Desafio 1 — Horário de exames com vigilâncias

Inteligência Artificial · Licenciatura em Engenharia Informática · 2026/27 · Lançado na aula prática 6.

Este desafio **não é avaliado e não se entrega**. Está tudo na página do desafio (`d1.html`): este enunciado, as onze instâncias e o verificador (também num só ficheiro, `desafio1.zip`). O objectivo é escrever um programa em ASP que resolva **cada uma das instâncias em menos de 45 segundos** e cuja resposta **passe no verificador**.

No teste e/ou no exame, poderá haver perguntas sobre o que se aprende neste desafio: porque é que uma codificação escala e outra não, onde é que o tempo se gasta (*grounding* ou procura), como se mostra que uma instância não tem solução. Quem pedir a um LLM que faça o desafio por si fica com um programa que corre, mas chega ao teste e ao exame sem ter percebido porquê.

O problema

A época de exames tem que ser calendarizada. Há um conjunto de **exames**, um conjunto de **slots** (períodos de exame, numerados 1, 2, 3, …), um conjunto de **salas** e um conjunto de **docentes** que vigiam os exames.

- Cada sala tem um **tipo** (`anfiteatro`, `sala` ou `laboratorio`) e uma **capacidade**.
- Cada exame tem um **tipo** (`escrito` ou `computador`), um número de **inscritos** e um número de **vigilantes** de que precisa. A instância diz que tipos de sala servem cada tipo de exame (`compativel/2`).
- Alguns exames partilham alunos (estão **em conflito**) e não podem ficar demasiado próximos.
- Alguns exames já têm o slot **fixado**.
- Cada docente pode fazer um **número máximo de vigilâncias** e pode estar **indisponível** em alguns slots.

É preciso escolher, para cada exame, um slot, uma sala e os seus vigilantes, de modo a respeitar todas as restrições seguintes.

	Restrição
H1	Cada exame é atribuído a exactamente um slot e a exactamente uma sala.
H2	O número de inscritos num exame não pode exceder a capacidade da sala.
H3	O tipo da sala tem que ser compatível com o tipo do exame.
H4	Uma sala não recebe dois exames no mesmo slot.
H5	Dois exames em conflito não podem ficar no mesmo slot.
H6	Dois exames em conflito não podem ficar em slots consecutivos (S e S+1).
H7	Um exame com slot fixado tem que ficar nesse slot.
H8	Cada exame tem exactamente o número de vigilantes que pede, todos diferentes.
H9	Um docente não vigia dois exames no mesmo slot.
H10	Um docente não vigia num slot em que está indisponível.
H11	Um docente não faz mais vigilâncias do que o seu máximo.

Algumas instâncias **não têm solução**. Nessas, a resposta certa é `UNSATISFIABLE`.

Formato das instâncias

Uma instância é um ficheiro de factos com os predicados seguintes.

Predicado	Significado
`exame(E)`, `slot(S)`, `sala(R)`, `docente(D)`	os exames, os slots (inteiros 1..N), as salas e os docentes
`tipo(R,T)`, `capacidade(R,C)`	a sala R é do tipo T e leva C alunos
`tipo_exame(E,T)`, `inscritos(E,N)`	o exame E é do tipo T e tem N inscritos
`compativel(TE,TR)`	um exame do tipo TE pode ser feito numa sala do tipo TR
`vigilantes(E,K)`	o exame E precisa de K vigilantes
`conflito(E1,E2)`	E1 e E2 partilham alunos
`fixado(E,S)`	o exame E tem que ficar no slot S
`max_vigias(D,M)`	o docente D faz no máximo M vigilâncias
`indisponivel(D,S)`	o docente D não pode vigiar no slot S

A relação `conflito/2` é **simétrica, mas os factos aparecem numa só direcção**: se o ficheiro tem `conflito(e1,e5)`, não tem `conflito(e5,e1)`. Tratem disso no programa.

