Methods for Improving and Updating the Knowledge of a Generalization System

Anne Ruas, Aurélie Dyevre, Cécile Duchêne, Patrick Taillandier
Laboratoire COGIT – IGN France
2 avenue Pasteur
94165 Saint Mandé - FRANCE
anne.ruas@ign.fr, cecile.duchene@ign.fr; patrick.taillandier@ign.fr
33 1 43 98 84 32 – fax : 3 1 43 98 81 85

Abstract: In this paper we present a method to improve and to update the knowledge used for the automation of the generalization of buildings based on agent paradigm. We propose to store 1/ each building decision, 2/ the reason why the decision was taken (the conflicts) 3/ the result of each algorithm (an improvement or not) and 4/ the successful process chain within all trials. At the end, the processes of all buildings are compared in order to identify the weakness (for example the case where a specific algorithm is often used but never succeeds). When a deficiency is identified we introduce new rules and we study the effect of this change on the efficiency of the process. It can be used either to improve existing knowledge or to introduce new rules associate to the use of a new measure or a new algorithm. The first study has been made on building independent generalization to set the learning methodology. We wish now to apply it on more complex cases such as contextual generalization which still needs knowledge improvement.

Keywords: Generalization, Learning techniques, Agent paradigm

1. Improving the automation of generalisation process

The complexity of the generalization process is well known in GIS community. However for 20 years, important progress has been made thanks to the intense use of physical models and artificial intelligence techniques. As a result for small scale changes– called graphic generalization- the automation is successful. When generalization is equivalent to a space distortion–a set of displacements and object emphasizing – some robust solutions already exist. These methods, based on strength computation, use known solving methods such as the finite elements (Hojholt 2000) or the least square method (Sester 2000, Harrie and Sarjakoski 2002) to adequately move and stretch the objects according to size and distance constraints.
However for larger scale changes, non continuous operations such as object removal or aggregation are required. For such generalizations, a single and recursive method does not yet exist. So we have to apply a set of different algorithms one after the other, and it is not possible to foresee the sequence of the appropriate algorithms. As a consequence the remaining difficulty is in the automation of the choice of appropriate algorithms during the process (how to generalize?). To do so, two types of solving methods exist: one is mainly based on random choice of operation and evaluation by means of a cost function and the other is based on knowledge to choose the appropriate operation at each step and on an evaluation to assess each choice:

- for the stochastic processes – using simulated annealing or genetic algorithms (Ware et al 2003)-, a very large number of operations are tried and the convergence towards a good solution strongly depends on the function of evaluation that computes the quality of each proposed solution,

- for the knowledge based processes (see section 2.1) the convergence towards a good solution strongly depends on the quality of the procedural knowledge to choose the appropriate algorithm according to the properties of local situation.

Both types of solving process are based on knowledge and evaluation. This knowledge ensures or limits the convergence towards a good solution. To describe the quality of the convergence two criteria are fundamental: the efficiency and the effectiveness. We can say that a generalization system is performing if it converges quickly to a good solution (figure 1).

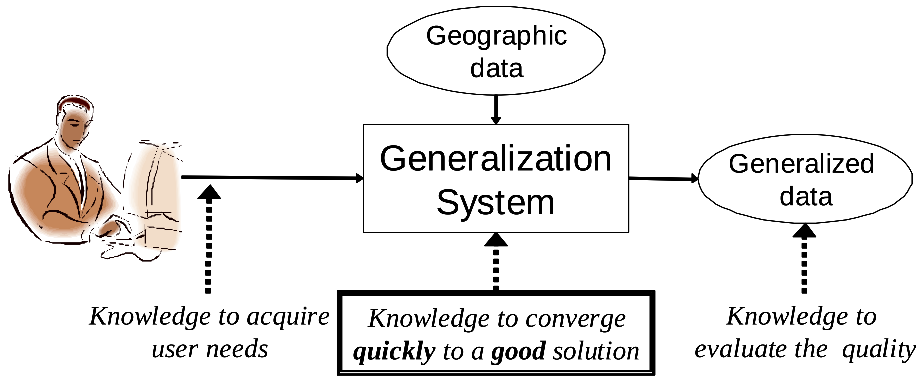

Geographic data
Generalization System
Generalized data
Knowledge to acquire user needs
Knowledge to converge **quickly** to a **good** solution
Knowledge to evaluate the quality

Figure 1. Knowledge to converge to generalized data

Classically speaking, knowledge is based on reasoning and experiments. The reasoning gives hypothesis (by deduction) assessed and improved by experiences and reversely experiences give rules (by induction) formalized and improved by means on reasoning. The aim of this research is to propose methods in order to improve the knowledge contains in a generalization system by means of experiments. In section 2 we present the generalization system we are using and the previous research work related to knowledge in the field of generalization. In section 3 we present our learning techniques approach and in section 4 we present our first results.

