Automatic Knowledge Revision in a Generalisation System

Patrick Taillandier
Laboratoire COGIT – IGN France
2 avenue Pasteur
94165 Saint Mandé CEDEX - France
patrick.taillandier@ign.fr

Abstract: We are interested in this paper in the automatic revision of procedural knowledge for generalisation systems based on the agent paradigm. Our approach consists in analysing the system execution logs and extracting new knowledge from thee logs using machine learning techniques. The objective is not only to improve the system in terms of efficiency and of effectiveness but also to allow it to automatically adapt itself to various uses and to be able to evolve when adding new elements. A first experiment has been carried out on the generalisation of housing estates to validate the approach.

Keyword: Generalisation, Procedural Knowledge Revision, Machine Learning, Agent Paradigm.

1 Introduction

The generalisation automation is a complex problem, which was in the centre of numerous research works in the recent years. Some of these works try to solve it by a local, step by step and knowledge-based approach (Brassel & Weibel, 1988; Beard 1991; McMaster & Shea, 1992).
The work presented in this paper takes for framework models based on this approach, which used the agent paradigm (Ruas, 1999; Duchêne, 2004).

The objective of our work is to propose methods to tune and to revise automatically, by the means of experiments and without the intervention of generalisation experts, the procedural knowledge contained in this kind of model. The interest of this revision is (a) to get the most effective (quality of the results) and most efficient (quickness to get the results) knowledge as possible from a good but not perfect knowledge set, (b) allow the knowledge to evolve when adding new elements (i.e. algorithms) in the system.

In part 2, we present the context in which our work takes place and its objective. We give in particular a short and non-exhaustive state-of-the-art of the use of the machine learning techniques within the context of generalisation. Then, we present the AGENT generalisation model for which our approach is dedicated. Part 3 is devoted to the presentation of our approach. Part 4 describes the first application that we carried out as well as its results. Part 5 concludes and presents the future works.

2 Context and objective

2.1 Generalisation and machine learning

The automation of generalisation processes requires the introduction of much knowledge, e.g. the choice of actions to apply to a geographical object. This knowledge can be acquired thanks to experts of the domain. Collecting and formalizing the knowledge is a problem known as the "*bottleneck of the acquisition of knowledge*". This problem was indeed identified within the field of the automation of the generalisation (Rieger & Coulson, 1993; Weibel et al., 1995; Kilpeläinen, 2000).
One of the approaches to face and to widen the bottleneck is to use the machine learning techniques. Several works have already used machine learning in the context of generalisation (Weibel et al., 1995; Mustière, 2001; Mustière & Ruas, 2004). The use of such techniques requires solving two kinds of difficulties (Ruas et al., 2006).

The first one concerns the choices of the knowledge to learn as well as its formalization (Mustière & Ruas, 2004). These choices appear capital for the quality of the results that it is possible to obtain.

The second difficulty is related to the collection of examples (Mustière, 2001; Ruas & Holzapfel, 2003). It is often fastidious and problematic to build learning sets from examples labelled by experts. To face this difficulty, a possible approach is to integrate directly in the GIS an environment facilitating the collection of examples as well as their analysis (Duchêne et al., 2005). Another approach to solve this problem is to directly acquire the examples by the means of experimentations without passing through experts. Two series of works illustrate this approach. They take two different systems of generalisation as a framework, which are both based on the exploration of search trees by tests/errors in order to determine the best sequence of actions to apply. These works seek to analyse previous generalisations to deduce knowledge on the choice of the actions to apply for a given state. (Burghardt & Neun, 2006) use the previous experiments to build this type of knowledge in the form of a case base. (Dyèvre, 2005; Ruas et al., 2006) seek to acquire this knowledge in the form of rules. The advantage of a rules based representation is to be directly interpretable. This latter approach, dedicated to the AGENT generalisation model (Ruas 1999; Barrault et al., 2001; Regnauld, 2001 described in 2.2), is based on the analysis of the execution logs to learn new rules. Experiments were carried out for the generalisation of buildings.

Our approach of automatic acquisition of procedural knowledge is in the continuity of the latter approach and it also takes for framework the AGENT generalisation model.

2.2 The AGENT model

The AGENT generalisation model, used in the AGENT European Project (Lamy et al., 1999; Barrault et al., 2001), originates in (Ruas 1999). In this model, geographical objects (roads, buildings, etc) are modelled by agents.
(Weiss, 1999, page 2) defines the agents as "*[…] autonomous, computational entities that can be viewed as perceiving their environment through sensors and acting upon their environment through effectors. […]. Agents pursue goals or carry out tasks in order to meet their design objectives […]*".
The geographical agents manage their own generalisation, choosing and applying generalisation algorithms to themselves.

In the model, two types of agents are considered:

- The **micro agents,** which represent the single geographical objects (building, section of road, etc)
- The **meso agents**, which represent groups of geographical objects. A meso agent can be composed of micro agents (a housing estate composed of buildings) as well as of others meso agents (a district composed of housing estates).
  This results in a hierarchical organisation. The agents of higher level will trigger off the generalisation of their sub-agents.

The generalisation of the agents (micro and meso) is guided by a set of constraints that translate the specifications of the desired cartographic product. An example of constraint is, for a building agent, to be sufficiently big to be readable. Each constraint related to an agent is modelled as an object associated with this agent. The constraints also have the role of proposing to their associated agent, for each state, a list of plans (one or more actions) to apply. For example, if the size constraint of a building assesses that the agent is too small, it will propose a scaling action to it. Each constraint carries the following attributes:

- **Goal value**: translates the specifications of the desired product.
- **Current value**: translates the current state of the constraint. This value is calculated for each state.
- **Satisfaction**: reflects the satisfaction degree of the constraint. This integer is calculated for each state from the current value and from the goal value. It is marked out of 10 (1 = not satisfied at all, 10 = perfectly satisfied).
- **Priority**: reflects the emergency of treatment of a constraint. The higher the priority of the constraint will be, the more its plans will be applied in priority. It is marked out of 5 (1 = not urgent, 5 = very urgent).
- **Importance**: corresponds to the importance for the final result that this constraint is satisfied. It is marked out of 5 (1 = not important, 5 = very important). This value is not correlated with the priority. Sometimes, a constraint absolutely needs to be satisfied, and thus has a high importance, but in order to achieve the optimal generalisation, plans proposed by other constraints have to be applied before the plans of this constraint.

