Overview of the Smallk Graph Coloring Program

Joseph Culberson

October 6, 2000

This is a brief description of the Smallk coloring program.

We define a list coloring problem[1, 4] as a graph $\mathrm{G} = (\mathbf{V}, \mathbf{E})$ where each vertex has a set of available colors, $\mathbf{A}(v)$, also called its availset. For our algorithm, we first formulate a k-coloring problem as a list coloring, initializing the availsets $\mathbf{A}(v) = \{1, \ldots, k\}, \forall v$.

In the description of the algorithm we will frequently need to express the fact that for every coloring c of G, for a pair of vertices x, y, either $c[x] = a$ or $c[y] = b$. We use the notation $\langle x = a \vee y = b \rangle$. A set of such clauses is maintained for every pair of vertices in the graph.

As a notational convenience, for $\mathbf{U} \subseteq \mathbf{V}$, we define $\mathbf{A}(\mathbf{U}) = \cup_{x \in \mathbf{U}} \mathbf{A}(x)$, the union of the available colors over $\mathbf{U}$. Thus, $\mathbf{A}(\mathbf{N}(v))$ is the set of available colors of the neighbors of v.

The algorithm proceeds in two stages.

Stage 1: In the first stage a recursive backtrack search reduces the graph by the deletion of edges and vertices and available colors, where each reduction step is recorded in a stack. When backtrack is forced, the stack is popped and the graph restored to its state at the previous search depth. If the graph becomes empty, the algorithm proceeds to the second stage, otherwise when no further options are available it reports failure.

Stage 2: In the second stage the stack information is used to reconstruct and color the graph, with each vertex assigned its final color as it is popped from the stack. In a final step, the coloring is verified for correctness on a copy of the original graph.

The graph reductions are generalizations and extensions of those in [2, 3]. Many of the following reductions are applied recursively at each step of the search. Some may only be applied to the initial graph due to efficiency reasons, as early experiments inicated the overhead exceeded the savings in practice.

1. If $\deg(v) < |\mathbf{A}(v)|$ then delete v.

2. If $\mathbf{A}(v) \setminus \mathbf{A}(\mathbf{N}(v)) \neq \emptyset$ then delete v.

3. If $\{u, v\} \in E$ and $\mathbf{A}(u) \cap \mathbf{A}(v) = \emptyset$ then delete edge $\{u, v\}$.

4. If x, y, z is a 3-clique and $|\mathbf{A}(\{x,y,z\})| \leq 3$ then for each w which is a neighbor of 2 out of the three vertices, with $\mathbf{A}(w) = \mathbf{A}(\{x,y,z\})$ then merge w to the remaining vertex, say u. (deleting w). Also, remove from $\mathbf{A}(u)$ any color not in $\mathbf{A}(w)$.

5. If $|\mathbf{A}(v)| = 1$ then delete v. For every $w \in \mathbf{N}(v)$, delete the 1 color in $\mathbf{A}(v)$ from $\mathbf{A}(w)$.

6. If $\langle x = a \vee y = b \rangle$ and $\langle x = b \vee y = a \rangle$ then since $c[x] = c[y]$ merge y to x (deleting y). Remove from $\mathbf{A}(x)$ any color not in $\mathbf{A}(y)$.

In addition to reduction, each time a color is assigned to a vertex there is extensive forward propagation of the consequences. The propagation hinges on the clauses of the type $\langle x = a \vee y = b \rangle$. When a vertex is colored then that color is removed from the availsets of all adjacent vertices. The following may then be used to initiate such clauses.

- If $\{x,y\} \in \mathbf{E}$ and $\mathbf{A}(x) = \{a,b\}$, $\mathbf{A}(y) = \{a,c\}$ then we have $\langle x = b \vee y = c \rangle$.
- If $\{x,y\} \in \mathbf{E}$, and $\mathbf{A}(x) = \mathbf{A}(y) = \{a,b\}$ then we have both $\langle x = a \vee y = a \rangle$ and $\langle x = b \vee y = b \rangle$.
- If x, y, z is a triangle with $\mathbf{A}(\{x,y,z\}) = \{a,b,c\}$ and $\mathbf{A}(x) = \{a,b\}$ then $\langle y = c \vee z = c \rangle$.

Whenever a new clause is generated it may then interact with other clauses to produce more clauses. Note that the following does not depend on the edges of the graph; thus we refer to it as a *universal propagator*.

- For any three distinct vertices x, y, z and any colors a, b, c, d where $b \neq c$, if we have $\langle x = a \vee y = b \rangle$ and $\langle y = c \vee z = d \rangle$ then we obtain $\langle x = a \vee z = d \rangle$.

The primary use of propagators is to remove colors from availsets which prevents incorrect choices at later stages of the search.

- If $\langle x = a \vee y = a \rangle$ then $\forall w \in \mathbf{N}(x) \cap \mathbf{N}(y), a \notin \mathbf{A}(w)$.

Occasionally we can detect forced colorings directly from clauses. Note that once a vertex is colored then that color is removed from all its neighbors and we no longer need to consider the vertex as part of any clause.

- Forced Coloring: If $\langle x = a \vee y = b \rangle$ and $\langle x = a \vee y = c \rangle$ with $c \neq b$ then $\mathbf{A}(x) = \{a\}$.

The following indicates that we can limit the number of clauses to at most two per vertex pair. This is important for implementation, and it also gives us early detection of partial assignments that cannot be extended to a full coloring.

- If there are three pairwise non-equivalent relations of this form on a pair of vertices x and y, then either the coloring is inconsistent, or at least one of the vertices x or y has a forced coloring.

For each color choice, these rules are iterated until none apply or the current coloring is detected as inconsistent. The algorithm also uses dynamic variable ordering with heuristics based on degree and availsets.

The second stage, which constructs and verifies the coloring, is fairly straightforward and runs in $O(n^2)$ time.

References

[1] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3446^n)$: a no-MIS algorithm. In *36th IEEE Symposium on Foundations of Computer Science*, pages 444–452, 1995.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of the 12th IJCAI*, pages 331–337, 1991.

[3] F. Glover, M. Parker, and J. Ryan. Coloring by TABU branch and bound. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 285–307. American Mathematical Society, 1996.

[4] T. R. Jensen and B. Toft. *Graph Coloring Problems.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, Inc., 1995.