Reactive Tabu Search for Measuring Graph Similarity

Sébastien Sorlin and Christine Solnon

LIRIS, CNRS FRE2672, bât. Nautibus, University of Lyon I
43 Bd du 11 novembre, 69622 Villeurbanne cedex, France
{sebastien.sorlin,christine.solnon}@liris.cnrs.fr

Abstract. Graph matching is often used for image recognition. Different kinds of graph matchings have been proposed such as (sub)graph isomorphism or error-tolerant graph matching, giving rise to different graph similarity measures. A first goal of this paper is to show that these different measures can be viewed as special cases of a generic similarity measure introduced in [8]. This generic similarity measure is based on a non-bijective graph matching (like [4] and [2]) so that it is well suited to image recognition. In particular, over/under-segmentation problems can be handled by linking one vertex to a set of vertices. In a second part, we address the problem of computing this measure and we describe two algorithms: a greedy algorithm, that quickly computes sub-optimal solutions, and a reactive Tabu search algorithm, that may improve these solutions. Some experimental results are given.

1 Introduction

Graphs are often used to model structured objects. In particular, graphs may be used for scene representation [2]: vertices represent scene regions, while edges represent binary relations between regions. In this context, image recognition and classification involves comparing graphs, *i.e.,* matching graphs to identify their common features [9]. This may be done by looking for an exact graph or subgraph isomorphism in order to show graph equivalence or inclusion. However, images are often corrupted by noise and distorsions and the assumption of the existence of an isomorphism is usually too strong. As a consequence, error-tolerant graph matchings such as maximum common subgraph and graph edit distance have been proposed [5, 9]. Such matchings drop the condition that the mapping must preserve all vertices and edges: the goal is to find a "best" mapping, *i.e.,* one which preserves a maximum number of vertices and edges.

Most recently, three different papers proposed to go one step further by introducing multivalent matchings, where a vertex in one graph may be matched with a set of vertices of the other graph:

– In [8], graphs are used to model design objects in a computer-aided design application. In this context, vertices are used to represent object components and one single component of an object may play the same role than a set of components of another object, depending of the granularity of object

description. Therefore, the authors introduce a similarity measure based on multivalent mappings so that one vertex in a graph may be associated with a set of vertices of the other graph.
– In [4], graph matching is used for model-based pattern recognition of brain images. In this application, the assumption of a bijection between regions of the model and the image is too strong: model has a schematic aspect easy to segment while image is noised and usually over-segmented. Therefore, scene recognition is better expressed as a multivalent matching problem where a set of vertices of the scene may be linked to a same vertex of the model.
– In [2], a new graph edit distance is proposed, that introduces two new edit operations —vertex splitting and merging— in order to handle the fact that images may be over- or under- segmented.

Motivation and outline. A first goal of this paper is to "point out" the similarities between these three recent kinds of graph matchings. Another goal is to propose practical algorithms for multivalent graph matchings. Section 2 briefly introduces the graph similarity measure of [8]. Section 3 compares this measure with other graph matchings. Section 4 addresses the problem of computing this measure: we first propose a greedy algorithm that quickly computes an approximation of the similarity and then a reactive Tabu search approach, that may improve the computed approximation. Section 5 presents some experimental results.

2 A generic similarity measure for multi-labeled graphs

A **directed graph** is defined by a couple $G = (V, E)$, where V is a finite set of vertices and $E \subseteq V \times V$ is a set of directed edges. Vertices and edges may be associated with labels that describe their properties. Given a set L_V of vertex labels and a set L_E of edge labels, a **multi-labeled graph** is defined by a triple $G = \langle V, r_V, r_E \rangle$ such that:

– V is a finite set of vertices,
– $r_V \subseteq V \times L_V$ is a relation associating labels to vertices, *i.e.,* r_V is the set of couples (v_i, l) such that vertex v_i is labeled by l,
– $r_E \subseteq V \times V \times L_E$ is a relation associating labels to edges, *i.e.,* r_E is the set of triples (v_i, v_j, l) such that edge (v_i, v_j) is labeled by l. Note that the set E of edges of the graph can be defined by $E = \{(v_i, v_j) | \exists l, (v_i, v_j, l) \in r_E\}$.

We shall call the tuples of r_V and r_E the vertex and edge features of G. The set $descr(G) = r_V \cup r_E$ of all vertex and edge features of a graph G completely describes the graph G.

