BDD Techniques for Graph Coloring and Related Problems

Steve Haynal

Department of Electrical and Computer Engineering
University of California, Santa Barbara

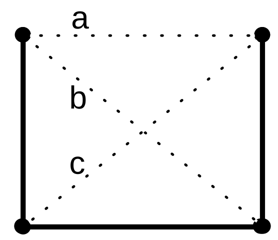
March 1997

Outline

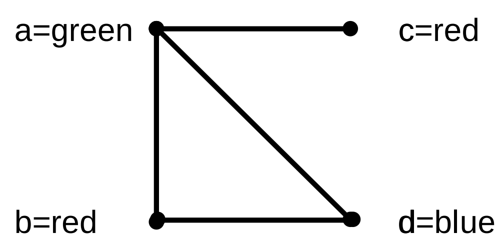
- **Problem Statements**
- **Why Solve These Problems?**
- **General Approach to Solutions**
- **Implementation Specifics**
- **Results**
- **Further Research Directions**
- **Conclusions**

Problem Statements

- **Original Problem**
  - **Find edges, which if added to an existing undirected graph, form odd (even) circuits**
    - Original graph contains no odd circuits
    - Edges *b* and *c* would form odd circuits
    - Edge *a* would form an even circuit



- **Evolved Problems**
  - **Find the chromatic number of an undirected graph**
    - Chromatic number = 3
  - **Find all valid colorings of an undirected graph**
    - c could be blue
    - a could be red.......

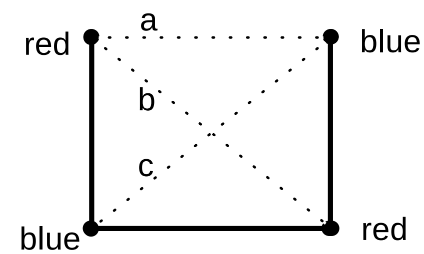


Why Solve These Problems?

- **Coloring Solves General Scheduling**
  - **Committee scheduling**
  - **Engineering examples**
    - Minimum width channel routing
    - Chip register/resource scheduling
    - Assignment of binary codes to state symbols
- **Coloring is NP-complete - Challenging!**
  - **Set and compression properties of BDDs could be useful...**

General Approach to Solutions: Finding Possible Odd (Even) Circuit Causing Edges

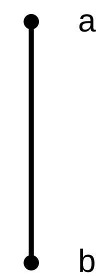
- **Theorem: A graph can be 2-colored if and only if it does not contain a circuit of odd length**
  - Any added edge that destroys 2-coloredness forms an odd circuit
  - Any added edge that preserves 2-coloredness and connects two vertices in the same graph forms an even circuit



- **Solution:**
  - Two-color the graph
  - For odd circuit causing edges, enumerate all edges between vertices in the same color set
  - For even circuit causing edges, enumerate all edges between vertices in opposite color sets that don’t exist in the original graph

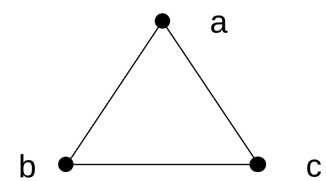
General Approach to Solutions: Chromatic Numbers and Valid Colorings

- **Two-coloring as a product of Boolean constraints**
  - $(ab' \vee a'b) \wedge (ba' \vee b'a) = (ab' \vee a'b)$



- **Generalized to >2 colors**

a	b	c
00	01	01
00	01	10
00	10	10
..	..	..
..	..	..
10	01	01

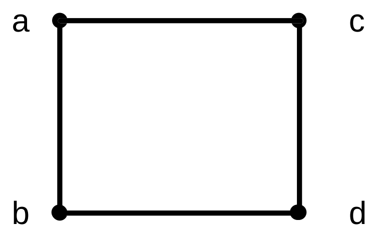


- **Solution:**
  - **Write constraints for *x* colors on *n* vertices**
  - **Compute product**
  - **Existing minterms verify chromatic number *x***
  - **Number of minterms indicate possible valid colorings**

