University of Calabria

Department of Mathematics and Computer Science

Bachelor's Degree Course in Computer Science

Thesis

ASP-Based System For Humanitarian Assistance

Supervisors:

Prof. Weronika Teresa Adrian

Prof. Carmine Dodaro

Candidate:

Stefan Antonov Yoshovski

Matriculation: 210438

Academic Year 2021/2022

Contents

Chapter 1

Introduction

1.1 Background and motivations

In an emergency situation caused by natural disasters or man-made disasters, the life and health of a large group of people are threatened. Hence the need to provide humanitarian assistance to help people in need. At the heart of a humanitarian aid system are volunteers who collaborate together, offering their time to provide lifesaving assistance and support in a timely manner.

One of the biggest challenges during a humanitarian crisis is the coordination of available human resources in such a way as to minimize the organizational imbalance. More specifically, it is important to distribute volunteers in designated places according to their availability and their abilities. This leads to the need to define a set of processes and strategic objectives for the allocation of human resources and can be classified as a workforce management problem [20].

The constantly evolving technologies in the field of artificial intelligence and knowledge representation and reasoning make it possible to find solutions to high-risk decision-making problems. One of these technologies is Answer Set Programming [16] oriented towards finding complex problems based on disjunction and the use of stable models.

The aim of this thesis is the creation of an ASP-based system for the coordination of available human resources involved in humanitarian assistance.

The thesis is structured as follows: in Chapter 2 are described the technologies and tools used for the realization of the project; In Chapter 3 the influx of refugees in Poland due to the war in Ukraine is analysed as a case study, problem requirements and goals to reach are reported; in Chapter 4 the model of the problem is further analysed, ASP-encoding is reported with the methodologies used to achieve the solution of the problem; in Chapter 5 is reported scalability analysis on simulated data; in the Conclusions, the results obtained are discussed and possible future articulations are presented.

1.2 Activities carried out and results obtained

As a case study, the influx of refugees in Poland due to the war in Ukraine is analysed in Chapter 3. In particular, the processes of receiving refugees in Poland and the role of volunteers in providing humanitarian assistance are analysed [4]. Furthermore, the mandatory requirements to be respected and the strategies to achieve the objectives are defined. The abstract model of the ASP program is based on the aforementioned preliminary analysis and the resulting program is divided into sub-programs dependent on each other following the Guess and Check methodology. In particular, the program is modelled in such a way that it can maximize the available human resources during a humanitarian crisis for a defined period of time, taking into account different factors related to geographical position, time availability and importance of a role carried out by a volunteer. Finally, a scalability analysis has been conducted on clustered data.

Chapter 2

Technologies and tools used for the development of the project

Answer Set Programming and the DLV system are on the basis of the implemented program and are further described in the following sections.

2.1 Answer Set Programming

Answer Set Programming (ASP) is a form of declarative logic programming oriented at finding solutions to complex decision problems [16]. The fields in which it finds greater application are those related to Artificial Intelligence and Knowledge Representation and Reasoning [15].

ASP is centred on the logical description of the characteristics that must be found in the set of solutions to a specific problem. The formalism used for representing real-world problems simplifies the interpretation of the problem itself and the manipulation of the data that compose it.

One of the biggest advantages of ASP is the ability to handle data incompleteness by formulating logical rules based on the deductive reasoning process. It is possible to add new information and knowledge at any time without the need to make any changes to the rest of the program [21].

The syntax and semantics used for encoding the ASP program reported in Chapter 4 are based on the ASP-Core-2 Input Language Format [11].

2.2 DLV

DLV is a disjunctive datalog system which supports expressive language constructs like aggregates, strong and weak constraints, functions, lists, and sets [10] which are part of the ASP-Core-2 standard language.

The system combines the answer set programming grounder and datalog reasoner I-DLV [12] and the ASP solver WASP [9]. DLV system includes a wide variety of built-in predicates for the manipulation of lists, integers and strings. Additional functionalities can be provided via external python scripts.

Chapter 3

Case Study: Influx of refugees from Ukraine into Poland

The Russian invasion of Ukraine, which began on 24 February 2022, has forced millions of people to flee their homeland and head towards neighbouring countries. In just 2 months, almost 3 million Ukrainian refugees have crossed the Polish border [13].

Around 16% of the arriving refugees were received in collective sites [3] called reception centres[1] which are located throughout the Polish territory.

