CP Models for Maximum Common Subgraph Problems

Samba Ndojh Ndiaye, Christine Solnon

CP Models for Maximum Common Subgraph Problems

Samba Ndojh Ndiaye and Christine Solnon

Université de Lyon, CNRS
Université Lyon 1, LIRIS, UMR5205, F-69622, France

Abstract. The distance between two graphs is usually defined by means of the size of a largest common subgraph. This common subgraph may be an induced subgraph, obtained by removing nodes, or a partial subgraph, obtained by removing arcs and nodes. In this paper, we introduce two soft CSPs which model these two maximum common subgraph problems in a unified framework. We also introduce and compare different CP models, corresponding to different levels of constraint propagation.

1 Introduction

Graphs are used in many applications to represent structured objects such as, for example, molecules, images, or biological networks. In many of these applications, it is necessary to measure the distance between two graphs, and this problem often turns into finding a largest subgraph which is common to both graphs [1]. More precisely, we may either look for a maximum common induced subgraph (which has as many nodes as possible), or a maximum common partial subgraph (which has as many arcs as possible). Both problems are NP-hard in the general case [2], and have been widely studied, in particular in bioinformatic and chemoinformatic applications [3, 4].

In this paper, we study how to solve these problems with CP. In Section 2, we recall definitions and we describe existing approaches. In Section 3, we introduce two soft CSPs which model these problems in a unified framework. In Section 4, we introduce different CP models, corresponding to different levels of constraint propagation. In Section 5, we experimentally compare these different models.

2 Background

2.1 Definitions

A graph G is composed of a finite set N_G of nodes and a set $A_G \subseteq N_G \times N_G$ of arcs. We implicitly consider directed graphs, such that each arc is a directed couple of nodes. Results introduced in this paper may be generalized to non directed graphs in a straightforward way, by associating two directed arcs (u, v) and (v, u) with every non directed edge linking u and v.

Let G and G' be two graphs. G is *isomorphic* to G' if there exists a bijective function $f : N_G \to N_{G'}$ which preserves arcs, *i.e.*, $\forall (u, v) \in N_G \times N_G, (u, v) \in$

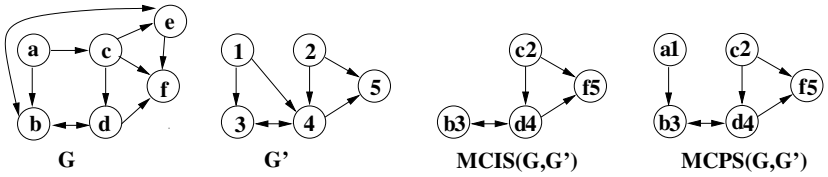

Fig. 1. Example of two graphs G and G' and their MCIS and MCPS.

$A_G \Leftrightarrow (f(u), f(v)) \in A_{G'}$. An *induced subgraph* is obtained by removing nodes, *i.e.*, G' is an induced subgraph of G if $N_{G'} \subseteq N_G$ and $A_{G'} = A_G \cap N_{G'} \times N_{G'}$. A *partial subgraph* is obtained by removing nodes and arcs, *i.e.*, G' is a partial subgraph of G if $N_{G'} \subseteq N_G$ and $A_{G'} \subseteq A_G \cap N_{G'} \times N_{G'}$.

We denote $G_{\downarrow S}$ the subgraph obtained by keeping a subset S of components of G: If S is a subset of nodes, then $G_{\downarrow S}$ is the induced subgraph obtained by keeping these nodes (*i.e.*, $N_{G_{\downarrow S}} = S$ and $N_{G_{\downarrow S}} = A_G \cap S \times S$); if S is a subset of arcs, then $G_{\downarrow S}$ is the partial subgraph obtained by keeping these arcs (*i.e.*, $N_{G_{\downarrow S}} = \{u \in N_G \mid \exists v \in N_G, (u,v) \in S \vee (v,u) \in S\}$ and $A_{G_{\downarrow S}} = S$).

