

# Memetic Algorithms for Inexact Graph Matching

Thomas Bärecke and Marcin Detyniecki

**Abstract**—The noise-robust matching of two graphs is a hard combinatorial problem with practical importance in several domains. In practical applications, a unique solution for a given instance can not be defined, i.e. the actual solution may be outscored by some other due to noise effects arising during feature extraction. Soft computing approaches in general provide fast but not necessarily globally optimal solutions. In this case, the lack of guarantee of the global optimum is not a real drawback, since the uncertainty already arises in the problem definition. This paper discusses the application of memetic algorithms on the error-correcting graph isomorphism problem. We show that permutation encoding is robust enough to allow addressing both, the matching problem for graphs of the same size, and the subgraph matching problem. Since gene order information is meaningless in this particular case, a strict position based crossover is applied providing better performance than the popular PMX. We evaluate our algorithm on a synthetic data set with larger graph sizes than used in traditional, exact approaches and other permutation-based genetic approaches.

## I. INTRODUCTION

Finding a match between two attributed relational graphs (ARGs) or deciding whether they are isomorph is a central task in structural pattern recognition. ARGs offer an abstract description of inherently structured real-world objects, for instance segmented images in the case of content-based image retrieval. In general, they describe entities of a structured object and their mutual relationships. In actual applications two graphs extracted from the same object are rarely exactly equal. The structural differences are due to uncertainties in the feature extraction process, noisy environments, etc.. Therefore, graph matching has to be noise-tolerant, i.e. identify correspondence even if small differences persist.

The problem of graph matching already has a long tradition in the domain of pattern recognition [1]. It consists of finding a one-to-one mapping between the nodes of two graphs so that corresponding nodes and edges have equal attributes. In the case of subgraph matching such a mapping is searched between the smaller graph and all node induced subgraphs of the larger one. Furthermore, we differentiate between exact and inexact matching problems. Exact matching requires two graphs to be exactly equal whereas inexact matching allows certain attribute or even structural differences between them. Thus, the inexact case is usually stated as a distance minimization.

Active research on graph matching problems from various points of view has been performed for several decades [1], [2] and a plethora of different approaches has been applied

to it. Although most practically important graph matching problems are NP-complete [3], it is not yet known whether the graph isomorphism problem for unweighted graphs is NP-complete or lies in P [4]. For restricted graph classes, however, efficient algorithms are available [5], [6].

The error-correcting (or inexact) graph isomorphism problem is a NP-complete [7] combinatorial optimization problem. Hard computing methods are only able to solve very small problem instances in reasonable time. Indeed, the imprecise nature of the problem favors the application of soft computing approaches.

The diversity of existing graph matching algorithms can be classified either as explicit tree search approaches (e.g., [8]) which are usually extensions of Ullmann's [9] depth first search algorithm and always find the optimal solution, or as approximate optimization methods which reduce the complexity at the cost of accepting sub-optimal solutions. Recent approaches tackle inexact graph matching with spectral [10], [11], least-squares [12], bayesian [13], and genetic [14], [15] methods. It has been shown, that the combination of an exhaustive and an approximate method may lead to better results than any of the approaches alone [16].

Genetic algorithms have proved their ability to efficiently solve other NP-hard combinatorial optimization problems (e.g., the traveling salesman problem [17] or the quadratic assignment problem [18]). However, the efficiency of genetic approaches highly depends on the chosen parameters. A crucial design issue is thus to define an appropriate representation of a solution candidate as a chromosome. Graph matching can be stated as an optimization problem over the permutation group. Hence, in this paper we focus on permutation encoding. Only few genetic algorithms address the error-correcting graph isomorphism problem using this particular encoding [19], [20]. The obtained results leave some space for significant performance improvements. In particular, these approaches have been validated on graphs of relatively small size (up to around 20 nodes) only.

We extend permutation encoding to graphs of different sizes, addressing the problem of subgraph matching. The graph variations are limited to changes in attribute distances and deleting nodes from one graph. A generalization considering insertions and deletions in both graphs unfortunately breaks up the permutation property and hence is out of scope for this study. Several algorithms have been proposed to address this problem. The different approaches itself vary in the genetic operators: Standard (one-point) crossover and mutation operators are applied to this representation in [15], [21]. Operators using special graph properties enhance the performance. Cross et al. propose a crossover aiming at exchanging consistent subgraphs [22]. Results also show

T. Bärecke and M. Detyniecki are with the Université Pierre et Marie Curie - Paris6 UMR 7606, DAPA, LIP6, 104 av du Président Kennedy, Paris, France (phone: +33 1 44 27 88 87; fax: +33 1 44 27 70 00; email: thomas.baerecke@lip6.fr).

that memetic algorithms perform better than pure genetic approaches [22], [23], [14].

Another roughly related idea is to incorporate a divide-and-conquer strategy in the crossover operator. For color crossover [24], the gene is split into substrings. Each substring contains only nodes of a single type. Then, standard crossover operators are applied on corresponding substrings. This approach is restricted to applications, where nodes may be classified into several meaningful, distinct classes.

In this paper, we present a permutation-based memetic algorithm for the error-correcting graph matching problem. Experiments are performed on artificially created data with different noise levels. The remainder of this paper is organized as follows: In Sect. II the problem is stated formally. The following section details our approach. Sect. IV presents the test scenario, illustrates and interprets experimental results. Conclusions are drawn in Sect. V.

