EPJ manuscript No.
(will be inserted by the editor)

Constraint Programming Project Report

Mohamed Ashry Hassan Mahmoud(40-18561)[1] and Abdelrahman Mahmoud Ismail Elbeltagy(40-17544)[2]

German University in Cairo

12/1/2021

Abstract. Residential Buildings Layouts Generator Report

1 Introduction

This is the report of the CP project. It is discussing the CSP model and the implementing predicates of the project.

2 CSP Model

The CSP model consists of the ¡Variables,Domains,Constraints¿ to represent the problem.

3 Variables and Inputs

3.1 The Variables

The floor consists of rectangles(rects) where each rectangle represents a room in the apartment. Each rect is a predicate of 4 Variables (X,W,Y,H), this approach is used for the disjoint2 predicate.
The rooms are represented as predicates which take the names of the rooms.

- bedroom
- kitchen
- dining_room
- master_bathroom
- minor_bathroom
- living_room
- main_hallway
- hallway
- elevator
- stairs
- dressing_room
- sun_room

Where main_hallway is the hallway that connects the apartments with each other.
So, these are the low-level main variables of the model. The the high-level variables are just arrays of the these variables.

3.2 The Inputs

The inputs are represented as the Knowledge Base as in AI techniques. They are represented as facts in the Prolog file.

- Width, Length: floor_dimensions(Width,Length).
- Opensides + Landscapes: open_sides(North,East,South,West).
- Optional global view constrains: optional_global(L,D,S,Di).
- Type of apartments: The apartments are just array of arrays that each subarray is rooms. For example to an floor input

```
floor_dimensions(100,200).
%            Width,Height
open_sides(1,1,0,1). % Line 1
%          N,E,S,W
optional_global(1,1,1,1). % Line 2
%  [LandscapeLook,DistanceToElevator,Symmetry,Divine]

test0(F):-
    As = [A1,A2], % Line 3
    A1 = [R1,R2,R3], % Line 4
    R1=[bedroom,50,10], % Line 5
    R2=[minor_bathroom,20,4,6] , % Line 6
    R3=[kitchen,10], % Line 7
    A2 = [R4,R5,R6],
    R4=[bedroom,10],
    R5=[minor_bathroom,10] ,
    R6=[kitchen,25] ,
    design_floor(As,F). % Line 8
```

  The Lines are illustrated as below :-
  1. 1 means it is on openside/landscape.
  2. 1 means which optional global constraints to activate.
  3. Array of apartments.
  4. Array of rooms of an apartments.
  5. Room defined with minimum area and the width.
  6. Room defined with min area, width and height.
  7. Room only defined with min area.
  8. Predicate which designs the whole floor.
- We did not take the number of apartments or the number of rooms as inputs because they can obtained from the length of the array.

4 Domains

The domains of the variables are bounded by the the width and the height of floor. That all Xs(and the rooms' width) are between 0..Width and Ys(and the rooms' heights) are between 0..Height.

5 Constraints

5.1 Hard Constraints

5.1.1 No Overlapping

This constraint is implemented using the

`disjoint2/1`

predefined predicate is true when the list of rects inputs are not overlapping.

5.1.2 Room Accessibility

This constraint is handled as follows:-

- In case there is an input hallway(s): So we would use the existing hallway to make all the rooms adjacent to them. And make all the hallway adjacent to each other.

- In case there is not an input hallway(s): We would add one hallway and it would the path that connects all the rooms with each other. This is enforced using the predicate "handle_hallway/2" and "make_all_rooms_adjacent_hallway/1"

5.1.3 Apartment Accessibility

This is handled by adding a main hallway and make one hallway of each apartment, the stairs and the elevators adjacent to it.
This is enforced using the predicate "add_main_hallway/2"

5.1.4 Whole space utilized

This constraint is enforced by summing the areas of all rooms in the floor and then maximize it when calling the predicate labeling over the variables.
This is enforced using the predicate "get_the_whole_area(F,A)" where the A is the summation of all areas in the floor and then we maximize it in the options in labeling.

5.1.5 Kitchen, Bathrooms and Ducts

Since the descriptions did not provide much info about the what the ducts are and after googling it, they are the pipes so we assumed them as a unit inside the kitchen and hence the the bathrooms must be adjacent to the kitchen.
This is enforced by the predicate "make_the_kitchen_bathrooms_adjacent/2".

5.1.6 Sun rooms and Day-lightening

This constraint is implemented as it first searches for a sun room it there is a one, it returns one of the options to labeling where it maximizes or minimizes the X or Y coordinate of the sun room to make it on the open sides.

```
make_a_room_on_landscape(R,O):-
    get_room_type(_,X,_,Y,_,R),
    open_sides(N,E,S,W),
    (
        (N=1,O=min(Y));
        (S=1,O=max(Y));
        (E=1,O=max(X));
        (W=1,O=min(X))
    ).
make_the_sun_room_light(L,O):-
    get_the_sun_room(L,S),
    ((S=none, O=none);
    make_a_room_on_landscape(S,O)).
```

It is implemented using the make_the_sun_room_light/2 predicate.