We now briefly describe the graph similarity measure introduced in [8]; we refer the reader to [8] for more details. This similarity measure is defined for two multi-labeled graphs $G = \langle V, r_V, r_E \rangle$ and $G' = \langle V', r_{V'}, r_{E'} \rangle$, defined over the same sets of vertex and edge labels L_V and L_E, and such that $V \cap V' = \emptyset$.

The first step for measuring graph similarity is to map vertices. The mapping considered here is multivalent, *i.e.*, each vertex of one graph is mapped with a possibly empty set of vertices of the other graph. More formally, a **multivalent mapping** of the two graphs G and G' is a set $m \subseteq V \times V'$ which contains every couple $(v, v') \in V \times V'$ such that vertex v is mapped with vertex v'.

Once a multivalent mapping is defined, the next step is to identify the set of features that are common to the two graphs with respect to this mapping. This set contains all the features from both G and G' whose vertices (resp. edges) are matched by m to at least one vertex (resp. edge) that has the same feature. More formally, the set of common features $descr(G) \sqcap_m descr(G')$, with respect to a mapping m, is defined as follows:

$$\begin{aligned} descr(G) \sqcap_m descr(G') \doteq\; & \{(v,l) \in r_V |\; \exists (v,v') \in m,\; (v',l) \in r_{V'}\} \\ \cup\; & \{(v',l) \in r_{V'} |\; \exists (v,v') \in m,\; (v,l) \in r_V\} \\ \cup\; & \{(v_i,v_j,l) \in r_E |\; \exists (v_i,v_i') \in m, \exists (v_j,v_j') \in m\, (v_i',v_j',l) \in r_{E'}\} \\ \cup\; & \{(v_i',v_j',l) \in r_{E'} |\; \exists (v_i,v_i') \in m, \exists (v_j,v_j') \in m\, (v_i,v_j,l) \in r_E\} \end{aligned}$$

Given a multivalent mapping m, we also have to identify the set of split vertices, *i.e.*, the set of vertices that are mapped to more than one vertex, each split vertex v being associated with the set s_v of its mapped vertices:

$$\begin{aligned} splits(m) = & \{(v, s_v) \;\;|\; v \in V, \quad s_v = \{v' \in V' | (v,v') \in m\},\; |s_v| \geq 2\} \\ \cup\; & \{(v', s_{v'}) \;|\; v' \in V',\; s_{v'} = \{v \in V | (v,v') \in m\},\; |s_{v'}| \geq 2\} \end{aligned}$$

The **similarity** of G and G' with respect to a mapping m is then defined by:

$$sim_m(G,G') = \frac{f(descr(G) \sqcap_m descr(G')) - g(splits(m))}{f(descr(G) \cup descr(G'))} \qquad (1)$$

where f and g are two functions that are introduced to weight features and splits, depending on the considered application. For example, if f is the cardinality function and g is the null function, then the similarity is proportional to the number of common features with respect to the total number of features. If g is the cardinality function, instead of the null function, then the similarity is decreased proportionally to the number of split vertices.

Finally, the **maximal similarity** $sim(G,G')$ of two graphs G and G' is the greatest similarity with respect to all possible mappings:

$$sim(G,G') = \max_{m \subseteq V \times V'} \frac{f(descr(G) \sqcap_m descr(G')) - g(splits(m))}{f(descr(G) \cup descr(G'))} \qquad (2)$$

3 Generic graph similarity and image recognition

The measure of similarity described in section 2 has been first proposed for comparing objects in a computer-aided design application. However, this measure is generic and it may be customized by properly defining functions f and g. In this section, we show how this measure relates to other graph matchings used in image recognition. These matchings are often defined for non labelled graphs. Hence we shall suppose that a non labelled graph is a particular labelled graph such that all vertices have a same label l_v and all edges have a same label l_e.

Graph isomorphism. The graph isomorphism problem between two graphs $G{=}(V,E)$ and $G'{=}(V',E')$ such that $|V|{=}|V'|$ consists in finding a bijective function $\phi : V \rightarrow V'$ such that $(v_1,v_2) \in E$ if and only if $(\phi(v_1),\phi(v_2)) \in E'$.

