Evacuation Planning using Answer Set Programming: An initial approach

Claudia Zepeda *and David Sol †

Abstract—**This paper describes a methodology based on Answer Set Programming (ASP) to work with incomplete geographic data. Source geographic data which describes a risk zone is translated to ASP description and it allows to solve query's which can not be solved by a normal GIS. An evacuation plan can change when new situations are presented, for instance traffic, a zone in extreme danger or other natural modification of the zone to be evacuated. Since 1994, Risk Management Office in Mexico has declared around 30 km from the Popocatépetl Volcano crater a danger zone. This office defined several roads to evacuate the people when a Volcano event can be presented. Our approach allows to simulate and to give support to generate new evacuation plans. The results developed by the ASP approach can be translated to a visual format and can be incorporated to a GIS to develop other kind of analysis by using the geographic data.**

Keywords: Evacuation planning, Answer Set Programming, Geographic Data, Risk Management, Query with incomplete data

1 Introduction

Currently, people involved in protection against disaster situations must make decisions about preparing and executing evacuation plans for potential causes of disaster. Hence, it would be desirable to develop an intelligent system capable of modeling and solve evacuation plan problems based on knowledge about a particular environment, geographic data, and its own capabilities. It would be desirable that such system could be capable of exchanging information and services with similar systems as well as with persons.

Popocatépetl Volcano grows its activity since 1994. It is located about 60 km from Mexico City and 40 km from Puebla City. About 200,000 people live around the Volcano distributed in 60 towns. Governments from Morelos Mexico and Puebla proposed an evacuation plan. Shelter rooms located in the principal towns around the Volcano were prepared. Our description will be focused in Puebla where the "Plan Operativo Popocatépetl" office is charged to propose, to coordinate and to organize an evacuation plan. Danger Zones are defined related with the Popocatépetl volcano activity and morphology. Pyroclastics, Ash and Earthquakes are the main dangers produced by the Volcano. Risk is defined when a town is located inside a danger zone. Evacuation order depends on the town location. Towns in risk are first evacuated [18].

Figure 1 shows the danger zones, the location of the towns and the roads to connect the risk towns with important cities in Puebla State. All the elements that compose the description of the area can be coded in geometric objects —points, polygons and line strings — to represent towns, danger areas and roads [18].

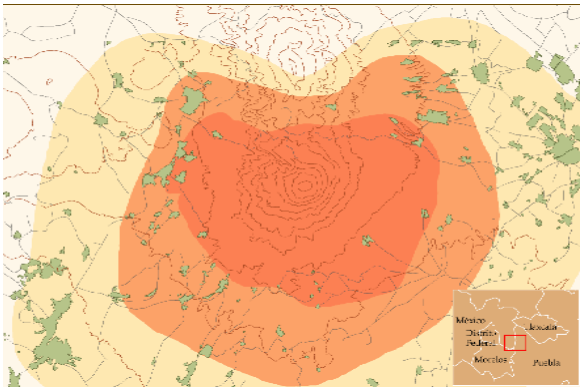

Figure 1: Digital Map, Danger Zones, Towns and evacuation roads.

In this work we present an initial analysis about the use of a formalism called Answer Set Programming (ASP) to model and solve evacuation plan problems. We consider that ASP could be used to develop the core of such intelligent system. ASP is a declarative knowledge representation and logic programming language [11]. The original definition of answer sets was given by Gelfond and Lifschitz in 1988. ASP is the realization of much

*Universidad Politécnica de Puebla, Tercer Carril del Ejido Serrano, San Mateo Cuanala, Juan C. Bonilla, Puebla, 72640 México, Email: czepedac@gmail.com

†Tecnológico de Monterrey, Vía Atlixcayotl 2301, Cholula, México, Email: dsol@itesm.mx

theoretical work on Non-monotonic Reasoning and AI applications. ASP represents a new paradigm for logic programming that allows, using the concept of negation as failure, to handle problems with default knowledge and produce non-monotonic reasoning. Two popular software implementations to compute answer sets are DLV[1] and SMODELS[2]. The efficiency of such programs allowed to increase the list of practical applications in the areas of planning, logical agents and artificial intelligence.

Specifically, the objective of our work is to investigate and evaluate the capabilities of ASP to model disaster situations in order to give support in defining evacuation plans.

The motivation of our work is based on the idea that ASP could inherit many of the desirable capabilities that the kind of intelligent systems described should have. It is possible to express restrictions [11]. ASP allows the concept of negation as failure to express exceptions and represent incomplete knowledge. It is possible translate geographic information into a format that ASP can handle [22]. There exists an approach called *Answer Set* (AS) Planning [12] that provides a natural and elegant way to model planning problems. Currently there are different ASP approaches to express preferences [7, 19, 4], updates [21, 1, 9], argumentation [8, 15], among others.

Our paper is structured as follows. In section 2 we propose a list of information which would be ideal to have in case of modeling evacuation plan problems, and we indicate how we can translate geographic information into a format that ASP can handle. In section 3 we present the background knowledge that we use to model of evacuation plan problems and its encoding in ASP. In section 3 we also presents how to use an action language to specify evacuation plan problems and its encoding in ASP. In section 3 we describe the alternative evacuation plan problem and we present an initial solutions of it based on ASP preference approaches. Finally, in section 5 we present some conclusions and future work.

