KIDS: an iterative algorithm to organize relational knowledge

Isabelle Bournaud[1], Mélanie Courtine[2] and Jean-Daniel Zucker[2]

[1] LRI, Bat. 490 Université Paris-Sud, Av. du Général de Gaulle,
F-91405 Orsay Cedex, France
Isabelle.Bournaud@lri.fr
[2] LIP6, Université Paris VI, 4, place Jussieu
F-75252 Paris Cedex 05, France
{Melanie.Courtine, Jean-Daniel.Zucker}@lip6.fr

Abstract. The goal of conceptual clustering is to build *a* set of embedded classes, which cluster objects based on their similarities. Knowledge organization aims at generating *the* set of most specific classes: the Generalization Space. It has applications in the field of data mining, knowledge indexation or knowledge acquisition. Efficient algorithms have been proposed for data described in <attribute, value> pairs formalism and for taking into account domain knowledge. Our research focuses on the organization of relational knowledge represented using conceptual graphs. In order to avoid the combinatorial explosion due to the relations in the building of the Generalization Space, we progressively introduce the complexity of the relations. The KIDS algorithm is based upon an iterative data reformulation which allows us to use an efficient propositional knowledge organization algorithm. Experiments show that the KIDS algorithm builds an organization of relational concepts but remains with a complexity that grows linearly with the number of considered objects.

1 Introduction

In Artificial Intelligence, the problem of the automatic construction of classifications has been the subject of much researches during the last fifteen years [6], [8], [13]. It consists in searching for similarities between objects which are not pre-classified and structuring them in a hierarchy of classes in which *similar* objects are clustered. A class is also called a *concept* since it is described by an extension (the set of objects clustered) and by an intension (the similarities of the descriptions of the objects clustered). Most of the existing *Conceptual Clustering* approaches defined this task as the search for *a* classification that would best predict unknown features of new objects [5], [7], [8]. This type of construction is guided by heuristics, which allow one to choose the best classes among the possible ones. The developed methods have proved their interest in various fields [6], [8], [9], [13]. In other words, the classifications built do not contain a class for each subset of objects whose descriptions have similarities. More recent researches concern the construction of classifications that *organize* knowledge [3], [15]. In these tasks, the goal is not to build a subset of the possible classes but all the classes clustering similar objects: the *Generalization Space*. In these methods, the process of construction is not based on a numerical distance among descriptions and on a function to be optimized but on a language to describe the similarities among the object descriptions. This language is called the *generalization language*.

Efficient algorithms have been proposed for organizing data described by a set of pairs <attribute, value> [15] and for taking into account domain knowledge [1], [3]. Our research concerns organization of relational data, i.e. data represented in more expressive formalisms (first-order logic, description logic, conceptual graphs ...). To avoid the problem of combinative explosion due to graph matching, we propose to take *gradually* the complexity of graphs into account through a hierarchy of abstraction spaces. The proposed approach, called KIDS, extends the propositional approach of knowledge organization COING [1] to the relational framework. Given a set of objects described using conceptual graphs [17] and domain knowledge represented in a generalization lattice [14], COING builds the Generalization Space of propositional descriptions of the objects. KIDS gradually enriches this space thanks to a generalization language which is made more and more expressive at each step of the algorithm. This idea, inspired from the REMO system [19], consists in increasing gradually the structure of matching. The KIDS algorithm is based upon an iterative reformulation of the data, which allows us to use COING on the reformulated descriptions of the objects.

In the next section, we present the COING propositional algorithm for knowledge organization. Although COING is based upon relational descriptions of data, it does not use the structure of the descriptions in the construction of the Generalization Space. Section 3 introduces the KIDS approach: we describe our method for graph reformulation by abstraction, present the KIDS algorithm and illustrate our approach on an example. In the next section, we evaluate KIDS on a Chinese characters database. These experiments show the feasibility of the proposed approach. Finally, in section 5, we conclude with a brief summary and outline directions for future research.

2 Organization of relational knowledge

2.1 A graphical representation of relational data and their generalization

In the automatic construction of classifications, choosing the right language for representing the objects is very important; it has an impact on the efficiency of the algorithms manipulating them. The more expressive a language is, the more complex are the algorithms manipulating it. Objects are structured, and this is true in many fields; they may be decomposed into several parts, and these are then linked together thanks to various relations (for example a *part-of* relation). Attribute-value languages do not allow to easily represent such structure. We use a language based on a higher-order logic and represent relational descriptions of objects in the conceptual graphs formalism. However, this representation is not a limitation of our approach, as it may be applied to any relational data described by graphs.

A *conceptual arc* is a triplet: *[concept$_s$]->(relation)->[concept$_d$]*, where *(relation)* corresponds to a relation between *[concept$_s$]* and *[concept$_d$]*. A *conceptual graph* is a graph composed of a set of conceptual arcs. For more information about conceptual graphs, the reader should refer to [19] [4].

Figure 1 below presents an example of a house description using conceptual graphs. The triplet *[Window]-> (color) -> [White]* is a conceptual arc. This example is used throughout the article to illustrate the algorithms presented.