2. Context of our research

2.1. From Multi Agent System Paradigm to Clarity

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors (Russell and Norvig, 2003). An agent can be thought of as an object that has a goal and acts autonomously in order to reach this goal thanks to capacities of perception, deliberation, action, and possibly communication with other agents (Weiss 1999, 32).

Our proposal in 1998 is to model geographical objects as agents. Each agent is able to perceive and evaluate its current state, and to choose and apply to itself generalisation algorithms to improve this state (Ruas 1999). The cartographic agents (such as a building object, a road object) have the goal to generalise themselves (individually and all together) in the most successful way. In this model, the specifications (such as the minimum size or the minimum distance) are represented by means of constraints. These constraints guide each object for its own generalization. Group constraints (such as density) are linked to group objects named meso objects. A meso object is composed of objects. It generalizes itself by means of contextual operations (such as object removal) and is guided in its decision by its own constraints (Ruas 1999).

This idea has been first implemented by Ruas on a prototype named Stratège at the COGIT laboratory (1997; 1998). It has been recoded and enriched during the AGENT European

Project onto Lamps2 GIS (Barrault et al 2001). A first version was proposed by laser-scan in 2001 at the end of the project. Then – with the help of a consortium of European National Mapping Agency named MAGNET – Laser-Scan Ltd proposed an improved version named Clarity^{TM}. Study is on going at the IGN-France through a production project to adapt the system to the production of 1: 50 000 scale map form the IGN-France BDTopo © (see figure 2).

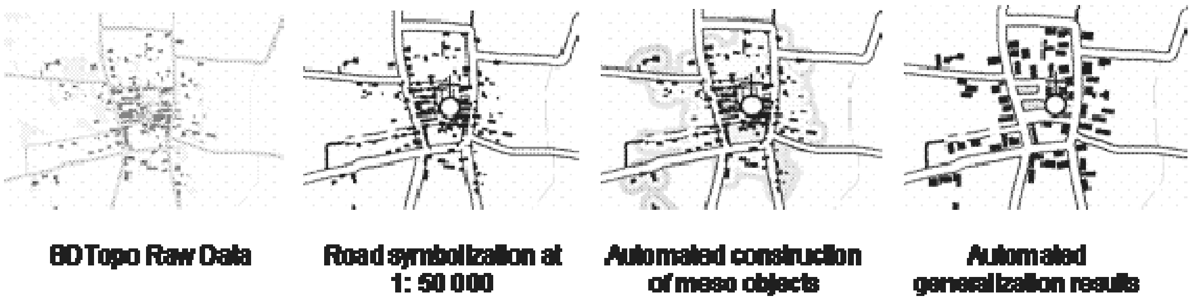

BDTopo Raw Data
Road symbolization at 1: 50 000
Automated construction of meso objects
Automated generalization results

Figure 2. Automated generalization at 1: 50 000 on small data set

More details can be found in (Ruas and Duchêne 2006) which focus on the principles of the agent engine and includes the enrichment proposed by Duchêne (2004) during her PhD.

2.2. Previous work on Knowledge acquisition in the field of generalization

A good way to ensure the relevance of the knowledge used in a system is to collect the necessary knowledge from experts of the domain. However, collecting knowledge from experts and formalizing it in a way that is usable by the system is problematic. This is classically identified in Artificial Intelligence as "the knowledge acquisition bottleneck", and this bottleneck has proved to be existing in the domain of generalization too (Rieger and Coulson 1993; Weibel et al. 1995; Kilpeläinen 2000).

