Optimal Placement of Base Stations in Wireless Indoor Communication Networks*

Thom Frühwirth
Ludwig-Maximilians-Universität München (LMU)
Oettingenstrasse 67, D-80538 Munich, Germany
fruehwir@informatik.uni-muenchen.de
http://www.pst.informatik.uni-muenchen.de/~fruehwir/

Pascal Brisset
Ecole Nationale de l'Aviation Civile (ENAC)
7 Av. Edouard Belin, BP 4005, F-31055 Toulouse Cedex, France
Pascal.Brisset@recherche.enac.fr

May 27, 1998

Abstract

Planning of wireless communication networks is about installing S small, local radio transmitters (base stations) to provide wireless devices with strong enough signals, wherever they are. POPULAR is an advanced industrial prototype that allows to compute the minimal number of base stations and their location given a blue-print of the installation site and information about the materials used for walls and ceilings. It does so by simulating the propagation of radio-waves using ray tracing and by subsequent optimization of the number of base stations needed to cover the whole building. Taking advantage of state-of-the-art techniques for programmable application-oriented constraint solving, POPULAR was among the first practical tools that could optimally plan wireless communication networks. In this article, we introduce the basics of this real-life application and show how easy it is to implement the necessary constraints to solve the problem.

Keywords: Constraint-based optimisation, picocellular radio, path loss model, simulation of radio cells, antenna placement, constraint programming.

*Work was done while the authors were at ECRC, Munich, Germany

1 Introduction

Mobile communication has become literally ubiquitous these days. According to a press release of British Telecommunications Plc and MCI Communications Corp of September 1996, there are 600 Mill. phones worldwide, 60 Mill. of them mobile (compare this figure to the world's 50 Mill. fax numbers and 40 Mill. email addresses).

More and more, mobile communications also comes to company sites by means of local, typically indoor wireless communication networks. No cabling is required and the employees can be reached at any time at any place. However, planning of wireless networks is quite different from planning traditional wire-based networks.

The specifics of radio wave propagation at the installation site have to be taken into account. Current systems are cellular in that a base station (i.e. senders, transmitters) controls the links to the tranceivers. A (radio) cell is the space that is covered by a single base station. The size of a cell is usually in the tens of meters. For buildings, multi-cellular systems are required, because walls and floors absorb part of the radio signal.

Today, the number and positioning of base stations is estimated by an experienced sales person. To help the sales person, Siemens has compiled a set of guidelines based on typical scenarios. However, a scenario may not always apply and the approach does not work well when it comes to position the base stations.

Computer-aided planning promises to ease some of the difficulties encountered. The idea is: Given a blue-print of the building or company site, together with information about the materials used for walls and floors, compute the minimal number of base stations and their location by simulation and subsequent optimization.

An advanced prototype, POPULAR (Planning of Picocellular Radio), was developed in collaboration with industry and research institutions in Germany: The Siemens Research and Development Department (ZFE), the Siemens Personal Networks Department (PN), the European Computer-Industry Research Center (ECRC) and the Institute of Communication Networks at the Aachen University of Technology.

In a few months of 1995, the authors implemented a protoype version while working at ECRC in the constraint logic programming language ECL^{i}PS^e [WNS97]. The language includes a library for Constraint Handling Rules (CHR) [Fru98], which are a high-level language extension to implement arbitrary constraint systems. The CHR library was essential for a rapid, flexible and efficient implementation of the geometric constraints that appear in this optimization problem. The prototype is part of the demo suite of ECL^{i}PS^e 3.4. Based on this prototype, J.-R. Molwitz, a student from the University of Aachen, implemented the tool POPULAR within one man-year while at Siemens.

In the next section, we introduce the physical model of radio wave propagation used in our application. Then we describe the implementation giving details

about the constraint solving involved in the optimization. Finally, we conclude the article with an evaluation of our tool. This article is a revised version of [FrBr97] and is a companion paper to [FMB96] which does not describe the implementation, but concentrates on the model used and its features (here section 2). For a complete introduction into constraint programming see [MaSt98], for a survey on its applications [Wal96].

