Answer Set Programing With Clingo

TReNDS Center MLBBQ

Sajad Abavisani - 12/03/2021

Part 1: introduction to Clingo

Declarative programming Vs. imperative

- Imperative : the program is an algorithm to solve the problem.
  - Python, C++, ..
- Declarative: the program is an encoding of the problem itself.
  - Clingo, Prolog, ...

Simple and concise

Eight queen puzzle

```
{q(1..8,1..8)} = 8.
:- q(R,C1), 1(R,C2), C1<C2.
:- q(R1,C), 1(R2,C), R1<R2.
:- q(R1,C1), 1(R2,C2), R1<R2,
 |R1-R2| = |C1-C2|.
```

```
program eightqueen1(output);

var i : integer; q : boolean;
    a : array[ 1 ..  8] of boolean;
    b : array[ 2 .. 16] of boolean;
    c : array[-7 ..  7] of boolean;
    x : array[ 1 ..  8] of integer;

procedure try( i : integer; var q : boolean);
    var j : integer;
    begin
    j := 0;
    repeat
        j := j + 1;
        q := false;
        if a[ j] and b[ i + j] and c[ i - j] then
            begin
            x[ i    ] := j;
            a[     j] := false;
            b[ i + j] := false;
            c[ i - j] := false;
            if i < 8 then
                begin
                try( i + 1, q);
                if not q then
                    begin
                    a[     j] := true;
                    b[ i + j] := true;
                    c[ i - j] := true;
                    end
                end
            else
                q := true
            end
    until q or (j = 8);
    end;

begin
for i :=  1 to  8 do a[ i] := true;
for i :=  2 to 16 do b[ i] := true;
for i := -7 to  7 do c[ i] := true;
try( 1, q);
if q then
    for i := 1 to 8 do write( x[ i]:4);
writeln
end.
```

Different flavors of declarative programs

- Functional programming: like Lisp, Haskell
    - Programming paradigm where programs are constructed by applying and composing functions that map values to other values.

```
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

- Logic programming: like Prolog, SMODELS, Clingo
    - Consists of rules and facts.

```
large(france).
size(france,65).
large(X) :- size(X,S1), size(france,S2), S2<S1.
```

Stable Model

- Stable model of a program consists of all the facts that can be derived using the rules:

```
p(1..10).
large(X) :- p(X), X>5.
```

- (Head) :- (Body) is like a ← if Body, then Head

```
clingo version 5.4.0
Reading from program1.lp
Solving...
Answer: 1
p(1) p(2) p(3) p(4) p(5) p(6) p(7) p(8) p(9) p(10) large(6) large(7) large(8) large
(9) large(10)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Atoms in Clingo

- Symbolic constants : like `country(france).`
  - Starts with lowercase letter
- Numeric constants : like `p(1).`
  - Starts with a digit
- Variable : like `large(X) :- p(X), X>5.`
  - Starts with uppercase letter

Choice Rule

- We can have several stable models using choice rules:

```
{p(a) ; p(b)}.
```

```
Solving...
Answer: 1

Answer: 2
p(b)
Answer: 3
p(a)
Answer: 4
p(a) p(b)
SATISFIABLE

Models       : 4
Calls        : 1
Time         : 0.001s (Solving: 0.00s 1st Model: 0.00s
CPU Time     : 0.001s
```

Example with choice rule

```
person(1..6).
{elected (X) : person(X)} = 3.
```

```
SATISFIABLE

Models        : 20
Calls         : 1
Time          : 0.001s (Solving: 0.00s 1st
CPU Time      : 0.001s
```

Grounding

- An atom without a variable is called “ground”
- Rules become “facts” after grounding:

```
p(1..10).
large(X) :- p(X), X>5.
```

```
clingo version 5.4.0
Reading from program1.lp
Solving...
Answer: 1
p(1) p(2) p(3) p(4) p(5) p(6) p(7) p(8) p(9) p(10) large(6) large(7) large(8) large
(9) large(10)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.002s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.002s
```

Grounding time Vs. Solving time

- Problem : Every nonnegative integer can be represented as sum of 4 complete squares: A = a^2 + b^2 + c^2 + d^2, find number that cannot be represented using three complete squares. Example 7, 15

```
three(N) :- N=1..n, I=0..n, J=0..n, K=0..n, N=I**2 + J**2 + K**2.
more_than_three(N) :- N=1..n, not three(N).
```

```
Models       : 1
Calls        : 1
Time         : 10.345s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 10.334s
```

Reduce time

Symmetry breaking

- The grounding takes a very long time if rules have a lot of variables and grounder needs to instantiate all of them.
- Avoid rules with many variables.
- Symmetry Breaking: using the symmetry of the problem to improve the performance

