Heuristic revision by Heuristic Space Exploration

Patrick Taillandier

Heuristic Revision by Heuristic Space Exploration

Patrick Taillandier
IRD, UMI UMMISCO 209, 32 avenue Henri Varagnat, 93143 Bondy, France
IFI, MSI, UMI 209, ngo 42 Ta Quang Buu, Ha Noi, Viet Nam
COGIT IGN, 2/4 avenue Pasteur, 94165 Saint-Mandé, France
patrick.taillandier@gmail.com

Abstract

Heuristics are often used to solve complex problems. Indeed, such problem-specific knowledge, when pertinent, helps to efficiency find good solutions to complex problems. Unfortunately, acquiring and maintaining a heuristic set can be fastidious. In order to face this problem, a approach consists in revising the heuristic sets by means of experiments. In this paper, we are interested in a specific revision method of this type based on the exploration of the heuristic space. The principle of this method is to revise the heuristic set by searching among all possible heuristics the ones that maximise an evaluation function. In this context, we propose a revision approach, dedicated to heuristics represented by production rules, based on the reduction of the search space and on a filtered local search. We present an experiment we carried out in an application domain where heuristics are widely used: cartographic generalisation.

1. Introduction

Since the beginning of artificial intelligence, problem solving is among the central topics. Thereby, the first artificial intelligence program, the logic theorist [10], was dedicated to the proving of mathematical theorem. This program uses heuristics, i.e. problem-specific knowledge, in order to guide the search for a solution.

Nowadays, heuristics are widely used in problem solver systems. When pertinent, they allow to find with efficiency a good solution to the problems. However, defining problem-specific knowledge can be complex. Edward Feigenbaum formulated this problem in 1977 as the knowledge acquisition bottleneck problem. This definition problem arises from the fact that expert knowledge is rarely formalised and is not easily translatable into a formalism usable by computers. Another drawback of problem solver systems based on heuristics concerns the evolution of the systems. Indeed, the heuristics have to be readapted at each evolution of the system, in particular when new elements (e.g. new actions to apply to find the solution) are integrated in the system. Thus, it is interesting to integrate, in the system, methods allowing the system to revise itself the heuristics by means of experiments. Several works were interested in the development of such methods [6, 9]. Some of them proposed to revise the heuristics by exploring the heuristic space [13]. In this context, the revision problem can be formalised as a search problem in which the objective is to find the heuristic set that maximises an evaluation function.

The work presented in this paper deals with the problem of exploration of the heuristic space, i.e. of the whole possible definable heuristics. We propose a search approach based on the reduction of the search space and on a filtered local search.In Section 2, we introduce the general context in which our work takes place. Section 3 is devoted to the presentation of our approach. Section 4 describes an application of our approach to cartographic generalisation. In this context, we present a real case study that we carried out as well as its results. Section 5 concludes and presents the perspectives of this work.

2. Context

2.1. Problem solver system

Many real world problems can be expressed as optimisation problems. The goal, in this kind of problems, 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 action application, the state of an entity that maximises an evaluation function. Many approaches were proposed to solve problems of this kind. Our work is dedicated to systems that solve it by exploring a state tree with the help of heuristics. The passage from a state to another corresponds to the application of an action. Such systems are often used

