On the Graph Coloring problem and its Generalizations

ThanhVu H. Nguyen
Advisor: Dr. Thang N. Bui

Master Thesis in Computer Science
Penn State Harrisburg

22nd July 2006
Outline

1. Problems Definition
2. ABGC Algorithm
3. Results
4. Conclusion
Outline

1. Problems Definition
2. ABGC Algorithm
3. Results
4. Conclusion
The classic Graph Coloring Problem (GCP)

Input: Undirected graph \( G = (V, E) \)

Output: A minimum coloring of \( G \) such that each vertex is assigned a color (an integer) such that adjacent vertices have different colors, and the total number of colors used is minimum.
The classic Graph Coloring Problem (GCP)

Input

Undirected graph \( G = (V, E) \)
The classic Graph Coloring Problem (GCP)

**Input**

Undirected graph \( G = (V, E) \)

**Output**

A **minimum** coloring of \( G \)

- That is, each vertex of \( G \) is assigned a color (an integer) such that adjacent vertices have different colors, and the total number of colors used is minimum.
# The Generalizations of Graph Coloring

The objective is to minimize the number of colors used. Additional constraints include:

1. Bandwidth Coloring
2. Multi Coloring
3. Bandwidth Multi Coloring
The Generalizations of Graph Coloring

- Similar objective: Minimize the number of colors used.
The Generalizations of Graph Coloring

- Similar objective: Minimize the number of colors used.
- Additional Constraints
The Generalizations of Graph Coloring

- Similar objective: **Minimize** the number of colors used.
- Additional Constraints
- 3 generalizations are considered:
The Generalizations of Graph Coloring

- Similar objective: Minimize the number of colors used.
- Additional Constraints
- 3 generalizations are considered:
  - 1. Bandwidth Coloring
The Generalizations of Graph Coloring

- Similar objective: Minimize the number of colors used.
- Additional Constraints
- 3 generalizations are considered:
  1. Bandwidth Coloring
  2. Multi Coloring
The Generalizations of Graph Coloring

- Similar objective: Minimize the number of colors used.
- Additional Constraints
- 3 generalizations are considered:
  1. Bandwidth Coloring
  2. Multi Coloring
  3. Bandwidth Multi Coloring
The Bandwidth Coloring Problem (BCP)

The Bandwidth Coloring Problem (BCP) is a variant of the Graph Coloring Problem (GCP) with an additional constraint. It involves an undirected graph \( G = (V, E) \) and an edge weight function \( d \). The goal is to assign colors to the vertices such that the colors of adjacent vertices differ by at least the weight of the edge connecting them. The constraint is formalized as:

\[ d(u, v) = 1, \quad \forall (u, v) \in E \]
The Bandwidth Coloring Problem (BCP)

**Input**

Undirected graph $G = (V, E)$ and edge weight function $d$
The Bandwidth Coloring Problem (BCP)

**Input**
Undirected graph $G = (V, E)$ and edge weight function $d$

**Output**
Similar to GCP with the additional constraint that the colors of adjacent vertices must differ by at least the weight of the edge connecting them
The Bandwidth Coloring Problem (BCP)

**Input**

Undirected graph $G = (V, E)$ and edge weight function $d$

**Output**

Similar to GCP with the additional constraint that the colors of adjacent vertices must differ by at least the weight of the edge connecting them

$$BCP = GCP \text{ if } d(u, v) = 1, \forall (u, v) \in E$$
The Multi Coloring Problem (MCP)

**Input**
Undirected graph \( G = (V, E) \) and vertex weight function \( w \)

**Output**
Each vertex \( u \) is assigned a set of \( w(u) \) distinct colors such that the color sets of any two adjacent vertices are disjoint, and the total number of colors used is minimum.

MCP = GCP if \( w(u) = 1 \), \( \forall u \in V \)
The Multi Coloring Problem (MCP)

Input

Undirected graph $G = (V, E)$ and vertex weight function $w$
The Multi Coloring Problem (MCP)

**Input**

Undirected graph $G = (V, E)$ and vertex weight function $w$

**Output**

Each vertex $u$ is assigned a set of $w(u)$ distinct colors such that the color sets of any two adjacent vertices are disjoint, and the total number of colors used is minimum.
# The Multi Coloring Problem (MCP)

## Input

Undirected graph \( G = (V, E) \) and vertex weight function \( w \)

## Output

Each vertex \( u \) is assigned a set of \( w(u) \) distinct colors such that the color sets of any two adjacent vertices are disjoint, and the total number of colors used is minimum.

\[ MCP = GCP \text{ if } w(u) = 1, \forall u \in V \]
The Bandwidth Multi Coloring Problem (BMCP)

Input
- Undirected graph $G = (V, E)$, edge weight function $d$
- Vertex weight function $w$

Output
- Similar to Multi Coloring problem with an additional constraint as in the Bandwidth Coloring problem
- If $u$ and $v$ are adjacent vertices, then each color in the color set of $u$ must differ from each color in the color set of $v$ by at least $d(u, v)$.