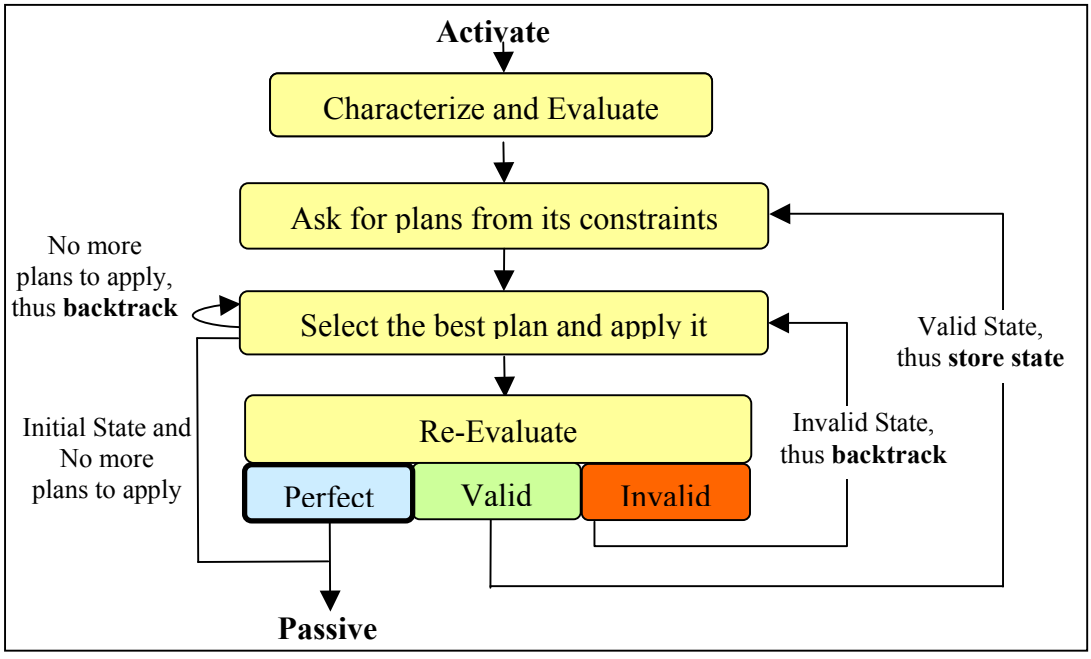


Figure 1 – *Simplified view of the actions cycle*

To satisfy its constraints as well as possible, a geographical agent carries out cycles of actions (figure 1) during which it tests different plans proposed by its constraints in order to reach a perfect state (where all of its constraints are perfectly satisfied) or at least the best possible state. The actions cycle results in a depth-first search exploration of a search tree. Figure 2 gives an example of the obtained tree for the generalisation of a building from 1 m resolution DB to 1:25 000. The passage from a state to another corresponds to the application by the agent of one of the plans suggested by at least one of its constraints.

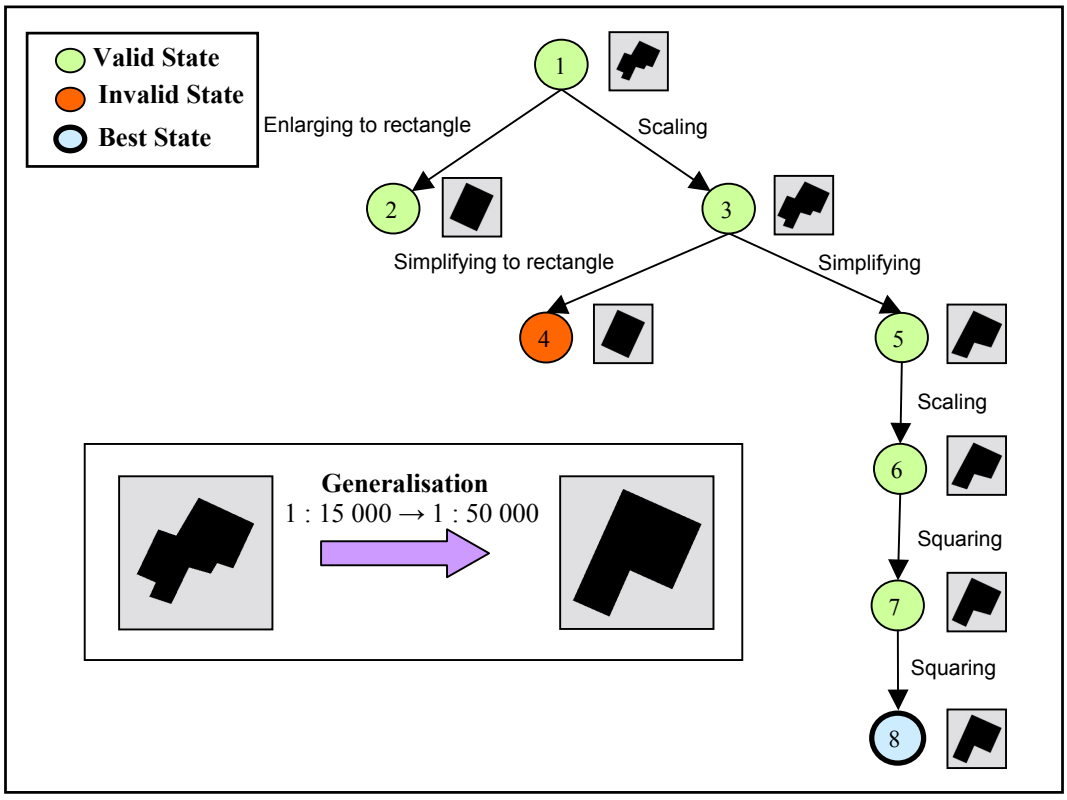


Figure 2 – *Example of a search tree for a building*

3 Proposed Approach for the knowledge revision

3.1 General approach

Just like the approach presented by (Dyèvre, 2005; Ruas et al., 2006), our general approach is based on three stages (figure 3):

- **Exploration stage**: consists in logging the process while it generalises a great number of geographical objects. During this phase, the process uses the procedural knowledge initially contained in the system. The logs contain the whole information related to successes/failures of the various procedural knowledge of the system.
- **Analysis stage**: consists in analysing the logs obtained during the previous stage and in deducing new knowledge from it.
- **Exploiting stage**: consists in testing the system with the new knowledge. If the result obtained at this stage, is not good enough, it is possible to go back to the two first stages in order to learn better knowledge.

