

---

# Combining Exhaustive and Approximate Methods for Improved Sub-Graph Matching

Thomas Bärecke and Marcin Detyniecki

Université Pierre et Marie Curie - Paris6 UMR 7606, DAPA, LIP6, Paris, France,  
thomas.baerecke@lip6.fr

**Summary.** Reams of different methods have been applied on the inexact graph matching problem in the last decades. In fact, there are two disjoint groups of approaches, exhaustive search and approximate methods. The first ones guarantee that the best solution is always found while the last ones generally have a significantly reduced time complexity at the expense of accepting sub-optimal solutions. This article aims, first, at comparing the two complementary approaches. Secondly, we show that one can bridge the gap between them and that their combination can lead to improved performance, i.e. maintains the guarantee for the best solution while reducing the convergence time.

**Key words:** Evolutionary Computation, Graph Matching, Structural Pattern Recognition, Graphical Models

## 1 Introduction

Despite the exact graph isomorphism problem, which has not yet been shown to be in P nor in NP [1, 2], all other instances of the graph matching problem, of practical importance, are NP-complete [3]. Attributed relational graphs (ARGs) are universal graphical models that occur frequently in structural pattern recognition. For example, content-based image retrieval can rely on graphs modeling the spatial entities and their mutual relationships in a segmented image. The nodes and edges are attributed with feature vectors. The retrieval process in such systems requires efficient methods to compare ARGs. In fact, the computation of the distance between two ARGs implies the resolution of an inexact (also referred to as error-correcting) (sub-)graph isomorphism problem. Even if filtering techniques (e.g., [4]) can reduce the number of matchings for the identification of the best-matching ARG for a given query, there will always remain numerous graph isomorphisms to compute.

Graph matching already has a long tradition in the domain of pattern recognition [5]. In general, it can be divided into exact and inexact matching. Exact matching requires two graphs to be exactly equal whereas inex-

act matching allows certain structural differences between them. Both sub-problems have been addressed by a wide range of methods. These can be classified either as explicit tree search approaches [6, 7, 8], which always find the optimal solution, or as optimization methods which reduce the complexity at the cost of accepting sub-optimal solutions. Recent approaches tackling inexact graph matching include spectral [9, 10], least-squares [11], bayesian [12], and genetic [13, 14] methods.

Comparison between these algorithms as in [15] is usually restricted to algorithms of the same class, i.e. search methods or optimization techniques. This paper focuses on the comparison of the two general approaches. We select one representative of each class and evaluate its performance against one algorithm of the other class. We aim at pointing out general advantages and restrictions and a possible combination of the two complementary methods in the domain of pattern recognition. We choose a state-of-the-art exhaustive tree search method [8] and compare it to two new genetic matching algorithms. These are universally applicable methods since they do not impose any constraints on the ARGs.

The remainder of the paper is organized as follows: We formally define the problem in the next section. In Sect. 3, the A\*-based [6] method is briefly reviewed. The genetic approaches are detailed in Sect. 4. Sect. 5 outlines how to combine the methods. In Sect. 6, we present experimental results on artificially created data. Finally, conclusions are drawn in Sect. 7.

## 2 Problem definition

Let us consider two attributed relational graphs  $S = (V_S, E_S)$  and  $G = (V_G, E_G)$ . We suppose, without loss of generality, that  $S$  is not larger than  $G$ , which means  $|V_S| \leq |V_G|$ . The exact (sub-)graph isomorphism problem consists in finding an injective function  $m : V_S \rightarrow V_G$  mapping each vertex of  $S$  to a distinct vertex of  $G$  such that the resulting node-induced subgraph  $G'$  of  $G$  is isomorph to  $S$ . That means, related nodes, and edges respectively, have the same attributes.

The inexact version of the problem arises when the graphs contain slightly different attributes. Thus, the inexact (sub-)graph isomorphism problem can be stated as the problem of finding a mapping that minimizes some kind of graph distance (e.g., the graph edit distance [16]).

