Automatic revision of the control knowledge used by trial and error methods: Application to cartographic generalisation

Patrick Taillandier, Cécile Duchêne, Alexis Drogoul

▶ **To cite this version:**

Patrick Taillandier, Cécile Duchêne, Alexis Drogoul. Automatic revision of the control knowledge used by trial and error methods: Application to cartographic generalisation. Applied Soft Computing, 2011, 11 (2), pp.2818-2832. 10.1016/j.asoc.2010.11.012 . hal-00594311

HAL Id: hal-00594311

https://hal.sorbonne-universite.fr/hal-00594311

Submitted on 29 May 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Automatic Revision of the Control Knowledge used by Trial and Error Methods : Application to Cartographic Generalisation

Patrick Taillandier[a,b,c], Cécile Duchêne[a], Alexis Drogoul[b,c,d]

[a] *IGN, COGIT, 2/4 avenue Pasteur, 94165 Saint-Mandé, France*
[b] *IRD, UMI UMMISCO 209, 32 avenue Henri Varagnat, 93143 Bondy, France*
[c] *IFI, MSI, UMI 209, ngo 42 Ta Quang Buu, Ha Noi, Viet Nam*
[d] *UPMC, UMI 209, 4 place Jussieu,75252 Paris, France*

Abstract

Humans frequently have to face complex problems. A classical approach to solve them is to search the solution by means of a trial and error method. This approach is often used with success by artificial systems. However, when facing highly complex problems, it becomes necessary to introduce control knowledge (heuristics) in order to limit the number of trials needed to find the optimal solution. Unfortunately, acquiring and maintaining such knowledge can be fastidious. In this paper, we propose an automatic knowledge revision approach for systems based on a trial and error method. Our approach allows to revise the knowledge off-line by means of experiments. It is based on the analysis of solved instances of the considered problem and on the exploration of the knowledge space. Indeed, we formulate the revision problem as a search problem: we search the knowledge set that maximises the performances of the system on a sample of problem instances. Our knowledge revision approach has been implemented for a real-world industrial application: automated cartographic generalisation, a complex task of the cartography domain. In this implementation, we demonstrate that our approach improves the quality of the knowledge and thus the performance of the system.

Key words: Knowledge Revision, Problem Solving, Trial and Error Method, Cartographic Generalisation

Email addresses: patrick.taillandier@gmail.com (Patrick Taillandier), cecile.duchene@ign.fr (Cécile Duchêne), drogoul@bondy.ird.fr (Alexis Drogoul)

1. Introduction

Since the beginning of Artificial Intelligence, researchers have drawn their inspiration from human behavioral and reasoning capabilities. In particular, the classical "trial and error" resolution methods exhibited by human beings when trying to solve a problem have been adapted and used very early in artificial systems. These methods, combined with the increasing speed of computers, have permitted to solve numerous problems, especially optimisation problems. However, the more complex the problems become, the more numerous the number of trials necessary to find an optimal solution are, and this increase is in most cases exponential. In order to limit the number of trials, one strategy is to provide the system with relevant knowledge about the problem instead or relying on a brute force approach. Indeed, this knowledge, called control knowledge or heuristics, can allow, when relevant, to intelligently guide the system toward the optimal solution and avoid useless trials. Unfortunately, acquiring such knowledge, generally from experts, is a difficult task. Eward Feigenbaum described this issue in 1977 as the knowledge acquisition bottleneck problem [1]. Indeed, the expert knowledge is seldom formalised and its translation into a formalism usable by computers can be very complex. Another issue concerns the evolution of this knowledge. Actually, as new elements (i.e. for instance, new actions to try) are integrated in the system, it is necessary to update the existing knowledge in order to take these elements into account. This update can be a complex process, which requires the appraisal of an expert. It is thus of primary importance to propose methods that would allow a knowledge-based system to revise its knowledge by itself. In this paper, we propose a generic knowledge revision approach dedicated to optimisation problem solver systems based on a specific trial and error approach: the informed tree search strategy. Our approach allows the system to revise its knowledge off-line by means of an exploration of the set of possible values that can be taken by its knowledge (the knowledge space). We propose a specialisation of our approach for knowledge expressed by production rules. In Section 2, we introduce the general context in which our work takes place and the difficulties that we had to face. Section 3 is devoted to the presentation of our knowledge revision approach. Section 4 describes an application of our approach to cartographic generalisation tasks. In this context, a real experiment we carried out, as well as its results, are

presented. Section 5 offers some conclusions and perspectives.

2. Context

2.1. Optimisation problems and solving systems considered

2.1.1. Description of considered optimisation problems

Many real world problems can be expressed as optimisation problems. In this type of problems, the goal of the system is to find, among all possible solutions, the one that maximises an evaluation function. In this paper, we are interested in a family of optimisation problems that consists in finding, by application of actions, the state of an entity that maximises an evaluation function. We formalise this kind of optimisation problems as follows. P is an optimisation problem that is characterised by:

- E_P: A class of entities
- $action_P$: a set of actions that can be applied to an entity belonging to E_P. The result of the application of an action is supposed to be non-predictable.
- Q_P: a function that defines the quality of the state of an entity belonging to E_P

An instance p of P is defined by an entity e_p of the class E_P characterised by its initial state. Solving p consists in finding the state s of e_p that optimises Q_P, by applying actions from $action_P$ to the initial state of e_p.
Let us consider the following example. Let P_{robot} be an optimisation problem where a robot, considering its initial position in a maze built according to a specific model, seeks to find the exit. E_P, $action_P$ and Q_P are defined as follows:

- $E_{P_{robot}}$: a kind of robot. A robot of the kind $E_{P_{robot}}$ is characterised by its initial position in the maze.
- $action_{P_{robot}}$: $moveforward, turnleft, turnright$
- $Q_{P_{robot}}$: distance separating the robot from the exit of the maze

In this example, an instance p_{robot} of P_{robot} is characterised by $e_{p_{robot}}$, a robot of the kind $E_{P_{robot}}$, with an initial position in the maze. Solving p_{robot} consists in allowing $e_{p_{robot}}$ to find the exit or at least to reach the closest possible position to the exit.

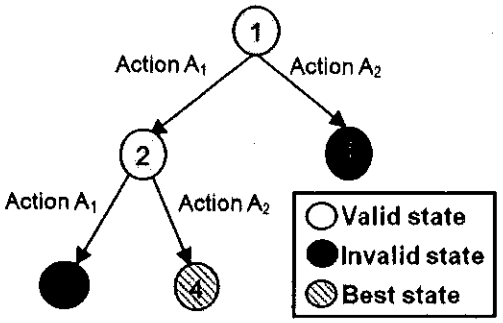

Figure 1: Example of state tree

2.1.2. Description of the considered systems

There are many ways to solve such an optimisation problem. In this paper, we are interested in systems that solve them by using a specific trial and error approach: the informed tree search strategy. This strategy consists in searching the best state of the entity by exploring a search tree. The transition from a state of the search tree to another corresponds to the application of an action from $action_P$. The "informed" aspect comes from the use of control knowledge (e.g. which action to apply) to guide the search tree exploration. This approach is commonly used in Artificial Intelligence. Figure 1 presents an example of state tree. The first artificial intelligence program, "Logic Theorist" [2], dedicated to proving theorems, was based on this type of approach. In order to find the proof of a logic problem, the Logic Theorist explores a state tree in which the root represents the initial hypothesis, and each branch, a deduction based on logic rules. This solving approach is particularly effective in contexts where expert knowledge can be used to guide the exploration process. Indeed, in contrary to optimisation approaches such as simulated annealing [3] or genetic algorithms [4] that allow to find a good solution to a problem without prior knowledge on it, the informed tree search approach requires to define domain specific knowledge to guide the exploration process. When this knowledge is relevant, the systems based on this approach can achieve excellent performances.

In this paper, we are interested in the revision of control knowledge used by systems based on an informed depth-first exploration of state trees. In order to build the state tree, the system carries out an action cycle. Figure 2 presents a classical action cycle.

It begins with the characterisation of the current state of the entity and its

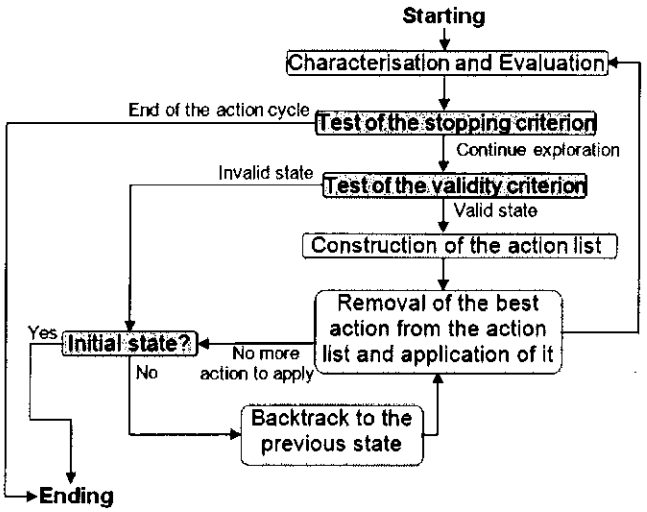

Figure 2: Action cycle

evaluation using the function Q_P. Then, the system tests if the current state is good enough or if it is necessary to continue the exploration of other states. If the system decides to continue the exploration, it tests if the current state is valid or not. A valid state is a state whose children have to be explored, an invalid state is a useless state from which it is not necessary to continue the exploration. If the state is not valid, the system backtracks to its previous state; otherwise, the system builds a list of actions to apply. If the action list is empty, the system backtracks to its previous state, otherwise it chooses the best action, applies it, then goes back to the first step. The action cycle ends when the stopping criterion is reached or when all actions have been applied for all valid states.
This generic system uses three types of control knowledge:

- *The action application knowledge* builds, for a current state, the action list, i.e. the list of the actions available in the current state and their application order. The role of this knowledge is to propose only relevant actions for each state, i.e. to avoid proposing actions that lead to uninteresting branches of the tree (ones that contains a priori no better state than the ones already found) and to order them in order to explore in priority the most interesting branches.

