Optimal Placement of Acoustic Sources in a Built-up Area using $CLP(\mathcal{FD})$

F. Avanzini[2], A. Belussi[1], A. Dal Palù[1], A. Dovier[3], and D. Rocchesso[1]

[1] Dip. di Informatica, University of Verona (Italy).
`belussi@sci.univr.it`, `aledalpa@libero.it`, `rocchesso@sci.univr.it`

[2] Dip. di Ingegneria dell'Informazione, University of Padova (Italy).
`avanzini@dei.unipd.it`

[3] Dip. di Matematica e Informatica, University of Udine (Italy).
`dovier@dimi.uniud.it`

Abstract. In this paper we face the problem of covering a physical area with acoustic signals emitted by different sources, and we present a declarative and effective solution based on Constraint Logic Programming over Finite Domains. After a testing phase on toy examples, we have successfully applied the code to the solution of a real-world placement problem.

Keywords. *Constraint Logic Programming, Real World Applications.*

1 Introduction

In this paper we present a declarative and effective solution to the problem of covering a physical area with acoustic signals emitted by different sources. More in detail, we are interested in finding the minimal number n of sources and their spatial allocation so that a signal power greater than a fixed threshold reaches all the specified area. We need to consider some source physical parameters (e.g., signal strength, direction, admissible places for sources) as well as environmental interferences with the signal over the area (e.g., signal absorption, diffusion, etc.). The problem can be specialized to that of adding acoustic alarms over a wide non-regular area (e.g., for high tide, fog, bombing, bacteriological attack). The problem is also related to the mobile phones network covering problem [3].

We face the problem using Constraint Logic Programming over Finite Domains ($CLP(\mathcal{FD})$—see e.g. [6]) and in particular, the clpfd library of SICStus Prolog [12]. The map of the built-up area is abstracted by a finite size grid. Each cell retains information concerning the mean and the maximum height of the buildings, the mean density of the materials of the buildings, the percentual of water (sea channels, rivers, ...), and a Boolean flag stating if the cell should be considered or not in the minimization phase. The signal propagation function is studied in detail with the assumption that in each cell of the grid the signal pressure is constant. We can assume to have a (possibly, very large) set of admissible places for the acoustic sources. In this way, the individual effect of each admissible source on each cell can be studied *before* the minimization phase. This, of course, allows to simplify the computation in the solutions' search phase. But it also allows us to compute the intensity maps using the results of *ad-hoc* sound simulators, such as SoundPLAN [11].

The solutions' search is then carried on by the built-in predicate `labeling`. This predicate allows several parameters in order to have *a priori* control on the search tree. With real-size simulations on various examples we have chosen the parameters that better fit this problem.

To give a taste of the effectiveness of the declarative approach, we tackle the (real) problem of optimal placement of sirens for the *acqua alta* (high tide) problem in Venice. We compare our results with the existing distribution of sirens in this case. The measured execution time and the form of the computed solutions are encouraging.

The paper is organized as follows: in Section 2 we discuss some relevant related work, while in Section 3 we formally describe the minimization problem. In Section 4 we show how to extract the required data from a geographical Database, and in Section 5 we present the sound propagation model. In Section 6 we discuss precisely the CLP implementation and the preliminary tests done to improve efficiency. In Section 7 we show the results on the Venice data. Some conclusions are described in Section 8.

2 Related Work

The problem presented is an optimization problem over finite domains and, as such, it has a lot of similarities with several other problems (see e.g. [6]). As far as we know, two works in literature are the closest to ours.

The first one is [1]. In [1], continuing the work done in [9], where declarative representation problems for the Venice lagoon are discussed, the authors describe a *Decision Support System* for helping the Venice Water Magistracy to control the pollution effects over that famous natural environment. The tool that they develop in CLP allows to compute the exact pollution concentration values on each point of the lagoon (on measured input data to be tested every day). If, in some points, the concentration bounds are not fulfilled, there are utilities for correcting the situation, either suggesting a *reduction* of the pollution generated by some sources (that can be implemented in practice, e.g., by forcing the closure of a factory), or suggesting a *relocation* of the sources (difficult to be implemented in practice; in this cases it is easier to change the bounds with a new law). There are some similarities with our work, such as the grid abstracting the geographical area, and the precomputations of the partial effects of each pollution source. However, their system is a decision system that must react dynamically to a change of pollution generation suggesting a new, feasible, solution. In our case, instead, the possible set of sources and the maximum strength of each generated signal are fixed *a priori* and we look for the *minimum* subset of them that must be turned on in order to ensure a suitable signal level in each point. Moreover, the signal propagation function on a urban environment is very different to that of pollution in lagoon.