In ARGs, nodes and edges are labeled with application-dependent values. We need to assume that distance metrics for both vertices ( $\delta_V$ ) and edges ( $\delta_E$ ) are provided. The attribute distance is defined by the sum of all edge distances [8]:

$$\mu_a^m(S, G) = \sum_{v \in V_S} \delta_V(v, m(v)) \quad (1)$$

Likewise, the total edge distance defines the relationship distance:

$$\mu_r^m(S, G) = \sum_{v, w \in V_S | v \neq w} \delta_E((v, w), (m(v), m(w))) \quad (2)$$

The overall distance is then defined as a convex combination of these two distances. A context-dependent coefficient usually controls the relative importance of vertex vs. edge distance. For the purpose of this article we suppose that both distances are equally influential which leads to the following equation:

$$\mu_m(S, G) = \mu_a^m(S, G) + \mu_r^m(S, G) \quad (3)$$

### 3 Exhaustive graph matching using an $A^*$ like approach

Exhaustive graph matching is based on explicitly searching the space of all possible assignments between the nodes of the graphs [17]. A solution is obtained incrementally by growing a partial assignment of the vertices of one (sub-)graph  $S$  to the vertices of the other graph  $G$ . The solution space is organized as a tree, where the  $k^{th}$  level contains all partial assignments of the first  $k$  entities of  $S$  [16, 17]. The evaluation of a partial assignment is based on Eqs. 1 and 2 restricted to the vertices already included in the mapping. Since the distance of two graphs under any partial mapping is monotonically non-decreasing with the tree level for any branch, partial assignments scoring a distance over a predefined threshold of maximum acceptable dissimilarity  $\mu_{max}$  can be safely discarded without risk of false dismissal. The  $A^*$  algorithm [6] performs a depth-first search, which always extends the partial mapping toward the local optimum, and which backtracks when the scored distance of the current assignment runs over a maximum acceptable threshold. When the inspection reaches a complete mapping, a match under the threshold is found. At this point, the global optimum is not guaranteed, but the obtained scored distance implies a stricter threshold for acceptable distance that is then used to efficiently extend the search until the global optimum is found.

Several approaches extend the  $A^*$  scheme using an admissible heuristic to increase the cost of the current partial assignment with a lower boundary of the future cost. In Berretti's [8] case, the extension cost estimate is obtained using the solution of a bipartite matching problem with a discrete relaxation algorithm. This requires only polynomial time. Based on this value, partial assignments that can not lead to a final match with acceptable similarity can be discarded earlier, thus accelerating the convergence of the algorithm while preserving result's optimality.

### 4 Genetic approaches

Genetic algorithms [18] are optimization methods inspired by the natural evolution process. They operate on a set (population) of solution candidates, the

chromosomes. These are randomly initialized and undergo genetic operators such as selection, recombination, and mutation. This creates very good approximations after several iterations.

GAs provide decent solutions in a very short time. Moreover, they can return a complete candidate solution *at any time*, its quality increasing with the time (i.e. number of generations). The user is able to balance the trade-off between accuracy and run-time directly by choosing an appropriate stop criteria.

GAs explore large heterogeneous search spaces very fast at the expense of a non-exhaustive exploitation. This means that not all possible solutions are evaluated. In particular, it can occur that the standard GA finds a good region and the search does not continue into the direction of the optimal solution. Genetic local search (GLS) algorithms are GAs which are combined with local search strategies as for instance 2-opt. We implemented both, a standard GA and a GLS. The difference between them is that the GLS does not apply mutation at all, but performs a 2-opt local search on each crossover offspring.

#### 4.1 Encoding and selection

A permutation is the most efficient way to encode a particular mapping between the nodes of two graphs of the same size. In this case, the application of the corresponding recombination operators is straightforward. In order to avoid repair mechanisms in the case of subgraph matching (due to operators capable of creating illegal solutions) and to generally prevent the operators from becoming too complex, we use the larger graph size as chromosome length. During the fitness evaluation, we do only take into account the portion of the chromosomes that actually corresponds to an existing node of the subgraph.

