# Graph-Based Relational Learning with a Polynomial Time Projection Algorithm

Brahim Douar, Michel Liquière, Chiraz Latiri, Yahya Slimani

# Graph-based relational learning with a polynomial time projection algorithm

Brahim Douar[1,2] and Michel Liquiere[1]

[1] LIRMM, 161 rue Ada 34392 - Montpellier, France
{douar,liquiere}@lirmm.fr
[2] URPAH Team, Faculty of Sciences of Tunis

**Abstract.** The paper presents a new projection operator for graphs, named AC- projection, which exhibits good complexity properties as opposed to the graph isomorphism ($\Theta$-subsumption) operator typically used in graph mining. We study the size of the search space and some practical properties of the projection operator. These properties give us a specialization algorithm using simple local operations. Then we prove experimentally that we can achieve an important performance gain (polynomial complexity projection) without or with non-significant loss of discovered patterns quality.

**Keywords:** Relational learning, Polynomial-complexity projection, Specialization algorithm

## 1 Introduction

One goal of machine learning is the search of patterns to regroup or separate some elements (examples or counter examples). For this goal, logic-based systems have dominated the area of relational concept learning, especially Inductive Logic Programming (ILP) systems. However, a part of first-order logic can naturally be represented as a graph [6].
In order to learn from a relational description, we need a partial order on expressions of the description language (projection operator which gives a partial order between two expressions). To deal with the complexity of such description, some authors limit the description language [1]. In [2], the author uses a different bias. The examples are described by graphs but the projection operator is not an homomorphism ($\Theta$-subsumption [4]) or a subgraph isomorphism (OI-subsumption [3]). It is a new matching based on arc consistency named AC-projection.
In this paper we present a novel graph mining algorithm, named AC-miner and based on the AC-projection operator, followed by some experimental evaluation of it on classical graph mining data sets.

## 2 The AC-projection Operator

**Definition 1.** *(Labeled Graph) A labeled graph can be represented by a 4-tuple,* $G = (V, E, L, l)$*, where*

- *$V$ is a set of vertices,*
- *$E \subseteq V \times V$ is a set of edges,*
- *$L$ is a set of labels,*
- *$l : V \cup E \to L$, $l$ is a function assigning labels to the vertices and the edges.*

**Definition 2.** *(Labeling) Let $G_1$ and $G_2$ be two graphs. We named labeling from $G_1$ into $G_2$ a mapping $\mathcal{I} : V(G_1) \to 2^{V(G_2)} | \forall x \in V(G_1),\ \forall y \in \mathcal{I}(x), l(x) = l(y)$.*

**Definition 3.** *(AC-compatible $\curvearrowright$ ) Let $G$ be a graph $V_1 \subseteq V(G), V_2 \subseteq V(G)$ $V_1$ is AC-compatible with $V_2$ iff*

1. *$\forall x_k \in V_1 \exists y_p \in V_2 | (x_k, y_p) \in E(G)$*
2. *$\forall y_q \in V_2 \exists x_m \in V_1 | (x_m, y_q) \in E(G)$.*

*We note $V_1 \curvearrowright V_2$*

**Definition 4.** *(Consistency for one arc) Let $G_1$ and $G_2$ be two graphs. We say that a labeling $\mathcal{I} : V(G_1) \to V(G_2)$ is consistent with an arc $(x, y) \in E(G_1)$, iff $\mathcal{I}(x) \curvearrowright \mathcal{I}(y)$.*

**Definition 5.** *(AC-labeling) Let $G_1$ and $G_2$ be two graphs. A labeling $\mathcal{I}$ from $G_1$ into $G_2$ is an AC-labeling iff $\mathcal{I}$ is consistent with all the arcs $e \in E(G_1)$.*

**Definition 6.** *(AC-projection $\rightharpoondown$ ) Let $G_1$ and $G_2$ be two graphs. An AC-labeling $\mathcal{I} : V(G_1) \to V(G_2)$ is an AC-projection iff $\forall$ AC-labeling $\mathcal{I}' : V(G_1) \to V(G_2)$ and $\forall x \in V(G_1)$, $\mathcal{I}'(x) \subseteq \mathcal{I}(x)$. We note it $G_1 \rightharpoondown G_2$*