A *common subgraph* is a graph which is isomorphic to subgraphs of G and G'. The similarity of two graphs is usually defined by means of the size of a common subgraph [1]: the larger the subgraph, the more similar the graphs. The size of a subgraph is defined differently whether we consider induced or partial subgraphs: A *Maximum Common Partial Subgraph (MCPS)* is a common partial subgraph which has a maximum number of arcs, whereas a *Maximum Common Induced Subgraph (MCIS)* is a common induced subgraph which has a maximum number of nodes. Fig. 1 displays two graphs and an example of MCPS and MCIS.

2.2 Existing complete approaches for solving MCIS and MCPS

Most complete approaches are based on a reformulation of the problem into a maximum clique problem in a compatibility graph (whose nodes correspond to couples of nodes that may be matched and edges correspond to pairs of compatible nodes) [5–7]. McGregor [8] proposes a different approach based on Branch & Bound: Each node of the search tree corresponds to the matching of two components, and a bounding function evaluates the number of components that can still be matched so that the current branch is pruned as soon as this bound becomes lower than the size of the largest known common subgraph.

Conte *et al* [9] compare these 2 approaches within a same programming framework on a large database of graphs. They show that no approach is outperforming the other: The best performing approach varies when changing graph features.

Vismara and Valery [10] show how to model and solve MCIS and MCPS with constraint programming. They consider particular cases of these two problems, where the subgraph must be connected, and they introduce a global connectivity constraint for this purpose. However, they ensure that node matchings are injective by using a set of binary difference constraints. They compare CP with a clique-based approach, and show that CP obtains better results.

3 Modeling MCIS and MCPS problems as soft CSPs

In this section, we introduce two soft CSPs which respectively model MCIS and MCPS problems for two graphs G and G'. These two models mainly differ with respect to their variables: For MCIS, variables are associated with nodes of G, whereas for MCPS, variables are associated with arcs. In both cases, the value assigned to the variable associated with a component (node or arc) of G corresponds to its matched component in G'. As some components may not be matched, we introduce a joker value $\perp$ which denotes the fact that a component is not matched. Hence, for MCIS, the domain of every variable x_u associated with a node $u \in N_G$ is $D(x_u) = N_{G'} \cup \{\perp\}$ whereas, for MCPS, the domain of every variable x_{uv} associated with an arc $(u, v) \in A_G$ is $D(x_{uv}) = A_{G'} \cup \{\perp\}$.

In both cases, there are two different kinds of constraints. A first set of binary constraints ensures that neighborhood relations defined by arcs are preserved. For MCIS, these binary constraints ensure that adjacency relations between matched nodes are preserved: Given two variables x_u and x_v respectively associated with nodes u and v of G, we define

$$C_{arc}(x_u, x_v) \equiv (x_u = \perp) \vee (x_v = \perp) \vee ((u, v) \in A_G \Leftrightarrow (x_u, x_v) \in A_{G'})$$

For MCPS, these binary constraints ensure that incidence relationships between matched arcs are preserved: Given two variables x_{uv} and x_{wy} respectively associated with arcs (u, v) and (w, y) of G, we define

$$C_{arc}(x_{uv}, x_{wy}) \equiv (x_{uv} = \perp) \vee (x_{wy} = \perp) \vee (R((u, v), (w, y), x_{uv}, x_{wy}))$$

where R is a predicate which checks that (u, v) and (w, y) have the same incidence relationships as the arcs of G' assigned to x_{uv} and x_{wy}, *i.e.*,
$R((u, v), (w, y), (u', v'), (w', y')) \equiv (u = v \Leftrightarrow u' = v') \wedge (u = w \Leftrightarrow u' = w') \wedge (u = y \Leftrightarrow u' = y') \wedge (v = w \Leftrightarrow v' = w') \wedge (v = y \Leftrightarrow v' = y') \wedge (w = y \Leftrightarrow w' = y')$.