Each reception centre has work shifts during which the volunteers must engage in carrying out activities related to psychological support services, translation and general medical and legal assistance.

Volunteers must be distributed in the various work shifts according to the requirements described in the next section [6, 1].

[1] As reported by UNHCR in [7]: "A reception centre is a location with facilities for receiving, processing and attending to the immediate needs of refugees."

3.1 Requirements

- Volunteers are committed to carrying out activities for the benefit of society by free will [19] and everyone can decide the days and times in which they are available, respecting the laws on working time limits;
- Each reception centre establishes how many volunteers it needs to cover a specific role for each shift;
- Each volunteer can provide assistance for only one reception center during the same day to avoid wasting time due to moving from one location to another.

3.2 Goals

For optimal coordination of volunteers to be able to provide humanitarian assistance, some goals must be set in order of priority from the lowest level to the highest level:

1. Coordinate as many different volunteers as possible, to avoid unused human resources.
2. Volunteers must offer humanitarian assistance in places as close as possible to their location of residence.
3. Volunteers must carry out humanitarian assistance activities for as many shifts as possible within the limits of the maximum time they can commit to.
4. Some critical roles need to be identified, those that are very important and have a greater impact. There should be no shortage of volunteers with skills capable of covering roles related to medical assistance.

Chapter 4

Answer Set Programming Encoding

The problem related to the coordination of volunteers in providing humanitarian assistance described in Chapter 3 has been represented in the form of an ASP program. As said before, the syntax used for this program is based on ASP-Core-2 Input Language Format [11]. The problem has been divided into subproblems. With this approach, the solution of each subproblem would imply the solution of the original problem [18]. The data model is described in Section 4.1 while the rules of the ASP program have been expressed in Sections 4.2 and 4.3.

4.1 Constructing the Data Model

The following predicates must be supplied as input to the program.

```
1 receptionCentre(Id,Location,Capacity).
2 volunteer(Id,Name,Location).
3 shift(Id,Rc,Day,Start,Duration).
4 canPerformRole(Volunteer,Role).
```

A reception centre (Line 1) is given by the geographical position in which it is located (e.g. town or city) and the maximum number of refugees it is capable of receiving. A volunteer (Line 2) is given by the full name and

the location of residence. Volunteers are associated with one or more roles that they are able to perform (Line 4). A shift (Line 3) is an interval of time characterised by a start and a duration both expressed in hours. It is also represented by the day and the reception centre `Rc` to which it refers. It is intentional having the duration of the shift instead of the end hour as this simplifies the implementation of the program in the next phases. Each atom on Lines 1-3 is characterized by a unique identification number to allow differentiating atoms having equal terms.

4.1.1 Reception Centre Requirements

```
1 requiredRoleForShift(Shift,Role,Min,Max).
2 requiredRoleAt(Rc,Role,Min,Max).
3 importantRole(Role).
```

The minimum and the maximum number of volunteers able to cover a specific role during a shift is defined by the instances on Line 1. The same requirement can be applied to all of the shifts of a reception centre `Rc` by the instances on Line 2. Roles are required for the reception centre to function properly and are crucial for some activities, they are expressed by instances on Line 3.

4.1.2 Availability and Time Preferences

```
1  maxHoursPerDay(Volunteer,Hours).
2  maxHoursPerWeek(Volunteer,Hours).
3  workingRights(Daily,Weekly).
4  day(D).
5  availableOn(Volunteer,Days).
6  isUnavailable(Volunteer).
7  isAbsentOnShift(Volunteer,Shift).
8  isAbsentOnDay(Volunteer,Day).
9  inoperative(Rc).
10 preferredShiftRange(Volunteer,Start,Duration,Day).
```

The total number of daily and weekly hours that each volunteer can employ in the various activities according to her/his preferences and availability is expressed by the instances on Lines 1-2. According to European Union directives [14], each volunteer must be granted a maximum limit of daily and working hours. The atom on Line 3 defines such minimum standards. The days of the week are expressed by the instances of the predicate on Line 4. Volunteers can be available on different days on the basis of their commitments and the time they want to dedicate to the voluntary service. In instances expressed on Line 5, days are represented by a list (e.g. ["mon","tue","wed"]). This structure allows to take advantage of the built-in predicates for manipulating lists offered by the library included in the DLV2 system [8]. A volunteer can no longer be available (Line 6) for an indefinite period of time (for example due to illness, other commitments or no longer wanting to carry out voluntary services). When the period of absence is known and concerns a specific shift or a specific day, it is expressed by instances on Lines 7-8. A reception centre can be temporarily or permanently inoperative (Line 8), for example, due to a natural disaster, missing equipment or other external factors. Possible hourly preferences of each volunteer are expressed by the instances on Line 10. The time interval for a specific day is characterized by a starting time and the maximum duration within which the volunteer expresses her/his availability to carry out voluntary work activities.