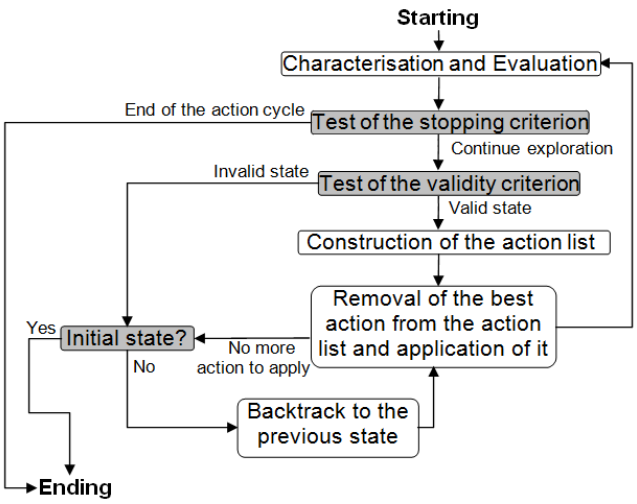

Figure 1. Action cycle

for real world problems thanks to their efficiency. In Figure 1, we present a classical action cycle for these systems based on informed (i.e. utilisation of heuristics) depth-first exploration of state trees.

The action cycle begins with the characterisation of the current state of the entity and its evaluation. Then, the system tests if the current state is good enough or if it is necessary to continue the exploration towards others 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 from which it is interesting to continue the exploration. If the state is not valid, the entity backtracks to its previous state; otherwise the system constructs a list of actions to try. If the actions list is empty the entity backtracks to its previous state; otherwise the system chooses the best action, and applies it. Then it goes back to the first step. The action cycle ends when the stopping criterion is met or when all actions have been applied for all valid states.

In this paper, we are interested in the heuristics used to construct the action list. We impose that these heuristics are expressed by production rules. Indeed, in many real world applications, heuristics are expressed by production rules. The interest of this kind of heuristic representation is to be easily interpretable by domain experts and thus to facilitate the heuristic validation and update. For each action, a set of production rules is defined. The set of rules allows to determine, for each possible state, if the action has to be tried and if so, with which weight. The higher the weight, the higher priority the action has (and thus will be tried first). The weight is an integer between 0 and WEIGHT_MAX (0: the action is not proposed for the state, WEIGHT_MAX: the action is applied first). A measure set is defined per action. Several actions can depend of the same measure set.

2.2. Automatic revision of heuristics

The question of automatic revision of heuristics has already been studied in the literature, in particular in the context of the speedup learning. The speedup learning seeks to improve the efficiency of problem solving systems by using experience.

Among speedup learning works, many propose to use domain knowledge in order to facilitate the learning [6, 8, 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 learnt knowledge is dependant of the quality of the domain theory. Thus, an incomplete or incorrect domain theory can cause numerous problems. In this paper, we make the assumption that we do not have a strong domain theory, just initial heuristics of uncertain quality. Thus, it will not be possible to directly use these works.

Other speedup learning works propose to revise heuristics without using domain theory. The most famous is the LEX system [9]. This system learns the application conditions of actions by analysing previously solved problem instances. The LEX system has some limitations, especially concerning the management of noisy data and the learning of disjunctive concepts.

In [13], we proposed a revision approach taking place in the continuity of the latter work. We proposed to revise the action application domains, represented by production rules, by exploring the space of the action application rules.

This approach is composed of two stages:

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

In this paper, we are interested in the *analysis* stage. We assume that a sample of problem instances has already been solved. We propose an approach, aiming at revising the heuristic set, based on the exploration of the heuristic space by using this sample of problem instances.

3. Proposed Approach

3.1. General approach

A difficulty of the heuristic space exploration is the size of the search space. In order to limit this size, we proposed in [13] to use experience (sample of resolved instances of the problem) to partition the measure sets associated to each action, into areas admitting a priori a same behaviour. An