To overcome this bottleneck, many works have tried to use supervised machine learning in order to build rules from examples, be it for generalization (Weibel et al. 1995; Mustière 2001), data enrichment (Plazanet et al. 1998; Sester 1998) or system calibration (Hubert and Ruas 2003). These works show that the use of supervised learning seems to be interesting for those purposes, but they also exhibit two main difficulties related to it. The first difficulty, identified for example in (Mustière and Ruas 2004), is related to methodology: no machine learning algorithm is "magic" in itself, and the most important part of the work is to formalize what is to be learnt and under which form. The second difficulty, namely pointed out in (Mustière 2001; Ruas and Holzapfel 2003), is practical: collecting the examples is always long and fastidious, because it requires to use several systems that are not interrelated: for instance, a GIS to choose the examples, a paper with screen captures labeled with object identifiers to collect the assessment of the experts on each example, an Excel file to summarize the experts assessments, and a machine learning software to induce rules. (Mustière and Ruas 2004) recommend that GIS should provide integrated environments to perform the tasks of expert knowledge acquisition and analysis. They identify the following needed functionalities:

– easy creation and instantiation of example data sets extracted from existing geo data bases,
– easy creation and instantiation of an expert knowledge data base to collect the expert's assessments on presented examples,
– generation of interfaces that populate the examples, ask the questions to the experts and collect their answers,
– choice of a machine learning method among several classical methods provided, and easy enrichment of the library of machine learning methods,

- triggering of the chosen machine learning method on assessed examples,
- visualisation of the results of learning in order to be able to validate or invalidate each learnt piece of knowledge.

(Duchêne et al 2005) presented a prototype – named MAACOL – based on these very principles and that is used to acquire expert knowledge. It has been used to normalize the properties of building. The paper presents the calibration of the measure of wall straightness regularity for buildings to illustrate the learning method.

Here, we propose another approach to improve the procedural knowledge based on experiments: we propose first to trace the generalization of a large set of objects and then to analyze these processes in order to detect the repetitive errors that could be avoid.

3. Our Learning Approach

3.1. What is learnt? Why?

In the system, an agent generalizes itself by means of rules that depend on the constraints violation. We commonly say that an agent has a *life cycle* during which it applies to itself a set of generalization operations in order to reach a perfect state (if possible) or at least a good one. During one cycle, an agent evaluates itself one time to find a set of alternative solutions to generalize itself ('plans for self generalization') and then another time to validate or not each solution tried. If its state is not better, it does not validate this solution and tries another one (see figure 3). When one solution is validated, if the state is perfect its generalization is finished, if not it carries on its generalization.

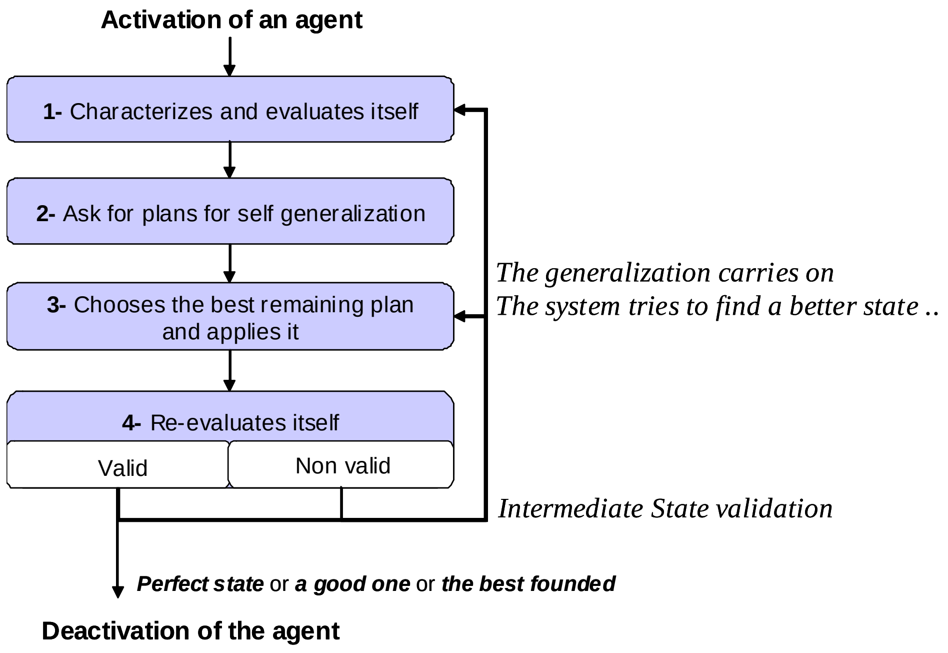

Activation of an agent
1- Characterizes and evaluates itself
2- Ask for plans for self generalization
3- Chooses the best remaining plan and applies it
4- Re-evaluates itself
Valid
Non valid
The generalization carries on
The system tries to find a better state ..
Intermediate State validation
Perfect state or a good one or the best founded
Deactivation of the agent

Figure 3. Simplified view on the generalization process of each agent

An agent that generalizes itself searches for its ideal sequence of generalization. The sequence changes according to the agent characteristic (for a building its size, its shape, its squareness) and its goal (the constraints on size, shape, orientation it should respect). The knowledge contained in the system is used to propose to the agent the best search tree. As the proposals depend on the agents' characteristics and goals, all agents of the same type do not have the same tree. The tree is computed on the fly during the process.