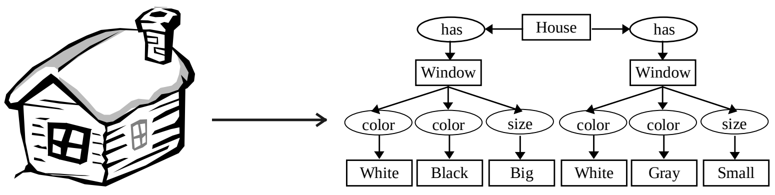

Fig. 1. A house and its description as a conceptual graph.

2.2 Organizing knowledge in a Generalization Space

Given a set of object descriptions and a generalization language, the associated *Generalization Space* (GS) is the set of the most specific conjunctive concepts generalizing these descriptions. In the GS, a node n_i is a pair (c_i, d_i). The element c_i, called the *coverage* of n_i, is the set of objects covered by n_i; and d_i, called the *description* of n_i, corresponds to the common features (most specific generalization) of the objects of c_i. In the GS, a node corresponds to a cluster of objects described in *intension* by its description d_i and in *extension* by its coverage c_i. Nodes of GS are partially ordered by a subsumption relation between concepts. Given a node n_i with coverage c_i, its ancestors are all the nodes n_j, such that $C_j \supset C_i$. This partial order provided the GS with a *pruned lattice structure*[1], which may be represented by an *inheritance network*. Indeed, GS nodes *inherit* the descriptions of the nodes which are more general.

Figures 4 and 9 present two different Generalization Spaces of the same objects (as explained in the next section, part of their node descriptions come from the use of a generalization lattice over the types). Their differences lie in the expressiveness of the generalization language used to build the GS. In effect, given a set of object descriptions, depending on the language chosen to describe the generalizations (the

[1] The Generalization Space may also be defined by the two isomorphic lattices: the Galois lattice of concept descriptions (partially ordered by the subsumption relation) and the lattice of objects (partially ordered by the inclusion relation) [12].

node descriptions), the nodes of the associated GS will not be the same. The node n'3 in the GS of figure 9 for example does not appear in the GS of figure 4. Moreover, for a given set of objects, nodes belonging to different GS but having the same coverage may have a more or less general description. The node n'2 in the GS of figure 9 and the node n2 in figure 4 have the same coverage on objects ({h2, h3}) but the description of the node n'2 is more specific than that of n2.

2.3 A classical simplification of the graph generalization problem

To avoid the exhaustive analysis of each of the 2^n partitions of n objects, COING adopts a bottom up approach generalizing objects descriptions to incrementally build the GS. In COING, objects are represented using conceptual graphs. In order to deal with the problem of matching graphs which is known to be NP-complete, COING transforms the graph representation into an arc representation. In other words, each graph describing an object is transformed into a set of *independent* arcs. This reformulation has the advantage to limit the complexity of the algorithm (in the worst case quadratic with the number of objects [1]) because, as the arcs are oriented they fully match. However, this restricts the generalization language since relations among arcs are not considered.

The COING principle for building the GS is as follows:

1. *Reformulate* each graph describing the objects to be organized as a set of arcs.
2. *Generalize* each arc describing the objects. COING integrates an efficient method for taking into account domain knowledge in the GS construction [1]. This knowledge, represented in a generalization hierarchy (called the "type lattice" in the conceptual graphs formalism [17]) expresses, for example in the domain of colors, that the type `Black and White` (noted `B&W`) is a generalization of the three types `White`, `Black` and `Gray`. Figure 2 below presents part of the concept type lattice used for the houses.

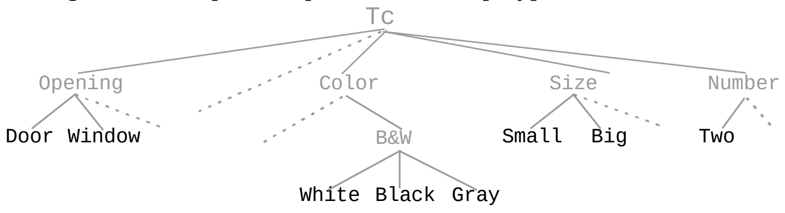

Fig. 2. Part of the concept type lattice used for the houses.

3. *Group* the generalized arcs and initial arcs covering the same set of objects. For example, the arc `[Window]->(color)->[B&W]` is a generalization of the two arcs (thanks to the type lattice above on figure 2): `[Window]->(color)->[Gray]` and `[Window]->(color)->[White].` This arc will be part of the description of the node covering objects described by one of these arcs.

4. *Filter* the generalized arcs. Indeed, for a given matching there are several possible generalizations. For example, the two arcs `[Window]->(color)->[B&W]` and `[Window]->(color)->[Colour]` are both generalization of the arcs: `[Window]->(color)->[White]` and `[Window]->(color)->[Gray].` This step considers each set of arcs for a node and chooses the arcs that will form the description of this node in the GS. In constructing the GS, the number of generalizations is limited while considering only the most specific ones. The filtering step thus consists in memorizing only the most specific arcs (on the example above, the arc `[Window]->(color)->[B&W]).`As COING is using a propositional language, the most specific generalization is unique.
5. Finally, the nodes are *connected* thanks to the inclusion relation existing among their coverage.

Figure 3 summarizes the principle of the GS construction.