BMCP = GCP if $d(u, v) = 1$, $\forall (u, v) \in E$ and $w(u) = 1$, $\forall u \in V$. 
The Bandwidth Multi Coloring Problem (BMCP)

Input

Undirected graph $G = (V, E)$, edge weight function $d$ and vertex weight function $w$
The Bandwidth Multi Coloring Problem (BMCP)

**Input**

Undirected graph $G = (V, E)$, edge weight function $d$ and vertex weight function $w$

**Output**

Similar to Multi Coloring problem with an additional constraint as in the Bandwidth Coloring problem

- If $u$ and $v$ are adjacent vertices, then each color in the color set of $u$ must differ from each color in the color set of $v$ by at least $d(u, v)$.
The Bandwidth Multi Coloring Problem (BMCP)

Input

Undirected graph $G = (V, E)$, edge weight function $d$ and vertex weight function $w$

Output

Similar to Multi Coloring problem with an additional constraint as in the Bandwidth Coloring problem

- If $u$ and $v$ are adjacent vertices, then each color in the color set of $u$ must differ from each color in the color set of $v$ by at least $d(u, v)$.

$BMCP = GCP$ if $d(u, v) = 1, \forall (u, v) \in E$ and $w(u) = 1, \forall u \in V$. 
## Applications

Some applications for the Graph Coloring problem:

- Scheduling (classrooms, jobs)
- CPU register allocation
- Air traffic flow control
- Cell phone network: different cells require frequencies that must be at some distance apart in order to minimize interferences. This can be modeled by the Bandwidth Coloring problem. If multiple frequencies are assigned to a cell then this problem can be modeled by the Bandwidth Multi Coloring problem.
Applications

Some applications for the Graph Coloring problem

- Scheduling (classrooms, jobs)
- CPU register allocation
- Air traffic flow control
### Some applications for the Graph Coloring problem

- Scheduling (classrooms, jobs)
- CPU register allocation
- Air traffic flow control

Some applications for the Graph Coloring problem

- Cell phone network: different cells require frequencies that must be at some distance apart in order to minimize interferences. This can be modeled by the Bandwidth Coloring problem.
- If multiple frequencies are assigned to a cell then this problem can be modeled by the Bandwidth Multi Coloring problem.
## Applications

### Some applications for the Graph Coloring problem

- Scheduling (classrooms, jobs)
- CPU register allocation
- Air traffic flow control

### Some applications for the Graph Coloring problem

- **Cell phone network**: different cells require frequencies that must be at some distance apart in order to minimize interferences. This can be modeled by the **Bandwidth Coloring** problem.
Applications

Some applications for the Graph Coloring problem

- Scheduling (classrooms, jobs)
- CPU register allocation
- Air traffic flow control

Some applications for the Graph Coloring problem

- **Cell phone network**: different cells require frequencies that must be at some distance apart in order to minimize interferences. This can be modeled by the Bandwidth Coloring problem.
- If multiple frequencies are assigned to a cell then this problem can be modeled by the Bandwidth Multi Coloring problem.
Complexity of the Graph Coloring Problems

- **Complexity**: GCP is a classic NP-Hard problem. The generalizations are NP-Hard (extensions to GCP).

- **Approximation Complexity**: For GCP, it is difficult to approximate. Not much is known about the approximation complexity for the generalizations. A cannot be approximated within $|V|/7 - \epsilon$, for any $\epsilon > 0$, unless $P \equiv NP$. 
# Complexity of the Graph Coloring Problems

## Complexity

- GCP is a classic \( \mathcal{NP} \)-Hard problem.
Complexity of the Graph Coloring Problems

**Complexity**

- GCP is a classic $\mathcal{NP}$-Hard problem.
- The generalizations are $\mathcal{NP}$-Hard (extensions to GCP).
Complexity of the Graph Coloring Problems

Complexity

- GCP is a classic \( \mathcal{NP} \)-Hard problem.
- The generalizations are \( \mathcal{NP} \)-Hard (extensions to GCP).
Complexity of the Graph Coloring Problems

**Complexity**
- GCP is a classic $\mathcal{NP}$-Hard problem.
- The generalizations are $\mathcal{NP}$-Hard (extensions to GCP).

