IA: Pràctica 3 - Sudoku Sat Solver

Sergi Sisó Godia

Enginyeria Tècnica Informàtica Sistemes
Escola Potitècnica Superior
Universitat de Lleida

15 de febrer de 2011

Índex

1 Traducció del problema a SAT

1.1 Exercici 1: Conjunt de restriccions

- Varibles fixades:

$$\phi = \bigwedge \{x_{ijk} | Sudoku_{i,j} = k\}$$

- Restriccions de cel·a. A cada cel·la sol hi pot haver un número:

$$X = \{x_{i,j,k} | i \in Files, j \in Columnes, 1 \leq k \leq N\}$$

$$\phi = \bigwedge_{1 \leq i,j \leq N} exactlyone(X)$$

- Restriccions de fila. A cada fila un número sol es pot repetir una vegada:

$$X = \{x_{i,j,k} | i \in Files, k \in Numeros, 1 \leq j \leq N\}$$

$$\phi = \bigwedge_{1 \leq i,k \leq N} exactlyone(X)$$

- Restriccions de columna. A cada columna un número sol es pot repetir una vegada:

$$X = \{x_{i,j,k} | j \in Columnes, k \in Numeros, 1 \leq i \leq N\}$$

$$\phi = \bigwedge_{1 \leq j,k \leq N} exactlyone(X)$$

- Restriccions de regió. A cada regió un número sol es pot repetir una vegada:

$$X = \{x_{i,j,k} | k \in Numeros, i = i_{ini} + i_{inc}, j = j_{ini} + j_{inc}, 1 \leq i_{inc}, j_{inc} \leq n\}$$

$$\phi = \bigwedge_{i_{ini}, j_{ini} \in regio} exactlyone(X)$$

1.2 Exercici 2: Traducció a clausules

La primera traducció té una complexitat quadràtica, es basa en l'igualtat $exactlyone(X) = atleastone(X) \wedge atmostone(X)$ on:

- $atleastone(X) = (x_1 \vee ... \vee x_n) \longrightarrow \theta(n)$
- $atmostone(X) = \bigwedge_{1 \leq i,j \leq |X|, i <> j} (\neg x_i, \neg x_j) \longrightarrow \theta(n^2)$

1.3 Exercici 3: Traducció amb menys clausules

Traducció "Regular Encoding", afegirem n - 1 noves variables per cada exactlyone(), però fem que el número de clausules que aquest generi tingui un creixement linial $\theta(n)$.Per tant, per cada restricció $exactlyone(X)$ tenim:

- Noves variables adicionals $R = \{r_2, ..., r_n\}$
- Les clausules sobre R: $\phi = \bigwedge_{3 \leq i \leq n}(r_i \rightarrow r_{i-1})$
- Clausules sobre X i R:
  - $(b_1 \leftrightarrow \neg r_2)$
  - $(b_i \leftrightarrow r_i \wedge \neg r_{n+1})$
  - $(b_n \leftrightarrow r_n)$

Llavors cada $\rightarrow, \leftrightarrow$ s'ha de transformar amb el seu equivalent en CNF.

Una segona manera de traduir l'$exactlyone()$ seria amb nodes intermitjos, aquesta amb cost logarítmic. Si suposem que tenim la restricció $exaclyone(b_1, b_2, ...b_n)$.

1. Primer dividim la restricció anterior en dos afegint un node auxiliar en cada part. De forma que ens queda $exaclyone(b_1, ..., b_{n/2}, Aux_1)$ i $exaclyone(b_{(n/2)+1}, ..., b_n, Aux_2)$
2. Llavors relacionem els 2 nodes auxiliars de la forma: $Aux_1 \leftrightarrow \neg Aux_2$

1.4 Exercici 4: Traducció amb menys variables

Per fer una traducció amb menys variables el que he fet es mirar quines variables no hem són útils per tal d'eliminar-les. Així totes les casselles amb un número prefixat generen un conjunt de variables amb valor "verdader", i s'en pot deduir un altre conjunt amb valor "fals". Aquests 2 conjunts de variables no cal que el SatSolver les tingui amb conte.
Per tant, per cada casella (i,j) fixada amb el numero k afegirem a:

- $Prefixed_{true} = x_{i,j,k}$
- $Prefixed_{false} = sameCell + sameRow + sameCol + sameBlock$

Llavors a l'hora de fer les restriccións tindrem en compte de no possar les variables que ja estan prefixades ni les restriccións de cel·la a aquelles que tinguin ja un numero assignat.

En aquesta traducció haurem de guardar el mapejat de les variables $x_{i,j,k}$ a les noves variables per tal de poder fer despres la decodificació del resultat.