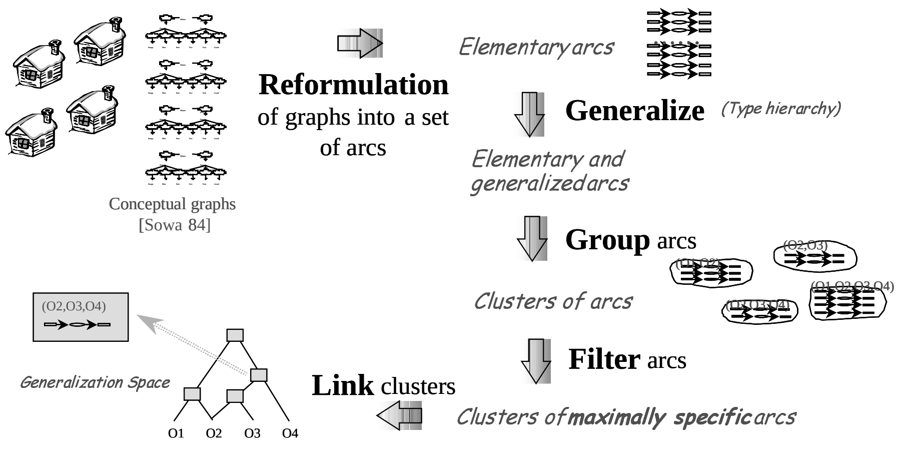
Conceptual graphs [Sowa 84]
Reformulation of graphs into a set of arcs
Elementary arcs
Generalize (Type hierarchy)
Elementary and generalized arcs
Group arcs
(O2,O3)
(O1,O2)
(O1,O2,O3,O4)
(O1,O3,O4)
Clusters of arcs
Filter arcs
Clusters of maximally specific arcs
Link clusters
(O2,O3,O4)
Generalization Space
O1
O2
O3
O4

Fig. 3. Principle of the construction of the most specific Generalization Space.

In order to illustrate the COING approach, let us consider the three houses h1, h2 and h3 whose descriptions need to be clustered. These houses are described by their windows which have two proprieties: a `color` and a `size`. Figure 4 below presents the GS build by COING for these houses.

This Generalization Space contains two class nodes (n1 and n2) and three object nodes corresponding to the houses (box nodes). The node n2, for example, clusters the houses h2 and h3. Its coverage is {h2, h3} and its description is the arc `[Window]->(color)->[Gray].` This class node indicates that h2 and h3 have at least a `gray` window in common in their descriptions and that this property is not shared by any other object considered. Thanks to the structure of the GS, we may add the description of the root node (n1) to this description. More precisely, we add the arcs from n1 which are not generalizations of arcs from n2, for example the arc `[Window]->(Size)->[Big].` Finally, the GS indicates that the two houses h2 and h3 have `window(s)`, which have a size (`Small,Big`) and a color (`Gray` and `Black`).

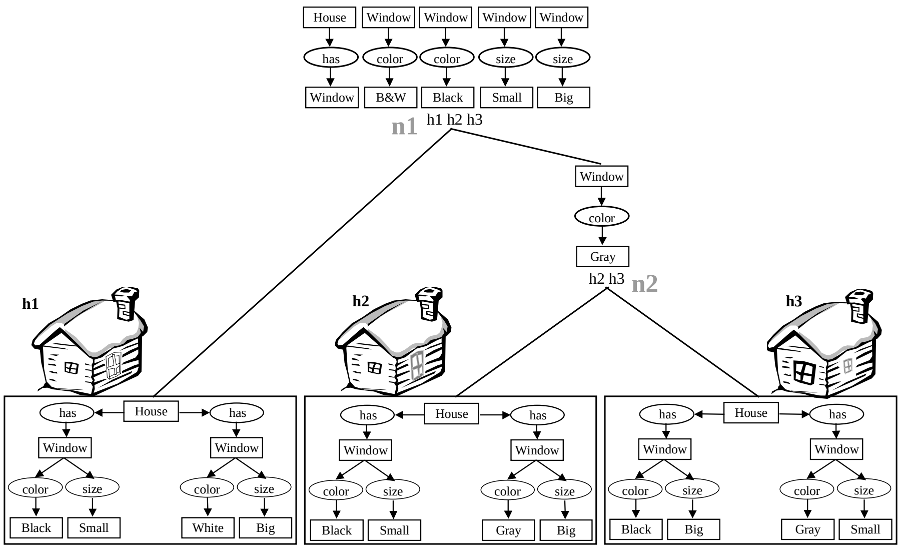

Fig. 4. Generalization Space built by COING.

Let us clarify why the arc `[Window]->(color)->[B&W]` appears in the root node and why the arc `[Window]->(size)->[Size]` does not. This explanation will clarify the 3rd step of the COING principle (cf. previous page).

