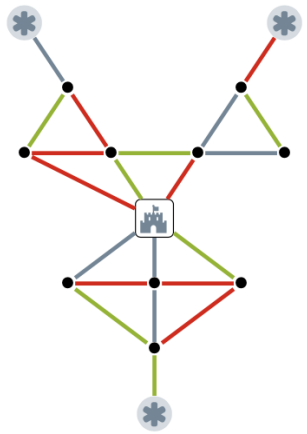
KRR - A tower defense game

Isa-Ali Kirca

ASP:



Day 1: ,

Day 2:

Day 3: ,

Guard types:	☹_a	☹_b	☹_c
- cost	€1	€2	€5
- degree	2	3	4
- max per day	1	1	2

Total budget: €11

Details:

- You have to protect your tower for as many days as possible
- Each day, several waves of attackers come from the nodes
  - You don't know which node they come from—you have to consider any strategy that they might use
  - Different waves can come from different nodes (or from the same)
- The waves can only pass along edges with a certain sequence of colors
- Each day, you can place some guards on some nodes (not on tower, not on node)
  - Guards will stop all attackers on that node (on that day)
  - There are different guard types, each with (i) their own cost, (ii) restrictions on the degree of nodes that they can be placed on, and (iii) maximum number of guards of that type that can be placed on any given day
- Given some budget, what is the maximum number of days that you can survive?

Encode input

Representing the graph:

```
% Nodes and colored edges
node(1..14).
edge(1,3,blue).
edge(3,6,red).
edge(3,5,green).
edge(5,6,red).
edge(6,7,green).
edge(7,8,blue).
edge(2,4,red).
edge(4,7,blue).
edge(4,8,green).
edge(5,9,red).
edge(6,9,green).
edge(7,9,red).
edge(9,10,blue).
edge(9,11,blue).
edge(9,12,green).
edge(10,11,red).
edge(11,12,red).
edge(10,13,green).
edge(11,13,blue).
edge(12,13,red).
edge(13,14,green).

% Edges are also reversed
edge(X,Y,C) :- edge(Y,X,C).

% Tower
tower(9).

% Spawn points
spawn(1).
spawn(2).
spawn(14).
```

Representing the waves:

```
% Waves – wave_on_day(W,D) indicates that wave W happens on day D
% wave(W,I,C) indicates that the I'th step in wave W has color C
wave_on_day(1,1).
wave(1,1,blue).
wave(1,2,red).
wave(1,3,green).
wave(1,4,red).
wave_on_day(2,1).
wave(2,1,red).
wave(2,2,green).
wave(2,3,blue).
wave(2,4,red).
wave_on_day(3,2).
wave(3,1,green).
wave(3,2,blue).
wave(3,3,red).
wave(3,4,blue).
wave_on_day(4,3).
wave(4,1,red).
wave(4,2,blue).
wave(4,3,red).
wave_on_day(5,3).
wave(5,1,blue).
wave(5,2,red).
wave(5,3,green).
% wave(W) :- wave_on_day(W,_).
```

Representing the guard types and budget:

```
% Guard types
guard_type(a;b;c).
guard_type_degree(a,2).
guard_type_degree(b,3).
guard_type_degree(c,4).
guard_type_cost(a,1).
guard_type_cost(b,2).
guard_type_cost(c,5).
guard_type_limit_per_day(a,1).
guard_type_limit_per_day(b,1).
guard_type_limit_per_day(c,2).

% Budget
budget(12).
```

Encode candidate solutions

Choosing a number of days to survive:

```
% Choose a sequence of days day(1), ..., day(j) to survive
max_day(D) :- D = #max { E : wave_on_day(_,E) }.
possible_day(D) :- max_day(D).
possible_day(D-1) :- possible_day(D), D > 1.
{ day(D) : possible_day(D) }.
:- day(D), possible_day(D-1), not day(D-1). % symmetry breaking
    % days that we play must be subsequent: 1,2,...
```

Choosing a basic strategy:

```
% Choose a strategy for placing guard on nodes on different days
% place(D,G,N) indicates that on day D, a guard of type G is placed on node N
{ place(D,G,N) : day(D), guard_type(G), node(N) }.

% Guards cannot be placed on spawn points or on the tower
:- place(D,G,N), spawn(N).
:- place(D,G,N), tower(N).
```

Encode solution properties

Respect budget, check guard restrictions and compute degree of nodes:

```
% Compute the degree of nodes
degree(N,D) :- node(N), D = #count { M : edge(N,M,C) }.

% Ensure that guard types are only placed on nodes with the allowed degree
:- place(D,G,N), day(D), guard_type(G),
   guard_type_degree(G,E), not degree(N,E).

% Ensure that on each day, the limits on the different guard types are respected
guard_type_used_per_day(G,D,M) :-
    guard_type(G), day(D),
    M = #count { N : place(D,G,N) }.

:- guard_type_used_per_day(G,D,M),
    guard_type(G), day(D),
    guard_type_limit_per_day(G,L), M > L.
```

Checking that the tower remains safe:

```
% Derive what nodes the different waves can reach
% reachable(W,D,N,I) indicates that wave W (up to index I) can reach node N on day D
reachable(W,D,N,0) :- wave_on_day(W,D), spawn(N).
reachable(W,D,N,I) :- wave_on_day(W,D), I > 0,
    reachable(W,D,M,I-1), edge(M,N,C), wave(W,I,C),
    not place(D,G,N) : guard_type(G).

% Ensure that the tower is never reached (after any part of a wave)
:- day(D), tower(N), reachable(_,D,N,_).
```

Optimization

Maximizing the number of survived days:

```
% Maximize the number of days survived
#maximize { 1,day(D) : day(D) }.
```

Show:

```
% Show only the strategy and how which days survived
#show day/1.
#show place/3.
```

Now using propositional logic:

We will construct a **CNF formula** φ_k that:

- is is statisfiable if and only if there is a strategy to survive the first k days,
- has satisfying assignments that correspond to strategies to survive the first k days.

