Reasoning about Lava effusion: from Geographical Information Systems to Answer Set Programming*

Isabella Cattinelli[1], Maria Luisa Damiani[2] and Andrea Nucita[2]

[1] M^2AG - Dipartimento di Scienze dell'Informazione
Università degli studi di Milano. Milan, I-20135 Italy
isabella@mag.dsi.unimi.it
http://mag.dsi.unimi.it/
[2] Dipartimento d'Informatica e Comunicazione
Università degli studi di Milano. Milan, I-20135 Italy
<damiani,nucita>@dico.unimi.it
http://www.dico.unimi.it/

Abstract. This article describes our implementation in Answer Set Programming of a reasoning system that models the flow of lava in volcanic eruptions. Our system can be employed in the validation of evacuation plans. To demonstrate the feasibility of such approach, we adopt a simplified yet realistic model of how lava flows, and apply it to altitude data from the Etna volcano.

1 Introduction

This article reports on our experience of applying Answer Set Programming (ASP) to reasoning about geographical data. We adopt the framework outlined by [Osorio & Zepeda, 2004], which, to the best of our knowledge, was the first work addressing this issue. We consider the problem of generating and validating an evacuation plan for a community living near an active volcano that now is erupting. Planning in Answer Set Programming is by now well-studied and understood; it can be found, e.g., in [Baral, 2003], which we adopted for our experiments. The novel question for ASP we would like to answer here is the following: *can the lava flowing from the eruption cross the evacuation path under some circumstances (which we have no control on)?*

Using ASP for the implementation of a system that generates and validates evacuation plans brings obvious benefits in terms of adapting the code to the fast-changing conditions. Another goal of this work is to apply Answer Set Programming solvers in realistic scenario and check their performance and usability vis-a-vis ad hoc solutions written using imperative languages. However, we hope

* Work supported by i) MIUR COFIN project *Rappresentazione e gestione di dati spaziali e geografici in WEB.* and ii) the Information Society Technologies programme of the European Commission, Future and Emerging Technologies under the IST-2001-37004 *WASP* and IST-2001-33058 *PANDA* projects.

that our solution will soon be of practical interest for emergency management and we are reaching out to geologists and risks managers. Indeed, the data used in the experiments here described come from an extensive set of data collected on the Etna volcano in Sicily, for which a wealth of present and historical data is available.

The available data are, essentially, of geographical nature. Each instance we consider consists of

- an integer matrix describing the orography of the considered area,
- the point where the eruption is located,
- the point where the community that needs to be evacuated is, and
- some parameter about the lava front: its height and the length it can go.

We see the motion of the lava front (which we will call *lava motion*) in terms of a partially-constrained, non-deterministic visit of the surrounding. The question we would like to address is the following: will the lava ever reach the considered evacuation point?

Of course, to answer such a question we need, beside the instance, a model of how the lava moves, which is described in Section 3. Our model of how the lava flows on the ground, which is illustrated in Section 4, is somewhat simplified w.r.t. those currently used in lava flow simulation and control; however, it is detailed enough to constitute a valid proof-of-concept that ASP can be applied to this context.

The only other work that addresses ASP in the context of GIS is that of [Osorio & Zepeda, 2004] from which we have taken the general framework and motivations for our work. One key point in Osorio and Zepeda's approach is to *split* the representation of the geography and of the lava flow from the representation of the planning activity needed to draw up evacuation plans.

To avoid the inherent complexity of reasoning with geographical data within ASP, Osorio and Zepeda adopt Constraint Logic Programming for the geographical part, and ASP for the planning part. To give an overall semantics to this combination, they propose the notion of semantic content, by which each module is given an external semantics, which will be combined with that of the other model *externally*. This approach has several practical advantages, but it comes at the cost of introducing an external semantics and study its compositional properties.

Our approach is to develop both the above modules *into one ASP program*. Therefore, the answer sets of the resulting program are the semantics of the system. Whether this approach is realistic, it is subject of experimentation and benchmark on realistic instances which will be outlined in the final part of the article, with preliminary but encouraging results.