Finally, we have to express that the matching must be injective (as two different components of G must be matched to two different components of G'). This kind of constraint could be modeled with a global *allDiffExcept⊥(X)* constraint which enforces all variables in X to take distinct values, except those variables that are assigned to a joker $\perp$ value [11]. To find a maximum common subgraph, we search for a partial injective matching which matches as many components as possible, *i.e.*, we have to minimize the number of variables assigned to $\perp$. This could be achieved by adding an *atmost*$(b - 1, X, \perp)$ constraint each time a feasible solution σ is found, where b is the number of variables assigned to $\perp$ in σ. However, this model achieves a weak filtering because it separates the evaluation of the cost function from the *allDiff* constraint.

Stronger filterings may be achieved by using optimization constraints, which relate constraints with cost variables to be optimized, as proposed in [12]. In particular, the soft *allDiff* constraint [13] relates a set X of variables to an additional *cost* variable which is constrained to be equal to the number of variables of X that should change their value in order to satisfy the *allDiff(X)* constraint, and which must be minimized (we consider variable-based violation costs).

To find an injective partial matching which minimizes the number of non matched components, we introduce an additional variable $x_\perp$ whose domain is $D(x_\perp) = \{\perp\}$ and we post a soft *allDiff*$(X \cup \{x_\perp\}, cost)$ constraint. Note that $x_\perp$ is always assigned to $\perp$: It ensures that all other variables are assigned to values different from $\perp$ whenever this is possible (*e.g.*, when G and G' are isomorphic).

Let us now formally define the two soft CSPs modeling MCIS and MCPS. For the MCIS, we define the soft CSP:

- Variables: $X = \{x_u \mid u \in N_G\} \cup \{x_\perp\}$
- Domains: $D(x_\perp) = \{\perp\}$ and $\forall u \in N_G, D(x_u) = N_{G'} \cup \{\perp\}$
- Hard constraints: $\forall \{u, v\} \subseteq N_G, C_{arc}(x_u, x_v)$
- Soft constraint: *allDiff* $(X, cost)$

For the MCPS, we define the soft CSP :

- Variables: $X = \{x_{uv} \mid (u, v) \in A_G\} \cup \{x_\perp\}$
- Domains: $D(x_\perp) = \{\perp\}$ and $\forall (u, v) \in A_G, D(x_{uv}) = A_{G'} \cup \{\perp\}$
- Hard constraints: $\forall \{(u, v), (w, y)\} \subseteq A_G, C_{arc}(x_{uv}, x_{wy})$
- Soft constraint: *allDiff* $(X, cost)$

Computing Maximum Common Subgraphs from soft CSP solutions. A solution is an assignment σ of the variables of X which satisfies all hard constraints, and which minimizes the violation cost of the soft constraint so that $\sigma(cost)$ is equal to this violation cost. Let $\sigma(X) \setminus \perp$ be the set of values different from $\perp$ which are assigned to variables of X. One can easily check that, for MCIS (resp. MCPS), $G'_{\downarrow \sigma(X) \setminus \perp}$ is isomorphic to an induced (resp. partial) subgraph of G.

Note that we cannot define the common induced subgraph by simply keeping every node of G whose associated variable is assigned to a value different from $\perp$. Indeed, when several nodes of G have the same neighborhood, it may happen that the variables associated with these nodes are assigned to a same value (different from $\perp$). Let us consider for example the graphs of Fig. 1. For MCIS, the assignment $\sigma = \{x_\perp = \perp, x_a = \perp, x_b = 3, x_d = 4, x_c = 2, x_f = 5, x_e = 4\}$ is an optimal solution. In this case, $\sigma(X) \setminus \perp = \{2, 3, 4, 5\}$ and $G'_{\downarrow \sigma(X) \setminus \perp}$ is the subgraph obtained by removing node 1 from G'. In this solution, both x_d and x_e are assigned to 4 because d and e have the same neighborhood.