- The arc `[Window]->(color)->[B&W]` is a generalization of the arc `[Window]->(color)->[Black].` As this last arc is more specific and since they have the same coverage on objects ({h1, h2, h3}), the arc `[Window]->(color)->[B&W]` should not appear. However, this arc is *useful* because its coverage on arcs is bigger than that of `[Window]->(color)->[Black]:` it also covers the arcs `[Window]->(color)->[White]` and `[Window]->(color)->[Gray]`. In fact, this arc tells us that there is a window whose color is `[B&W].`
- Consider now the arc `[Window]->(size)->[Size].` It is more general than both the arcs `[Window]->(size)->[Small]` and `[Window]->(size)->[Big]`. The coverage on objects of these three arcs is the same ({h1, h2, h3}). The coverage on arcs of `[Window]->(size)->[Size]` is exactly the union of the coverage on the arcs of the two arcs `[Window]->(size)->[Big]` and `[Window]->(size)->[Small].` The arc `[Window]->(size)->[Size]` is therefore not useful and not informative; it should not be part of the root node description.

In order to deal with the traditional knowledge representation tradeoff [11] between an expressive language and an efficient algorithm, COING reformulates conceptual graphs into conceptual arcs. This simplification supplies the COING algorithm with a quadratic complexity in the number of objects, but restricts the generalization language, i.e. the expressiveness of the GS node descriptions. Let us illustrate this point using the house example. The three houses h1, h2 and h3 all have a small window and a black window; for h1 and h2 it is the same window, whereas for h3 it is not. This difference does not appear in the classification built by COING (see fig.4) since it requires representing relations between arcs.

3 Organize knowledge in a hierarchy of generalization abstraction spaces

Building an organization of relational descriptions requires to build a Generalization Space whose nodes use a relational representation. Given a set of objects described as graphs in the conceptual graph formalism, each node in the GS would ideally be represented by the graph that is the most specific generalization of the graphs describing the objects it covers. Let us note this Generalization Space as GS_{max}. In fact, due to the complexity of the subsumption relation and the exponential growth of the length of the least general generalization, building GS_{max} directly using an exhaustive method is not practical. The matching curse is also true for the first-order languages used in Inductive Logic Programming (ILP); they define syntactic restrictions on clauses to devise efficient ILP algorithms [16] which are similar to the restrictions on graphs used to devise graph-based algorithms [1], [12].

The solution proposed in this paper is to build an initial GS using a propositional language and then to iteratively enrich this GS. This enrichment consists of refining the descriptions of existing nodes or adding new nodes. We present in the following sections our approach, called KIDS, which is using COING and relies upon the abstraction of relational data.

3.1 KIDS principle

KIDS is based upon the following property of the GS which allows us to limit the search space at each step of the algorithm:

> *If there exists a sub-graph S_{gn} which generalizes n object descriptions, then there is in the GS built by COING a node whose coverage contains these n objects (and possibly others) and whose description contains all the arcs of the generalizing sub-graph S_{gn}.*

In other words, this property of GS means that to enrich any node of a GS, it is sufficient to restrict the search for richer descriptions only to the objects it covers. This principle simplifies the process of enriching a GS. In effect, the nodes of GS_0 (found by COING) are a subset of the nodes of a GS whose generalization language is richer than the one used in COING and the description of each node of GS_0 is more general than that of GS.

In order to find richer descriptions of GS nodes, our approach consists of *gradually* increasing the matching structure, i.e. the matching structure is made more complex at each step of the algorithm. At each step, the objects descriptions are reformulated based upon this structure into a propositional language. The reformulates descriptions may then be processed by the COING algorithm.

More precisely, KIDS uses sub-graphs to represent the relational nature of the descriptions. In order to reformulate these sub-graphs into a propositional language that may be performed by COING we make an *abstraction*. This abstraction transforms the sub-graphs representation into a representation appropriate to COING : a structure like *[concept-type]->(relation-type)->[concept-type]*. In fact, the relation-type is replaced by an " *abstract relation* " representing the matching structure. For example, at the 1st level of KIDS (first step of the algorithm), an arc performed by COING is:

[House] $\boxed{\texttt{-> (has) -> [Window] -> (size) -}}$*> [Small]*

The triplet *(has)->[Window]->(size)*, which is in the box, is an abstract relation. Figure 5 below presents the general KIDS principle.

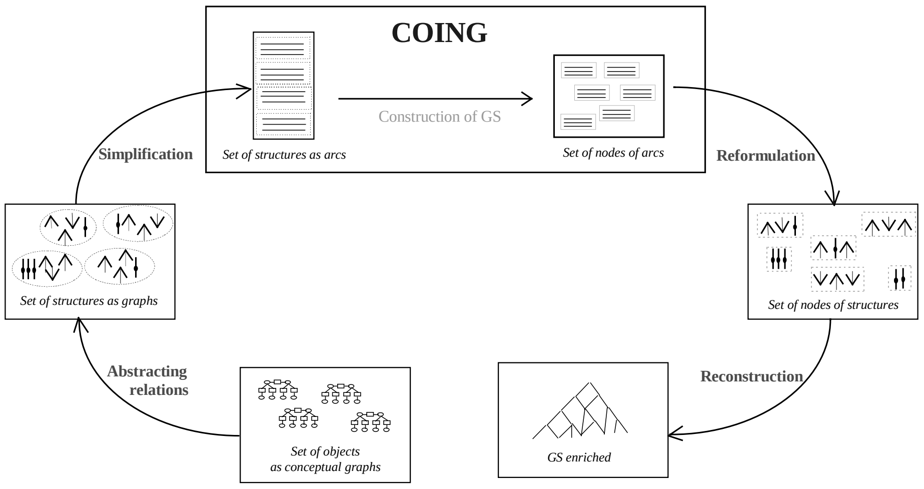

Fig. 5. Principle of KIDS.