- *The validity criterion* determines, with respect to the previously visited

states, if the current state is valid or not. The role of this knowledge is to allow the system to skip uninteresting branches of the tree.

- *The ending cycle criterion* determines, with respect to the previously visited states, if the system action cycle has to continue the exploration or not. The role of this knowledge is to allow the system to stop the exploration when it is a priori not possible to find a state better than the ones already found.

Each type of knowledge can be expressed by one piece of knowledge or by several. For example, a system can integrate one knowledge base that defines, given a particular state, which action to apply or one knowledge base for each action that defines its application domain (when the action has to be applied).

In this paper, we are particularly interested in pieces of knowledge expressed by production rules. This kind of representation is commonly used in domains in which expert knowledge exists. Its interest is to be easily interpretable by domain experts and thus to ease the knowledge validation and update.

A production rule has the following form: *If Condition then Conclusion*

We assert that a set of measures is defined per piece of knowledge. For example, if a system integrates one piece of knowledge per action application domain, a measure set is defined per action. For each piece of knowledge, the conditions of the rules depend on the set of measures linked to the piece of knowledge.

2.2. Related works

The question of automatic revision of control knowledge has already been studied in the literature; in particular, in the context of speedup learning. Speedup learning seeks to improve the efficiency of problem solving systems with experience.

In the research works dedicated to speedup learning, many propose to use domain knowledge in order to ease the learning [5–7]. This knowledge, called domain theory, allows to learn from very few examples. However, this approach has some limitations. The major one is that the quality of the knowledge learnt this way is dependent of the quality of the domain theory. Thus, an incomplete or incorrect domain theory can cause numerous problems. In the context of our knowledge revision problem, it is not possible to

directly use these works. Indeed, we do not have a strong domain theory, just initial knowledge of uncertain quality. Other speedup learning works propose to acquire control knowledge without using domain theory. The most famous is the *LEX* system [8]. This system learns operator application condition by analysing previously solved problem instances. The transition from a particular operator application condition to a more generalised one is done by exploring the version space. The *LEX* system has some limitations, especially concerning the management of noisy data and the learning of disjunctive concepts. Other recent works such as [9] deal with the problems of learning a ranking function for the action application. Our work is in the continuity of the latter works. Our goal is also to allow the system to revise itself its own knowledge by means of experience. However, we propose to take advantage of the analysis of several resolved problem instances at the same time and not examine only a resolved instance at a time. Moreover, we propose to take better account of the interdependence between the different pieces of knowledge. Indeed, sometimes, it is not possible to determine if a piece of knowledge is really defective or if it is another piece of knowledge that is defective and that influences results of the application of the first piece of knowledge [10, 11].

In this paper, we are interested in the revision of knowledge expressed by production rules. Whereas many learning algorithms propose to induce rules from examples labelled by experts, very few allow to take initial rules into account. Among them, some are interested in the inductive knowledge base refinement. The goal of these works is to improve the system expert knowledge base. Most of them make the assumption that the knowledge base is almost valid and that only small improvements are needed [12]. Thus, some approaches propose to improve rule bases only by refining or deleting existing rules without giving the possibility to add new ones [13]. Others do not aim at refining rule bases directly, but aim at supporting the user during the refining process [14]. Many of these works are based on logical operators and thus rarely deal with noisy data [15]. Another drawback of many of these works is the increase of the number of rules [16] that can lead to readability problems. One common point of all these works is that they concern the revision of a unique rule base and do not allow to simultaneously revise several rule bases that depend on various measure sets. Thus, it is not possible to directly apply these approaches to our revision problem when several pieces of knowledge have to be considered at the same time (typically

when one base of rules is defined per actions).

3. Approach proposed

3.1. General approach

As explained in Section 1, our goal is to automatically revise the knowledge of a system based on an informed tree search strategy. The system has already a defined initial knowledge that we propose to take into account for the revision process. Indeed, we state the hypothesis that it is more interesting to revise the existing knowledge that just acquiring new one. Sometimes, the initial knowledge can contain information that cannot be learnt by Machine Learning techniques, such as the time-consuming aspect of some actions, the defects of the evaluation function, etc.
Our approach is based on the analysis of the execution logs. Thus, we propose to give the system capabilities to observe its own behaviour. The interest to give systems such observation capabilities are presented in [17]. We do not seek for on-line knowledge revision, i.e. to revise the control knowledge each time a problem instance is solved by the system. Indeed, we consider situations where it is possible to stop the system execution during several hours (e.g. during the night) to improve the performances of the system. In order to determine when the off-line revision process needs to be triggered, we propose to integrate in the system a diagnosis module able to evaluate on-line the knowledge quality. In this paper, we will not detail this diagnosis module, interest readers can refer to [10, 18].

Our knowledge revision approach is composed of two stages:

- *Exploration stage* consists in logging the process while the system solves a sample of problem instances.
- *Analysis stage* consists in analysing the logs obtained during the previous stage and in using them to revise the knowledge.

The *exploration stage* is independent of the way the knowledge is expressed. On the contrary, the *analysis stage*, which is based on the exploration of the knowledge space, has to be specialised for each type of knowledge representation. As mentioned above, in this paper, we present a specialisation of this stage for knowledge expressed in the form of production rules.

3.2. Exploration stage

The exploration stage is composed of two steps: the selection of a sample of problem instances that will be used for the revision process (Section 3.2.1) and the resolution of these problem instances with a specific knowledge set that will ensure a weak pruning of the state trees (Section 3.2.2).

3.2.1. Problem instance sample selection

In order to retrieve pertinent information from their resolution, the instances belonging to the selected sample have to be versatile enough to be representative of all available instances of the considered problem. However, their number has to be restricted in order to limit the computing time of the revision process: as it is off-line the time is not a hard constraint, but the resolution of a problem instance with a weak pruning introduces a high computational complexity and is very time consuming.
In order to select a representative sample, we proceed as follows:

1. Characterisation of all available instances of the considered problem;
2. Clustering of the instances;
3. Selection of a sample of instances in each cluster.

Characterisation of instances This step consists in building a *characterised instance set* by characterising all available instances of the considered problem by a measure set. A key point of this step is to choose the relevant measure set in order to pertinently characterise the instances. This choice has to be made according to the application domain. Feature subset selection techniques such as the one proposed by [19] can be used to select a pertinent subset of measures.

Clustering of the instances The goal of this step is to divide the available instances into groups composed of *similar* instances. Defining such groups allow to ensure that all kinds of instances will be represented in the instance sample. The available instances are divided into groups thanks to clustering techniques used on the *characterised instance set.* Clustering algorithms such as EM [20] can divide a set of objects described by a set of attributes into disjoint clusters. An important point of this step concerns the parameters used for the clustering and in particular, the choice of the group number and of the distance between objects. Concerning the first point, in the literature, several methods allowing to automatically infer the best number of groups

have been proposed [21, 22]. For the distance choice problem, it is interesting to use domain knowledge to define the distance. In the case where no domain knowledge is available, it is still possible to use classical distances like the Euclidian or the Manhattan ones. Most clustering algorithms allow to compute, from the distance used to divide the objects into groups, the probability for each object to belong to each group. In the context of our sampling method, this probability is used for the selection of instances in each group.

Instance selection This last step consists in selecting a sample of instances in each previously formed group. The selected instances are the ones that are the most representative of their group, *i.e.* the ones that have the highest probability of belonging to their group. The number n_i of instances selected from a group i depends on the expected size of the sample and on the relative size of the group in terms of the number of instances, compared to the total number of instances. The goal is to keep a proportional representation of the group, and, at the same time, to ensure that even the smallest groups are represented in the sample. The number n_i is computed as follows:

$$n_i = Max([\frac{expectedsizeofthesample \times nbofinstancesinthegroupi}{totalnbofavailableinstances} + 0.5], 1)$$

For example, let us consider a problem P, for which 100 instances are available. 20 instances of P are needed for the revision process. The clustering algorithm divides the instances in 2 groups, the first one composed of 25 instances, the second one composed of 75 instances. The representative sample will be composed of 5 instances of the first group and 15 instances of the second one.