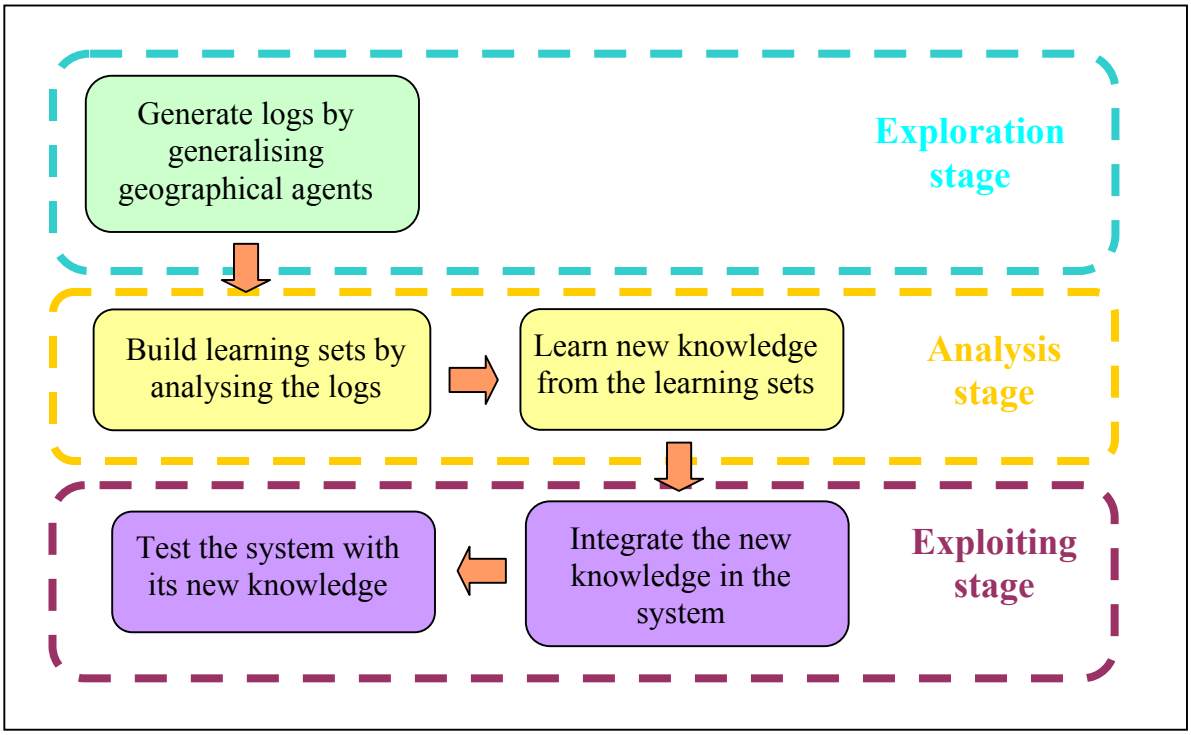


Figure 3 – *General approach*

3.2 Problematic

Applying this methodology for the revision of the procedural knowledge of the AGENT model arises some questions.

Firstly, what should be logged in order to be able to analyse the process? This question joins the problem arising by (Mustière & Ruas, 2004). The first step is to precisely define the pieces of knowledge that we wish to acquire. In our case, we are interested in all procedural knowledge that concerns the exploration of the search tree, e.g. the choice of the actions to apply for a given states or the validity of a state. The objective is to find the best possible state by the quickest way (to well guide the agent in its choices applying actions) and to limit the number of useless states (to prune the tree). For each pieces of knowledge that we seek to acquire, it is necessary to determine the elements that will be used as a basis for the learning. The most important point, which also belongs to the problems rose by (Mustière & Ruas, 2004), is the definition of the description language of the examples and particularly the state characterisation of the search tree. It is necessary to get a representation language rich enough to be able to correctly describe the states and to avoid the noises of description. At the same time, this representation language should not be too wide in order to avoid the explosion of the size of the hypothesis space, i.e. the total number of rules that can be derived from the examples description. It would indeed degrade the result of the learning. In our case, to characterize a state, we will take the satisfaction of its constraints. These values are integers ranging between 0 and 10. They allow a correct characterisation of the agent's state even if this characterisation is not complete. The quality of this characterisation directly depends on the considered constraints. If some descriptive aspects of the state of the agents are not covered by the constraints, the quality of the learnt knowledge might not be ensured.

Another question to consider is the choice of the learning method. We should in particular choose the type of algorithm to use for the learning. We are in the case of a supervised learning because we have labelled examples that are examples described in the form of attributes/label, the attributes describing the characteristics of the example (its state) and the label, the example class. The analysis a posteriori of the system execution logs allows us to build sets of labelled examples. For example, we can determine by analysing the search tree

obtained from the generalisation of an object, the best algorithm to use for some states of the tree.
We wish to be able to validate the learnt knowledge by generalisation experts. An advantage to get an interpretable predictive model is to bring useful knowledge on generalisation itself. The simplest techniques to carry out this learning are the techniques of symbolic learning such as the techniques based on decision trees induction.
There are no simple rules to help to choose a machine-learning algorithm particularly fit to a learning problem. Therefore, we will not seek to test several learning algorithms. We will just use the well-established algorithm C4.5 (Quinlan, 1993) that builds decision trees, which are easily translatable into rules.

3.3 Knowledge to acquire

The knowledge we seek to acquire can be divided into two groups: those related to the way the search is explored (for a given state, the order in which the action are tried) and those related to the pruning of the search tree.

3.3.1 Priority of the constraints

Two factors are taken into account in the AGENT model for the order assigned by the agent to the plans proposed by its constraints for a given state:

1 The priority of the constraint which proposes the plan. The higher the priority, the more the plans it proposed is tested in priority.
2 If ever a constraint proposes several plans, the order is determined by the weight which is attributed for each plan. The higher the weight, the more the plan would be tested in priority.

Thus, we can acquire two kinds of knowledge: the priority of each constraint for a given state and the weight of each plan. In this paper, we only develop the first one.

The solution we propose for the determination of the constraints priorities is that each constraint learns a rules base saying if it should or not be considered as having priority for a given state, ie if the constraint has the highest priority compared to the others ones or not. Each learned rule has a calculated confidence rate according to the successes/failures that the rule met on the training set.

The rules are used as follows during the exploiting phase: when an agent is in a given state, it checks each constraint. Among the constraints that declare themselves as having priority, the agent chooses the constraint whose checked rules have the highest confidence rate. As a consequence, the plans of this constraint will be tried first.

Example
In the following example, the satisfaction of a constraint C_1 will be noted S(C1), and the confidence rate, CR.

If an agent A has two constraints C_1 and C_2 that have computed the followings rules bases for their priority:

- **Rules for C_1**: if $S(C_1) > 4$ and $S(C_2) < 3$ then priority with CR = 80%
- **Rules for C_2**: if $S(C_2) < 10$ then priority with CR = 75%

If A is in the following state:

- State 1 : $S(C_1) = 7$, $S(C_2) = 8$: only the C_2 rule is checked, then C2 is given priority (and therefore the plans it proposed are tried first)

- State 2 : $S(C_1) = 8$, $S(C_2) = 2$: the C1 rule is checked and CR = 80%. The C_2 rule is checked and CR = 75%. Then, as the CR of the C_1 rule is higher than the CR of the C_2 rule then C1 is given priority (and therefore the plans it proposed are tried first)

The construction of the learning set is done by the analysis of the search trees and more particularly by successes/failures that each plan suggested by the constraints encounters. One learning set by constraint is built. The algorithm of construction of a learning set is the following:

```
For all of the states E of the tree do:
    If E belongs to the best path do:
            For all of the states E_succ successors of E do:
                If E_succ belongs to the best path do:
                    Add training example: E_succ, Priority
                Else do:
                    Add training example: E_succ, Non-priority
                End if
            End for
       End if
End for
```

Each state belonging to the best path (the states sequence starting from the initial state and ending with the best state) is analysed. If, from one of these states, one of the plans suggested by a constraint leads to another state belonging to the best path, the constraint is considered as having priority. If all the plans suggested by the constraint lead to useless states (states that does not belong to the best path), the constraint is considered as not having priority. Figure 4 gives an example of obtained learning sets.