1.5 Exercici 5: Coincidencies entre la primera i última regió $\leq K$

Per afegir aquesta restricció podem fer el seguent:

1. Crear un conjunt de variables auxiliars que indiquin coincidencies de un numero entre dues cel·les de la mateixa possició en les regions primera i última. (rlr = row last region i clr = col last region)

$$\forall i, j, k \in region1 : X_{i,j,k} \wedge X_{rlr+i,clr+j,k} \rightarrow Aux_{cell \in region,k}$$

2. Fer un $atmostk()$ d'aquest conjunt de variables auxiliars. Com que ja tenim els restriccions de cel·la sabem que les variables auxiliars que puguin ser certes en aquest $atmostk()$ ja seràn de diferents cel·les.

$$\phi = atmostk(Aux)$$

1.6 Implementació

- Finalment la implementació de l'exactlyone() l'he fet híbrida entre les 2, ja que per les restriccions de fins a 7 variables la codificació normal hem genera menys clausules i no hem cal afegir variables adicionals. És a partir de les 8 variables on el regular encoding és la millor opció. A més l'Optimized Encoding que he implementat varia en cada restricció.

- En l'Optimitzed Encoding no he tret les variables que estan prefixades a çert"del problema. Això tot hi que m'augmenta el nombre de variables finals ho he fet perque hem facilita despres de decodificació del resultat del SatSolver. A la primera taula de l'evaluació experimental el número de clausules i variables encara podria ser menor tinguen en compte això.

- Les opcions -test i -solve del sudokusolver.py fan una crida al SatSolver Glucose 1.0 per tal de solucionar la formula CNF. Per a que funcioni correctament el Satsolver ha d'esta compilat i en ./glucose1.0/glucosestatic

2 Evaluació Experimental

Evaluació realitzada amb un Intel P4 de 3GHz i 1GB de Memoria RAM. Aquestes proves es basen en buscar una solució per a cada problema de sudoku. Si vulguessim obtenir totes les solucions possibles hauriem d'executar el SatSolver afegint les solucions trovades de forma negada al arxiu cnf. Això ho repetiriem fins que ens retornés Insatisfactible.

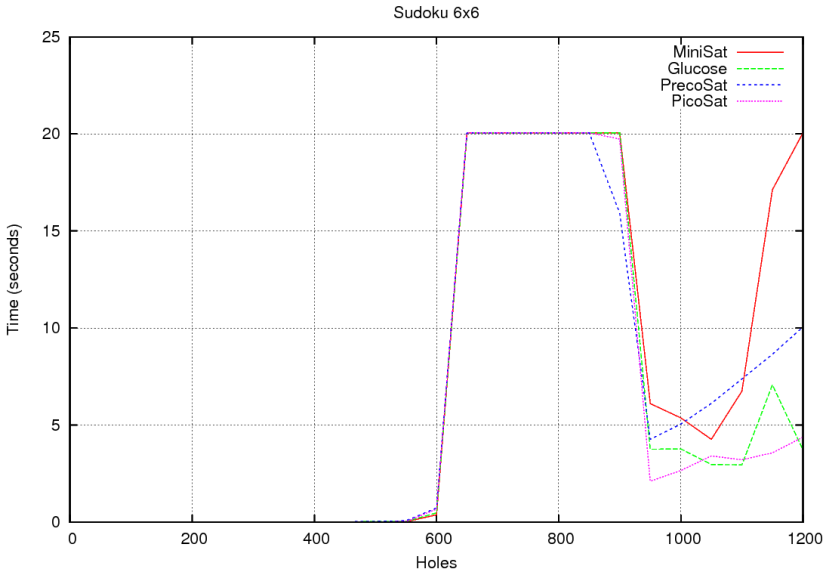
2.1 Clausules i Variables de les diferents traduccions

