Progetto di Software per Sistemi Embedded
Graph and SAT problem coloring

Fuser Alessandro - VR405372

21 novembre 2017

Indice

1 Obiettivo del Progetto

Gli obiettivi del progetto assegnato sono:

1. Definire una funzione per leggere dei grafi da file, nel formato standard DIMACS;
2. Usare Espresso per risolvere il problema della colorazione di un grafo;
3. Usare un SAT solver per risolvere il problema della soddisficibilità booleana in relazione alla colorazione di un grafo;
4. Confronto di prestazioni dei due metodi.

2 Background

2.1 Problema della Colorazione di un grafo

Un grafo è un insieme di elementi detti nodi o vertici che possono essere collegati fra loro da linee chiamate archi o lati o spigoli. Più formalmente, si dice grafo una coppia ordinata G = (V, E) di insiemi, con V insieme dei nodi ed E insieme degli archi, tali che gli elementi di E siano coppie di elementi di V.
Il problema della colorazione di un grafo può essere visto come un problema di etichettatura dei vertici del grafo, nel quale è richiesto di assegnare un colore ad ogni vertice, con la condizione che vertici tra loro connessi non abbiano lo stesso colore assegnato. La k-colorabilità si tratta di trovare un numero k il più piccolo possibile tale per cui il grafo possa essere colorato senza violare il vincolo del problema.
Una volta trovato il numero di colori k, il problema richiede l'assegnamento di un colore c_i con i <= k per ogni vertice v_j con j <= $|V|$.

2.2 Problema della soddisfacibilità booleana

Una formula è in forma normale congiuntiva o congiunta (FNC), indicata anche come CNF (acronimo di Conjunctive Normal Form) se è una congiunzione di clausole, dove le clausole sono una disgiunzione di letterali. Una formula in CNF ha quindi la seguente struttura:

$$\bigwedge_{i=1}^{n} \left(\bigvee_{k=1}^{m(i)} L_{i,k} \right)$$

nel quale:

- n è il numero di clausole;
- m(i) è il numero di letterali della clausola i-esima;

- $L_{i,k}$ è il k-esimo letterale della i-esima clausola.

Un letterale può essere una variabile booleana (cioè può valere solo 0 o 1, ossia vero o falso) o la negazione di una variabile.
Formalmente, la soddisfacibilità booleana, o soddisfacibilità proposizionale o SAT, è il problema di determinare se una formula booleana è soddisfacibile o insoddisfacibile. La formula si dice soddisfacibile se le variabili possono essere assegnate in modo che la formula assuma il valore di verità vero. Viceversa, si dice insoddisfacibile se tale assegnamento non esiste (pertanto, la funzione espressa dalla formula è identicamente falsa).

2.3 Da k-graph a k-SAT

Ma come si passa da un problema di colorabilità di un grafo ad un problema di soddisfacibilità booleana?
Una volta che si è trovato un numero k tale per cui il grafo è colorabile, la conversione viene effettuata seguendo tali regole:

1. Per tutti i vertici, si crea la possibilità che abbia un colore da 1 a k;
2. Per tutti i vertici, si nega la possibilità che lo stesso vertice possa avere più di un colore;
3. In funzione degli archi, mi assicuro che i vertici toccati da due archi non abbiano mai lo stesso colore.

$$\bigvee_{1 \le i \le k} p_{vi} \qquad (v \in V),$$

$$\neg(p_{vi} \wedge p_{vj}) \qquad (v \in V,\ 1 \le i < j \le k),$$

$$\neg(p_{vi} \wedge p_{wi}) \qquad (\{v, w\} \in E,\ 1 \le i \le k).$$

Figura 1: Regole riduzione da k-Color a SAT (descritte sopra)

3 Formato DIMACS

Il formato scelto per i grafi è lo standard DIMACS edge, nel quale:

- Le righe che iniziano con "c" sono dei commenti per spiegare il grafo;
- La riga che inizia con "p" indica il numero di nodi e di archi;
- Le righe che iniziano con "e" indicano il collegamento tra due vertici.

