Pràctica de PDA, SAT (2024-2025)

Pablo Pozo Alonso i Sergi López Romero

Gener 2025

1 Introducció

El problema del Buscamines (*Minesweeper*) és un conegut joc de lògica que es pot modelar com un problema de satisfacció de restriccions (CSP, per les seves sigles en anglès). Aquest informe descriu el model utilitzat per representar i resoldre el problema mitjançant un solver SAT.

L'objectiu és determinar la ubicació de les mines en el taulell, seguint les pistes proporcionades per les cel·les numèriques i respectant les regles del joc.

2 Model del problema

El model utilitza el següent *viewpoint*:

2.1 Variables

- Cada cel·la del taulell es modela com una variable booleana $x_{i,j}$, on:

$$x_{i,j} = \begin{cases} \text{true (1)} & \text{si la cel·la } (i,j) \text{ conté una mina,} \\ \text{false (0)} & \text{si la cel·la } (i,j) \text{ no conté una mina.} \end{cases}$$

2.2 Domini

El domini de cada variable és $\{0, 1\}$, corresponent als valors booleans (mina o no mina).

2.3 Funció objectiu

El problema no busca optimitzar cap valor, sinó trobar una assignació que satisfaci totes les restriccions imposades pel taulell i les pistes proporcionades.

2.4 Restriccions

Les restriccions s'han definit de la següent manera:

2.4.1 Restriccions bàsiques

1. Cel·les amb pistes conegudes: - Si una cel·la conté un número n, aquest indica que exactament n de les cel·les adjacents contenen mines. Aquesta restricció es modela utilitzant una suma:

$$\sum_{(k,l)\in \text{adjacents}(i,j)} x_{k,l} = n.$$

- A més, una cel·la amb pista no pot ser una mina:

$$x_{i,j} = 0.$$

2. Cel·les marcades com segures (`X` al taulell): - Si una cel·la està marcada com a segura, no pot contenir una mina:

$$x_{i,j} = 0.$$

3. Nombre total de mines: - Si es coneix el nombre total de mines M al taulell, es pot imposar una restricció global:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} x_{i,j} = M.$$

2.4.2 Restriccions implicades

Les restriccions implicades són aquelles que es dedueixen indirectament: - Si una cel·la té una pista $n = 0$, totes les cel·les adjacents han de ser segures:

$$\forall (k,l) \in \text{adjacents}(i,j),\ x_{k,l} = 0.$$

- Si una cel·la té una pista $n = 8$, totes les cel·les adjacents són mines:

$$\forall (k,l) \in \text{adjacents}(i,j),\ x_{k,l} = 1.$$

2.5 Trencament de simetries

No s'han aplicat tècniques específiques per trencar simetries. Tot i això la restricció global del nombre total de mines (M) evita assignacions redundants.

3 Configuracions de cardinality constraints

En el nostre model del buscamines només fem servir `AddEK`. Per a les proves de rendiment de les codificacions AMO i EO, hem fet servir el model de les n-reines, ja que també ens permet jugar amb la mida del tauler per observar com varia l'eficiència entre ambdues implementacions.

- Les diferències conforme va canviant la mida són les esperades. Amb una mida del tauler petita, la diferència de temps no és rellevant. En alguns casos, la codificació quadràtica obté un temps lleugerament menor, però la diferència és mínima.
- Quan arribem a `nreines` = 300, ja es pot observar que la codificació logarítmica comença a obtenir unes marques de temps bastant per sobre de la quadràtica.
- Amb `nreines` = 350, la màquina utilitzada donava error per excés de memòria amb la codificació quadràtica, mentre que amb la logarítmica sí que s'aconseguia resoldre el model.
- Fent servir aquesta codificació logarítmica, vam arribar a obtenir una solució sense excedir els límits de memòria amb una mida del tauler de fins a `nreines` = 1000.

nreines	**encoding quad**	**encoding log**
20	0.091 s	0.089 s
50	0.053 s	0.30 s
250	1.55 s	1.39 s
300	5.78 s	1.73 s
350	out of memory	91.43 s
500	out of memory	44.71 s
700	out of memory	256.48 s
1000	out of memory	41.70 s

Taula 1: Resultats per encoding quad i encoding log en nreines.

4 Proves amb diferents instàncies

Per a les següents proves, hem resolt totes les instàncies que es donaven i n'hem afegit una de pròpia que vam trobar en una pàgina d'internet, aquesta és la més gran que vam poder trobar i consta de 14 files, 24 columnes i 108 bombes. Per a la primera prova vam fer servir el model base, és a dir sense cap restricció implicada. Per a la segona prova vam afegir la següent restricció implicada: si una cel·la té un 0, obligatòriament les cel·les adjacents són segures, és a dir que no hi pot haver cap bomba. La darrera restricció implicada que vam implementar és que al mirar les caselles adjacents d'una cel·la, si aquestes són iguals al nombre de bombes de la pròpia cel·la, aleshores totes elles han de ser una bomba per força.

	model bàsic	**restricció 0 adjacents**	**restricció nBombes = nCaselles adjacents**
buscamines1	0.003 s	0.0031 s	0.0037
buscamines37	0.004 s	0.0045 s	0.0052
buscamines50	0.014 s	0.0147 s	0.0196
buscamines70	0.015 s	0.0141 s	0.012
buscamines86	0.059 s	0.0302 s	0.0298
buscamines90	0.384 s	0.2834 s	0.312
buscamines273	0.026 s	0.0210 s	0.0289
buscamines350	0.175 s	0.1808 s	0.1678

Taula 2: Resultats amb diferentes variacions.

Com podem veure els temps no han variat significativament, això és degut al fet que les instàncies amb les quals hem fet les proves no son prou grans com per que optimitzar aquests casos concrets suposi una millora significativa. Si haguéssim trobat alguna instància de mida 1000x1000 o superior, o una instància on aquests casos es donessin molt freqüentment, probablement hauríem pogut veure amb més claredat el guany de temps que suposa cada restricció.