Reduce time

Symmetry breaking

- Trick one: re-ordering variables : I <= J <= K :

```
three(N) :- N=1..n, I=0..n, J=I..n, K=J..n, N=I**2 + J**2 + K**2.
more_than_three(N) :- N=1..n, not three(N).
```

```
Models       : 1
Calls        : 1
Time         : 1.917s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 1.913s
```

- Trick two: I, J, K cannot be larger than square root of N:

```
sqrt(S) :- S=1..n, S**2 <= n, (S+1)**2> n.
three(N) :- sqrt(S), N=1..n, I=0..S, J=I..S, K=J..S, N=I**2 + J**2 + K**2.
more_than_three(N) :- N=1..n, not three(N).
```

```
Models       : 1
Calls        : 1
Time         : 0.005s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.005s
```

Reduce time

Multi-thread

- ASP is used for intrinsically difficult and NP-Hard problem. Therefore, the worst case will have exponential solving time.
- We can assign multiple CPUs to Clingo so that it can use several solving strategies in parallel
    - To compete against each other:

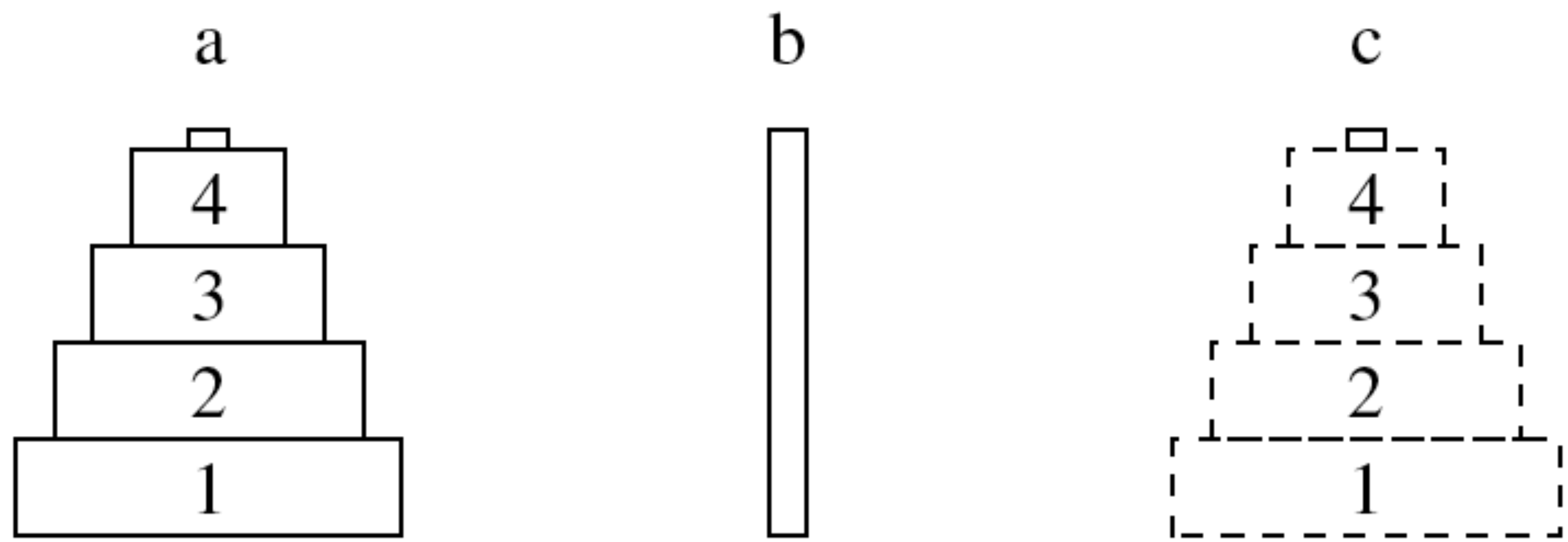
```
$ Clingo program.lp -t 4,compete
```

    - To split the computation:

```
$ Clingo program.lp -t 4,split
```

Part 2: Example with Clingo

Towers of Hanoi puzzle



- The goal is to move all disks from the left peg to the right one, where only the topmost disk of a peg can be moved at a time. Furthermore, a disk cannot be moved to a peg already containing some disk that is smaller.