3.2.2. Resolution with a full-exploration knowledge set and logging

Once a sample of objects has been selected, a resolution process is run on these instances and logged. In order to produce as much information as possible, we propose to use a *full-exploration* knowledge set for this resolution. We define the notion of *full-exploration* knowledge set as a knowledge set that allows the system to construct all the possible states that can be visited by all possible knowledge sets for each resolved problem instance. Typically, a *full-exploration* knowledge set is a knowledge set for which all possible actions are tested for each valid state and for which the ending cycle criterion and the validity criterion are as weak as possible to just ensure the convergence of the

exploration. The *full-exploration* knowledge set has two interests. First, it ensures to know the best states that can be obtained according to the action set and to know the action sequences to apply to reach them. Secondly, once an instance has been resolved with this knowledge set, it becomes possible for this instance to simulate any possible knowledge set by rearranging the states, without having to build new states i.e. without running again the resolution process. Thus, once the problem instance sample selected during the first step has been resolved with the *full-exploration* knowledge, it becomes possible to test any knowledge set on this sample with very few computation resources.

3.3. Analysis stage

3.3.1. General revision approach by log analysis

After the *exploration stage*, we have a sample of problem instances resolved with a *full-exploration* knowledge set and the associated state trees that compose the log. We are now interested in the use of these state trees in order to revise the knowledge. We propose to formulate the revision problem as a search problem: we will search the knowledge set that maximises the performances of the system. The problem of the system performance evaluation is discussed in the next section (Section 3.3.2). Concerning the search problem, the search space corresponds to the set of values that can take the different pieces of knowledge. In order to reduce the search space, we do not propose to revise all pieces of knowledge at the same time, but by kind of knowledge. For example, to revise the pieces of knowledge that define the constraint priority in a first step, then, in second step, the pieces of knowledge that define the application domain of the actions are revised, and so on. The search approach used to revise each piece of knowledge depends on the nature of the piece. In section 3.3.3, we propose a search approach dedicated to the rule base revision.

3.3.2. Evaluation of the system performance

As stated above, our revision approach consists in searching the knowledge set that maximises the performances of the system.

The system performance can be expressed in terms of *effectiveness* and *efficiency*. The *effectiveness* concerns the quality of the results obtained by the system, i.e. the quality of the best state found. The *efficiency* concerns the computational cost of the problem instances resolution, i.e. the system speed to carry out the tree search exploration. Good knowledge allows the

system to be both effective and efficient, i.e. to guide the exploration directly toward an optimal state without visiting useless states. However, to maximise both *effectiveness* and *efficiency* can be impossible for some applications. Thus, we are in the context of a multi-objective optimisation problem. Several approaches exist to solve such problem. In this work, we chose the simplest one, which consists in aggregating both *effectiveness* and *efficiency* in an single objective function. The system performance is then represented by an unique evaluation function. More complex approaches, for example based on multicriteria analysis (e.g. [23, 24]) or on the building of the Pareto set (e.g. [25]), could be studied.

We note $Perf(S_K, P)$ the function used for evaluating the performance of a system S while using a knowledge set K for the solving of all problem instances of the class P. The computation of the function requires solving all instances p of P. In practice, most of the time, it is impossible to compute $Perf(S_K, P)$. Indeed, it is rarely possible to solve each p of P. Thus, for our revision approach, we will just estimate $Perf(S_K, P_n)$ on the sample of n problem instances selected and solved during the *exploration stage*. In fact, in the *analysis stage* we will search the knowledge set K that maximises $Perf(S_K, P_n)$.

The choice of favouring the *effectiveness* or the *efficiency* has to be made according to the user needs. In fact, there is no generic performance function $Perf(S_K, P_n)$. This one has to be specifically defined for each application. As an example, in Section 4.2.3, we describe the evaluation function that we used for our application. The definition of this function can be difficult. In the perspectives section of the paper (Section 5), we come back to this definition problem.

3.3.3. Search approach dedicated to the rule base revision

In this Section, we propose a search approach dedicated to the revision of rule bases. In Section 2.1.2, we asserted that a set of measures is defined per rule base. All predicates of the rules are expressed (only) according to the measures linked to the rule base. Thus, the search space associated to a rule base is the set of the combinations of values that can be assumed by the measures. This search space is infinite as soon as at least one measure returns continuous values. In order to reduce the search space, we propose to divide the measure set space into areas admitting *a priori* a same behaviour, an area corresponding to the *condition* of a rule, and the behaviour associated to this area to the *conclusion* of the rule. Actually, we state the hypothesis

that the ideal behaviour of the system is not chaotic, so that continuous subspaces of the measure set space exist in which this ideal behaviour is constant. Thus, our approach consists in trying to determine these areas that admit a homogenous ideal behaviour of the system (*i.e.* homogeneous conclusions of the rules), based on an analysis of the successes and failures encountered when resolving the instances sample. For example, consider a system that can propose only the action A that depends on a measure set composed of only one real measure M. An example of partitioning can be to decompose the domain of M (and thus the measure set space of A) into two areas: $(M < 0)$ and $(M \geq 0)$. The revision problem then consists in assigning the best possible conclusion to each area of each piece of knowledge. We call solution, a complete assignment of conclusions for the considered areas (a conclusion is assigned to each area). Our approach is composed of four steps as presented in Figure 3. The first one consists in building, from the logs (i.e. from the state trees associated with the generalised sample objects), example sets that translate the ideal behaviour of each rule base. The second step consists in partitioning the measures set of each rule base in areas that admit a priori a homogenous decision to apply (conclusion of the rule). The third step consists in searching by the mean of a local search the best conclusion to assign to each area. The last step consists in simplifying the rule bases by rule aggregation.

Step 1: construction of the example sets The construction of the example sets is achieved by analysing the state trees obtained during the *exploration stage*. An example set is build per rule base. An example corresponds to one state of a state tree. It is composed of n predictors and a label. The predictors are the measures associated with the rule base, and the label is the decision assessed as the good one for this experienced state. For example, in the case of a rule base concerning the application domain of an action, the decision can be "apply this action" or "do not apply any action".

The method used for the construction of the example sets has been described in [26]. It first consists in extracting the best paths from each state tree. A best path is a sequence of at least two states, which has the root of a tree (or of a sub-tree) for initial state and the best state of this tree (or sub-tree) for final state. Then, from each state of each best path, an example is built that contains the relevant information, depending on the rule base under revision.

For example, concerning the action application knowledge, if a state belongs to a best path and if the application of an action leads to another state

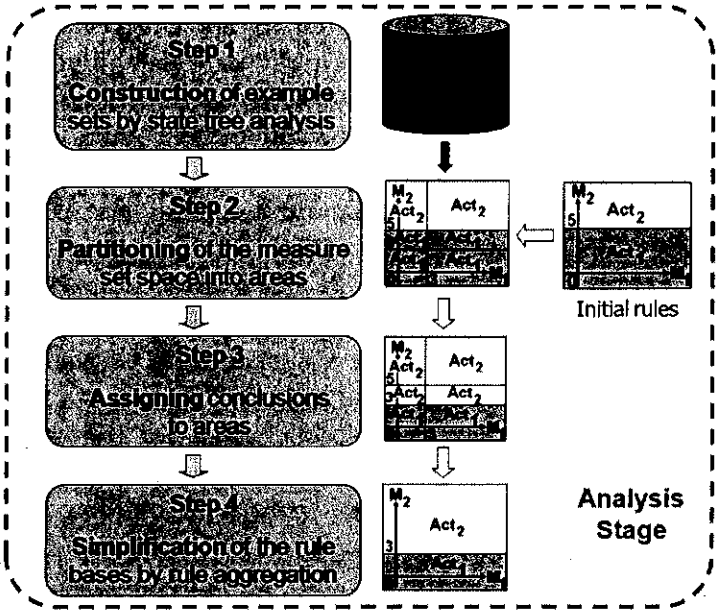

Figure 3: Search approach for rule base revision

of the same best path, the ideal decision is "apply this action". Figure 4 gives a simplified example of the example set built from the resolution of a problem instance with two actions Act_1 and Act_2.

Step 2: partitioning of the measure set space The second step of our approach requires to partition the measures set into disjoint areas. Each area corresponds to the *condition* of a production rule. One constraint of this partitioning is to take the initial rules into account. Initial rules already form a partitioning of the measures space for which each area admits one conclusion. In order to take initial rules into account, we impose that each so-obtained area be a sub-area of an initial area, *i.e.* that a rule defining a final area be either one of the initial rules or a specialisation of one of the initial rules. The interest of this constraint is to keep the possibility to obtain rules similar to the initial rules after the revision process. The partitioning approach that we propose consists in learning a new partitioning from the example set thanks to supervised Machine Learning techniques and in combining it with the partitioning formed by the initial rules. Several algorithms such as C4.5 [27] or RIPPER [28] can be used to learn a new partitioning. The learnt partitioning is independent of the initial rules and is