4.1.3 Location Preferences

```
1 maxLocationRange(Volunteer,Location,Range).
2 city(C).
3 distance(C1,C2,D).
4 withinRange(V,Rc,D).
```

A volunteer may be willing to carry out activities outside her/his place of origin. This preference is expressed through instances of the predicate on Line 1. It is characterized by the main location where the volunteer wants to provide voluntary service and the maximum distance (Range) indicated in kilometres, within which the volunteer has defined its availability.

Each city within which reception centres are located or which is relevant for showing the distance D expressed in kilometres between two cities C1 and C2 is expressed by the predicates on Lines 2-3. The facts mentioned above constitute the input of the subprogram reported in Section 4.3.

The instances of the predicate on Line 4 express the distance D between a volunteer V and a reception centre `Rc`, which is located within the maximum location range (Line 1). This predicate is the output of the subprogram described in Section 4.3 and it is an input for the subprogram described in Section 4.2.

4.2 Coordination of Volunteers

To find the Answer Sets, the program is based on the Guess and Check methodology. It consists of guessing all of the possible combinations of how volunteers can be assigned to a shift and which role they can perform during that shift.

The first step is to handle data incompleteness with the following rules:

```
1 isAbsentOnShift(V,S) :- shift(S,_,D,_,_), isAbsentOnDay(V,D).
2 requiredRoleForShift(S,R,Min,Max) :- requiredRoleAt(Rc,R,Min,Max),
    shift(S,Rc,_,_,_).
```

The absence of a volunteer `V` on a specific day `D` implies the absence of the volunteer in a shift `S` taking place on that specific day (Line 1). A role `R` required at a reception centre `Rc` implies that the role is required for all the shifts in that reception centre (Line 2).

The rule described below filters the volunteers who can perform a role `Role` during a shift `S` at a given reception centre `Rc`. Its purpose is to reduce the field of admissibility of the solutions guessed through the Guess and Check methodology and consequently speed up the execution times of the program.

```
1  canBeIn(V,S,R) :-
2      shift(S,Rc,D,_,Duration),
3      &member(D,Days;),
4      availableOn(V,Days),
5      not isAbsentOnShift(V,S),
6      not isUnavailable(V),
7      maxHoursPerDay(V,H),
8      Duration ≤ H,
9      withinRange(V,Rc,_),
10     canPerformRole(V,R),
11     requiredRoleForShift(S,R,_,_),
12     not inoperative(Rc).
```

A volunteer `V` can perform a role R during a shift `S` if and only if all of the following conditions are met:

- A volunteer is available on the day `D` in which the shift S takes place. For this purpose, the library of built-in predicates for list manipulation of the DLV2 system is used. Given the day `D` and a list `Days`, `&member(D,Days;)` is true iff `D` $\in$ `Days` [15] (Lines 2-4).
- A volunteer `V` is not absent on a shift `S` (Line 5).
- A volunteer `V` is not unavailable (Line 6).
- The number of daily hours `H` that a volunteer `V` can be employed in voluntary activities is less than or equal to the time duration `Duration` of a shift `S` (Lines 2,7-8).
- A reception centre `Rc` is located within the maximum distance defined by the volunteer `V`. This atom is further described in Section 4.3 (Lines 2,9).
- A volunteer `V` can perform a role `R` that is required for a shift `S` (Lines 10-11).
- A reception centre `Rc` is operative, meaning that only reception centres where activities continue regularly will be considered (Line 12).

The disjunctive rule described below is based on the Guess and Check methodology consisting of guessing all of the possible combinations.

```
1 inShift(V,S,R) | outShift(V,S,R) :- canBeIn(V,S,R).
```

A volunteer `V` is assigned to a shift `S` with role `R`. The disjunctive rule can be seen as two subsets, where only `inShift` must be considered as it consists of all the elements which belong to the answer set.