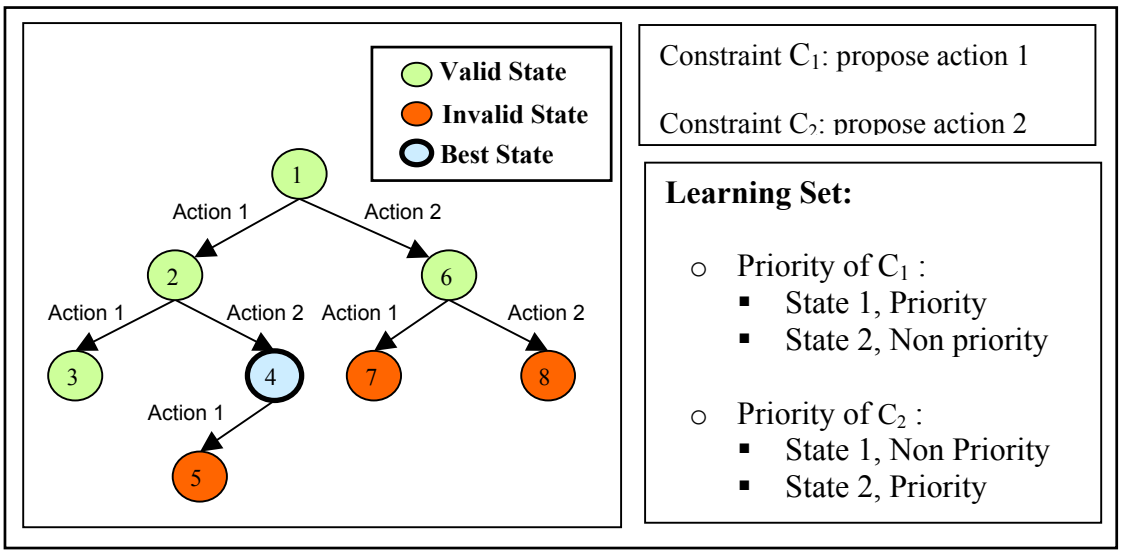


Figure 4 – *Example of learning set for the priority of the constraints*

3.3.2 Pruning of the search tree

We seek to prune the tree in order to improve the efficiency of the process. The tree can be pruned either by a states validity criterion, or by a criterion to exit from the search tree. An invalid state is a state from which the system does not continue to explore in depth the search

tree (even if some constraints are not satisfied and propose actions). To exit the actions cycle means that the system stops to search a better state than the best one already found.

3.3.2.1 Validity Criterion

We first try to learn a states validity criterion. The basic criterion that we choose to integrate in the system is that a state is valid if it is not similar, from constraints satisfaction point of view, to an already visited state. This criterion has been chosen in order to avoid the loops and to get the most expanded trees. Indeed, for our knowledge revision, it is important to learn as much as possible from the generalisation made during the exploration stage. It is possible to add state validity rules to prune the trees more strongly. We will thus learn the concept of validity for a state. The algorithm of construction of the learning set is the following:

```
For all of the states E of the tree do:
     If E belongs to the best path then do:
            Add training example: E, Valid
             For all of the states E_succ successors of E do:
                  If E_succ does not belong to the best path then do:
                        Add training example: E_succ, Invalid
                   End if
             End for
       End if
End for
```

In other words, a state is valid if it belongs to the best path. It is not valid if it is useless and its father belongs to the best path. Figure 5 gives an example of the learning set we can obtain.

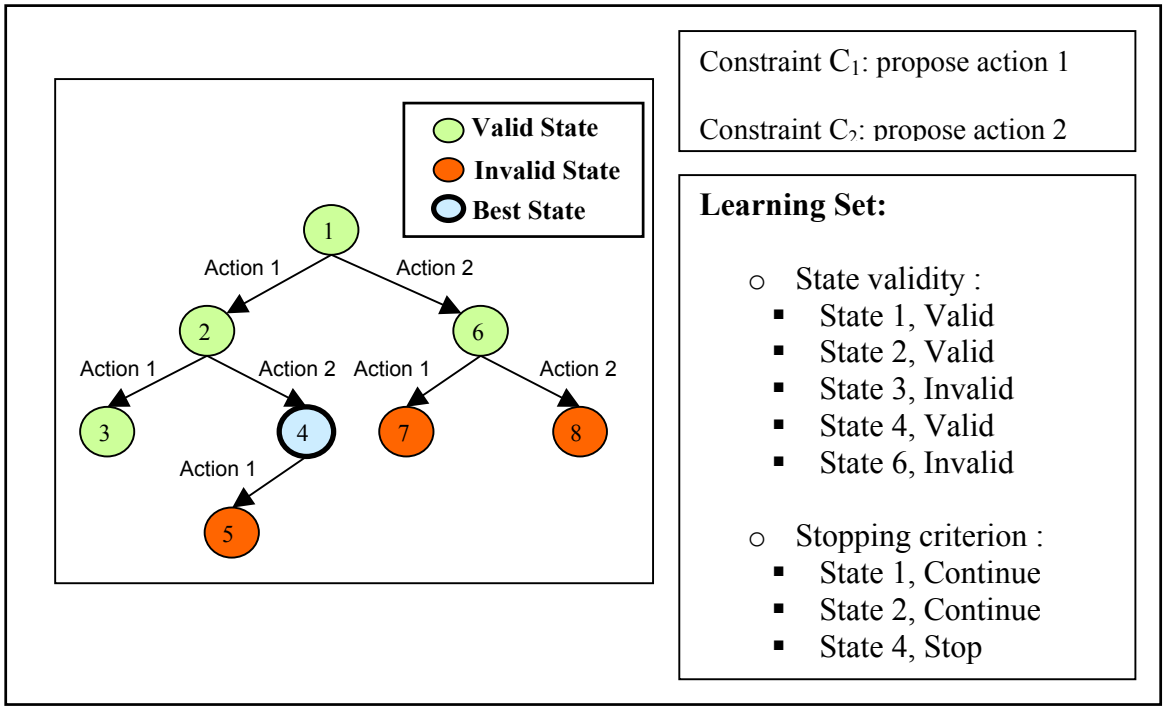


Figure 5 – *Example of learning set for pruning*

3.3.2.2 Stopping Criterion

As mentioned above, the second thing we try to learn in order to prune the trees is a less restrictive stopping criterion. The basic criterion is that the actions cycle stops when a perfect

state is reached (or when all the possible actions were tested). Now, it appears in certain configurations, that it is useless for the system to continue to explore the tree because it is not possible to find a better state than the ones already found. We will learn these configurations in the form of states for which it is useless to continue to explore the search tree. For that, we know that for the states belonging to the best path, it is necessary to continue to explore the search tree as long as the best state is not reached, but once this one reached, it is useless to continue to explore the tree. The algorithm of construction of the learning set is the following:

```
For all of the states E of the tree do:
     If E belongs to the best path then do:
            If E is the best state then do:
                  Add training example: E, Stop
            Else do:
                  Add training example: E, Continue
            End if
       End if
End for
```

Figure 5 gives an example of learning set, which we can build.