2 Modeling Picocellular Radio

Radio wave propagation suffers mainly from the following effects:

- attenuation (weakening) of the signal due to the distance,
- shadowing (absorption) through obstacles,
- multipath propagation due to reflection and diffraction.

Figure 1 shows an example of the resulting path loss over distance on a logarithmic scale. At $6m$ and $9m$ walls are weakening the signal.

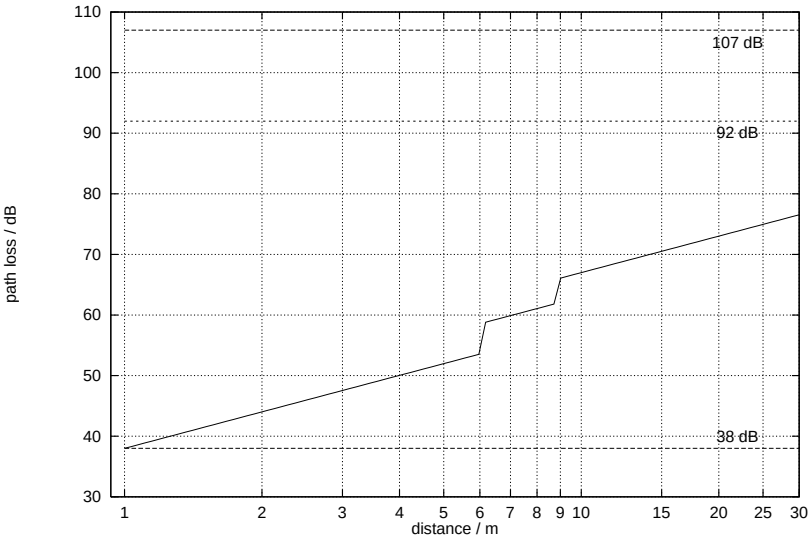

Figure 1: Path loss with additional attenuation at $6m$ and $9m$ due to walls

The COST[1] Subgroup 'Propagation Models' proposed the following path loss model [COS90]:

$$L_P = L_{1m} + 10\,n\,\log_{10} d + \sum_i k_i\,F_i + \sum_j p_j\,W_j \qquad (1)$$

[1]European Cooperation in the field Of Scientific and Technical research

where L_P : total path loss in dB,
L_{1m}: path loss in 1 m distance from base station,
n : propagation factor,
d : distance between base station and receiver,
k_i : number of floors of kind i in the propagation path,
F_i : attenuation factor of one floor of kind i,
p_j : number of walls of kind j in the propagation path,
W_j : attenuation factor of one wall of kind j.

The model is based on the power balance of wireless transmission. It combines a distance dependent term with correction factors for extra path loss due to floors and walls of the building in the propagation path.

This path loss model does not take reflection and hence multipath effects into account. Even with sufficient receiver sensitivity a radio link could fail due to fading and too many bit errors that result from it. Thus a *fading reserve* (fade margin) is introduced. We also extended the model to take the directional effect of an antenna into account, since antennas do not beam with the same energy in every direction.

3 Planning in POPULAR

Given a blue-print of the building and information about the materials used for walls and ceilings, POPULAR computes the minimal number of base stations and their location by simulating the propagation of radio-waves using ray tracing and subsequent optimization of the number of base stations needed to cover the whole building.

To get a description of a building one scans in a blue-print. From the scanned image, the walls and ceilings are redrawn. Each wall and ceiling gets its own attenuation factor.

3.1 Simulation of Radio Cells by Ray-Tracing

The characteristics of the building are computed using of test points. Each test point represents a possible receiver position. The test points are placed on a 3-dimensional grid inside the volume that should be covered. At each floor of the building, there is one such layer of test points (fig. 2). For each test point the space where a base station can be put to cover the test point, the "radio-cell", is calculated. If the test grid is sufficiently small (several per squaremeter), we can expect that if two neighbouring test points are covered, the space inbetween - hence the whole building - can also be covered.