4.2.1 Maximum Working Hours

The number of daily and weekly working hours that each volunteer employs to carry out activities during shifts must respect the hourly availability of the volunteer himself and the working rights settled by EU directives, which are provided as input to the program according to the data model previously reported in Section 4.1.2.

First, the total number of daily and weekly working hours for each volunteer is calculated, according to the rules defined below:

```
1 totalDailyHours(V,T) :- volunteer(V,_,_), day(D),
2     #sum{H,S : inShift(V,S,_), shift(S,_,D,_,H)} = T.
3 totalWeeklyHours(V,T) :- volunteer(V,_,_),
4     #sum{H,V,S : inShift(V,S,_), shift(S,_,_,_,H)} = T.
```

In particular, the rule on Lines 1-2 calculates the total sum of the working hours of all the shifts which take place on the same day `D` and in which the volunteer `V` is assigned. The rule for calculating weekly working hours on Lines 3-4 differs from the previous rule as it sums the hours of all shifts to which a volunteer is assigned. Since shifts are assigned on a weekly basis, it is sufficient to add up the hours of all shifts to which a volunteer is assigned.

Second, all the answer sets in which the total number of daily or weekly scheduled working hours of a volunteer exceeds the maximum number of available working hours per day that (s)he is able to contribute are discarded through the following strong constraints:

```
1 :- totalDailyHours(V,T), maxHoursPerDay(V,Max), T > Max.
2 :- totalWeeklyHours(V,T), maxHoursPerWeek(V,Max), T > Max.
```

Finally, the daily and weekly working hours must not exceed the maximum limits established in the working rights as follows:

```
1 :- totalDailyHours(V,T), workingRights(Daily,_,_), T > Daily.
2 :- totalWeeklyHours(V,T), workingRights(_,Weekly,_), T > Weekly.
```

4.2.2 Time Frame Availability

Some volunteers have preferences about the time frame within which they can work, therefore volunteers cannot be assigned to shifts that don't satisfy their hour range preferences. However, only volunteers who have a time frame preference are filtered:

```
1 hasHourPreference(V,Day) :- preferredShiftRange(V,_,_,Day).
```

In particular, `Day` is the day of the week for which this preference is valid.

Then, it is necessary to check for which volunteers the time frame preference is satisfied:

```
1 satisfyHourPreference(V,Day) :-
2     hasHourPreference(V,Day), preferredShiftRange(V,Ps,Pd,Day),
3     inShift(V,S,_), shift(S,_,Day,Start,Duration),
4     Ps ≤ Start, Ps+Pd ≥ Start + Duration.
```

Not all volunteers have expressed any time frame preferences, therefore the rule above takes into consideration only volunteers who have a time frame preference (Line 2). The time frame preference of a volunteer `V` is defined by a preferred starting hour `Ps` and a preferred time duration `Pd`. It is satisfied if it meets these conditions: the shift starts at the same time or later than the preferred start time; the shift ends at the same time or earlier than the preferred end time. As the data model does not contain the end time, it is given by the sum between the start time and the duration.

Finally, all of the answer sets that do not satisfy the time frame preference are discarded through the following strong constraint:

```
1 :- inShift(V,S,_), shift(S,_,Day,_,_),
2     not satisfyHourPreference(V,Day), hasHourPreference(V,Day).
```

4.2.3 Reception Centre Requirements

A volunteer must not be assigned to shifts taking place in different reception centres during the same day:

```
:- inShift(V,S,_), inShift(V,S1,_), shift(S,Rc,Day,_,_),
   shift(S1,Rc1,Day,_,_), Rc ≠ Rc1, S ≠ S1.
```

To ensure greater productivity by volunteers and to avoid slowdowns during the activities carried out, each volunteer must cover at maximum one role during a shift. For this purpose, the following constraint is added:

```
:- inShift(V,S,R), inShift(V,S,R1), R ≠ R1.
```

In particular, a volunteer `V` cannot carry out activities related to two different roles `R1` and `R2` during the same shift `S`.

The number of volunteers carrying out activities for a specific role during a shift, must not exceed the maximum number of volunteers required for that shift:

```
:- requiredRoleForShift(S,R,_,Max), #count{V : inShift(V,S,R)} > Max.
```

4.2.4 Optimal Solution

Weak constraints are used to search for optimal solutions. In general, when the weak constraint cannot be satisfied in an answer set, a cost is assigned. The best optimal answer set is the one with the lowest cost to pay.