The size of the common subgraph $G'_{\downarrow \sigma(X) \setminus \perp}$ is equal to $c - \sigma(cost)$ where c is the number of components of G ($c = |N_G|$ for MCIS and $c = |A_G|$ for MCPS), and $\sigma(cost)$ is the value of the cost variable of the soft *allDiff* constraint. As the value of *cost* is minimal, the size of $G'_{\downarrow \sigma(X) \setminus \perp}$ is maximal. On our previous example, we have $\sigma(cost) = 2$ and $|N| = 6$ so that $G'_{\downarrow \sigma(X) \setminus \perp}$ has 4 nodes.

Extension to Labeled Graphs. In labeled graphs, nodes and edges are associated with labels. In this case, the common subgraph must match components the labels of which are equal. This kind of constraints is handled in a straightforward way. For MCIS, we restrict the domain of every variable x_u to nodes which have the same label as u, and we ensure that arc labels are preserved in C_{arc} constraints. For MCPS, we restrict the domain of every variable x_{uv} to arcs which have the same label as (u, v) and whose endpoints have the same labels.

4 Constraint propagation

The two soft CSP models introduced in the previous section are very similar: they both combine a set of binary hard constraints with a soft *allDiff* constraint. We consider different levels of propagation of these constraints.

Propagation of the soft allDiff *constraint.* We consider 3 levels of propagation. The strongest propagation, denoted *GAC(allDiff)*, ensures the generalized arc consistency as proposed in [13]. More precisely, we search for a maximum matching in the bipartite graph $G_b = (X, V, E_b)$ where X is the set of variables, V is the set of values in variable domains, and E_b is the set of edges $(x, v) \in X \times V$ such that $v \in D(x)$. A matching of G_b is a subset of edges of E_b such that no two edges share an endpoint. The cardinality of a largest matching of G_b gives the maximum number of variables that may be assigned to different values. Therefore the number of nodes which are not matched provides an upper bound for *cost*. When this number is larger than the lower bound of *cost*, we cannot filter variable domains. However, as soon as it is as large as the lower bound of *cost*, we can filter domains by searching for every edge (x, v) which does not belong to any maximum matching in G_b. As proposed in [13–15], we use the algorithm of [16] to compute a maximum matching, and we exploit the fact that this algorithm is incremental: at each node, we update the last computed maximum matching by removing edges corresponding to removed values, and we complete this matching until it becomes maximum.

We have considered a weaker filtering, denoted *bound(cost)+FC(diff)*. This filtering does not ensure the generalized arc consistency, but simply checks if there exists a matching of G_b such that the number of non matched nodes is greater than or equal to the lower bound of *cost*. This is done in an incremental and lazy way: at each node, once we have updated the last computed matching, we try to extend it only if its number of non matched nodes is strictly lower than the lower bound of *cost*. We combine this with a simple forward-checking of the binary decomposition of the *allDiff* constraint which simply removes a value v such that $v \neq \perp$ whenever v has been assigned to a variable.

The weakest propagation of the soft *allDiff* constraint, denoted *FC(diff)*, is a forward-checking of its binary decomposition which simply removes a value v such that $v \neq \perp$ whenever v has been assigned to a variable. The upper bound of the *cost* variable is updated each time a variable is assigned to $\perp$.

Propagation of the binary hard constraints C_{arc}. When the domain of the *cost* variable has not been reduced to a singleton by the propagation of the soft *allDiff* constraint, the joker value $\perp$ belongs to the domain of every non assigned variable. In this case, a forward-checking of C_{arc} constraints actually ensures arc consistency. Indeed, for every pair (x_i, x_j) of non assigned variables, and for every value $v \in D(x_i)$, the value $\perp$ belongs to $D(x_j)$ and is a support for v as $C_{arc}(x_i, x_j)$ is satisfied as soon as x_i or x_j is assigned to $\perp$.