3.4 Conclusion for the revision process

Part 3 describes our approach of knowledge revision. The approach is based on tree stages : the *exploring* stage, which consists in logging the process while it generalises a great number of geographical objects, the *analysis* stage which consists in analysing the logs obtained during the previous stage and in deducing new knowledge from it and the *exploiting* stage which consists in testing the system with the new knowledge.

Tree kinds of knowledge are learnt during the analysis stage : the priority of the constraints, the validity criterion and the stopping criterion. We use the C4.5 algorithm that builds decision trees, which are easily translatable into rules, to learn this knowledge from the learning set. The figure 6 summarizes the process.

Now that our approach is described, the next part deals with its applications to the cartographic generalisation of housing estates.

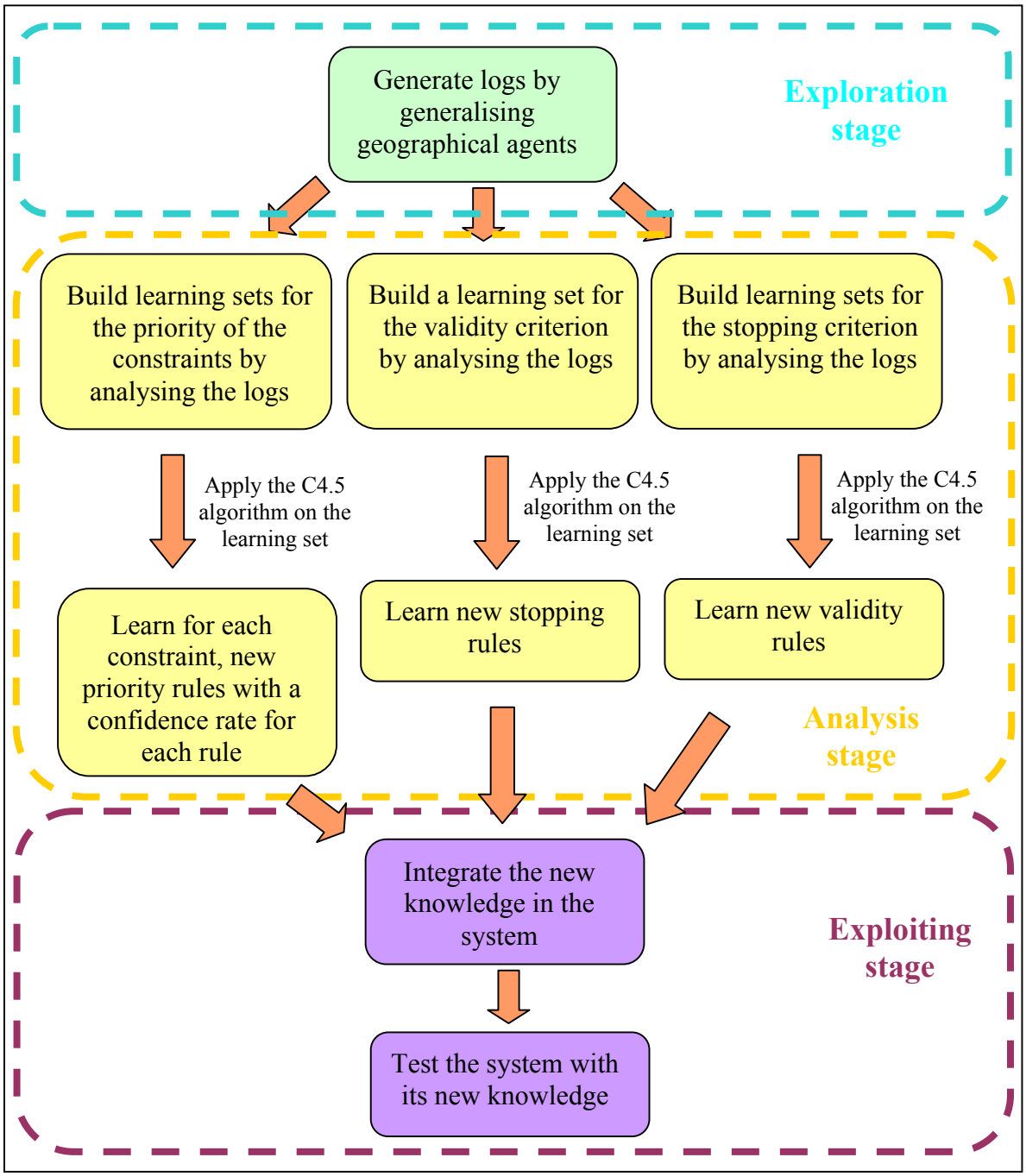


Figure 6 – *Processing scheme*

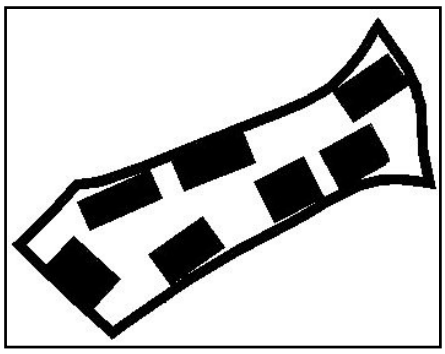
4 Case Study: the generalisation of housing estate

We present in this part, an application of our approach described in part 3, to the cartographic generalisation of the housing estates.

4.1 Application case

We tested our approach of procedural knowledge revision on the generalisation housing estate agents. A housing estate agent is a meso agent composed of various micro agents (buildings, roads, rivers, etc), that matches the following conditions:

- To have a non strong density of buildings
- To contain at least two buildings
- To have an area lower than 300 m^2
- To be composed of buildings of relatively similar size (standard deviation between area, lower than 70 m^2)



Constraints:

- *bldg_satisf* : Satisfaction of buildings
  - Plan : generalisation of buildings
  - Priority = 5 , Importance = 5
- *prox_bldg* : proximity between buildings
  - Plan : aggregation of buildings
  - Priority = 5 , Importance = 4
- *prox_road_bldg* : proximity between buildings and proximity between roads and buildings
  - Plan : removal and displacement of buildings
  - Priority = 5 , Importance = 4
- *buildings_nb* : preservation of a minimal number of buildings
  - No Plan
  - Priority = 5 , Importance = 4

Figure 7 – *Housing estate constraints*

The reference scale of the initial data is approximately 1:15 000, the target scale is 1:50 000. The agents are guided by four constraints (figure 7) for which we have defined static priorities all equals. Thus, the plans application order depends only on the constraints satisfaction. The lower the satisfaction of a constraint, the more the plans of this constraint will be applied in priority. Most of the time, choosing the same priority for all of the constraints is a correct choice even if it is rarely the best choice.
The learning was carried from the generalisation of 40 pieces of housing estate taken in the surroundings of the town of Orthez (South-west of France).