The selection of individuals for reproduction is based on the accumulated distances of nodes and edges (see Eq. 3), and follows a tournament strategy. A set of tournament size individuals is chosen randomly and the best individual produces offspring. This strategy is very efficient since it neither requires normalization nor ordering of the individuals by their fitness and decreasing fitness (error) functions can be directly incorporated.

#### 4.2 Recombination and mutation

Although varied recombination operators for permutation encodings are available, none was specially developed for graph matching. Since the influence of any node pair, and thus of a specific allele in the gene, depends equally on all other alleles, it is reasonable to apply an unbiased strict position-based crossover operator. Thus, we combine the idea of position-based crossover [19] with the idea of uniform crossover. This differs from the original position-based crossover in the point that the order from the second parent is not imposed onto the child, but as many alleles as possible are placed on the same

locus as of the parents. The average number of mappings inherited from the two parents is approximately equal. An allele that has the same position in both parents always keeps this position.

A swap operator is used for the mutation in the GA. It simply exchanges the loci of two alleles.

### 4.3 Local search strategy

The local search strategy of the GLS replaces each offspring by the dominating gene of its neighborhood. We use the 2-opt algorithm in which the set of chromosomes emerging from a given chromosome by exchanging two values is considered. This space is exhaustively searched for the locally optimal individual which replaces the initial chromosome. The evaluation of a 2-opt-switch is computationally less expensive ( $\mathcal{O}(n)$ ) than the evaluation ( $\mathcal{O}(n^2)$ ) of entire chromosomes, since only the nodes and edges whose mappings changed are considered.

## 5 Combined algorithms

We combine the Berretti's algorithm[8] with both versions of our genetic approach. A simplified pseudo-code is given in Alg. 1 to illustrate the approach. We first perform a genetic search. The scored distance of the best individual in the final generation is obviously an upper bound for the globally minimal distance. Thus, we perform Berretti's matching algorithm with this value as initial maximal acceptance threshold. This ensures that the optimal solution is always found. The advantage over the original form of Berretti's algorithm is that several non-promising branches of the search tree can be discarded earlier. In the following, the combined approaches will be referred to as CGA, and CGLS respectively.

## 6 Experimental results

We compare the algorithms using a set of randomly attributed complete graphs. A graph  $G$  of size  $n$  is complete when it consists of  $n$  vertices and any node pair is connected by an edge. During graph generation, both edge and node attributes are uniformly chosen random values from the unit interval.

A node-induced subgraph  $S$  is extracted from each graph  $G$ . Technically, we create a (uniform) random permutation of the nodes of  $G$  and select its first  $o : o \leq n$  nodes in order to establish  $S$ . In our tests,  $o$  is either  $n$  (graph isomorphism) or  $n - 5$  (subgraph isomorphism).

We perform the matching of every subgraph  $S$  against all original graphs. We consider two test cases: a *positive* case for isomorph graphs and a *negative*

```

Input:  $S = (V_S, E_S)$  and  $G = (V_G, E_G)$ ,  $|V_S| \leq |V_G|$ ;      /* Two Graphs */
/* First step: genetic algorithm with elite replacement          */
1 Initialize Population p randomly ;
2 repeat
3   Tournament Selection ;
4   for  $i \leftarrow 0$  to crossover.probability * size(p) do
5     ( $p[i], p[i+1]$ )  $\leftarrow$  Crossover ( $p[i], p[i+1]$ ) ;
6      $i \leftarrow i+2$  ;
7   forall Gene  $g \in p$  do
8     if Algorithm.type=GLS then
9       Local Search;
10    else
11      Mutate with mutation.probability ;
12 until fitness(best individual) has not decreased during last n generations ;
13  $f =$  fitness (best individual);
    /* Second step: Berretti's algorithm                          */
14  $acct \leftarrow f$  ;                                          /* Use f as acceptance threshold */
15  $m \leftarrow \emptyset$  ;                                       /* Start with empty mapping */
16 repeat
17   Evaluate partial mapping m by  $\delta_m(S, G)$  ;
18   Estimate future extension cost  $c_m(S, G)$  ;                /* Lower boundary */
19   Add best matching node pair  $(v_S, v_G)$  to m ;
20   if  $\delta_m(S, G) + c_m(S, G) > acct$  then
21     Backtrack ;
22 until m is complete or no valid extension is possible ;
23 if m is complete then
24    $acct \leftarrow \delta_m(S, G)$  ;
25   Continue Search ;
26 else
27   return last complete mapping ;