When the domain of *cost* is reduced to a singleton, $\perp$ is removed from the domain of all non assigned variable. In this case, maintaining arc consistency

(MAC) may remove more values than a simple forward-checking (FC). Hence, we have considered two different levels of propagation: FC(C_{arc}) performs forward checking, whereas MAC(C_{arc}) maintains arc consistency (however, as FC ensures AC while $\perp$ has not been removed from domains, we still perform FC until $\perp$ is removed, and maintain AC only when $\perp$ has been removed).

5 Experimental results

Compared models. We compare the five following models:

- FC = FC(C_{arc})+FC(diff);
- FC+bound = FC(C_{arc})+bound(cost)+FC(diff);
- FC+GAC = FC(C_{arc})+GAC(allDiff);
- MAC+bound = MAC(C_{arc})+bound(cost)+FC(diff);
- MAC+GAC = MAC(C_{arc})+GAC(allDiff).

The FC model basically corresponds to the Branch & Bound approach proposed by McGregor in [8], and to the CP model proposed in [10] (except that, in [10], a connectivity constraint is added in order to search for connected subgraphs). All models have been implemented in C. We have considered the *minDom* variable ordering heuristic, and values are assigned by increasing order of value.

Test Suites. We consider a synthetically generated database described in [9]. For each graph, there are 3 different labelings such that the number of different labels is equal to 33%, 50% or 75% of the number of nodes.

We report results obtained on 3 test suites of increasing hardness. Test suite 1 considers MCIS on directed and labeled graphs such that the number of labels is equal to 33% of the number of nodes (when increasing this ratio, the problem becomes easier). Test suite 2 considers MCIS on non directed and non labeled graphs. Test suite 3 considers MCPS on directed and non labeled graphs. We have adapted the size of the graphs with respect to the difficulty of these test suites so that they may be solved within a reasonable CPU time limit: in test suite 1 (resp. 2 and 3), we consider graphs with 40 (resp. 30 and 20) nodes. For each test suite, we report results obtained on different classes of graphs: randomly connected graphs with connectivity $\eta \in \{0.05, 0.2\}$ (r005 and r02); 2D, 3D, and 4D regular meshes (m2D, m3D, m4D); 2D, 3D, and 4D irregular meshes with $\rho = 0.6$ (m2Dr, m3Dr, and m4Dr); regular bounded valence graphs with $V \in \{3, 9\}$ (b03 and b09) and irregular bounded valence graphs with $V \in \{3, 9\}$ (b03m and b09m). Each class contains 150 pairs of graphs corresponding to the first 30 instances for each of the 5 possible sizes of the MCIS (*i.e.*, 10%, 30%, 50%, 70% and 90% of the number of nodes of the original graphs).

Discussion. Table 1 compares the 5 CP models on the 3 test suites. It shows us that the FC model (which basically corresponds to approaches proposed in [8] and [10]) is clearly outperformed by all other models, which perform stronger filterings. Indeed, FC achieves a kind of passive bounding on the *cost* variable, by