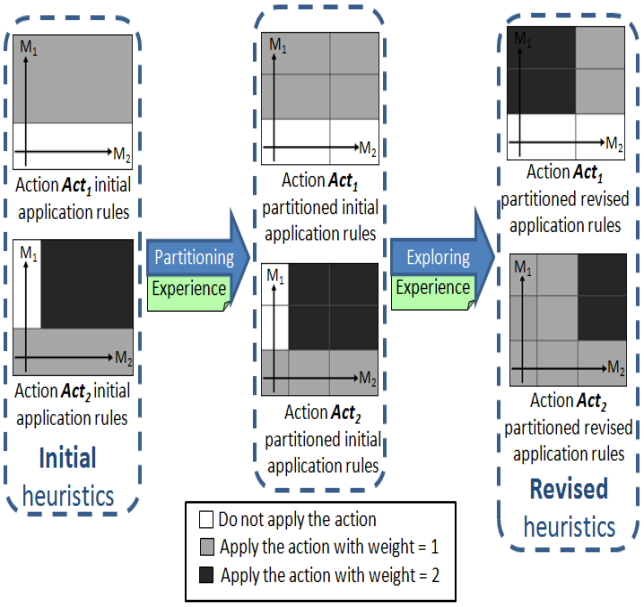

Figure 2. Revision approach

area corresponds to the premise of a rule (a set of conditions concerning the measures), and the behaviour associated to this area corresponds to the conclusion of the rule (the weight of the action). Thus, the exploring problem consists in assigning the best conclusion to each obtained areas. As illustrated Figure 2, the approach proposed in [13] is composed of two steps: the first one consists, for each action, in partitioning the measures set linked to it in areas that admit a priori a homogenous behaviour (conclusion of the rule); the second step consists in searching the best conclusion to assign to each area.

In this paper, we propose to use the same general approach. In this context, we propose a new conclusion assignment method. This method is based on the use of example sets in order to extract information from experience. In the next section, we describe the process used to build these example sets. Section 3.3 is dedicated to the description of our conclusion assignment search method.

3.2. Construction of the example sets

In order to extract information from experience, we propose to build an example set describing the "ideal" behaviour of the actions. 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 to the actions. The label is the behaviour assessed as the good one for this experienced state, i.e. is the action that should be tested in priority.

The construction of the example sets is achieved by analysing the state trees obtained during the *exploration*

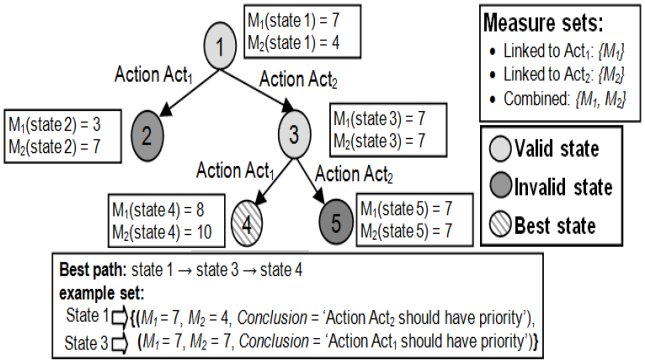

Figure 3. Example of a built example set

stage (each state tree resulting from the resolution of one sample instance). The method used for the construction of the example sets has been described in [12]. 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. The next step consists in analysing each state of each best path. If a state belongs to a best path and if the application of an action leads to another state of the same best path, the ideal behaviour is "this action should have priority". Figure 3 gives a simplified example of example sets.

3.3. Weight assignment problem

3.3.1 Problematic of the weight assignment space exploration

As stated in Section 3.1, the last step of our revision approach consists in searching the conclusion assignment set that maximises an evaluation function. This evaluation function translates the user needs toward the problem resolution system. In particular, this function defines the balance between the efficiency (speed to carry out the resolution of a problem instance) and the effectiveness (quality of the found problem instance solution) of the system. The definition of this function is a key point of our approach. Thus, it is essential that this one is in adequacy with the user needs. We note $Perf(S_H, I)$ the function that is used to estimate the performance of the system S when using a heuristic set H for the resolution of a sample of problem instance I. There is no generic $Perf(S_H, I)$ performance function. This one has to be specifically defined for each application.