2 ASP background knowledge

In [20] the UNESCO present a list of recommendations to develop volcanic emergency plans. Additionally, the state of art of models and algorithms for evacuation planning including the information needed to model evacuation plan problems is presented in [13]. So, we combine the list of information given in [20] and [13] to propose a list of information which would be ideal to have in case of modeling evacuation plan problems. The list of information is the following:

— *network of roads* where evacuees will travel,
— number of *locations in the hazard zone*,
— *source and location of hazard(s)*,
— *locations in risk* depending on the kind of hazard,
— *evacuation routes* from locations in the hazard zone to safe destinations,
— *means of transport* indicating their *capacity* and the need resources needed to travel (such as *fuel*),
— the *number of person* that must be evacuated by government,
— *availability of emergency services* such as facilities and personnel, i.e., personnel and equipment for search and rescue, hospitals and medical services,
— *alert procedures* and communication of public warning.

Most of the information listed above to model an evacuation plan problem corresponds to geographic information, sometimes abbreviated as GIS information. GIS information follows one of the standard formats: shapefile [10] or OpenGIS [6]. In general these formats of GIS information divide the data into two parts [14]. The first part contains spatial data, and the second part contains non spatial data or descriptive information. The spatial data manage the position of geometric objects within an area where the basic objects are: points, lines and polygons. The non spatial data describe what is at a point, along a line or in a polygon, and contains the socio-economic characteristics, such as demographic data, occupation data for a village, or traffic volume for roads. So, in order to model and solve evacuation plan problems using ASP, it is necessary consider GIS information to construct the hazard zone background knowledge.

The Popocatépetl Volcano area of danger includes three Mexican states: Mexico, Morelos and Puebla. Our data set will describe only the state of Puebla. Puebla is located to the east of the volcano. Our work will be focused on the data set describing this state.

An important number of towns are located inside this area and the risk is quantified by the number of people that live there. We divide the data set in two groups: the first includes the town description with the number of people that live there and information related with the resources needed to evacuate the people; the second describes the state of the evacuation roads with the number of lanes, the speed limit and the important cross roads. Figure 2 presents the general data model which includes both groups of the data set. The application section shows the dynamic data which is another important set of attributes that the application gets from the user.

The data set includes 26 towns in the State of Puebla and all the roads to connect with the main cities around the volcano. The "Plan Operativo Popocatépetl" office

[1] http://www.dbai.tuwien.ac.at/proj/dlv/
[2] http://www.tcs.hut.fi/Software/smodels/

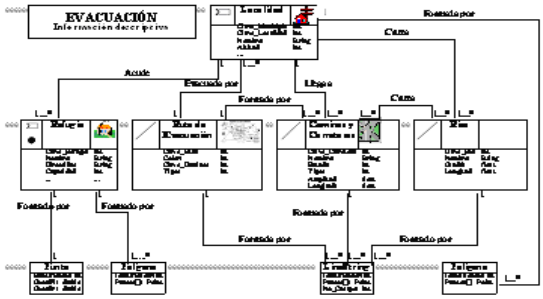

Figure 2: General Data Model.

Figure 3: Network of Towns and Shelters.

prepared a data set to describe the number of inhabitants who live there, how many of them can evacuate by themselves, the number of kilometers to be safe and the name and the geographic location of the shelter room [18].

The roads are described by line strings which are geometric objects and the network has been constructed to manage directions and connections from the towns to the shelter rooms and to all the important cities. The original network is shown in the Figure 3. Each line string describing a segment of the evacuation road has a set of attributes. The network is described by nodes and stretches. The nodes describe important points and the stretches allow the node connection [17].

As result of our direct work over the geographic information of volcano Popocatépetl risk zone we can indicate that to construct such background knowledge we have to:

1. extract descriptive information from geographic information,
2. repair inconsistencies of descriptive information,
3. translate the descriptive information into a format that ASP can handle, i.e., into a set of facts.

It is worth mentioning that in this work we tested our results using only part of the geographic information needed to model evacuation plan problems described in this section. Specifically, we only considered part of the geographic information about towns and roads.

3 Modeling evacuation plan problems using AS Planning

In an evacuation plan problem, we are interested in looking for a sequence of actions that leads from an initial state representing a risk scenery to a given goal state representing a scenery where people will be out of risk.

Currently, there exist different action languages used for talking about the effects of actions and to model plan problems [12]. Some of these action languages are $\mathcal{A}, \mathcal{B}$, and $\mathcal{C}$ [12]. Moreover, a plan problem specified in these kinds of languages has a natural and easy encoding in ASP [3].

This section is divided into two parts. The first part presents the ASP encoding of the background knowledge that we use in our work. The second part presents the use of action language $\mathcal{B}$ to specify evacuation plan problems and its encoding in ASP.

3.1 Modeling the background knowledge

In our model of evacuation plan problems, the background knowledge corresponds to a network of roads between towns in a hazard zone and shelters connected by roads. Of course the information to define the network of roads must correspond to the translation of geographic information as it was described in section 2. The network of roads is represented as a directed graph. In this directed graph some nodes represent towns and evacuation routes are paths in the graph. Each segment is represented by `road(P,Q,R)` where `P` and `Q` are nodes and `R` is the route number. Segments with route number different to zero belong to some evacuation route. Some nodes correspond to a shelter.