Esempio di grafo DIMACS:

```
c FILE: myciel3.col
c SOURCE: Michael Trick (trick@cmu.edu)
c DESCRIPTION: Graph based on Mycielski transformation.
c              Triangle free (clique number 2) but increasing
c              coloring number
p edge 11 20
e 1 2
e 1 4
e 1 7
e 1 9
e 2 3
e 2 6
e 2 8
e 3 5
e 3 7
e 3 10
e 4 5
e 4 6
e 4 10
e 5 8
e 5 9
e 6 11
e 7 11
e 8 11
e 9 11
e 10 11
```

Il formato scelto per la rappresentazione delle formule CNF è lo standard DIMACS cnf, dove:

- La prima riga, che comincia con "p", indica che il formato è in CNF ed il numero di variabili e clausole;
- Le righe successive indicano le clausole ed ogni riga termina uno zero; un numero rappresenta una variabile e, se preceduta da un -, allora tale variabile, all'interno della clausola, è negata.

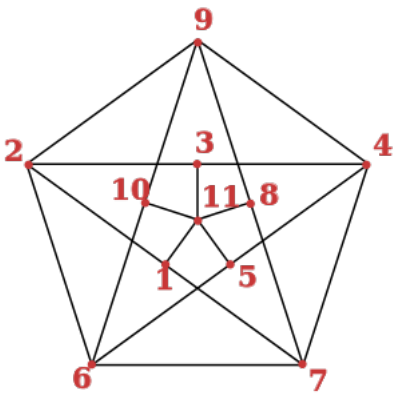

Figura 2: Rappresentazione del grafo myciel3

4 Colorabilità del grafo tramite espresso

Dato un generico grafo G(V, E), con V l'insieme dei nodi e E l'insieme dei vertici, costruiamo in funzione delle sottoregioni presenti nel grafo, un PLA nel modo seguente:

- Il numero delle variabili di input è uguale al numero dei nodi $|V|$ presenti nel grafo;
- Restituiamo in output 1 bit che indicherà la colorabilità della sottoregione;
- Costruiamo il PLA di tipo fr, pertanto il significato dei termini è il seguente:
  - con il termine 1, indichiamo che il prodotto dei termini appartenenza all'insieme ON-Set;
  - con il termine 0, indichiamo che il prodotto dei termini appartenenza all'insieme OFF-Set;
  - con il termine -, indichiamo l'insieme DC-Set, il quale rappresenta il complemento dell'unione tra ON-Set e OFF-Set;
- Definiamo l'insieme degli ON-Set, come $|V|$ righe che rappresentano le aree della mappa che possono avere il medesimo colore;
- Definiamo l'insieme degli OFF-Set, pari al numero degli archi contenuti in E, come l'insieme delle regioni che devono avere colore differenti.

Esempio di trasformazione, prendendo il grafo presentato prima:

```
.i 11
.o 1
```

```
.type fr
-0000000000 1
0-000000000 1
00-00000000 1
000-0000000 1
0000-000000 1
00000-00000 1
000000-0000 1
0000000-000 1
00000000-00 1
000000000-0 1
0000000000- 1
11--------- 0
1--1------- 0
1-----1---- 0
1-------1-- 0
-11-------- 0
-1---1----- 0
-1-----1--- 0
--1-1------ 0
--1---1---- 0
--1------1- 0
---11------ 0
---1-1----- 0
---1-----1- 0
----1--1--- 0
----1---1-- 0
-----1----1 0
------1---1 0
-------1--1 0
--------1-1 0
---------11 0
.end
```

4.1 Minimizzazione tramite espresso

Passiamo l'input generato a espresso il quale, mediante l'algoritmo Expand Reduce, si preoccuperà di minimizzare la funzione in input F, restituendo l'insieme dei cubi che rappresentano gli implicanti primi che ricoprono F.
Più in dettaglio, l'algoritmo di espresso:

1. **Espansione**: un cubo viene espanso fino a che non è primo (non include vertici dell'offset); questo comporta fare il complemento di ogni input, a turno, per verificare se il nuovo vertice è un membro dell'off-set o dell'on-set. Alla fine del processo di espansione, si ha un copertura

prima della funzione tale che nessun cubo primo contiene un altro cubo primo;

2. **Copertura Irridondante**: i primi vengono classificati come cubi "essenziali relativi" se non possono essere omessi senza distruggere la proprietà di copertura della funzione e, invece, classificati come cubi "ridondanti" se possono essere rimossi senza distruggere tale proprietà. Quest'ultio insieme può essere a sua volta partizionato in "totalmente ridondanti" e "parzialmente ridondanti" ed una procedura di scelta di minimi irridondanti cerca di prendere una copertura minimale;

3. **Riduzione**: le procedure precedenti danno una soluzione localmente ottima che potrebbe anche non essere quella globalmente ottima (dato che il problema è NPC). La riduzione trasforma un copertura prima in una nuova copertura sostituendo ogni cubo con un cubo più piccolo contenuto in esso.