As reported in Chapter 3, the following goals must be achieved:

1. Maximize the number of different volunteers assigned to a shift.
2. Maximize the number of volunteers assigned to closer reception centres.
3. Maximize the number of shifts that a volunteer is assigned to.
4. Maximize the number of shifts with volunteers covering important roles.

The goals are ordered by priority from the less important to the most important ones.

Goal 1

The following weak constraint assigns a cost for each volunteer who is assigned to two different shifts:

```
1 :∼ inShift(V,S,_), inShift(V,S1,_), S ≠ S1. [1@1,S]
```

This weak constraint will increase the number of different volunteers assigned to a shift, therefore volunteers will be equally distributed and it will be taken full advantage of the number of people who are willing to volunteer.

Goal 2

The second goal is to maximize the number of volunteers assigned to closer reception centres. To achieve this, instances of the predicate generated as output from the solution of the subproblem described in Section 4.3 are needed. To ensure data consistency, the following rule is introduced:

```
1 withinRange(V,Rc,Distance) :- volunteer(V,_,Location),
     receptionCenter(Rc,Location,_), Distance = 0.
```

In particular, the rule above generates as reception centres that are within the range of the volunteer, all those that are in the same location of residence. These reception centres will represent the best choice since the location of the

reception centre coincides with that of the volunteer, therefore the distance is equal to zero.

The following weak constraint achieves the goal of the second priority:

```
:~ inShift(V,S,_), shift(S,Rc,_,_,_), withinRange(V,Rc,D). [D@2,V]
```

The weak constraint assigns as a cost to the answer set, the distance `D` between the volunteer `V` and the reception centre `Rc`.

Goal 3

The following weak constraint assigns a cost for each volunteer that can perform a role but is not assigned to any shift.

```
:~ volunteer(V,_,_), not inShift(V,S,R), canPerformRole(V,R),
   shift(S,_,_,_,_). [1@3,V,S,R]
```

Goal 4

The following rule calculates the number of volunteers still needed per shift for a specific role:

```
uncoveredRoles(S,X,R) :- requiredRoleForShift(S,R,Min,_),
    #count{V : inShift(V,S,R)} = N, N < Min, X = Min - N.
```

The rule counts all of the volunteers assigned to a shift with the required role `R`. The number `X` represents the volunteers still required for the shift. It is given by the difference between the minimum volunteers required `Min` and the actual number of volunteers `N` assigned to the shift `S`.

The following weak constraint is used to maximize the number of shifts covering important roles:

```
:~ importantRole(R), uncoveredRoles(S,N,R). [N@4,S]
```

In particular, the weak constraint will assign a cost measuring the number `N` of volunteers still needed to cover the important role `R` during a shift `S`.

4.3 Reachability of Cities Within a Distance Range

Given a starting city and a distance range, it is required to find all reachable cities within the specified range and the distance travelled for each destination city. This subprogram generates the input necessary for Goal 4 of the subprogram reported in Section 4.2.4. A set of cities and the distances that can be travelled between some of them is known. The model of these facts is reported in Section 4.1.3 This problem can be viewed as a set of nodes connected with arcs with different costs - an undirected graph. The solving process for this problem is based on the Traveling Salesman Problem in [21].

For greater readability and to maintain integration between subprograms, instances of the following predicates are generated:

```
1 node(X) :- city(X).
2 arc(X,Y,Cost) :- distance(X,Y,Cost).
3 arc(X,Y,Cost) :- arc(Y,X,Cost).
4 start(V,X) :- maxLocationRange(V,X,_).
5 end(X) :- receptionCenter(_,X,_).
```

A city is a node (Line 1). The distance between the two cities is an arc (Line 2). Generating arcs for both directions to have a bilateral oriented graph (Line 3). The starting city `X` for volunteer `V` is the starting node (Line 4). The location `X` of a reception centre is an end node (Line 5).

The encoding has been created based on the Guess and Check methodology. Since there can be many reception centres, therefore many end nodes. The following disjunctive rules are expressed:

```
1 {inRealStart(V,Start) : start(V,Start)} = 1.
2 {inRealEnd(End) : end(End)} = 1.
```

In particular, for the next steps of the program, only one start node and only one end node will be considered respectively as the real start and end nodes.

The rest will be treated as intermediate[1] nodes.