In figure 4, when an agent is at a specific node (a state with a certain level of happiness) the plans returns a list of ordered plans (in figure 4 the list is (algo1, algo2, algo3)). As the search strategy is the depth first, the agent next action in figure 4 is to first try the solution ‘algo1’.

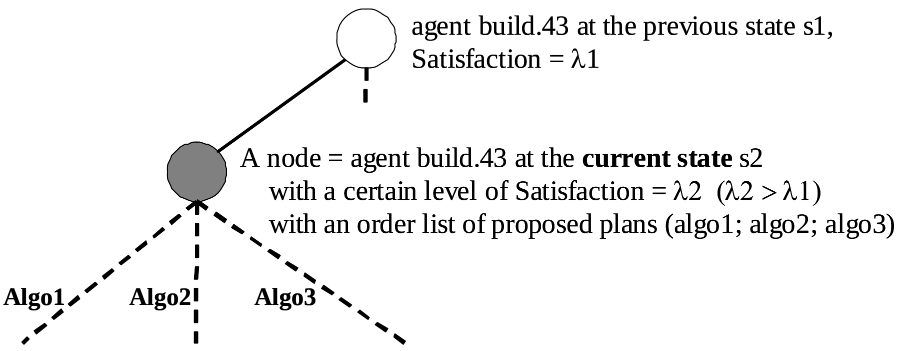

agent build.43 at the previous state s1,
Satisfaction = λ1
A node = agent build.43 at the **current state** s2
with a certain level of Satisfaction = λ2 (λ2 > λ1)
with an order list of proposed plans (algo1; algo2; algo3)
Algo1
Algo2
Algo3

Figure 4. An agent search tree

“The *depth-first search* always expands the deepest node in the current fringe of the search tree. The search proceeds immediately to the deepest level of the search tree, where the nodes have no successors. As those nodes are expanded, they are dropped from the fringe, so that the such backs up to the next shallowest node that still has unexplored successors (Russell & Norvig, 2003, 75)”. The process stops as soon as the agent reaches a good level of happiness (its satisfaction). If the requirement is very high, the agent never stops: it tries its entire tree, even if the best solutions are supposed to be the first solution tested.
In order to avoid long and useless trials, during the AGENT project, Nicolas Regnauld (Regnauld 2001) proposed to use the Hill-Climbing mechanism in order optimize the search. Whereas the depth-first search strategy can explore all branches of a tree to obtain the required minimum value, the Hill Climbing expends a branch only if the state is as good as the best previous recorded state. “The Hill Climbing search algorithm is simply a loop that continually moves in the direction of increasing value – that is uphill. It terminates when it reaches a peak where no neighbour has a higher value” (Russell & Norvig, 2003,111).
Figure 5 illustrates the search tree of a building. It first applied an algorithm of ‘dilation’ which improved its state (from S= 5 to S = 5.94). Then it tried a ‘Squaring’ which has not been validated because it did not improve its state. It tried a ‘Simplification’ which improved its state (from S = 5.94 to S=8.58). As the state was not yet perfect, the building tried a ‘Squaring’ that allows it to reach a perfect state (S = 10). This building reached a perfect state by means of a sequence of 3 algorithms (dilation, simplification and squaring) whereas it tried 4 algorithms, only one trial was unnecessary.

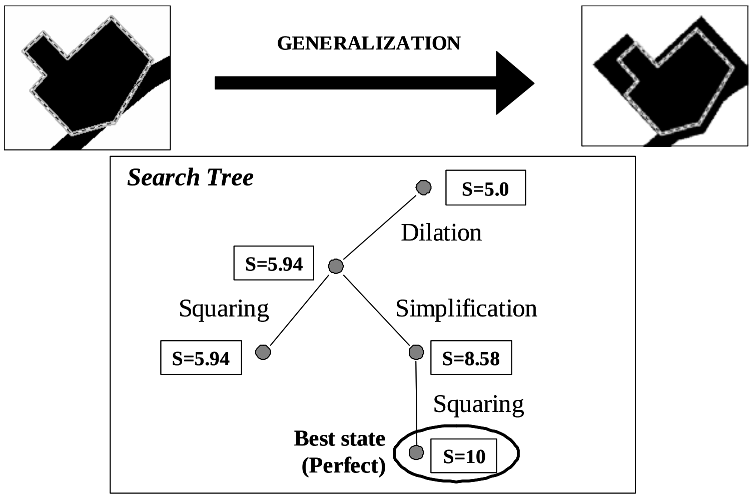


Figure 5. A building search tree

What is learnt, why ?

