Building Design Optimization using Constraint Logic Programming

John S. Mashford[1], Robin M. Drogemuller[1], and Peter J. Stuckey[2]

[1] Commonwealth Scientific and Industrial Research Organisation
A.I. Laboratory, CMIT Division,
P.O. Box 56, Highett, Vic. 3190, Australia
{John.Mashford,Robin.Drogemuller}@csiro.au
[2] Department of Computer Science & Software Engineering,
University of Melbourne, 3010 Australia pjs@cs.mu.oz.au

Abstract. Formal or mathematical methods have been widely studied in architectural design. A particular problem in architectural design which is amenable to formal solution is described as follows. We are given a 2-dimensional building configuration in which the topology is fixed but the geometry, represented by wall lengths, is flexible. The problem is to minimize the total area of the building, which is proportional to the cost of the building, subject to the constraints that the room areas are within set limits, that length/width ratios for rooms are within set limits and that wall lengths are within set limits. The natural formulation is a non-linearly non-convex constrained problem with quadratic objective function. In this paper we examine the use of a real interval constraint solver for tackling such problems. We explore effective models and search strategies in order to, as quickly as possible, determine optimal solutions.

1 Introduction

Constraint Logic Programming (CLP) has been used in a wide variety of practical applications, from internet applications [2], the solution of the Warehouse Location Problem [13] to the solution of the problem of protein side-chain placement [12].

Constraint programming has been used in architectural design [6, 10, 11]. In [6] a knowledge-based assistant for sub-system layout is described which utilises constraint propagation. More generally, formal or mathematical methods have been widely used in architectural design [7, 9].

A particular problem in architectural design which is amenable to formal solution is described in Gero [5] and Radford and Gero [9] Chapter 4. The problem may be described as follows. There is specified a 2-dimensional building configuration in which the topology is fixed but the geometry, represented by wall lengths, is flexible. The problem is to minimize the total area of the building, which is proportional to the cost of the building, subject to the constraints that the room areas are within set limits, that length/width ratios for rooms are within set limits and that wall lengths are within set limits. In this paper we investigate using constraint logic programming to tackle this problem.

The problem is similar to the problem of floorplan optimisation in VLSI described in [16] and [15].

In the next section we introduce the problem, and give an analytical solution for the one room case, before giving the example of Radford and Gero [9]. In Section 3 we discuss how to use a real interval solver to tackle this problem, and in Section 4 we describe experiments. Finally in Section 5 we conclude.

2 Building Design Optimization Problems

We define the *building design optimization problem* as follows.

We are given a plan of a building, with rectangular rooms. Each room is constrained to have a minimum and a maximum length, width, and area. In some cases these minimums or maximums may be 0 or ∞ respectively, hence effectively there is no constraint. Some rooms may additionally have constraints on the minimum and maximum ratio of length to width. There are other ad-hoc constraints that we can add to represent for example: door placement and window placement.

The usual objective function is to minimize the area, although we shall also use a variation of this for the example problem of Radford and Gero.

2.1 The one room case

Consider the case of a building consisting of a single room with wall lengths w_1 and w_2, where we only specify minimum length, width and area. In this case the problem is convex and can be solved analytically. We consider the problem P defined by:

$$\begin{gathered}\min(w_1 w_2) \text{ such that}\\ w_1, w_2 > 0,\\ w_1 w_2 \geq A_1,\\ w_2/w_1 \in [\rho_1, \rho_2],\\ w_1 \geq \alpha,\\ w_2 \geq \beta,\end{gathered}$$

where $A_1, \rho_1, \rho_2, \alpha, \beta > 0$. Example constraints are illustrated in Figure 1.

This problem has a closed form solution defined as follows. Let

$$M = \max(\alpha\beta, \rho_1\alpha^2, \beta^2/\rho_2, A_1)$$

A minimal area solution is given where the maximal value M occurs.
(a) if $\alpha\beta = M$ then a minimal solution is $w_1 = \alpha \wedge w_2 = \beta$ else
(b) if $\rho_1\alpha^2 = M$ then a minimal solution is $w_1 = \alpha \wedge w_2 = \rho_1\alpha$ else
(c) if $\beta^2/\rho_2 = M$ then a minimal solution is $w_1 = \beta/\rho_2 \wedge w_2 = \beta$ else
(d) if $A_1 = M$. The final case is more complicated. If $A_1\rho_1 \leq \beta^2$ then the solution is
(d1) $w_1 = A_1/\beta \wedge w_2 = \beta$, otherwise it is
(d2) $w_1 = \sqrt{A_1/\rho_1} \wedge w_2 = \sqrt{A_1\rho_1}$.

