Approximate Constrained Subgraph Matching [*]

Stéphane Zampelli, Yves Deville, and Pierre Dupont

Université Catholique de Louvain,
Department of Computing Science and Engineering,
2, Place Sainte-Barbe
1348 Louvain-la-Neuve (Belgium)
{sz, yde, pdupont}@info.ucl.ac.be

1 Introduction

Our goal is to build a declarative framework for approximate graph matching where various constraints can be stated upon the pattern graph, enabling approximate constrained subgraph matching, extending models and constraints proposed by Rudolf [1] and Valiente et al. [2]. In the present work, we propose a CSP approach for approximate subgraph matching where the potential approximation is declaratively stated in the pattern graph as mandatory/optional nodes/edges. Forbidden edges, that is edges that may not be included in the matching, can be declared on the pattern graph. We also want to declare properties between pairs of nodes in the pattern graph, such as distance properties, that can be either stated by the user, or automatically inferred by the system. In the former case, such properties can define new approximate patterns. In the latter case, these redundant constraints enhance the pruning.

2 Approximate Subgraph Matching

2.1 Problem Definition

A **subgraph monomorphism** between a pattern graph $G_p = (N_p, E_p)$ and a target graph $G_t = (N_t, E_t)$ is an injective function $f : N_p \rightarrow N_t$ respecting $(u, v) \in E_p \Rightarrow (f(u), f(v)) \in E_t$. A constraint model to solve the exact subgraph matching problem has been proposed by several authors [2] [1]. This model focuses on monomorphism and will form our basic monomorphism constraints. The variables $\mathcal{X} = \{x_1, ..., x_n\}$ are the nodes of the pattern graph and their respective domain $D(x_i)$ is the set of target nodes. The assignment must respect two conditions: all variables have a different value and the structure of the pattern must be kept (monomorphism condition). In a CSP framework, the first condition is implemented with the classical *Alldiff*$(x_1, ..., x_n)$ constraint [3] [4]. The second condition is translated into a monomorphism constraint.

A useful extension of subgraph matching is approximate subgraph matching, where the pattern graph and the found subgraph in the target graph may differ with respect to their structure.

Optional nodes In our framework, the approximation is *declared* upon the pattern graph. Some *nodes* are declared *optional*, i.e. nodes that may not be in the matching. Specifying optional edges in a monomorphism problem is useless as it is equivalent to omitting the edge in the pattern. The status of the edges depends on the optional state of their endpoints. An edge having an optional node as one of its endpoints is optional. An

[*] Acknowledgments : This research is supported by the Walloon Region, project BioMaze (WIST 315432). Thanks also to the EC/FP6 Evergrow project for their computing support.

optional edge is not considered in the matching if one of its endpoints is not part of the matching. Otherwise, the edge must also be a part of the matching.

Forbidden edges *Edges* may also be declared as *forbidden* between their two endpoints (u, v), meaning that if u and v are in the domain of f, then (u, v) must not exist in the target graph. A pattern graph with all its complementary edges declared as forbidden induces a subgraph isomorphism instead of a subgraph monomorphism.

A pattern graph with optional nodes and forbidden edges forms an *approximate pattern graph*.

Definition 1 *An* ***approximate pattern graph*** *is a tuple (N_p, O_p, E_p, F_p) where (N_p, E_p) is a graph, $O_p \subseteq N_P$ is the set of optional nodes and $F_p \subseteq N_p \times N_p$ is the set of forbidden edges, with $E_p \cap F_p = \emptyset$.*

The corresponding matching is called an *approximate subgraph matching*.

Definition 2 *An* ***approximate subgraph matching*** *between an approximate pattern graph $G_p = (N_p, O_p, E_p, F_p)$ and a target graph $G_t = (N_t, E_t)$ is a* partial *function $f : N_p \rightarrow N_t$ such that :*

1. $N_p \setminus O_p \subseteq dom(f)$
2. $\forall\ i, j \in dom(f) : i \neq j \Rightarrow f(i) \neq f(j)$
3. $\forall\ i, j \in dom(f) : (i, j) \in E_p \Rightarrow (f(i), f(j)) \in E_t$
4. $\forall\ i, j \in dom(f) : (i, j) \in F_p \Rightarrow (f(i), f(j)) \notin E_t$

The notation $dom(f)$ represents the domain of f. Elements of $dom(f)$ are called the selected nodes of the matching. This means that $dom(f)$ can be represented by a finite set variable. Its lower bound f_{lb} consists of all selected nodes, and its upper bound f_{glb} consists of selected nodes and nodes that could be selected.