3.2 Towards a new generalization language

To enrich at each step the matching structure is equivalent to modify at each step the generalization language. KIDS starts with a language of arcs (provided by COING), then it uses at the first step a language of couples of connected arcs, then at the second step a language of triplets of connected arcs, etc.. These successive generalization languages are expressed according to particular connected sub-graphs: *sequence, star* and *hole structures*.

Definition 1: A *sequence* is composed of a succession of arcs, which are connected one-to-another thanks to a common concept. This concept is the origin of the first arc and the target of the other one.

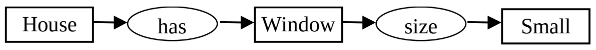

Fig. 6. Example of a sequence-structure composed of two arcs through the common concept of Window

Definition 2: A *star* is composed of a set of conceptual arcs which have the same origin.

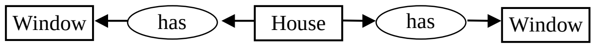

Fig. 7. Example of a star-structure composed of two arcs through the concept of House

Definition 3: A *hole-structure* is composed of a set of conceptual arcs which have the same target.

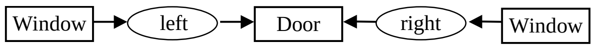

Fig. 8. Example of a hole-structure composed of two arcs

The number of arcs of an abstract relation depends on the level of KIDS (the step of the algorithm): two connected arcs at the 1st level, 3 at the 2nd level, …, i+1 arcs at the ith level. The more the sub-graph structure is complex, the more the matching for the reformulation is expensive. Nevertheless, the specific structure of the GS and the iterative method of KIDS allow us to limit the number of nodes to explore at each step.

3.3 KIDS algorithm

The principle of the KIDS algorithm is to explore, at the *ith step*, only the nodes which may be enriched, i.e. the nodes whose descriptions potentially contain an ith level structure. In practice, at step (i+1)th, KIDS explores all the nodes which were modified in step i. Indeed, an (i+1)th level structure is the aggregation of an ith structure and *one* arc. We define a *candidate* node for KIDS at step i+1 a node which has been modified in step i. In the first step, KIDS explores all the GS nodes built by COING. The GS enrichment algorithm is as follows (cf. Table 1):

1. For each object covered by a candidate node, determine its ith level description: (i+1) connected arcs. It consists of abstracting the object descriptions using the three structures: sequence, star and hole.
2. Apply COING to the reformulated object descriptions. The result is the addition of new nodes to the GS and/or the modification of the descriptions

of existing GS nodes. Notice that the new descriptions found by COING have to be reformulated in terms of sub-graphs. It consists of reformulating the descriptions using the abstract relations.

3. If KIDS modifies the GS at the ith step, then repeat the method from 1) at the (i+1)th level (i+2 connected arcs).

```
KIDS_Algorithm (GS: Generalization Space; l: level)
GS_modified ← false
Nodes_List ← list of GS candidate nodes
for all the nodes n of Nodes_ List do
   Objects_ List ← Description of n’s objects at the lth level
   GS_enriched ← COING_Algorithm(Objects _List)
   if GS_enriched modified then GS_modified ← true
   GS ← Add (GS_enriched, GS)
end for
if GS_modified == true then KIDS_ Algorithm(GS,l+1)
```

Table 1: KIDS main algorithm

While the complexity of the matching for generalization is avoided by the use of abstract relations, the complexity of graph matching is not suppressed; it is instead moved to the reformulation of the descriptions. In fact, the more complex the abstract relations are (the higher the KIDS level), the more complex the reformulation is. Nevertheless, the GS’s specific structure and KIDS’s iterative method allow us to limit the number of nodes to be explored at each step, while exploring only the ones that can be enriched.

However, in order to find all the structural similarities among the descriptions, KIDS needs to be applied up to the level of structure of the maximum level in the objects descriptions. In other words, if there are at least two descriptions including a structure of level l, KIDS will have to be applied up to the l level to assure a search for all the similarities.

KIDS stops either when there is no more candidates node, or when it is not possible to describe the objects at the next level (there is no structures of (i+2) arcs in the descriptions). Experimentally, the time needed to apply the algorithm at the next level may be evaluated from the time needed to build the GS at the previous level. It is possible to approximate the time required for the next level and to stop KIDS if this time is too long. Experiments in section 5 show that in our particular domain, the increase of time required between two successive levels is linear.

3.4 Organizing relational data with KIDS

Let us consider again the example of the houses presented in section 2.2 (figure 4) to illustrate KIDS improvement over COING. Figure 9 below presents the enriched GS obtained by KIDS at the 1st level ; the information drawn in black is the result of KIDS and in gray those of COING.