4.2 Learnt rules

We note S(Ci) to indicate the satisfaction of the constraint Ci and CR to indicate its confidence rate. The rules obtained by the C4.5 algorithm are the detailed hereafter.

4.2.1 Rules of choice of the priority constraint

In this part, rules for the constraints priority were learnt . The learning process for this knowledge is described part 3.3.1.

- Constraint prox_road_bldg:

Rules	Confidence Rate
IF (S(bldg_satisf) > 7) THEN **priority**	CR = 92%
IF (S(bldg_satisf) ≤ 7) AND (S(prox_road_bldg) ≤ 5) THEN **priority**	CR = 83%
IF (S(bldg_satisf) ≤ 7) AND (S(prox_road_bldg) > 6) AND (S(prox_bldg) = 8) THEN **priority**	CR = 71%
IF (S(bldg_satisf) ≤ 7) AND (S(prox_road_bldg) = 7) AND (S(prox_bldg) > 8) THEN **priority**	CR = 86%

- Constraint bldg_satisf:

Rules	Confidence Rate
IF (S(bldg_satisf) ≤ 8) AND (S(prox_road_bldg) > 4) AND (S(prox_bldg) > 5) THEN **priority**	CR = 91%
IF (S(bldg_satisf) > 8) AND (S(prox_road_bldg) > 7) AND (S(prox_bldg) > 5) THEN **priority**	CR = 93%

- Constraint prox_bldg:

Rules	Confidence Rate
IF (S(prox_bldg) ≤ 7) THEN **priority**	CR = 54%
IF (S(bldg_satisf) > 7) AND (S(prox_bldg) > 7) THEN **priority**	CR = 67%

According to these rules, the plan from the constraint bldg_satisf will be applied in priority in most of the cases. This plan consists in triggering the generalisation of buildings of the housing estate. In the very large majority of cases, this plan is carried out only once by path of the tree. We can notice that for a non-dense group of buildings like our housing estates, the fact that the buildings generalisation should be performed in priority had already been noticed by (Duchêne, 2004).

The only case when the plans of the constraint bldg_satisf will not be triggered in priority is when the buildings are already well-generalised and there are few conflicts of proximity between buildings. In this case, the plans of the road/building proximity constraint will be applied in priority.

Once the building individual generalisation of the buildings has been performed, the next stage will be a phase of displacement and removal of buildings. Indeed, the buildings generalisation of a scale of 1:50 000 generally involves their scaling what has often for consequences some overlapping between buildings and between roads and buildings. It is thus often necessary to resort to actions solving this kind of problems.
We can notice that the plan suggested by the constraint prox_bldg (aggregation) which theoretically could also help to solve this problem are in general never suggested in the learnt rules. The system seems to largely prefer the plans suggested by the constraint prox_road_bldg.

4.2.2 Validity criterion for a state

In this part, a rule for the validity criterion was learnt. The learning process for this knowledge is described part 3.3.2.1. The rule obtained is the following:

Rules
IF (S(bldg_satisf) ≤ 9) AND (S(buildings_nb) ≤ 8) THEN **invalid state**

The interpretation of this rule is that the only case when the constraint will consider a state as invalid is the case where the buildings were not generalised yet or were badly generalised, and where at the same time too many buildings have already been removed. This rule seems to be pertinent. Indeed, generalising another time the buildings could involve superposition and conflicts of proximity, and the system cannot allow the agent to remove any more buildings because of the risk to see the satisfaction of the constraint buildings_nb decreasing even more.

4.2.3 Stopping criterion for the actions cycle

In this part, a rule for the stopping criterion was learnt. The learning process for this knowledge is described part 3.3.2.2. The rule obtained is the following:

Rules
IF (S(bldg_satisf) = 10) AND (S(prox_road_bldg) = 10) THEN **stop the actions cycle**

According to this rule, the system considers that it is useless to continue to explore the tree if the constraints bldg_satisf and prox_road_bldg are perfectly satisfied.
To justify that, we can notice that these two constraints are the only ones which really propose effective plans. Moreover, once their maximum satisfaction is reached, the constraints do not propose any more plans. Without this rule, even if the system reaches a state that would perfectly satisfy this two constraints, it will continue to explore this path and/or others paths in the tree even if the chance of finding a better state is very small.

4.3 Tests of learnt knowledge

The process was tested with initial knowledge (with basic validity and stop criteria described in part 3.3.2) and then with the newly learnt knowledge on 40 pieces of housing estate taken in a different zone from the one used for the learning.
The basic criterion for the state validity is that a state is valid if it is not similar, from constraints satisfaction point of view, to an already visited state. The basic criterion is that the actions cycle stops when a perfect state is reached (or when all the possible actions were tested).

The results show an improvement at the same time in terms of efficiency (number of explored states) and of effectiveness (the mean satisfaction obtained by housing estate) (figure 8). The mean number of visited states by housing estate was divided by approximately 5 while the quality of the obtained result slightly improved.