## II. PROBLEM STATEMENT

An attributed relational graph  $G$  is defined over a vertex set  $V$ . Each node is labeled with an attribute  $a \in A_V$ . Their relationships are labeled with edge attributes  $e \in A_E$ .  $G$  is formally defined [8] by:

$$G = (V, \alpha, \beta) \quad (1)$$

The edge set is implicitly given by  $E = V \times V$ . The functions  $\alpha : V \rightarrow A_V$  and  $\beta : V \times V \rightarrow A_E$  assign the attributes to nodes and edges. The attributes are application-dependent content-descriptors. We assume that appropriate distance measures  $\delta_V$  and  $\delta_E$  are defined:

$$\delta_V : A_V \times A_V \rightarrow \mathcal{R} \quad (2)$$

$$\delta_E : A_E \times A_E \rightarrow \mathcal{R} \quad (3)$$

These have to fulfill the metric properties of non-negativity, auto-similarity, symmetry, and triangle inequality.

For simplicity of notation we will write in the following, for the distance of any two vertices  $v_1$  and  $v_2$  from two graphs  $G_1 = (V_1, \alpha_1, \beta_1)$  and  $G_2 = (V_2, \alpha_2, \beta_2)$ ,  $\delta_V(v_1, v_2)$  instead of  $\delta_V(\alpha_1(v_1), \alpha_2(v_2))$  and analogically for edge distances.

The error-correcting graph isomorphism problem, given two graphs of equal size  $G = (V, \alpha, \beta)$  and  $G' = (V', \alpha', \beta')$ , consists in finding the mapping  $m : V \rightarrow V'$  between the nodes of the two graphs that minimizes some dissimilarity metric, for example the graph edit distance [25]. Here, we use the *joint distance* [8] which is a convex combination of the (vertex) attribute distance and the relationship distance. The distance of two graphs implied by the mapping  $m$  is defined by:

$$\begin{aligned} \delta(G, G', m) &= \lambda \sum_{v \in V} \delta_V(v, m(v)) + \\ &(1 - \lambda) \sum_{v, w \in V} \delta_E((v, w), (m(v), m(w))) \end{aligned} \quad (4)$$

The parameter  $\lambda \in [0, 1]$  influences the relative importance of attribute distance, and relationship distance respectively,

thus, it is highly application dependent. Since in this study we focus on a general description of our algorithm and evaluate it on artificial data, we assume equal impact of both, i.e.  $\lambda = 0.5$ . For simplicity, we slightly change the original definition by normalizing the weights to 1. Thus, the absolute distance between two graphs is defined as the minimum over all legal mappings:

$$\delta(G, G') = \min_m \delta(G, G', m) \quad (5)$$

Any legal solution candidate has an equivalent representation in the space of permutations of  $|V'|$  elements. In fact, consider that the elements of  $V$ , and  $V'$  respectively, are arbitrarily numbered from 1 to  $|V'|$ . Then the corresponding mapping  $m$  will contain all pairs of nodes  $(v, v')$  that have the same number. In other words, any permutation of the numeration of  $V'$  implies a unique mapping and vice versa.

Moreover, the error-correcting *subgraph* isomorphism problem also aims to minimize Eq. 4, but this time for graphs of different sizes. If we assume  $|V| < |V'|$ , without loss of generality, the solution space can be transferred into the space of permutations of size  $|V'|$ . Although this leads to a memory overhead of  $\mathcal{O}(|V'| - |V|)$  and to several permutations having identical mappings, it facilitates the application of our memetic algorithm as we will detail in Sect. III.

## III. MEMETIC ALGORITHM

Genetic algorithms [26], [27], [28] are population based optimization methods inspired by the natural evolution process. They provide decent solutions for hard problems 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 may balance, inside of certain boundaries, the trade-off between accuracy and run-time directly by choosing an appropriate stop criterion. Genetic algorithms cover 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, in the standard genetic algorithm, that a good region is found, but the search does not continue into the direction of its optimal solution.

While genetic algorithms perform well for global search but poorly for convergence to local optima [29], [30], local improvement heuristics do the opposite. Thus, combining both approaches can significantly enhance the search performance [31], [32]. Those hybrid approaches are now usually referred to as memetic algorithms [33], [34], [35], [36], [37]. Several authors have used other terms, for example hybrid genetic algorithm [38], Lamarckian evolution [39], genetic local search [40], or cultural algorithm [41]. Another major advantage over standard genetic algorithms lies in the explicit use of existing domain-specific knowledge in order to refine each individual. All this leads to higher quality solutions and faster convergence.

The main drawback of genetic algorithms is that in general they are over-parameterized methods and the preliminary optimization of the parameters is not a trivial task. In the

following, we will list and explain the parameters with the highest impact on the performance for the inexact graph matching problem. This study does not aim at optimizing all of them, rather at showing the general feasibility of an algorithm solving the inexact (sub-)graph matching problem and at illustrating its performance.