Let us remind that we deal with the revision of the action application domains. Each action has a rule set, which defines the weight of the action for each value of its measure set (see Section 2.1). The difficulty of the knowledge revision process comes from the distributed nature of this

knowledge. Actually, if the application rule sets of each action are not dependent on each other in their expression (each action has a rule set which only depends on its own measure set), the results (the weight) can only be analysed if compared to the weight of the other actions. Therefore, it is not possible to search the best assignment of weights of each action independently: we have to take into account all actions at the same time. The size of the search space is then equals to $(WEIGHT\_MAX + 1)^{total\_nb\_of\_areas}$. Thus, the size of the search space is most of time too high to carry out a complete search.

3.3.2 Filtered Local search

In order to face the exploration problem, we propose to use a local search approach. The principle of these search methods is to start the search from an initial solution and to try to improve it by successive movements in the solution space. The movement in the solution space corresponds to a transition from a solution to another one belonging to the neighbourhood of the former one. The interest to use a local search in our context is to take advantage of the initial heuristics that constitutes, most of the time, a good initial solution. In the literature, many local search methods were proposed [5, 4]. In this paper, we proposed to use one of these local search methods but as well to improve its efficiency by using information contains in the example set. Thus, we proposed to use this information to filter the considered neighbourhood at each search step. Our filtering method is composed of four steps that are detailed hereafter.

Step 1

This step is the classical movement list building of the local search. The building of the movement list depends of the used local search method.

Step 2

This step consists in characterising the state of each area. The goal of this characterisation is to use the local information to detect the conflicts between the ideal behaviour of the actions (given in the example set) and their real behaviour (the current weight assignments). We propose to characterise the state of each area by the number of false positives, of false negatives, of true positives, and of true negatives. Figure 4 illustrates how these numbers are computed: for each example of the example set (here, E_a and E_b), the ideal behaviour of the actions (for E_a, Act_2 should have priority; for E_b, Act_1 should has priority) is compared to their current behaviour (for E_a, Act_2 has priority; for E_b, Act_2 has priority). If an action has priority and should have priority, the area that allows the action to have priority meets one true positive. If this action should not have priority, the area that allows the action to have

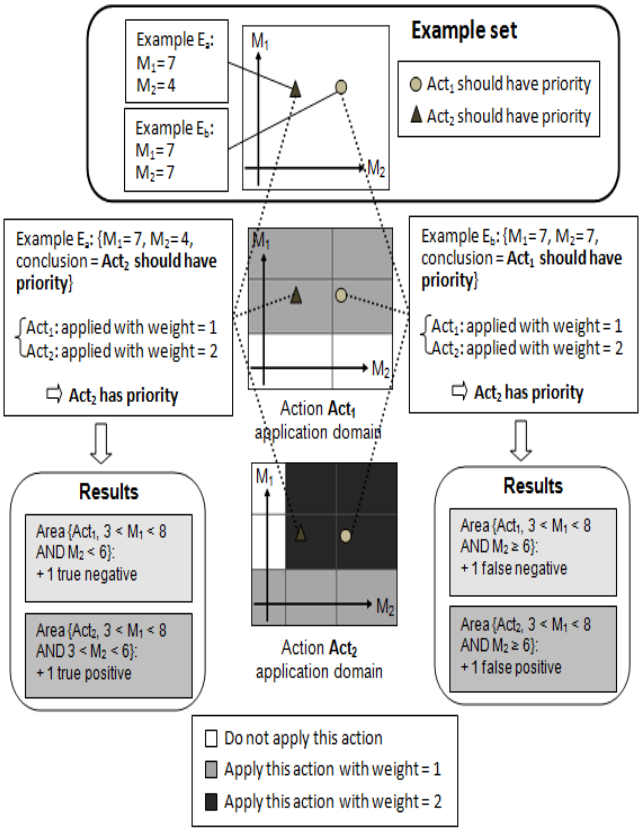

Figure 4. Area state characterisation

priority meets one false positive. In the same way, if an action does not have priority and should not have priority, the area that allows the action to have priority meets one true negative. If this action should have priority, the area that allows the action to have priority meets one false negative.

