Possible World Explorer

Mentor: Prof. Bertram Ludaescher
Intern: Sahil Gupta

Goal

To develop a stand-alone tool that takes as input the output of the Clingo ASP Tool and analyses its attributes and populates them in a relational database, in our case a SQLite Database and Panda DataFrame.

Then use SQL/Panda queries to analyse these possible worlds based on its attributes such as complexity (usually represented by size of the world), overlaps with other worlds, distinguishing features, etc.

To cluster these possible worlds based on these attributes.
Ideal Scenario: Equivalent Solutions can be identified and grouped together.

Analyse if the 'simplest solutions' are the correct solutions (in underspecified problems such as taxonomy alignments). Also referred to as Occam's Razor.

What is ASP?

- ASP stands for Answer Set Programming.
- Tools such as DLV, Clingo used to solve logic problems, optimisation problems, checking satisfiability, etc.
- Often used to find solution to NP-Complete problems such as graph coloring, largest clique, hamiltonian cycle, towers of hanoi, etc.

What the PW-Explorer Tool Does

Input a Clingo Problem → Parse the Output produced by Clingo (using Antlr) → Populate the solution in Panda DataFrames and export to various formats (sqlite .db) to enable querying → Run powerful queries on these DFs/ SQLite DBs → Create Distance Matrix of these PWs and use clustering algorithms to identify similar PWs

Types of Queries

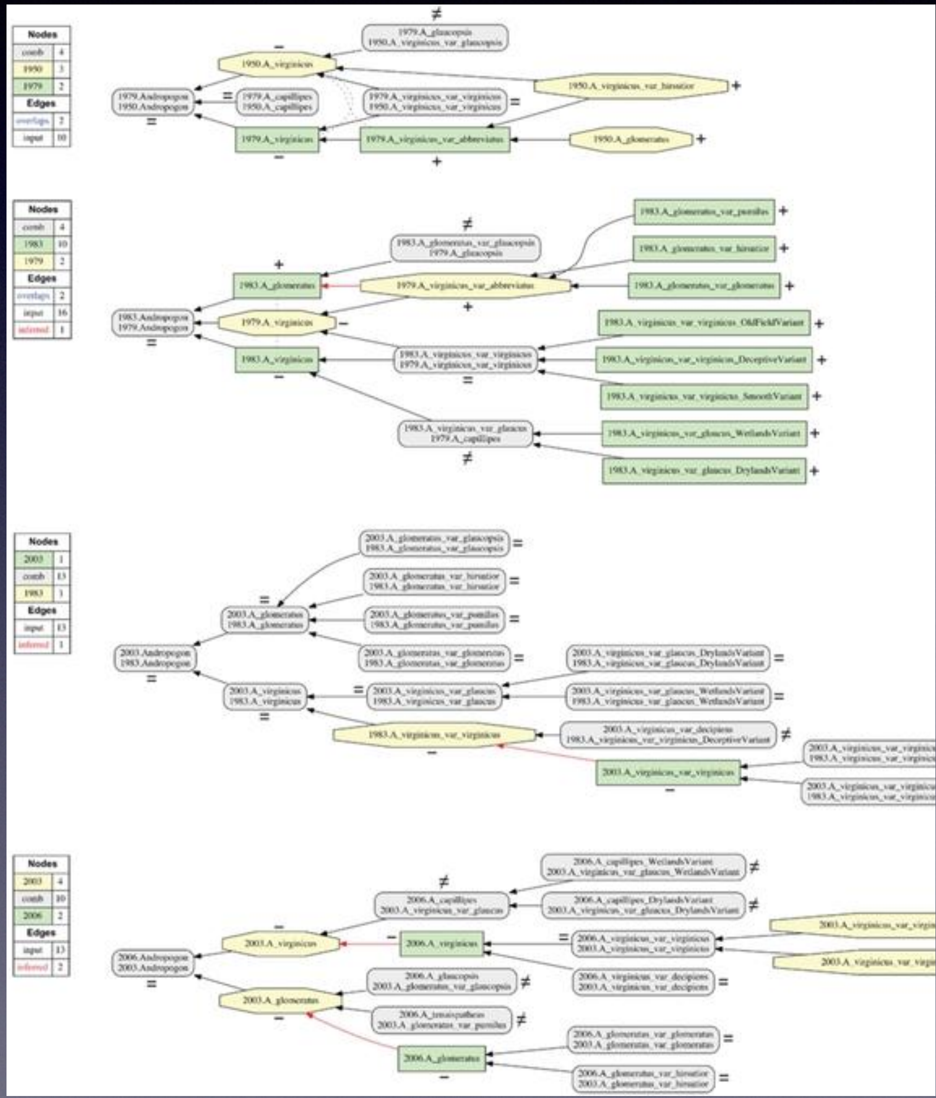
- Intersection
- Union
- Difference and Symmetric Difference
- Frequency of a component across the PWs
- Number of Tuples (can be useful to gauge complexity of a particular solution/PW).
- Identify Redundant Attributes of a Relation
- Identify Unique Tuples (distinguishing features of the PWs from other PWs).