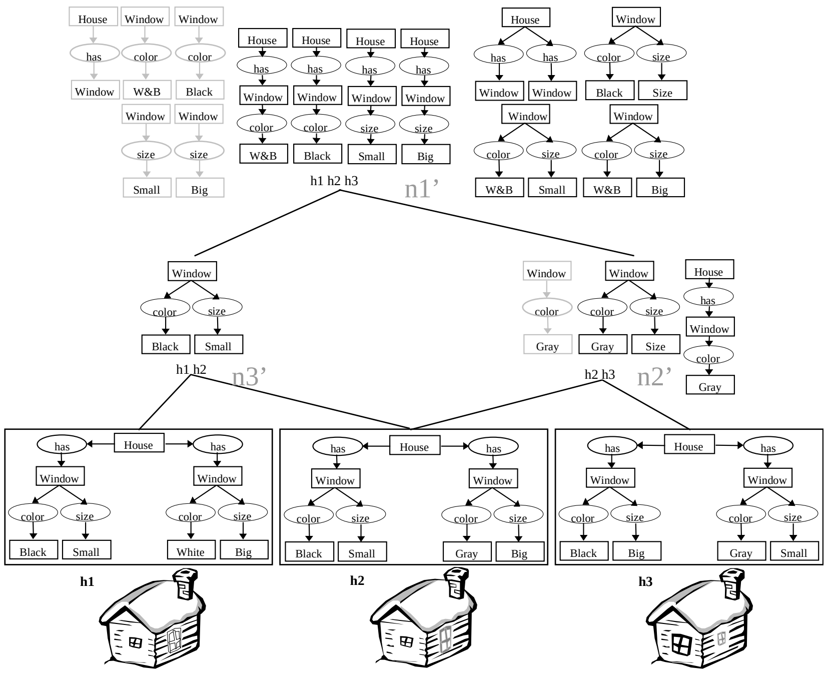

Fig. 9. Generalization Space enriched by KIDS.

The abstraction allows us to discover common substructures between the objects descriptions. At the 1st level, KIDS finds structural descriptions which were not find by COING. For example, COING did not find that all the houses have (at least) two windows and that all these windows have a color (`W&B` or `Black`) and a size (`unknown,Small` or `Big`). Furthermore, COING did not find a class clustering h1 and h2 and only these two houses whereas they have a small black window in common and this window does not appear in the description of h3 (even if h3 has a small window and a black window but it is not the same window). This similarity is found by KIDS at the 1st level, because it is a particular composition of two arcs. On this example, KIDS enriched the description of existing nodes and added a new node clustering h1 and h2. From a GS built using a propositional language, KIDS has allowed to give more precise descriptions on the existing similarities between the objects thanks to an abstraction of sub-graphs.

On this example, it is useless to apply KIDS at the 2nd level. Indeed, the stars and sequences of h1, h2 and h3 descriptions are of 1st level, i.e. they connect 2 arcs. Once the descriptions are reformulated using 1st level structures, there is only one way to rebuild the description; the reformulation using first level structures is not ambiguous, nor losses information. Figure 10 illustrates this idea.

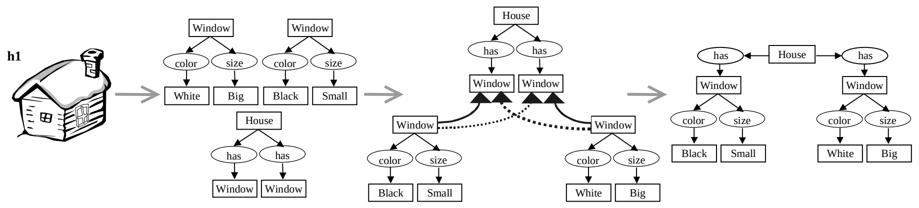

Fig. 10. Rebuilding a graph from its decomposition in structures.

4 Experiments

This section presents an application of the above method in the framework of the construction of a classification of Chinese characters. We briefly remind the context of this work. For more information about this application, the reader should refer to [2]. These experiments aim to show the feasibility of KIDS in terms of complexity and to illustrate its interest for relational data organization.

4.1 Description of the relational data

The database considered is a collection of 6780 Chinese characters. Each character is represented by a conceptual graph. Characters are described by : their initial and final pronunciation, the ton of this pronunciation, the components (between 1 and 5) and their relative positions and the key component. For example, the character 情, which is composed of the radicals C5381 and C2843, which is pronounced “ qing ”, which is in ton 2 and means "feeling", is represented by the conceptual graph of figure 11.

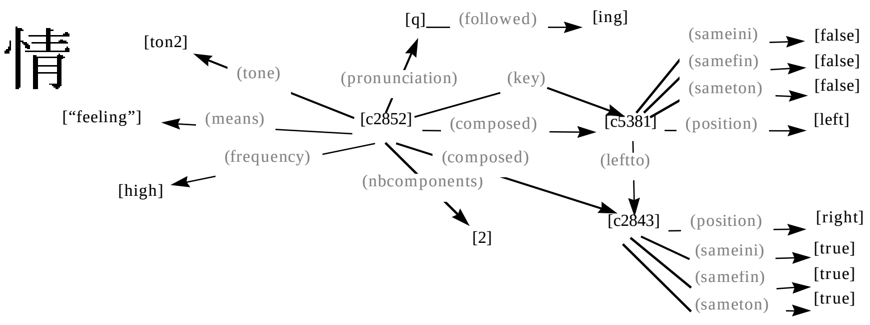

Fig. 11. Conceptual graph describing the character 情.