Step 3

This step consists in computing the quality a priori of the possible movements from the local information computed in the last step.

Each area that is directly or indirectly concerned by a movement gives a mark to it. An area is directly concerned by a movement if this one implies to modify the weight assigned to the area. An area is indirectly concerned by a movement if the area shares examples with an area directly concerned by the movement. Two areas share an example if the example state is situated in the two areas. For example, in Figure 5, the areas (Act_1, $3 < M_1 < 8$ AND $M_2 < 6$) and (Act_2, $3 < M_1 < 8$ AND $3 < M_2 < 6$) share the examples E_a (the triangle).

The mark given by an area is a real between 0 and 1 (0: the area strongly advises against the movement; 1: the

area strongly recommends the movement). It depends on the presumed evolution of the success and failures that the area could meet if the movement is carried out.

If a movement implies to increase the weight of an area, the number of false negatives met by this one should decrease. An area, which shared examples with the area, should see its number of false positives decrease but could also see its number of true positives decrease. In the same way, if a movement implies to decrease the weight of an area, the number of false positives met by this one should decrease. An area, which shared examples with the area, should see its number of false negatives decreasing but could also see its number of true negatives decreasing.

The quality *a priori* of a movement is equal to the mean of the marks given by the areas weighted by their importance.

Let $area$ be an area, we note $fneg(area)$, its number of false positives, $fpos(area)$ its number of false negatives, $tneg(area)$ its number of true negatives, and $tpos(area)$ its number of true positives. We note, as well, $weight(area)$ the current weight assigned to the area and $weight(area, mvt)$ the weight assigned to the area after the application of the movement mvt. The marks and the importance given by the areas to the movements are the following:

- Mark given by an area $area_d$ directly concerned by a movement mvt:
  A. If $weight(area_d) < weight(area_d, mvt)$:
  - $Mark(area_d, mvt) = \frac{fneg(area_d)}{fneg(area_d) + tneg(area_d)}$
  - $Importance(area_d, mvt) = fneg(area_d) + tneg(area_d)$

  B. If $weight(area_d) > weight(area_d, mvt)$:
  - $Mark(area_d, mvt) = \frac{fpos(area_d)}{fpos(area_d) + tpos(area_d)}$
  - $Importance(area_d, mvt) = fpos(area_d) + tpos(area_d)$

- Mark given by an area $area_i$ indirectly concerned by a movement mvt and sharing examples with an area $area_d$ directly concerned by mvt:
  C. If $weight(area_i) > weight(area_d, mvt)$ and $weight(area_i) \leq weight(area_d)$:
  - $Mark(area_i, mvt) = \frac{fneg(area_i)}{fneg(area_i) + tneg(area_i)}$
  - $Importance(area_i, mvt) = fneg(area_i) + tneg(area_i)$

  D. If $weight(area_i) < weight(area_d, mvt)$ and $weight(area_i) \geq weight(area_d)$:
  - $Mark(area_i, mvt) = \frac{fpos(area_i)}{fpos(area_i) + tpos(area_i)}$
  - $Importance(area_i, mvt) = fpos(area_i) + tpos(area_i)$

Once each concerned area has given its mark to a movement, the quality *a priori* of the movement is computed.

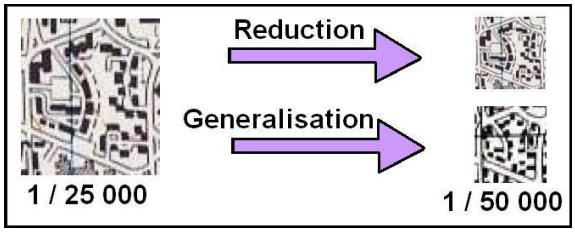

Figure 5. Cartographic Generalisation

Step 4

This last step consists in filtering the movements according to their quality *a priori*. If the quality *a priori* of a movement computed during the last step is higher than the value of a filtering coefficient the movement is kept; otherwise, it is removed from the movement list. The filtering coefficient is a real between 0 and 1. This filtering allows to limit the number of tested solutions. The lower the value of this filtering coefficient, the less solutions will be tested, thus the faster the system will converge towards a solution. However, a high value for the filtering coefficient can have for consequence to miss good solutions.