Example 1 *Figure 4 shows a directed graph that corresponds to two evacuation routes of the volcano Popocatépetl risk zone (see Figure 1). This directed graph is defined using a short part of the geographic information of volcano Popocatépetls risk zone. So we can define the*

Figure 4: Two evacuation routes: A short example.

ASP encoding of the background knowledge corresponding to the directed graph of Figure 4 as follows:

```
% Number of routes:
route(2). route(1). route(0).

% node(point,route)
% Nodes of Rute 1:
node(1,1).   node(2,1).  node(8,1).  node(9,1).
% Nodes of Route 2:
node(13,2).  node(12,2).  node(17,2). node(19,2).
% Nodes without route:
node(2,0).  node(4,0).  node(5,0).  node(11,0). node(12,0).
node(15,0).  node(16,0). node(16,0). node(13,0).

% segment (ini,fin,route)
% Segments of Rute 1:
segment(1,2,1).     segment(2,8,1).      segment(8,9,1).
% Segments of Route 2:
segment(12,17,2).   segment(13,17,2).  segment(17,19,2).
% Segments without route:
segment(2,11,0).    segment(2,4,0).      segment(4,5,0).
segment(4,9,0).     segment(12,15,0).  segment(15,16,0).
segment(16,19,0).   segment(13,15,0).



%Towns located at:                      Towns in risk:
% townAt(town, node)
townAt(p1,1).                           townInRisk(p1).
townAt(p2,12).                          townInRisk(p2).
townAt(p3,13).                          townInRisk(p3).

% Buses and initial position of buses
% bus(busNumber)                    busIniAt(bus,point).
  bus(b1).                              busIniAt(b1,p1).
  bus(b2).                              busIniAt(b2,p2).
  bus(b3).                              busIniAt(b3,p3).

%location of shelters
  shelther(9).            shelter(19).
```

3.2 Using Action language $\mathcal{B}$ to model evacuation plan problems and its ASP encoding

In order to model evacuation plan problems we consider three fluents: *position(B,Q,R)*, *blocked(P,Q,R)*, and *end(N)*. The fluent *position(B,Q,R)* indicates that bus B is at position Q of evacuation route R. The fluent *blocked(P,Q,R)* indicates that the segment of road from P to Q of route R is blocked. The fluent *end(B)* indicates that bus B is in a position that corresponds to a shelter.

Additionally, we consider the action *travel(B,P,Q,R)* to allows bus B to travel from P to Q of route R.

Using action language $\mathcal{B}$, fluents and actions can be represented as: $position(B, Q, R)$, $blocked(P, Q, R)$, $end(B)$, and $travel(B, P, Q, R)$.

It is important to mention that all action has executablity conditions and effects. The executability conditions of action *travel(B,P,Q,R)* are the following:

1. There is an unblocked segment of road from P to Q such that this segment belong to an evacuation route, i.e., there exist a $segment(P, Q, R)$ with R different to zero.
2. Bus B is at position P, i.e., the fluent $position(B, Q, R)$ holds.
3. Nodes P and Q belong to the same evacuation route R, i.e., there exist $node(P, R)$ and $node(Q, R)$ with R different to zero.

The effects of the execution of action *travel(B,P,Q,R)* are the following:

1. Position of bus B is not P, i.e., the fluent $\sim position(B, P, R)$ holds.
2. position of bus B is Q, i.e., the fluent $position(B, Q, R)$ holds, and
3. position of bus B is the position of a shelter, i.e., if fluent $position(B, N, R)$ holds and the position of a shelter is N then, fluent $end(B)$ holds too.

The representation of executablity conditions and effects of the action *travel* using action language $\mathcal{B}$ is:

$travel(B, P, Q, R)$ **causes** $position(B, Q, R)$ **if**
$position(B, P, R), \sim blocked(P, Q, R), R <> 0.$
$travel(B, P, Q, R)$ **causes** $\sim position(B, P, R).$
$position(B, N, R), refuge(N)$ **causes** $end(B).$

Additionally we must to indicate the initial and final states of the evacuation planning problem that we are modeling. Both states are defined as a set of fluents. For instance, if we consider the directed graph of Figure 4, these states can be represented using action language $\mathcal{B}$ as follows:

initially $position(b1, 1, 1)$ **initially** $position(b2, 12, 2)$
initially $position(b3, 13, 2)$
finally $end(b1)$ **finally** $end(b2)$
finally $end(b3)$

Finally we most translate the plan problem modeled in language $\mathcal{B}$ to an ASP encoding. In order to show how to make this translation we shall describe an example with a possible ASP encoding of the plan problem modeled in language $\mathcal{B}$ previously. In general the ASP encoding of an action problem has two parts [3]: a domain dependent part and a domain independent part. The first part defines initial state, goal state, actions and effects of actions depending on the problem domain. The second part is needed to indicate the change of the current state (modification of the truth values of some fluents) due to the execution of some action when some preconditions hold in the current state. This ASP encoding follows the SMODELS syntax.