		FC			FC+bound			FC+GAC			MAC+bound			MAC+GAC		
		%S	time	#Kn	%S	time	#Kn	%S	time	#Kn	%S	time	#Kn	%S	time	#Kn
Test Suite 1	b03	100	105.79	56074	100	20.81	6066	100	24.83	4831	100	**13.43**	3166	100	13.86	1502
	b03m	100	143.74	80297	100	26.17	7754	100	28.86	5651	100	16.91	4021	100	**15.06**	1742
	b09	100	0.11	50	100	0.07	19	100	0.08	17	100	**0.06**	15	100	0.08	10
	b09m	100	0.12	54	100	0.08	22	100	0.09	20	100	**0.07**	17	100	0.10	11
	m2D	100	98.62	53960	100	18.05	5200	100	20.19	3664	100	12.25	2876	100	**10.94**	1220
	m2Dr	100	8.06	3990	100	3.14	864	100	3.65	724	100	**2.21**	523	100	2.38	291
	m3D	100	15.05	7532	100	5.55	1536	100	5.82	1157	100	**3.69**	865	100	4.86	439
	m3Dr	100	3.90	1913	100	1.62	419	100	1.82	359	100	1.14	273	100	**1.13**	154
	m4D	100	97.33	50940	100	12.09	3147	100	12.94	2496	100	**8.17**	1832	100	9.28	835
	m4Dr	100	5.85	2745	100	1.87	471	100	2.04	387	100	**1.38**	325	100	1.50	169
	r005	100	19.47	10540	100	4.72	1295	100	5.68	1040	100	**3.17**	741	100	3.57	393
	r02	100	0.02	10	100	0.02	6	100	0.02	5	100	**0.01**	4	100	0.02	3
Test Suite 2	b03	72	756.25	312080	100	**68.87**	10256	100	77.93	7728	97	212.41	3679	98	231.77	2301
	b03m	57	1081.52	441952	100	**101.77**	14749	100	121.99	12043	97	343.77	6017	97	397.14	4010
	b09	100	147.80	62050	100	**35.09**	7709	100	40.27	6699	100	41.49	6068	100	44.69	3531
	b09m	99	342.89	149613	100	**86.07**	20054	100	94.98	16364	100	101.07	15347	100	103.71	8439
	m2D	61	985.35	394241	100	**103.17**	16003	100	131.48	13532	96	365.66	7582	95	411.89	4491
	m2Dr	62	998.30	383680	100	**171.70**	29757	100	201.49	24344	99	428.35	17228	98	482.21	9128
	m3D	76	737.81	277429	100	**81.78**	13206	100	101.71	11570	100	240.58	7538	98	284.13	4331
	m3Dr	83	681.18	266386	100	**115.49**	21872	100	156.56	20579	100	254.37	13715	100	316.93	8262
	m4D	46	1276.08	498405	100	**120.71**	17386	100	129.53	13257	100	360.14	8699	96	423.83	5704
	m4Dr	50	1448.08	549375	100	**165.51**	27849	100	195.86	23127	100	421.02	16238	100	467.62	8966
	r005	50	1236.75	515935	**99**	**142.04**	22122	98	175.62	17625	93	443.26	8601	92	494.76	5099
	r02	100	474.12	222831	100	246.16	67044	100	283.70	58036	100	**238.91**	53792	100	242.84	32445
Test Suite 3	b03	100	34.44	9364	100	8.67	1178	100	10.93	1163	100	**7.13**	582	100	8.37	392
	b03m	100	47.41	13809	100	9.62	1350	100	10.41	1172	100	7.33	700	100	**7.04**	386
	b09	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
	b09m	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
	m2D	100	214.00	59029	100	32.17	3933	100	33.69	3324	100	26.16	2159	100	**25.83**	1165
	m2Dr	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
	m3D	90	1122.39	263519	100	206.22	23399	100	282.27	24543	100	**187.90**	14170	100	226.57	9000
	m3Dr	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-
	r005	100	11.28	4333	100	2.12	354	100	2.56	358	100	**1.39**	144	100	1.56	98
	r02	0	-	-	0	-	-	0	-	-	0	-	-	0	-	-

Table 1. Comparison of the 5 CP models on the 3 test suites. Each line successively displays the name of the class and, for each model, the percentage of solved instances within a CPU time limit of 30mn (%S), the CPU time (time) in seconds on a 2.26 GHz Intel Xeon E5520 and the number of thousands of nodes (#Kn) in the search tree. CPU time and number of nodes are average results (if an instance is not solved within 30mn, we consider in the average results the CPU time and the number of nodes reached when the search was stopped).

simply counting the number of variables that must be assigned to $\perp$. All other models achieve an active bounding by checking that the number of variables that can be assigned to different values is large enough. The lazy bounding introduced in Section 4 drastically reduces the search space and CPU times of FC+bound are always significantly lower than those of FC. GAC(allDiff) reduces even more the search space but the difference is not so obvious so that CPU times of FC+GAC are always greater than those of FC+bound. This tendency is observed on the 3 test suites.