Ray tracing simulates the propagation of radio waves through the walls and ceilings of the building. To get to the point of minimal sensitivity (i.e. maximal

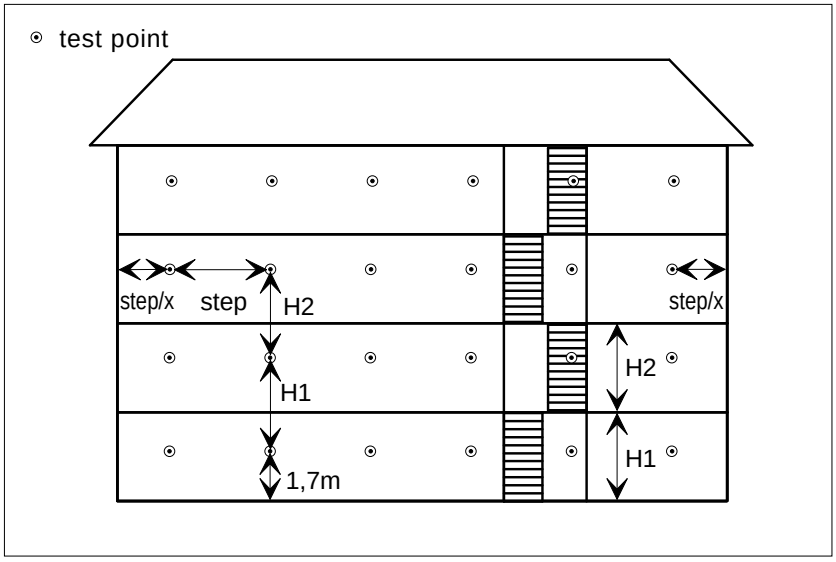

Figure 2: Grid of test points in a building

permissible path loss), each path must be followed through the whole building (fig. 3). The values of antenna attenuation in the direction of the path, the path loss due to the distance and the insertion losses due to intersections of the path with walls and floors are added up to the maximal permissible path loss. The resulting end points are used to describe the hull of the radio cell. We use binary (dichotomy) search to find the threshold location for each ray to speed up the simulation. For each test point, 128 rays are computed in POPULAR.

Note that the radio-cell will usually be a rather odd-shaped object, since the coverage is not a smooth or even differentiable function. The received power at a single point may exhibit discontinuities because of tiny changes in the base station location - for example, a move around the corner can cause an entirely different pattern of transmitted rays. This is the reason, why the path loss formula cannot be directly expressed as a constraint and why ray-tracing has to be used. Also note that the coverage that can be computed is noisy, i.e limited in accuracy, otherwise we would have to calculate the effect of e.g. every piece of furniture.

In practice, the base stations are installed at the same height at each floor, on the ceiling or on the walls. This means that on each floor, the possible space of locations for base stations is on a single plane. This plane is intersected with the radio cells, reducing them from a polyeder to a series of connected polygons (one for each floor) (fig. 4).

3.2 Constraint-Based Optimization

For each of the resulting polygons a constraint is set up that there must be (at least) one location of a base station (geometrically speaking, a point) somewhere

