

May 19, 2018 at 02:31

1* Intro. This little program outputs clauses that are satisfiable if and only if the graph g can be c -colored, given g and c .

(It generalizes SAT-PIGEONS, which is the case where $g = K_m$ and $c = n$.)

Suppose the graph has m edges and n vertices. Then there are nc variables $v.k$, meaning that vertex v gets color k . And there are n clauses of size c (to ensure that each vertex gets at least one color), plus mc clauses of size 2 (to ensure that adjacent vertices don't share a color).

```
#include <stdio.h>
#include <stdlib.h>
#include "gb_graph.h"
#include "gb_save.h"
int c;
int n; /* order of flower snark line graph (a command-line parameter) */
char buf[20];
main(int argc, char *argv[])
{
    register int i, j, k;
    register Arc *a;
    register Graph *g;
    register Vertex *v;
    < Process the command line 2*>;
    < Generate the positive clauses 3*>;
    < Generate the negative clauses 4*>;
}

2* < Process the command line 2*> ≡
if (argc ≠ 2 ∨ sscanf(argv[1], "%d", &n) ≠ 1) {
    fprintf(stderr, "Usage: %s\n", argv[0]);
    exit(-1);
}
sprintf(buf, "fsnarkline%d.gb", n);
g = restore_graph(buf);
if (¬g) {
    fprintf(stderr, "I couldn't reconstruct graph %s!\n", buf);
    exit(-2);
}
c = 3;
printf("~sat-color-snark2%d\n", n);
printf("b1.1\n"); /* start with three unary clauses to break symmetry */
printf("c1.2\n");
printf("d1.3\n");
```

This code is used in section 1*.

```
3. < Generate the positive clauses 3*> ≡
for (v = g-vertices; v < g-vertices + g-n; v++) {
    for (k = 1; k ≤ c; k++) printf("%s.%d", v-name, k);
    printf("\n");
}
```

This code is used in section 1*.

```

4.  ⟨ Generate the negative clauses 4 ⟩ ≡
    for ( $k = 1$ ;  $k \leq c$ ;  $k++$ )
      for ( $v = g\text{-vertices}$ ;  $v < g\text{-vertices} + g\text{-}n$ ;  $v++$ )
        for ( $a = v\text{-arcs}$ ;  $a$ ;  $a = a\text{-next}$ )
          if ( $a\text{-tip} > v$ ) printf("%s.%d_~%s.%d\n",  $v\text{-name}$ ,  $k$ ,  $a\text{-tip}\text{-name}$ ,  $k$ );

```

See also section 5*.

This code is used in section 1*.

5* In this variant we also generate negative clauses to ensure that no vertex gets more than one color. We use the fact that $c = 3$.

```

⟨ Generate the negative clauses 4 ⟩ +≡
  for ( $v = g\text{-vertices}$ ;  $v < g\text{-vertices} + g\text{-}n$ ;  $v++$ ) {
    printf("%s.1_~%s.2\n",  $v\text{-name}$ ,  $v\text{-name}$ );
    printf("%s.2_~%s.3\n",  $v\text{-name}$ ,  $v\text{-name}$ );
    printf("%s.3_~%s.1\n",  $v\text{-name}$ ,  $v\text{-name}$ );
  }

```

6* Index.

The following sections were changed by the change file: 1, 2, 5, 6.

a: 1*
Arc: 1*
arcs: 4.
argc: 1*, 2*
argv: 1*, 2*
buf: 1*, 2*
c: 1*
exit: 2*
fprintf: 2*
g: 1*
Graph: 1*
i: 1*
j: 1*
k: 1*
main: 1*
n: 1*
name: 3, 4, 5*
next: 4.
printf: 2* 3, 4, 5*
restore_graph: 2*
sprintf: 2*
sscanf: 2*
stderr: 2*
tip: 4.
v: 1*
Vertex: 1*
vertices: 3, 4, 5*

- ⟨Generate the negative clauses 4, 5*⟩ Used in section 1*.
- ⟨Generate the positive clauses 3⟩ Used in section 1*.
- ⟨Process the command line 2*⟩ Used in section 1*.

SAT-COLOR-SNARK2

	Section	Page
Intro	1	1
Index	6	3