**Approximation Complexity**
- For GCP, it is difficult to approximate\(^a\)

\(^a\)can not be approximated within $|V|^{1/7-\epsilon}$, for any $\epsilon > 0$, unless $P \equiv NP$
Complexity of the Graph Coloring Problems

### Complexity

- GCP is a classic $\mathcal{NP}$-Hard problem.
- The generalizations are $\mathcal{NP}$-Hard (extensions to GCP).

### Approximation Complexity

- For GCP, it is difficult to approximate$^a$
- Not much is known about the approximation complexity for the generalizations

$^a$can not be approximated within $|V|^{1/7-\epsilon}$, for any $\epsilon > 0$, unless $P \equiv NP$
<table>
<thead>
<tr>
<th>Problems Definition</th>
<th>ABGC Algorithm</th>
<th>Results</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous heuristic approaches</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Previous heuristic approaches

For GCP

- Tabu search, local search, genetic and ant-based algorithms
Previous heuristic approaches

For GCP
- Tabu search, local search, genetic and ant-based algorithms

For the Generalizations
Previous heuristic approaches

For GCP
- Tabu search, local search, genetic and ant-based algorithms

For the Generalizations
- Local search and constraint propagation
Previous heuristic approaches

For GCP
- Tabu search, local search, genetic and ant-based algorithms

For the Generalizations
- Local search and constraint propagation
- Squeaky wheel optimization (SWO)
The algorithm features agents (or ants), exploring and coloring the graph.
An Agent Based Algorithm For Graph Coloring (ABGC)

- The algorithm features agents (or ants), exploring and coloring the graph.
- Each agent works on a local portion of the graph. Individual agents *do not* build complete solutions.
An Agent Based Algorithm For Graph Coloring (ABGC)

- The algorithm features agents (or ants), exploring and coloring the graph.
- Each agent works on a local portion of the graph. Individual agents *do not* build complete solutions.
- Additional techniques to help the agents:
An Agent Based Algorithm For Graph Coloring (ABGC)

- The algorithm features agents (or ants), exploring and coloring the graph.
- Each agent works on a local portion of the graph. Individual agents do not build complete solutions.
- Additional techniques to help the agents:
  - Tabu list, greedy-based local optimization, perturbations (to avoid local optima)
Algorithm Outline

<table>
<thead>
<tr>
<th>Problems Definition</th>
<th>ABGC Algorithm</th>
<th>Results</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Algorithm Outline**

**Initialization**
- $k \leftarrow$ number of available colors

**Repeat**
- ** Exploration**
  - Each agent moves around a portion of the graph and colors some of the visited vertices using at most $k$ colors.
- **Exploitation**
  - A local optimization technique is used to improve the coloring done in the exploration phase.
  - Keep the best coloring found so far.
  - $k \leftarrow$ number of colors in the bestColoring $- 1$

**Jolt operation**
- If $k$ has not changed for a while, perturb the current coloring

**Until** some criteria are met

**Return** the best coloring found
Algorithm Outline

- **Initialization**
  - $k \leftarrow \# \text{ of available colors}$

- **Repeat**
  - **Exploration**
    - Each agent moves around a portion of the graph and colors some of the visited vertices using at most $k$ colors.
  - **Exploitation**
    - A local optimization technique is used to improve the coloring done in the exploration phase.
    - Keep the best coloring found so far.
    - $k \leftarrow \# \text{ of colors in the bestColoring} - 1$
  - **Jolt operation**
    - If $k$ has not changed for a while, perturb the current coloring

- **Until** some criteria are met

- **Return** the best coloring found
Pre-processing

- If the input is an instance of BCP (or GCP), no processing is needed.
Pre-processing

- If the input is an instance of BCP (or GCP), no processing is needed.
- If the input is an instance of MCP or BMCP, it is transformed into an instance of BCP.
# Initial Coloring

<table>
<thead>
<tr>
<th>Problems Definition</th>
<th>ABGC Algorithm</th>
<th>Results</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Objective: Find an initial valid coloring quickly</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>How it works: Find 21 colorings of the graph and keep the best coloring found</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Do one greedy coloring</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Vertices are considered in decreasing order of degree. When a vertex is considered, it is colored with the smallest feasible color.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Do twenty (20) random colorings</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Same as above but vertices are considered in a random order</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Return the best of these 21 colorings</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly
Initial Coloring

Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

How it works
- Find 21 colorings of the graph and keep the best coloring found
Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

**How it works**

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
Initial Coloring

Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

How it works

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
  - Vertices are considered in decreasing order of degree.
Iterative Greedy algorithm

Objective: Find an initial valid coloring quickly

How it works

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
  - Vertices are considered in decreasing order of degree.
  - When a vertex is considered, it is colored with the smallest feasible color.
Initial Coloring

Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

**How it works**

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
  - Vertices are considered in decreasing order of degree.
  - When a vertex is considered, it is colored with the smallest feasible color.
- Do twenty (20) random colorings
Initial Coloring

Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

How it works

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
  - Vertices are considered in decreasing order of degree.
  - When a vertex is considered, it is colored with the smallest feasible color.
- Do twenty (20) random colorings
  - Same as above but vertices are considered in a random order
Initial Coloring

Iterative Greedy algorithm

- **Objective**: Find an initial valid coloring quickly

**How it works**

- Find 21 colorings of the graph and keep the best coloring found
- Do one greedy coloring
  - Vertices are considered in decreasing order of degree.
  - When a vertex is considered, it is colored with the smallest feasible color.
- Do twenty (20) random colorings
  - Same as above but vertices are considered in a random order
- Return the best of these 21 colorings
Initial Coloring

Iterative Greedy algorithm returns a coloring with $k$ colors.

Attempt a new goal for the agents, $k \leftarrow k - 1$.
Initial Coloring

- Iterative Greedy algorithm returns a coloring with $k$ colors.
Initial Coloring

- Iterative Greedy algorithm returns a coloring with $k$ colors.
- Attempt a new goal for the agents, $k \leftarrow k - 1$
Initial Coloring

- Iterative Greedy algorithm returns a coloring with $k$ colors.
- Attempt a new goal for the agents, $k \leftarrow k - 1$
  - $k$ is the number of colors the agents can use to color the Graph with.
General Ideas

Each of the agents executes the following sequence of operations:
Each of the agents executes the following sequence of operations:

- Place itself on a vertex with *maximum conflict*
Each of the agents executes the following sequence of operations:

- Place itself on a vertex with *maximum conflict*
- Make a number of moves
Each of the agents executes the following sequence of operations:

- Place itself on a vertex with *maximum conflict*
- Make a number of moves
- Color the vertices
Exploration: How an Agent Moves

A move consists of two steps (assume an agent is at vertex $u$):

Step 1: Randomly select a vertex $v$ among the vertices adjacent to $u$. Go to $v$.

Step 2: Select the vertex with the highest conflict, say $w$, among the vertices adjacent to $v$. Go to $w$. The agent colors $w$, then adds it to the Tabu list of the agent. (Each agent has its own fixed sized Tabu List)
Exploration: How an Agent Moves

A move consists of two steps
(assume an agent is at vertex $u$)
Exploration: How an Agent Moves

A move consists of two steps
*(assume an agent is at vertex u)*

- **Step 1:** Randomly select a vertex \( v \) among the vertices adjacent to \( u \). Go to \( v \).
A move consists of two steps
(assume an agent is at vertex $u$)

- Step 1: Randomly select a vertex $v$ among the vertices adjacent to $u$. Go to $v$.
- Step 2: Select the vertex with the highest conflict, say $w$, among the vertices adjacent to $v$. Go to $w$. 

Exploration: How an Agent Moves

A move consists of two steps
(assume an agent is at vertex \( u \))

- Step 1: Randomly select a vertex \( v \) among the vertices adjacent to \( u \). Go to \( v \).
- Step 2: Select the vertex with the highest conflict, say \( w \), among the vertices adjacent to \( v \). Go to \( w \).
  - The agent colors \( w \), then adds it to the **Tabu list** of the agent. (Each agent has its own fixed sized Tabu List)
Exploration: How an Agent Colors a Vertex

Objective: each agent colors or re-colors a vertex so that the conflict at that vertex is zero, if possible. Color decision is based on local information, no global knowledge.
Exploration: How an Agent Colors a Vertex

- **Objective**: each agent colors or re-colors a vertex so that the conflict at that vertex is zero, if possible.
**Objective**: each agent colors or re-colors a vertex so that the conflict at that vertex is zero, if possible.