#### A. Encoding, fitness function, and selection strategy

Given the problem statement of Sect. II it is quite obvious that the most natural encoding scheme for the solution candidates of the graph matching problem is permutation encoding. Since this is a common scheme, several well-studied operators are available [42], [43]. The fitness of an individual solution is determined by the distance of the currently matched graph pair, and it is given by Eq. 4. An individual is fitter if it has a smaller value. Tournament selection, different from, for instance fitness proportionate or rank based selection strategies, does not require neither normalization nor ordering of fitness values. It can further operate directly on an inverse fitness function and is computationally very inexpensive. Unfortunately, it requires an additional parameter, the tournament size, which is necessarily an integer. Thus, it does not allow to continuously influence the selection pressure. Since our primary goal is to present a general algorithm and not to optimize its various parameters, we did not implement a selection strategy that allows continuous parameterization of the selection pressure.

#### B. Robustness for subgraph matching

In the case of subgraph matching, a straightforward encoding, where the chromosome length would equal the size of the smaller graph, would imply the development of new genetic operators. On the one hand, this is still close to permutation encoding, but on the other hand the operators designed for permutation encodings would need to be at least extended with methods to incorporate and exchange missing values and/or repair mechanisms for possibly illegal chromosomes. This would increase not only the complexity of the algorithm, but also the number of parameters.

Instead, the distinction between phenotype and genotype permits the application of standard permutation encoding even for subgraph matching. The trick is quite simple: the mapping is always considered in the direction from the smaller graph to the larger one. However, the chromosome consists of a complete permutation of the vertices of the larger graph. This is the genotype of the solution. The phenotype, i.e. its external meaning, however, is determined only by a subset (whose size equals the size of the smaller graph) of its alleles. Technically, the fitness evaluation is restraint to the nodes contained in this subset. The remaining alleles may be considered as the memory of missing values, maintaining the permutation properties during the application of all genetic operators, which includes local search. This results in a robust encoding that enables the same instance of the memetic algorithm to treat the error-correcting graph isomorphism problem as well as its subgraph equivalent.

#### C. Recombination, mutation and termination

Crossover operators for permutation encoding problems can be divided into three families, depending on which kind of information is preserved. Order based crossover (e.g., OX [44]) preserves relative position, while position-based crossover (e.g., CX [45]) focuses on the absolute position of the alleles. A popular hybrid approach is partially mapped crossover (PMX [42]), which exploits simultaneously the ordering and the absolute positions. The choice of the operators is strongly application dependent, a well-performing operator for one specific optimization problem may perform badly on other kinds of problems. For the graph matching case, the order of the values is not meaningful at all, since by definition all labels are randomly assigned. The influence of a specific allele on the fitness function is determined by its relations to all other alleles. In other words, each allele interacts equally with all other alleles. Thus, the chromosome can be considered as an array representation of an unordered set of corresponding node pairs. Therefore, we suggest a new strict position based crossover (PBX):

- (1) Create a list  $l$  containing each allele and its positions in the two parents.
- (2) Shuffle the list in order to traverse it in a random order and arbitrarily select the alleles that will swap positions.
- (3) Construct the children by traversing the shuffled list. In case of collision (another allele already placed in the desired position) in either of the children store this allele and its positions.
- (4) Try to place the non-used alleles in the position they had in the other parent.
- (5) Fill the genes with the eventually still missing alleles in the list order.

The algorithm is illustrated on an example in Fig. 1.

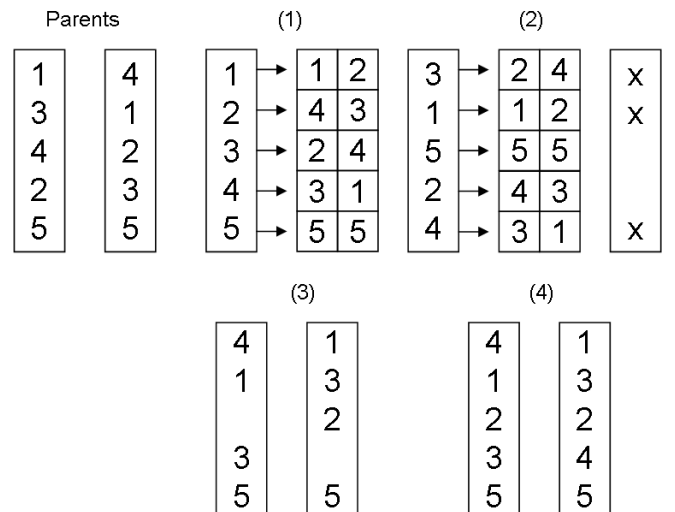


Figure 1. Crossover operator

This operator has some special properties, compared to other position-based methods like CX or Syswerda's method [46]. The relative order from the second parent is not

imposed onto the child but as many alleles as possible are placed on the same locus from either of the parents. More importantly, the average number of mappings inherited from the two parents is approximately equal and an allele that has the same position in both parents always keeps its position.

For the mutation we could have suggested a swap operator, which simply exchanges the loci of two alleles. Its effect would be to maintain the population diversity during evolution. However, our preliminary experiments showed that for the memetic algorithm, in contrast to the standard genetic algorithm, the use of mutation does not improve neither solution quality nor convergence speed. Moreover, the local search step works like an intelligent swap mutation, and can thus replace the ordinary mutation completely.