The second one is [3]. In that paper the authors use Constraint Programming techniques for the Optimal Placement of Base Stations in Wireless Indoor Communication. The problem is that of finding sources locations inside and outside buildings so as each point of interest is covered by a signal exceeding a fixed threshold. The main difference with our approach lays in the granularity of the problem. In [3] the authors consider all details of a single building: walls, ceilings, different materials and so on. Each point can be used to install a source; each point must be covered by at least one source. The direct encoding of the problem does not work since the relations are too complex. The authors then develop a clever dual version of the problem that becomes tractable and they implement using it Frühwirth Constraint Handling Rules [2]. In our case, instead, we use a more rough abstraction of a geographical map based on cells in which we considered only some average parameters (c.f. Section 3). A (finite) set of possible locations for sources is given a priori (not all possible places of an urban environment are allowed/suitable to contain a source). Moreover, in our model, two or more sources add their effects on a single cell: it is not needed that each point is covered by a single source. In our case, as proved by the running time on the example in Section 7, a direct encoding of the problem is sufficient for a realistic implementation in SICStus Prolog.

3 Technical Definition of the Problem

In this section we provide a formal definition of the problem.

Area. The physical, built-up, area dealt with is abstracted by a finite grid of cells. Let A be a two-dimension array (a grid) representing finitely the area. For each cell $A[x, y]$ of the grid we are interested in a set of attributes:

- the density of buildings in the cell BuildDensity,
- the density of (water) canals in the cell CanalDensity,
- the mean height of the buildings in the cell HeightMean,
- the maximum height of a building in the cell HeightMax, and
- a Boolean flag, set to true if the cell has to be covered by the signal (*active* cell), to false, otherwise.

The first four fields can be automatically obtained from a geographical Database representing explicitly the map (see Section 4). The last field, instead, must be explicitly decided as part of the input of the problem. Each cell location $\langle x, y\rangle$ assumes values on the finite domain: $\{1, \dots, \mathsf{Dimx}\} \times \{1, \dots, \mathsf{Dimy}\}$.

Pool of sources. Let P be a set of possible locations for the acoustic sources. We refer to P as to the *pool of sources.* For instance, P could be the set of locations of buildings higher than a fixed level. For each location we also store:

- the maximum power of the signal emitted (max_pow_source) and

– an angular region where the signal is propagated (defined by an interval of angles $\langle \alpha, \beta \rangle$).

These two parameters can depend by urban constraints, such as nearness to a church or to a concert hall. We say the a source is *'on'* if it emits some signal.

Propagation function. The *acoustic propagation function* f is a function that models the physics of the acoustic signal propagation. The details on this function are discussed in Section 5). Here we only stress that we assume that f has a *discrete* behavior; namely, all the points of the same cell $A[x, y]$ receive the same signal strength. Each source S in the pool P has an independent contribution $\mathsf{Energy}(S, x, y)$ in $A[x, y]$. In our model, the global signal energy on a cell $\langle x, y \rangle$ is simply:

$$\mathsf{Energy}(x, y) = \sum_{S \in P, S \text{ } is \text{ } on} \mathsf{Energy}(S, x, y).$$

The Problem. The problem we deal with can be formalized as follows: given a pool P, a grid A, a propagation function f, a threshold value t, and a number $n \leq |P|$, the *n-sources location problem* is that of finding a subset of n elements of P such that in each active cell $A[x, y]$, the signal strength $\mathsf{Energy}(x, y)$ is greater than or equal to the threshold t. If no n-elements subset of P satisfies the requirements, return false.

The minimization version of the problem can be immediately formulated: find *the minimum* $n \leq |P|$ *and the position for the* n *sources* that admits a solution for the above defined version or return false if such n does not exists.

Of course, the two problems are strongly related. The minimization problem clearly implies the decision one. On the other hand, once a solution for the former is provided, the latter can be easily solved by guessing n by bisection. For instance, start with $\mathsf{min} = 0$, $\mathsf{max} = |P|$. Try with $n = \lceil (\mathsf{min} + \mathsf{max})/2 \rceil$. If the answer is yes, update $\mathsf{max} = n$, otherwise $\mathsf{min} = n$ and repeat. In $\log |P|$ iterations the minimum n is found (if it exists).