The aim of our research is to improve the knowledge used by the agents to generalized themselves (task 2 in figure 3) in order to improve the efficiency of the generalization, i.e. to avoid as much as possible useless trials. If we refer to the agent search tree, we want the knowledge to propose the best algorithm in the first order. In figure 5 it would mean at the second step to try *Simplification* before *Squaring* to directly converge to a good state.

3.2. How to learn?

Several learning techniques exist. In our case, we already have knowledge built from reasoning and previous experiences (in particular tests to tune the algorithm order). We want to improve this knowledge in order to reach a higher level of automation. Two learning techniques can be used, *Explanation based learning* and *Reinforcement based learning:*

- Explanation based learning (ELB) is a method for extracting general rules from individual observations. In our system that would mean that we would perform several generalisations and detect cases of choices of algorithm that give good result. We add these rules to the agents' knowledge base so instead of using its knowledge base, it first applies these new rules. If the agent is in a situation described in the new rules, it applies the proposed algorithm in order to reach a good state in a quicker way. Case based reasoning is a specific case of ELB based on analogy. Instead of generating new rules generated by induction from examples, it consists in adding successful sequences. Then during the generalisation you compute a degree of similarity between the agents to generalise and these successful cases. If an agent looks like a recorded case, it will use a recorded sequence instead of its own generalisation engine.
- Reinforcement learning consists in awarding good decisions and to give negative rewards for bad decisions. In such a case, the good knowledge (the appropriate rules) is reinforced whereas the bad one is minimized. The Q-Learning (Watkins 1989) is often used to perform learning by reinforcement.

The reinforcement based learning is the most adequate because we do not wish to extend the rule base but to improve it. On the other way round, ELB is easier to begin with because it does not need to remodel the system as you just add new rules on top of the existing ones. For our first study we used ELB techniques but we will try reinforcement techniques in a near future.

Exploration, Analysis and Exploitation:

Whatever the learning techniques, we distinguish three learning phases:

- The **exploration** phase consists in making lots of generalization trials using the initial knowledge. The agents generalise themselves, creating their own search trees as the one presented in figure 5. During this step there is no knowledge improvement. All decisions are stored including the characteristic of the agent and the result of the application of each algorithm: the success (the algorithm improved the state of an agent) or the failure (the algorithm did not improve the state of the agent.).
- The **analysis** phase consists in generating new rules from the statistical analysis of the exploration phase. The new rules are expressed in a symbolic and readable manner, so they are checked by experts for validation.
- The **exploitation** phase consists in testing if these new rules improve or not the convergence of the system. To do so, generalisation process is performed twice on new objects: the first time with the initial rules, the second one with the new rules. If the convergence is better and faster, the new rules are confirmed. As a consequence we need to compute indicators of efficiency and effectiveness to compare both rule bases.

3.3. First testing the method on well known cases

Our objective is to first elaborate a learning process on simple and well known cases and to extend this learning process on more complex generalisation cases. Thus we have chosen to generalize independent buildings and we will extend the method to urban blocks which are more complex because they require contextual algorithms such as object removal and displacement that are very time consuming.
The buildings are generalized on Clarity^{TM} using the agent engine and the building knowledge based. This generalisation is normally very good because it has intensively been studied during the AGENT project and after the project by the Jenny Trevisan at IGN-France and Nicolas Regnauld at the Ordnance Survey. The test is also used to check if this knowledge is as good as we thought.

4. Implementation and first results

4.1. The stored information

During the exploration phase, we created a file which stores: The name of the agent, All its successive states, each applied algorithm, if the algorithm made a backtracking or not, each level of satisfaction. This file is used to build the search tree of each building and also to group common cases together.
In table 1, the line records one building generalisation step. The building has conflicts of size, granularity and squareness but no concavity conflict. As solving the Size constraint was the priority, the first algorithm tried was ‘enlarge-to-rectangle’. This algorithm improved the state of the agent. We also recorded that this trial belonged to the best chain.