Towers of Hanoi puzzle

Stating facts

```
peg(a;b;c).
disk(1..4).
init_on(1..4,a).
goal_on(1..4,c).
moves(15).
```

Towers of Hanoi puzzle

Generating moves

```
% Generate
{ move(D,P,T) : disk(D), peg(P) } = 1 :- moves(M),  T = 1..M.
% Define
move(D,T)    :- move(D,_,T).
on(D,P,0)    :- init_on(D,P).
on(D,P,T)    :- move(D,P,T).
on(D,P,T+1) :- on(D,P,T), not move(D,T+1),         not moves(T).
blocked(D-1,P,T+1) :- on(D,P,T), not moves(T).
blocked(D-1,P,T)   :- blocked(D,P,T), disk(D).
% Test
:- move(D,P,T), blocked(D-1,P,T).
:- move(D,T), on(D,P,T-1), blocked(D,P,T).
:- goal_on(D,P), not on(D,P,M), moves(M).
:- { on(D,P,T) } != 1, disk(D), moves(M), T = 1..M.
% Display
#show move/3.
```

- the variables D, P, T, and M are used to refer to disks, pegs, the number of a move, and the length of the sequence of moves, respectively.

Solution

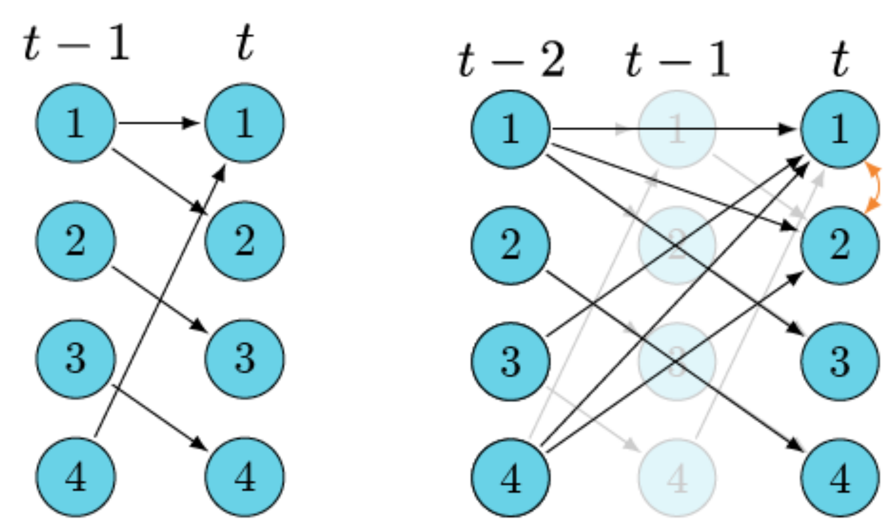
```
Solving...
Answer: 1
move(4,b,1) move(3,c,2) move(4,c,3) move(2,b,4) move(4,a,5) move(3,b,6)
move(4,b,7) move(1,c,8) move(4,c,9) move(3,a,10) move(4,a,11) move(2,c,12)
move(4,b,13) move(3,c,14) move(4,c,15)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.010s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.009s
```

Part 3: Using Clingo for causal structure learning

Problem definition

Undersampled causal graphs

- We can obtain causal graph from its time series. But depending on the intrinsic time scale and the measurement time scale, this graph may not be accurate.
- Having the under sampled graph, we want to retrieve what was the original underlying graph.