- Color decision is based on local information, no global knowledge.
Exploration: How an Agent Colors a Vertex

1. AvailableColors: set of potential colors that the agent could use
2. ForbiddenColors: set of colors that cannot be used to color the current vertex as they will create conflict
3. EligibleColors = AvailableColors - ForbiddenColors

This is usually a union of intervals, e.g.,

\{6, 7, 8, 9, 12, 13, 14, 15\} = [6...9] ⋃ [12...15]
Exploration: How an Agent Colors a Vertex

To color a vertex, an agent considers the following three sets:

1. **AvailableColors**: set of potential colors that the agent could use
To color a vertex, an agent considers the following three sets:

1. **AvailableColors**: set of potential colors that the agent could use.
2. **ForbiddenColors**: set of colors that *cannot* be used to color the current vertex as they will create conflict.
Exploration: How an Agent Colors a Vertex

To color a vertex, an agent considers the following three sets:

1. **AvailableColors**: set of potential colors that the agent could use
2. **ForbiddenColors**: set of colors that *cannot* be used to color the current vertex as they will create conflict
3. **EligibleColors** = **AvailableColors** − **ForbiddenColors**
Exploration: How an Agent Colors a Vertex

To color a vertex, an agent considers the following three sets:

1. **AvailableColors**: set of potential colors that the agent could use.
2. **ForbiddenColors**: set of colors that *cannot* be used to color the current vertex as they will create conflict.
3. **EligibleColors** = **AvailableColors** − **ForbiddenColors**
   - This is usually a union of intervals, e.g., \{6, 7, 8, 9, 12, 13, 14, 15\} = \([6 \ldots 9] \cup [12 \ldots 15]\)
Exploration: How an Agent Colors a Vertex

If \( \text{EligibleColors} = \emptyset \), choose the color that has the fewest conflicts with the adjacent vertices.

If \( \text{EligibleColors} \neq \emptyset \), choose the color that is the median of the largest interval.

Idea: Give neighbor vertices more room to meet their constraints.
Exploration: How an Agent Colors a Vertex

If $\text{EligibleColors} = \emptyset$

Choose the color that has the fewest conflicts with the adjacent vertices.

If $\text{EligibleColors} \neq \emptyset$

Choose the color that is the median of the largest interval.

Idea: Give neighbor vertices more room to meet their constraints.
**Exploration: How an Agent Colors a Vertex**

If $\text{EligibleColors} = \emptyset$

Choose the color that has the fewest conflicts with the adjacent vertices
Exploration: How an Agent Colors a Vertex

If $\text{EligibleColors} = \emptyset$

Choose the color that has the fewest conflicts with the adjacent vertices

If $\text{EligibleColors} \neq \emptyset$

Idea: Give neighbor vertices more room to meet their constraints
Exploration: How an Agent Colors a Vertex

If EligibleColors = ∅
Choose the color that has the fewest conflicts with the adjacent vertices

If EligibleColors ≠ ∅
Choose the color that is the median of the largest interval

■ Idea: Give neighbor vertices more room to meet their constraints
**Example: How an Agent Colors**

| Problems Definition | ABGC Algorithm | Results | Conclusion |
|---------------------|----------------|---------|------------|------------|
|                     |                |         |            |            |
Exploitation: Local Optimization

Local Optimization

**Input:** a (valid) coloring for graph G

**Output:** a different coloring for graph G

**How it works:**

Similar to the Iterative Greedy algorithm

Sort the vertices to be re-colored in decreasing order of the input coloring

Erase all colors from the graph then re-color the vertices using the sorted order

Replace the current coloring with the one returned from Local Opt (if it is better)

Attempt a new goal (fewer colors) for the agents

\[ k \leftarrow \# \text{of colors in the current best coloring} - 1 \]
Exploitation: Local Optimization

Local Optimization

Input: a (valid) coloring for graph G
Output: a different coloring for graph G

How it works
Similar to the Iterative Greedy algorithm
Sort the vertices to be re-colored in decreasing order of the input coloring
Erase all colors from the graph then re-color the vertices using the sorted order
Replace the current coloring with the one returned from Local Opt (if it is better)
Attempt a new goal (fewer colors) for the agents

\[ k \leftarrow \# \text{of colors in the current best coloring} - 1 \]
Exploitation: Local Optimization

**Local Optimization**

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

How it works

Similar to the Iterative Greedy algorithm

1. Sort the vertices to be re-colored in decreasing order of the input coloring
2. Erase all colors from the graph then re-color the vertices using the sorted order
3. Replace the current coloring with the one returned from Local Opt (if it is better)
4. Attempt a new goal (fewer colors) for the agents

$$k \leftarrow #\text{of colors in the current best coloring} - 1$$
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**