Distance between Solutions?

- Currently using the size of the symmetric set difference of the two PWs as the distance metric.
- Could also account for size of the intersection set, difference in complexity (size of the PWs), etc.
- Dependent on the problem.

Use Cases

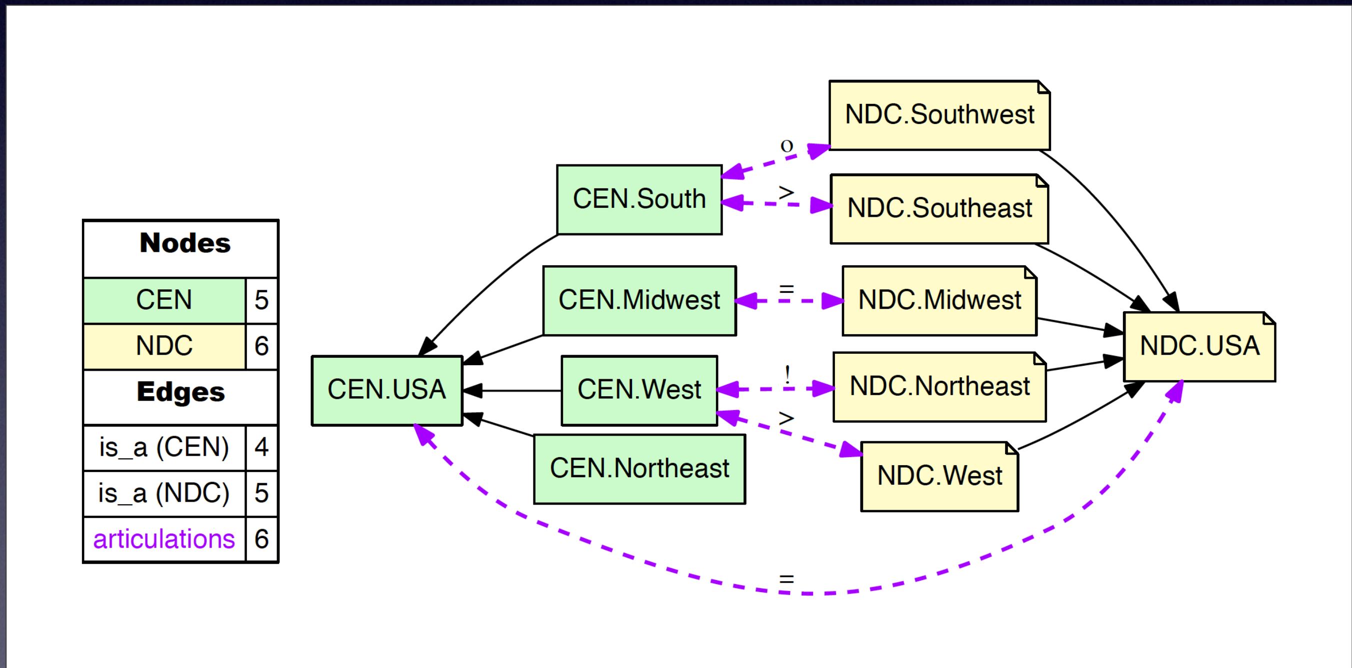
- Taxonomy Alignment (similar to what Euler, a tool developed by Prof. Ludaescher, does)
- Analyse and understand solutions of interesting logic/optimisation problems
- Analyse how certain articulations affect solutions
- Research Tool