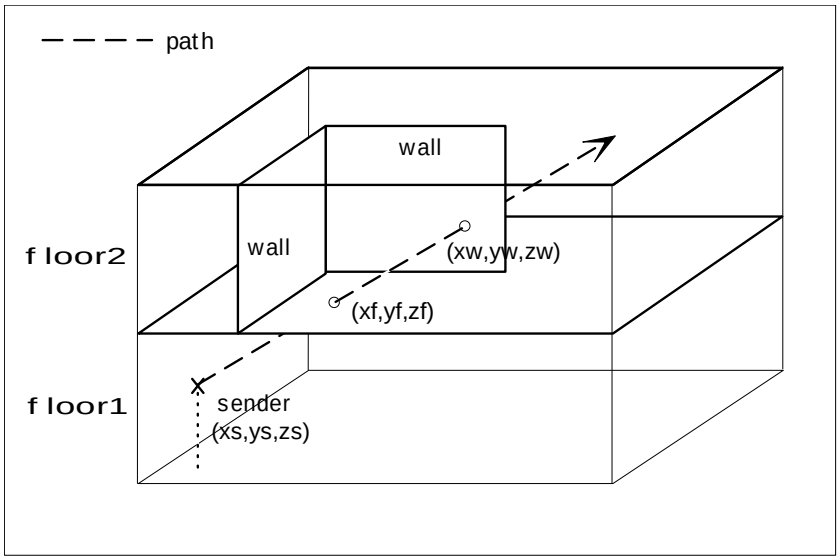

Figure 3: Path from base station intersecting a ceiling and a wall

in that space. Then, we try to find locations that are in as many polygons at the same time as possible. This means that a base station at one of these locations will cover several test points at once. Thus the possible locations are constrained to be in the intersections of the polygons covered. In this way, a first solution is computed. Next, to minimize the number of base stations, we use a branch-and-bound method. It consists in repeatedly searching for a solution with a smaller number of base stations until the minimal number is found.

In a first attempt restricted to two dimensions, we approximated a polygon by a single rectangle. The 2-D coordinates are of the form `X#Y`, rectangles are orthogonal to the coordinate system and are represented by a pair, composed of their left lower and right upper corner coordinates. For each polygon, simply a constraint `inside(Sender, Rectangle)` is imposed, where `Sender` refers to a point that must be inside the `Rectangle`.

The constraint solver was implemented using Constraint Handling Rules (CHR) [Fru98], which are our proposal to allow for more flexibility and application-oriented customization of constraint systems. CHR are a declarative language extension especially designed for writing user-defined constraints. CHR are essentially a committed-choice language consisting of multi-headed guarded rules that rewrite constraints into simpler ones until they are solved. The CHR code for the `inside` constraint is simply:

```
% inside(Sender, LeftLowerCorner - RightUpperCorner)

not_empty @ inside(S,A#B-C#D) ==> A<C,B<D.

intersect @ inside(S,A1#B1-C1#D1),inside(S,A2#B2-C2#D2) <=>
```

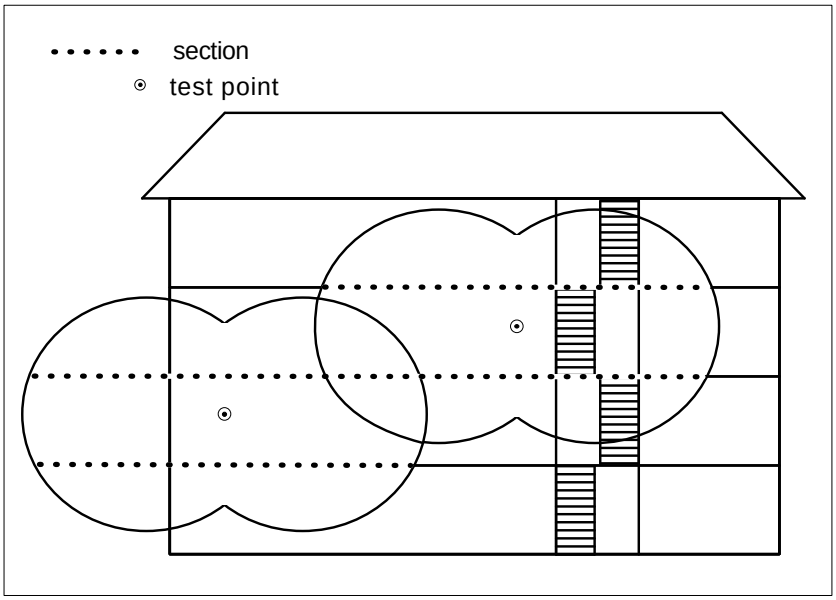

Figure 4: Typical radio coverage areas in a building

```
        A is max(A1,A2), B is max(B1,B2),
        C is min(C1,C2), D is min(D1,D2),
        inside(S,A#B-C#D).
```

The first rule (named `not_empty`) says that the constraint `inside(S,A#B-C#D)` is only valid if also the condition `A<C,B<D` is fulfilled, so that the rectangle has a non-empty area. The `intersect` rule says that if a base stations location `S` is constrained by two `inside` constraints to be in two rectangles at once, we can replace these two constraints by a single `inside` constraint whose rectangle is computed as the intersection of the two initial rectangles.