- Similar to the Iterative Greedy algorithm
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**
- Similar to the Iterative Greedy algorithm
- Sort the vertices to be re-colored in *decreasing* order of the input coloring
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**

- Similar to the Iterative Greedy algorithm
- Sort the vertices to be re-colored in *decreasing* order of the input coloring
- Erase all colors from the graph then re-color the vertices using the sorted order
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**
- Similar to the Iterative Greedy algorithm
- Sort the vertices to be re-colored in *decreasing* order of the input coloring
- Erase all colors from the graph then re-color the vertices using the sorted order
- Replace the current coloring with the one returned from Local Opt (if it is better)
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

- **How it works**
  - Similar to the Iterative Greedy algorithm
  - Sort the vertices to be re-colored in *decreasing* order of the input coloring
  - Erase all colors from the graph then re-color the vertices using the sorted order
  - Replace the current coloring with the one returned from Local Opt (if it is better)
  - Attempt a new goal (fewer colors) for the agents
Exploitation: Local Optimization

Local Optimization

- **Input**: a (valid) coloring for graph G
- **Output**: a different coloring for graph G

**How it works**

- Similar to the Iterative Greedy algorithm
- Sort the vertices to be re-colored in *decreasing* order of the input coloring
- Erase all colors from the graph then re-color the vertices using the sorted order
- Replace the current coloring with the one returned from Local Opt (if it is better)
- Attempt a new goal (fewer colors) for the agents
  - $k \leftarrow \# \text{ of colors in the current best coloring} - 1$
Perturbation: Jolt Operation

Objective: Attempt to escape local optima by perturbing the current coloring.

How it works:
- Randomly recolor neighbors of the top $\beta$% conflicted vertices.
- Perturbation is done whenever a period of time has passed without any improvement.
Perturbation: Jolt Operation

Objective: Attempt to escape local optima by perturbing the current coloring.
Perturbation: Jolt Operation

Jolt Operation

- **Objective**: Attempt to escape local optima by perturbing the current coloring.
- **How it works**
Perturbation: Jolt Operation

- **Objective**: Attempt to escape local optima by perturbing the current coloring.
- **How it works**
  - Randomly recolor neighbors of the top $\beta \%$ conflicted vertices
Jolt Operation

- **Objective**: Attempt to escape local optima by perturbing the current coloring.

- **How it works**
  - Randomly recolor neighbors of the top $\beta$ % conflicted vertices
Perturbation: Jolt Operation

**Objective:** Attempt to escape local optima by perturbing the current coloring.

**How it works**
- Randomly recolor neighbors of the top $\beta\%$ conflicted vertices
- Perturbation is done whenever a period of time has passed without any improvement.
Stopping Criteria

ABGC terminates when

A number of cycles has passed
OR
A number of cycles has passed without any improvement
ABGC terminates when

- A number of cycles has passed
Stopping Criteria

ABGC terminates when

- A number of cycles has passed

OR
ABGC terminates when

- A number of cycles has passed
- OR
- A number of cycles has passed without any improvement
Outline

1. Problems Definition
2. ABGC Algorithm
3. Results
4. Conclusion
<table>
<thead>
<tr>
<th>Problems Definition</th>
<th>ABGC Algorithm</th>
<th>Results</th>
<th>Conclusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing Details</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Implemented in C++

Test machine: 3GHz Pentium 4, 2GB of RAM, Linux operating system

Benchmark Instances

99 instances were created from 33 (DIMACS) graphs for 3 different coloring problems. The algorithm is run 100 times for each instance.
# Testing Details

## Implementation Details
- Implemented in C++
- Test machine: 3GHz Pentium 4, 2GB of RAM, Linux operating system
Testing Details

Implementation Details
- Implemented in C++
- Test machine: 3GHz Pentium 4, 2GB of RAM, Linux operating system

Benchmark Instances
Testing Details

Implementation Details
- Implemented in C++
- Test machine: 3GHz Pentium 4, 2GB of RAM, Linux operating system

Benchmark Instances
- 99 instances were created from 33 (DIMACS) graphs for 3 different coloring problems.
Testing Details

Implementation Details

- Implemented in C++
- Test machine: 3GHz Pentium 4, 2GB of RAM, Linux operating system

Benchmark Instances

- 99 instances were created from 33 (DIMACS) graphs for 3 different coloring problems.
- The algorithm is run 100 times for each instance.
Comparison

Results are compared against the following algorithms:

- Squeaky Wheel Optimization (SWO) (all generalizations, Lim et al, 2003)
- SWO + Tabu Search (SWO/TS) (all generalizations, Lim et al, 2005)
- Local Search & Constraint Propagation
  - FCNS (Bandwidth Coloring only, Prestwich, 2002)
  - SATURN (Bandwidth Multi-Coloring only, Prestwich, 2002)
  - There are no results for the Multi Coloring Problem.
Comparison

Results are compared against the following algorithms

- Squeaky Wheel Optimization
Comparison

Results are compared against the following algorithms

- Squeaky Wheel Optimization
  - SWO (all generalizations, Lim et al, 2003)
Comparison

Results are compared against the following algorithms

- **Squeaky Wheel Optimization**
  - SWO (all generalizations, Lim et al, 2003)
  - SWO + Tabu Search (**SWO/TS**) (all generalizations, Lim et al, 2005)
Results are compared against the following algorithms

- **Squeaky Wheel Optimization**
  - SWO (all generalizations, Lim et al, 2003)
  - SWO + Tabu Search (SWO/TS) (all generalizations, Lim et al, 2005)
    - Consider only results from SWO/TS since it outperforms SWO.
Comparison

Results are compared against the following algorithms

- **Squeaky Wheel Optimization**
  - SWO (all generalizations, Lim et al, 2003)
  - SWO + Tabu Search (SWO/TS) (all generalizations, Lim et al, 2005)
    - Consider only results from SWO/TS since it outperforms SWO.

- **Local Search & Constraint Propagation**
Comparison

Results are compared against the following algorithms

- **Squeaky Wheel Optimization**
  - SWO (all generalizations, Lim et al, 2003)
  - SWO + Tabu Search (SWO/TS) (all generalizations, Lim et al, 2005)
    - Consider only results from SWO/TS since it outperforms SWO.

- **Local Search & Constraint Propagation**
  - FCNS (Bandwidth Coloring only, Prestwich, 2002)
Comparison

Results are compared against the following algorithms

- **Squeaky Wheel Optimization**
  - **SWO** (all generalizations, Lim et al, 2003)
  - **SWO + Tabu Search (SWO/TS)** (all generalizations, Lim et al, 2005)
    - Consider only results from SWO/TS since it outperforms SWO.

- **Local Search & Constraint Propagation**
  - **FCNS** (Bandwidth Coloring only, Prestwich, 2002)
  - **SATURN** (Bandwidth Multi Coloring only, Prestwich, 2002)
Comparison

Results are compared against the following algorithms:

- **Squeaky Wheel Optimization**
  - **SWO** (all generalizations, Lim et al, 2003)
  - **SWO + Tabu Search (SWO/TS)** (all generalizations, Lim et al, 2005)
    - Consider only results from SWO/TS since it outperforms SWO.

- **Local Search & Constraint Propagation**
  - **FCNS** (Bandwidth Coloring only, Prestwich, 2002)
  - **SATURN** (Bandwidth Multi Coloring only, Prestwich, 2002)
  - There are no results for the Multi Coloring Problem
Result: Bandwidth Coloring Problem

gem20a
gem20b
gem30
gem30a
gem40a
gem40b
gem50a
gem50b
gem60
gem60a
gem60b
gem70a
gem70b
**Result: Bandwidth Coloring Problem**