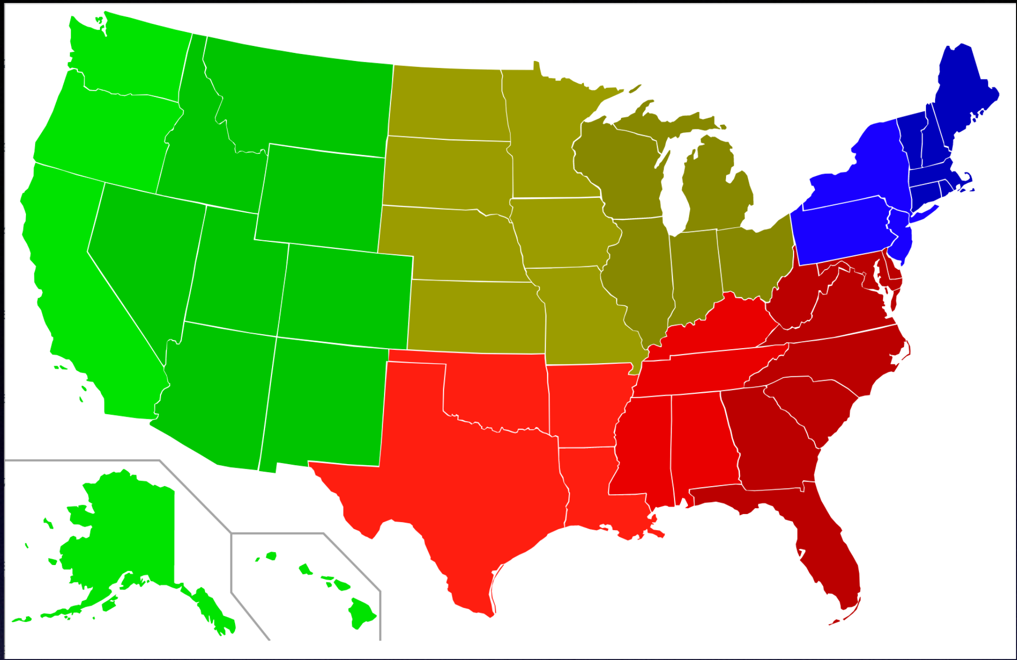
Taxonomy Alignment Example (from Euler)

Input

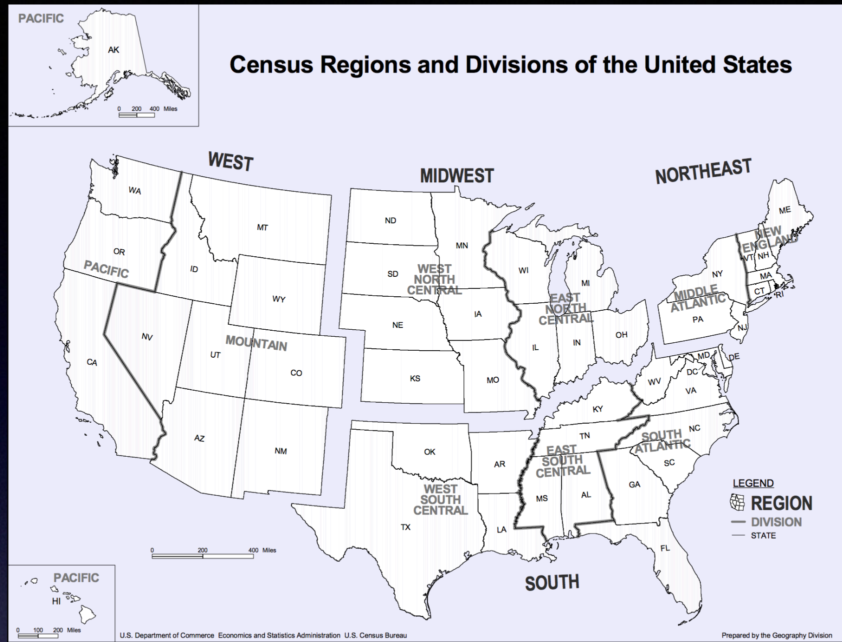
```
taxonomy CEN Census_Regions
(USA Northeast Midwest South West)

taxonomy NDC National_Diversity_Council
(USA Midwest Northeast Southeast Southwest West)

articulations CEN NDC
[CEN.USA equals NDC.USA]
[CEN.West includes NDC.West]
[CEN.West disjoint NDC.Northeast]
[CEN.South overlaps NDC.Southwest]
[CEN.South includes NDC.Southeast]
[CEN.Midwest equals NDC.Midwest]
#[CEN.Northeast is_included_in NDC.Northeast]
```