2 Answer Set Programming

In this paper we describe Answer Set Programs starting with the $DATALOG^{\neg}$ syntax which is now standard for deductive databases and more restricted than

traditional logic programming; for a broader introduction and historical perspective, the reader may refer to the overview by [Marek & Truszczyński, 1999] for a through introduction and discussion. In the following, we will implicitly consider the ground version of $DATALOG^{\neg}$ programs. A rule ρ is defined as usual, and can be seen as composed of a conclusion $head(\rho)$, and a set of conditions $body(\rho)$. The latter can be divided into positive conditions $pos(\rho)$ each one of the form A, and negative conditions $neg(\rho)$, each one of the form $not\ A$. In what follows, Π will denote a generic logic program. For the sake of simplicity, we assume that no two rules in Π with the same conclusion have the bodies in subset relation.

The answer sets semantics for logic programs has been introduced by Gelfond and Lifschitz, first as *stable models* semantics in [Gelfond & Lifschitz, 1988] and later [Gelfond & Lifschitz, 1991] extended to handle explicit negation. Gelfond and Lifschitz take logic programs as sets of inference rules (more precisely, default inference rules). Alternatively, one can see a program as a set of constraints on the solution of a problem, where each answer set represents a solution compatible with the constraints expressed by the program. Consider for instance the simple program $\{q \leftarrow not\ p.\ \ p \leftarrow not\ q.\}$: the first rule is read as "assuming that p is false, we can *conclude* that q is true." This program has two answer sets. In the first one, q is true while p is false; in the second one, p is true while q is false.

A subset M of the Herbrand base B_Π of a $DATALOG^{\neg}$ program Π is an answer set of Π, if M coincides with the least model of the reduct Π^M of Π with respect to M. This reduct is obtained by deleting from Π all rules containing a condition $not\ a$, for some a in M, and by deleting all negative conditions from the other rules. Answer sets are minimal supported models, and form an antichain. Whenever a program has no answer sets, we will say that the program is *inconsistent*. Correspondingly, checking for consistency means checking for the existence of answer sets. In this article, Answer Set computation is used for model checking purposes, i.e., the construction of the answer set will correspond to the attempt to prove that one possible lava path indeed can reach the point of interest. Inconsistency of the program makes us conclude that the point of interest is safe from lava flow. To the contrary, each answer set will contain a description of the path the lava can follow to reach the point of interest. Such description can be passed on to human experts that can validate it or raise/decrease the estimate of its likelihood.

3 A model of the Eruption domain

In spite of the dangerousness of the volcanic region, there are numerous towns and important economic activities located nearby and even along the flanks of the volcano, so that a hazard map would be definitely needed. Nevertheless, the only hazard map that have been realized so far for the Mount Etna goes back to twenty years ago and is based on incomplete knowledge of past eruptions. The concept of risk here concerns specifically the impact of eruptions on human activities. For instance, the risk in a desert area in presence of a violent eruption is null. However, risk evaluation directly depends on hazard. The hazard is intended

as the probability that an area has to be affected by a certain natural event (e.g. lava flows or ash fall), regardless of the influence on human activities. Hence, in the proximity of a crater the hazard is very high, while the risk is null, because of the absence of human activities. Nevertheless, the hazard maps are crucial tools for the risk evaluation in the case of natural disasters.

In the last years, Computer Science played a crucial role in monitoring, analyzing and diffusing data about volcanoes activities and it became essential for building hazard maps. Nevertheless, a computer system is always based on mathematical or statistical models, and needs suitable collections of historic data to be tested. Unfortunately, such data are not always available for volcanoes since often a new lava emission covers the paths of the previous lava flows.

3.1 The Mount Etna Case

Mount Etna represents an important testing case, thanks to the large quantity of data on historic eruptions, with reliable informations starting as far back as 693 B.C.! Moreover, the continuous activity of this volcano is an important requirement for the testing of the hazard models, and makes mount Etna a very rare case in the world.

4 Models of volcanic activity in literature

Many approaches have been proposed for the assessment of the volcanic hazard. The challenging model that we have chosen as basis for the experimentation has been developed by [Felpeto et al., 2001] and applied so far only to the Lanzarote volcano (Canary islands).

In [Felpeto et al., 2001], a maximum slope model is proposed for the simulation of the lava flow. The rationale of that model is that the topography plays the major role on determining the path that the lava flow will follow. We will discuss this model in detail in the next section.