Example 2 *This example shows a possible ASP encoding of the plan problem modeled in language $\mathcal{B}$ in this section. So we consider the ASP encoding of the background knowledge corresponding to the directed graph of Figure 4 and described in Example 1.*

```
%%%%%%%%%%% DOMAIN DEPENDENT PART %%%%%%%%%%%

% Fluents and Actions
 fluent(position(B,Q,R)).
 fluent(blocked (P,Q,R)).
 fluent(end(B)):- bus(B).

 action(travel(B,P,Q,R)) :- bus(B), segment(P,Q,R).

% Initial  and final states
 initially(position(b1, 1, 1)).
 initially(position(b2, 12, 2)).
 initially(position(b3, 13, 2)).

 finally(end(b1)).  finally(end(b2)). finally(end(b3)).

% Effects of actions
% caused(F, A) means action A causes fluent F.
 caused( position(B,Q,R) , travel(B,P,Q,R) ).
 caused( neg(position(B,P,R)) , travel(B,P,Q,R) ).
 caused( end(B) , position(B,P,R) ) :- shelter(P).

% Executability Conditions
% noaction_if(A, neg(F)) means if fluent F
% does not hold then action A is not executed.
 noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% noaction_if(A, F) means if fluent F holds then,
% action A is not executed.
 noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).

%%%%%%%%%%% DOMAIN INDEPENDENT PART %%%%%%%%%%
% if there is not evidence of action A is not
% executable at time T then A is executable at T
 executable(A,T) :- action (A), time (T),
    not not_executable(A,T).

% if fluent F holds at time T then
% action A is not executable
 not_executable(A,T) :- action(A), time(T),
                  noaction_if(A,F), h(F,T).

% ------- concurrency --------
% It is possible action A at time T whenever
% A is executable and there is not evidence of
% the goal holds.
 possible(A,T) :- action(A), time(T),
                executable(A,T), not goal(T).

% if action A is possible at time T and there is not
% evidence of A does not occur at T then,
% A occurs at time T.
 occurs(A,T) :- action(A),time(T),
      possible(A,T), not not_occurs(A,T).

% if an action AA occurs at time T and
% AA is different of action A then,
% A does not occur at the same time T.
 not_occurs(A,T) :-
    action(A),action(AA),time(T),
    occurs(AA,T), neq(A,AA).

% all initial fluents hold at Time 1
 h(F,1) :- literal(F),initially(F).

% if there is not evidence of fluent F holds
% at time 1 then, fluent -F holds at time 1.
h(neg(F), 1) :- fluent(F), not h(F,1).

% if action A is executed and occurs at time T,
% and A causes fluent F, then F holds at time T.
 h(F,T+1) :- literal(F),
         time(T), action(A), T< n,
         executable(A,T), occurs(A,T), caused(F,A).

% ------- inertia -------
% if fluent F holds at time T and there is not
% evidence of -F holds at time T+1 then,
% F holds at the next time T+1.
  h(F, T+1) :- literal(F), literal(G),
               contrary(F,G), time(T), T < n,
               h(F,T), not h(G, T+1).

% miscellaneous
  contrary(F, neg(F)) :- fluent(F).
  contrary(neg(F), F) :- fluent(F).

  literal(G) :- fluent(G).
  literal(neg(G)) :- fluent(G).
```

Finally, we can get the plans of this plan problem that correspond to the answer sets of its ASP encoding. In these plans we assumed that each action take one unit of

time and we consider the background knowledge of example 1. In particular, this example has only one plan (see the directed graph of Figure 4):

time 0	*time 1*	*time 2*
travel(b1,1,2,1)	*travel(b1,2,8,1)*	*travel(b1,8,9,1)*
travel(b2,12,17,2)	*travel(b2,17,19,2)*	————
travel(b3,13,17,2)	*travel(b3,17,19,2)*	————

This plan shows that all buses should follow their predefined evacuation route from their initial positions to their final positions, exactly as it is expected when there are predefined evacuation routes and these predefined routes are not blocked.

4 Alternative evacuation plans

Normally, in a risk zone there is a number of predefined evacuation routes. Each evacuation route starts in a set of places in risk, traverses other places in risk and arrives to a place out of risk. Some times the place out of risk corresponds to a shelter. However, some hazards that can accompany a disaster can result on the blocking of some predefined evacuation routes. Therefore it is necessary to define alternative evacuation plans. We can state the *alternative evacuation plan* (AEP) problem as follows: *There is a set of predefined evacuation routes for people living in a hazard zone. Each predefined evacuation route may have several initial points, but one single final point. Sometimes the predefined evacuation routes can become blocked. In case part of a predefined evacuation route is blocked, then evacuees should travel by an alternative path to arrive to a final point. We can define a decreasing order of preferences to choice the alternative path. If an alternative path follows part of a predefined evacuation route and it arrives to a shelter then this path is the most preferred. If an alternative path does not follow a predefined evacuation route but it arrives to a shelter then this path is the second most preferred alternative path. Finally, if an alternative path does not follow a predefined evacuation route and it arrives to a place out of risk where a shelter is not located then this path is the last preferred alternative path.*

In this section we present an initial solutions to AEP problem. This solution uses ASP approaches to represent and solve preference problems: *consistency restoring rules* [2] and *$\mathcal{PP}$ language* [19].

4.1 Using Consistency Restoring rules

Let us consider the AEP problem and let us suppose that part of a predefined evacuation route is blocked. If action `travel` allows buses to travel only by a predefined evacuation route then it is not possible to define an evacuation plan. For instance, if we consider the example 2 and we suppose that segment $road(12, 17, 2)$ is blocked then, action `travel` cannot occur using this blocked segment. So, under this conditions it is not possible to obtain the evacuation plans, i.e., the answer sets.