It is a simple matter to check that the resulting valuations satisfy all the constraints. We illustrate the result for case (d), the remaining cases follow more simply.

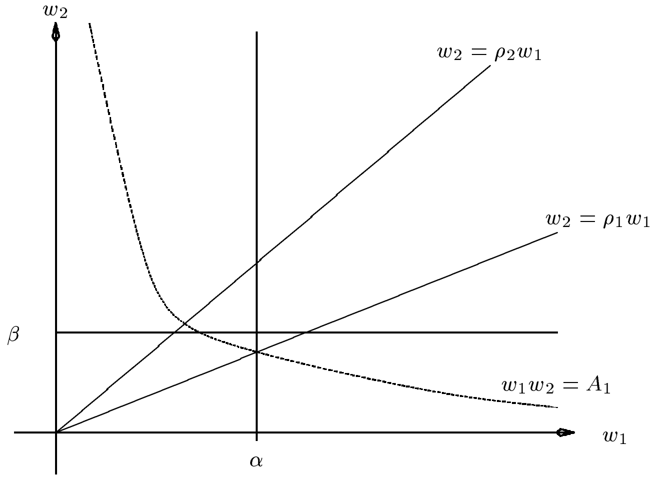

Fig. 1. Solution space for the one room problem.

Clearly $w_1 w_2 = M = A_1$.

In the case that $A_1\rho_1 \leq \beta^2$ we have solution (d1). For (d1) clearly $w_2 \geq \beta$, and since $A_1 \geq \alpha\beta$, $w_1 \geq \alpha$. Now $w_2/w_1 = \beta^2/A_1$. Since $A_1 \geq \beta^2/\rho_2$ we have that $w_2/w_1 \leq \rho_2$. Also $\beta^2/A_1 \geq \rho_1$ hence $w_2/w_1 \geq \rho_1$.

Otherwise $A_1\rho_1 \geq \beta^2$ and we have solution (d2). For (d2) $w_2/w_1 = \rho_1 \leq \rho_2$. Now $A_1 \geq \alpha^2\rho_1$ and hence $w_1 \geq \alpha$. Also $\sqrt{A_1\rho_1} \geq \beta$ and hence $w_2 \geq \beta$.

2.2 A more complex example problem

The following building design optimization problem was defined in Radford and Gero [9] Chapter 4.

Consider the house design depicted in Figure 2. The design is constrained to fit this picture, and each room must be rectangular. The aim is to minimize the area of the house, while satisfying certain constraints. Each dimension must respect certain limits: define the length of a room to be its horizontal dimension, and the width to be its vertical dimension. Similarly the area of each room must fall within certain bounds. Finally the ratio of length/width must not be too great for some rooms. The constraints for each room are shown in Table 1. For example the constraints on the living room implied by the table entry are

$$8 \leq w_1 \wedge w_1 \leq 20 \wedge 8 \leq w_{11} \wedge w_{11} \leq 20 \wedge 120 \leq w_1 w_{11} \wedge w_1 w_{11} \leq 300 \wedge w_1 \leq 1.5 w_{11}$$

Other constraints arise from the relationships of dimensions shown in the diagram. These are

$$\begin{aligned} w_1 + w_6 &= w_4 + w_7, \\ w_{11} + w_5 &= w_9 + w_{10}, \\ w_{11} &= w_2 + w_3. \end{aligned}$$

Finally Radford and Gero add some additional practical constraints arising from the fact that the doors to bed 2 and bed 3 must come from the hallway. They thus require that there is exactly enough space to put a doorway (3 units=feet) to each from the end of the hall. This is encoded as

$$w_9 = w_2 + 3,$$
$$w_{10} = w_5 + 3.$$

Let C_{rg} be the total set of constraints defined above.