To compute a solution, after we have set up all the `inside` constraints, we try to equate as many base stations as possible. (This is called the "labeling phase" of the constraint computation). A simple way to do this is the following recursive procedure:

```
equate_senders([]) <=>
equate_senders([S|L]) <=>
        (member(S,L) or true), % equate S with another sender or not
        equate_senders(L).
```

For each base station `S`, the piece of code `(member(S,L) or true)` nondeterministically equates `S` with one of the remaining base stations in the lest `L` using `member` or does not do so (`true`). Equating base stations causes the `intersect` rule to fire with the constraints associated with the base stations. As a result of this labeling procedure, a base stations location will be constrained more and

more and thus the `intersect` rule will be applied again and again until the rectangle becomes very small and finally empty. Then the `not_empty` rule applies, causes failure and so initiates chronological backtracking that will lead to another choice.

A good heuristic for equating base stations takes advantage of the geometric nature of the problem and equates base stations from polygons associated with nearby test points first. The result of the optimization is a few base stations constrained to be located inside some small rectangle.

It took just 10 minutes to extend this solver so that it works with union of rectangles, that can describe the polygon more accurately, actually to any desired degree of precision. This corresponds to a disjunctive constraint of the form `inside(S,R1) or inside(S,R2) or ... or inside(S,Rn)` which is more compactly implemented as `inside(S,[R1,R2,...,Rn])`. "Geom" stands for "geometrical object".

```
% inside(Sender, GeomList)

intersect @ inside(S, L1), inside(S, L2) <=>
        intersect_geoms(L1, L2, L3),
        L3 = [_|_],         % at least one geom left
        inside(S, L3).

intersect_geoms(L1, L2, L3) <=>
        setof(Geom, intersect_geom(L1,L2,Geom), L3).

intersect_geom(L1, L2, rect(A#B,C#D)) <=>
        member(rect(A1#B1,C1#D1), L1),
        member(rect(A2#B2,C2#D2), L2),
        A is max(A1,A2), B is max(B1,B2),
        C is min(C1,C2), D is min(D1,D2),
        A<C, B<D. % not empty
```

The above solver can be adapted quickly to work with other geometric objects than rectangles by changing the definition of `intersect_geom/3`. Also, the lifting to 3 dimensions just amounted to adding a third coordinate and code analogous to the one for the other dimensions. The simplicity of the solver does not mean primitiveness or triviality, it rather illustrates the power of CHR, since it would be quite hard to implement the functionality in a hard-wired black-box solver:

Already in the initial non-disjunctive case there are problems: Finite domains are in principle applicable, however coordinates would have to be rounded to integers. Also, we found that for our application the built-in finite domain solver of ECL^{i}PS^e was slightly slower than the CHR implementation. Using linear polynomial constraints would be an overkill and thus inefficient, too. Interval arithmetic

[Ben95] can express the required constraints more adequately - even when we move from rectangles to other geometric objects that are described by non-linear equations. However, the actually used disjunctive geometric constraints would require recasting using auxiliary variables, which is expensive, error-prone and limits the amount of propagation.

4 Evaluation and Conclusions

Taking advantage of state-of-the-art techniques for programmable application-oriented constraint solving, POPULAR was among the first practical tools that could optimally plan wireless communication networks. While we worked on POPULAR, without knowing from each other, the WiSE tool [FGK*95] was developed with exactly the same functionality. WiSE is written in about 7500 lines of C++. For optimization WiSE uses an adaptation of the Nelder-Mead direct search method that optimizes the percentage of the building covered. WiSE has been patented and is in commercial use by Lucent Technologies to plan their DEFINITY Wireless Business System-PWT since 1997. Another approach [StEp96] uses the Nelder-Mead method for continuous space and Hopfield Neural Networks for a modelling in discrete space. The authors shortly mention a tool called IWNDT which is programmed in C.