If functions f and g of formula (2) are defined as cardinality functions, then $sim(G,G'){=}1$ if and only if there exists a mapping m such that $descr(G) \sqcap_m descr(G'){=}descr(G) \cup descr(G')$ and $splits(m){=}\emptyset$, *i.e.,* $sim(G,G'){=}1$ if and only if G and G' are isomorphic.

Partial subgraph isomorphism. The partial subgraph isomorphism problem between two graphs $G{=}(V,E)$ and $G'{=}(V',E')$ such that $|V| \leq |V'|$ consists in finding an injective function $\phi{:}V \rightarrow V'$ such that $(v_1,v_2){\in}E \Rightarrow (\phi(v_1),\phi(v_2)){\in}E'$.

Let us define function g of formula (2) as the cardinality function and function f as a weighted sum where the weight of the features of G (resp. G') is 1 (resp. 0). In this case, $sim(G,G'){=}1$ if and only if there exists a mapping m such that $descr(G) \subseteq descr(G) \sqcap_m descr(G')$ (as $f(descr(G) \cup descr(G')){=}|descr(G)|$) and $splits(m){=}\emptyset$, *i.e.*, $sim(G,G'){=}1$ if and only if there exists a partial subgraph isomorphism.

Subgraph isomorphism. The subgraph isomorphism problem is a special case of partial subgraph isomorphism: it adds the constraint that for each couple $(v_1,v_2) \in V^2$, if (v_1,v_2) is not an edge of G, then $(\phi(v_1),\phi(v_2))$ must neither be an edge of G'.

To check for subgraph isomorphism, we have to add "not-an-edge" labels to all couples of vertices that are not edges, and then to check that all "not-an-edge" labels of G are preserved by the mapping. More formally, given a graph $G{=}(V,E)$, we define the labelled graph $G_{label}{=}(V, r_V, r_E)$ such that $r_V{=}\{(v,l_v)|v \in V\}$ and $r_E{=}\{(u,v,l_e)|(u,v) \in E\} \cup \{(u,v,l_{notE})|(u,v) \in V \times V - E\}$. Let us then define functions f and g of formula (2) as done for the partial subgraph. In this case, $sim(G_{label},G'_{label}){=}1$ if and only if there exists a subgraph isomorphism.

Maximum common partial subgraph (mcps). The $mcps$ of two graphs G and G' is the largest graph (with respect to the number of vertices and edges) which is a partial subgraph of both G and G'.

Let us define function f of formula (2) as the cardinality function, and function g so that split vertices are forbidden (*i.e.,* $g(S){=}{+}\infty$ if $S \neq \emptyset$ and $g(\emptyset){=}0$). In this case, the mapping m that maximizes formula (1) is the mapping which maximizes the number of common features in $descr(G) \sqcap_m descr(G')$, while forbidding split vertices, and therefore it corresponds to a $mcps$.

These definitions can be extended to the problem of finding the maximum common subgraph (mcs) of two graphs, *i.e.,* the problem of finding the largest non partial subgraph. This is done by considering labelled graphs that associate "not-an-edge" labels to all couples of vertices which are not edges, like for the subgraph isomorphism problem. The mcs of two graphs is used in [7, 6, 5] to define the similarity of two graphs as $sim_{mcs}(G,G'){=}\frac{|mcs(G,G')|}{\max(|G|,|G'|)}$. One can appropriately define functions f and g in such a way that the mapping which maximizes formula (1) corresponds to the mapping which maximizes $sim_{mcs}(G,G')$.

Graph edit distance (ged). The *ged* of two graphs G and G' is the minimum cost set of weighted operations needed to transform G into G'. Considered operations are insertions, substitutions, and deletions of vertices and edges. [5] shows that, when considering appropriate weight definitions, *ged* is closely related to the maximum common subgraph, and therefore it is also closely related to the similarity measure of formula (2).

If we consider non labelled graphs (so that one only perform insertion and deletion operations) then, given a mapping m, each vertex or edge feature contained in $descr(G) - (descr(G) \sqcap_m descr(G'))$ (resp. in $descr(G') - (descr(G) \sqcap_m descr(G')))$ corresponds to a vertex or an edge deletion (resp. insertion). Let us then define function f as a weighted sum where weights are defined by operation costs. In this case, the mapping m which maximizes formula (1) gives the set of insertion and deletion operations which minimizes the *ged*.