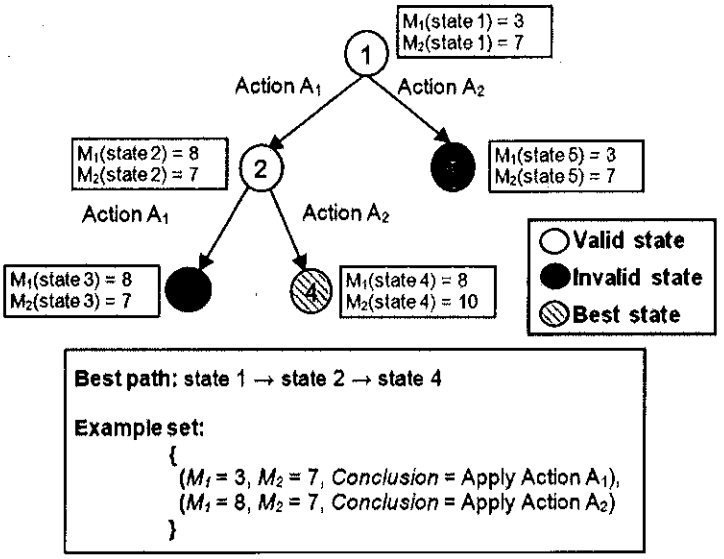

Figure 4: Example of a built example set

based on the experience (the example set). The areas obtained by it are thus homogenous according to the experience. At this stage of the partitioning process, we have two partitionings of the same measure space: one coming from the initial knowledge and the other from the experience. We state the hypothesis that each partitioning contains pertinent pieces of information that are not contained in the other one. Indeed, as stated in Section 3.1, we assume that the initial knowledge can contain information that cannot be learnt by Machine Learning techniques. Thus, we propose to not only consider the learnt partitioning but to combine the two partitioning together. We then specialise the partitioning formed by the initial rules by combining it to this new partitioning (Figure 5). In practice, we propose to independently partition each initial area (formed by an initial rule) by combining it with the set of new areas. The method used to partition an initial area and thus to specialise an initial rule is the following one. First, the initial area is combined with a first new area. Then, the result of this combination, which is a partition of the initial area, is combined with a second new area, and so on until the result has been combined with all areas of the new partitioning.

It is possible to compute the maximum number of areas that can be obtained at the end the partitioning process. Let RB be a rule base linked

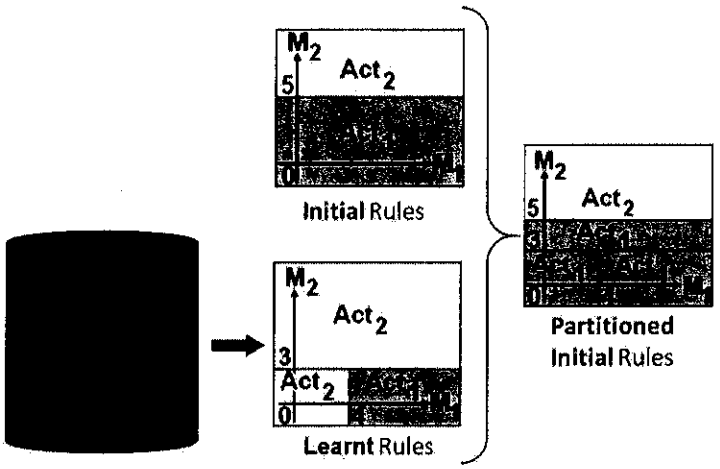

Figure 5: Partitioning approach

to a measure set MS. We defined two partitioning for the RB: P_a and P_b. We note $nb_{intvls}(P, m)$ the function returning the number of intervals for the measure m defined by the partitioning P. For example, in Figure 5, the partitioning formed by the initial rule base does not decompose M_1 in any interval (this measure is not used by the initial rule base) and decompose the measure M_2 in 2 intervals $(]-\infty, 5]$ and $]5, \infty[)$. The learnt partitioning decomposed the measure M_1 in 2 intervals $(]-\infty, 4]$ and $]4, \infty[)$ and the measure M_1 in 2 intervals $(]-\infty, 3]$ and $]3, \infty[)$. The maximum number of areas that can be obtained by combining the two partitioning P_a and P_b is bound by:

$$nb_areas(P_a, P_b) \leq \prod_{m \in MS} [nb_{intvls}(P_a, m) + nb_{intvls}(P_b, m) - 1]$$

Step 3: conclusion assignments by local search Once the partitioning carried out, the next step of our approach consists in assigning the best possible conclusion to each area obtained at the end of the previous step. We call *solution* a complete assignment of conclusions for each area of the measured set linked to the rule base considered. We formalise the conclusion assignment problem as an optimisation problem in which we search, for a given system S, the solution sol among the possible solutions set Sol that maximises the performance of the system (presented Section 3.3.2). According to the fact that, for each area, we have to assign a conclusion among a

set of conclusions $Concl$, the size of the solution space (size of Sol) is equals to $|Concl|^{numberofareas}$. We remind that the revision can concern several rule bases, and thus several area sets, at the same time (e.g. the action application knowledge). Therefore, the number of areas can be very high. To help this search, we dispose of an initial solution (the initial rule base) that is often good. There are numerous methods to solve a problem of this kind. Due to the size of the solution space, it is impossible to use a complete search approach. Thus, we use an incomplete approach. Indeed, in order to solve this problem, we propose to use a local search algorithm. The principle of this kind of algorithm is to start with an initial solution and to try to improve it, step by step, by exploring its neighbourhood. Most of the time, these algorithms are very effective for this kind of exploration problem. There are numerous local search algorithms such as *hillclimbing*, *tabusearch* [29] or *simulatedannealing* [3] that can be used to solve this problem. In the context of our off-line revision process, the efficiency of the search method is not a major issue. Thus, it is preferable to use methods allowing to avoid to be stuck at local optimum like the *tabusearch* or the *simulatedannealing* rather than a simple $hill - climbing$. Concerning the choice between these two methods, the experiments we carried out showed similar results. In this paper, we propose to use the *tabusearch*. Local search approaches require to define the notion of neighbourhood of a solution. For our problem, we define it as the set of solutions for which only one of the areas has its conclusion value changed. Concerning the parameters specific to the *tabusearch*, we have to define an ending criterion and the size of the tabu list. In out context, we chose a simple criterion: the exploration process stops after 1 minute. This time was high enough to converse toward a best solution in all the experiments we carried out. For the size of the tabu list, we propose to linkit with the number of areas. Thus, the size of the tabu list is computed with the following formulae:

$$TabuList_size = 1 + \frac{numberofareas}{4}$$

Remark that other types of neighbourhood and parameter values can be used. The ones we proposed gave good results during the experiments we carried out, however a deeper study is needed.

Step 4: rule base simplification The exploring step allows to assign a good conclusion to each area. The last step of our approach consists in simplifying the obtained (revised) rules bases by merging the areas. The

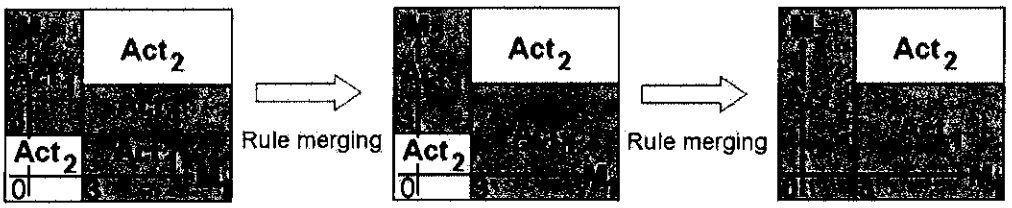

Figure 6: Example of rule base simplification

goal of this step is to improve the readability of the rule bases as well as their generalisation capacity. Thus, two kinds of merging are done: merging between areas admitting the same conclusion (to make the rule base more readable) and merging between areas that do not admit the same conclusion (to generalise the rule base). At each iteration, two areas are merged in order to form a bigger one. Two areas which admit different conclusions can be merged. In this case, both conclusions are tested for the area obtained after merging. If the performances of the system were better before the merging, this one is cancelled and another merging is tested. Otherwise, the conclusion kept is the one that maximise the performances of the system. The performances of the system are computed by the same evaluation function as the one used during the conclusion assignment step (Section 3.3.2). However, in order to avoid the overfitting problem, we propose to use a method inspired by the one used for the post-pruning of decision trees [30]. Thus, we propose to divide the sample of problem instances selected and solved during the *exploration stage* in two sub-samples: two thirds of the problem instances are used for the conclusion assignment step, and the last third is used for the simplification step.

Figure 6 gives an example of simplification: in a first iteration, the area defined by the rule "if $M_1 > 3$ and $M_2 > 3$ and $M_2 \leq 5$ then apply action Act_1" is merged with the area defined by the rule "if $M_1 > 3$ and $M_2 \leq 3$ then apply action Act_1". The resulting area is defined by the rule "if $M_1 > 3$ and $M_2 \leq 5$ then apply action Act_1". In a second iteration, the area defined by the rule "if $M_1 \leq 3$ and $M_2 > 3$ then apply action Act_1" is merged with the area defined by the rule "if $M_1 \leq 3$ and $M_2 \leq 3$ then apply action Act_2". The resulting area is defined by the rule "if $M_1 \leq 3$ then apply action Act_1".

3.4. Theoretical justification of the approach and comparison with existing ones

The general principle of our approach is to search the knowledge base that optimises an evaluation function estimated on a sample of problem instances. Concerning the rule base revision, the knowledge base can only be modified during two steps: the conclusion assignments step (step 3) and the rule base simplification step (step 4). During these two steps, each modification of the knowledge base is assesed by the evaluation function. Indeed, only modifications that do not deteriorate the value of this function are taken into account. So, after revision, the value of the evaluation function will inevitably be equal or higher than before revision. Thus, ensuring that the evaluation function is in total adequacy with the user needs and that the problem instance sample used by the revision process is perfectly representative of all instances of the considered problem allows to guarantee that the knowledge base obtained after revision is better than the initial one.