In [Felpeto et al., 2000] the authors identify the major hazard factors in Tenerife (Canary islands) as the lava flow (related to the basalt-effusive eruptions) and the ash fall (due to the salic-explosive eruptions). To generate the probability map corresponding to an effusive eruption, the authors simulate the area covered by the lava flow with the maximum slope model proposed in [Felpeto et al., 2001]. For the evaluation of the effect of a plinian eruption, with the related fall out of ash, an advection-diffusion model has been applied [Armienti et al., 1988]. The authors then use the informations provided by the application of those models for building the so-called *hazard zonation* map.

4.1 Felpeto et al. Lava flow model

Since in the case of mount Etna the eruptions are basically of basaltic-effusive type, we referred to the model in [Felpeto et al., 2001] for the simulation of the lava flows. According to that model, hazard maps are built simulating the paths

of the lava flows, with an algorithm that assumes that the topography mainly determines the path of the lava.

The data are represented as a DEM (Digital Elevation Model), in which the only attribute of a cell is the altitude over the sea level. Then, given a lava emission point (which belongs to one of the cell of the DEM), the algorithm returns a possible path of the lava flow. The procedure is repeated and every path is stored into a matrix, at the same resolution level of the DEM, in which the value of each cell represents how many times it has been invaded by the lava.

Hence, the determination of the probability of each point being covered by lava is obtained by computing several random paths by means of a Monte Carlo algorithm as discussed below. Moreover, the algorithm considers two basic assumptions:

- the paths cannot propagate upward, and
- the higher the slope, the higher the probability of the path passing there.

Let us consider a cell (say $i = 0$) where the lava flow is located. The probability that the flow enters one of the eight surrounding cells ($i = 1, 2, ..., 8$) is:

$$P_i = \frac{\Delta h_i}{\sum_{j=1}^{8} \Delta h_j} \tag{1}$$

where Δh_i is the difference in height between the cell where the flow is and each of the neighboring ones. In these differences a corrective factor is added to the value of the starting cell. This corrective factor takes into account the possibility of the lava flow of crossing small topographic barriers. Hence, Δh_i is evaluated as follows.

$$\begin{aligned} &\Delta h_i = h_0 + h_c - h_i \ \text{ if } \ (h_0 + h_c - h_i \geq 0) \\ &\Delta h_i = 0 \ \text{ if } \ (h_0 + h_c - h_i < 0) \end{aligned} \tag{2}$$

Equations (1) and (2) ensure that the probability that the lava flow propagates up-wards is null. The selection of the cell in which the flow will propagate is obtained by a Monte Carlo algorithm. Hence, given a random number $0 \leq t \leq 1$ and the sums $S_i = \sum_{j=1}^{i} P_j, \ \ i = 1, ..., 8$, the cell i will be covered by the lava flow if:

$$\begin{aligned} &S_{i-1} \leq t \leq S_i, \quad i = 1, ..., 8 \\ &S_0 = 0 \end{aligned} \tag{3}$$

The algorithm then terminates in the following three cases:

- the height of the cell covered by the lava is lower than the ones of the eight surrounding cells
- a fixed number of cells covered by the lava flow (that is the maximum flow length) is reached
- the cell covered by the lava is in the edge of the map.

5 The available data and the input format

Data pertaining Mount Etna have been acquired from different sources. For what concerns the Digital Elevation Model, it is based on the topographic data provided by a national volcanology research group (GNV-CNR). We used that high resolution DEM to obtain the morphological constraint of the lava flow simulation. In order to make data acceptable as input for the ASP program, we extracted several portions of the DEM, which is represented in a matrix form, and rewritten them as lists of points defined as follows:

$$p(i, h_i). \quad i = 1, ..., n \tag{4}$$

where i represents a number indicating the cell of the DEM, h_i represents the altitude of cell i, and n is the total number of cells of the considered portion of the DEM. In the experiments we considered 12 maps grouped in three types, depending on the total number of included cells. That is we made maps with 16, 64 and 256 cells and we used four maps of each type.

