.00    | 249  |
| (80, 360)                | 863  | 863.00    | 21   | 863  | 863.00    | 187  | 864  | 863.94    | 565  |
| (90, 300)                | 762  | 762.00    | 98   | 762  | 762.00    | 326  | 762  | 762.00    | 213  |
| (90, 320)                | 780  | 780.00    | 51   | 780  | 780.00    | 572  | 780  | 780.00    | 409  |
| (90, 320)                | 816  | 816.00    | 69   | 816  | 815.45    | 546  | 816  | 815.45    | 602  |
| (100, 260)               | 697  | 696.63    | 628  | 697  | 696.90    | 976  | 697  | 697.00    | 812  |
| (100, 300)               | 780  | 780.00    | 148  | 780  | 780.00    | 278  | 780  | 780.00    | 279  |
| (100, 320)               | 828  | 828.00    | 46   | 828  | 828.00    | 286  | 828  | 828.00    | 218  |
| (100, 360)               | 915  | 915.00    | 90   | 915  | 915.00    | 267  | 915  | 915.00    | 152  |

### Comparison of ANT-GM’06 without LS and ANT-GM’06 with LS

Introducing LS improves ANT-GM’06 without increasing CPU-time.
## Test suite 1: Experimental results

| Problem $(|V_1|, |E_1|)$ | RTS | ANT-GM’06 | ANT-GM’06+LS |
|--------------------------|-----|-----------|--------------|
|                          | Best| Avg       | Time         | Best| Avg       | Time         | Best| Avg       | Time         |
| (80, 200)                | 511 | 511.00    | 57           | 511 | 511.00    | 131          | 512 | 511.10    | 140          |
| (80, 240)                | 644 | 644.00    | 60           | 644 | 644.00    | 266          | 644 | 644.00    | 239          |
| (80, 320)                | 821 | 820.97    | 279          | 821 | 820.50    | 498          | 822 | 821.20    | 660          |
| (80, 340)                | 753 | 753.00    | 55           | 753 | 753.00    | 111          | 753 | 753.00    | 130          |
| (80, 360)                | 856 | 855.97    | 187          | 855 | 855.00    | 321          | 855 | 855.00    | 249          |
| (80, 360)                | 863 | 863.00    | 21           | 863 | 863.00    | 187          | 864 | 863.94    | 565          |
| (90, 300)                | 762 | 762.00    | 98           | 762 | 762.00    | 326          | 762 | 762.00    | 213          |
| (90, 320)                | 780 | 780.00    | 51           | 780 | 780.00    | 572          | 780 | 780.00    | 409          |
| (90, 320)                | 816 | 816.00    | 69           | 816 | 815.45    | 546          | 816 | 815.45    | 602          |
| (100, 260)               | 697 | 696.63    | 628          | 697 | 696.90    | 976          | 697 | 697.00    | 812          |
| (100, 300)               | 780 | 780.00    | 148          | 780 | 780.00    | 278          | 780 | 780.00    | 279          |
| (100, 320)               | 828 | 828.00    | 46           | 828 | 828.00    | 286          | 828 | 828.00    | 218          |
| (100, 360)               | 915 | 915.00    | 90           | 915 | 915.00    | 267          | 915 | 915.00    | 152          |

### Comparison of RTS with ANT-GM’06 with LS

ANT-GM’06+LS often obtains better results... but needs more time.
Test suite 2: instances of [Boeres et al. 2004]

7 non-bijective graph matching instances

Match a model graph $G_1$ to an over-segmented image graph $G_2$

Find a multivalent mapping $m$ such that:

- Each vertex of $G_1$ is matched to at least one vertex
- Each vertex of $G_2$ is matched to exactly one vertex
- Only connected set of vertices of $G_2$ are merged
- Maximize a weighted sum w.r.t. given similarity matrices

Experimental settings

- ACO: $\alpha=2$, $\beta=10$, $\rho=0.98$, $nbAnts=20$, $nbCycles=1000$
- RTS:
  - $15 \leq tabuLength \leq 50$, $freqUpdate=5000$, $\Delta_{tabuLength}=15$
  - $nbMaxMoves=50000$
    Only one starting point (Greedy is deterministic)
Test suite 2: Experimental results

| Problem (| | V₁ |, | V₂ |) | LS+ | RTS | ANT-GM’06 | ANT-GM’06+LS |
|---------|---------|-----|---------|---------|
|         | Best    | Best| Time    | Best    | Avg    | Time    | Best    | Avg    | Time    |
| (10, 30)| .5474   | .5481| 0.9     | .5601   | .5598  | 16      | .5608   | .5604  | 15      |
| (10, 30)| .5435   | .5529| 4.6     | .5638   | .5638  | 10      | .5645   | .5641  | 7       |
| (12, 95)| .4248   | .4213| 0.0     | .4252   | .4251  | 211     | .4252   | .4251  | 215     |
| (14, 28)| .6319   | .6333| 2.1     | .6369   | .6369  | 7       | .6376   | .6369  | 5       |
| (30, 100)| .5186  | .5210| 1.3     | .5229   | .5226  | 462     | .5232   | .5228  | 229     |
| (30, 100)| .5222  | .5245| 1.3     | .5263   | .5261  | 456     | .5269   | .5264  | 241     |
| (50, 250)| .5187  | .5199| 81.7    | .5201   | .5201  | 4133    | .5203   | .5202  | 2034    |

- ACO and RTS are (nearly always) better than LS+, the local search approach proposed in [Boeres et al. 04] for this problem.
- ACO obtains better results than RTS but needs much more time.
Two different approaches for the graph matching problem

- **Ant Colony Optimization**
  - New pheromonal strategy
  - Hybridation with LS
  - Obtains better results on harder instances...
  - ...but is time consuming

- **Reactive Tabu Search**
  - Local search guided by a Tabu list
  - The length of the Tabu list is dynamically adapted...
  - ...but other parameters still have to be adapted!
  - Much quicker, but slightly worse results on « hard » instances
Further work

- Improve algorithms
  - Try other local searches for ACO
  - Diversify the search of RTS by iterating from more different starting points

- Compare algorithms on other kinds of GM problems
  - Subgraph isomorphism problems
  - Maximum common subgraph problems
  - Extended graph edit distance

- Applications
  - Measuring the similarity of documents described with RDF
  - Measuring the similarity of images
  - ...