La procedura è iterata fino a che non ho ulteriori miglioramenti nella minimizzazione della funzione e può essere visualizzata nella seguente figura:

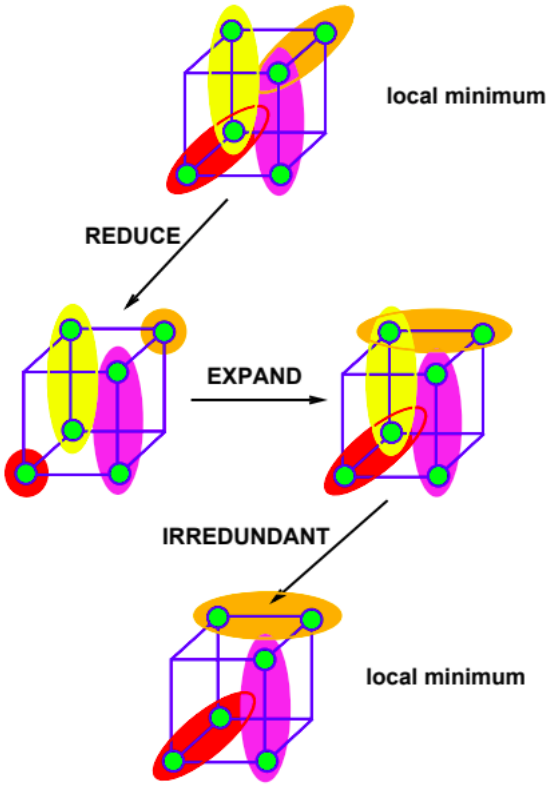

Figura 3: Algoritmo illustrato di Espresso

La minimizzazione mediante espresso restituirà n termini, indicati nella clausola ".p n", il quale rappresenta il numero di colori necessario per colorare il grafo di input secondo tale minimizzazione.
L'insieme dei DC contenuto in ogni termine, rappresenterà una sotto regione dello spazio indipendente dal grafo,che può avere lo stesso colore. Pertanto, per ogni termine prodotto da espresso, leggiamo quali sono le variabili che hanno il termine Don't-Care, $0 - 0$, e, alle variabili associate a cui non è stato ancora assegnato un colore, procediamo assegnando il primo colore disponibile.
Per il grafo precedente quindi:

```
.i 11
.o 1
.p 4
00000-----0 1
0000---00-0 1
-0-0000000- 1
0-0-00-0-00 1
.e
```

Al quale corrisponde la seguente colorazione:

```
A6 assegno colore 1
A7 assegno colore 1
A8 assegno colore 1
A9 assegno colore 1
A10 assegno colore 1
A5 assegno colore 2
A1 assegno colore 3
A3 assegno colore 3
A11 assegno colore 3
A2 assegno colore 4
A4 assegno colore 4
```

5 Soddisfacibilità CNF

5.1 Algoritmo DPLL

DPLL (Davis, Putnam, Logemann e Loveland) è un algoritmo che mediante la tecnica del backtracking, permette di risolvere il problema della soddisfacità di un espressione booleana in forma normale congiunta (CNF-SAT). L'algoritmo, ipotizza e assegna un valore di verità a un letterale (Vero o falso), e propaga tale valore in tutte le clausole ed in maniera ricorsiva verifica ogni volta se la formula è soddisfacibile o meno, in caso negativo si complementa il valore di verità che si era ipotizzato. Una volta ipotizzato vero un valore di verità, è possibile rimuovere tutte le clausole dove è presente il valore vero, al contrario se il valore è falso è possibile rimuovere il letterale dalla clausola. L'algoritmo ad ogni passo esegue due iterazioni:

- verifica della presenza di clausole unarie: se vi sono clausole dove è rimasto solo un letterale, l'unico modo per avere la formula vera è quello di assegnare il letterale rimasto a vero e propagarlo;
- verifica della presenza di letterali puri: se una variabile appare solo positiva o solo negativa, viene definita pura. Avendo una sola possibilità di assegnamento. L'algoritmo semplifica l'equazione rimuovendo tutte le clausole che contengono i letterali puri.