6 The ASP implementation

Let us describe the ASP program which computes the lava paths. The program, called *Lava Motion*, has been proposed in [Cattinelli, 2004] as a realistic benchmark for evaluating the performance of ASP solvers. At this point, however, we will show only the code, and the timings, for *Smodels* [ASP solvers] which we initially adopted for the development.

At each time T, the lava flows to a new point, according to the given flow model. For the sake of simplicity, we have first implemented the *simplest* flow model, which is sometimes called *water behavior*: among all reachable areas, lava will flow toward the one with the *steepest negative descent*. The selection of which point in the map will be reached next is operated by the `steepest_descent` predicate, which constitutes the heart of *Lava Motion* program.

The ternary relation `steepest_descent(From, To, T)`, with its obvious meaning, is defined having the following computation steps in mind. First, select all areas that are adjacent to the `From` point. Then, discard from the set those areas that are not reachable by the lava flow, because their average height cannot be exceeded by the front. Among all reachable points, the one whose average height is the lowest is selected. Since aggregate predicates are not currently supported by *Smodels* syntax, we had to introduce the `no_steepest_descent` predicate to compute the minimum height.

Intuitively, `no_steepest_descent(From, To, T)` will be true if there are other areas, reachable from `From`, which have a lower height than `To`. So, the area(s) for which `no_steepest_descent(From, To, T)` is false will be the one(s) for which `steepest_descent(From, To, T)` is true.

In order to avoid to compute paths containing cycles, we restricted lava movement imposing that an area cannot be flooded at time T if it was previously flooded. Therefore, when selecting the steepest descent, previously-visited

areas are not taken into account. Lava will go on flowing until it reaches a *local minimum*, where it stops.

```
% Lava Simulation v.10 of 12/4/03
%
% typical call: lparse mat_nxn startpoint.sm data.sm lava0.3.sm | smodels
%%% Problem data may be found in files 'mat_nxn' %%%
%
% - size of the matrix                    e.g. const n=10.
% - number of start points                e.g. const n_start_points=4.
% - front height                          e.g. const front height=5.
% - height values                         e.g. p(P, H) -> point P
%                                              has average height H
%%% In file "startpoint.sm" you can find
%
% - start point                           e.g. const start_point=7.
%
%%% In file "data.sm" you can find
%
% - length of lava path                   e.g. const l=10.
% - query                                 e.g. :- not flooded(5, l)
%                                         (will the lava reach point 5?)
%%% end data %%%

%%% domains %%%

point(1..n*n).

time(0..l).

% lava starts flowing at time 0 on point P.

flows(start_point, 0).

% WATER-LIKE MODEL: At time T, lava flows on point To if it was previously
% flowing on point From and the descent between the two points is the steepest.
flows(To, T) :- flows(From, T-1),
                time(T),
                adj(From, To),
                steepest_descent(From, To, T),
                not other_flows(To, T).

other_flows(P, T) :- flows(From, T-1),
                     adj(From, P),
                     adj(From, P1),
                     time(T),
                     P != P1,
                     flows(P1, T).

% flooded(P, T) if P was already reached by the lava in time T1, previous to T.
flooded(P, T) :- point(P),
                 time(T),
                 time(T1),
                 flows(P, T1),
                 T1 < T.

% tha lava is running down the volcano
running(T) :- flows(P, T), point(P), time(T).

% the lava has stopped in point P
stops(P, T) :- point(P), time(T), flows(P, T-1), not running(T).
stops(P, T) :- point(P), time(T), stops(P, T-1).


% the descent between P and A is maximum for P if it is
% greater than the one between P and each adjacent A1.
steepest_descent(From, To, T) :- adj(From, To),
                                 p(From, HF),
```

```
                                    p(To, HT),
                                    HT <= HF + front_height,
                                    time(T),
                                    not flooded(To, T),
                                    not no_steepest_descent(From, To, T).

no_steepest_descent(From, To, T) :- adj(From, To),
                                       adj(From, To_other),
                                       p(To, HT),
                                       p(To_other, HTO),
                                       HTO < HT,
                                       not flooded(To_other, T),
                                       time(T).


% P is adjacent to P1 if it is immediately above, or below,
% or right, or left of P1, respectively.
adj(P, P1) :- point(P), point(P1), P1 = P - n.
adj(P, P1) :- point(P), point(P1), P1 = P + n.
adj(P, P1) :- not at_fringe_dx(P), point(P), point(P1), P1 = P + 1.
adj(P, P1) :- not at_fringe_sx(P), point(P), point(P1), P1 = P - 1.
adj(P, P1) :- not at_fringe_dx(P), point(P), point(P1), P1 = P - n + 1.
adj(P, P1) :- not at_fringe_sx(P), point(P), point(P1), P1 = P - n - 1.
adj(P, P1) :- not at_fringe_dx(P), point(P), point(P1), P1 = P + n + 1.
adj(P, P1) :- not at_fringe_sx(P), point(P), point(P1), P1 = P + n - 1.

at_fringe_dx(P) :- P mod n == 0, point(P).
at_fringe_sx(P) :- P mod n == 1, point(P).
% end of program
```