If we consider labelled graphs (where each vertex and edge is associated with a single label), then substitution operations may be performed to change vertex or edge labels. If two vertices (or edges) are mapped by m but have a different label in G and G', then these labels will not belong to the set of common features $descr(G) \sqcap_m descr(G')$. Hence, one can also define a function f such that the mapping m which maximizes formula (1) gives the set of operations, including substitution operations, which minimizes the *ged*.

Extended ged. In order to compare over- and under-segmented images, [2] proposes to extend *ged* with two new operations: vertex splitting —to split one vertex of G into several vertices of G'— and vertex merging —to merge several vertices of G into one single vertex of G'.

Given a mapping m, the set of couples $(v, s'_v) \in splits(m)$ such that $v \in G$ corresponds to the set of splitting operations whereas the set of couples $(v', s_v) \in splits(m)$ such that $v' \in G'$ corresponds to the set of merging operations. Hence, if function g of formula (2) is defined as a weighted sum, where weights correspond to splitting and merging costs, and if f is defined as done for non extended *ged*, then the mapping which maximizes formula (1) corresponds to the extended *ged*.

Non bijective graph matching problem. This problem is introduced in [4] to find the best matching between models and over-segmented images of brains. Given a model graph $G{=}(V, E)$ and an image graph $G'{=}(V', E')$, a matching is defined as a function $\phi : V \rightarrow \wp(V')$ which associates to each vertex of the model graph G a non empty set of vertices of G', and such that (i) each vertex of the image graph G' is associated to exactly one vertex of the model graph G, (ii) some couples $(v, v') \in V \times V'$ are forbidden so that v' must not belong to $\phi(v)$, and (iii) the subgraph induced by every set $\phi(v)$ must be connected. A weight $s^v(v_i, v'_i)$ (resp. $s^e(e_i, e'_i)$) is associated with each couple of vertices $(v_i, v'_i) \in V \times V'$ (resp. of edges $(e_i, e'_i) \in E \times E'$). The goal is to find the matching which maximizes a function depending on these weights of matched vertices and edges.

One can define functions f and g so that the mapping which maximizes formula (1) corresponds to the best matching as defined in [4]. To handle the fact that couples of vertices and edges are associated with weights, and also that condition (ii) is verified, we have to associate labels to vertices and edges in such a way that the label (v, v') (resp. (e, e')) belongs to $descr(G) \sqcap_m descr(G')$ if and only if v is mapped to v' (resp. e to e'). We can then define f as a weighted sum where a label (v, v') (resp. (e, e')) can be weighted with respect to $s^v(v, v')$ (resp. $s^e(e, e')$) or with negative infinite weight if mapping v to v' is forbidden. Function g is defined in such a way that mappings that do not verify conditions (i) or (iii) are forbidden, *i.e.,* g returns an infinite value when when image vertices are split or merged vertices are not connected.

Discussion

The graph similarity measures proposed in [2] and [4] are based on multivalent mappings (*i.e.*, one vertex may be mapped to several vertices). Both measures are used for image recognition. Indeed, images are often over- or under-segmented so that one has to associate one (under-)segmented region of an image to several (over-)segmented regions of another image.

However, the two similarity measures of [2] and [4] are specific to the addressed problem. In particular, [4] is used for matching brain images to models, and in this context they added specific constraints (*e.g.*, all model vertices must be mapped and each image vertex must be mapped to exactly one model vertex).

The similarity measure proposed in [8] is also based on multivalent mappings, but it is more generic, in the sense that specific constraints or preferences can be expressed thanks to functions f and g. The advantage of such a generic measure, where application-dependent constraints are specified via the two parameters f and g, is that algorithms for computing this measure can be used for different applications. As a counterpart, these algorithms may be less efficient than tailor-made programs, that have been designed for a particular application so that they can exploit specific knowledge to speed-up the solution process.

Moreover, the similarity measure proposed in [8] is defined for multi-labelled graphs, such that each vertex and edge can be associated with a set of labels that describe its properties. Such a multi-labelling could be very useful to describe images more accurately. For example, vertices could be labelled by colours, shapes, or sizes of corresponding regions, while edges could be labelled by distances, relative positions, or relative sizes of corresponding couples of regions.

4 Algorithms for measuring graph similarity

All matching problems described in section 3 are NP-complete or NP-hard problems, except for graph isomorphism, the complexity of which is not exactly stated, and for particular graphs (such as trees or planar graphs) for which some problems are polynomial ([1, 13, 15]).