In order to solve the AEP problem we propose to add Consistency Restoring (CR) rules [2]. CR rules are rules that are added to standard disjunctive logic programs, but they are only applied when the standard rules in the program lead to inconsistency. A CR rule is written $\alpha \stackrel{+}{\leftarrow} \beta$, which is intuitively read as: *If the program is inconsistent, then assume $\alpha \leftarrow \beta$. Otherwise, ignore the CR rule.* Programs with CR rules are called CR programs. We recall the following example presented in [2], since it illustrates clearly the use of CR rules. In this example r_1 denotes the name of the CR rule. Let P the following CR program:

$$a \leftarrow \neg b.$$
$$\neg\, a.$$
$$r_1 : b \stackrel{+}{\leftarrow} .$$

The first two rules are *regular rules* and the third rule is a CR rule. We can see that P without the rule r_1 is inconsistent. However, if the CR rule is used then consistency is restored, thus the answer set of P is $\{\neg a, b\}$. The semantics of CR programs are defined in terms of *minimal generalized answer sets* [2]. Due to space reasons, we do not present the translation. The intuitive understanding of how CR rules can be applied is sufficient for the purposes of this paper. For further reading refer to [2].

The idea about using the CR rule concept to solve the AEP problem is to use it when there is no way to obtain an evacuation plan following a predefined evacuation route since part of it is blocked and action `travel` cannot be executed. Hence, the following CR rule is defined:

$$r_2 : action(travel(B, P, Q, R)) \stackrel{+}{\leftarrow} bus(B), road(P, Q, R).$$

The intuition of r_2 indicates that it is possible to travel from P to Q if there is a segment of road from P to Q. Moreover r_2 does not check if the road from P to Q is part of a predefined evacuation route, i.e, it is not important whether R is equal to zero or not. In contrast to the regular action `travel` defined in section 3 where R must be different to zero, i.e., the segment of road from P to Q must be part of an evacuation route.

The following example shows how we can use CR-rules to obtain the alternative evacuation plans.

Example 3 *Let us consider the ASP encoding of example 2 and let us suppose that part of the evacuation route 2 is blocked because* $road(12, 17, 2)$ *is blocked, i.e, we add* **initially** $blocked(12, 17)$ *to that ASP encoding. Since, it is not possible that bus* $b2$ *follows its predefined evacuation route assigned, then it is not possible to define an evacuation route. So, we propose to add the following CR-rule:*

$$r_2 : action(travel(B, P, Q, R)) \stackrel{+}{\leftarrow} bus(B), road(P, Q, R).$$

Thus, we can rewrite the domain independent part of the ASP encoding in example 2 as follows:

```
% Fluents
 fluent(position(B,Q,R)).
 fluent(blocked (P,Q,R)).
 fluent(end(B)):- bus(B).

% Initial state
 initially(position(b1, 9, 1)).
 initially(position(b2, 12, 2)).
 initially(position(b3, 13, 2)).
 initially(blocked(12,17)).    % road(12,17,2) is blocked.

% Goal
 finally(end(b1)).
 finally(end(b2)).
 finally(end(b3)).

% REGULAR action travel:
% buses only travel by a predefined evacuation route,
% i.e., they travel by segments of road where R<>0.
 action(travel(B,P,Q,R)) :-segment(P,Q,R),bus(B),R<>0.

% CR-rule where R can be zero,
 action(travel(B,P,Q,R)) :- bus(B), road(P,Q,R).

% Effects of actions
% caused(F,A) means action A causes fluent F.
 caused( position(B,Q,R) , travel(B,P,Q,R) ).
 caused( neg(position(B,P,R)) , travel(B,P,Q,R) ).
 caused( end(B) , position(B,P,R) ) :- shelter(P).

% Executability Conditions
% noaction_if(A, neg(F)) means if fluent F
% does not hold then, action A is not executed.
 noaction_if(travel(B,P,Q,R),neg(position(B,P,R))).

% noaction_if(A,F) means if fluent F holds then,
% action A is not executed.
 noaction_if(travel(B,P,Q,R),blocked(P,Q,R)).
```

We can verify that we get four alternative evacuation plans, i.e., four answer sets. This evacuation plans are the following:

Plan1:

time 0	*time 1*	*time 2*
travel(b1,1,2,1)	*travel(b1,2,4,0)*	*travel(b1,4,9,0)*
travel(b2,12,15,2)	*travel(b2,15,16,0)*	*travel(b2,16,19,0)*
travel(b3,13,15,2)	*travel(b3,15,16,0)*	*travel(b3,16,19,0)*

Plan2:

time 0	*time 1*	*time 2*
travel(b1,1,2,1)	*travel(b1,2,8,1)*	*travel(b1,8,9,1)*
travel(b2,12,15,0)	*travel(b2,15,16,0)*	*travel(b2,16,19,0)*
travel(b3,13,15,0)	*travel(b3,15,16,0)*	*travel(b3,16,19,0)*

Plan3:

time 0	*time 1*	*time 2*
travel(b1,1,2,1)	*travel(b1,2,8,1)*	*travel(b1,8,9,1)*
travel(b2,12,15,0)	*travel(b2,15,16,0)*	*travel(b2,16,19,0)*
travel(b3,13,17,2)	*travel(b3,17,19,2)*	—

Plan4:

time 0	*time 1*	*time 2*
travel(b1,1,2,1)	*travel(b1,2,4,0)*	*travel(b1,4,9,0)*
travel(b2,12,15,0)	*travel(b2,15,16,0)*	*travel(b2,16,19,0)*
travel(b3,13,17,2)	*travel(b3,17,19,2)*	—

Plan 3 indicates that bus $b1$ *and bus* $b3$ *should follow their predefined evacuation routes while bus* $b2$ *should travel by nodes out of a predefined evacuation route. The other three plans are explained similarly.*

We point out that in [16] we present how CR programs can be properly represented using Logic Programs with Ordered Disjunction (LPOD) [4]. So, we can use PSMODELS[3] to compute the preferred answer sets under the ordered disjunction semantics and to obtain the alternative evacuation plans.