5.2 Conversione e soluzione

Dato il grafo generico G=(V,E), è possibile effettuare la conversione in formato DIMACS cnf, descritto sopra, tramite le regole presentate. Ma come trovo il numero di colori per effettuare tale conversione? Due sono gli approcci presentati, entrambi basati sul problema Clique, ossia un insieme di vertici di V tale esiste un arco tra tutti i vertici dell'insieme, per cui, nel nostro problema, corrisponde al limite minimo di colori in quanto vertici collegati devono avere colori diversi:

1. Algoritmo di ricerca binaria, che comincia usando un k massimo (=$|V|$), un minimo dato dalla dimensione massima della clique e, ogni volta che la formula è soddisfacibile con tale k, ripone il problema con un k dimezzato, fino a che la formula non è più soddisfacibile ed allora aumenta k di un valore a metà tra l'ultimo soddisfacibile e quello attuale;
2. sfruttando il problema della Clique, trovo quella più grande, che mi rappresenta il lower bound di colori e da questo aumento k di 1 fino a che la formula non è soddisfacibile.

Il metodo che è stato utilizzato per la conversione è il secondo, in quanto si è dimostrato più veloce in tutte le situazioni. Una volta scelto un k, la conversione viene effettuata nel formato DIMACS cnf, presentato precedentemente. Per l'esempio presentato prima, la conversione con 4 colori è:

```
p cnf 44 91
1 2 3 4 0
5 6 7 8 0
9 10 11 12 0
13 14 15 16 0
17 18 19 20 0
21 22 23 24 0
25 26 27 28 0
29 30 31 32 0
33 34 35 36 0
37 38 39 40 0
41 42 43 44 0
-1 -5 0
-2 -6 0
-3 -7 0
-4 -8 0
-1 -13 0
-2 -14 0
-3 -15 0
-4 -16 0
-1 -25 0
-2 -26 0
-3 -27 0
-4 -28 0
-1 -33 0
-2 -34 0
-3 -35 0
-4 -36 0
-5 -9 0
-6 -10 0
-7 -11 0
-8 -12 0
-5 -21 0
-6 -22 0
-7 -23 0
-8 -24 0
-5 -29 0
-6 -30 0
-7 -31 0
```

```
-8 -32 0
-9 -17 0
-10 -18 0
-11 -19 0
-12 -20 0
-9 -25 0
-10 -26 0
-11 -27 0
-12 -28 0
-9 -37 0
-10 -38 0
-11 -39 0
-12 -40 0
-13 -17 0
-14 -18 0
-15 -19 0
-16 -20 0
-13 -21 0
-14 -22 0
-15 -23 0
-16 -24 0
-13 -37 0
-14 -38 0
-15 -39 0
-16 -40 0
-17 -29 0
-18 -30 0
-19 -31 0
-20 -32 0
-17 -33 0
-18 -34 0
-19 -35 0
-20 -36 0
-21 -41 0
-22 -42 0
-23 -43 0
-24 -44 0
-25 -41 0
-26 -42 0
-27 -43 0
-28 -44 0
-29 -41 0
-30 -42 0
-31 -43 0
```

```
-32 -44 0
-33 -41 0
-34 -42 0
-35 -43 0
-36 -44 0
-37 -41 0
-38 -42 0
-39 -43 0
-40 -44 0
```

La risoluzione tramite MINISAT porta al seguente assegnamento:
SAT -1 -2 -3 4 5 -6 -7 -8 -9 -10 11 -12 -13 -14 15 -16 17 -18 -19 -20 -21 22 -23 -24 -25 26 -27 -28 -29 30 -31 -32 -33 34 -35 -36 -37 38 -39 -40 41 -42 -43 -44 0
Dove SAT indica che la formula è soddisfacibile tramite l'assegnamento delle variabili indicato successivamente, dove se ho un "-" allora la variabile è posta a FALSE, altrimenti a TRUE e lo zero indica la fine del file.

6 Prestazioni

Tutte le prove sono state fatto su una macchina portatile con un i7-4710HQ e 16GB di RAM.
Il confronto è stato fatto su una serie di grafi presi dal sito Graph Coloring Benchmarks. Sono stati relazionati i tempi relativi alla soluzione con Espresso, con Minisat [2] (solutore per SAT), DPLL [3](solutore per SAT) e i risultati della [1]. Per ogni problema, è stato impostato un timer di 30 minuti in quanto eravamo interessati ad una risoluzione veloce del problema. L'unica eccezione viene data dal problema **queen8**, che viene preso come esempio principale della difficoltà computazionale. Le tempistiche sono riassunte nella tabella successiva, dove:

- La prima colonna riporta il nome del problema;
- La seconda colonna riporta il numero di vertici del grafo;
- La terza colonna riporta il numero di archi del grafo;
- La quarta colonna riporta la densità del grafo;
- La quarta colonna riporta il minimo della dimensione della massima clique;
- La quinta colonna riporta la dimensione della massima clique;
- La sesta colonna riporta il numero di cromatico migliore trovato in letteratura;
- La settima ed ottava colonna riportano il numero cromatico ed il tempo con Espresso;
- La nona e decima colonna riportano il numero cromatico ed il tempo con Minisat;
- L'undicesima colonna riporta il tempo riferito nella [1];
- Le ultime due colonne riportano il numero cromatico ed il tempo con Minisat usando il primo metodo di scelta del coloro per cnf.

	|V|	|E|	d%	w_LB(G)	w_(G)	X(G)	ESPRESSO	TIME	MINISAT	TIME	TESI	MINISAT-2	TIME
myciel3	11	20	0,36	2	2	4	4	0,14	4	0,015	0,006	4	0,017
myciel4	23	71	0,28	2	2	5	5	0,21	5	0,035	0,06	5	0,037
queen5_5	25	160	0,53	5	5	5	5	0,19	5	0,004	0,02	5	0,02
1-FullIns_3	30	100	0,23	3	3	4	4	0,23	4	0,0089	0,013	4	0,02
queen6_6	36	290	0,46	6	6	7	7	2,34	7	1,12	1,87	7	1,13
2-Insertions_3	37	72	0,11	2	2	4	4	0,36	4	0,022	0,03	4	0,03
myciel5	47	236	0,22	2	2	6	6	0,37	6	65,16	256,66	6	65,38
queen7_7	49	476	0,4	7	7	7	7	0,44	7	0,008	0,06	7	0,05
2-FullIns_3	52	201	0,15	4	4	5	5	1,16	5	0,009	0,024	5	0,028
3-Insertions_3	56	110	0,07	2	2	4	4	198,63	4	0,08	0,19	4	0,092
queen8_8	64	728	0,36	8	8	9		>24h			>24h		
1-Insertions_4	67	232	0,1	2	2	5	5	15,36	5	137,86	238,12	5	138,08
huck	74	301	0,11	11	11	11		>30m	11	0,0075	125,67	11	0,054
jean	80	254	0,08	10	10	10		>30m	10	0,0057	5,01	10	0,44
queen9_9	81	1056	0,33	9	9	10		>1h					
david	87	406	0,11	11	11	11		>30m	11	0,0077	66,11	11	0,06
mug88_1	88	146	0,04	3	3	4		>30m	4	0,011		4	0,028
mug88_25	88	146	0,04	3	3	4		>30m	4	0,012		4	0,03
queen8_12	96	1368	0,3	12	12	12		>30m	12	0,026		12	0,18
games120	120	638	0,09	9	9	9		>30m	9	0,009		9	0,11
r125.1	125	209	0,03	5	5	5		>30m	5	0,0049		5	0,05
DSJC125.1	125	736	0,09	4	4	5		>30m	5	0,076	0,3	5	0,14
r125.5	125	3838	0,5	36	36	36		>1h	36	3,96		36	4,34
r125.1c	125	7501	0,97	46	46	46	47	5,45		>30m			
miles250	128	387	0,05	8	8	8		>30m	8	0,008		8	0,08
miles500	128	1170	0,14	20	20	20		>30m	20	0,026		20	0,24
miles750	128	2113	0,26	31	31	31		>30m	31	0,1		31	0,53
miles1000	128	3216	0,4	42	42	42		>30m	42	0,012		42	0,74
miles1500	128	5198	0,64	73	73	73		>30m	73	0,91		73	2,63