Underspecified Problem

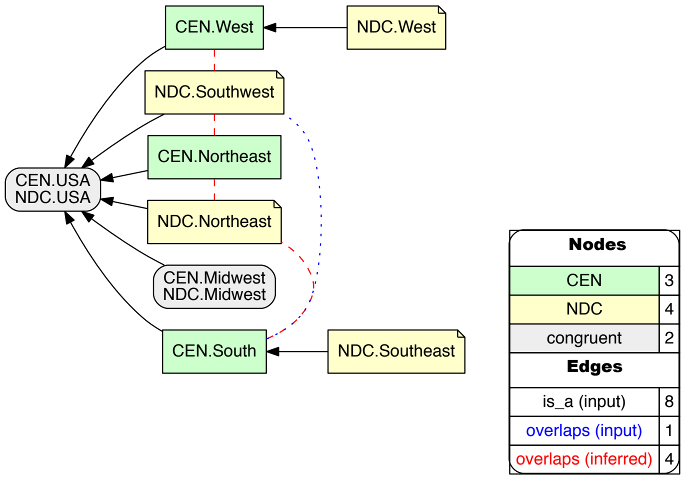


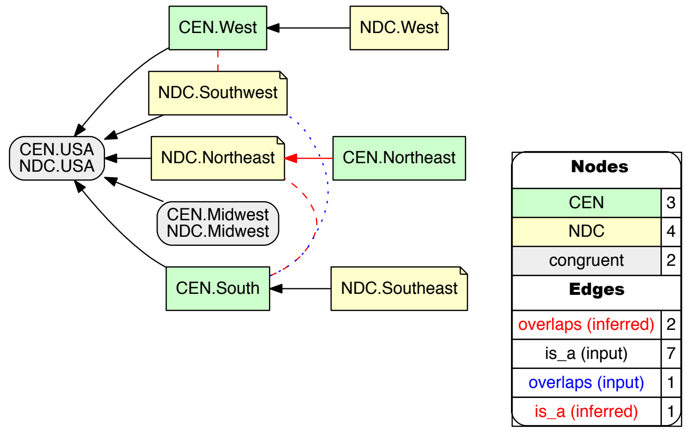
CENSUS Division into 4 regions

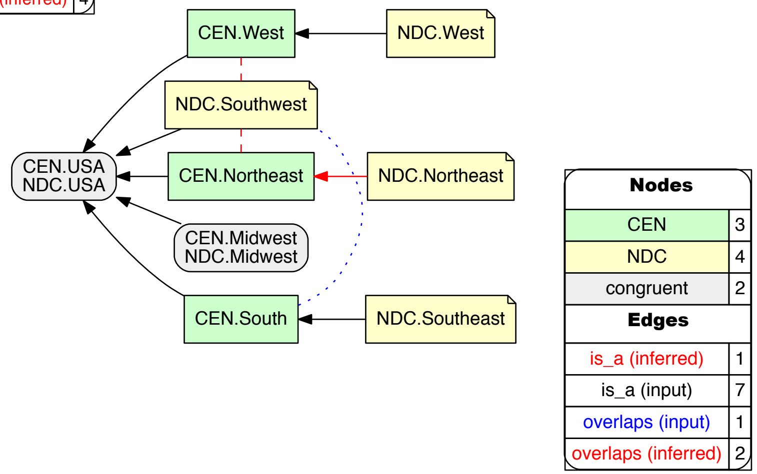


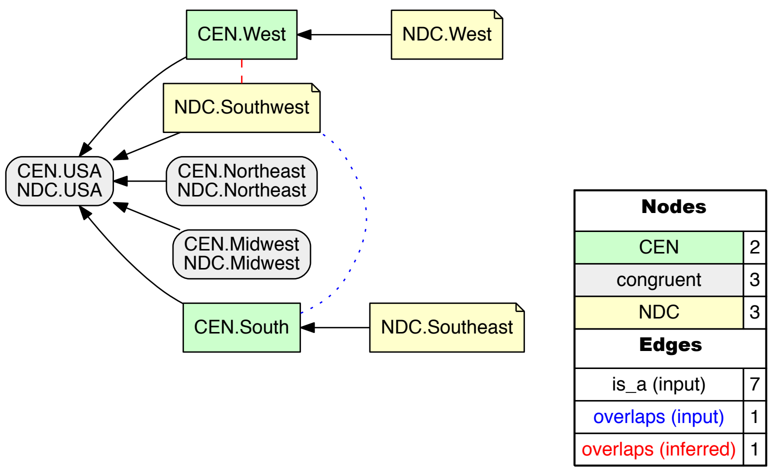
Division into 5 regions by NDC

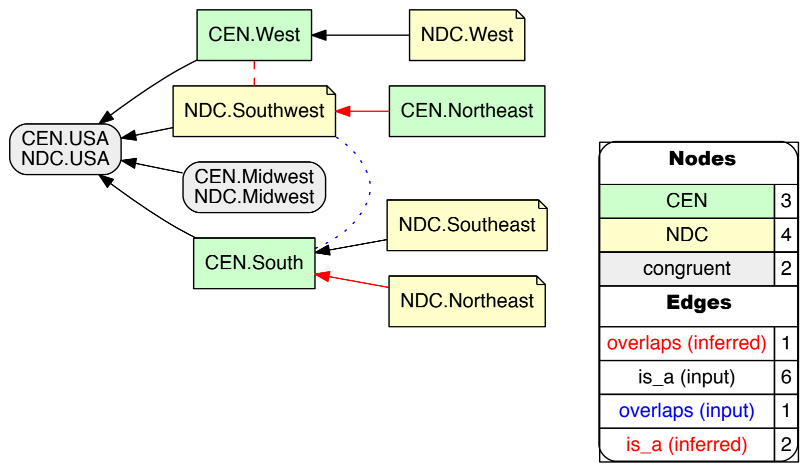
5 PWs


CEN.West
NDC.West
NDC.Southwest
CEN.Northeast
NDC.Northeast
CEN.USA
NDC.USA
CEN.Midwest
NDC.Midwest
CEN.South
NDC.Southeast
Nodes
CEN 3
NDC 4
congruent 2
Edges
is_a (input) 8
overlaps (input) 1
overlaps (inferred) 4


CEN.West
NDC.West
NDC.Southwest
NDC.Northeast
CEN.Northeast
CEN.USA
NDC.USA
CEN.Midwest
NDC.Midwest
CEN.South
NDC.Southeast
Nodes
CEN 3
NDC 4
congruent 2
Edges
overlaps (inferred) 2
is_a (input) 7
overlaps (input) 1
is_a (inferred) 1


CEN.West
NDC.West
NDC.Southwest
CEN.Northeast
NDC.Northeast
CEN.USA
NDC.USA
CEN.Midwest
NDC.Midwest
CEN.South
NDC.Southeast
Nodes
CEN 3
NDC 4
congruent 2
Edges
is_a (inferred) 1
is_a (input) 7
overlaps (input) 1
overlaps (inferred) 2