Implementation Specifics: Representing a Graph

- Each edge is a minterm
- Each minterms has only two 1's - adjacent vertices

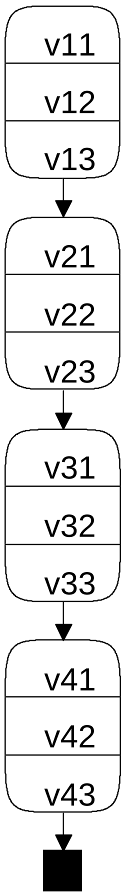
a	b	c	d
1	1	0	0
1	0	1	0
0	1	0	1
0	0	1	1



- Universe is a tuple
- Useful for partitioning
  - 00-0000----0000-
  - Intersect with graph
  - Returns all edges not adjacent to 0's
- Strings of zeros in the BDD
  - 11 vertex, 18 edge graph
  - 38 nodes BDD
  - 23 nodes ZBDD

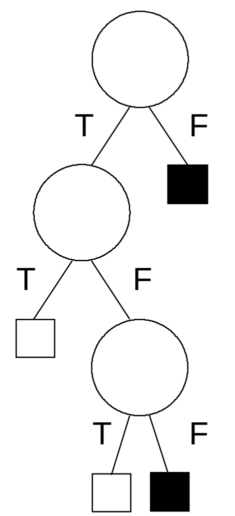
Implementation Specifics: Boolean Relations

- **Construct a BDD which describes Boolean Relations among vertex colorings**
- **A path to one is a valid coloring**
- **Two-Coloring =** $\prod_{j,k}\left(v_j \oplus v_k\right)$ **where** $\left(v_j, v_k\right) \in$ **edges**
- $\mathbf{2^{\mathit{i}}}$**-Coloring =** $\prod_{j,k}\sum_{i}\left(v_{ji} \oplus v_{ki}\right)$ **where** $\left(v_j, v_k\right) \in$ **edges**



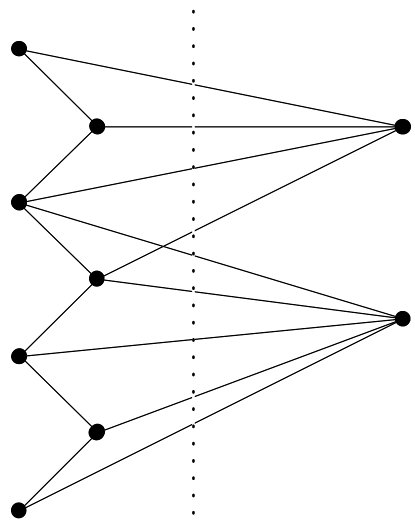
Implementation Specifics: Tighter Constraints

- **Logarithmic color encoding - how to specify exactly 5 colors?**
- **Add additional constraints c_j and c_k**
  - **c_j and c_k are BDDs with i levels and minterms from 0 to x-1**
  - **x-Coloring =** $\prod_{j,k}\left(c_j \cap c_k \cap \sum_i \left(v_{ji} \oplus v_{ki}\right)\right)$
- **Building this BDD is straightforward:**
  - **Contains 5 colors, 0-- and 100**



Further Research

- **Chromatic number and valid coloring techniques explode rapidly**
- **Needed:**
  - **Tighter constraints**
  - **Partitioning**
- **Many unexplored avenues**
  - **One possible partitioning:**
    - Partition between high and low degree vertices
    - Color low degree side leaving sufficient “holes” for high degree side
    - Color high degree side
    - Check for compatible colorings



Conclusions

- **Efficient two-coloring and odd-circuit producing edge finding techniques**
  - **Only 2 minterms in BDD**
  - **Maximum number of BDD nodes is roughly two times number of vertices**
  - **Time spent is on order of number of edges in original graph**
- **Chromatic numbering and valid colorings works for small cases**
  - **Explodes rapidly**
  - **Possibility of more constraints**
  - **Possibility of recursive partitioning**