anna	138	493	0,05	11	11	11		>30m	11	0,0089	78,74	11	0,1
mulsol.i.3	184	3916	0,23	31	31	31		>30m	31	0,055		31	0,51
mulsol.i.4	185	3946	0,23	31	31	31		>30m	31	0,06		31	0,54
mulsol.i.5	186	3973	0,23	31	31	31		>30m	31	0,1		31	0,54
mulsol.i.2	188	3885	0,22	31	31	31		>30m	31	0,15		31	0,51
mulsol.i.1	197	3925	0,2	49	49	49		>30m	49	0,15		49	0,67
zeroin.i.3	206	3540	0,17	30	30	30		>30m	30	0,08		30	0,52
zeroin.i.2	211	3541	0,16	30	30	30		>30m	30	0,07		30	0,54
zeroin.i.1	211	4100	0,19	49	49	49	49	121,38	49	0,12		49	0,76
r250.1	250	867	0,03	8	8	8		>30m	8	0,007		8	0,15
r250.1c	250	30227	0,97	63	63	64		>30m		>30m			
school1	385	19095	0,26	14	14	14		>30m	14	0,17		14	3,03
fpsol2.i.3	425	8688	0,1	30	30	30		>30m		>30m			
le450_5d	450	9757	0,1	5	5	5		>30m	5	0,05	8,28	5	2,1
le450_5c	450	9803	0,1	5	5	5		>30m	5	0,075	5,43	5	1,72
ash331GPIA	662	4181	0,02	3	3	4		>30m	4	0,063	4,5	4	1,68
will199GPIA	701	6772	0,03	6	6	7		>30m	7	0,07	6,41	7	1,97
inithx.i.1	864	18707	0,05	54	54	54		>30m	54	0,34		54	8,25
r1000.1	1000	14378	0,03	20	20	20		>30m	20	2,63		20	10,35
4-Insertions_3	79	156	0,05	2		4		>30m	4	0,61	1,53	4	0,63
3-FullIns_3	80	346	0,11	5		6		>30m	6	0,01	0,054	6	0,04
1-FullIns_4	93	593	0,14	3		5		>30m	5	0,02	0,089	5	0,058
myciel6	95	755	0,17	2		7		>30m		>30m			
mug100_1	100	166	0,03	3		4		>30m	4	0,01	0,06	4	0,38
mug100_25	100	166	0,03	3		4		>30m	4	0,01	0,05	4	0,04
4-FullIns_3	114	541	0,08	6		7		>30m	7	0,03	0,147	7	0,08
5-FullIns_3	154	792	0,07	7		8		>30m	8	0,23	0,77	8	0,31
2-FullIns_4	212	1621	0,07	4		6		>30m	6	0,05	0,52	6	0,21
school1_nsh	352	14612	0,24	14		14		>30m	14	0,08		14	2,37
le450_15a	450	8168	0,08	15		15		>30m	15	0,9		15	3,08
wap05a	905	43081	0,11	50		50		>30m	50	1,99		50	20,86

6.1 Commenti

Come si può notare, in tutti i problemi la risoluzione tramite MINISAT è molto più rapida rispetto ad Espresso. Questo potrebbe essere dovuto alla complessità computazionale della minimizzazione di espresso, che può essere molto dispendiosa sia in termini di tempo che di risorse, dato che le 3 funzioni interne di espresso portano a colorare il grafo in funzione degli implicanti primi essenziali della funzione di partenza (complessi da calcolare).
E' interessante notare però come il problema queen8 (così come il problema queen9) non è stato risolto nè da espresso nè da MINISAT in termini ragionevoli (24 ore), il che ci fa capire che problemi difficili sono tali sia per espresso che per MINISAT.
Rispetto al DPLL, MINISAT è più veloce in quanto si basa sullo stesso algoritmo usato per DPLL, ma aggiungendo molte ottimizzazioni, come si legge in [2]. DPLL è stato abbandonato dopo pochi problemi in quanto riportano sempre tempi maggiori rispetto a MINISAT, per cui ci si è concentrati sul migliore risolutore per SAT.
punto di partenza della scelta dei colori è più efficace rispetto ad usare un approccio binario per la ricerca dei colori e la clique come lower bound. Questo è dovuto al fatto che, nella maggior parte dei casi, il valore della clique più grande è molto vicino al numero cromatico del grafo, per cui le iterazioni sono minori.

7 Futuri Lavori

Per futuri lavori, sarebbe interessante provare altri solutori SAT in modo da vedere se anche i problemi che non vengono risolti usando i solutori presentati vengono decisi.
Un altro aspetto interessante sarebbe quello di capire se la non risoluzione è dovuta alla semplice mancanza di sufficiente potenza di calcolo, per cui accedere ad un computer dalle prestazioni superiori.
Un'ulteriore tema che si può affrontare è la multi-colorazione dei grafi, per cui estendere gli attuali programmi e provare su altri grafi.

Riferimenti bibliografici

[1] Stefan Kugele, *Efficient Solving of Combinatorial Problems using SAT-Solvers*

[2] http://minisat.se/

[3] https://it.wikipedia.org/wiki/DPLL