CEN.West
NDC.West
NDC.Southwest
CEN.USA
NDC.USA
CEN.Northeast
NDC.Northeast
CEN.Midwest
NDC.Midwest
CEN.South
NDC.Southeast
Nodes
CEN 2
congruent 3
NDC 3
Edges
is_a (input) 7
overlaps (input) 1
overlaps (inferred) 1


CEN.West
NDC.West
NDC.Southwest
CEN.Northeast
CEN.USA
NDC.USA
CEN.Midwest
NDC.Midwest
NDC.Southeast
CEN.South
NDC.Northeast
Nodes
CEN 3
NDC 4
congruent 2
Edges
overlaps (inferred) 1
is_a (input) 6
overlaps (input) 1
is_a (inferred) 2

N-Queens Example

Clingo Problem Description

```
#const n = 8.

% domain
number(1..n).

% alldifferent
1 { q(X,Y) : number(Y) } 1 :- number(X).
1 { q(X,Y) : number(X) } 1 :- number(Y).

% remove conflicting answers
:- number(X1;X2;Y1;Y2), q(X1,Y1), q(X2,Y2), X1 < X2, Y1 == Y2.
:- number(X1;X2;Y1;Y2), q(X1,Y1), q(X2,Y2), X1 < X2, Y1 + X1 == Y2 + X2.
:- number(X1;X2;Y1;Y2), q(X1,Y1), q(X2,Y2), X1 < X2, Y1 - X1 == Y2 - X2.


#show q/2.
```

Clingo Output

```
Answer: 1
q(4,1) q(6,2) q(8,3) q(2,4) q(7,5) q(1,6) q(3,7) q(5,8)
Answer: 2
q(3,1) q(5,2) q(2,3) q(8,4) q(1,5) q(7,6) q(4,7) q(6,8)
Answer: 3
q(5,1) q(3,2) q(1,3) q(7,4) q(2,5) q(8,6) q(6,7) q(4,8)
Answer: 4
q(6,1) q(4,2) q(7,3) q(1,4) q(8,5) q(2,6) q(5,7) q(3,8)
Answer: 5
q(5,1) q(3,2) q(1,3) q(6,4) q(8,5) q(2,6) q(4,7) q(7,8)
Answer: 6
q(4,1) q(6,2) q(8,3) q(3,4) q(1,5) q(7,6) q(5,7) q(2,8)
Answer: 7
q(5,1) q(7,2) q(1,3) q(3,4) q(8,5) q(6,6) q(4,7) q(2,8)
Answer: 8
q(8,1) q(3,2) q(1,3) q(6,4) q(2,5) q(5,6) q(7,7) q(4,8)
.
.
.
.
q(3,1) q(5,2) q(7,3) q(1,4) q(4,5) q(2,6) q(8,7) q(6,8)
Answer: 85
q(5,1) q(7,2) q(1,3) q(4,4) q(2,5) q(8,6) q(6,7) q(3,8)
Answer: 86
q(5,1) q(7,2) q(2,3) q(4,4) q(8,5) q(1,6) q(3,7) q(6,8)
Answer: 87
q(5,1) q(3,2) q(8,3) q(4,4) q(7,5) q(1,6) q(6,7) q(2,8)
Answer: 88
q(2,1) q(5,2) q(7,3) q(4,4) q(1,5) q(8,6) q(6,7) q(3,8)
Answer: 89
q(6,1) q(3,2) q(7,3) q(4,4) q(1,5) q(8,6) q(2,7) q(5,8)
Answer: 90
q(6,1) q(8,2) q(2,3) q(4,4) q(1,5) q(7,6) q(5,7) q(3,8)
Answer: 91
q(3,1) q(5,2) q(8,3) q(4,4) q(1,5) q(7,6) q(2,7) q(6,8)
Answer: 92
q(5,1) q(1,2) q(8,3) q(4,4) q(2,5) q(7,6) q(3,7) q(6,8)
SATISFIABLE

Models      : 92
Calls       : 1
Time        : 0.033s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time    : 0.017s
```