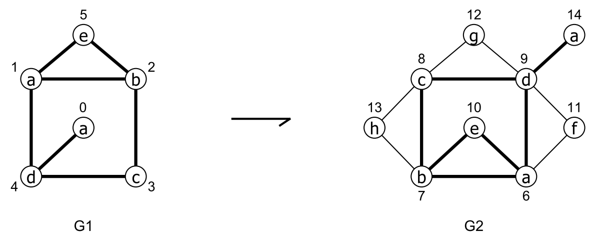


**Fig. 1.** An AC-projection example ($G_1 \rightharpoondown G_2$)

**Definition 7.** *(AC-equivalent graphs $\leftrightharpoons$)*
*Two graphs $G_1$ and $G_2$ are AC-equivalent iff both $G_1 \rightharpoondown G_2$ and $G_2 \rightharpoondown G_1$ are fulfilled. We note it $G_1 \leftrightharpoons G_2$.*

We have an equivalence relation between graphs using the AC-projection. The smallest element in this equivalence class will be its unique representative, and for which we give then the name of “AC-reduced graph”.

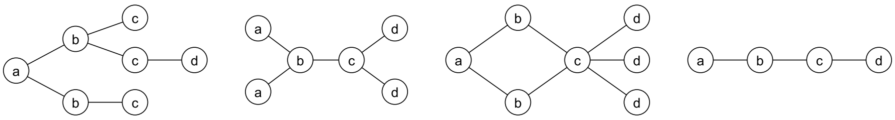

**Fig. 2.** AC-equivalent graphs and the associated AC-reduced one (extreme right)

## 3 Search space with AC-projection

In this section, we study the size of the search space using AC-projection. We present some properties of the AC-projection, using this properties we can find an upper bound of the search space. We present this result for one labeled graph G, but these results can be easily extended for n graphs (one for each example). In this case G is the disjoint union of the graphs describing the examples.
Notation: For a labeled graph G:(V,E,L) we note $\mathcal{P}_l$(V), the power set of vertices, in V, with label l ∈ L.

**Definition 8.** *(AC-graph ) For a labeled graph G:(V,E,L) and a set P of element* $\in \bigcup \mathcal{P}_l(V)$ *with* $l \in L$.
*We construct a graph G':(V',E',L') with:*
*- a vertex v for each element in P. We note p(v) ∈ P the associated element.*
*- The label of a vertex v in the label of the element in p(v)*
*-* $(V_1,V_2) \in V'$ *iff* $p(V_1) \curvearrowright p(V_2)$
*G' is an AC-graph of G.*

So an AC-graph is built from a list of set of vertices from a graph G.
Now, we study some links between AC-graph and AC-projection.

**Proposition 1.** *For each AC-projection between two graphs G', G there is an associated AC-graph.*

*Proof.* Since an AC-projection $\mathcal{I}$, gives, for each vertex of x of G', a set of vertex of G. The AC-graph built from an AC-projection is the one build from the set of I(x), x ∈ V'.

**Proposition 2.** *For each AC-graph G' of a graph G we have* $G' \rightharpoondown G$.

*Proof.* The labeling $\mathcal{I}$ with, for each V ∈ G', $\mathcal{I}$(V) = p(V) is an AC-labeling from G' into G by construction.

Now for a graph G we can define a specific AC-graph built from the power set of vertices of G.

**Definition 9.** *(Max-AC-graph ) For a graph G:(V,E,L) the Max-AC-graph of G is the AC-graph built from the set P of all element* $\in \bigcup \mathcal{P}_l(V)$ *with label* $l \in L$. *We note this graph Max-AC-graph(G)*

All subgraphs of Max-AC-graph(G) has an AC-projection into G. Since the Max-AC-graph is the biggest AC-graphe, we have our seach space. The complexity of the construction of the Max-AC-graph is $O(2^n)$ where n is the number of vertices in G. This complexity is big but for many structural descriptions (graph with homorphism projection ..) the size of the search space is bigger by an order of magnitude.