The local search strategy of the memetic algorithm consists in replacing each offspring by the dominating gene of its neighborhood. The neighborhood of a given solution  $s$  is defined as the set of solutions that can be reached by applying an elementary operator  $M : S \rightarrow P(S)$  to the solution  $s$ . It is denoted as  $N(s) = M(s) \subset S$ . We use the two-opt heuristic in which the set of chromosomes emerging from a given chromosome by exchanging two values is considered, i.e.  $M$  is the swap mutation. This space is exhaustively searched for the locally optimal individual, which replaces the initial chromosome. A specific two-opt switch may be measured just by the distance reduction it imposes on the graphs, instead of calculating the fitness function for the entire chromosome. In fact, consider the nodes  $x$  and  $y$  of Graph  $G$  whose corresponding nodes in Graph  $G'$  are  $x' = m(x)$  and  $y' = m(y)$ . The mapping  $m'$  that is build by switching  $x'$  and  $y'$  is almost equal to  $m$  except that  $m'(x) = m(y)$  and  $m'(y) = m(x)$ . The local search replaces  $m$  by the mapping  $m'$  in its two-opt neighborhood that minimizes the function:

$$g = \delta(G, G', m') - \delta(G, G', m) \quad (6)$$

For all elements  $v$  and  $w$  of  $V' = V \setminus \{x, y\}$  the following equations hold:

$$\delta_V(v, m(v)) = \delta_V(v, m'(v)) \quad (7)$$

$$\delta_E((v, w), (m(v), m(w))) = \delta_E((v, w), (m'(v), m'(w))) \quad (8)$$

This means that the evaluation of the two-opt switches requires only the computation of the nodes and edges whose mappings changed. Thus, the function  $g$  is collapses to:

$$\begin{aligned} g = & \delta_V(x, m'(x)) - \delta_V(x, m(x)) \\ & + \delta_V(y, m'(y)) - \delta_V(y, m(y)) \\ & + \sum_{v \in V, w \in \{x, y\}} \delta_E((v, w), (m(v), m'(w))) \\ & - \delta_E((v, w), (m(v), m(w))) \end{aligned} \quad (9)$$

Thus, such a switch is evaluated in linear time while the evaluation of entire chromosomes requires ( $\mathcal{O}(n^2)$ ). Indeed,

the 2-opt algorithm is an *informed* mutation operator performing the best possible swap. Thus, it replaces the simple mutation operator completely in the memetic algorithm.

Our algorithm terminates when the fitness of the best individual does not change anymore for a fixed number of iterations. For completeness we list the exact parameters of our approach in Tab. I.

Table I  
ADDITIONAL PARAMETERS

Parameter	Value
Tournament size	2
Termination	10
Crossover Probability	0.9
Swap Mutation Probability	0.0
2-opt Probability	1.0
Population size	100
UPMX Parameter	0.33
Elitism	1

## IV. EXPERIMENTAL RESULTS

### A. Test data generation

Following the style of the benchmark used in the literature [19], [20] we compare our approach on sets of 50 graph pairs and perform three independent runs on each pair. For each graph pair one graph is randomly initiated, i.e. its node and edge attributes are set to uniformly distributed pseudo-random numbers from the interval  $[0, 1]$ . The corresponding graph is then obtained by shuffling the nodes of the original graph and adding noise in the form of uniform distributed random numbers from the interval  $[-\epsilon, +\epsilon]$  to each of its attributes. The permutation used for shuffling the nodes and the corresponding distance are later used as references. We evaluate our algorithm with  $\epsilon$ -values between 0 and 0.2 in steps of 0.05. We evaluate each algorithm based on  $50 \times 3 \times 5 \times 5 = 3750$  independent runs. In the following we will refer to the three algorithms only by their crossover strategy since apart from that they are identical. PBX denotes our approach based on strict position based crossover, while PMX and UPMX denote their corresponding algorithm instances.

### B. Accuracy evaluation

We measure the accuracy of the approach for different graph sizes and noise levels, as the fraction of runs where the best mapping found by the memetic algorithm implies a graph distance equal or less than the reference solution. Note, that for higher noise levels, it is in theory possible that the noise introduction increases the distance such that other mappings with smaller values are possible. In real-world applications, this has also to be considered a real match, since no one can really know if noise was the cause. For this reason, we also accept mappings that differ from the created one if they are actually better or equal. Fig. 2 shows the results obtained with the memetic algorithm using a strict position based crossover for different graph sizes and noise levels. The results are compared with two variations of our algorithm, which replace the strict position based crossover

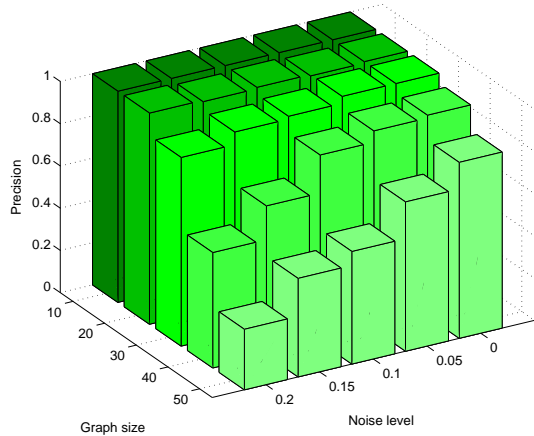


Figure 2. Precision with position based crossover (PBX)