The objective function is

$$\text{minimize } ((w_1 + w_6 + w_8)(w_{11} + w_5)) \text{ subject to } C_{rg}$$

In fact for Radford and Gero's example they used a more sophisticated objective function where the kitchen and bath are considered more expensive to build, and hence their area counts double to the objective function. Hence the final problem is

$$\text{minimize } ((w_1 + w_6 + w_8)(w_{11} + w_5) + w_4 w_5 + w_6 w_2) \text{ subject to } C_{rg}$$

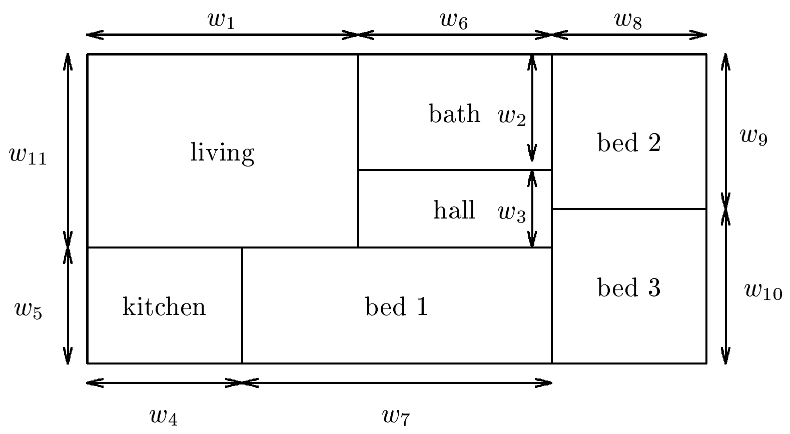

Fig. 2. An example building from Radford and Gero [9], with dimensions labelled.

The problem is a non-convex non-linear optimization problem with non-linear constraints. Hence standard optimization methods such as linear programming or quadratic programming are not directly applicable.

2.3 Radford and Gero's approach

The building design optimization problem can not be solved analytically for non-trivial cases. In such cases other solution techniques must be used. Radford and Gero [9] use sequential linear programming to solve the problem.

Room	Length		Width		Area		Ratio	
	Min	Max	Min	Max	Min	Max	Min	Max
living	8	20	8	20	120	300		1.5
kitchen	6	18	6	18	50	120		
bath	5.5	5.5	8.5	8.5				
hall	0	15	3.5	6	0	72		
bed 1	10	17	10	17	100	180		1.5
bed 2	9	20	9	20	100	180		1.5
bed 3	8	18	8	18	100	180		1.5

Table 1. Dimensional constraints on the rooms in Figure 2.

Starting from an initial feasible solution, each non-linear constraint and non-linear component of the objective function is replaced by a linearization representing the tangents to the actual non-linear constraint or objective function component. The resulting linear program is solved, and this gives a new feasible solution. The process is repeated until it appears to converge.

There are a number of problems with applying sequential linear programming to the building design optimization problem. Firstly, we require an initial feasible solution. This may be completely non-trivial to find, and without it the whole process cannot start. Secondly, numerical problems can prevent quick convergence to the optimal solution. And thirdly, depending on the problem there is no guarantee of convergence.

3 Solving the building design optimization problem using CLP

Constraint programming systems allow us to directly express linear and non-linear constraints and non-linear objective functions. The constraint solving technology applicable to such constraints is called interval arithmetic, and it is widely used in tackling non-linear optimization problems. Essentially, interval arithmetic constraint solving maintains for each variable a lower and upper bound which are floating point numbers, and guarantees never to eliminate a real solution.

Interval arithmetic constraint solving was suggested by John Cleary in [3], and first appeared as a constraint solver in the BNR Prolog system in 1987. There has been substantial development of techniques for solving constrained optimization problems using interval arithmetic. Benhamou *et al.* [1] introduced Box-consistency in the development of the `Newton` interval constraint solver. This line ended in the development of the `Numerica` language [14] for constrained optimization of non-linear constraints.

For our experiments we are making use of the `ria` real interval arithmetic library of the Eclipse [4] constraint programming language. This is a propagation based interval arithmetic library. Each variable maintains a lower and upper bound, and each constraint *propagates* to reduce as much as possible the bounds of its variables without removing any possible solutions.

The `ria` library does not directly provide any optimization facilities, rather it provides two approaches to improving interval domains. The `squash` algorithm reduces the domains of variables, by checking that the endpoints satisfy every possible constraint. The `locate` algorithm repeatedly splits domains in order to find a solution to a constraint problem to within some accuracy limit. Both predicates take as an argument the *precision* of solving which is treated as the least accurate of the absolute precision or relative precision. For all of our examples, the precision is a relative precision. For this class of problems we did not find any examples where the `squash` algorithm was useful, and hence we omit further consideration of it.