4. Application to cartographic generalisation

4.1. Automatic cartographic generalisation

We apply our heuristic space exploration approach in the domain of cartographic generalisation. Cartographic generalisation is the process that aims at simplifying vector geographic data to suit the scale and purpose of a map. Figure 5 gives an example of cartographic generalisation.

The automation of the generalisation process is an interesting industrial application context which 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 creating one's own map increases the needs of reliable and effective automatic generalisation processes. One classical approach to automate the generalisation process is to use a local, step-by-step and knowledge-based method [2, 14]: each vector object of the database (representing a building, a road segment, etc.) is transformed by application of a sequence of generalisation algorithms realising atomic transformations. The choice of the applied sequence of algorithms is not predetermined but built on the fly for each object according to heuristics and to its characteristic.

4.2. The generalisation system

The generalisation system that we used for our experiment is based on the AGENT model [1, 11]. It follows the

specification presented in Section 2.1. It generalises a geographic object or a group of geographic objects by means of an informed tree search strategy. Each state represents the geometric state of the considered geographic objects and is evaluated by a satisfaction function, which translates the respect of cartographic constraints by the geographic objects. The actions cycle used is the one presented in Figure 1. The weight of the actions is ranged between 0 and 5.

4.3. Case study

The actual case study that we carried out concerned the generalisation of geographic object of the type "building group" [3]. 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 building group generalisation is an interesting case study because it is not yet well mastered and because it is very time consuming.

We defined five actions for the building group generalisation as well as two heuristic sets. The first one was defined by a generalisation expert (H_{expert}). The second one corresponds to the "basic" case where all actions are proposed for all states (H_{basic}). The revision of the "expert" heuristic set corresponds to the classical scenario of knowledge revision where we have a good initial rule base that we want to refine. The revision of the "basic" heuristic set corresponds to the scenario where we have no knowledge on the generalisation of the considered type of object.

50 building groups from the French city of Orthez were automatically selected among more than 300 available for the revision process.

Concerning the algorithm used for the local search, we used the well-established tabu search algorithm [4].

For the $Perf(S_H, I)$, we defined a function favouring the effectiveness of the system (computed according to the mean satisfaction) over its efficiency (computed according to the mean number of visited states).

Concerning the test protocol, we tested the initial and the revised knowledge on a different area (the French city of Salies-de-Barn) than the one used for the revision. This area was composed of 200 building groups. In order to evaluate our filtering approach, we tested the local search method with and without the filtering method. Concerning the filtering method, two values of the filtering coefficient were tested: 0.1 (weak filtering) and 0.5 (strong filtering).

4.4. Results

Table 1 shows the results on the test area with the different heuristic sets. The two heuristic sets were improved, even the expert knowledge set that was already good. Indeed, concerning H_{basic}, we obtain close results in terms of effectiveness (slightly less good for the revised version), but far better results in terms of efficiency for the revised version: the revision allowed to divide the mean number of visited states per generalisation by a factor 10.

Table 1. System performances on the test area

Heuristic set	$Perf(S_H, I)$	
	Initial	Revised
H_{basic}	0.761	0.784
H_{expert}	0.783	0.788

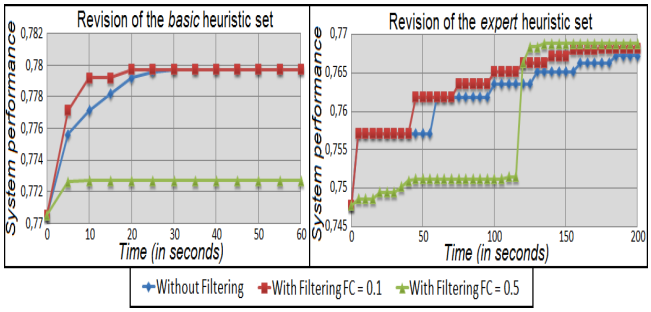

Figure 6. Revision results

Considering H_{expert}, the initial and the revised versions obtained equivalent results in terms of effectiveness, but the revised version obtained better results in terms of efficiency: the revision allowed to divide the mean number of visited states per generalisation by a factor 2.