Constraints. Several data structures and relative constraints are involved with this problem. Assume that the pool $P = \{S_1, \ldots, S_p\}$.

– An array $\mathsf{Sources} = [V_1, \ldots, V_p]$ of values of emission for the sources $S_1, \ldots, S_p$. It must hold that $V_i \in [0, \mathsf{MaxPower}]$ where $\mathsf{MaxPower}$ is the maximum power that can be emitted by a source.
– A Boolean array $\mathsf{On\_Sources} = [B_1, \ldots, B_p]$ stating if the source S_i is on ($B_i = 1$) or off ($B_i = 0$).
– The main constraint: for each active cell $A[x, y]$, $\mathsf{Energy}(x, y) \geq t$, where t is the input threshold.
– A set of possible constraints of the form $\mathsf{Energy}(x, y) < t'$ for cells $A[x, y]$ where a huge value t' for the signal is not admitted (e.g., when concert halls are in $A[x, y]$).

– We also add a constraint imposing a minimum (Euclidean) distance between two *on* sources. This can be imposed as:

$$i \neq j \land B_i = 1 \land B_j = 1 \rightarrow \mathsf{distance}(S_i, S_j) \geq \mathsf{Min\_distance}$$

Other constraints can be added to cut the search tree.

4 Grid Calculation

The attributes of the grid cells are computed by a set of geographical data (that can be, with difficulty, obtained by the suitable offices). These datasets are organized in thematic layers containing both spatial data (in vector format) and descriptive data (in alphanumeric format) according to the geo-relational data model [5, 10, 4]. In the real-world example that we have studied, these layers contain the following information

$L_{Buildings}$**:** This layer contains a set of polygons representing the buildings. For each polygon the area is precomputed and stored.

L_{Height}**:** This layer contains a set of points for which the height with respect to the sea level is measured. Typically, these points are the highest buildings of the area.

L_{Canal}**:** This layer contains a set of polygons representing the water entities (rivers, canals, etc.) occurring in the area.

A graphical example of the representation of the layers content is shown in Figure 1. The grid of cells with the required attributes can be obtained from the geographical data through the following steps.

1. The first phase is the *discretization* phase. We decide which is the size of each cell grid (e.g., 50m) and, looking at the maximum x and y coordinates of the data, we compute the size Dimx, Dimy of the matrix we will use (we call this layer L_{Grid}).
2. Then we perform an *overlay* operation [4] between the L_{Grid} and $L_{Buildings}$ in order to compute for each grid cell the density of buildings (attribute BuildDensity). This operation can be implemented by using the tools of a Geographical Information System (GIS). The algorithm used is described as follows. for each cell c in L_{Grid}:

$$\mathsf{c.BuildDensity} = \frac{\sum_{\mathsf{p} \in L_{Buildings}} \mathsf{area(p.geo \cap c.geo)}}{\mathsf{area(c.geo)}}$$

   where the c.BuildDensity represent the value of the attribute BuildDensity on the cell c; p.geo is the form of the polygon representing p (its geometry) and c.geo represents the form of the grid cell c.
3. By performing an *overlay* operation between L_{Grid} and L_{Height}, we compute the mean and maximum height for each grid cell (attributes HeightMean and HeightMax).

4. Finally, an *overlay* operation between L_{Grid} and L_{Canal} produces the attribute CanalDensity.

The layer L_{Grid} is converted into an ASCII file and then in a set of Prolog facts, as shown in Section 6.

5 The acoustic problem

In this section we enter in some details on the problem of acoustic propagation within an urban environment. The threshold t (see Section 3) can be determined through psychoacoustic considerations: a common assumption states that the sound pressure level of the alarm signal must be at least 15 dB above the noise level, in order for the alarm to be clearly perceived. If an average noise level of 60 dB is assumed, then a good choice for the threshold is therefore $t = 75$ dB.

The propagation function f is less easily determined, and must be specified by suitably simplifying the acoustic problem. In particular, it is assumed that the phases of signals coming from different sources are randomly mixed at the listening point (the assumption is especially valid in complex urban environments), so that the intensities of the component tones can be summed up constructively [8]. This assumption allows separate computations of the sound fields of each source in the pool P, regardless of the nature of the sound signals being emitted.