The next step is to apply the Guess and Check methodology to guess the answer sets in which the real end node is reached. The following disjunctive rule is applied:

```
1 inPath(V,X,Y,C) | outPath(V,X,Y,C) :- arc(X,Y,C), start(V,_).
```

The volunteer `v` will be taken into account, to allow proper integration with the subprogram of Section 4.2.4.

With the following rules, nodes are being explored:

```
1 reached(V,X) :- inRealStart(V,X).
2 reached(V,X) :- inPath(V,Y,X,C), reached(V,Y).
```

The first node considered reached is the starting node itself (Line 1). Then a node directly connected to the reached one, is explored. (Line 2).

A real end node does not have any outward arcs, because when reached, no other nodes must be explored:

```
1 :- inPath(V,X,Y,_), inRealEnd(X).
```

A real start node does not have any inward arcs, otherwise, it means the path is being repeated:

```
1 :- inPath(V,X,Y,_), inRealStart(V,Y).
```

There must not be two arcs starting at the same node:

```
1 :- inPath(V,X,Y,_), inPath(V,X,Y1,_), Y ≠ Y1.
```

There must not be two arcs ending at the same node:

```
1 :- inPath(V,X,Y,_), inPath(V,X1,Y,_), X ≠ X1.
```

The path travelled by the volunteer must reach the real end node:

```
1 :- inRealEnd(X), not reached(V,X), inRealStart(V,_).
```

The subset of arcs in the path must not include an element if the node has not been reached:

```
1 :- inPath(V,X,Y,_), not reached(V,Y).
```

[1] A node along the way between a start node and an end node.

The total cost of the arcs travelled from a start node to an end node is calculated through the following rule:

```
totalDistance(X) :- #sum{C,V,A,B : inPath(V,A,B,C)} = X.
```

A requirement is that the actual distance travelled from the start location point of the volunteer, must be less than the maximum distance range requirement of the volunteer. The answer sets which do not meet this requirement, are discarded through the following strong constraint:

```
:- totalDistance(X), maxLocationRange(V,_,Y), inRealStart(V,_), X>Y.
```

Finally, instances of the rule described below are generated:

```
withinRange(V,Rc,D) :- inRealStart(V,X), inRealEnd(L),
   receptionCenter(Rc,L,_), totalDistance(D).
```

In particular, the rule generates predicates indicating a reception centre `Rc` that is within the preferred distance range of volunteer `V` and the actual distance `D` between the two locations. The predicate is structured in this way to be used in weak constraints of the subprogram in Section 4.2.4.

Chapter 5

Scalability Analysis on Simulated Data

The subprograms were tested on an Intel Core i7 2.8 GHz. The ASP system used was DLV2. Geographical position data of the reception centres is based on real data provided by a humanitarian information service provided by the United Nations Office for the Coordination of Humanitarian Affairs [5]. Roles carried out in reception centres are based on data of an already existing reception centre in Poland [2]. The rest of the data is based on assumptions resulting from the analysis conducted in the preliminary examination of the case study reported in Chapter 3. The data for the simulation has been generated on an Excel spreadsheet with distribution in classes with different frequencies.

5.1 Coordination of Volunteers

The simulated data used for Figure 5.1 is based on the distribution of volunteers in shifts for 17 reception centres. For the simulation, each volunteer was able to commit for up to 2 different roles. Among all, 5% of the volunteers carry out an important role. This factor influences the results as it is related to the highest priority goal of the ASP program as reported in Section 3.2.