```

**Algorithm 1:** Simplified pseudo-code of the combined algorithm

one that aims at finding the optimal (distance-minimizing) mapping between non-corresponding graphs. We compare the execution time and accuracy of all algorithms. The results of the exhaustive approach serve as benchmarks for the genetic approaches. Since, in the negative test case, the exhaustive search becomes intractable with increasing graph sizes, this type of test is only performed for small graphs. Since, in the positive case, the time complexity of Berretti's algorithm does not climb that fast, we evaluate the behavior for larger graph sizes by focusing only on isomorph graph pairs, i.e. we ensure that an optimal solution with zero distance exists.

### 6.1 Solution quality for pure genetic approaches

The relative distance of the solutions obtained by the pure genetic approaches with respect to the optimal solution for small graph sizes is illustrated in

**Table 1.** Error of the pure genetic approaches

Algorithm	Graph Size		Positive case		Negative case	
	S	G	Errors	Mean distance	Mean	Maximum
GLS	10	10	0	0	0.74 %	10.33 %
GA	10	10	1	0.4788	6.93 %	20.48 %
GLS	10	15	0	0	2.21 %	13.83 %
GA	10	15	3	2.0456	13.86 %	39.21 %

**Table 2.** Time for graph matching. On the left graph matching between graphs of size 10 is performed. On the right subgraphs of size 10 are matched against graphs of size 15

	Graph isomorphism			Subgraph isomorphism		
	Positive	Negative	Mean	Positive	Negative	Mean
A* Ref. Time	31.3 ms	12683 ms	11418 ms	76.6 ms	286930 ms	258250 ms
A* Rel. Time	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
GLS	208.95 %	1.40 %	1.46 %	191.64 %	0.13 %	0.14 %
GA	35.46 %	0.34 %	0.35 %	38.64 %	0.02 %	0.02 %
CGLS	233.87 %	86.04 %	86.08 %	201.70 %	83.38 %	83.38 %
CGA	54.95 %	93.36 %	93.35 %	71.41 %	97.43 %	97.43 %

Tab. 1. For the negative case, let  $m'$  be the globally optimal mapping. Then, for any mapping  $m$ , the term  $(\mu_m(S, G)/\mu'_m(S, G)) - 1$  describes the gap to  $m'$ . In the positive case, we cannot use this measure for the assessment of the solution quality since  $\mu'_m(S, G) = 0$ . Thus, we use the distance imposed by  $m$  itself (i.e.,  $\mu_m(S, G)$ ) and the absolute number of mismatches (denoted as Errors).

We observe that the GLS obtains nearly perfect results and always finds the optimal solution in the positive case. The solutions of the standard GA are of lower quality. Our tests on increased graph sizes showed that the quality of the solution decreases with the graph size. This effect is especially strong for the standard GA whereas the error of the GLS does only increase marginally.

### 6.2 Time complexity

As shown in Tab. 2 the execution time of the GA is lower than for the GLS. However, this is inversed when we observe the combined versions, since the better quality solution provide lower maximum acceptance thresholds for Berretti’s algorithm. An interesting result is that the combined algorithms outperform the exhaustive search although it is included in them. In the case of exact matching Berretti’s algorithm outperforms the GLS. However, due to the huge difference of the absolute execution times this has nearly no influence on the mean values.