The type lattices used for the Chinese characters are the following :

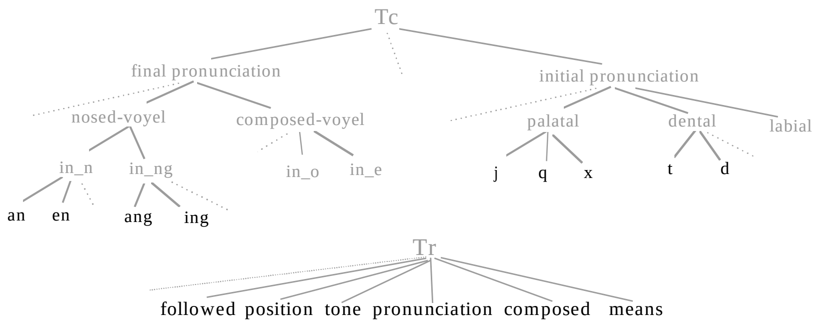

Fig. 12. Part of the type lattices for the Chinese characters.

4.2 Results and discussion

We evaluated KIDS on several databases of characters composed of 10 to 140 or 416 characters. Figure 13 shows the total time required for generating the GS for 8 of these databases using the COING and the KIDS algorithms.

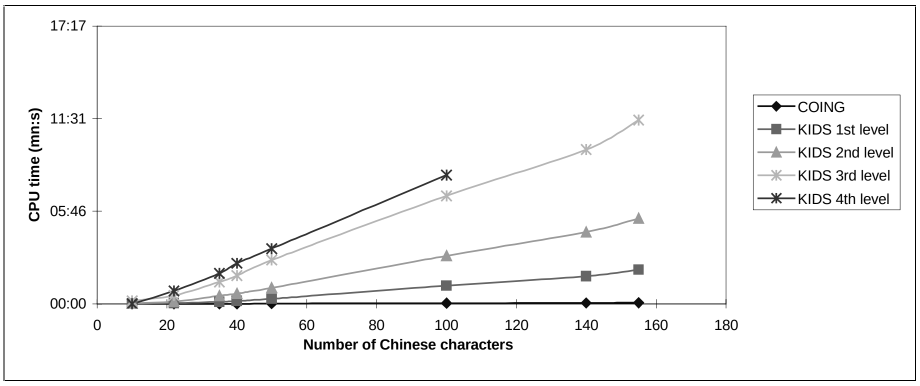

Fig. 13. Average execution time of COING and KIDS on Chinese characters databases.

In practice, the CPU time of the proposed algorithms is linear (it is quadratic in the worst case in COING [1]) with the number of objects. This results may surprise because, as it manipulates sub-graphs, KIDS introduces a complexity factor. However, the combinatorial explosion due to the generalization of sub-graphs is limited since the bigger the level of KIDS is (i.e. the more complex are the graphs to generalize) the less the number of sub-graphs to perform is.

The level introduces a multiplicative factor. The linear growth means that on the average, the time necessary to move to the next level is very close to be constant. Figure 14 illustrates this result.

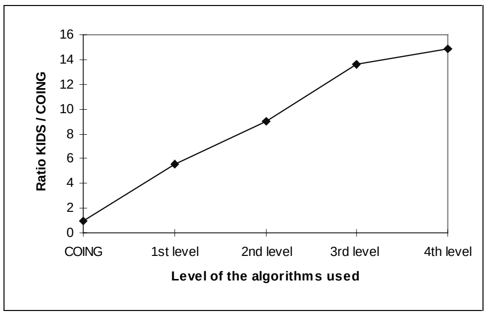

Fig. 14. Evolution of the multiplicative factor as a function of the algorithms used.

During these experiments, we also evaluated the evolution of the number of nodes of the GS as a function of the algorithms used. For COING, this number is in the worst case in O(N) [1]. Figure 15 summarizes these results.

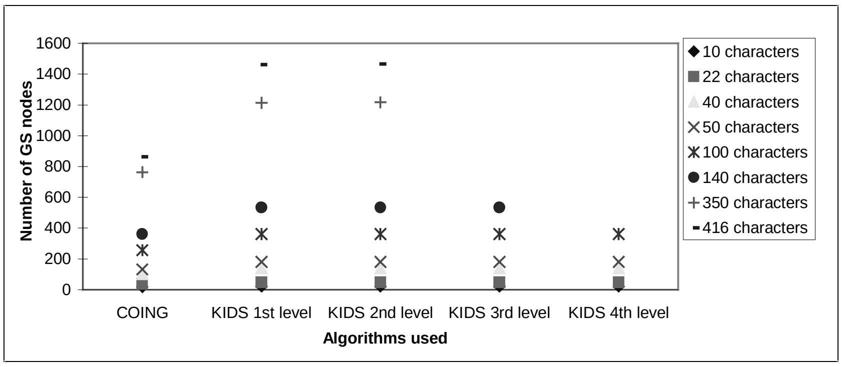

Fig. 15. Evolution of the number of nodes of the GS.

This graph shows that the number of nodes of the GS grows until a specific level – 1st level for the small bases and 2nd level for largest – then it becomes constant. This may be explained by the fact that from a specific level, KIDS does not allow to create new classes, but only to enrich the already existing ones with more complex descriptions.