Our approach allows to improve the performance of problem solving systems with experience. In this context, it is in the continuity of the speedup learning approaches. However, while most of speedup learning approaches [5, 6] do not provide the system with the ability to solve new problem instances (just improve the system efficiency), our approach allows it to infer new knowledge. Moreover, unlike many speed up learning approaches, it does not require a domain theory [5–7] and allows to manage noisy data [8] –which is particularly important in the context of systems that apply actions which results are non-predictable. In addition, our approach can be applied to revise different kinds of control knowledge and not only knowledge related to action application order [9]. At last, our approach allows to revise the knowledge and not just to add new pieces of knowledge allowing the system to be more efficient. Our approach can also be compared to case-based reasoning. Systems based on cased-based reasoning solve new problem instances by using solutions of similar past problem instances. Numerous works such as [31] dealt with the revision and the reorganisation of the knowledge in this kind of systems. The main difference between case-based reasoning and our approach concerns the type of knowledge considered. Indeed, in case-based reasoning, the knowledge is represented by a set of solved cases, while it is represented as sets of rules in our proposal.

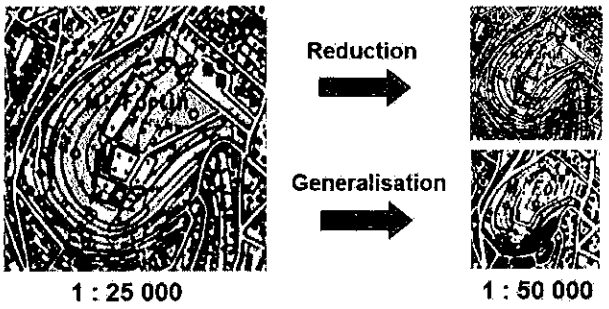

Figure 7: Cartographic Generalisation

4. Application to cartographic generalisation

4.1. Context of the application

4.1.1. Automatic cartographic generalisation

We implemented our knowledge revision approach in the domain of cartographic generalisation. Cartographic generalisation is a process that aims at decreasing the level of details of geographic data in order to produce a map at a given scale (smaller than the reference scale of the initial data). The goal of this process is to ensure the readability of the map while keeping the essential information of the initial data. The cartographic generalisation is not a simple map reduction; it requires to apply numerous operations such as object scaling, displacements and eliminations. Figure 7 gives an example of cartographic generalisation.

The automation of the generalisation process from vector geographic databases is an interesting industrial application context. Indeed, this problem is far from being solved. Moreover, it directly interests the mapping agencies that wish to improve their map production lines. At last, the multiplication of web sites allowing to create one's own map increases the needs of reliable and effective automatic generalisation processes. The problem of the generalisation automation is complex. One approach to solve it is to use a local, step-by-step and knowledge-based method [32–34]: each vector object of the database (representing a building, a road segment, etc.) is modified by application of a sequence of generalisation algorithms realising atomic transformations. The sequence of algorithms is not predetermined but built on the fly for each object according to control knowledge, depending on its characteristics and the measured effects of the algorithms on it. This approach implies to manage a knowledge base. In particular, it requires

to adapt the knowledge when new elements such as new generalisation algorithms are integrated in the generalisation systems or when the user needs (the map specifications) change. Nowadays, this knowledge adaptation is done "manually" by generalisation experts and is often long and fastidious. Indeed, it requires facing the problem of knowledge collecting and formalising [33]. Thus, it is interesting to give the system capabilities to revise by itself its own knowledge base. Several works have already used Machine Learning to learn relevant control knowledge [33, 35, 36] but only few have proposed to automatically revise existing knowledge. Among them, Burghardt and Neun [37] proposes to use previously generalised objects to build a case base. Concerning the rule base revision, the only work that we are aware of is [38]. It proposes to use experience to learn new rules that are added to the system. Contrarily to our work, this work does not propose to revise existing rules, but only to add new ones. Thus, even if an initial rule is not pertinent, this one cannot be removed or modified.

4.1.2. The generalisation system

The generalisation system that we use for our experiment is based on the AGENT model. This model, which is well-established in the generalisation community, originates in [39] and was used during the AGENT European Project [40]. The AGENT model has been described in details in [34]. In this model, objects of the vector geographic database to generalise (roads, buildings, etc) are modelled as agents. The geographic agents manage their own generalisation, choosing and applying generalisation algorithms (actions) to themselves. The generalisation of the agents 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. In addition, constraints have the role of computing for their associated agent, for each state, a list of actions to try. For example, if the size constraint of a building assesses that the agent is too small, it will propose a scaling action to the agent. Each constraint has a level of satisfaction ranged between 1 (constraint not satisfied at all) to 10 (constraint perfectly satisfied). For each state, the agent computes its own satisfaction as the sum of each constraint satisfaction weighted by their importance. To satisfy its constraints as well as possible, a geographical agent carries out a cycle of actions during which it tests several actions 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 action cycle results in an in-

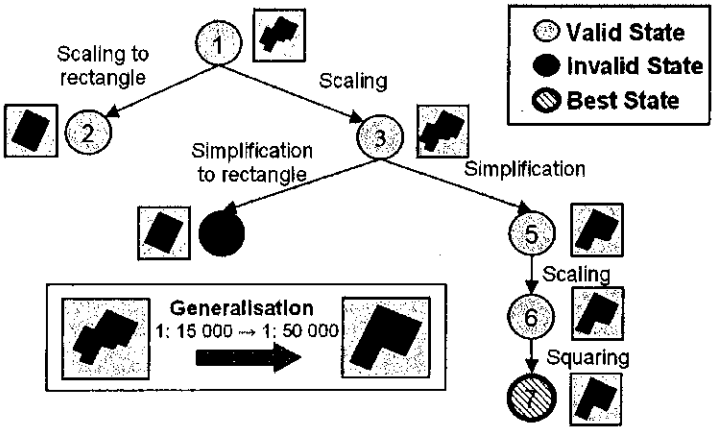

Figure 8: Example of a state tree for the generalisation of a building

formed exploration of a state tree. Each state represents the geometric state of considered geographic objects The action cycle used is the one presented in Figure 2. Figure 8 gives an example of a state tree obtained with the generalisation system.

4.2. Settings of our test case

The knowledge revision method described in Section 3 has been implemented in a research platform based on Clarity (1Spatial), resulting in a prototype for knowledge revision associated to the AGENT generalisation model. In this section, we report on one of the experiments carried out in order to assess this knowledge revision method. This test case focuses on the generalisation of urban space. Generalising the urban space is a complex problem. It requires to manage a high density of data. The use of different levels of analysis has proven to be a good solution for dealing with this complexity. Several works were interested in the definition of these levels of analysis [39, 41, 42]. In this test case, we concentrate on the analysis level of the building groups defined by [43]: a building group is a space composed of a set of "close" buildings belonging to the same building block (space surrounded by a minimum cycle of roads). The initial data are stemming from BD TOPO, the 1m resolution database of the French NMA (reference scale approximately 1:15 000). The target scale is 1:50 000. The choice of this class of agents and of this target scale is due to the difficulty to define good knowledge for their generalisation. Actually, the generalisation of

building groups poses efficiency and effectiveness problems. The generalisation algorithms used to generalise them are often time consuming. Thus, it is important to find a good state while exploring few states. Now, defining knowledge allowing this is particularly complex.

4.2.1. Defined generalisation constraints and available generalisation algorithms

In this section, we present the constraints and the actions used for our experiments. We defined, with the contribution of generalisation experts, six constraints and five actions for building group agents:

- *Proximity constraint*: this constraint states that the buildings should not be too close to each others, neither to close to the roads. It is assessed by computing distances between neighbouring buildings and roads, following [44]. This constraint can propose building removal and displacement actions:
  - *Building displacement action*: this action displaces buildings that have proximity problems. It is based on the displacement algorithm proposed by [44].
  - *Local building removal action:* this action removes buildings according to a local context. It removes in priority the buildings that have the most serious overlapping problems. The algorithm of this action is presented in [11].
  - *Building removal/displacement action:* this action selects the building that has the most serious proximity problems and removes it. If another building is close to the removed one, it is displaced in order to be closer to the removed building. This action is based on the displacement algorithm proposed by [39].

  Building satisfaction constraint: this constraint states that buildings composing the building group should individually satisfy their internal constraints. It is assessed by analysing the individual satisfaction of the buildings. This constraint can propose a building generalisation action:
  - *Building generalisation action*: this action triggers the individual generalisation of the building agents composing the building group.

Density constraint: this constraint states that the building density should not be too high in comparison to the initial building density. It is assessed by comparing the black/white ratios at the current state and at the initial state. This constraint can propose a building removal action:

- *Global building removal action*: this action removes buildings according to the global context. It removes in priority buildings that have the less space to move. This action is based on the algorithm proposed by [39].

Spatial distribution constraint: this constraint states that the current building spatial distribution should be close to the initial building spatial distribution. It is assessed by checking that no empty space have appeared where there were buildings initially. *Big buildings preservation constraint*: this constraint states that buildings, having an area bigger than a threshold, should not be removed. It is assessed by comparing the number of big buildings at the current state and at the initial state. *Corner buildings preservation constraint*: this constraint states that buildings located in a corner of roads should not be removed. It is assessed by comparing the number of corners, in which there are buildings, at the current state and at the initial state. This constraint is presented in [11].