5.1.7 Dressing rooms and Bedrooms

The convention is to write first the dressing room and then the bedrooms assigned to it. It is enforced by using the predicate "make_the_dressing_rooms_bedrooms_adjacent/1".

5.1.8 Bedrooms and Minor Bathrooms

The convention is to write the bedroom first then the minor bathroom adjacent to it.
It is enforced using the predicate "make_the_bedrooms_minor_bathrooms_adjacent/2"

5.1.9 Kitchen and Dining room

It searches for a dining room, if it exists, it enforce the constraint by making both of them adjacent. It is enforced using the predicate "make_the_kitchen_dining_room_adjacent/1"

5.2 Soft Constraints

We have followed the approach mentioned in this paper that states the best approach to implement the soft constraints as follows:

```
solve( Variables ) :
    declare_variables( Variables, CostVariables ),
    post_soft_constraints( Variables ),
    post_hard_constraints( Variables ),
    minimize( Variables, sum(CostVariables) ).
```

So the way to enter the soft constraints is an array of arrays where each subarray is the soft constraints for the corresponding apartment.
It is an array of length 4 of zeroes and ones. One means that the constraint will be activated.

```
implement_soft_constraints([A|T1],[S|C],V):-
    [ExposedDayLight,Distance,BedroomsCloser,MainBathroom] = S,
    (ExposedDayLight=1,calc_exposed_fn(A,EV);ExposedDayLight=0,EV=0),
    (Distance=1,get_all_bedrooms(A,Bs),calc_distance_fn(A,DV);Distance=0,DV=0),
    (BedroomsCloser=1,get_all_bedrooms(A,Bs),calc_bedroom_closer(A,BV);BedroomsCloser=0,BV=0),
    (MainBathroom=1,calc_bathroom_fn(A,MBV);MainBathroom=0,MBV=0),
    V #= EV+DV+BV+MBV+V1,
    implement_soft_constraints(T1,C,V1).
```

The way the soft constraints are implemented is by calculating the cost over the constraints and then minimize it.

5.2.1 All rooms shall be exposed to some form of day-lighting

This is implemented by counting the number of rooms that exposed to the day-light and then when there is a room that is not exposed to the day-light it charges it by one.

5.2.2 Distance between two rooms

It is implemented by getting the two rooms then make sure that the distance is between specific range. If the distance is out of the range, the cost will be increased by one.

5.2.3 Bedrooms are closer

It is implemented by getting the bedrooms and make sure that each adjacent pair of bedrooms are within specific range. If the bedrooms are out of range, the cost will be 1.

5.2.4 Main Bathrooms is easy to access

The cost function is the summation of the distance between the master bathroom and the other rooms then minimize it.

5.3 Optional Global Constraints

These constraints are set whenever it is set to one

```
optional_global(1,1,1,1).
    %  [LandscapeLook,DistanceToElevator,Symmetry,Divine]
```

It is enforced using the "handle_optional_global_constraints/2" predicate.

```
handle_optional_global_constraints(F,Options):-
    optional_global(LandscapeLook,DistanceToElevator,Symmetry,Divine),
    [[E,_],_|Apartments] = F,
    flatten(F, G),
    (LandscapeLook =1, option_landscape(Apartments,Options);LandscapeLook=0,Options=[]), % Line 1
    (DistanceToElevator =1, option_equal_distance_to_elevator(Apartments,E);DistanceToElevator=0), % Line
    (Symmetry =1, option_symmetry([G]);Symmetry=0), % Line 3
    (Divine =1, option_divine_propotion(G);Divine=0). % Line 4
```

5.3.1 All apartment should have a landscape look

This is implemented by one of two ways:-

- **If the apartment has a sun room** That means the apartment has already a look the landscape.

-**If the apartment does not have a sun room** So we choose any room and make it has a look on the landscape.

5.3.2 Equal Distance to the elevator

It is implemeted by finding the hallways in all apartments and then get the middle point in all of them and calculate the distance from them to the elevator and then enforce the equality of the distance that all of them must be equal.

5.3.3 Symmetry

It is implemented as it make sure that the apartment are symmetric using the X and Y coordinates of the rooms.

5.3.4 Divine Proportion

It is implemented by getting the dimensions of each room and then enforce the condition of the divine property

```
option_divine_propotion([R|T]):-
    floor_dimensions(Width,_),
    get_room_type(_,_,W,_,H,R),
    (var(W),
    var(H),
    B in Width,
    H+B#=W,
    W//H #= H//B;true),
    option_divine_propotion(T).
```

6 Caveat

Every predicate is tested successfully, but when the test is about 2 apartments or more it takes so much time.[1]
You can make sure by test them as much as you want.

[1] Literally about half an hour or more to get a solution