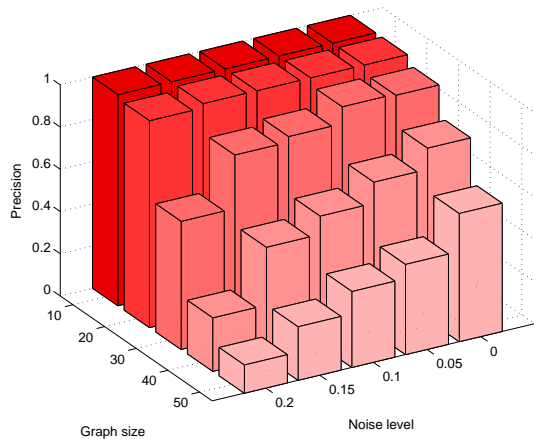


Figure 3. Precision with partially mapped crossover (PMX)

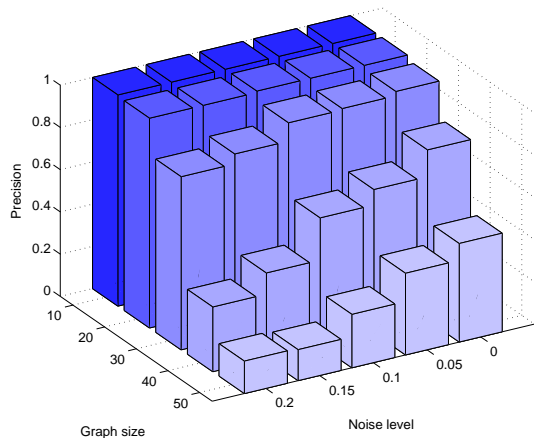


Figure 4. Precision with uniform partially mapped crossover (UPMX)

by (1) the PMX operator (on Fig. 3) as it was used in the related literature [19], [20] and (2) its more recently proposed uniform version UPMX [47] (on Fig. 4).

Wang et al. [19] and Torres-Velazquez et al. [20] do only report results for graph sizes up to 20. In fact, we observe

that our algorithm with any of the implemented crossover operators provides the accurate solution in 99.9% of the runs for graphs up to 20 nodes. This is actually better than Wang’s approach, but equal to the results reported by Torres-Velazquez. For larger graphs the accuracy decreases and a difference between the application of the three crossover operators becomes visible. Strict position based crossover outperforms both PMX variants. PMX’s and UPMX’s overall solution quality is equal, but it seems that UPMX is better for graph sizes of about 30, while it is slightly outperformed by PMX for graph sizes of 50.

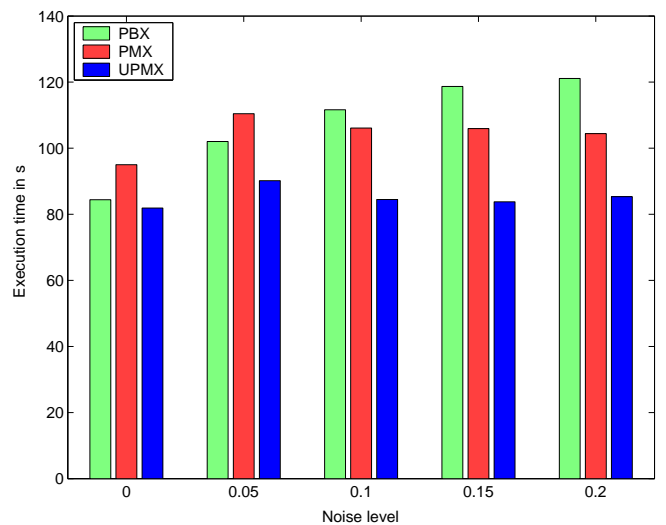


Figure 5. Runtime vs. noise

### C. Runtime evaluation

In this study, we are mainly interested in the effect that different graph sizes and noise levels have on the runtime. We report the dependence of the runtime from the noise level on Fig. 5 and from graph size on Fig. 6. The execution time is highly dependent on the platform and implementation. The average time needed to perform a single matching of size 50 is slightly less than 90 seconds on a 3.2 GHz Pentium IV PC.

The influence of the noise level is in general limited. While it had almost no impact on the PMX and UPMX versions, the PBX version converges slower at higher levels. However, one should take into account that Fig. 5 illustrates the behavior at the highest evaluated graph size (i.e. 50), where the accuracy of the PBX is also much better. Thus, we think that the lower computation time of PMX and UPMX can be considered as premature convergence.

We observe that the graph size has a significant influence on the runtime. The approach incorporating UPMX also converges most quickly in this case. The differences are, however, small for graph sizes up to 40. Only for graphs of size 50, one can observe a significant advantage of UPMX and a small advantage of PMX vs. the PBX approach. As stated above, we consider that the accuracy-runtime trade-off favors PBX.