The last two constraints represent a partial expression of the expectation that most important buildings of the group should be preserved. It can be noticed that no action is associated with the spatial distribution, big buildings preservation and corner buildings preservation constraints. These constraints are used by the system only to reject states of which the degree of violation would be too high.

4.2.2. Initial knowledge set

For our experiment, we defined three different initial knowledge sets: one ensuring to find the best possible state for each generalised building group (but of much too high computational complexity), and two others defined by experts. The experts defined their knowledge sets with the help of tools allowing them to manually test the actions, to display measures values for a given agent state, to generalise geographical agents with the AGENT system including different pieces of knowledge and to visualise the resulting states

trees. In order to tune the control knowledge, they carried out their experiments with building groups localised inside the town of Orthez (South-west of France). This town is composed of 280 building groups. The three knowledge sets are described as follows:

- *Most effective knowledge set*: knowledge set derived from the *full-exploration* knowledge set. For each generalised building group, this knowledge set ensures to find the best possible state considering the constraints and the available actions proposed by them. Nevertheless, it requires to explore many states per generalisation and is thus not efficient at all. This efficiency problem leads to the impossibility to use this knowledge set for real application.

- *Cartographic expert knowledge set*: knowledge set defined by an expert on cartography but not on the AGENT model. The heuristic used by the expert while defining this knowledge set was to search in priority to define an effective knowledge set. Then, he tried to improve the efficiency of its knowledge set by adding knowledge related to the pruning of the states trees.

- *AGENT model expert knowledge set*: knowledge set defined by an expert on both cartography and the AGENT model. The heuristic used by the expert while defining this knowledge set is the same as the one used by the previous expert: the expert searched in priority to define an effective knowledge set. Then, he tried to improve the efficiency of its knowledge set by adding knowledge related to the pruning of the states trees.

4.2.3. Revision parameters and test protocol

The implementation of our approach requires to choose three algorithms and to define a performance function. Our general algorithm choice approach was to choose well-established and well-known algorithms. Thus, for the instance selection part (Section 3.2.1), which consists in choosing a sample of building groups, we used the EM algorithm [20] to divide the building groups into cluster. We used the $C4.5$ algorithm [27] to learn rules for our partitioning method, and the *tabusearch* [29] for our conclusion assignment approach. The function $Perf(S_K, Obj)$ was defined empirically by means of experiments with different knowledge sets and different building groups

(not the ones used during this test case, in order to avoid risks of over-fitting). We were considering the situation of a production line, where the most important is to obtain good cartographic results and where it is possible to retouch manually (by technicians) some generalised building groups. We used the following $Perf(S_K, Obj)$ function:

$$Perf(S_K, Obj) = \frac{4 \times Effectiveness(S_K, Obj) + Efficiency(S_K, Obj)}{5}$$

This function favours the effectiveness of the system over its efficiency thanks to factor 4. The value of this factor was defined with the help of a generalisation expert. The expert analysed the results obtained with different knowledge sets on several datasets in terms of efficiency and effectiveness and chose, from this analyse, a value for the effectiveness factor. Of course, this value can be changed to reflect the needs of the user. Approaches such as [45] can be used to adjust the weights.

The effectiveness evaluation function has been defined as follows:

$$Effectiveness(S_K, Obj) = \left(\frac{4 \times FirstQuartile(S_K, Obj) + Mean(S_K, Obj)}{20} \right)^2$$

This function allows to take the mean satisfaction of the generalised object into account and to balance this result by the value of the satisfaction first quartile. The interest of this weighting comes from the fact that it is preferable for the mapping agencies to obtain three quarters of well generalised objects and one quarter of bad-generalised objects (that can be retouched by technicians) rather than obtaining average homogeneous results (which requires much more retouches). The factor 1/20 is used to normalise the value of this function. Actually, we remind that the satisfaction of a geographic agent is ranged between 1 to 10. We add a power 2 in order to accentuate the difference between values. Indeed, in terms of satisfaction, a good result for our application scenario (building group that does not need to be retouched) corresponds to a value higher than 9. A value lower than 8 corresponds to a non-acceptable generalisation result.

Concerning the efficiency evaluation function, we used the following one:

$$Efficiency(S_K, Obj) = NbStates(S_K, Obj)^{\frac{1}{3} \times (\sqrt[-5]{NbStates(S_K, Obj)} - 1)}$$

The choice of this efficiency function can be explained by the wish that the efficiency function covers a large range of values (on the interval [0,1])

for values of mean number of states ranged between 10 to ∞. Actually, for our application scenario, we have empirically defined that a mean number of visited states of 10 can be considered as a very good result in terms of efficiency (*Efficiency* value higher than 0.64). A number of states ranges between 10 to 15 can be considered as a good result in terms of efficiency (*Efficiency* value ranged between 0.52 and 0.64); a number of states ranged between 15 to 25, an average result (*Efficiency* value ranged between 0.38 and 0.52); a number of states higher than 30 is not acceptable in terms of efficiency for a real application (*Efficiency* value lower than 0.33).
Concerning the test protocol, we used 50 building groups to revise the three initial knowledge sets. These 50 building groups were selected by our object sample selection approach in the same area as the one used by the experts to tune their knowledge sets (the town of Orthez). We then assessed the initial knowledge sets and their revised versions on the 200 building groups contained in another town of the South-West of France, namely Salies-de-Barn.

4.2.4. Results

Figure 9 shows the results, obtained with the initial and revised knowledge sets in terms of effectiveness and efficiency. One point represents a generalised building group. The y-coordinate corresponds to the generalised building group satisfaction (the system effectiveness) and the x-coordinate to the number of states visited to generalise the building group (the system efficiency). Ideally, all points (i.e. generalised building groups) should be located in the top-left corner. This figure shows as well diagrams representing the distribution of the final satisfactions. Figure 10 gives an example of cartographic results.

Firstly, we can notice the difficulties of defining a knowledge set that will ensure both the effectiveness and the efficiency of the generalisation system. Indeed, even with a good command of the AGENT model, it is not easy to define a knowledge set that is both effective and efficient for all geographic objects. This is shown by the presence of a high number of light grey squares on the bottom right parts of the diagram on Figure 9c (satisfaction lower than 8 and number of visited states higher than 30). The AGENT model expert defined a good knowledge set (better than the one defined by the cartographic expert in terms of effectiveness) but not a perfect one. Whereas this knowledge set permits the acquisition of good cartographic results (most squares above the threshold of 9 on Figure 9c, few errors in Figure 10c), it is

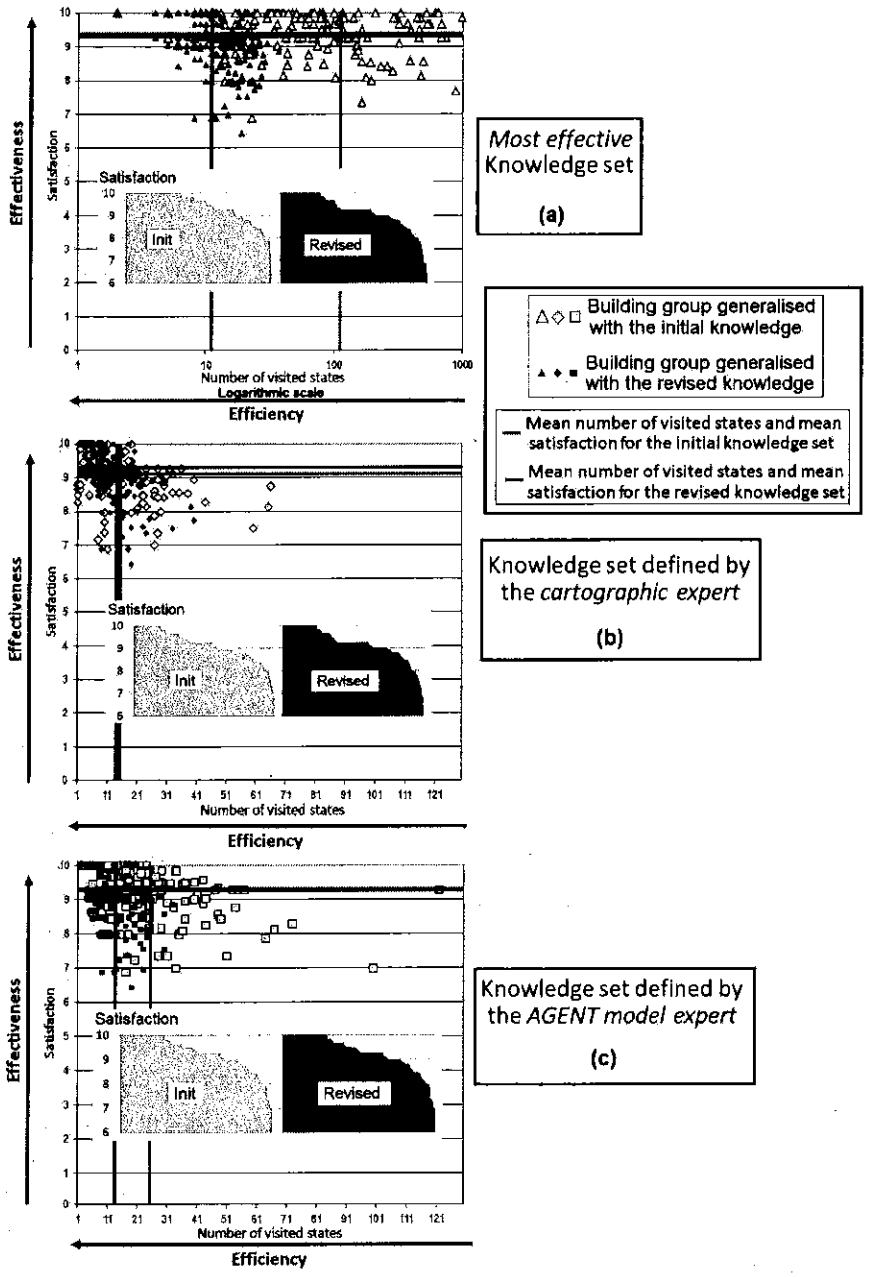

Figure 9: System effectiveness and efficiency: comparison results between the initial knowledge sets and the revised ones