Figures 1 and 2 show the effect of increasing graph sizes on the performance of the algorithms for the case of exact graph matching, both for the

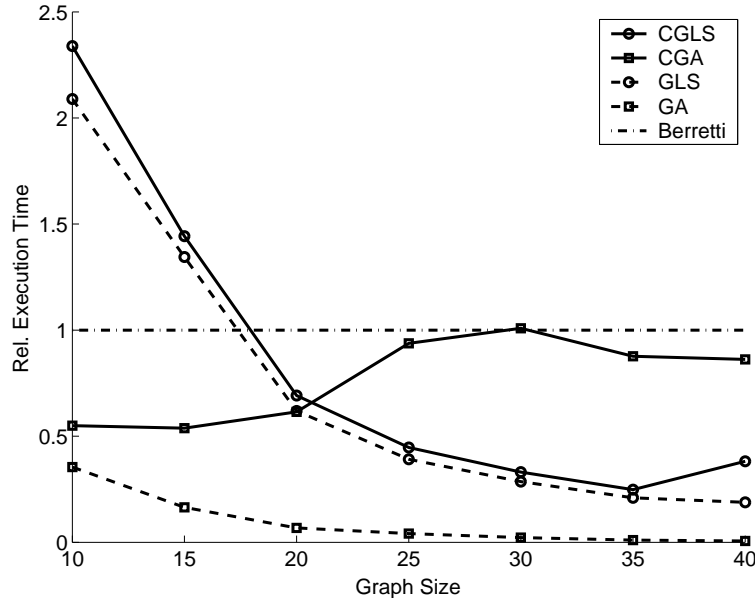


Fig. 1. Relative execution time, graph matching

graph isomorphism and subgraph isomorphism problem. The execution time is normalized by the execution time of Berretti’s algorithm. In other words, the relative execution time  $t$  is obtained by dividing the absolute execution time of a given algorithm by the absolute execution time of Berretti’s approach for the each graph size. This allows to compare the evaluation of execution times vs. the exhaustive reference algorithm on a linear scale.

We observe, that the execution time of the genetic approaches grows much slower than that of the exhaustive one. Additionally, the combined versions compare favorably to the A\*-like approach. The difference between pure GA and its corresponding combined versions depends on the quality of the intermediate solutions. We also observe that the better the solution, the lower the time spent for its verification.

## 7 Conclusions

The implementation of local search into the genetic approach enhances the solution quality significantly. Indeed, the GLS found every exact matching until the size of 35. The pure GA obtains only very poor approximations for increasing graph sizes.

We combine two complementary types of algorithms. We observe, that this combination can obtain better results than with either of them in general. We keep the best-solution guarantee from Berretti’s algorithm and reduce its