As a first approximation, a source is considered to be a point source. If it is omnidirectional, then in a free-field the sound propagates along spherical surfaces and sound intensity on each spherical surface is inversely proportional to the square of its radius [8]. Free-field propagation is clearly a very rough approximation for an urban environment, where sound is absorbed, reflected and diffracted by buildings, ground, and green areas. However, such approximation can be reasonably assumed if the sources are located on high spots (as it is in most real cases), such as bell-towers.

The model can be further refined if directivity information is available for the sound sources. Specifically, when the source is not omnidirectional then the radiated power is angle dependent, and this dependency can be determined from the directivity pattern. Wind effects can be also taken into account: given the wind velocity and direction, the actual sound velocity is the vectorial sum of the sound velocity in still air with the wind velocity, and the propagation pattern is consequently modified. The sound field produced by each source can be computed using *ad hoc* tools for noise propagation simulation, such as SoundPLAN [11]. For instance, in Figure 1 we report the automatic generated propagation function related to a siren on the world-famous S. Marco Bell Tower in Venice. Since we implement the sound propagation computation as a preprocessing step, we can use these results. Currently, we are experimentally extracting from SoundPLAN results (as that in Figure 1) a realistic propagation function.

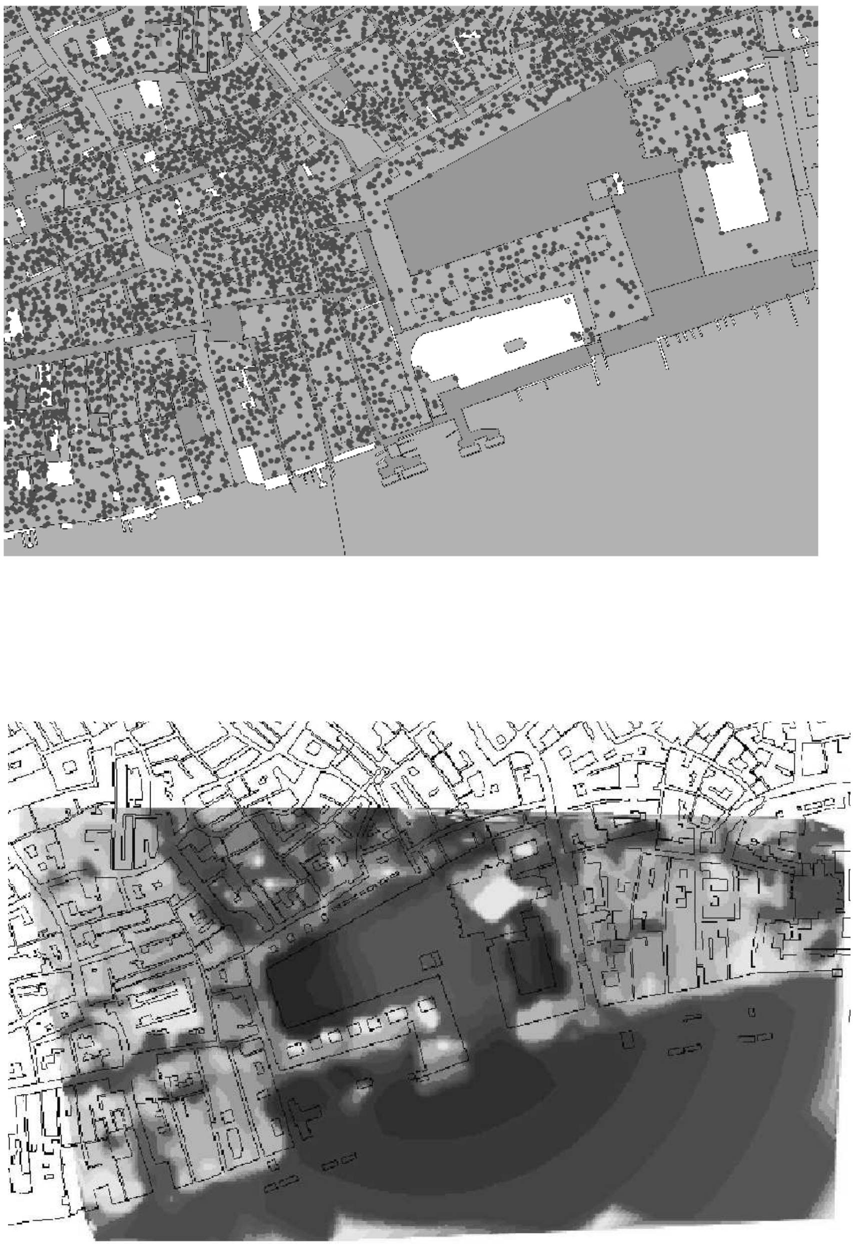


Fig. 1. A portion, including S. Marco Square, of the geographical datasets concerning Venice: polygons denote buildings, where points denote the known height measures. In the lower diagram it is reported the sound propagation of a siren on S. Marco Bell Tower, as computed by SoundPLAN.