We used the inbuilt `bb_min` predicate of the Eclipse `branch_and_bound` library to search for solutions. This search predicate allows a number of search possibilities to tackle the minimization. The default is to combine the backtracking search (in this case defined by `locate`), while adding a constraint after a solution with objective value m is found to require subsequent solutions to have an objective function value less than $(1 - \epsilon)m$, where ϵ is the (relative) *accuracy limit*.

As we shall see the default strategy can be very poor on optimization problems where there are a large number of solutions, since we can slowly detect better and better solutions. Hence we also investigate the use of optimistic partitioning (or dichotomic search) to find optimal solutions.

In optimistic partitioning, we first determine the lower and upper bounds of the variable to be minimized after all constraints are applied. We then split this range in half, and look for a solution in the lower half. If that fails, we reset the lower bound to be the midpoint, and look for a solution in the upper half, if this also fails we know the best solution found so far is optimal. If either search succeeds we make the value of the variable in the solution the new upper bound, and continue the process.

```
opt-part(C,V,min,max,ε,best)
    if min > max return best
    mid := (min + max)/2
    θ := solv(C ∧ V ≥ min ∧ V ≤ mid)
    if (θ = false)
        min := mid
        θ := solv(C ∧ V ≥ mid ∧ V ≤ max)
        if (θ = false)
            return best
    max := θ(V)(1 − ε)
    return opt-part(C,V,min,max,ε,θ)
```

Fig. 3. Psuedo-code for optimistic partitioning.

Psuedo-code for optimistic partitioning is given in Figure 3. The initial call is with the problem constraints C, the variable to be minimized V, and lower and upper bounds min and max. The (relative) accuracy limit ϵ determines how close to the actual optimum we need to be before terminating. Initially the current best solution is $false$. The solv function returns a solution θ to the argument constraint or $false$. $\theta(V)$ is the value taken by V in the solution.

4 Experiments

All experiments were carried out on a Pentium IV 2.4 GHz with 512MB RAM running Red Hat Linux 2.4.20-8.

Solving the example given by Radford and Gero [9] is straightforward. An accuracy limit of $\epsilon = 0.01$ was used. Here the accuracy limit is that applied to the objective function.[3] The initial bounds on the objective function are determined as [656.7 .. 1488] and the first solution found is the best with optimal value 719.25. The search requires 0.02 seconds and 94 splits of variable domains and 5,053 constraint propagations to prove optimality. The resulting dimensions are shown in Table 2.

Our solution			Radford and Gero's solution		
Room	Length	Width	Room	Length	Width
living	10.5	14.5	living	10.5	14.5
kitchen	6.0	10.0	kitchen	5.95	9.95
bath	5.5	8.5	bath	5.5	8.5
hall	5.5	6.0	hall	5.5	6.0
bed 1	10.0	10.0	bed 1	10.05	9.95
bed 2	9.0	11.5	bed 2	8.95	11.5
bed 3	9.0	13.0	bed 3	8.95	12.95

Table 2. Optimal solution for the Radford and Gero problem, and their best "solution."

Increasing the accuracy to 0.0001 requires significantly more work, requiring 0.36 seconds, and 139 splits and 76,479 constraint propagations to prove optimality. While increasing accuracy to 0.000001 requires 32.8 seconds, 185 splits and 6,958,122 constraint propagations to prove optimality.

The default search strategy can have very slow convergence for high accuracies, hence the optimistic partitioning approach is preferable. Using the optimistic partitioning method then the results are faster but with a similar behaviour. For an accuracy of 0.01 again the first solution found is optimal. The time required is 0.04 seconds, using 94 splits and 5473 constraint propagations. If we increase the accuracy required to 0.0001 then the the time increases to 0.18 seconds, 139 splits, and 33032 constraint propagations. Increasing accuracy further to 0.000001 increases the time required to 2.28 seconds, 185 splits, and 465898 constraint propagations. Clearly optimistic partitioning scales better.

The straightforward modelling does not take into account that in the optimization process we will repeatedly resolve the problem after simply changing information about the lower and upper bounds on the objective function value. In order to improve the optimization problem we should make sure that the formulation can make maximum usage of bounds on the objective function.

To improve the model we changed the basic model as follows:

[3] To ensure the interval solver is applied accurately enough the precision of the `locate` is set to one tenth of this accuracy limit, while the arc consistency threshold is set to one thousandth of this accuracy limit for all experiments.

	Default Search			Optimistic Partitioning			Revised Model (Default)		
Accuracy	Time	Splits	Props	Time	Splits	Props	Time	Splits	Props
0.01	0.02	94	5053	0.04	94	5473	0.03	96	7310
0.0001	0.36	139	76479	0.18	139	33032	0.05	140	10900
0.000001	32.80	185	6958122	2.28	185	465898	0.07	188	17173

Table 3. Summary of results for the Radford and Gero solution.