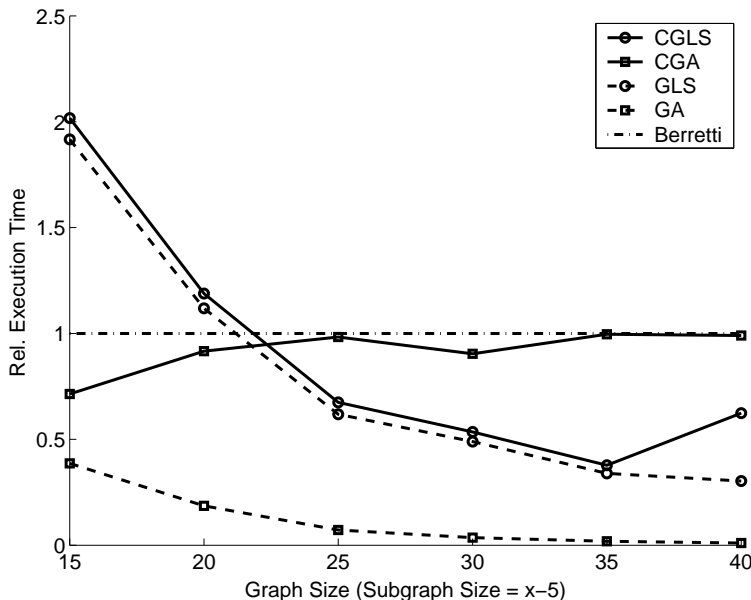


Fig. 2. Relative execution time, subgraph matching

execution time by supplying it with an approximate solution from the GA. A decreased starting value of the acceptable threshold speeds it up significantly. Therefore, in applications, where the maximal acceptable error is a priori known and sufficiently small, the exhaustive algorithm is preferable. For the general case one should prefer combined genetic approaches. In fact, if the GA provides high quality solution the combination does not influence the time complexity very much. However, we did not compare the algorithms on real-world data so far and we expect that this would favor the GLS more than its corresponding combined approach.

We are planning to apply them to content-based image retrieval in the near future, in particular we are going to address the problem of identifying a person based on a two-dimensional image using a database of three-dimensional face models.

### Acknowledgements

The authors thank Stefano Berretti for his support and for providing the source code of his algorithm. We also acknowledge the valuable comments of the reviewers.

## References

1. Köbler, J., Schöning, U., Torán, J.: *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhäuser, Boston (1993)
2. Arvind, V., Kurur, P.P.: Graph isomorphism is in SPP. *Information and Computation* **204** (2006) 835–852
3. Garey, M., Johnson, D.: *Computers and Intractability - A Guide to the Theory of NP-Completeness*. Freeman and Company (1979)
4. Irniger, C., Bunke, H.: Decision trees for error-tolerant graph database filtering. In: *Graph-Based Representations in Pattern Recognition*. Volume 3434 of LNCS. (2005) 301–311
5. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. *Int. Journal of Pattern Recognition and Artificial Intelligence* **18**(3) (2004) 265–298
6. Ullmann, J.R.: An algorithm for subgraph isomorphism. *J. ACM* **23**(1) (1976) 31–42
7. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(10) (2004) 1367–1372
8. Berretti, S., Del Bimbo, A., Vicario, E.: Efficient matching and indexing of graph models in content-based retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(10) (2001) 1089–1105
9. Shokoufandeh, A., Macrini, D., Dickinson, S.J., Siddiqi, K., Zucker, S.W.: Indexing hierarchical structures using graph spectra. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(7) (2005) 1125–1140
10. Robles-Kelly, A., Hancock, E.R.: Graph edit distance from spectral seriation. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**(3) (2005) 365–378
11. van Wyk, B.J., van Wyk, M.A.: Kronecker product graph matching. *Pattern Recognition* **36**(9) (2003) 2019–2030
12. Caetano, T.S., Caelli, T., Schuurmans, D., Barone, D.A.C.: Graphical models and point pattern matching. *IEEE Trans. Pattern Anal. Mach. Intell.* **28**(10) (2006) 1646–1663
13. Myers, R., Hancock, E.: Least-commitment graph matching with genetic algorithms. *Pattern Recognition* **34**(2) (2001) 375–394
14. Suganthan, P.N.: Structural pattern recognition using genetic algorithms. *Pattern Recognition* **35**(9) (2002) 1883–1893
15. Cesar, Jr., R.M., Bengoetxea, E., Bloch, I., Larrañaga, P.: Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms. *Pattern Recognition* **38**(11) (2005) 2099–2113
16. Eshera, M., Fu, K.S.: A graph distance measure for image analysis. *IEEE Trans. Syst. Man, Cybern.* **14** (1984) 398–407
17. Tsai, W., Fu, K.S.: Error-correcting isomorphism of attributed relational graphs for pattern analysis. *IEEE Trans. Syst. Man Cybern.* **9** (1979) 757–768
18. Mitchell, M.: *An introduction to genetic algorithms*. MIT Press, Cambridge (1996)
19. Syswerda, G.: Schedule optimization using genetic algorithms. In Davis, L., ed.: *Handbook of Genetic Algorithms*, New York, Van Nostrand Reinhold (1991) 332–349