Replacing FC(C_{arc}) with MAC(C_{arc}) also significantly reduces the number of explored nodes but this stronger filtering has a higher time complexity so that it does not always reduce CPU times: it improves performances on Test suites 1 and 3 but deteriorates them on Test Suite 2. Hence, the best performing approaches are MAC+bound and MAC+GAC on Test suites 1 and 3 whereas the best performing approach is FC+bound on Test suite 2. These results may be explained by the fact that constraints of instances of Test suites 1 and 3 (such that graphs are directed or labeled) are tighter than those of Test suite 2.

Further works. Further work will concern the integration of symmetry breaking techniques and more advanced propagation techniques such as those proposed in [17–19] for graph and subgraph isomorphism. We shall also study the integration of ordering heuristics. Indeed, when solving an optimization problem, ordering heuristics aim at guiding the search towards the best assignments, thus allowing the bounding functions to prune more branches.

Acknowledgement. This work was done in the context of project Sattic (Anr grant Blanc07-1 184534).

References

1. H. Bunke and K. Sharer. A graph distance metric based on the maximal common subgraph. *Pattern Recognition Letters*, 19(3):255–259, 1998.
2. M. R. Garey and D. S. Johnson. Computer and intractability. In *Freeman*, 1979.
3. J.-C. Régin. *Développement d'Outils Algorithmiques pour l'Intelligence Artificielle. Application à la Chimie Organique*. PhD thesis, Université Montpellier II, 1995.
4. J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of computeraided molecular design*, 16(7):521–533, 2002.
5. E. Balas and C. S. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 15(4):1054–1068, 1986.
6. P. J. Durand, R. Pasari, J. W. Baker, and C. Tsai. An efficient algorithm for similarity analysis of molecules. *Internet Journal of Chemistry*, 2, 1999.
7. J. W. Raymond, E. J. Gardiner, and P. Willett. Calculation of graph similarity using maximum common edge subgraphs. *The Computer Journal*, 45(6), 2002.
8. J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12(1):23–34, 1982.
9. D. Conte, P. Foggia, and M. Vento. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *Graph Algorithms and Applications*, 11(1):99–143, 2007.
10. P. Vismara and B. Valery. Finding maximum common connected subgraphs using clique detection or constraint satisfaction algorithms. *Communications in Computer and Information Science*, 14(1):358–368, 2008.
11. N. Beldiceanu, M. Carlsson, S. Demassey, and Thierry Petit. Global constraint catalog: Past, present and future. *Constraints*, 12(1):21–62, 2007.
12. F. Focacci, A. Lodi, and M. Milano. Cost-based domain filtering. In *CP'99*, 1999.
13. T. Petit, J.-C. Régin, and C. Bessière. Specific filtering algorithms for over-constrained problems. In *CP'01, LNCS 2239*, pages 451–464. Springer, 2001.
14. J.-C. Régin. A filtering algorithm for constraints of difference in csps. In *AAAI-94*, pages 362–367, 1994.
15. I. Gent, I. Miguel, and P. Nightingale. Generalised arc consistency for the alldiff constraint: An empirical survey. *Artificial Intelligence*, 172(18):1973–2000, 2008.
16. J. E. Hopcroft and R. M. Karp. An n5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.
17. S. Sorlin and C. Solnon. A parametric filtering algorithm for the graph isomorphism problem. *Constraints*, 13(4):518–537, 2008.
18. S. Zampelli, Y. Deville, and C. Solnon. Solving subgraph isomorphism problems with constraint programming. *Constraints*, 15(3):327–353, 2010.
19. C. Solnon. Alldifferent-based filtering for subgraph isomorphism. *Artificial Intelligence*, 174(12-13):850–864, 2010.