- Each room was given an area variable and the area constraints were defined on this variable. E.g. $w_1 w_{11} \geq 120 \wedge w_1 w_{11} \leq 400$ is replaced by $A_{living} = w_1 w_{11} \wedge A_{living} \geq 120 \wedge A_{living} \leq 400$
- A new total length variable L is created and constrained to all possible sums that give the total length. $L = w_1 + w_6 + w_8 \wedge L = w_4 + w_7 + w_8$.
- A new total width variable W is created and constrained to all possible sums that give the total width. $W = w_{11} + w_5 \wedge W = w_2 + w_3 + w_5 \wedge W = w_9 + w_{10}$.
- The area is also defined using these new variables. $A = A_{living} + 2A_{kitchen} + 2A_{bath} + A_{hall} + A_{bed1} + A_{bed2} + A_{bed3} \wedge A = LW + A_{kitchen} + A_{bath}$.
- The objective function is changed to minimize A.

Using this improved model and accuracy limit of 0.01, the initial bounds on the objective function are now determined as [675.5 .. 1402]. This illustrates how the redundant constraints have improved propagation behaviour.

Using the default search strategy the first solution found in 0.03 seconds is again the best with optimal value 719.25. The search requires 96 splits of variable domains and 7310 constraint propagations. Increasing the accuracy to 0.0001 hardly makes the revised problem more difficult, requiring 0.05 seconds, and 140 splits and 10900 constraint propagations. Again increasing the accuracy to 0.000001 still requires only 0.07 seconds, 188 splits and 17173 constraint propagations.

The new model is significantly faster although it includes more constraints since it translates upper bounds information for optimization much more effectively. For this problem the default search is now so fast that there is no benefit in using optimistic partitioning. A summary of the results is shown in Table 3.

There are two points of interest. Although the method of Radford and Gero obtains an optimal value of 715.98 which is lower than what we obtain, their solution does not satisfy all the constraints. The length of the kitchen is 5.95, clearly violating the minimum of 6. The length of bed 2 is 8.95, clearly violating the minimum of 9. And the area of bed 1 is too small. This clearly illustrates that the sequential linear programming method was not very successful. It appears they have relaxed the constraints involved.

The second point is that really this is a very small problem. Given 11 decision variables, 2 of which are in reality fixed, and 5 equations, there are only really 4 independent decisions. Hence we get a very small search space, and `locate` quickly finds the optimal.

4.1 A less constrained larger example

We now introduce a slightly larger example, which requires more substantial search. The house plan shown in Figure 4 is constrained by the dimensional constraints shown in Table 4. In addition, the relationships of lengths from the diagram are

$$w_2 + w_1 = w_7 + w_8 + w_9 \wedge w_4 + w_5 + w_6 = w_{14} + w_{13} + w_{12} \wedge$$
$$w_{11} + w_{10} = w_9 \wedge w_{15} = w_5 + w_6$$

And a further constraint $w_{14} \geq w_4 + 1.5$ is added ensuring that the door to bed 2 is in hall-c. Given the discussion above we also added redundant constraints defining the areas of each room, the total length and width, and the area in terms of these.

$$L = w_3 + w_4 + w_5 + w_6 \wedge L = w_3 + w_{14} + w_{13} + w_{12} \wedge L = w_3 + w_4 + w_{15}$$
$$W = w_1 + w_2 \wedge W = w_7 + w_8 + w_9 \wedge W = w_7 + w_8 + w_{10} + w_{11}$$
$$A = \begin{array}{l} A_{entry} + A_{bed1} + A_{bed2} + A_{bed3} + A_{kitliv} + A_{halla} + A_{hallb} + A_{hallc} + \\ A_{wardrobe} + A_{ensuite} + A_{bath} \end{array}$$
$$A = LW$$

For this problem there are effectively 11 independent variables, and a much more weakly constrained design, making the optimization search much more difficult.