Satisfaction (from 0 to 10- excellent)				Highest priority	level	Algorithm name	Success ?	Best chain
Size	Granu.	Square	Conc.					
3	1	2	10	Size	1	Enlarge-to-rectangle	Yes	Yes

Table 1. one step of generalisation

Rules computation
Identical cases are grouped together. Cases are identical if the level of satisfaction of the constraints are the same. We sort the table 1 by constraint satisfaction and we see if some algorithms that have a high priority level failed and reversely if some algorithm that have a low priority level succeeded. In such a case we add new rules on top of the others such as :

If [satisfaction.constraint(A) =$\lambda 1$ and satisfaction.constraint(B)= $\lambda 2$, and…]
then use algorithm$_i$

In other words, if a building is in a recognised situation related to its level of satisfaction for its four constraints, it will first use the algorithm proposed by the new rules instead prior to its the previous rules.

Computation of Efficiency and Effectiveness
To compute the effectiveness, we compute the ratio of buildings that reach a perfect level of satisfaction (S= 10). To compute the efficacy, we compute the average number of algorithms tried per building and the average number of useful algorithms per building.

4.2. Results on buildings independent generalization

We distinguish two data sets: one composed of 412 buildings that has been used to detect new rules for the exploration phase, and another one composed of 165 buildings for the exploitation phase. In the following we give some results, more results can be found in (Dyevre 2005).

Exploration phase.

We first made an analysis of success and failure in the use of our algorithm to improve the building satisfaction. We noticed for example that the algorithm that simplify a building to a rectangle nearly never improve the building state (ratio = 7,25% in table 2).

	number_used	**nb_positive_used**	**Quality Ratio**
_polygon_squaring	302	220	72.85%
_polygon_enlarge_to_rectangle	234	195	83.33%
_polygon_simplify	189	142	75.13%
_polygon_scale	286	278	97.20%
_polygon_simplify_to_rectangle	69	5	7.25%

Table 2. Analysis of algorithm efficiency

Then we builtnew rules such as *if (x,y,z,t) then use this algorithm* where (x,y,z,t) is the vector of constraint satisfaction for the constraint of size, squareness, concavity and granularity. We found the following rules:

- If vector of satisfaction = ((6,2,3,10) or (6,2,5,10) or (6,2,10,8) or (6,2,10,10) or (6,4,10,8) or (6,4,10,10) or (6,7,10,10)) then use 'polygon-scale' first
- If vector of satisfaction = ((10,2,10,1) or (10,7,10,1) or (10,7,10,5) or (10,10, 10, 3)) then use 'polygon_simplify' first
- If vector of satisfaction = ((10,4,10,8)) use 'squaring' first

Exploitation phase 1: analysis of the initial knowledge

Before beginning introducing the new rules, we first computed the quality of the initial knowledge. We used the clarity implemented search strategy (the Hill Climbing) which does not investigate the entire tree but only branches that are better than the last best recorded state.

The data set is composed of 165 buildings:

- 146 building reached a perfect state (S=10), the effectiveness is 88,5%.
- The average number of algorithms per building is 2.63, the average useful number is 1.74, average of algorithms tested after the retained solution: 0.54. The best retained chain is often the first tested (the left side of the tree)

These numbers show that the initial knowledge is very good.

The capacity of representing the search tree allows to precisely analysing the convergence case by case. When the level of satisfaction is not perfect (S <10), the buildings try other solutions to improve themselves as illustrated in figure 6. The hill climbing mechanism in such a case avoids testing too many solutions. In the following example the building tried 6 algorithms whereas the two first were the best. Without the hill climbing it would have tested much more cases.

This also illustrates the impact of the computation of the global satisfaction on the speed of the convergence: the more severe the evaluation function, the better final quality but also the slower the convergence. Focus on bad results illustrated on table 3 helps to understand the level of required quality contained in the evaluation function.