6 CLP Implementation

The problem can be directly encoded into a Constraint Logic Programming Language. First we focus on the representations of the grid A, the pool P, and the propagation function f. Then we discuss the main predicate.

6.1 Representation of the Geographical Area

We need to represent a matrix with the five parameters BuildDensity, CanalDensity, HeightMean, HeightMax, and the 'active' Boolean flag (see Section 3). We represent the matrix in the following way:

– We add a fact for each row of the associated matrix.
– Instead of storing a matrix of tuples we choose to store 5 simpler matrices of elementary values.

Using these tricks, it is possible to access quickly a row, using the fast (hash-table based) techniques for retrieving the suitable fact. We can access the element in a row with the built-in predicate nth of the library list of SICStus Prolog. This built-in predicate is faster than the similar predicate element of the library clpfd. Anyway, if the size of the matrix grows, it would be better to use the library arrays of SICStus Prolog, where arrays are stored using trees and the element access can be performed in logarithmic time.

Thus, the facts concerning the matrix A are of the form:

```
build_density_row(1,[0.000000,...,0.000000,0.000000]).
build_density_row(2,[0.000000,...,0.075900,0.339400]).
...
canal_density_row(1,[0.000000,...,0.000000,0.000000]).
canal_density_row(2,[0.000000,...,0.075900,0.339400]).
...
mean_height_row(1,[0.000000, 0.000000, 0.000000,...,0.000000]).
mean_height_row(2,[0.000000,23.23444,61.256679,...,0.000000]).
...
max_height_row(1,[0.000000,  0.000000,  0.000000,...,0.000000]).
max_height_row(2,[0.000000,167.000000,153.000000,...,0.000000]).
...
flag_row(1,[0,0,0,0,...,0,1,0,0]).
flag_row(2,[0,0,1,1,...,0,1,0,0]).
...
```

We also explicitly add two facts representing the matrix' size.

```
dimx(106).
dimy(66).
```

6.2 The Pool of Sources

One of the inputs of the problem is a list of possible locations for the signal sources, together with some information for each source. This information is added using three facts. First, the list of possible locations for the sources, which is of the form:

```
pool([[20,14],[14,7],[18,19],[26,6],...,[20,30]]).
```

The other two parameters for the sound sources are stored in different facts: `max_pow_source` that states, in a suitable scale, the maximum power admitted for a source and `orient` that constrains, for each source of the pool, the angular interval $[\alpha, \beta]$ admitted for propagation.

```
max_pow_source([1,3,2,1,3,1,4,1,4,...,1,1,1]).
orient([[0,360],[0,180],[180,240],...,[0,360],[0,360],[0,360]]).
```

6.3 Sound Propagation Computation

Constraint-based programming is typically split into two phases. A constraint definition/introduction phase, that runs once, and a constraint-based search phase. This second phase should be kept as simple as possible in order to improve efficiency. For doing that, we compute *a priori* the effects of the various acoustic sources on each point of the grid. Precisely, for each source in the pool P and for each cell we `assert` a fact containing the signal strength in that cell referred to the appropriate source. Each fact asserted is of the form:

```
compute_energy(S,Spos,X,Y,P).
```

where `S` is the index of the current source, `Spos` the coordinates of the position of the source `S`, and `P` is the power associated to the cell $A[\texttt{X},\texttt{Y}]$ according to the sound model. Basically, we compute the distance between the cells $[\texttt{X},\texttt{Y}]$ and `Spos`. We pick the orientation of the source, its maximum power, and we compute the energy value depending on distance, orientation, and the average height and density of all the cells between the two considered. For a realistic propagation function, we are currently using the results of a computation executed by SoundPLAN [11].