Complete algorithms have been proposed for computing the mapping which maximizes formula (1) in [8] and for computing the cheapest set of edit operations in [2]. This kind of algorithms, based on an exhaustive exploration of the search space combined with pruning techniques, guarantees solution optimality. However, these algorithms are limited to very small graphs.

Therefore, incomplete algorithms, that do not guarantee optimality but have a polynomial time complexity, appear to be good alternatives. In particular, [4] proposes a randomized construction algorithm —that quickly computes a set of possible non-bijective graph matchings— and a local search algorithm that improves these matchings until a locally optimal point is reached. These algorithms are dedicated to the particular application of matching models with real images.

In this section, we describe three incomplete algorithms —a greedy one, a tabu search one and a reactive tabu search one— for measuring the similarity of two labelled graphs as defined by formula (2). These algorithms are generic in the sense that they are parameterized by the two functions f and g that contain domain-dependant knowledge.

Greedy algorithm. This algorithm has been first proposed in [8]. We briefly describe it because it is used as a starting point of tabu search algorithms. More information can be found in [8].

The algorithm starts from the empty mapping $m = \emptyset$, and iteratively adds to m couples of vertices chosen within the set of candidate couples $cand = V \times V' - m$. At each step, the couple to be added is chosen in a greedy way: we first select from $cand$ the subset of couples that most increase the similarity as defined by formula (2). This subset often contains more than one candidate. To break ties between them, we look ahead the potentiality of each candidate (v, v') by taking into account the features that are shared by edges starting from (resp. ending to) both v and v' and that are not already in $descr(G) \sqcap_{m \cup \{(v,v')\}} descr(G')$. If there are still more than one couple which maximizes these looked-ahead common edge features, then one couple is randomly chosen. This greedy addition of couples to m is iterated until m is locally optimal, *i.e.*, until no more couple addition can increase the similarity.

This greedy algorithm has a polynomial time complexity of $\mathcal{O}((|V| \times |V'|)^2)$, provided that the computation of the f and g functions have linear time complexities with respect to the size of the mapping. As a counterpart of this rather low complexity, this algorithm never backtracks and is not complete. Hence, it may not find the best mapping; moreover, even if it actually finds the best mapping, it cannot be used to prove its optimality. Note however that, since this algorithm is not deterministic, we may run it several times and keep the best found mapping.

Local search. The greedy algorithm returns a "locally optimal" mapping in the sense that adding or removing one couple of vertices to this mapping cannot improve it. However, it may be possible to improve it by adding and/or removing more than one couple to this mapping. A local search [12, 14] tries to improve a solution by locally exploring its neighborhood: the neighbours of a mapping m are the mappings which can be obtained by adding or removing one couple of vertices to m:

$$\forall m \in \wp(V \times V'), neighbourhood(m) = \{m \cup \{(v,v')\} | (v,v') \in (V \times V') - m\}$$
$$\cup \{m - \{(v,v')\} | (v,v') \in m\}$$

From a good initial mapping, computed by the greedy algorithm, the search space is explored from neighbour to neighbour until the optimal solution is found (when the optimal value is known) or until a maximum number of moves have been performed. A heuristic selects the next neighbour to move on at each step.

Tabu meta-heuristic. *Tabu* search [12, 10, 16] is one of the best known heuristic to choose the next neighbour to move on. At each step, one chooses the best neighbour with respect to the same criteria than for the greedy algorithm. Note that this best neighbour may be worse than the current mapping if it is locally optimal. Hence, to avoid to stay around locally optimal mappings by always performing the same moves, a Tabu list is used. This list has a length k and memorizes the last k moves (*i.e.,* the last k added/removed couples) in order to forbid backward moves (*i.e.,* to remove/add a couple recently added/removed). An exception named "aspiration" is added: if a forbidden move reaches a better mapping than the best known mapping, the move is always done. Figure 1 describes the Tabu algorithm for computing formula (2).

```
fonction Tabu($G = \langle V, r_V, r_E \rangle, G' = \langle V', r_{V'}, r_{E'} \rangle, k, optBound, maxMoves$)
return a mapping $m \subseteq V \times V'$
    $m \leftarrow Greedy(G, G')$ ; $best_m \leftarrow m$ ; $nbMoves \leftarrow 0$
    while $sim_m(G, G') < optBound$ and $nbMoves < maxMoves$ do
        $cand \leftarrow \{m' \in neighbourhood(m) / sim_{m'}(G, G') > sim_{best_m}(G, G')\}$
        if $cand = \emptyset$ then/* no aspiration */
            $cand \leftarrow \{m' \in neighbourhood(m) / isNotTabu(m, m', k)$
        end if
        $cand \leftarrow \{m' \in cand / m'$ is maximal wrt formula (2) and look-ahead$\}$
        choose randomly $m' \in cand$
        $makeTabu(m, m', k)$ ; $m \leftarrow m'$ ; $nbMoves \leftarrow nbMoves + 1$
        if $sim_m(G, G') > sim_{best_m}(G, G')$ then $best_m \leftarrow m$ end if
    end while
    return $best_m$
```