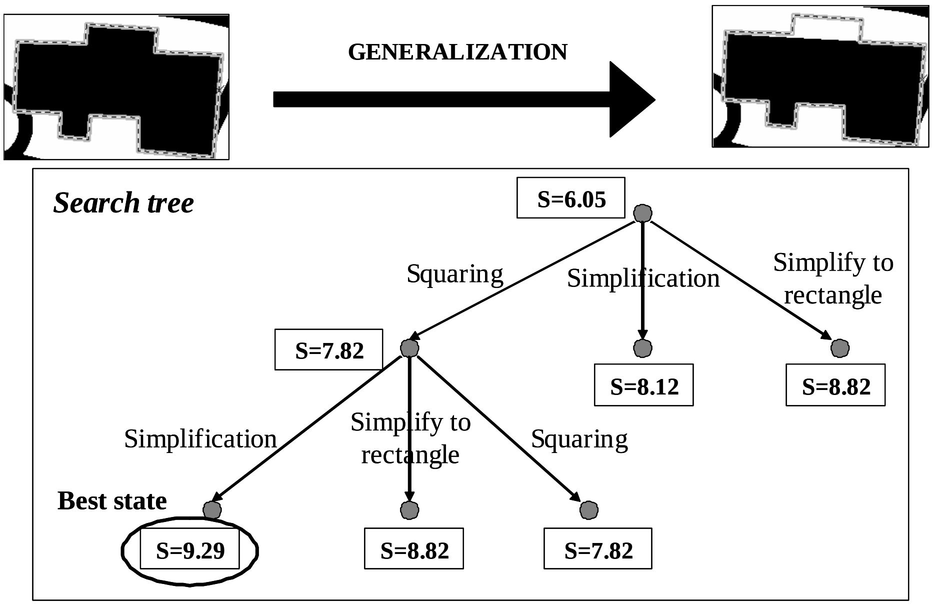


Figure 6. Illustration of the implemented search strategy

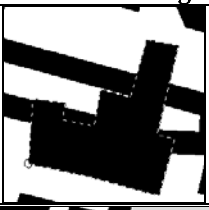
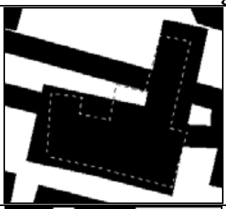
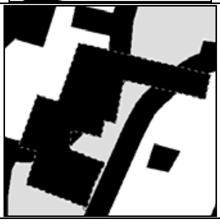
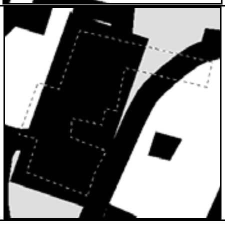
Initial Building	Generalised Building	Final level of satisfaction
		S = 9,28
		S = 7,88 *The shape is not very well preserved.*

Table 3. Example of non perfect results

Exploitation phase 2: analysis of the initial knowledge:

The new rules have been added to the initial ones. The engine has been slightly adapted. During step 2 of figure 3, if the building is in one of the situation described in one of these new rules, it adds the related algorithm on the top of the list computed by the constraints.

The generalisation is triggered on the same 165 buildings, and quality criteria are computed:

- 146 building reached a perfect state (S=10), the effectiveness is 88,5%. The result has not changed at all.
- The average number of algorithms per building is 2.73 (previously 2.63), the average useful number is 1.84 (previously 1.74), average of algorithms tested after the retained solution: 0.54.

These numbers show that the initial knowledge is slightly better than the new proposed one. It also shows that adding a new algorithm on top of the list of computed one is not an appropriate method because it slows down the process as the algorithm might be tested twice.

Checking the priority value :

We also tried to randomly change the priority of treatment (see table 1) which define which constraint should be solved first. When we compute the average number of algorithm per

building we obtained results around 3.11, 3.97, 3.88 whereas the default value is 2.63. We notice that we always obtain worst results than the initial one.
Here again we noticed that the initial knowledge was very well tuned.

5. Conclusion: towards learning agents?

The aim of the research was to propose a learning process to improve the knowledge of a generalization system. We have chosen to use Explanation Based-Learning approach that creates new rules form experiments. We decomposed the learning into three steps: an exploration step that traces the agents self generalization, an analysis step that build rules from repetitive success cases and an exploitation step that checks the improvement of the generalization on new cases. In order to set the method, we have chosen to analyze the knowledge used for building generalization in Clarity^{TM}. This knowledge have been intensively studied during the AGENT project and after at the IGN-France and at the Ordnance Survey. We noticed that this knowledge is very good and efficient as the building agents converge quickly to very good solutions, results that we suspected but that we could not check very easily before this study.
After Aurelie Dyevre study (2005), Patrick Taillandier, a new COGIT PhD student, started to extend this method for building block generalization. This case is very critical because it requires heavy analytical structures such as Delaunay Triangulation. As a consequence each bad decision dramatically slows down the process (and fills the memory). Two strategies of learning are investigated. The first is based on case based-reasoning to add a sequence of successful algorithms that avoids the classical agent engine, the second strategy is based on reinforcement learning techniques. Learning agents are under study.

To conclude we would insist on the necessity of including learning techniques inside complex solving methods such as generalization in order to allow the evolution of such system. It is all the most important today where more and more algorithms are shared through the web. We should be able to easily adapt knowledge based as soon as new and better algorithms appear.