Because of the sound function shape, this choice allows us to assert only few elements over the map and approximate to zero the others. In the computation of the sound intensity associated to a particular cell, we consider the decibels measured at one meter from the source and use the propagation model to obtain the sound energy in the cell. We decide to store the energy value divided by the threshold of energy considered not relevant to the hear, to avoid large numbers. By using a decibel scale for intensity, we trim energies higher than a fixed threshold, so that the upper bounds of the domains are heavily reduced. The threshold should be chosen in such a way that the energy is dominant over possible contributions from other sources. At the end of this phase, we convert all real values associated to the energy in a cell to integer numbers, cutting them to the lower integer. This choice allows to avoid the real number management penalty in the search phase. Moreover, any feasible solution of this approximation will be also a solution of the original one, since the real sound pressure is surely (slightly) higher. Basically, for each source S and cell C, we:

– remove any possible previous assertion involving C and S,
– compute the energy present in C, because of the emission of S,
– assert the information computed.

After this preprocessing phase we have a map of sound propagation for each source, generated by the function f. In this phase we also add a fact storing the threshold value t. For instance:

```
threshold(75).
```

6.4 Main Predicate

The top-level clause of the $CLP(\mathcal{FD})$ solution of the problem is a classical constrain & generate clause:

```
go( N ):-
   dimx(Dimx),
   dimy(Dimy),
   pool(P),
   length(P,Np),
   length(Sources,Np),
   compute_pressures(Np,Dimx,Dimy),
   constrain(Sources, N ,Dimx,Dimy),
   reverse(Sources,S),
   labeling([...parameters...],S),
   write(Sources).
```

The main parameter `N` is the number of allowed sources. The x and y sizes of the grid (`Dimx` and `Dimy`) are retrieved from the facts storing them. The pool P of sound sources is also here retrieved. Then an array `Sources` of variables of the same length `Np` as `P` is generated. Then, `compute_pressures` computes the local effects of each source of the pool `P` as explained in Section 6.3.

6.5 Constraint generation

In the constraint generation phase (predicate `constraint`), we first force the `Np` variables in `Sources` to be defined over the domain `0..MaxPower`, representing the powers of each source. A set of constraints is set to limit each source's power if needed. The auxiliary Boolean list `On_Sources` of `Np` variables is used to specify if each source is currently used or not. This constraint can be formalized as:

$$\texttt{Sources(i)} > 0 \Leftrightarrow \texttt{On\_Sources(i)} = 1$$

and defined by means of reified constraints in SICStus Prolog. The constraint

$$\texttt{sum}(\texttt{On\_Sources}, \leq, \texttt{N})$$

states that we want to use up to N sources to solve the problem. Finally, we impose that each active cell has to be covered by at least the threshold t (stored in the fact threshold(t), as shown in Section 6.3). The summation of individual contributions is easily accomplished with the built-in predicate scalar_product. We also add the constraint stating that two sources in the final solution must have a distance greater than a certain value. For instance

```
minimum_distance(700).
```

states that this minimum value between two different sources is 700 meters.

6.6 labeling Phase

We assume that the list of sources is provided in decreasing order of importance. To maintain this preference we reverse that order during the search labeling phase, because the Prolog solver starts exploring the solution tree from the leftmost variable in a list. Since the min selection strategy is chosen (see Section 6.6), the variables set to higher values are usually on the right.

We discuss here some benchmarks we run over a simulated problem. We consider a problem defined with a map of 6000 elements and a simplified version, with 1500 elements. In particular, we compare the different variable choice strategies that are admissible parameters for the built-in predicate labeling of SICStus Prolog. Looking at the Table 1, note how the min strategy drastically decrements the time required to explore the whole search tree. A complete strategies' comparison is provided with the example with 1500 elements. In this case, once again the min strategy is the best. Conceptually this strategy is the one that chooses the leftmost variable with the lowest bound domain. In fact only n from $|P|$ sources are used and the other values are set to 0, thus the min strategy can inform the search tree that, in general, 0 values are preferred. We try to modify the way a variable is managed, adding the bisect option. In this case there is a degradation of the running time. Hence, we can conclude that the default option step in combination with the increasing ordering up, provides the best labeling strategy for this problem. The benchmarks used a Amd Duron machine with 1.0 GHz processor and 256Mb Ram.

7 Computations

We have finally tested our solution on a real-world problem. The problem is that of placing high-tide Sirens in Venice. A system made of 8 sirens currently exists in Venice, that we report in Figure 2.