92 Possible Solutions

12 Fundamental Solutions

DF Representation

```
Number of Models: 92
q_2
    pw x1 x2
0    1  4  1
1    1  6  2
2    1  8  3
3    1  2  4
4    1  7  5
5    1  1  6
6    1  3  7
7    1  5  8
8    2  3  1
9    2  5  2
10   2  2  3
11   2  8  4
12   2  1  5
13   2  7  6
14   2  4  7
15   2  6  8
16   3  5  1
17   3  3  2
18   3  1  3
19   3  7  4
20   3  2  5
21   3  8  6
22   3  6  7
23   3  4  8
24   4  6  1
..  .. .. ..
720 91  3  1
721 91  5  2
722 91  8  3
723 91  4  4
724 91  1  5
725 91  7  6
726 91  2  7
727 91  6  8
728 92  5  1
729 92  1  2
730 92  8  3
731 92  4  4
732 92  2  5
733 92  7  6
734 92  3  7
735 92  6  8

[736 rows x 3 columns]
```

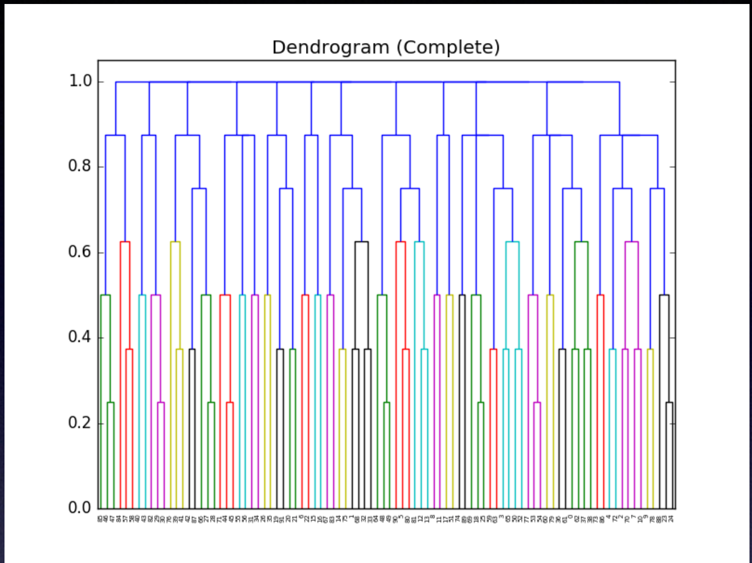
Sqlite Schema:

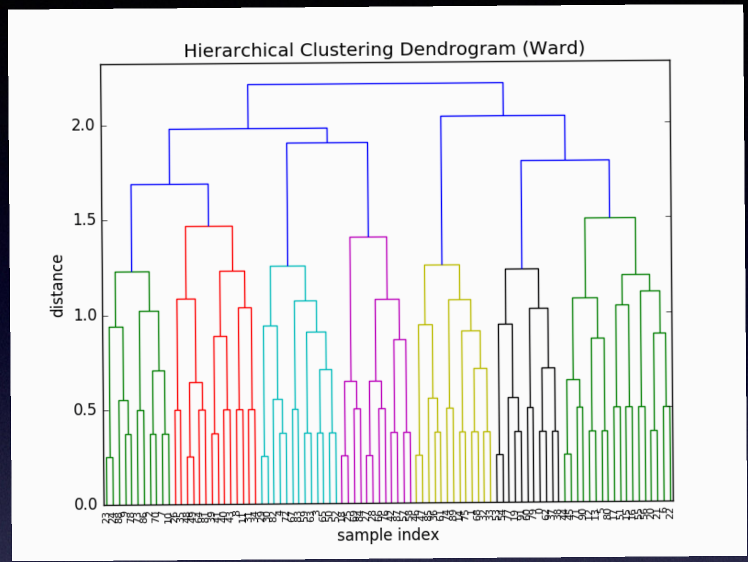
```
CREATE TABLE "q_2"
("index" INTEGER,
  "pw" INTEGER,
  "x1" TEXT,
  "x2" TEXT
)
```