## 4 AC-miner: A graph mining approach with a polynomial time projection

In this section we will present a basic algorithm for frequent AC-reduced subgraphs mining. The goal of this algorithm is the construction of a part of the Max-AC-graph(G) where G is the disjoint union of the graphs describing the examples (G is technically materialized by a graph database $\mathscr{D}$ in the following). We are using a support parameter ($\sigma$) as a bias which limits the search space.

### 4.1 AC-compatible extension

**Definition 10.** *(Vertex group) Given a graph database $\mathscr{D}$, a vertex group $\mathcal{V}$ is a set of vertices of the same label $l$ and belonging to graphs in $\mathscr{D}$. The most general vertex group $\mathcal{V}^l$ is the maximal vertex group of a given label $l$.*

The AC-compatible extension, is the core operation of the AC-miner algorithm. Given a vertex group $\mathcal{V}$ and a vertex label $l$, the AC-compatible extension consists in finding the maximal subset $\mathcal{V}$ that is AC-compatible with a maximal subset of the most general vertex group ($\mathcal{V}^l$). The AC-compatible extension is considered to be valid w.r.t. a minimal support parameter ($\sigma$) if and only if the vertices in $\mathcal{V}$ appears at least in $\sigma$ graphs of the graph database $\mathscr{D}$.

### 4.2 The AC-miner algorithm

The AC-miner algorithm (see Algorithm 1) starts by adding for each vertex label in the graph database $\mathscr{D}$ its associated most general vertex group $\mathcal{V}^l$ in the *jobs* list (Algorithm 1 line 1). This list contains the remaining vertex group to extend. Then, based on this list (*jobs*) it starts the main computational loop. During each iteration it will try to make an AC-compatible extension for the current vertex group with each one of the graph database labels (Algorithm 1 line 4). If there is an AC-compatible extension, AC-miner will add (if not already done) the two vertex group children as well as an edge between them to the $\mathscr{G}$ AC-graphe and the *jobs* list (lines 6-12). The algorithm will iterates till the *jobs* list becomes empty. At this stage, the algorithm will extract all the connected components from the $\mathscr{G}$ AC-graph. These subgraphs represent the frequent AC-reduced subgraphs.

**Algorithm 1**: AC-miner

**Input** : Graph database $\mathscr{D}$, Minimal Support $\sigma$, AC-graphe $\mathscr{G}$ (local)
**Output**: $\mathcal{F}$ = frequent AC-reduced subgraphs

$jobs = \{\bigcup \mathcal{V}^l | l \in \mathscr{D}.getLabels()\}$;
**while** $jobs \neq \emptyset$ **do**
  $\mathcal{V} = jobs.getFirst()$;
  **for** $\{\forall l | l \in \mathscr{D}.getLabels(), l \notin \mathcal{V}.getForbidden()\}$ **do**
    **if** `AC-compatible-Extension`$(\mathcal{V}, \mathcal{V}^l, \mathcal{V}_{child}, \mathcal{V}^l_{child}, \sigma)$ **then**
      **if** $\mathcal{V}_{child} \notin \mathscr{G}$ **then**
        $\mathscr{G} = \mathscr{G} \cup \mathcal{V}_{child}$;
        $jobs = jobs \cup \mathcal{V}_{child}$;
      **if** $\mathcal{V}^l_{child} \notin \mathscr{G}$ **then**
        $\mathscr{G} = \mathscr{G} \cup \mathcal{V}^l_{child}$;
        $jobs = jobs \cup \mathcal{V}^l_{child}$;
      $\mathscr{G}.addEdge(\mathcal{V}_{child}, \mathcal{V}^l_{child})$;
**return** $\mathscr{G}.getConnectedComponents()$;

## 5 Experiments And Comparative Study

In order to prove the usefulness of the AC-projection for graph mining, we present in the following a qualitative evaluation of the AC-reduced patterns which consists in a calculation of their discriminative power within a supervised graph classification process.