POPULAR is a state-of-the-art tool. For a typical office building, an optimal placement is found by POPULAR within a few minutes. This is impressive since everything (including ray tracing and a graphical user interface) was implemented in a CLP language. The CLP code is just about 4000 lines with more than half of it for graphics and user interface. The overall quality of the placements produced is comparable to that of a human expert. The precision of the placements is within 0.7 meters. It is influenced by the underlying path loss model with its the fading reserve, the number of rays used in the simulation and the approximation of radio cells by unions of rectangles.

While the simulation phase has linear complexity in the number of test points, the optimization phase has a theoretical exponential complexity. Our practical experience shows, however, that the actual complexity is much lower. Ray tracing is fast by limiting the number of rays to 128 and by using binary (dichotomy) search for the threshold of each ray. The labeling benefits from heuristics that take the geometric nature of the problem into account. The average run-time on a SUN SPARCStation 10 was almost linear in the number of walls and test points, with about $25ms$ per wall and per test point. In a big building, this number may reach several thousands, resulting in computation times of about a minute and placing up to 25 base stations.

The necessary constraints were expressed and implemented with ease in CHR. Simplicity, flexibility, efficiency and rapid prototyping were the advantages of using CHR: The application was extended from rectangles to unions of rectangles,

from 2-D to 3-D. Also, restricting allowed base stations locations to walls or near ceilings or to aisles for ease of installation and maintenance was just a matter of constraining the base station positions to the union of the allowed spaces.

References

[Ben95] F. Benhamou, Interval constraint logic programming, In A. Podelski, editor, Constraint Programming: Basics and Trends. LNCS 910, March 1995.

[COS90] COST 231 'Propagation Models' Subgroup, Building Penetration Losses, Report COST 231 TD (90) 116, Darmstadt, Germany, December 1990.

[FGK*95] S. J. Fortune, D. M. Gay, B. W. Kernighan et al., WiSE Design of Indoor Wireless Systems, IEEE Computational Science and Engineering, Vol. 2, No. 1, pp. 58-68, Spring, 1995.

[FMB96] T. Frühwirth, J.-R. Molwitz and P. Brisset, Planning Cordless Business Communication Systems, IEEE Expert Magazine, Special Track on Intelligent Telecommunications, February 1996.

[FrBr97] T. Frühwirth and P. Brisset, Optimal Planning of Digital Cordless Telecommunication Systems, Third International Conference on The Practical Application of Constraint Technology (PACT97), London, U.K., April 1997.

[Fru98] T. Frühwirth, Theory and Practice of Constraint Handling Rules, Special issue on constraint logic programming (K. Marriott and P. J. Stuckey, Eds.), Journal of Logic Programming, to appear 1998.

[MaSt98] K. Marriott and P. J. Stuckey, Programming with Constraints, MIT Press, USA, March 1998.

[WNS97] M. Wallace, St. Novello and J. Schimpf, ECLiPSe: A Platform for Constraint Logic Programming, Technical Report, IC-Parc, Imperial College, London, August 1997.

[StEp96] D. Stamatelos and A. Ephremides, Spectral Efficiency and Optimal Base Placement for Indoor Wireless Networks, IEEE Journal on Selected Areas in Communications, Vol. 14, No.4, May 1996.

[Wal96] M. Wallace, Survey: Practical Applications of Constraint Programming, Constraints Journal, Vol.1, No.1, 1996.

Authors

Thom Frühwirth is a research assistant at Ludwig-Maximilians-University in Munich, Germany, where he also received his habilitation. His research interests focus on constraint-based programming and constraint reasoning. From 1991-1996 he worked at the European Computer-Industry Research Centre (ECRC) in Munich, before that he was assistant at the Technical University of Vienna, Austria, where he received his MSc and PhD in computer science. During studies, he worked as a freelance software contractor and received a one year Fulbright grant.

Pascal Brisset received a PhD in computer science from the University of Rennes, France. His research interests focus on constraint programming for global optimisation. He is particularly interested in relating contraints reasoning and stochastic methods (i.e. genetic algorithms).