An interesting point concerns the way the initial heuristic set is taken into account. The revised heuristic set obtained from the revision of H_{expert} obtained better results than the one obtained from the revision of H_{basic}. This result shows the interest of taking into consideration the initial heuristic set for the revision process.

Concerning the contribution of the neighbourhood filtering, the results (cf. Figure 6) show that the introduction of the filtering phase with the filtering coefficient equals to 0.1 allowed to converge more quickly toward a good solution for both heuristic sets. Concerning the results obtained with the filtering coefficient equals to 0.5, they shows that it allowed to find a better solution for the H_{basic} but not for H_{expert}. Indeed, for this last heuristic set, the filtering had for consequence a high deterioration of the quality of the best found solution. Thus, it is important to pay intention to the filtering coefficient used. A high value of it can have for consequence to miss good solutions.

5. Conclusion

In this paper, we propose a revision approach, dedicated to heuristics represented by production rules, based on the reduction of the search space and on a filtered local search. We showed the efficiency of our revision approach as well as the interest of the neighbourhood filtering on a actual case study.

If we revised the action application knowledge, we did not try to revise others pieces of knowledge like the validity criterion or the actions cycle ending criterion. Some adaptation of our approach could be proposed to revise as well this kind of knowledge. In the same way, adaptations could be proposed to revise knowledge expressed in others formalisms than production rules.

A key point of our approach is the $Perf(S_H, I)$ function used to evaluate the performance of the system. Designing such a performance function to translate the system user needs can be very complex. Thus, an interesting future work will consist in developing methods to help users design this function.

References

[1] M. Barrault, N. Regnauld, C. Duchene, K. Haire, C. Baeijs, Y. Demazeau, P. Hardy, W. Mackaness, A. Ruas, and R. Weibel. Integrating multi-agent, object-oriented, and algorithmic techniques for improved automated map generalization. In *ICC*, volume 3, pages 2110–2116, 2001.

[2] K. Brassel and R. Weibel. A review and conceptual framework of automated map generalisation. *IJGIS*, 2(3), 1988.

[3] J. Gaffuri and J. Trevisan. Role of urban patterns for building generalisation: An application of agent. In *workshop on gen. and multiple repr.*, 2004.

[4] F. Glover. Tabu search. *Journal on Computing*, 1989.

[5] S. Kirkpatrick, C. Gellatt, and M. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] J. Laird, P. Rosenbloom, and A. Newell. Chunking in soar: the anatomy of a general learning mechanism. *Machine Learning*, 1:11–46, 1986.

[7] S. Minton. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–392, 1990.

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

[9] T. Mitchell, P. Utgoff, and R. Banerji. Learning by experimentation: acquiring and refining problem-solving heuristics. *Machine Learning*, 1, 1983.

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

[11] A. Ruas and C. Duchene. A prototype generalisation system based on the multi-agent system paradigm. In W. Mackaness, A. Ruas, and L. Sarjakoski, editors, *Generalisation of Geographic information: cartographic modelling and applications*, chapter 14, pages 269–284. Elsevier Ltd, 2007.

[12] P. Taillandier. Automatic knowledge revision of a generalisation system. In *workshop on gen. and multiple repr.*, 2007.

[13] P. Taillandier, C. Duchene, and A. Drogoul. Knowledge revision in systems based on an informed tree search strategy: application to cartographic generalisation. In *CSTST*, 2008.

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