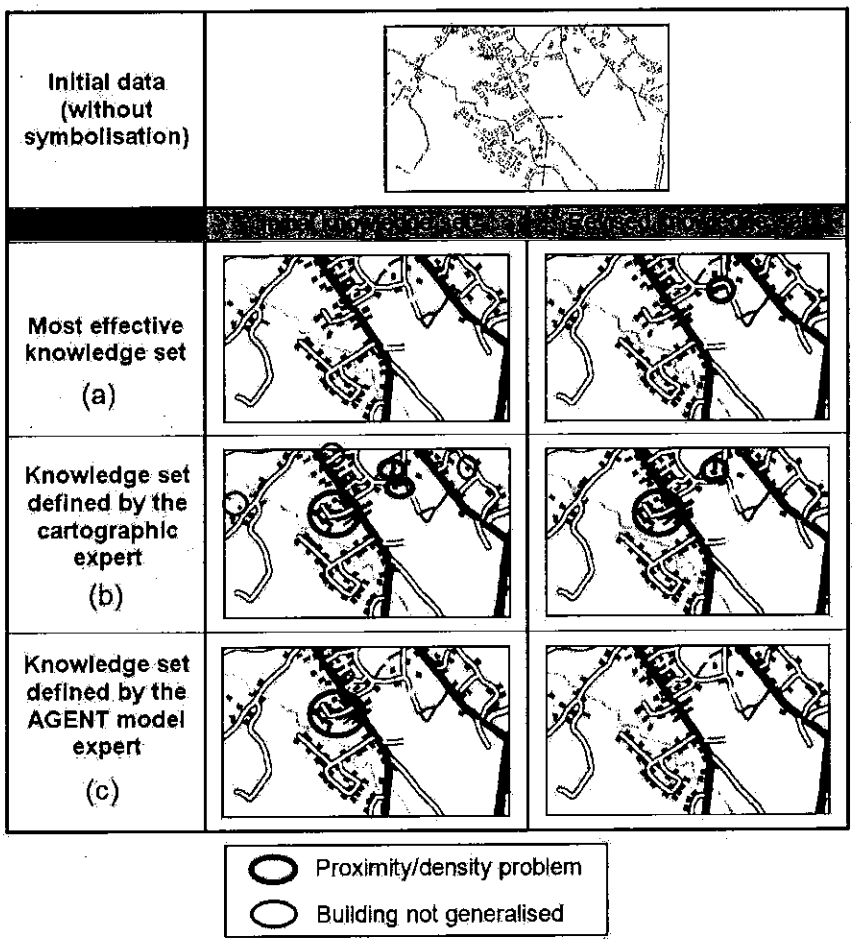

Figure 10: Example of cartographic results

not efficient (many light grey squares above the threshold of 30 on Figure 9c). The cartographic expert whose effort favoured obtaining good efficiency came up short in terms of effectiveness (light grey diamonds to low in Figure 9b). Actually, the corresponding cartographic results (Figure 10b) suffer from several problems: several buildings were not generalised while density and proximity problems are evident. As to the results obtained with the most effective knowledge set (Figures 9a and 10a), we can remark that the results are very good in terms of effectiveness but quite poor in terms of efficiency. Actually, the mean number of visited states is close to 150 per generalisation. This number is far too high to consider using this knowledge set for a real world application.

With regard to the result obtained after revision, we see that in all cases they are good–both in terms of effectiveness and efficiency. Essentially, for the most effective knowledge set, if the results are slightly less good in terms of effectiveness, they are far better in terms of efficiency. Remember, for each generalised building group, the most effective knowledge set ensures finding the best possible state considering the constraints and the available actions proposed. Therefore, after revision, it was not possible to obtain better results in terms of effectiveness. Recall also that before the revision, the most effective knowledge set was not usable for real applications due to its efficiency problems. The revision allowed dividing the mean number of visited states per generalisation by a factor of 10 and thus rendering it perfectly usable. For the knowledge set defined by the cartographic expert, the results are better in terms of both efficiency and effectiveness (Figure 9b). The effectiveness is the most improved, notably, after revision; no more buildings remain ungeneralised (Figure 10b). For the knowledge set defined by the AGENT expert, we observe that the quality in terms of effectiveness is very close for both initial and revised knowledge sets. Nevertheless, the revision process allowed for great improvement in the system's efficiency: actually, the mean number of visited states per generalisation was divided by a factor 2 (Figure 9c). Comparing the results attained with the three revised knowledge sets, we notice that they all achieved results that were very close in terms of efficiency. Their mean number of visited states was lower than 13, and thus lower than the threshold value of 15–which marks the limit between a good and an average result in terms of efficiency. The best cartographic results were achieved while using the revised version of the knowledge set defined by the AGENT expert, which was the best of the three initial knowledge sets. This experiment shows that if it is possible to obtain good knowledge sets

when revising bad knowledge sets, it is possible to get even better results when revising good knowledge sets.

4.2.5. Conclusion of the presented experiment

In this section, we have presented an experiment carried out for the generalisation of building groups with the AGENT model.

This experiment shows that defining a knowledge set that allows the system to be both effective and efficient is very complex. Our experts successfully defined good knowledge in terms of effectiveness. Nevertheless, they did not succeed in introducing pruning knowledge in order to improve the efficiency of the system while ensuring the cartographic quality of the results. Between our two experts, one chose to ensure the quality of the cartographic result at the expense of the efficiency of the system (the AGENT expert). The other one tried to improve the efficiency by introducing strict pruning knowledge but deteriorated greatly the cartographic quality of the results (the cartographic expert).

The results obtained with our revision approach shows that it allows to answer the problem of the definition of a both effective and efficient knowledge set. Actually, the three knowledge sets obtained after revision allowed to obtain good generalisation results both in terms of efficiency and in terms of effectiveness.

This experiment showed as well that our revision approach allows to take the specificities of the initial knowledge into account. Actually the revision from the knowledge set defined by the AGENT expert (which is the best initial knowledge set of the three) allowed to obtain the best knowledge set after revision. An explanation is that the revision process preserved some pertinent elements defined by the AGENT expert that could not be acquired directly by the experience (i.e. from the logs generated during the exploration stage). Thus, it appears particularly interesting to revise existing knowledge rather than just trying to acquire directly new knowledge. This result confirms our initial hypothesis (Section 3.1).

5. Conclusion

In this paper, we have underlined the interest of integrating an automatic module of knowledge revision inside a problem solving system based on an informed tree search strategy. We have proposed a generic approach for the revision of the control knowledge based on logs analysis. We developed a

specialisation of this approach allowing to revise knowledge expressed in the form of production rules. We have assessed this method by implementing it in the context of cartographic generalisation and by carrying out an experiment with the AGENT model for the generalisation of building groups. This experiment has showed that our revision approach can allow the system to improve the initial control knowledge in terms of efficiency or/and effectiveness. It also confirms our initial hypothesis that it is interesting to take the initial knowledge into account and to revise it rather than just acquiring new knowledge.

Our approach can be applied to other kinds of geographic objects and to others scales. Carrying experiments for other geographic objects could allow to test our approach in a more intensive manner and to study its limits. In the same way, as our approach is generic, it can be applied to other systems based on an informed tree search strategy. The implementation of our approach for other systems could require adapting our approach to other kinds of knowledge. In particular, adaptations could be proposed to revise knowledge expressed in other formalisms than production rules. In this context, proposing a specific approach for knowledge expressed by fuzzy logic based on this logic mathematical properties (see [46]) could be particularly interesting.