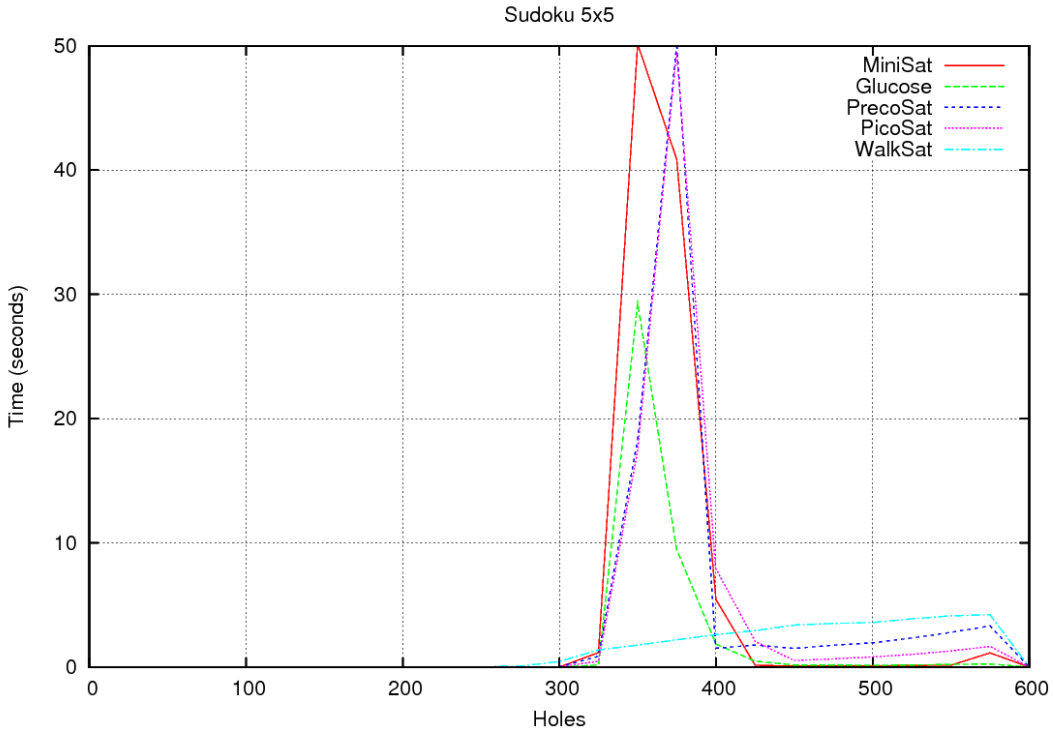
Taula amb el número de variables i clausules que generen les diferents codificacións del problema. Tots els sudokus d'aquesta prova tenen un 50% de forats.

	Extended encoding		Extended encoding (Regular Encoding)		Optimized encoding (Hybrid Enconding)	
Sudoku	Variables	Clausules	Variables	Clausules	Variables	Clausules
3 x 3	729	12029	3321	10409	123	545
4 x 4	4096	124032	19456	61568	643	3975
5 x 5	15625	752813	75625	240313	2417	14960
6 x 6	46656	3271752	228096	726408	10725	49945
7 x 7	0	0	578641	1845169	34459	130803
8 x 8	0	0	1294336	4130814	90270	304109
9 x 9	0	0	0	0	173188	565292

En aquesta taula es pot observar com amb el regular encoding, augmentant notablement el nombre de variables fem que les clausules augmentin linealment amb la mida del sudoku, cosa que ens permet resoldre sudokus més grans.
Mitjançant el Optimitzed Encoding aconseguim reduir dràsticament el nombre de variables i clausules. Això ens aporta un bon increment en la ràpidesa dels satsolvers. Però aquest increment no es proporcional amb el descens de les clausules i variables, ja que realment el que estem fent es treurè aquelles que són fàcils(immediates) de deduir.

2.2 Temps de resolució dels SatSolvers





En aquestes dues gràfiques de la resolució de sudokus de 5x5(25x25) amb timeout de 50 segons i 6x6(36x36) amb timeout de 20 segons variant el número de forats i amb diferents SatSolvers es poden observar diferents comportaments.

1. L'eficiència/rapidessa del SatSolver varia notablement amb el número de forats del problema. Es pot obervar que des d'un 0% a un 50% els SatSolvers ho resolent amb molta facilitat. Llavors del 52% al 65% hi ha un pic molt elevat de dificultat. A partir del 65% de forats torna a baixar a un nivell normal, el qual es va apujant fins casi al final. En l'últim 5% es torna a reduir la dificultat del problema.

2. També es pot observar que els SatSolvers incomplets (WalkSat) són capaços d'evitar aquest pic. Però els costa una mica més la part final del problema. A més no soporta problemes de mida gairè més gran que el 5x5.

3. Dins del SatSolvers complets, el "Glucose"és el que treu més bons resultats en el problema del Sudoku. Fet que es constata sobretot en el pic del 5x5, el qual es l'únic SatSolver complet que aconsegueix resoldrel amb un temps màxim de 30 segons.

Bibliografia

[1] Carlos Ansótegui, *Satisfiability*, May 2010

[2] Gihwon Kwon, Himanshu Jain, *Optimized CNF Encoding for Sudoku Puzzles*

[3] Will Klieber, Gihwon Kwon, *SAT Encoding For Sudoku Puzzles*