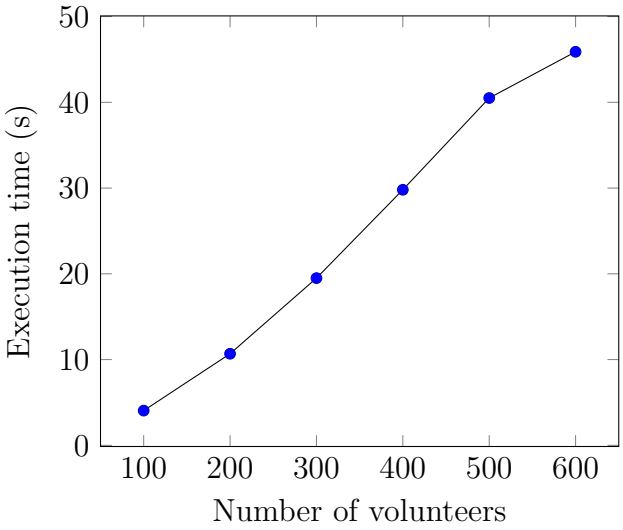

Figure 5.1: Scalability Analysis

As the number of volunteers increases, so does the number of shifts in which they are assigned, keeping the increase in execution time constant. An increase in the number of shifts covered by volunteers was also observed with the increase in the number of volunteers. The program is therefore able to assign volunteers on a small and large scale for less than a minute. However, it is important to remind that this data is highly variable in real-life situation as it is strictly dependent on the availability and preferences of volunteers which is unpredictable.

5.2 Reachability of Cities Within a Distance Range

The data used for the simulation of the city reachability problem is based on the use case "Refugee Influx from Ukraine into Poland" retrieved by [5, 17]. The main cities where reception centres have been located are considered. The cost associated with each arc in the graph is based on the approximate real distance between cities expressed in kilometres.

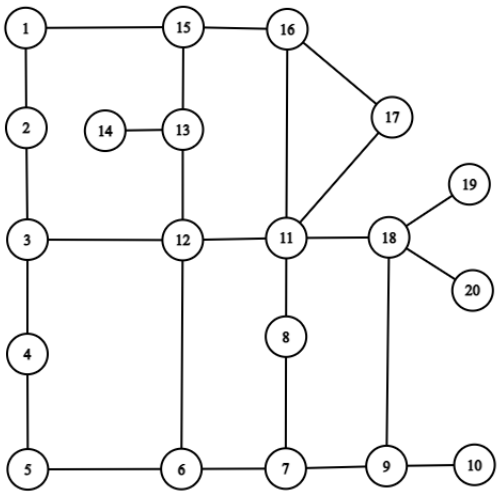

Figure 5.2: Map of Poland Represented as an Undirected Graph

The nodes in Figure 5.2 portray the following cities:

1. Szczecin; 2. Gorzów Wielkopolski; 3. Poznań; 4. Wrocław; 5. Opole; 6. Katowice; 7. Kraków; 8. Kielce; 9. Rzeszów; 10. Przemyśl; 11. Warszawa; 12. Łódź; 13. Toruń; 14. Bydgoszcz; 15. Gdańsk; 16. Olsztyn; 17. Białystok; 18. Lublin; 19. Chełm; 20. Tomaszów Lubelski.

The city of Kraków, associated with Node 7, has been chosen as a starting node for the test.

The graph reported in Figure 5.3 shows how the execution time varies as the distance increases. In particular, as expected, as the distance increases, more paths are explored, therefore the problem becomes harder and then the solver needs more time to find the solutions. However, even in the case with a range of 800 km, the solver is able to find solutions within 30 seconds.

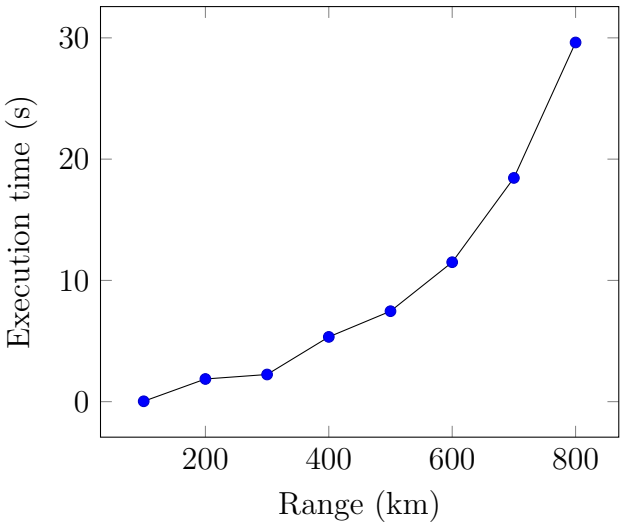

Figure 5.3: Scalability Analysis

Chapter 6

Conclusions

In this work, whose main objective is to create a system based on Answer Set Programming for the coordination of volunteers during a humanitarian crisis, issues related to the definition of strategies to maximize the number of available human resources have been addressed. These strategies have been transformed into ASP sub-programs that interact with each other. The subprogram for calculating the reachability of cities serves as input to the subprogram dealing with the actual coordination of volunteers. The approach used in both subprograms is based on the Guess and Check methodology. Then, the ASP system was tested on synthetic data created in such a way as to perform also a scalability analysis. The results of the experimental analysis are positive since the ASP program is able to solve the modelled problem within a few seconds.