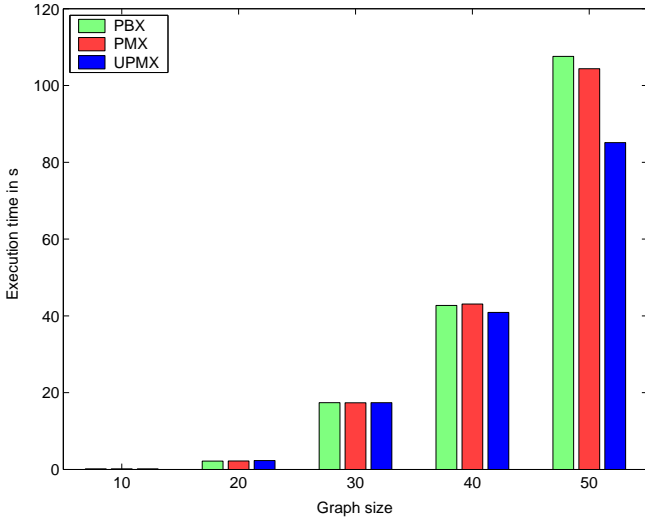


Figure 6. Runtime vs. graph size

#### D. Comparing the population evolution

In order to better compare and understand the three crossover strategies, we re-evaluated the algorithms on a single problem instance with graph size 30 and noise level 0.1 and evaluated the population at each generation using the following measures (the values are accumulated, or averaged respectively, over 150 runs, i.e., three runs on each of the 50 graph pairs). The results of PBX are shown with green lines, those of PMX in red, and those of UPMX in blue color.

1) *False Mappings*: Here we count the number of false elementary mappings in the best individual of the population, i.e. a value of 2 means that two alleles have to be exchanged in order to comply with the reference solution. We only accept the initial mapping, and not any other even if it is better due to noise effects. Fig. 7.

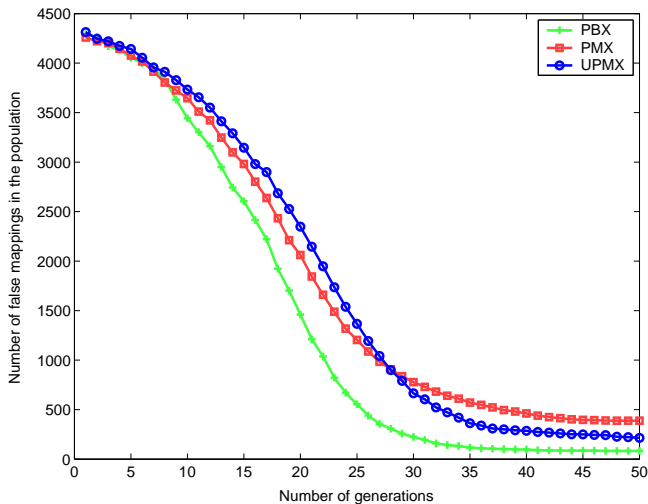


Figure 7. Number of false mappings

2) *Fitness*: This value corresponds to the value of  $\delta$  from Eq. 4 (normalized by the corresponding reference value),

which is to be minimized. The continuous lines illustrate the values of the elite individual while the dotted lines show the population's average. Fig. 8.

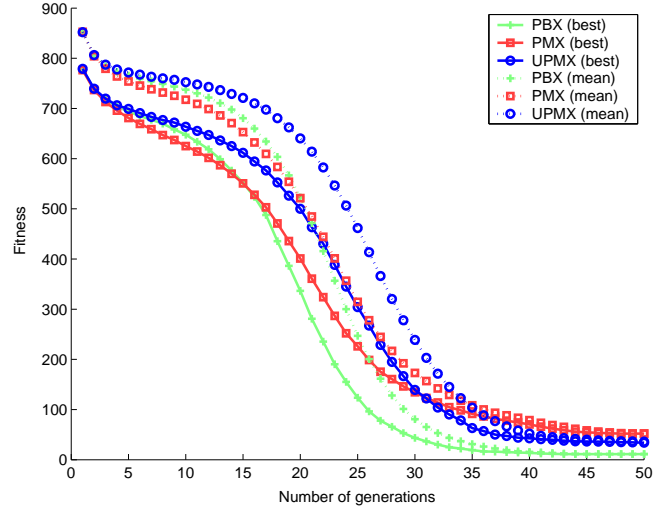


Figure 8. Fitness

3) *Diversity*: The population diversity is measured by the number of distinct fitness values in the population. Although, this value would be even more reliable if the genes would be directly compared, we choose this heuristic to reduce the complexity of the evaluation. In fact, it should occur rarely that two distinct mappings imply exactly the same distance on the graphs. Fig. 9.

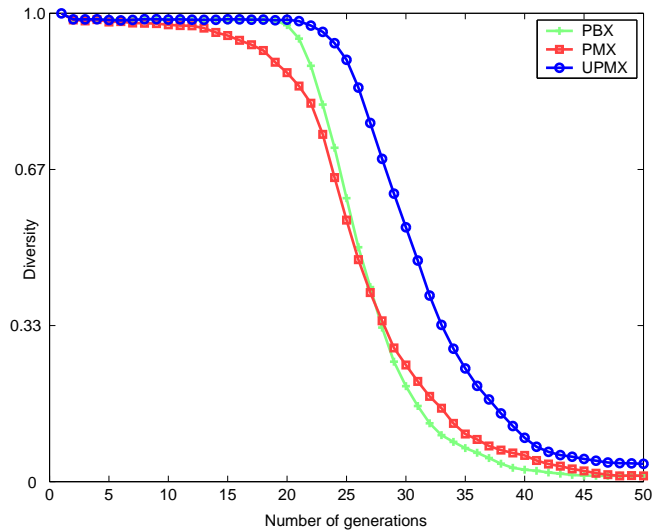


Figure 9. Diversity

The number of false mappings and the fitness values show that PBX leads to better solution qualities. PMX shows better fitness values at the very beginning, but is not able to increase its quality like UPMX and PBX and thus ends up with the worst solution quality. We further observe that the behavior is equal when considering the population average or even only the best individual. In all approaches, the mean fitness

converges to the fitness of the best individual toward the end of the run. This means that the population converges to a uniform population composed of individuals with very small fitness variations. Thus, it contains many copies of relatively good individuals.