We have tested the solution on two matrices A, abstracting the Venice Map with different precision. The first one is a 78×51 grid where each side cell

El.	n	$\|P\|$	Preproc. T.	Constr. T.	Solution	Labeling	Labeling T.
6000	9	20	4'52"	3'20"	yes	leftmost	0"
6000	8	20	4'52"	3'20"	no	leftmost	1' 20"
6000	7	20	4'52"	3'20"	no	leftmost	$\geq$ 8'
6000	7	20	4'52"	3'20"	no	min	40"
1500	6	30	20"	23"	yes	ff	1' 20"
1500	6	30	20"	23"	yes	max	1' 15"
1500	6	30	20"	23"	yes	ffc	22"
1500	6	30	20"	23"	yes	min	2"
1500	6	30	20"	23"	yes	min bisect	18"
1500	8	40	34"	40"	yes	min	19"
1500	7	40	34"	40"	yes	min	55"
1500	6	40	34"	40"	yes	min	25"
1500	5	40	34"	40"	yes	min	1'

Table 1. Summary of benchmarks

corresponds approximately to 100 meter. The second is a 106×66 grid where each side cell corresponds approximately to 50 meters.[4]

The computational results are reported in Table 2. We have set the Sound pressure to 400 in the first case and to 500 in the second case. For 7 sirens the execution finds no solutions.

Grid	Sound dB	Sound proc.	Constr gen.	Pool	Min Dist.	labeling
78×51	60	2' 20"	37"	30	800	2"
78×51	60	3' 10"	55"	40	700	1"
106×66	60	5' 50"	3' 45"	20	700	10"

Table 2. Computational Results over the Venice data

The solutions proposed are depicted in Figures 3 and 4. A bitmap construction is generated from the map: different colors are associated to the mean density of the cell (pixel). White means 0, namely, water and black means 1. Black circles are the computed locations for working sirens. We can observe that there are some similarities with the current solution, but also several differences. For instance, a siren in the west part (*Stazione Marittima, Tronchetto*) is always computed, but no siren is present there now. Note also that, in the example with 7 sources in Figure 4, the least important source is excluded, still preserving the constraints imposed. Also note that increasing the dB level to 59, the disposition is modified in the north-east area. In this model, this is the highest level

[4] Data refer to slightly different cut & paste of the lagoon around the venice map. This explain why the size of the latter grid is not exactly the double of the size of the former.

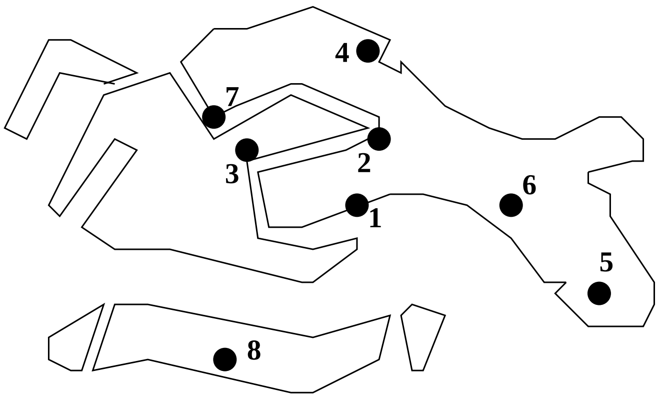


Fig. 2. The Existing Siren Allocation in Venice

reachable, corresponding to a frontier solution which implies a good disposition of sources.

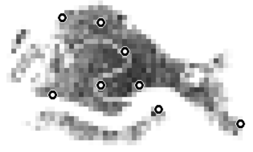
8 Conclusions

In this paper we have faced the problem of covering a physical area with acoustic signals emitted by different sources, presenting a declarative and effective solution in SICStus Prolog. We have tested the code over the high-tide sirens allocation problem in Venice. The code has an acceptable running time on the real-wold data. However, some more modeling work has yet to be done. First of all, it is necessary to refine the sound propagation function, using also other parameters, such as building reflections, that have been not yet considered. We are currently trying to find the function implemented by SoundPLAN. Another possible refinement relates to the signal threshold t: assuming a constant threshold on the grid is not completely realistic, since the background noise may vary largely from place to place. Also, we may want to provide stronger signals in certain locations, e.g. those grid cells where the population density is higher.