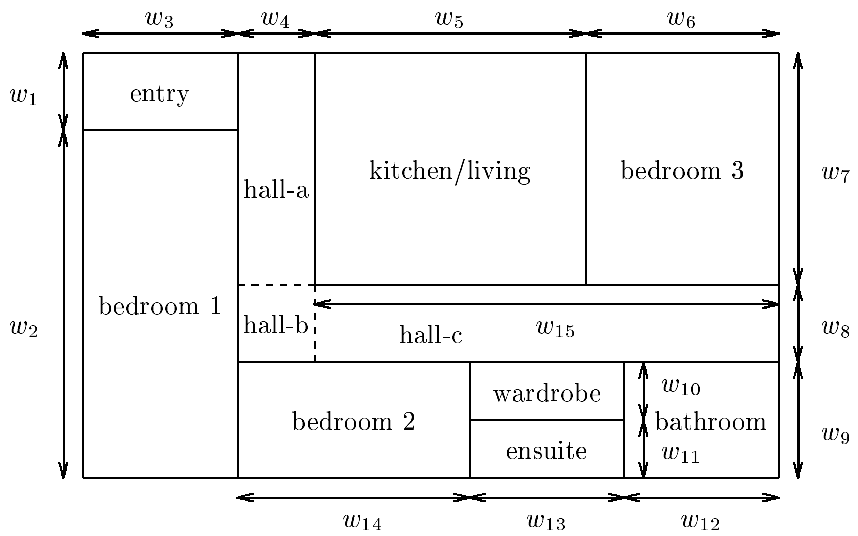

Fig. 4. A possible house design

On this problem the default search with an accuracy limit of 0.01 finds 8 solutions on the way to finding the best solution with an area of 85.3. It requires 1.2 seconds, 1523 splits and 283825 constraint propagations. Default search with an accuracy limit of 0.0001 finds 158 solutions and a best area of 84.8 and requires 67.6 seconds, 29274 splits and 16244374 constraint propagations.

Room	Length		Width		Area		Ratio	
	Min	Max	Min	Max	Min	Max	Min	Max
entry	1.5		1.5		12.8	19.2	1.2	1.8
bed 1	1.5		1.5					
bed 2	1.5		1.5		9.6	14.4		
bed 3	1.5		1.5		9.6	14.4		
kit/liv	1.5		1.5		12.8	19.2		
hall-a	1.5		1.5					
hall-b	1.5		1.5					
hall-c	1.5		1.5					
wardrobe	1.5		1.5				1.6	2.4
ensuite	1.5		1.5				1.6	2.4
bath	1.5		1.5		6.4	9.6	0.8	1.2

Table 4. Dimensional constraints on the rooms in Figure 4.

In contrast, optimistic partitioning with the accuracy limit 0.01 finds 2 solutions on its way to finding the best at 85.3, requiring 0.82 seconds, 1235 splits, and 191748 constraint propagations. Increasing the accuracy to 0.0001 resulted in 5 solutions being found for the best area of 84.8, in 4.48 seconds, 2260 splits and 1035617 constraint propagations. Hence in this case the benefit of optimistic partitioning is much clearer. A summary of the results is shown in Table 5

The best solution found is $A = 84.75$, $w_1 = 4.80$, $w_2 = 3.15$, $w_3 = 2.67$, $w_4 = 1.50$, $w_5 = 3.71$, $w_6 = 2.79$, $w_7 = 3.45$, $w_8 = 1.50$, $w_9 = 3.00$, $w_{10} = 1.50$, $w_{11} = 1.50$, $w_{12} = 2.40$, $w_{13} = 2.40$, $w_{14} = 3.20$.

	Default Search				Optimistic Partitioning			
Accuracy	Obj	Time	Splits	Props	Obj.	Time	Splits	Props
0.01	85.3	1.2	1523	283825	85.3	0.82	1235	191748
0.0001	84.8	67.6	29274	16244374	84.8	4.48	2260	1035617

Table 5. Summary of results for the less constrained larger problem.

4.2 A real world example

We now describe a harder example of building design problem. This is based on a real brief for an architect from the client for a school building. The building layout together with dimension labels is shown in Figure 5. The constraints for each room are given in Table 6