As mentioned in Section 3.4, ensuring that the evaluation function is in total adequacy with the user needs and that the problem instance sample used by the revision process is perfectly representative of all instances of the considered problem allows to guarantee that the knowledge obtained after revision is better than the initial one. In practice, it is difficult to ensure these two points. Concerning the representativeness of the sample, in Section 3.2.1, we proposed a method to automatically select a sample of problem instances. This method is based on the utilisation of a clustering technique. A key point of our method, in addition to the choice of the clustering algorithm, concerns the choice of the measure set used to characterise the problem instances. If the measure set is not pertinent, the clustering will be bad, and thus the select problem instance sample will be not representative of all the problem instances. Different methods proposed in the literature can be used to evaluate the measure set [11, 47]. Concerning the evaluation of the system performances, we already stated the difficulty of designing such performance function (Section 3.3.2). Indeed, if the user can easily determine if a given result is good enough, the lack of formalisation of its needs make the design of the evaluation function complex. Thus, an interesting future

work will consist in developing methods to help users design this function. A first approach that could be used to face this problem consists in directly using machine learning techniques. Thus, a sample of solved instances of the considered problem would be proposed to an expert. This one would give an effectiveness mark, an efficiency mark and a global performance mark to each of the results. These marks would be then used to learn an effectiveness function, an efficiency function as well as a performance function depending on the two previous marks. A second approach, more complex, could consist in designing the performance function thanks to an active learning. Thus, it could be interesting to use as a base the approaches presented in [48] and in [49]. The system would present several samples of results (resolved with different knowledge sets) to the expert. This one could define which one contains the best results and add commentaries about them through a dedicated interface. The system would then use these commentaries to refine the effectiveness function, the efficiency function and the performance function and to choose new result samples to present to the expert. In that case, the learning would be the result of a form of participatory design, emerging from the dialogue between the expert and the system.

Acknowledgements

The authors wish to thank Laurence Jolivet and Julien Gaffuri from the COGIT laboratory for their participation to the experiments.

References

[1] E. Feigenbaum, The art of artificial intelligence 1: Themes and case studies of knowledge engineering., Tech. rep., Stanford University, Department of Computer Science (1977).

[2] A. Newell, H. Simon, The logic theory machine, IRE Transactions on Information Theory 2(3) (1956) 61–79.

[3] S. Kirkpatrick, C. Gellatt, V. M.P., Optimization by simulated annealing, Science 220 (1983) 671–680.

[4] J. Holland, Adaptation in Natural and Artificial Systems, Ann Arbor, 1975.

[5] S. Minton, Quantitative results concerning the utility of explanation-based learning, Artificial Intelligence 42 (1990) 363–392.

[6] T. Mitchell, R. Keller, S. Kedar-Cabelli, Explanation-based generalization: a unifying view, Machine Learning 1 (1986) 45–80.

[7] R. aler, D. Borrajo, P. Isasi, Using genetic programming to learn and improve control knowledge, Artificial Intelligence 141 (2002) 29–56.

[8] T. Mitchell, P. Utgoff, R. Banerji, Learning by experi-mentation: acquiring and refining problem-solving heuristics, Machine Learning 1 (1983) 163–190.

[9] Y. Xu, A. Fern, S. Yoon, Learning linear ranking functions for beam search with application to planning, Journal of Machine Learning Research 10 (2009) 1571–1610.

[10] P. Taillandier, Knowledge diagnosis in systems based on an informed tree search strategy: application to cartographic generalisation, in: CSTST Student Workshop, 2008.

[11] P. Taillandier, Révision automatique des connaissances guidant lexploration informée darbres détats. application au contexte de la généralisation de données géographiques, Ph.D. thesis, Thèse de luniversité Paris-Est, Laboratoire COGIT (2008).

[12] L. Carbonara, D. Sleeman, Effective and efficient knowledge base refinement, Machine Learning 37 (1999) 143–181.

[13] A. Ginsberg, S. M. Weiss, P. Politakis, Automatic knowledge base refinement for classification systems, Artificial Intelligence 35 (1988) 197–226.

[14] M. Atzmueller, J. Baumeister, Introspective subgroup analysis for interactive knowledge refinement, in: Proceedings of the 19th Intl. Florida Artificial Intelligence Research Society Conference, 2006.

[15] D. Ourston, R. Mooney, Changing the rules: A comprehensive approach to theory refinement, in: Proceedings of the Eighth National Conference on Artificial Intelligence, 1990.

[16] G. Webb, Dlgref2: Techniques for inductive rule refinement, in: IJCAI Workshop W16: Machine Learning and Knowledge Acquisition, 1993.

[17] J. Pitrat, An intelligent system must and can observe its own behavior, in: COGNITIVA, 1991, pp. 119–128.

[18] P. Taillandier, C. Duchêne, A. Drogoul, Using belief theory to diagnose control knowledge quality. application to cartographic generalisation, in: RIVF, 2009.

[19] P. Mitra, C. Murthy, S. Pal, Unsupervised feature selection using feature similarity, in: IEEE Transactions on Pattern Analysis and Machine Intelligence, 2002.

[20] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via the em algorithm (with discussion), Journal of the Royal Statistical Society 39 (1977) 1–38.

[21] A. Jain, R. Dubes, Algorithms for Clustering Analysis, Printice-Hall, 1988.

[22] L. Xu, A. Krzyzak, E. Oja, Competitive learning for clustering analysi, rbf net and curve detection, IEEE Trans. Neural Networks 4 (1993) 636–649.

[23] L. Deng-Feng, C. Guo-Hong, H. Zhi-Gang, Linear programming method for multiattribute group decision making using if sets, Information Sciences 180(9) (2010) 1591–1609.

[24] J. Figueira, V. Mousseau, B. Roy, Electre methods, in: Multiple Criteria Decision Analysis: State of the Art Surveys, -, 2005.

[25] Y. Kim, O. de Weck, Adaptive weighted sum method for multiobjective optimization: a new method for pareto front generation, Structural and Multidisciplinary Optimizations 31 (2006) 105–116.

[26] P. Taillandier, Automatic knowledge revision of a generalisation system, in: workshop on generalisation and multiple representation, ICA, Moscow, Russia, 2007.

[27] J. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers Inc., 1993.

[28] W. Cohen, Fast effective rule induction, in: ICML, 1995.

[29] F. Glover, Tabu search, Journal on Computing 1 (1989) 190–206.

[30] L. Breslow, D. Aha, Simplifying decision trees: A survey, The Knowledge Engineering Review 12(1) (1997) 1–40.

[31] T. Roth-Berghofer, I. Iglezakis, Six steps in case-based reasoning:towards a maintenance methodology for case-based reasoning systems., in: Proceedings of the 9th German Workshop on Case-Based Reasoning,, 2001.

[32] K. Brassel, R. Weibel, A review and conceptual framework of automated map generalisation, International Journal of Geographical Information Systems 2 (3) (1988) 229–244.

[33] R. Weibel, S. Keller, T. Reichenbacher, Overcoming the knowledge acquisition bottleneck in map generalization: the role of interactive systems and computational intelligence, in: 2nd COSIT conference, 1995.

[34] A. Ruas, C. Duchêne, A prototype generalisation system based on the multi-agent system paradigm, in: W. Mackaness, A. Ruas, L. Sarjakoski (Eds.), Generalisation of Geographic information: cartographic modelling and applications, Elsevier Ltd, 2007, Ch. 14, pp. 269–284.

[35] S. Mustière, Cartographic generalization of roads in a local and adaptative approach: a knowledge acquisition problem, international journal of geographical information science 19 (8–9) (2005) –, fisher P., Gahegan M., Lees B. (ed.).

[36] T. Kilpelinen, Knowledge acquisition for generalization rules, Cartography and Geographic Information Science 27(1) (2000) 41–50.

[37] D. Burghardt, M. Neun, Automated sequencing of generalisation services based on collaborative filtering, in: 4th International Conference GIScience, 2006.

[38] A. Ruas, A. Dyevre, D. Duchêne, P. Taillandier, Methods for improving and updating the knowledge of a generalization system, in: Autocarto, 2006.

[39] A. Ruas, Modèle de généralisation de données géographiques a base de contraintes et d'autonomie, Thèse de doctorat en informatique,

spécialité science de l'information géographique, Université de Marne la Vallée, laboratoire COGIT (1999).

[40] M. Barrault, N. Regnauld, C. Duchne, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas, R. Weibel, Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization, in: 20th international conference of cartography, Vol. 3, Beijing, Chine, 2001, pp. 2110–2116.

[41] A. Boffet, S. Rocca Serra, Identification of spatial structures within urban block for town qualification, in: International Cartographic Conference, Vol. 2, 2001, pp. 1974–1983.

[42] S. Steiniger, T. Lange, D. Burghardt, R. Weibel, An approach for the classification of urban building structures based on discriminant analysis techniques, Transactions in GIS 12(1) (2008) 31–59.

[43] J. Gaffuri, J. Trévisan, Role of urban patterns for building generalisation: An application of agent, in: Workshop on Generalisation and Multiple representation, 2004.

[44] A. Ruas, A method for building displacement in automated map generalisation, international journal of geographical information sciences 12 (8) (1998) 789–803.

[45] L. Deng-Feng, An approach to fuzzy multiattribute decision making under uncertainty, Information Sciences 169(1-2) (2005) 97–112.

[46] S. Gottwald, Mathematical fuzzy logic as a tool for the treatment of vague information, Information Sciences 1972 (2005) 41–71.

[47] L. Molina, L. Belanche, A. Nebot, Feature selection algorithms: A survey and experimental evaluation, Tech. rep., Universitat Politcnica de Catalunya, Barcelona, Spain (2002).

[48] S. Christophe, Creative cartography based on dialogue, in: In proceedings of AutoCarto, 2008.

[49] P. Taillandier, J. Gaffuri, Objective function designing led by user preferences acquisition, in: International Conference on Information Technology and Applications, Hanoi, Vietnam, 2009.