(a) DBN $\mathbf{G}^1$ (b) DBN $\mathbf{G}^2$

Approach one: Imperative programming

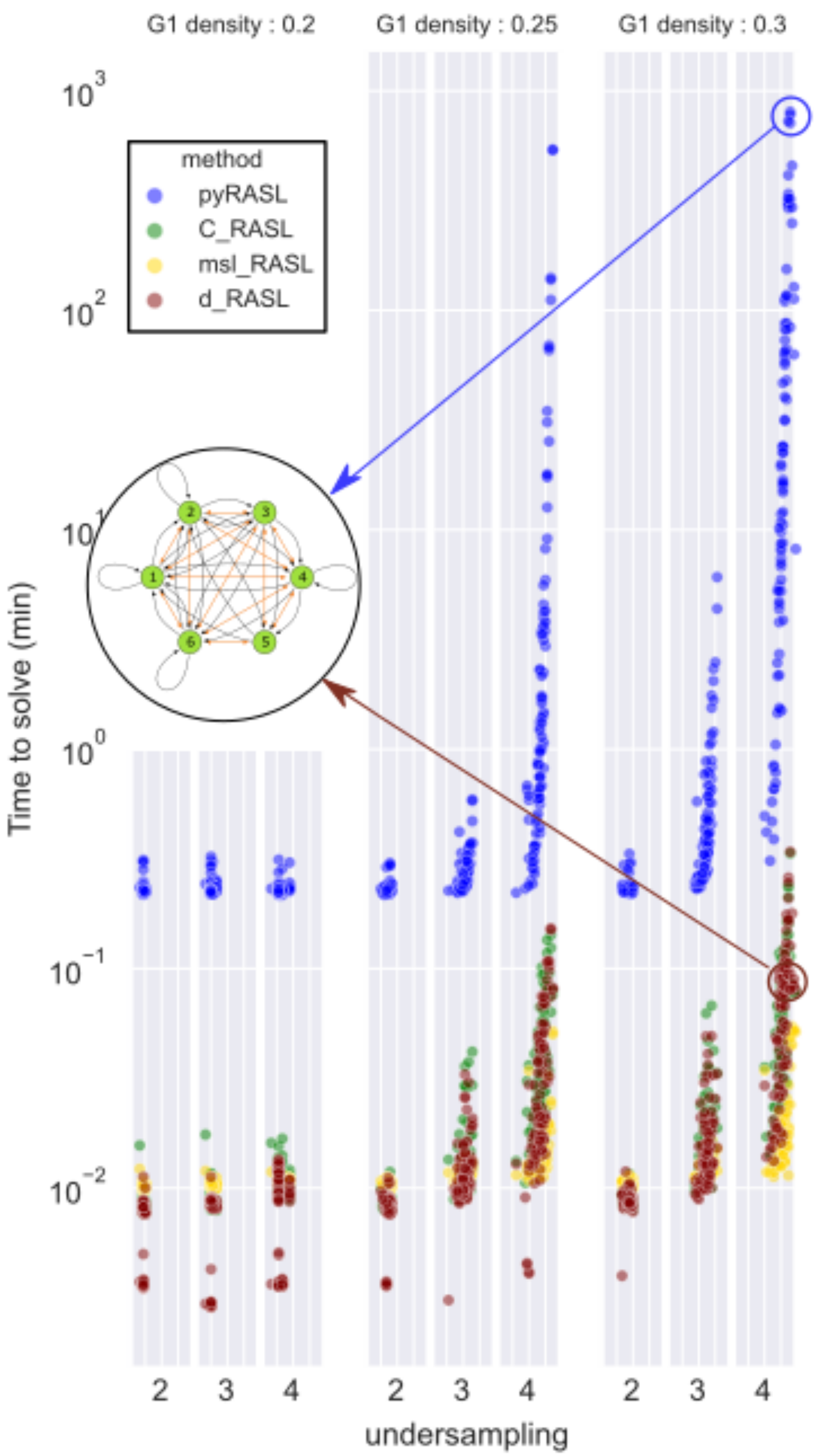
- Using recursive algorithms, we can find all the equivalent class of graphs that can produce the graph at hand by undersampling.
- Draw backs:
    - Cannot handle graphs of size more than 6 nodes.
    - Complicated long code.
    - Takes a long time (17 hours) to compute equivalent class of a graph size 6.

Approach two: ASP with Clingo

- Same problem encode in Clingo gives the same answers much faster (three orders of magnitude) and is able to structured graphs of size up to 100 nodes.

- 14 line of code in total Vs. 600 lines of code!!

Computation time comparison



- 1000 minutes Vs. 6 seconds

Encoding the undersampling graph problem in Clingo

```
#const n = 5. node(1..n). dagl(4). hdirected(1,1,1). hbidirected(1,2,1).
hbidirected(1,3,1). hbidirected(1,5,1). hdirected(2,2,1).
hdirected(2,1,1). hbidirected(2,5,1). hdirected(3,3,1). hdirected(3,1,1).
hbidirected(3,4,1). hdirected(4,4,1). hdirected(4,3,1). hdirected(4,1,1).
hdirected(5,5,1). hdirected(5,1,1). hdirected(5,2,1).
1 {u(1..16, 1)} 1.
{edge1(X,Y)} :- node(X), node(Y).
directed(X, Y, 1) :- edge1(X, Y).
directed(X, Y, L) :- directed(X, Z, L-1), edge1(Z, Y), L <= U, u(U, _).
bidirected(X, Y, U) :- directed(Z, X, L), directed(Z, Y, L), node(X;Y;Z),
X < Y, L < U, u(U, _).

:- directed(X, Y, L), not hdirected(X, Y, K), node(X;Y), u(L, K).
:- bidirected(X, Y, L), not hbidirected(X, Y, K), node(X;Y), u(L, K), X <
Y.
:- not directed(X, Y, L), hdirected(X, Y, K), node(X;Y), u(L, K).
:- not bidirected(X, Y, L), hbidirected(X, Y, K), node(X;Y), u(L, K), X <
Y.
:- M = N, {u(M, 1..1); u(N, 1..1)} == 2, u(M, _), u(N, _).
#show edge1/2.
#show u/2.
```

References

- Answer set programming - by Vladimir Lifschitz (Amazon)
- A User's Guide to gringo, clasp, clingo, and iclingo (pdf)
- Code: Gitlab