The population diversity illustrated in Fig. 9 also supports this fact since we observe a decrease similar to the improvement of the best individual. However, this occurs some generations behind. This means, once the algorithm finds some good solutions it still continues the exploration for a small number of generations and if by then no significant improvements have occurred the diversity automatically decreases.

## V. CONCLUSIONS AND FUTURE WORK

In this paper, we present a memetic algorithm, which is based on permutation encoding, for the inexact (sub-)graph matching problem. The problem is NP-complete and the best known general hard computing approaches have exponential complexity. Thus, they can only be applied to small graphs of the order of 20 nodes in a reasonable time. Memetic algorithms (and soft computing approaches in general) provide faster solutions. On the one hand, one can argue that they do not guarantee the globally optimal solution. On the other hand, there is no unique well-defined solution since even the globally optimal solution might contain false mappings that score better than the real ones due to noise. The definition of the inexact graph-matching problem already inherits this uncertainty and it is reasonable to tackle a vague problem with a powerful method providing approximate solutions.

We show that a permutation based encoding can even be applied to the subgraph matching problem by differentiating between the phenotype and genotype of the individuals. Thus, the same instance of the memetic algorithm can handle both problems. Since the influence of an allele on the fitness function requires its relations to all other alleles, and thus, gene order information has no effect on the individual fitness, we conclude that crossover operators preserving order or relative position are not well-suited to the graph matching problem. Our strict position-based crossover (PBX) operator preserves exclusively absolute positions. The algorithm can handle attribute variances as well as the subgraph problem. However, other graph variations like node insertions or deletions that might occur in real-world applications are not reflected.

We perform experiments on graph sizes, which are much larger than those of the (few) similar approaches. We observe that our approach nearly always (in 99.9% of our experiments) converges to the global optimal solution for the maximum graph sizes reported by other permutation-based approaches. We show that our strict position-based crossover PBX outperforms the PMX operator used in the literature as well as its more recent uniform version UPMX.

In the future, we are planning to apply the algorithm in the context of content-based image retrieval. We also think that better local search heuristics and further parameter

optimization in general might still improve the algorithm's performance.

## REFERENCES

- [1] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *Int. Journal of Pattern Recognition and Artificial Intelligence*, vol. 18, no. 3, pp. 265–298, May 2004.
- [2] R. Read and D. Corneil, "The graph isomorphism disease," *Journal of Graph Theory*, vol. 1, pp. 339–363, 1997.
- [3] M. Garey and D. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.
- [4] J. Köbler, U. Schöning, and J. Torán, *The Graph Isomorphism Problem: Its Structural Complexity*. Boston: Birkhäuser, 1993.
- [5] J. Hopcroft and R. Karp, "An  $n^{5/2}$  algorithm for maximum matchings in bipartite graphs," *SIAM Journal on Computing*, vol. 2, pp. 225–231, 1973.
- [6] J. Hopcroft and J. Wong, "A linear time algorithm for the isomorphism of planar graphs," *Proc. 6th Annual ACM Symp. on Theory of Computing*, pp. 172–184, 1974.
- [7] W. Tsai and K.-S. Fu, "Error-correcting isomorphism of attributed relational graphs for pattern analysis," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 757–768, 1979.
- [8] S. Berretti, A. D. Bimbo, and E. Vicario, "Efficient matching and indexing of graph models in content-based retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, pp. 1089–1105, 2001.
- [9] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, no. 1, pp. 31–42, 1976.
- [10] A. Shokoufandeh, D. Macrini, S. J. Dickinson, K. Siddiqi, and S. W. Zucker, "Indexing hierarchical structures using graph spectra," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1125–1140, 2005.
- [11] A. Robles-Kelly and E. R. Hancock, "Graph edit distance from spectral seriation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 365–378, 2005.
- [12] B. J. van Wyk and M. A. van Wyk, "Kronecker product graph matching," *Pattern Recognition*, vol. 36, no. 9, pp. 2019–2030, 2003.
- [13] T. S. Caetano, T. Caelli, D. Schuurmans, and D. A. C. Barone, "Graphical models and point pattern matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1646–1663, 2006.
- [14] R. Myers and E. R. Hancock, "Least-commitment graph matching with genetic algorithms," *Pattern Recognition*, vol. 34, no. 2, pp. 375–394, 2001.
- [15] P. N. Suganthan, "Structural pattern recognition using genetic algorithms," *Pattern Recognition*, vol. 35, no. 9, pp. 1883–1893, 2002.
- [16] T. Bärecke and M. Detyniecki, "Combining exhaustive and approximate methods for improved sub-graph matching," in *Proc. Int. Workshop on Advances in Pattern Recognition*, Plymouth, UK, July 2007, to appear.
- [17] P. Merz and B. Freisleben, "Memetic algorithms for the traveling salesman problem," *Complex Systems*, vol. 13, no. 4, pp. 297–345, 2001.
- [18] D. Tate and A. Smith, "A genetic approach to the quadratic assignment problem," *Computers & Operations Research*, vol. 22, no. 1, pp. 73–83, 1995.
- [19] Y.-K. Wang, K.-C. Fan, and J.-T. Horng, "Genetic-based search for error-correcting graph isomorphism," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 27, no. 4, 1997.
- [20] R. Torres-Velázquez and V. Estivill-Castro, "A memetic algorithm guided by quicksort for the error-correcting graph isomorphism problem," in *EvoWorkshops*, S. Cagnoni, Ed., 2002, pp. 173–182.
- [21] R. M. Cesar, Jr., E. Bengoetxea, I. Bloch, and P. Larrañaga, "Inexact graph matching for model-based recognition: Evaluation and comparison of optimization algorithms," *Pattern Recognition*, vol. 38, no. 11, pp. 2099–2113, Nov. 2005.
- [22] A. D. J. Cross, R. C. Wilson, and E. R. Hancock, "Inexact graph matching using genetic search," *Pattern Recognition*, vol. 30, no. 6, pp. 953–970, 1997.
- [23] A. D. Cross and E. R. Hancock, "Convergence of a hill climbing genetic algorithm for graph matching," in *EMMCVPR*, ser. LNCS, E. Hancock and M. Pelillo, Eds., vol. 1654, 1999, pp. 221–236.