A instância mais pequena, `std1.lp`, completa:

```
exame(e1). exame(e2). exame(e3). exame(e4). exame(e5).
exame(e6). exame(e7). exame(e8). exame(e9). exame(e10).
exame(e11). exame(e12).
slot(1..6).
sala(a1). sala(s1). sala(s2). sala(l1).
docente(d1). docente(d2). docente(d3). docente(d4).
docente(d5). docente(d6).

compativel(escrito,anfiteatro). compativel(escrito,sala).
compativel(computador,laboratorio).
tipo(a1,anfiteatro). tipo(s1,sala). tipo(s2,sala).
tipo(l1,laboratorio).
capacidade(a1,300). capacidade(s1,40). capacidade(s2,80).
capacidade(l1,40).
tipo_exame(e1,escrito). tipo_exame(e2,computador).
tipo_exame(e3,escrito). tipo_exame(e4,escrito).
tipo_exame(e5,escrito). tipo_exame(e6,escrito).
tipo_exame(e7,escrito). tipo_exame(e8,escrito).
tipo_exame(e9,escrito). tipo_exame(e10,escrito).
tipo_exame(e11,computador). tipo_exame(e12,escrito).
inscritos(e1,78). inscritos(e2,35). inscritos(e3,168).
inscritos(e4,287). inscritos(e5,27). inscritos(e6,22).
inscritos(e7,79). inscritos(e8,32). inscritos(e9,253).
inscritos(e10,69). inscritos(e11,31). inscritos(e12,40).
vigilantes(e1,2). vigilantes(e2,1). vigilantes(e3,3).
vigilantes(e4,3). vigilantes(e5,1). vigilantes(e6,1).
vigilantes(e7,2). vigilantes(e8,1). vigilantes(e9,3).
vigilantes(e10,2). vigilantes(e11,1). vigilantes(e12,1).
conflito(e1,e4). conflito(e1,e5). conflito(e1,e8).
conflito(e9,e3). conflito(e4,e7). conflito(e4,e12).
conflito(e7,e12). conflito(e10,e9). conflito(e11,e10).
fixado(e2,3). fixado(e12,3).
max_vigias(d1,4). max_vigias(d2,3). max_vigias(d3,5).
max_vigias(d4,3). max_vigias(d5,4). max_vigias(d6,3).
indisponivel(d3,3). indisponivel(d3,6). indisponivel(d6,3).
```

Formato da resposta

Se a instância tem solução, o modelo devolvido tem que conter:

- exactamente um átomo `assign(E,S,R)` por exame — «o exame E realiza-se no slot S, na sala R»;
- para cada exame E, um átomo `vigia(E,D)` por cada um dos seus vigilantes — «o docente D vigia o exame E».

Outros átomos podem aparecer (o verificador ignora-os), mas o mais limpo é terminar o programa com

```
#show assign/3.
#show vigia/2.
```

Se a instância não tem solução, a resposta é `UNSATISFIABLE`.

As instâncias

Instância	Exames	Slots	Salas	Docentes	Conflitos	Tem solução?
`std1.lp`	12	6	4	6	9	sim
`std2.lp`	40	10	13	14	90	sim
`std3.lp`	120	20	42	40	621	sim
`std4.lp`	300	30	110	90	2774	sim
`std5.lp`	450	35	140	120	5147	sim
`ap1.lp`	240	20	60	55	2817	sim, apertada
`ap2.lp`	240	20	60	55	2866	sim, apertada
`ap3.lp`	240	20	60	55	2804	sim, apertada
`ss1.lp`	250	24	71	70	2283	não
`ss2.lp`	250	24	71	70	2138	não
`ss3.lp`	250	24	71	70	2322	não

As cinco `std` crescem de tamanho e têm folga. As três `ap` são do mesmo tamanho, mas muito mais apertadas: muitos conflitos, laboratórios quase todos ocupados e docentes sem folga (nenhum docente pode fazer mais vigilâncias do que as que lhe tocam). As três `ss` não têm solução, e cada uma por uma razão diferente.

Há uma solução que resolve cada uma das onze instâncias em menos de 5 segundos num portátil normal. Não é preciso chegar aí: 45 segundos chegam.

Como correr e como medir

Podem correr o CLINGO como quiserem: com as opções que quiserem, mais do que uma vez, ou a partir de um programa em Python que use o módulo `clingo` (`pip install clingo`). O que conta é o **tempo total**, desde que a instância é lida até à resposta. A forma mais simples:

```
clingo programa.lp std3.lp > resposta.txt
python3 verifica.py std3.lp resposta.txt
```

ou, numa só linha, `clingo programa.lp std3.lp | python3 verifica.py std3.lp -`.

O verificador (`verifica.py`, só precisa do Python 3) lê a resposta, tal como o CLINGO a escreve (usa o último `Answer:`), e diz `CORRECTO` ou `ERRADO`. Quando está errada, lista todas as violações, uma por

linha, com a restrição (H1 a H11) e os exames, salas e docentes envolvidos. Se a resposta for `UNSATISFIABLE`, o verificador diz se a instância é de facto uma das que não têm solução.

Para saber onde se gasta o tempo, `clingo --stats` mostra, no fim, o tempo total e o tempo de procura (*Solving*), o número de átomos e de regras depois do *grounding*. A diferença entre o tempo total e o de procura é, quase toda, *grounding*.

Por onde começar

1. Escrevam primeiro a versão mais directa que conseguirem e ponham-na a resolver `std1`, `std2` e `std3`. Confirmem com o verificador.
2. Corram as maiores com `--stats`. Para a `std5`, quantos átomos tem a vossa escolha? Comparem com 450 × 35 × 140. Onde se gasta o tempo?

Sobre o uso de LLMs

Podem usar o Claude, o Codex ou o que quiserem: não há nota nem nada para policiar. Mas saibam ao que vão. Um LLM actual escreve, à primeira, um programa que resolve as instâncias mais pequenas, e talvez resolva todas. No entanto, não compreenderão porque é que uma versão demora dez minutos e outra dois segundos. E saber isso poderá ser importante no teste e no exame. Para além disso, ao usarem um LLM, perdem o prazer de superar o desafio.