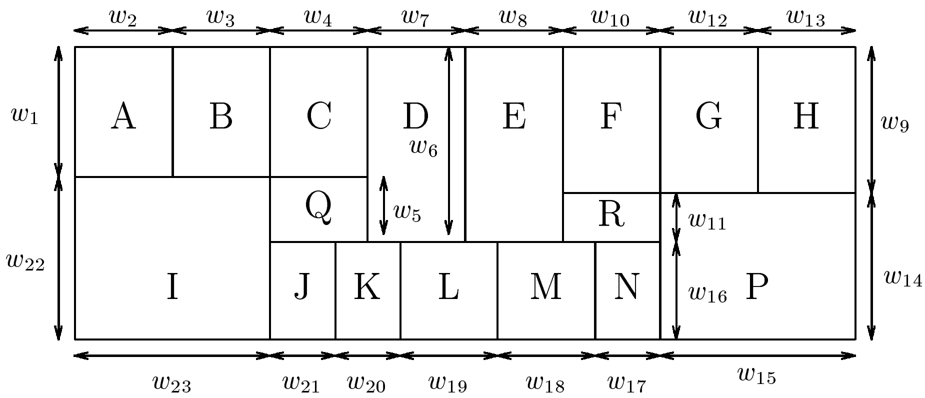

Fig. 5. A real-world building design optimization problem with dimensions labelled.

Room	Length		Width		Area		Ratio	
	Min	Max	Min	Max	Min	Max	Min	Max
A–H	5.0		1.0		52.0	57.2		
I,P	5.0		1.0		102.0	112.2		
J,K,N	4.0		1.0		28.5	31.5		
L,M	4.0		1.0		34.0	37.4		
Q,R	1.0		1.1					

Table 6. Dimensional constraints on the rooms in Figure 5.

The constraints that are required to ensure that the rooms are rectangular and the building is rectangular are as follows:

$$\begin{aligned}
w_{23} &= w_2 + w_3, \\
w_{22} &= w_5 + w_{16}, \\
w_6 &= w_1 + w_5, \\
w_1 + w_{22} &= w_6 + w_{16}, \\
w_6 &= w_9 + w_{11}, \\
w_9 + w_{14} &= w_1 + w_{22}, \\
w_{15} &= w_{12} + w_{13}, \\
w_{21} + w_{20} + w_{19} + w_{18} + w_{17} &= w_4 + w_7 + w_8 + w_{10}.
\end{aligned}$$

As usual we add variables representing the areas of each room, and the total length, width and area, and the redundant constraints defining them. The objective function to be minimized is simply the area. We only consider using the optimistic partitioning search for this example.

The initial bounds on the area arising from the constraints are that the area is in the range [784.5 .. 1989.2]. Using an accuracy limit of 0.1, searching for a first solution requires 1.64 seconds and 404 splits and 418223 constraint propagations. It finds a first solution with area 815.8, which is optimal (within the accuracy limit). Decreasing the accuracy limit to 0.05 finds the best area 815.8 in 1.82 seconds, 416 splits and 457581 constraint propagations, and decreasing it further to 0.025 finds the best area 792.9 in 18.0 seconds, 8103 splits and 4633310 constraint propagations. Decreasing the accuracy limit further caused the search to take over four hours without finding a best solution.

Although it is undocumented in the Eclipse system, the precision used by `locate` is not guaranteed to be better than the worst of the absolute and relative accuracy. It appears however that in our experiments the precision arrived at is the absolute precision rather than the relative precision which is greater.

So treating the precision as an *absolute* accuracy we can obtain more precise solutions. We modified the search to find a solution with absolute accuracy limit of 1.[4] This search finds a solution with area 815.8, then 791.8 and finally a best solution with area 790.8. It requires 33.5 seconds, 20216 splits and 8621161 constraint propagations. Decreasing the accuracy limit to 0.5 finds a best solution at 791.2 in 10 minutes with 299304 splits and 155961676 constraint propagations. With accuracy limit 0.25 the best solution is 791.9 in almost 19 minutes with 444647 splits and 291013424 constraint propagations.

There are a few points worth discussing. The first point is that the low precisions used here (1.0, 0.5 and 0.25) lead to solutions which are highly approximate. We can take the resulting bounds from the best solution and refine this to locate a solution with much better precision (say 0.0001) but in doing so we obtain a much higher area (794.5, 794.4 and 792.95 respectively). In contrast, the solutions found using the relative accuracy are much more reliable, we can refine the precision on the solution with area 792.9 to 0.00001 and still obtain the same area. In that sense the best *accurate* solution is 792.9 found using relative accuracy.

Another point is that each of these (accurate) solutions is highly variant, for example in the solutions with area 794.5 and 794.4 there are no corresponding lengths or widths that take a value within 13% of each other. This shows the inherent difficulty in solving such non-convex problems.