5 Conclusion

We have presented KIDS, an algorithm for organization of relational data. This algorithm is iterative and is based upon an abstraction of the description. In a first step, it builds the space of the most specific generalizations using a propositional language. Then it uses reformulation to find more complex descriptions. We have implemented and successfully tested our approach. Our experiments suggest that the proposed method provides an organization of relational concepts while keeping a linear complexity in practice with the number of objects. This result is due to the fact that the more complex are the structure, the less are the nodes to explore.

The first perspective of this work is to characterize more precisely the generalized language used in the enriched GS. Indeed, as soon as we work on the o-level structures, there is no longer a unique most specific generalization and GS nodes may be redundant. The characterization of the enriched language of GS allows us to evaluate the usefulness of the sub-graphs and to filter them in order to keep the useful one.

Another possible improvement of the algorithm is to define methods to evaluate the interest of KIDS for a given database. Indeed, when the concepts in the objects of a conceptual graphs database appear only once, it is not necessary to apply KIDS to this database, because the decomposition does not cause a loss of information. In contrast, if a concept appears several times in the objects descriptions (like in the houses), it is not possible to differentiate them. So, we can consider a pre-processing on the data to evaluate the maximal level of KIDS application.

Finally, we plan to extend this method for a more efficient processing of numerical data. Currently, the numerical information contained in descriptions is processed like symbols ; the implicit order existing between numbers is not taken into account. A preprocessing on descriptions would make it possible to determine a hierarchy of generalization of the numerical values. The creation of new values of attributes, as it is the case in constructive induction, would make it possible to better account for the similarities between descriptions [10], [18].

6 References

1. Bournaud I., Ganascia J.-G.: Accounting for Domain Knowledge in the Construction of a Generalization Space. ICCS'97, Lectures Notes in AI n°1257, Springer-Verlag (1997) 446-459.
2. Bournaud I., Zucker J.-D.: Integrating Machine Learning Techniques in a Guided Discovery Tutoring Environment for Chinese Characters. International Journal of Chinese and Oriental Languages Information, Processing Society, 8(2) (1998).
3. Carpineto C., Romano G.: GALOIS: An order-theoretic approach to conceptual clustering. Tenth International Conference on Machine Learning (1993).
4. Chein M., Mugnier M.L.: Conceptual Graphs : Fundamental Notions. Revue d'Intelligence Artificielle, 6(4) (1992) 365-406.
5. Fisher D.: Approaches to conceptual clustering. Ninth International Joint Conference on Artificial Intelligence, Los Angeles, CA, Morgan Kaufmann (1985).
6. Fisher D.: Knowledge Acquisition Via Incremental Conceptual Clustering. In: Michalski, R.S., Carbonell, J., Mitchell, T.(eds.): Machine Learning: An Artificial Intelligence Approach. San Mateo, CA, Morgan Kaufmann. II (1987) 139-172.
7. Fisher D.: Iterative Optimization and Simplification of Hierarchical Clusterings. Journal of Artificial Intelligence Research 4 (1996) 147-179.
8. Gennari J. H., Langley P., Fisher D.: Models of incremental concept formation. Artificial Intelligence 40-1(3) (1989) 11-61.

9. Ketterlin A., Gancarski P., Korczak J.J.: Conceptual clustering in Structured databases : a Practical Approach. Proceedings of the Knowledge Discovery in Databases KDD'95, AAAI Press (1995).
10. Kietz J.U. & Morik K.: A polynomial approach to the constructive induction of structural knowledge. Machine Learning 14(2) (1994) 193-217.
11. Levesque H.J. and Brachman R.J.: A fundamental tradeoff in knowledge representation and reasoning. In: Brachman, R.J, Levesque, H.J. (eds.): Readings in Knowledge Representation. Morgan Kaufmann (1985) 41-70.
12. Liquiere M., Sallanatin J.: Structural Machine Learning with Galois Lattice and Graphs. *Fifteen International Conference on Machine Learning (ICML),* (1998).
13. Michalski R. S., Stepp R. E.: An application of AI techniques to structuring objects into an optimal conceptual hierarchy. Seventh International Joint Conference on Artificial Intelligence (1981).
14. Michalski R. S.: A theory and methodology of inductive learning. Machine Learning: An Artificial Intelligence Approach I, Morgan Kaufmann (1983) 83-129.
15. Mineau G., Gecsei J., Godin R.: Structuring knowledge bases using Automatic Learning. Sixth International Conference on Data Engineering, Los Angeles, USA (1990).
16. Muggleton, S., Raedt L. D.: Inductive Logic Programming: Theory and Methods. Journal of Logic Programming 19(20). (1994). 629-679.
17. Sowa J. F.: Conceptual Structures: Information Processing in Mind and Machine. Addisson-Wesley Publishing Company (1984).
18. Wnek J., Michalski R.: Hypothesis-driven constructive induction in AQ17-HCI : a method and experiments. Machine Learning 14(2) (1994) 139-168.
19. Zucker J.-D., Ganascia J.-G.: Changes of Representation for Efficient Learning in Structural Domains. International Conference in Machine Learning, Bari, Italy, Morgan Kaufmann (1996).