Fig. 1. Tabu algorithm

Reactive Tabu search. The length k of the tabu list is a critical parameter that is hard to set: if the list is too long, search diversification is too strong so that the algorithm converges too slowly; if the list is too short, intensification is too strong so that the algorithm may be stuck around local maxima and fail in improving the current solution. To solve this parameter tuning problem, [3] introduces *reactive Tabu search* where the length of the Tabu list is dynamically adapted during the search. To make the Tabu algorithm reactive, one must evaluate the need of diversification of the search. When the same mapping is explored twice, the search must be diversified. In order to detect such redundancies, a hashing key is memorized for each explored mapping. When a collision occurs in the hash table, the list length is increased. On the contrary, when there is no collision during a fixed number of moves, thus indicating that search is diversified enough, one can reduce the list length. Hashing codes are incrementally computed so, this method has a negligible added cost.

5 Experimental results

5.1 Experimental settings

We now experimentally compare the three previously introduced algorithms: iterated greedy (which repeatedly computes 500 mappings with the greedy algorithm, and return the best one), Tabu search and reactive Tabu search. Non reactive version of Tabu search obtains its best average results when the length k of the Tabu list is between 10 and 20. Note that small variations on this length may have an important influence on the results and that the best setting for k is different from one instance to another. Best reactive Tabu search parameters are 10 (resp. 50) for the minimal (resp. maximal) length of the list, 15 for the size of extension and shortening of the list and 1000 moves for the frequency of reducing of the list. By opposition to (non reactive) Tabu search, these parameter settings are more "robust" in the sense that small variations on them do not significantly change performances.

5.2 Graph and subgraph isomorphism problems

We first have done a set of experiments on graph and subgraph isomorphism problems coming from [11]. The iterated greedy algorithm solves 80% of hard graph isomorphism problem instances (regular graphs having 100 vertices, see [11]) in less than 10 seconds (on a Pentium IV 2Ghz, 512Mo of RAM) ; within the same time limit, reactive local search solves 100% of these problems. Moreover, reactive local search can solve 100% of any kind of graph isomorphism problems on graphs having up to 200 vertices in less than 20 seconds.

Subgraph isomorphim problems are much harder: within a limit of 200s, reactive Tabu search solves 66% of subgraph isomorphism problems on graphs with 100 vertices and iterated greedy algorithms only 4,4%. These rather poor results can be explained by the fact that our algorithms do not use any kind of filtering techniques and potentialy explore all kinds of mappings, even multivalent ones.

5.3 Multivalent mapping problems

In order to compare our three algorithms on multivalent mapping problems, we have used a random graph generator to generate "similar" pairs of graphs: it randomly generates a first graph and applies some vertex splitting/merging and some edge and vertex insertion/suppression to build a second graph which is similar to the first one. From the set of transformations, a minimal bound of the similarity is computed. It is used to evaluate our algorithms, by counting the number of times they succeeded in reaching this bound (or an higher one).

When graph components have many different labels, the best mapping is trivially found as nearly all vertices/edges have different labels. Therefore, to obtain harder instances, we have generated 100 graphs such that all vertices and edges have the same label. These graphs have between 80 and 100 vertices and between 200 and 360 edges. The second graph is obtained by doing 5 vertex merging/splitting and 10 edge or vertex insertion/suppression. Functions f and g of formula (2) are defined as cardinality functions.