3 Constraints for Approximate Subgraph Matching

Alldiff Constraint The *Alldiff* constraint must be adapted to variables that may not be assigned. One solution is to create symbolic values $e_1, ..., e_n$ and put e_i in the initial domain of x_i. In a solution, $x_i = e_i$ if x_i is not assigned to any target node. Using these n symbolic values, a global *Alldiff* constraint can still be posted as in the exact case.

Morphism Constraint The basic morphism condition forcing the matching to respect the pattern structure (see [2]) has been generalized to handle more general morphism conditions as well as optional nodes :

$$MC(x_1, ..., x_n, A, B) \equiv \bigwedge_{i,j} (i, j) \in A \wedge i, j \in dom(f) \Rightarrow (x_i, x_j) \in B$$

The former constraint states that a morphism relation between two pattern nodes x_i and x_j must be forced if and only if they are present in the domain of f. The MC constraint can be rewritten as:

$$\begin{aligned} &\forall\ i \in N_p\ \forall\ a \in N_t : (\ |D(x_i) \cap V_t(a)| = 0 \\ &\qquad\qquad\qquad \wedge\ i \in dom(f)\ ) \Rightarrow a \notin D(x_j) \quad \forall\ j \in V_p(i)\ . \end{aligned}$$

The proposed propagator keeps track of relations between all the target nodes and the domain $D(x_i)$ in a structure $S(i, a) = |D(x_i) \cap V_t(a)|$ representing the number of relations between a target node a and $D(x_i)$. Whenever the neighbors of a target node a

have no relation with $D(x_i)$, that is when $S(i,a) = 0$, node a is pruned from all neighbors of x_i. It has a $O(NDd)$ amortized time complexity, and the structure $S(i,a)$ has O(ND) spatial complexity [2]. The preprocessing to compute $S(i,a)$ costs $O(NDd)$. The global MC constraint is thus *algorithmically* global as it achieves the same consistency than the original conjunction of constraints, but more efficiently.

The additional condition $i \in dom(f)$ states that only selected nodes should propagate under the morphism condition. The propagation of the morphism constraint of an optional i is computed but performed only when i is in the domain of f. As depicted in Figure 1, all selected nodes propagate in their neighborhood but optional nodes propagate only when they are selected.

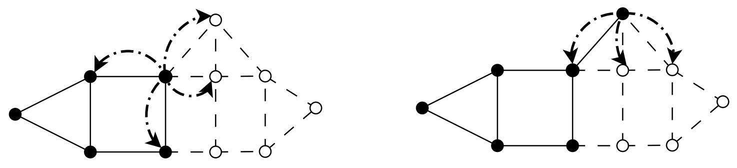
Fig. 1. Pruning method for the approximate morphism condition

Forbidden Edges Constraint A constraint for the forbidden edges (condition 4 in the matching) can be obtained by parameterizing MC with $\overline{E}_t = \{(a,b) \notin E_t\}$:

$$MC(x_1, ..., x_n, F_p, \overline{E}_t) \ .$$

The constraints for the approximate subgraph isomorphism problem are then :

$$\text{alldiff}(x_1, ..., x_n) \ \wedge \ MC(x_1, ..., x_n, E_p, E_t) \ \wedge \ MC(x_1, ..., x_n, F_p, \overline{E}_t) \ .$$

However a single propagator $MCPA$ handling the last two constraints was implemented, using the relation $|D(x_i) \cap \overline{V}_t(a)| = 0 \Leftrightarrow |D(x_i) \cap V_t(a)| = |D(x_i)|$.