4.2 Using $\mathcal{PP}$ language

We have seen that using the CR program concept makes it possible to solve part of the AEP problem since it allows obtain alternative evacuation plans. However in the case that we get more than one alternative plan it is necessary to prefer one of them according to different criteria. So, in this section, we propose to consider language $\mathcal{PP}$ [19] to express preferences at different levels over the alternative plans.

Language $\mathcal{PP}$ [19] is a language useful to specify user preferences with a logic programming implementation of it based on ASP. Language $\mathcal{PP}$ allows someone to specify preferences among feasible plans and temporal preferences over plans too. The preferences representing time are expressed using the temporal connectives *next, always, until* and *eventually.* There are three different classes of preferences in $\mathcal{PP}$ *basic desires*, *atomic preferences* and *general preferences.* In our work we studied the two first classes of preferences because they are useful to express preferences in evacuation plans.

A *basic desire*, denoted as φ, is a $\mathcal{PP}$ formula expressing a preference about a plan with respect to the execution

[3] http://www.tcs.hut.fi/Software/smodels/priority/

of some specific action or with respect to the states that the plan gets when an action is executed. In particular, $\mathcal{PP}$ defines *goal preferences* to define preferences in the final state or goal. The set of basic desires of language $\mathcal{PP}$ can be defined inductively by the following context-free grammar $G_{PP} := (N, \Sigma, P, S)$, such that $N := \{S\}$ is the finite set of non terminals; $\Sigma := \mathbf{A} \cup \mathcal{F}_F$ is the finite set of terminals ($N \cap \Sigma = \emptyset$) where $\mathbf{A}$ and $\mathcal{F}_F$ represent the set of actions of the problem and the set of all fluent formulas (propositional formulas based on fluent literals) respectively; $S \in N$ is the initial symbol of the grammar; and $P := \{S \rightarrow p|\mathbf{goal}(p)|occ(a)|S \wedge S|S \vee S|\neg S|\mathbf{next}(S)|\mathbf{until}(S,S)|\mathbf{always}(S)|\mathbf{eventually}(S)\}$ is the finite set of productions or rules where $p \in \mathcal{F}_F$ and $a \in \mathbf{A}$.

An *atomic preference* is defined as a formula of the type $\varphi_1 \lhd \varphi_2 \lhd \ldots \lhd \varphi_n$ $(n \geq 1)$ where $\varphi_1, \varphi_2, \ldots, \varphi_n$ are basic desire formulas. An atomic preference represents an ordering between basic desire formulas. It indicates that plans that satisfy φ_1 are preferable to those that satisfy φ_2, etc. Clearly, basic desire formulas are special cases of atomic preferences.

Due to space reasons, we do not present the formal definition of language $\mathcal{PP}$, and its semantics. The intuitive understanding of how basic desires and atomic preferences can be used to prefer a plan is sufficient for the purposes of this paper. For further reading refer to [19].

In this section we give an idea about how to use language $\mathcal{PP}$ to solve the AEP problem. We considered to use $\mathcal{PP}$ because it allows us to express preferences over plans where the satisfaction of these preferences depends on time and on their temporal relationships. We think that in particular in evacuation planning it is very useful to express preferences in terms of time. For instance, it is *always* preferred to evacuate people form a place in risk following the defined evacuation routes. However, if *eventually* part of the evacuation route is blocked then evacuees will travel out of some evacuation route *until* they arrive to some shelter. The following example shows the use of language $\mathcal{PP}$ to express preferences among the alternative evacuation plans.

Example 4 *Let us consider the ASP encoding of example 3. We could define the following basic desires over the plans:*

—**travelERass** *to indicate that it is preferred that buses travel by the predefined evacuation route assigned until they arrive to a shelter. So, it is preferred that bus*1 *travels by segments of route 1 until it arrives to node* 9 *where its assigned shelter is located, and bus*2 *and bus*3 *travel by segments of route 2 until they arrive to node* 19 *where their assigned shelter is located.*

$$
\begin{aligned}
&travelERass := \\
&\mathbf{until}(\mathbf{occ}(travel(bus1, 11, 2, 1)) \vee \mathbf{occ}(travel(bus1, 2, 8, 1)) \vee \\
&\quad \mathbf{occ}(travel(bus1, 8, 9, 1)) \ , \ position(bus1, 9, 1)) \ \wedge \\
&\mathbf{until}(\mathbf{occ}(travel(bus2, 12, 17, 2)) \vee \mathbf{occ}(travel(bus2, 17, 19, 2)) \ , \\
&\quad position(bus2, 19, 2)) \ \wedge \\
&\mathbf{until}(\mathbf{occ}(travel(bus3, 12, 17, 2)) \vee \mathbf{occ}(travel(bus3, 17, 19, 2)) \ , \\
&\quad position(bus3, 19, 2))
\end{aligned}
$$