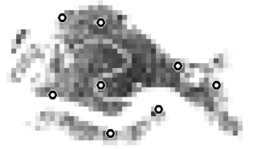
Acknowledgments

We thank the anonymous referees for their useful and constructive comments. The student R. Benazzato helped producing the data grid from geographical data. D. Piccolo provided guidance to the use of SoundPlan.

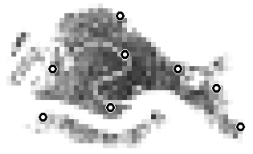
The work is partially supported by the *Consorzio Venezia Ricerche (CVR).* We thank D. Bovo (CVR) and P. Canestrelli (*Centro Previsioni Maree*) for their continuing support and advice. A. Dovier is partially supported by the MIUR project: *Automatic Aggregate —and number— Reasoning for Computing.*



Pixel size: 100m, Pool of 40 Sirens, Min. Distance: 600m , Level: 61 dB



Pixel size: 100m, Pool of 30 Sirens, Min. Distance: 800m, Level: 61 dB

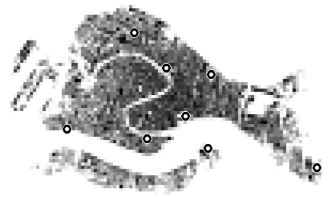


Pixel size: 100m, Pool of 30 Sirens, Min. Distance: 800m, Level: 62 dB

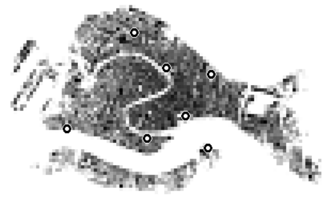
Fig. 3. Minimization Results. Greyscale: mean-density (white = water)

References

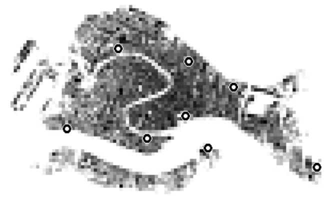
1. G. Festa, G. Sardu, and R. Felici. A decision Support System for the Venice Lagoon. In A. Herold (Ed), The Handbook of Parallel Constraint Logic Programming Applications, Chapter 5, ECRC, 1995 (available from `http://www-icparc.doc.ic.ac.uk/eclipse/reports`).
2. T. W. Frühwirth. Theory and Practice of Constraint Handling Rules. *Journal of Logic Programming* 37(1-3): 95-138 (1998).
3. T. W. Frühwirth and P. Brisset. Placing Base Stations in Wireless Indoor Communication Networks. *IEEE Intelligent Systems* 15(1):49–53 (2000).
4. R. H. Güting and M. Schneider. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4:243-286, 1995.



Pixel: 50m, Pool of 20 Sirens, Min. Distance: 600m, Level: 57 dB



Pixel: 50m, Pool of 20 Sirens, Min. Distance: 600m, Level: 57 dB, 7 sources



Pixel: 50m, Pool of 20 Sirens, Min. Distance: 600m, Level: 59 dB

Fig. 4. Minimization Results Greyscale: mean-density (white = water)

5. T. Hadzilacos and N. Tryfona. Logical Data Modelling for Geographical Applications. *International Journal of Geographical Information Systems*, 10(2):179-203, 1996.
6. K. Marriott and P. J. Stuckey. *Programming with Constraints.* The MIT Press, 1998.
7. P. M. Morse and K. U. Ingard, *Theoretical Acoustics*, Princeton University Press, Princeton, New Jersey, 1968.
8. J. G. Roederer, *The Physics and Psychophysics of Music.* Springer Verlag, New York, 1995.
9. G. Sardu, G. Serrecchia, E. G. Omodeo, L.-L. Li, M. Reeve, K. Schuerman, A. Véron. Safeguarding the Venice Lagoon: An Application of a knowledge-based DSS*. In D. Saccà ed., Proc. of GULP'93, Eight Conference on Declarative Programming, Gizzeria (IT), 1993.
10. M. Scholl and A. Voisard. Thematic Map Modeling. In *LNCS 409: Proc. of the Int. Symp. on the Design and Implementation of Large Spatial Databases*, pages 167–190, 1989.
11. SoundPLAN LLC, Braunstein + Berndt GmbH. SoudPLAN Home Page. `http://www.soundplan.com/`.
12. Swedish Institute for Computer Science. Sicstus Prolog Home Page. `http://www.sics.se/sicstus/`.