Local Alldiff Constraint A local alldiff redundant LA^+ constraint checks if there are enough candidate target nodes for x_i neighborhood if x_i is assigned to a target node a. If it not the case, a can be pruned from $D(x_i)$. Note that the x_i neighborhood is restricted to the variables representing selected nodes, noted as $V_p^+(i)$. The LA constraint is expressed as:

$$LA^+(x_i, A, B) \equiv |\cup_{j \in V_p^+(i,A)} D(x_j) \cap V_t(x_i, B)| \geq |V_p^+(i, A)|$$
$$LA^+(x_1, ..., x_n, A, B) \equiv \bigwedge_i LA^+(x_i, A, B) \ .$$

Constraint LA^+ plays a pruning role. It can be implemented by maintaining the neighborhood variable, with an $O(d)$ time complexity, whenever the domain of x_i is pruned. The structure $R^+(i,a) = |\{ \ j \in V_p^+(i) \mid a \in D(x_j) \ \}|$ depends not only on the domain of the neighborhood of x_i but also on the neighborhood variable. Whenever the lower bound of $V_p^+(i)$ changes, the structure $R^+(i,\cdot)$ must be updated in $O(D)$, resulting in a $O(ND^2)$ amortized complexity. Moreover, $R^+(i,a)$ may be incremented from zero to one, resulting in an increment of $CT^+(i,a) = |\cup_{j \in V_p^+(i)} D(x_j) \cap V_t(a)|$, which is not monotone. Nevertheless, when condition $CT^+(i,a) < |V_p^+(i)|$ is fulfilled, a can be safely pruned from x_i, because if there is not enough candidates for a subgroup of the

Table 1. Comparison over GraphBase directed graphs.

One solution 5 min.					All solutions 5 min.				
	solved	unsol	total time	mean time		solved	unsol	total time	mean time
vflib C++	80,5%	19,5%	8.89 min.	0.02 min.	vflib C++	63,7%	36,3%	12.01 min.	0.02 min.
ozvflib	78,5%	21,5%	17.67 min.	0.04 min.	ozvflib	59,8%	40,2%	11.52 min.	0.02 min.
CSP	87%	13%	36.64 min.	0.09 min.	CSP	68,7%	31,3%	31.4 min.	0.07 min.

neighborhood, node i cannot be mapped to node a, even if the condition still holds for the group.

Distance Constraints Thanks to the parameters A and B, former MC and LA constraints can be used to create redudant constraints such as shortest path constraint MC_{dist}, generalizing other works on shortest path distance [5]. If $dist(a, b)$ denotes the shortest-path distance between node a and b, then the MC_{dist} constraint can be formulated as:

$$MC_{dist}(x_1, ..., x_n, k) \equiv \bigwedge_{i,j} dist(i, j) = k \Rightarrow dist(x_i, x_j) \leq k.$$

Suppose $E_p^k = \{(i, j) \mid dist(i, j) = k\}$ and $E_t^k = \{(a, b) \mid dist(a, b) \leq k\}$. Then MC_{dist} is equivalent to :

$$MC_{dist}(x_1, ..., x_n, k) \equiv MC(x_1, .., x_n, E_p^k, E_t^k).$$

4 Experiments

Our CSP model for approximate subgraph matching has been implemented inside the CSP framework of Oz/Mozart (`www.mozart-oz.org`). Parametric propagators were implemented.Various transformations of E_p and E_t were automated to instantiate propagators for the forbidden edges and the distance constraints. We also included facility constraints to declare distance constraints between specific pattern nodes.

First part of the experimental tests aims at comparing the CSP approach with a dedicated algorithm for subgraph matching. The selected algorithm is an improvement of Ullmann's algorithm [6] called vflib, described into [7]. The C++ implementation provided by the authors is used. We have also reimplemented the vflib algorithm in Oz/Mozart.

Two distinct sets of graphs were selected. The first set comes from [8]. The graphs are characterized by their probability η ($eta = 0.01$ is noted r001 in Table 1) that an edge is present between two distinct node n and n'. Those graphs were used to evaluate vflib algorithm performance [7]. In our experiments, pattern graph size is 20% of the target graph size, target graph size ranges from 20 to 200, and all solutions are searched. The second set contains graphs having different topological structures as explained in [2]. These graphs were generated using the Stanford GraphBase [9] and are all graphs tested in [2], consisting of 406 directed instances and 1225 undirected instances.