	Optimistic Partitioning				
Relative Accuracy	Obj	Time	Splits	Props	
0.1	815.8	1.64	404	418223	
0.05	815.8	1.82	416	457581	
0.025	792.9	18.00	8103	4633310	
Absolute Accuracy	Obj	Time	Splits	Props	Refined Obj.
1.0	790.8	33.5	20216	8621161	794.50
0.5	791.2	≈600	299304	155961676	794.40
0.25	791.9	≈1140	444647	291013424	792.95

Table 7. Summary of results for the real world problem.

[4] We maintain that the precision is one tenth of accuracy limit, and the propagation threshold is one thousandth.

5 Conclusions

The building design optimization problems are challenging. Even relatively small examples prove difficult to find optimal solutions. There are considerably more constraints that we could take into account for these problems, such as space for windows, space for doors, different costs for different room spaces (as in Radford and Gero's example).

The very slow convergence of the problem for higher accuracies suggests that a more heuristic approach, such as the perturbation search, exemplified in [8] Section 7.8.2 where we increase the accuracy while restricting the possible values of the variables to be close to the optimal solution found at a lower accuracy, might be beneficial. Although, because of the non-convex nature of the problem, this will certainly lead to only a local optimum being discovered.

It would be worth investigating whether sequential quadratic programming could tackle the problems better, but it seems likely that the same problem that makes sequential linear programming fail will also occur, namely the linear approximation of quadratic constraints in a non-convex system.

6 Further Work

There are two main threads of further work for this project. Firstly, there needs to be some automation in the generation of the constraints to the optimization problem. If the building is designed in CAD software, it is feasible to extract the geometry and topology of the plan layout. This would make this system useable by architects on real projects. The second thread will be to extend the work to multi-storey buildings where the external boundaries of the building need to be consistent for several floors and some areas, such as lift shafts, will need to be constrained to be continuous up several stories.

7 Acknowledgements

The authors thank I.R. Aitchison for very helpful discussions.

References

1. Benhamou, F., McAllester, D., and Van Hentenryck, P., CLP(Intervals) Revisited, *Proceedings of the International Symposium on Logic Programming (ILPS-94)*, pp. 124–138, 1994.
2. Bressan, S. and Goh, C., Semantic integration of disparate information sources over the internet using constraint propagation, *Workshop on Constraint Reasoning on the Internet, Third International Conference on Principles and Practice of Constraint Programming (CP97)*, 1997.
3. Cleary, J.C., Logical Arithmetic, *Future Computing Systems*, 2(2): 125–149, 1987.
4. ECL^{i}PS^e constraint programming system. `http://www.icparc.ic.ac.uk/eclipse/`
5. Gero, J.S., Note on "Synthesis and optimization of small rectangular floor plans" of Mitchell, Steadman and Liggett, *Environment and Planning B*, Vol. 4, 1977, pp. 81-88.

6. Gross, M., Knowledge based support for subsystem layout in architectural design, *Applications of Artificial Intelligence in Engineering V, Proceedings of the fifth international conference*, Comput. Mech. Publications, 1990, pp. 57-70, Vol. 1, Southhampton, UK.
7. March, L. and Steadman, P., *The Geometry of Environment*, 1971, RIBA Publications.
8. Marriott, K. and Stuckey, P.J., *Programming with Constraints*, 1998, MIT Press.
9. Radford, A.D. and Gero, J.S., *Design by Optimization in Architecture, Building and Construction*, 1988, Van Nostrand, p. 85.
10. Sapossnek, M., Research on constraint-based design systems, *Proc. 4th Intl. Conf. Applications of AI in Engineering*, Southampton, UK, Computational Mechanics Press, 1989.
11. Serrano, D. and Gossard, D., Constraint management in conceptual design, in *Knowledge-Based Expert Systems in Engineering: Planning and Design*, ed. Adey, R.A. and Sriram, D., pp. 211-224, Southampton, UK, Computational Mechanics Press, 1987.
12. Swain, M.T. and Kemp, G.J.L., A CLP Approach to the Protein Side-Chain Placement Problem, *Principles and Practice of Constraint Programming - CP2001*, Paphos, Cyprus, pp. 479-493, 2001, Springer.
13. van Hentenryck, P., *Constraint Satisfaction in Logic Programming*, 1989, MIT Press.
14. van Hentenryck, P., *Numerica: A Modeling Language for Global Optimization*, 1997, MIT Press.
15. Vijayan, G. and Tsay, R.-S., A new method for floor planning using topological constraint reduction, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 12, 1991, pp. 1494-1501.
16. Wang, T.-C. and Wong, D.F., Optimal floorplan area optimization, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 11, No. 8, 1992, pp. 992-1002.