***Datasets:*** We carried out classification experiments on two real-world datasets group widely cited in the literature : The anti-cancer screen datasets (nci) and the AIDS antiviral screen data (aids) as in [7].

***Methods:*** We evaluated the classification accuracy using two different feature sets : Isomorphic and AC-reduced. Each chemical compound is represented by a binary vector with length equal to the number of mined subgraphs. Each subgraph is mapped to a specific vector index, and if a chemical compound contains a subgraph then the bit at the corresponding index is set to one, otherwise it is set to zero.

***Results:*** All classifications have been done using the well-known C4.5 decision tree classifier [5]. We have reported results of the prediction accuracy over 10 cross-validation trials. According to results shown in Figure 3a and 3b, we see that for all datasets we have very few AC-reduced frequent patterns compared to the isomorphic ones. We have on average 35% less patterns. This ratio is bigger for lower supports and can reach up to 58% for the aids dataset with a minimal support of 10%. In the qualitative point of view (Figure 3c) we see that the percentage of correctly classified (PCC) instances is almost the same for all minimal supports. Taking a more in-depth look to the results, we see that, for some datasets and minimal support values, we even have better PCC for AC-reduced feature set. This is due to the better generalization power of the AC-reduction process, which helped supervised classifiers avoiding over-fitting learning problem.

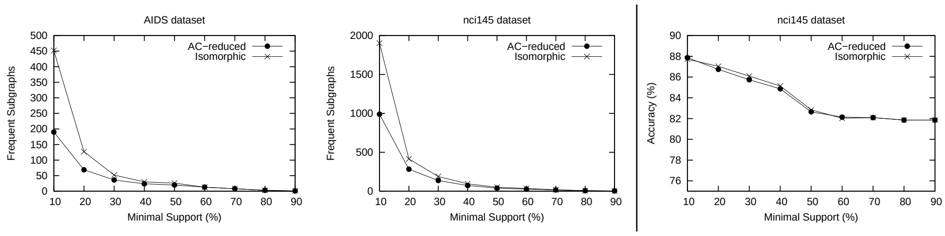


**Fig. 3.** Comparison of the number of frequent patterns (a,b) and classification accuracy (c) for aids and nci145 datasets

## 6 Conlusion

In this paper, we have studied the use of a new polynomial projection operator named AC-Projection initially introduced in [2]. We have then presented a novel algorithm named AC-miner which proceeds by specialization of expressions using very simple and fast set and neighborhood operators. This simplicity allows us to obtain a very fast algorithm which can be easily adapted for a depth first or a breadth first search strategy and can be easily parallelized as well. AC-miner is intended to mine frequent AC-reduced subgraphs from a graph database. We have experimentally showed that the number of these subgraphs is clearly smaller than isomorphic subgraphs but having a very comparable quality and discriminative power.

## References

1. Cook, D.J., Holder, L.B.: Mining Graph Data. John Wiley & Sons (2006)
2. Liquiere, M.: Arc consistency projection: A new generalization relation for graphs. In: ICCS. LNCS, vol. 4604, pp. 333–346. Springer (2007)
3. Malerba, D., Lisi, F.: Discovering associations between spatial objects: An ilp application. In: Inductive Logic Programming, LNCS, vol. 2157. Springer (2001)
4. Plotkin, G.D.: A note on inductive generalization. Machine Intelligence 5 (1970)
5. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
6. Sowa, J.F.: Conceptual graphs summary, pp. 3–51. Ellis Horwood (1992)
7. Thoma, M., Cheng, H., Gretton, A., Han, J., Kriegel, H.P., Smola, A., Song, L., Yu, P.S., Yan, X., Borgwardt, K.M.: Discriminative frequent subgraph mining with optimality guarantees. Statistical Analysis and Data Mining 3(5), 302–318 (2010)