—*travelBlSh to indicate that it is preferred that buses travel by its assigned evacuation route until eventually part of its evacuation route is blocked and then they travel out of some predefined evacuation route until they arrive at its assigned shelter. For instance, it is preferred that bus*1 *travels by segments in route 1 until eventually some of these segments are blocked, and then bus*1 *travels by segments out of some predefined evacuation route until it arrives to position 9 where it is located its assigned shelter.*

$$
\begin{aligned}
&travelBlSh := \\
&\mathbf{until}(\mathbf{occ}(travel(bus1, 11, 2, 1)) \vee \mathbf{occ}(travel(bus1, 2, 8, 1)) \vee \\
&\quad \mathbf{occ}(travel(bus1, 8, 9, 1)) \ , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(11, 2, 1) \vee blocked(2, 8, 1)) \vee \\
&\qquad blocked(8, 9, 1)), \\
&\qquad travel(bus1, 2, 4, 0)) \vee travel(bus1, 4, 9, 0)) \vee \\
&\qquad \vee position(bus1, 9, 1)\ )\ ) \\
&\ \wedge \\
&\mathbf{until}(\mathbf{occ}(travel(bus2, 12, 17, 2)) \vee \mathbf{occ}(travel(bus2, 17, 19, 2)) \ , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(12, 17, 2) \vee blocked(17, 19, 2))), \\
&\qquad travel(bus2, 12, 15, 0)) \vee travel(bus2, 15, 16, 0)) \vee \\
&\qquad travel(bus2, 16, 19, 0)) \vee position(bus2, 19, 2)\ )\ ) \\
&\ \wedge \\
&\mathbf{until}(\mathbf{occ}(travel(bus3, 13, 17, 2)) \vee \mathbf{occ}(travel(bus3, 17, 19, 2)) \ , \\
&\quad \mathbf{until}(\ \mathbf{eventually}(blocked(13, 17, 2) \vee blocked(17, 19, 2))), \\
&\qquad travel(bus3, 13, 15, 0)) \vee travel(bus3, 15, 16, 0)) \vee \\
&\qquad travel(bus3, 16, 19, 0)) \vee position(bus3, 19, 2)\ )\ )
\end{aligned}
$$

In a similar way we could express any other basic desire.

A possible atomic preference ψ indicating the order in which the set of basic desires formulas should be satisfied is the following: $\psi = travelERass \lhd travelBlSh$

The atomic preference ψ says that plans satisfying $travelERass$ are preferred, but otherwise plans satisfying $travelBlSh$ are preferred.

In [19] is described how to obtain the most preferred plan with respect to a basic desire or an atomic preference. The idea is to assign a weight to each plan, so the plan with the maximal weight is the most preferred plan. The weight associated to each plan results of adding the weight of each basic desire in the atomic preference. The weight of each basic desire in the atomic preference is

defined according to two things: the basic desire is satisfied by the plan and order of basic desires in the atomic preference. Additionally in [19] is shown how an atomic preference of $\mathcal{PP}$ can be mapped to a collection of standard ordered rules as defined by Brewka [4] in order to obtain the most preferred trajectory.

We noticed that the use of the mapping or weights proposed in [19] result in a complicated ASP encoding. Thus in order to allow a simpler and easier encoding, in [24] we proposed to use a particular kind of extended ordered rules [16] to compute the preferred plans of a planning problem with respect to an atomic preference. Moreover, it is worth mentioning that we can easily translate the particular kind of extended ordered program programs used to a standard ordered program and then use PSMODELS to obtain the preferred plan, i.e., the preferred answer sets.

In particular, if we consider the example 4 the most preferred plan with respect to the atomic preference ψ is the Plan3:

time 0	time 1	time 2
travel(b1,1,2,1)	travel(b1,2,8,1)	travel(b1,8,9,1)
travel(b2,12,15,0)	travel(b2,15,16,0)	travel(b2,16,19,0)
travel(b3,13,17,2)	travel(b3,17,19,2)	—

We can see that this most preferred plan satisfies the **travelBlSh** basic desire of the atomic preference ψ since $b2$ travels by a road out of the predefined evacuation route until it arrives to node 19 of evacuation route 2.

While we used $\mathcal{PP}$ to express preferences we realized that there are some preferences that cannot be expressed in a simple and natural way since they result very large. Then, in order to have a natural representation of these kind of preferences in [23] we defined $\mathcal{PP}^{par}$ language. $\mathcal{PP}^{par}$ is an extension of $\mathcal{PP}$ language where propositional connectives and temporal connectives allow us to represent compactly preferences having a particular property.

For instance, a natural and compact representation of preference *travelERass* of example 4 using a *parametric or* would be:

$$\textbf{until}(\bigvee\{occ(travel(B,I,F,R)) : bus(B), road(I,F,R), neq(R,0)\},$$
$$\bigwedge\{position(B,Fi,R) : bus(B), shelter(Fi), route(R), neq(R,0)\})$$

5 Conclusions and future work

In this work we have proposed to investigate and evaluate the capabilities of ASP to represent disaster situations in order to give support in defining evacuation plans. In spite of our work is an initial analysis, we came to the conclusion that using ASP is possible to model disaster situations and overall to take advantage of its capabilities and its different approaches developed to obtain the alternative evacuation plans. Additionally, we presented some lacks of these ASP approaches that we addressed in this work.