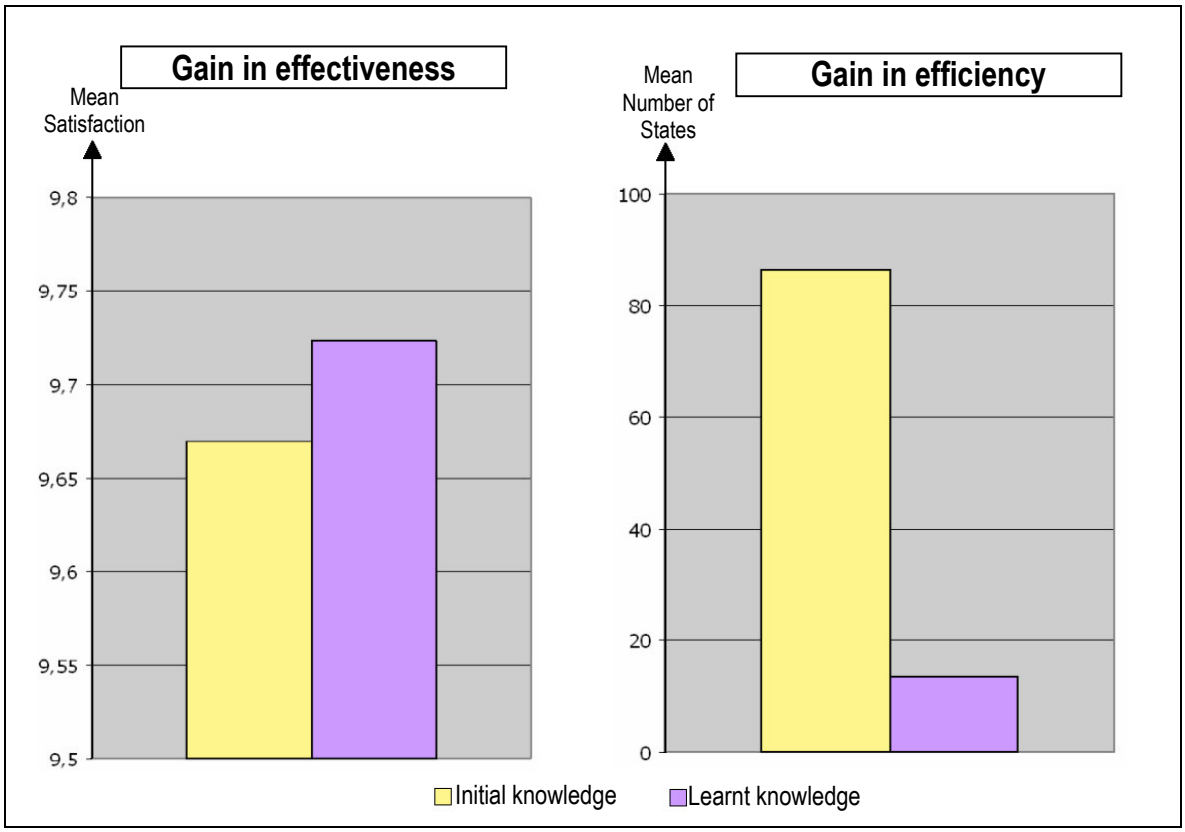


Figure 8 – *Gains in effectiveness and efficiency: comparison results between the initial and the learnt knowledge*

We can observe on figure 9 that the learnt knowledge allows the system to solve some problems of proximity and avoid the problems of large agglomerates generation that are present with the initial knowledge. The problems are particularly visible on figure 9 where the results obtained with the initial knowledge are quite bad.

This application shows that it is possible to distinctly improve the knowledge included in a generalisation system by an introspective process, and thus to validate our general approach.