Let's focus on the `other_flows` predicate. In an earlier version, it was defined as:

```
other_flows(P, T) :- time(T),
                     point(P),
                     point(P1),
                     P != P1,
                     flows(P1, T).
```

So, `other_flows` was instantiated on all points P of the input matrix, thus producing a huge number of ground rules. It is easy to observe, however, that lava, starting from point P, can only flow to one of the eight zones that are adjacent to P, and not to *every* point. Therefore, the size of the instantiation (and thus the problem's complexity) can be relevantly reduced by restricting the instantiation of `other_flows` only to the eight points that can actually be reached at that time. This restrinction proved to be efficient, reducing computation times up to 60%.

6.1 A variant: variable front height

Besides the above ASP implementation, we developed another ASP program, let's call it *Lava Motion Linear Front — LF*, that differs from the previous one only in the definition of the front height. While in fact the original version of Lava Motion considers the front height as a *constant*, in Lava Motion LF the front increases *linearly* with the length of the lava path. More precisely, given a maximum value the front height can take (be it *max_front*), at time T the height is $max\_front \times T / max\_path\_length$.

The new program has the same structure as the one above, except for the following parts:

```
% front height range. max_front is a constant defined in file 'mat_nxn'
front_r(0..max_front).

% The front height at time 0 is 0.
front_h(0, 0).

% at time T, the front height is = max_front * T / max_step.
front_h(N, T) :- time(T), T > 0,
                 V = max_front * T,
                 N = V / 1,
                 front_r(N).

% the new steepest_descent predicate differs from the original
% one only when obtaining the present front height
steepest_descent(From, To, T) :- adj(From, To),
                                 p(From, HF),
                                 p(To, HT),
                                 front_h(N, T),
                                 front_r(N),
                                 HT <= HF + N,
                                 time(T),
                                 not flooded(To, T),
                                 not no_steepest_descent(From, To, T).
```

7 Benchmarking

Table 1 shows execution times for solver *Smodels* when invoked on *Lava Motion LF*. CPU times are reported in seconds as the sum of *user* and *system* time using *time* UNIX command. The values recorded in the table include time required by the grounder, Lparse. The tests were run on a Pentium IV, 512 MB of RAM, with Red Hat Linux 9.0 operating system.

We considered four 4×4 matrices, four 8×8 matrices and four 16×16 matrices: in table 1 we record the average time values for each matrix type. We fixed the length of lava path to realistic values according to the dimension of the matrix.

Instance		*Solver*
Matrix	Path Length	*Smodels*
4×4	4	0.30
4×4	8	0.59
8×8	4	1.88
8×8	8	3.44
8×8	16	7.02
16×16	4	10.42
16×16	16	36.64

Table 1. *Lava Motion LF* Results

The complexity key parameters are the matrix dimension and the length of the lava path. In particular, performances dramatically get worse while passing from a small to a bigger matrix. For a 4×4 matrix, in fact, *Smodels* requires less

than one second to find a model, while on a 16×16 map the time spent by the solver is more than 30 times greater. The path length is relevant, too: doubling this parameter causes the increase in the required time by a factor of about 1.8 to 2.

Times recorded for the base version, Lava Motion, are pretty close to the ones collected in Table 1: passing from a constant front height to a linear growing one, even though this means adding new predicates for computing the height of the lava front to the program, does not seem to affect *Smodels* performances.