- [24] M. Singh, A. Chatterjee, and S. Chaudhury, "Matching structural shape descriptions using genetic algorithms." *Pattern Recognition*, vol. 30, no. 9, pp. 1451–1462, 1997.
- [25] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 3, pp. 353–362, 1983.
- [26] J. Holland, *Adaptation in natural and artificial systems*. Ann Arbor, MI: The University of Michigan Press, 1975.
- [27] D. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*. Reading, MA: Addison Wesley, 1989.
- [28] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd revised and extended ed.)*. Berlin, Germany: Springer, 1996.
- [29] J.-M. Renders and S. Flasse, "Hybrid methods using genetic algorithms for global optimization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 2, 1996.
- [30] C. Houck, J. Joines, and M. Kay, "Comparison of genetic algorithms, random start, and two-opt switching for solving large location-allocation problems," *Computers & Operations Research*, vol. 23, no. 6, pp. 587–596, 1996.
- [31] L. Davis, *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.
- [32] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, April 1997.
- [33] P. Moscato, "On evolution, optimization, genetic algorithms and martial arts: Towards memetic algorithms," California Institute of Technology, Pasadena, CA, Tech. Report. Caltech Concurrent Computation Program 826, 1989.
- [34] Y. Ong, M. Lim, N. Zhu, and K. Wong, "Classification of adaptive memetic algorithms: A comparative study," in *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 36, no. 1, 2006, pp. 141–152.
- [35] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: model, taxonomy, and design issues." *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 5, pp. 474–488, 2005.
- [36] P. Merz, "Memetic algorithms for combinatorial optimization problems: Fitness landscapes and effective search strategies," Ph.D. dissertation, Parallel Syst. Res. Group, Dept. Electr. Eng. Comput. Sci., Univ. Siegen, Germany, 2000.
- [37] W. Hart, "Adaptive global optimization with local search," Ph.D. dissertation, Univ. California, San Diego, CA, 1994.
- [38] H.-F. Wang and K.-Y. Wu, "Hybrid genetic algorithm for optimization problems with permutation property," *Computers & Operations Research*, vol. 31, pp. 2453–2471, 2004.
- [39] D. L. Whitley, V. S. Gordon, and K. E. Mathias, "Lamarckian evolution, the baldwin effect and function optimization," in *Parallel Problem Solving from Nature - PPSN III*, Y. Davidor, H.-P. Schwefel, and R. Männer, Eds. Springer, 1994, pp. 6–15.
- [40] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transactions on Systems, Man, and Cybernetics, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, 1998.
- [41] R. Reynolds, "An introduction to cultural algorithms," in *3rd Conf. on Evolutionary Programming*, 1994.
- [42] D. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," in *Proc. of the 1st Int. Conf. on Genetic Algorithms*, Mahwah, NJ, USA, 1985, pp. 154–159.
- [43] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley, and C. Whitley, "A comparison of genetic sequencing operators," in *Proc. of the 4th International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. San Mateo, CA: Morgan Kaufman, 1991, pp. 69–76.
- [44] L. Davis, "Applying adaptive algorithms to epistatic domains," in *Proc. Int. Joint Conf. on Artificial Intelligence*, 1985.
- [45] I. Oliver, D. Smith, and J. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Proc. of the 2nd Int. Conf. on Genetic Algorithms*, Mahwah, NJ, USA, 1987, pp. 224–230.
- [46] G. Syswerda, "Schedule optimization using genetic algorithms," in *Handbook of Genetic Algorithms*, L. Davis, Ed. New York: Van Nostrand Reinhold, 1991, pp. 332–349.
- [47] V. Cicerello and S. Smith, "Modeling GA performance for control parameter optimization," in *GECCO-2000: Proc. of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers, 2000, pp. 235–242.