We introduce propositional variables $p_{d\,,g\,,n}$ for each day $d \in \{1, \ldots, k\}$, each guard type g and each node n.

- The truth assignment to these variables will represent the strategy:
  $p_{d\,,g\,,n}$ represents whether on day d we place a guard of type g on node n.
- All other variables that we introduce will only be auxiliary variables that we use to express the various constraints.

For each tower or node n, and each day $d \in \{1, \ldots, k\}$ and guard type g, we add the clause:

$$(\neg p_{d\,,g\,,n})$$

Expressing the degree restrictions:

- Let $\mathbf{d}(n)$ be the degree of a node, and let $\mathbf{d}(g)$ be the degree of nodes that guards of type g may be placed on.
- For each node n, for each guard type g such that $\boldsymbol{d}(n) \neq \boldsymbol{d}(g)$, and for each day $d \in \{1, \ldots, k\}$, we add the clause:

$$(\neg p_{d,g,n})$$

First auxiliary variables:

We add auxiliary variables $q_{d\,,n}$ to represent whether on day d node n is guarded. To **enforce** this, we add the following clauses:

$$(\neg p_{d,g,n} \vee q_{d,n}) \text{ for each } d, g, n;$$
$$(\neg q_{d,n} \vee \bigvee_{g \in G} p_{d,g,n}) \text{ for each } d, n;$$

Expressing reachability:

We add variables $r_{d,w,n,i}$ to represent whether wave w on day d can reach node n after i steps. To **enforce** this, we add the following clauses:

$(r_{d,w,n,0})$	for each d, w and each node n;
$(\neg r_{d,w,n_1,i} \vee q_{d,n_2} \vee r_{d,w,n_2,i+1})$	for each d, w, i, n_1, n_2 such that there is an
$(\neg q_{d,n_2} \vee \neg r_{d,w,n_2,i+1})$	edge between n_1 and n_2 with color c, where
	the $(i+1)$-th 'leg' of w has color c;
$\left(\bigvee_{n_2 \in N_c} r_{d,w,n_2,i} \vee \neg r_{d,w,n_1,i+1}\right)$	for each d, w, i, n_1 where N_c is the set of
$(\neg q_{d,n_2} \vee \neg r_{d,w,n_2,i+1})$	nodes n_2 with an edge of color c to node n_1
	and where the $(i+1)$-th 'leg' of w has
	color c;

Finally, for each tower node n, and each d, w, i, we add the clause:

$$(\neg r_{d,w,n,i})$$

Expressing the budget restrictions:

Let's order the variables $p_{d,g,n}$ in some order: $p_1, \ldots, p_u$.

- So $p_1, \ldots, p_u$ are just different names that we use to refer to the variables $p_{d,g,n}$, for the sake of presentation.

We add variables $s_{j,b}$ for each $0 \leq j \leq u$ and each $0 \leq b \leq$ budget to represent whether the assignment of variables $p_1, \ldots, p_j$ corresponds to spending b.

To enforce this, we add the following clauses:

$(s_{0,0})$	
$(\neg s_{0,b})$	for each $b > 0$;
$(\neg s_{j-1,b-c} \vee \neg p_j \vee s_{j,b})$	for each $j > 0$, where p_j is a variable for guard placement
$(s_{j-1,b-c} \vee \neg p_j \vee \neg s_{j,b})$	with cost c, and for each $b \geq c$;
$(\neg p_j \vee \neg s_{j,b})$	for each $j > 0$, where p_j is a variable for guard placement
	with cost c, and for each $b < c$;
$(\neg s_{j-1,b} \vee p_j \vee s_{j,b})$	for each $j > 0$ and for each b;
$(s_{j-1,b} \vee p_j \vee \neg s_{j,b})$	

Finally, we add the following constraint that expresses that we must stay within the budget:

$$\left(\bigvee_{1 \leq b \leq \text{ budget}} s_{u,b} \right)$$

Expressing the budget restrictions:

To express the constraints that on each individual day, only a limited number of guards of type g may be placed, we would need to do the following:

- For each day d, and each guard type g, we need to add auxiliary variables to express how many guards of type g have been placed on day d.
- And we need to add clauses that enforce this intended meaning, and that express that these amounts are within the limit, for each d and each g.
- (The details are similar to the budget restrictions, and we don't show them here.)

Expressing the budget restrictions:

In propositional logic we cannot express optimization statements, so to find the largest number of days that we can survive, we do the following:

- Construct all the formulas $\varphi_1, \varphi_2, \varphi_3, \ldots, \varphi_m$, where m is the number of days.
- If φ_i is satisfiable, then φ_{i-1} is also satisfiable.
  If φ_i is unsatisfiable, then φ_{i+1} is also unsatisfiable.
  So $\varphi_1, \ldots, \varphi_\ell$ are satisfiable and $\varphi_{\ell+1}, \ldots, \varphi_m$ are unsatisfiable, for some ℓ.
- Use a SAT solver to check which formulas are satisfiable.
- The first ℓ for which $\varphi_{\ell+1}$ is unsatisfiable is the maximum number of days that you can survive.
- To compute a strategy to survive ℓ days, look at the truth assignment that satisfies φ_ℓ.