We also presented the information which would be ideal to have in order to model an evacuation plan problem and in particular a volcano evacuation plan problem. Since most of the information needed to model an evacuation plan problem corresponds to geographic information, we analyzed how geographic information about the disaster zone can be translated into a format that ASP is capable to understand. In particular, we explained how to construct the hazard zone background knowledge from geographic information.

The scenario that we considered in this work models a hazard zone where towns, roads, towns in risk, shelters, and predefined evacuation routes are defined. Our scenario also considers that in a real case it is possible that part of the predefined evacuation routes are blocked, and then generation of alternative evacuation plans is necessary. We realized that using only AS Planning to specify these kind of planning problems does not result in a natural way. So, we studied and applied different ASP approaches that were useful to obtain the evacuation plans and alternative evacuation plans, such as, CR rules, Logic Programs with Ordered Disjunction, and language $\mathcal{PP}$.

As future work, we plan to consider scenarios more real than those considered in this initial work to generate evacuation plans. These scenarios should consider the characteristics of relevant exogenous events (for instance an explosion or lava flows) occurring in a hazard zone and the complete information about a hazard zone can be considered. The Answer Set approaches used in [2] for model dynamic systems and in [5] to model lava flows could help us in this research direction.

References

[1] J. Alferes and L. Pereira. Logic programming updating: a guided approach, 2002.

[2] Marcello Balduccini and Michael Gelfond. Logic Programs with Consistency-Restoring Rules. In Patrick Doherty, John McCarthy, and Mary-Anne Williams, editors, *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, Mar 2003.

[3] Chitta Baral. *Knowledge Representation, reasoning and declarative problem solving with Answer Sets.* Cambridge University Press, Cambridge, 2003.

[4] Gerhard Brewka. Logic Programming with Ordered Disjunction. In *Proceedings of the 18th National Conference on Artificial Intelligence, AAAI-2002.* Morgan Kaufmann, 2002.

[5] I. Cattinelli, M. L. Damiani, and A. Nucita. Reasoning about Lava effusion: from Geographical Information Systems to Answer Set Programming. In *To appear LA-NMR 2004 CEUR Workshop proceedings*, volume 92, 2004.

[6] Inc. OpenGIS Consortium. Opengis reference model, version: 0.1.2, editor : Kart buehler. http://www.opengis.org, 2003.

[7] J. Delgrande, T. Schaub, H. Tompits, and K. Wang. A classification and survey of preference handling approaches in nonmonotonic reasoning. *Computational Intelligence*, 2004.

[8] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–358, 1995.

[9] Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, 2(6):711–767, 2002.

[10] Inc. Environmental Systems Research Institute. ESRI Shapefile Technical Description. An ESRI White Paper, New York, July 1998.

[11] Michael Gelfond and Vladimir Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *5th Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[12] Michael Gelfond and Vladimir Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.

[13] S. A. Tjandra H. W. Hamacher. Mathematical Modelling of Evacuation Problems: A State of Art. Technical Report 24, Institut Techno- und Wirtschaftsmathematik, 2001.

[14] P. A. Longley, M. F. Goodchild, D. J. Maguire, and D. W. Rhind. *Geographic Information Systems and Science.* Wiley, May 2001.

[15] Juan Carlos Nieves, Mauricio Osorio, Claudia Zepeda, and Ulises Cortés. Argumentation for decision making in CARREL using Answer Set Programming., Submmited to 28th German Conference on Artificial Intelligence, 2005. http://correo.udlap.mx/∼sc098382/dungsPaper/.

[16] Mauricio Osorio, Magdalena Ortiz, and Claudia Zepeda. Using CR-rules for evacuation planning. In Guillermo De Ita Luna, Olac Fuentes Chaves, and Mauricio Osorio Galindo, editors, *IX Ibero-american Workshops on Artificial Inteligence*, pages 56–63, 1994.

[17] D. Sol and M. Schmidt. Evacuating the inhabitants by using cars in the popocatépetl volcano area. In *Internal Report Universidad de las Américas-Puebla*, 2004.

[18] David Sol. Data models and applications to support risk management in the popocatépetl volcano zone. In *ESRI Press Conference*, 2001.

[19] Tran Cao Son and Enrico Pontelli. Planning with preferences using logic programming. In *LPNMR*, pages 247–260, 2004.

[20] UNDRO and UNESCO. Volcanic Emergency Management, United Nations, New York, 1985.

[21] Fernando Zacarias, Mauricio Osorio Galindo, J. C. Acosta Guadarrama, and Jürgen Dix. Updates in Answer Set Programming based on structural properties. In *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning. Dresden University Technical Report (ISSN 1430-211X)*, 2005.

[22] C. Zepeda. *Evacuation Planning using Answer Sets.* PhD thesis, Universidad de las Americas, Puebla and Institut National des Sciences Appliquées de Lyon, 2005.

[23] C. Zepeda, M. Osorio, D. Sol, and C. Solnon. Extending pp language: An answer set planning problem language. In *Proceedings of the Avances en la ciencia de la computacion Workshop, 6th Mexican International Conference on Computer Science (ENC 2005)*, pages 57–62, Puebla, Mexico, 2005.

[24] Claudia Zepeda, Mauricio Osorio, Juan Carlos Nieves, Christine Solnon, and David Sol. Applications of preferences using answer set programming. In *Submmited to Answer Set Programming: Advances in Theory and Implementation (ASP 2005).*