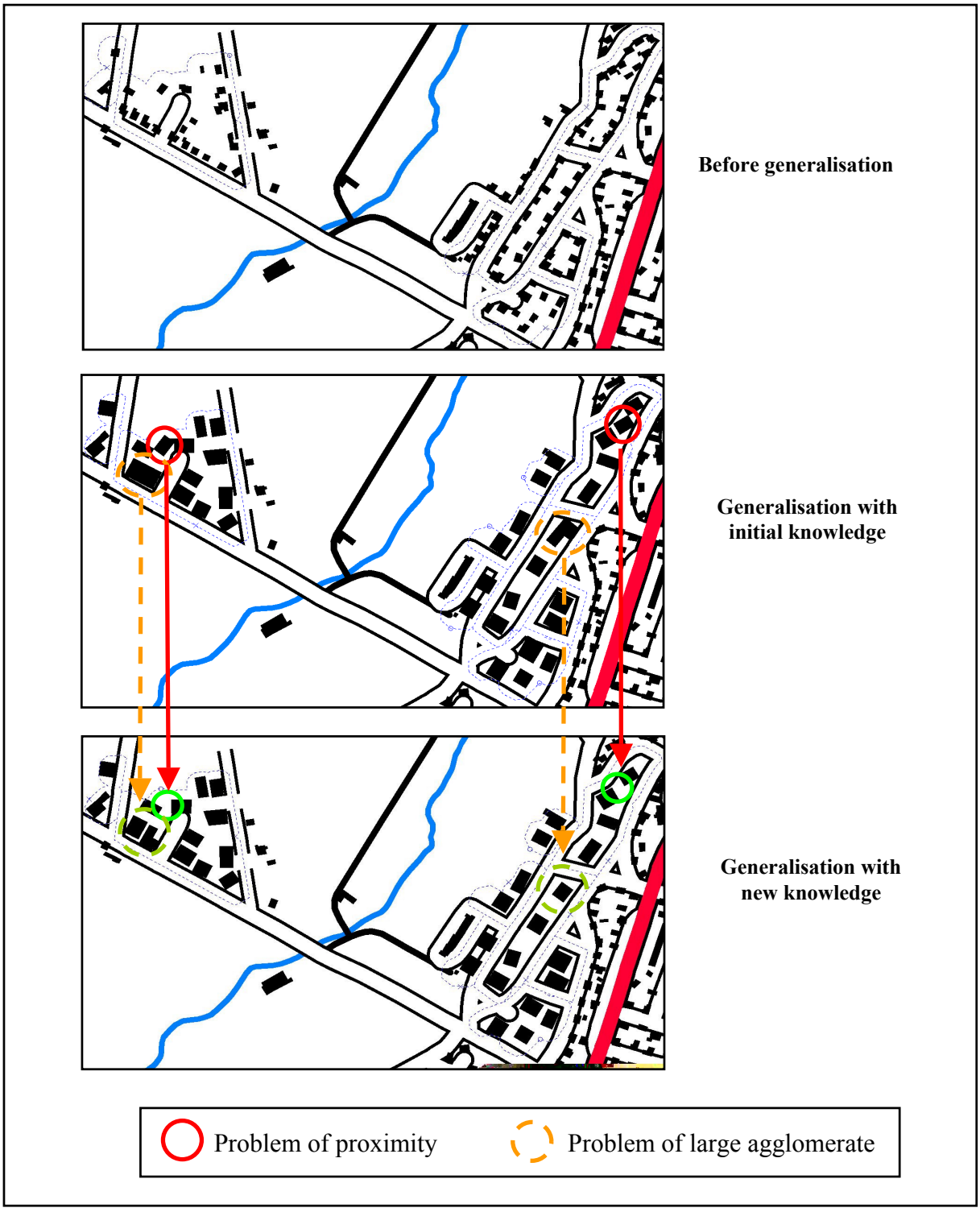


Figure 9 – *Example of results obtained with initial/new knowledge, Area 1*

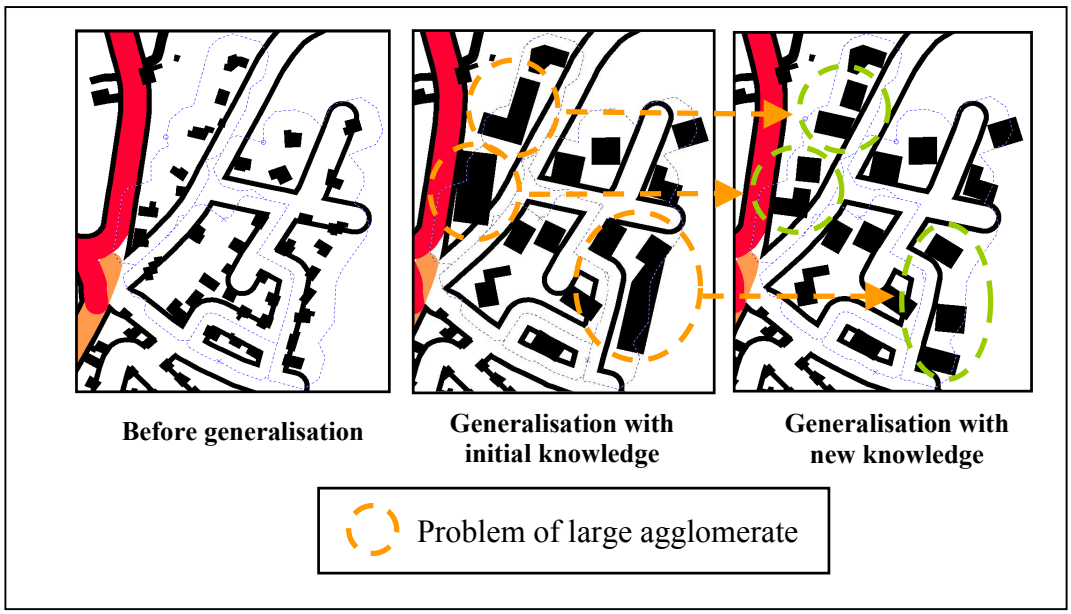


Figure 10 – *Example of results obtained with initial/new knowledge, Area 2*

However, it also arises a problem of this system: the satisfaction indicator is not very good. In fact, some of the results obtained with the initial knowledge are far from being good (in particular, the one obtained in the figure 10) but the mean satisfaction we obtained for the generalisation is very high (close to 9,67). It is true that the problematic generalisations are just a small part of the whole generalisations and that most of the generalisations are good, but the satisfaction must be a better indicator of the cartographic results. It is even more problematic that our learning approach is based on this indicator (by the means of the best state). A better indicator will help the system to learn better.

5 Conclusion and future works

In this paper, we underlined the interest of integrating of an automatic module of knowledge revision module inside a generalisation system based on trials and backtracks.
We proposed a method for procedural knowledge revision based on introspection and machine learning techniques. Within the framework of the AGENT generalisation model, we developed various methods and algorithms allowing acquiring knowledge concerning the order of the applied plans but also concerning the pruning of the built search trees. Application on the generalisation of the housing estate at 1:50 000 validates our general approach and proves the gains in terms of efficiency and effectiveness that it can bring.

This application also arises the problem of the "satisfaction" indicator used to describe the cartographic quality of the result. Our learning approach being based on it, this indicator must be reliable. Further work must be carried out to improve this indicator

We did not seek to acquire in the work presented in this paper the piece of knowledge relating to the choice of the plans suggested by each constraint as well as their associated weight. Further learning methods have to be developed in order to revise it.
Until here, we have just tried, from knowledge initially contained in this system, to acquire new knowledge that we substituted to the old. One of the goals is now to be able to carry out a real knowledge revision, i.e. identify with precision the pieces of knowledge that are bad and to revise only this part of knowledge.

Moving on this kind of precise revision requires having a common formalism for the knowledge expression, as well as structures of storage allowing managing them in a completely automatic way. Thus a work of knowledge modelling is currently being undertaken.

References

BARRAULT M. REGNAULD N., DUCHENE C., HAIRE K.,, BAEIJS C., DEMAZEAU Y., HARDY P., MACKANESS W., RUAS A. & WEIBEL R. (2001). Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalisation. *20th ICC conference*. Vol. 3. p. 2100-2116.

BEARD K. (1991). Constraints on rules formation, map generalization, *Buttenfield B. & McMaster R. (ed)*, Longman, p. 121-135.

BRASSEL K. & WEIBEL R. (1988). A review and conceptual framework of automated map generalization. *International Journal of Geographical Information Systems*. vol. 2, n°3, p. 229-244.

BURGHARDT D. & NEUN M. (2006). Automated sequencing of generalisation services based on collaborative filtering. *4th International Conference GIScience*.

DUCHENE C. (2004). Généralisation cartographique par agents communicants : le modèle CartACom, *Thèse de doctorat,* université Paris VI et laboratoire COGIT.

DUCHÊNE C., Dadou D. & Ruas A. (2005), Helping the capture of expert knowledge to support generalisation. *ICA Workshop on generalization and multiple representation*, La Corona, Spain.

DYEVRE A. (2005). Analyse d'un processus de généralisation cartographique à l'aide d'apprentissage artificiel. *Rapport de Master 2*, Université Paris IV et Laboratoire COGIT.

KILPELÄINEN T. (2000), Knowledge Acquisition for Generalisation Rules. *Cartography and Geographic Information Science*, vol. 27, n°1, p. 41-50.

MCMASTER R.B. & SHEA K.S. (1992). Generalization in Digital Cartography. *Assocoation of American Geographers,* Washington.

MUSTIERE S. (2001). Apprentissage supervisé pour la généralisation cartographique. *Thèse de doctorat*, université Paris VI, laboratoire COGIT.

MUSTIERE S. & DUCHENE C. (2001). Comparison of different approaches to combine road generalisation algorithms: GALBE, AGENT and CartoLearn. *4th ICA Workshop on generalisation*.

MUSTIERE S. & RUAS A. (2004), Vers une réconciliation des experts et des systèmes - Expériences d'utilisation de méthodes d'apprentissage artificiel pour la généralisation et l'intégration des bases de données géographiques. *Journées Cassini*.

QUINLAN, J. R. (1993), C4.5 Programs for Machine Learning*, CA: Morgan Kaufmann*, San Mateo.

REGNAULD N. (2001), Constraint based mechanism to achieve automatic generalisation using an agent modelling, *9th GISRUK conference*.

RIEGER M. & COULSON M. (1993). Consensus or confusion : cartographers' knowledge of generalization. *Cartographica.* vol. 30. n°2-3. pp. 69-80.

RUAS A. (1999). Modèle de généralisation de données géographiques à base de contraintes et d'autonomie. *Thèse de doctorat,* université de Marne La Vallée.

RUAS A. & HOLZAPFEL F. (2003). Automatic characterisation of building alignments by means of expert knowledge. *21th ICC conference*. p. 1604-1616.

RUAS A., DYEVRE A., DUCHENE C. & TAILLANDIER P. (2006). Methods for improving and updating the knowledge of a generalization system, *Autocarto*. Portland USA.

WEIBEL R., KELLER S. & REICHENBACHER T. (1995). Overcoming the Knowledge Acquisition Bottleneck in Map Generalization : the Role of Interactive Systems and Computational Intelligence. *Proceeding of the 2nd COSIT conference*. p. 139-156.

WEISS G. (1999). Multiagent Systems. A modern Approach to Distributed Artificial Intelligence. *The MIT Press.*