The comparison between times shown in Table 1 and execution times for the older version of Lava Motion points out that, even thought for small input matrices time saving is minimal, the optimization introduced in the final program causes, on bigger instances, a reduction of computation times of about 60%.

8 Conclusions

The ASP program presented above and the results shown in Table 1 are perhaps too preliminary to conclude that ASP can successfully address lava motion problems. However, we would like to stress that both the lava motion model and the geographical data are comparable with those used in the current GIS literature, so the present times are relevant. Moreover, we are only at the beginning of the process that will lead us to simplify and optimize our ASP program. If optimization leads us to execution in reasonable times even for large maps, we will develop the evacuation planner and use our program for verifying the computed plans, or even existing ones already available from monitoring and emergency organizations.

References

[Alferes & Pereira, 2002] J. J. Alferes and L. M. Pereira, 2002. *Logic Programming Updating - A Guided Approach,* Computational Logic: Logic Programming and Beyond, Essays in Honor of Robert A. Kowalski, Part II, LNAI 2408, Springer-Verlag, Berlin, pp. 382-412.

[Armienti et al., 1988] P. Armienti, G. Macedonio, and M.T. Pareschi, 1988. *A numerical model for simulation of Tephra transport and deposition: applications to May 18 Mount St. Helen Eruption.* J. Geophys. Res. Vol. 93, pp. 6463-6376.

[Baral, 2003] C. Baral, 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving.* Cambridge Press.

[Cattinelli, 2004] I. Cattinelli, 2004. *Nuovi benchmark per la valutazione di risolutori per l'Answer Set Programming.* Graduation Project in Informatics (in Italian), University of Milan. Tables and interpretation of the results are available from *http://mag.usr.dsi.unimi.it/*

[Costantini et al., 2003] S. Costantini, B. Intrigila and A. Provetti, 2003. *Coherence of Updates in Answer Set Programming.* In Brewka and Peppas (eds.) Proc. of NRAC03 Workshop.

[Felpeto et al., 2000] V. Araña, A. Felpeto, M. Astiz, A. García, R. Ortiz and R. Abella, 2000. *Zonation of the main volcanic hazards (lava flows and ash fall) in Tenerife, Canary Islands. A proposal for a surveillance network.* Journal of Volcanology and Geothermal Research, Vol. 103, pp. 377–391.

[Felpeto et al., 2001] A. Felpeto, V. Araña, R. Ortiz, M. Astiz and A. García, 2001. *Assessment and Modeling of Lava Flow Hazard on Lanzarote (Canary Islands).* Natural Hazards, 23, pp. 247–257.

[Gelfond & Lifschitz, 1988] M. Gelfond, and V. Lifschitz, 1988 *The stable model semantics for logic programming.* Proc. of 5th ILPS conference, pp. 1070–1080.

[Gelfond & Lifschitz, 1991] M. Gelfond, and V. Lifschitz. *Classical negation in logic programs and disjunctive databases.* New Generation Computing, pp. 365–387.

[Marek & Truszczyński, 1999] W. Marek, and M. Truszczyński. *Stable models and an alternative logic programming paradigm,* The Logic Programming Paradigm: a 25-Year Perspective, Springer-Verlag, pp. 375–398.

[Niemelä et al., 2000] I. Niemelä, P. Simons, and T. Syrjanen, 2000. *Smodels: a system for answer set programming.* Proc. of the 8th International Workshop on Non-Monotonic Reasoning.

[Osorio & Zepeda, 2004] M. Osorio and C. Zepeda, 2004. *Towards the use of Semantic Contents in ASP for planning and Diagnostic in GIS.* Proc. of MICAI04 Conference, Springer LNAI.

[ASP solvers] Web location of the most known ASP solvers:
CCALC: *http://www.cs.utexas.edu/users/mcain/cc*
Cmodels: *http://www.cs.utexas.edu/users/tag/cmodels.html*
DeReS: *http://www.cs.engr.uky.edu/~lpnmr/DeReS.html*
DLV: *http://www.dbai.tuwien.ac.at/proj/dlv/*
NoMoRe: *http://www.cs.uni-potsdam.de/~linke/nomore/*
SMODELS: *http://www.tcs.hut.fi/Software/smodels/*