Comparative results. Each algorithm has been run 200 times on each of the 100 generated problems. 51 problems appeared to be "easy" ones as they were always solved by the iterated greedy algorithm. Over the 49 remaining "harder" problems, that could not be solved by the iterated greedy algorithm, 35 were easily —and always— solved, both by reactive and non reactive Tabu search (in less than 500 moves, corresponding to less than 4 seconds). The 14 last instances appeared to be really hard ones, that needed more than $25,000$ moves to be solved. For these 14 instances, (non reactive) Tabu search succeeded in finding a solution for 64% of the runs whereas reactive Tabu search succeeded for 79% of the runs. Furthermore, reactive Tabu search appeared to be more robust than its non reactive version in the sense that parameter settings influence less the results. So, the reactive Tabu search is more efficient and easier to tune than the non reactive version.

Conclusion

We have shown that the graph similarity measure of [8] is more generic than other graph similarity measures used in image recognition [6,9]. It is based on multivalent graph matching so that over- and under- segmentation problems [2, 4] can be overrided by linking one vertex of a graph to a set of vertices of the other graph. We have given three algorithms with a polynomial complexity: a greedy algorithm, a local search based on Tabu meta-heuristic and an improved version of this search named "reactive Tabu search". These algorithms can compute a similarity measure based on multivalent mapping of two graphs having 100 vertices in a reasonable amount of time.

Further works. We have shown that the similarity measure proposed in [8] is generic in the sense that it can be used to formulate many graph similarity measure. Next step will be to test our algorithms on extended graph edit distance problems [2] and on non-bijective graph matching problems [4], in order to evaluate the efficiency of this formulation in a pratical point of view, and the usefulness of our algorithm in image recognition.

Our generic similarity measure associated with the expressive power of multi-labelled graphs could be used to define new powerful image similarity measures and, therefore, we plan to evaluate them in the field of image recognition and for content-based image querying system.

References

1. A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The design and analysis of computer algorithms.* Addison Wesley, 1974.
2. R. Ambauen, S. Fischer, and H. Bunke. Graph Edit Distance with Node Splitting and Merging, and Its Application to Diatom Identification. In *IAPR-TC15 Wksp on Graph-based Representation in Pattern Recognition*, pages 95–106, 2003.
3. R. Battiti and M. Protasi. Reactive local search for the maximum clique problem. In Springer-Verlag, editor, *Algorithmica*, volume 29, pages 610–637, 2001.
4. M. Boeres, C. Ribeiro, and I. Bloch. A randomized heuristic for scene recognition by graph matching. In *WEA 2004*, pages 100–113, 2004.
5. H. Bunke. On a relation between graph edit distance and maximum common subgraph. *PRL: Pattern Recognition Letters*, 18, 1997.
6. H. Bunke. Graph matching : Theoretical foundations, algorithms, and applications. In *Proc. Vision Interface 2000, Montreal*, pages 82–88, 2000.
7. H. Bunke and X. Jiang. *Graph Matching and Similarity*, volume Teodorescu, H.-N., Mlynek, D., Kandel, A., Zimmermann, H.-J. (eds.): Intelligent Systems and Interfaces, chapter 1. Kluwer Academic Publishers, 2000.
8. P.-A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *5th International Conference on Case-Based Reasoning (ICCBR 2003)*, volume Lecture Notes in Artificial Intelligence 2689-Springer-Verlag, pages 80–95, 2003.
9. D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(3):265–298, 2004.
10. R. Dorne and J. Hao. *Tabu Search for graph coloring, T-coloring and Set T-colorings*, chapter 3. I.H. Osman et al. (Eds.), Kluver Academic Publishers, 1998.
11. P. Foggia, C. Sansone, and M. Vento. A database of graphs for isomorphism and sub-graph isomorphism benchmarking. *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 176 –187, 2001.
12. F. Glover. Tabu search - part I. *Journal on Computing*, pages 190–260, 1989.
13. J.E. Hopcroft and J-K Wong. Linear time algorithm for isomorphism of planar graphs. 6^{th} *Annu. ACM Symp. theory of Comput.*, pages 172–184, 1974.
14. S. Kirkpatrick, S. Gelatt, and M. Vecchi. Optimisation by simulated annealing. In *Science*, volume 220, pages 671–680, 1983.
15. E.M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer System Science*, pages 42–65, 1982.
16. S. Petrovic, G. Kendall, and Y. Yang. A Tabu Search Approach for Graph-Structured Case Retrieval. In *STAIRS 2002*, pages 55–64, 2002.