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\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-snark5␣%d\n", n);
  ⟨ Force a bad vertex order 5* ⟩;
  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");
}
```

See also section 6*.

This code is used in section 1*.

4. ⟨ Generate the negative clauses 4 ⟩ ≡

```
for (k = 1; k ≤ c; k++)
  for (v = g→vertices; v < g→vertices + g→n; v++)
    for (a = v→arcs; a; a = a→next)
      if (a→tip > v) printf("~%s.%d␣~%s.%d\n", v→name, k, a→tip→name, k);
```

This code is used in section 1*.

5*. ⟨ Force a bad vertex order 5* ⟩ ≡

```
for (k = 1; k ≤ c; k++) {
  for (v = g→vertices; v < g→vertices + g→n; v++) printf("␣%s.%d", v→name, k);
  printf("␣~%s.%d\n", g→vertices→name, k);
}
```

This code is used in section 2*.

6*. In this variant we also generate positive clauses to ensure that every color class is a kernel.

⟨ Generate the positive clauses 3 ⟩ +≡

```
for (k = 1; k ≤ c; k++)
  for (v = g→vertices; v < g→vertices + g→n; v++) {
    printf("%s.%d", v→name, k);
    for (a = v→arcs; a; a = a→next) printf("␣%s.%d", a→tip→name, k);
    printf("\n");
  }
```

7.* Index.

The following sections were changed by the change file: 1, 2, 5, 6, 7.

SAT-COLOR-SNARK5