Experiments show that CSP approach for subgraph matching solves more problem within a time limit against C++ specialized checking-based methods [7]. Table 1 and 2 show the percentage of instances solved within a time limit of 5 minutes, for directed and undirected instances. Single specialized propagator $MCPA$ for forbidden edges is more efficient than the version with two propagators. Table 3 supports this assertion. Preliminary results show that matching with 40% of optional nodes and few (≤ 5) additional distance constraints is tractable.

Table 2. Comparison over GraphBase undirected graphs.

One solution 5 min.					All solutions 5 min.				
	solved	unsol	total time	mean time		solved	unsol	total time	mean time
vflib C++	64,4%	35,6%	8.14 min.	0.006 min.	vflib C++	48,3%	51,7%	9.31 min.	0.007 min.
ozvflib	58,2%	41,8%	8.6 min.	0.04 min.	ozvflib	39,5%	60,5%	4.43 min.	0.003 min.
CSP	64,4%	35,6%	18.24 min.	0.01 min.	CSP	57,7%	42,3%	11.39 min.	0.009 min.

Table 3. MC and MC versus $MCFA$

CSP MC_p and MC_{fa} 5 min.							CSP $MCFA$ 5 min.						
	r001		r005		r01			r001		r005		r01	
	solved	unsol	solved	unsol	solved	unsol		solved	unsol	solved	unsol	solved	unsol
100	100	0	100	0	96	4	100	100	0	100	0	99	1
200	79	21	62	38	10	90	200	83	17	80	20	34	66

5 Perspectives

The proposed framework for declarative approximate subgraph matching open various research directions. Better heuristics could be developed when searching for an approximate matching. Our algorithm for exact matching could also be compared with other algorithms dedicated to the largest common subgraph problem. We also intend to apply our approximate matching algorithm for the analysis of biochemical networks. Extensive expirements should highlight the benefits of distance constraints. Finally, as the (approximate) matching is expressed as a combination of (parameterized) constraints, subgraph matching could be integrated in a constraint language handling graph variables, such as CP(Graph) [10] [11].

References

1. Rudolf, M.: Utilizing constraint satisfaction techniques for effi cient graph pattern matching. In Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G., eds.: TAGT. Volume 1764 of Lecture Notes in Computer Science., Springer (1998) 238–251
2. Larrosa, J., Valiente, G.: Constraint satisfaction algorithms for graph pattern matching. Mathematical. Structures in Comp. Sci. **12**(4) (2002) 403–422
3. Regin, J.C.: A fi ltering algorithm for constraints of difference in CSPs. In: Proc. 12th Conf. American Assoc. Artifi cial Intelligence. Volume 1., Amer. Assoc. Artifi cial Intelligence (1994) 362–367
4. van Hoeve, W.J.: The alldifferent constraint: A survey. CoRR **cs.PL/0105015** (2001)
5. Sorlin, S., Solnon, C.: A global constraint for graph isomorphism problems. In Régin, J.C., Rueher, M., eds.: CPAIOR. Volume 3011 of Lecture Notes in Computer Science., Springer (2004) 287–302
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.: Performance evaluation of the vf graph matching algorithm. In: ICIAP, IEEE Computer Society (1999) 1172–1177
8. Foggia, P., Sansone, C., Vento, M.: A database of graphs for isomorphism and sub-graph isomorphism benchmarcking. CoRR **cs.PL/0105015** (2001)
9. Knuth, D.E.: The Stanford GraphBase. A Platform for Combinatorial Computing. acm, ny (1993)
10. Dooms, G.: Cp(graph): Introducing a graph computation domain in constraint programming (accepted paper). CP2005 (2005)
11. Deville, Y., Dooms, G., Zampelli, S., Dupont, P.: Cp(graph+map) for approximate graph matching. 1st International Workshop on Constraint Programming Beyond Finite Integer Domains, CP2005 (submitted paper) (2005)