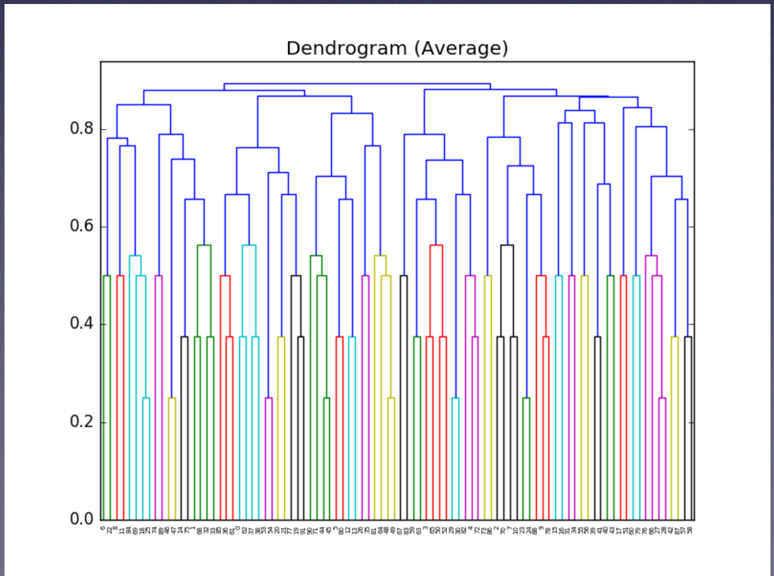
Distance Matrix (Normalised):

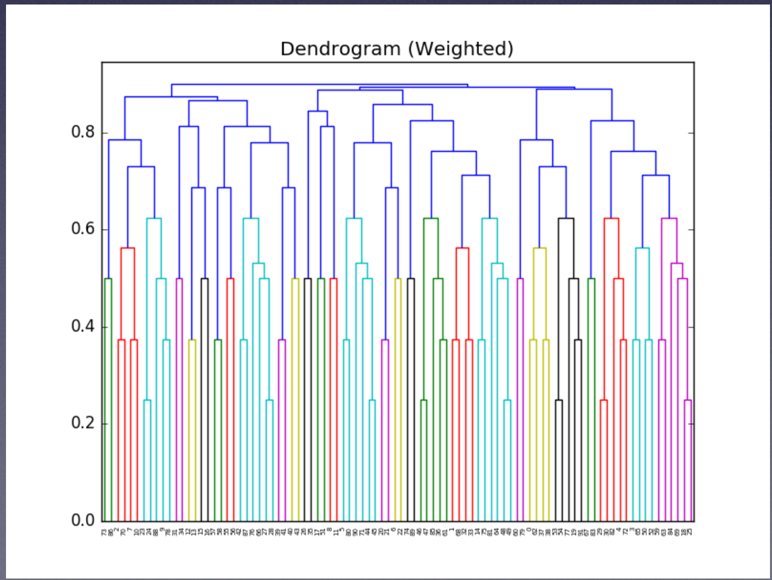
```
[[ 0.     1.     1.    ...,  1.     0.875  0.75 ]
 [ 1.     0.     1.    ...,  0.625  0.375  0.75 ]
 [ 1.     1.     0.    ...,  1.     1.     0.75 ]
 ...,
 [ 1.     0.625  1.    ...,  0.     0.625  0.75 ]
 [ 0.875  0.375  1.    ...,  0.625  0.     0.5  ]
 [ 0.75   0.75   0.75  ...,  0.75   0.5    0.   ]]
```

Some Attempts at Clustering (Still in works)









Some Query Results:

- No redundant columns
- No tuple (in this case an arrangement of a queen on a board) is unique (i.e. occurs in exactly one PW)
- Difference between PWs 21 and 35 is same as their union (i.e. no similarities, i.e. distance between them is 1 (max)).
- The third queen is placed in the 6th column in exactly 4 possible solutions.
- ……and more.

Next Steps

- Formulate a definition of ‘distance’ that helps in clearly grouping similar solutions together (although, there may not be a general definition for this)

- Work on interactive visualisations

Thank You!

Questions?