REFERENCES

Barrault M., N., Regnauld, C. Duchene, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas and R. Weibel. 2001. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. *Proceedings 20th International Cartographic Conference*, Beijing, China, 6-10 August, 2210-2216

Duchêne C. 2004. The CartACom model : a generalisation model for taking relational constraints into account. 6th *ICA Workshop on progress in automated map generalisation,* Leicester, 2004, available on the web site of the ICA commission for generalisation : http://aci.ign.fr/Leicester/paper/duchene-v2-ICAWorkshop.pdf

Duchêne C., M. Dadou, A. Ruas. 2005, Helping the capture of expert knowledge to support generalisation - *ICA workshop on generalisation and multiple representation,* La Corona, Spain, 2005 http://ica.ign.fr

Dyevre A. 2005. *Analyse d'un processus de généralisation cartographique à l'aide d'apprentissage automatique* Master Dissertation. Paris VI University and COGIT Laboratory.

Harrie L. and T. Sarjakoski. 2002. Simultaneous graphic generalization of vector data sets *GeoInformatica* ,Vol. 6, N° 3, 233-262

Hojholt P. 2000. Solving Space Conflicts in Map Generalization: Using a Finite Element Method. *Cartography and Geographic Information Science*, Vol. 27, No. 1, pp. 65-73.

Hubert F. and A. Ruas. 2003. A method based on samples to capture user needs for generalisation. 5th *ICA Workshop on progress in automated map generalisation,* Paris, 2003, available on the former web site of the ICA commission for generalisation : http://www.geo.unizh.ch/ICA/docs/paris2003/papers03.html

Kilpeläinen T. 2000. Knowledge Acquisition for Generalization Rules. *Cartography and Geographic Information Science*, vol.27, n°1, 2000, pp.41-50..

Mustière S. 2001. *Apprentissage supervisé pour la généralisation cartographique*. PhD Thesis, University of Paris VI, COGIT Laboratory.

Mustière S. and Ruas A. 2004. Vers une réconciliation des experts et des sytèmes In *Actes des 7èmes Journées Cassini*, Grenoble, France, 2004, p.47-52.

Plazanet C., N. Martini Bigolin and A. Ruas. 1998. Experiments with Learning Techniques for Spatial Model Enrichment and Line Generalization. *GeoInformatica,* 2(4), dec. 98, pp.315-333.

Regnauld N., 2001. Constraint Based Mechanism to Achieve Automatic Generalisation using agent modelling. *Proceedings GIS Research in the UK (GISRUK) 2001*, University of Glamorgan (UK).

Rieger M. & Coulson M. 1993. Consensus or confusion: cartographers' knowledge of generalization. *Cartographical*, vol.30, n°2-3, 1993, pp.69-80.

Ruas A. 1999. *Modèle de généralisation de données géographiques à base de contraintes et d'autonomie.* PhD Thesis, University of Marne-la-Vallée, COGIT Laboratory.

Ruas A. and Holzapfel F. 2003. Automatic characterisation of building alignments by means of expert knowledge. In *Proceedings of the 21th International Cartographic Conference,* Durban, 2003, pp. 1604-1616.

Ruas A. and C. Duchêne. 2006. A prototype of Generalisation based on multi agent system paradigm. In Mackaness, Ruas and Sarjakoski *Generalisation of Geographic information: Models and applications*. Elsevier.

Russell S. and P. Norvig . 2003. *Artificial Intelligence : A modern Approach*. Second Edition. Prentice Hall.

Sester M. 1998. Interpretation of Spatial Data Bases using Machine Learning Techniques. *Proc. of 8th International Symposium on Spatial Data Handling,* Vancouver, pp. 88-97

Sester M., 2000. Generalization Based on Least-squares Adjustment. *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXIII, Part B4, Amsterdam, pp. 931-938.

Ware J.M., C.B. Jones, and N. Thomas. 2003. Automated Map Generalization with Multiple Operators: A Simulated Annealing Approach. *International Journal of Geographical Information Science*, **17**(8): 743-769.

Watkins C.J. 1989. *Models of Delayed Reinforcement Learning*. PhD thesis, Psychology Department, Cambridge university, UK.

Weibel R., S. Keller and T. Reichenbacher. 1995. Overcoming the Knowledge Acquisition Bottleneck in Map Generalization : the Role of Interactive Systems and Computational Intelligence. In *Proceedings of the 2nd International Conference on Spatial Information Theory (COSIT'95)*, p. 139-156.

Weiss G. 1999. *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. The MIT Press.