In conclusion, the proposed ASP program represents only the first building block in providing a tool for managing a humanitarian crisis on a global scale. The influencing factors are numerous and concern different areas. Other subprograms can be linked to the developed system.

Bibliography

[1] How Volunteers Help Refugees. URL: https://www.unhcr.org/en-my/how-volunteers-help-refugees.html. Consulted in date 20/04/2022.

[2] New refugee reception centre opens in overwhelmed Poland. URL: https://www.nrc.no/news/2022/march/new-refugee-reception-centre-opens-in-overwhelmed-poland. Consulted in date 20/04/2022.

[3] Refugees from Ukraine in Poland - Profiling Update (July 2022). URL: https://data.unhcr.org/en/documents/details/95735. Consulted in date 22/09/2022.

[4] Stabilisation of emergency measures: Poland's refugee reception system one month after the Russian attack on Ukraine. URL: https://www.asileproject.eu/polands-refugee-reception-system-one-month-after-the-russian-attack-on-ukraine. Consulted in date 04/05/2022.

[5] Ukraine: Complex - Reception Points in Poland. URL: https://reliefweb.int/map/poland/ukraine-complex-reception-points-poland-21-mar-2022. Consulted in date 22/04/2022.

[6] Ukraine Regional Refugee Response Plan And Flash Appeal. URL: https://data.unhcr.org/en/documents/details/92258. Consulted in date 04/05/2022.

[7] *UNHCR Master Glossary of Terms*. Rev.1. UN High Commissioner for Refugees (UNHCR), June 2006. URL: https://www.refworld.org/docid/42ce7d444.html.

[8] Mario Alviano, Francesco Calimeri, Carmine Dodaro, Davide Fuscà, Nicola Leone, Simona Perri, Francesco Ricca, Pierfrancesco Veltri, and Jessica Zangari. *The ASP system DLV2*. 2017.

[9] Mario Alviano, Carmine Dodaro, Nicola Leone, and Francesco Ricca. Advances in WASP. In *LPNMR*, volume 9345 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2015.

[10] Mario Alviano, Wolfgang Faber, Nicola Leone, Simona Perri, Gerald Pfeifer, and Giorgio Terracina. The Disjunctive Datalog System DLV. In Oege de Moor, Georg Gottlob, Tim Furche, and Andrew Sellers, editors, *Datalog Reloaded*, pages 282–301, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[11] Francesco Calimeri, Wolfgang Faber, Martin Gebser, Giovambattista Ianni, Roland Kaminski, Thomas Krennwallner, Nicola Leone, Francesco Ricca, and Torsten Schaub. Asp-core-2: Input language format. *ASP Standardization Working Group*, 2012.

[12] Francesco Calimeri, Davide Fuscà, Simona Perri, and Jessica Zangari. I-DLV: The New Intelligent Grounder of dlv. AI*IA 2016: 192-207.

[13] Maciej Duszczyk and Paweł Kaczmarczyk. War and migration: the recent influx from ukraine into poland and possible scenarios for the future. *CMR Spotlight*, 4(39), 2022.

[14] Vito Leccese. Directive 2003/88/ec concerning certain aspects of the organisation of working time. pages 1285–1321, 2018.

[15] Nicola Leone. Disjunctive Logic Programming: Knowledge Representation Techniques, Systems, and Applications, 2017.

[16] Vladimir Lifschitz. *Answer set programming*. Springer Heidelberg, 2019.

[17] Google Maps. Distances Between Cities in Poland. Retrieved 12 November 2022.

[18] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.

[19] G Porcaro. Volunteering Charter-European Charter on the Rights and Responsibilities of Volunteers. 2012.

[20] Francesco Ricca, Giovanni Grasso, Mario Alviano, Marco Manna, Vincenzino Lio, Salvatore Iiritano, and Nicola Leone. Team-building with answer set programming in the Gioia-Tauro seaport. *Theory and Practice of Logic Programming*, 12(3):361–381, 2012.

[21] Francesco Ricca Wolfgang Faber, Nicola Leone. Answer Set Programming. *Encyclopedia of Computer Science and Engineering (Benjamin Wah, ed.) Vol. 1*, pages 149–162, January 2009.