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
</tr>
</tbody>
</table>
## Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
</tr>
</tbody>
</table>
## Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
</tr>
</tbody>
</table>
# Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/ TS</th>
<th>ABGC</th>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/ TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>geom80</td>
<td>41</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>geom80a</td>
<td>63</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>geom80b</td>
<td>61</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
<td>geom90a</td>
<td>64</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>geom90b</td>
<td>72</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom100</td>
<td>50</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
<td>geom100a</td>
<td>70</td>
<td>76</td>
<td>71</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
<td>geom100b</td>
<td>73</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom110</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
<td>geom110a</td>
<td>74</td>
<td>82</td>
<td>75</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>geom110b</td>
<td>79</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
<td>geom120</td>
<td>60</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
<td>geom120a</td>
<td>84</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>geom120b</td>
<td>87</td>
<td>98</td>
<td>91</td>
</tr>
</tbody>
</table>
## Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>geom80</td>
<td>41</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>geom80a</td>
<td>63</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>geom80b</td>
<td>61</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
<td>geom90a</td>
<td>64</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>geom90b</td>
<td>72</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom100</td>
<td>50</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
<td>geom100a</td>
<td>70</td>
<td>76</td>
<td>71</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
<td>geom100b</td>
<td>73</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom110</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
<td>geom110a</td>
<td>74</td>
<td>82</td>
<td>75</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>geom110b</td>
<td>79</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
<td>geom120</td>
<td>60</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
<td>geom120a</td>
<td>84</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>geom120b</td>
<td>87</td>
<td>98</td>
<td>91</td>
</tr>
</tbody>
</table>
Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>geom80</td>
<td>41</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>geom80a</td>
<td>63</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>geom80b</td>
<td>61</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
<td>geom90a</td>
<td>64</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>geom90b</td>
<td>72</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom100</td>
<td>50</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
<td>geom100a</td>
<td>70</td>
<td>76</td>
<td>71</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
<td>geom100b</td>
<td>73</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom110</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
<td>geom110a</td>
<td>74</td>
<td>82</td>
<td>75</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>geom110b</td>
<td>79</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
<td>geom120</td>
<td>60</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
<td>geom120a</td>
<td>84</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>geom120b</td>
<td>87</td>
<td>98</td>
<td>91</td>
</tr>
</tbody>
</table>
Result: Bandwidth Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>geom80</td>
<td>41</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>geom80a</td>
<td>63</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>geom80b</td>
<td>61</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
<td>geom90a</td>
<td>64</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>geom90b</td>
<td>72</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom100</td>
<td>50</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
<td>geom100a</td>
<td>70</td>
<td>76</td>
<td>71</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
<td>geom100b</td>
<td>73</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom110</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
<td>geom110a</td>
<td>74</td>
<td>82</td>
<td>75</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>geom110b</td>
<td>79</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
<td>geom120</td>
<td>60</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
<td>geom120a</td>
<td>84</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>geom120b</td>
<td>87</td>
<td>98</td>
<td>91</td>
</tr>
</tbody>
</table>
**Result: Bandwidth Coloring Problem**

<table>
<thead>
<tr>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/ABGC</th>
<th>ABGC</th>
<th>Instance</th>
<th>FCNS</th>
<th>SWO/ABGC</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20a</td>
<td>20</td>
<td>22</td>
<td>20</td>
<td>geom80</td>
<td>41</td>
<td>42</td>
<td>41</td>
</tr>
<tr>
<td>geom20b</td>
<td>13</td>
<td>14</td>
<td>13</td>
<td>geom80a</td>
<td>63</td>
<td>66</td>
<td>64</td>
</tr>
<tr>
<td>geom30</td>
<td>28</td>
<td>29</td>
<td>28</td>
<td>geom80b</td>
<td>61</td>
<td>65</td>
<td>64</td>
</tr>
<tr>
<td>geom30a</td>
<td>27</td>
<td>32</td>
<td>27</td>
<td>geom90a</td>
<td>64</td>
<td>69</td>
<td>65</td>
</tr>
<tr>
<td>geom40a</td>
<td>37</td>
<td>38</td>
<td>37</td>
<td>geom90b</td>
<td>72</td>
<td>77</td>
<td>74</td>
</tr>
<tr>
<td>geom40b</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom100</td>
<td>50</td>
<td>51</td>
<td>50</td>
</tr>
<tr>
<td>geom50a</td>
<td>50</td>
<td>52</td>
<td>50</td>
<td>geom100a</td>
<td>70</td>
<td>76</td>
<td>71</td>
</tr>
<tr>
<td>geom50b</td>
<td>35</td>
<td>38</td>
<td>36</td>
<td>geom100b</td>
<td>73</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>geom60</td>
<td>33</td>
<td>34</td>
<td>33</td>
<td>geom110</td>
<td>50</td>
<td>53</td>
<td>50</td>
</tr>
<tr>
<td>geom60a</td>
<td>50</td>
<td>53</td>
<td>50</td>
<td>geom110a</td>
<td>74</td>
<td>82</td>
<td>75</td>
</tr>
<tr>
<td>geom60b</td>
<td>43</td>
<td>46</td>
<td>43</td>
<td>geom110b</td>
<td>79</td>
<td>88</td>
<td>83</td>
</tr>
<tr>
<td>geom70a</td>
<td>62</td>
<td>63</td>
<td>62</td>
<td>geom120</td>
<td>60</td>
<td>62</td>
<td>59</td>
</tr>
<tr>
<td>geom70b</td>
<td>48</td>
<td>54</td>
<td>51</td>
<td>geom120a</td>
<td>84</td>
<td>92</td>
<td>86</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>geom120b</td>
<td>87</td>
<td>98</td>
<td>91</td>
</tr>
</tbody>
</table>
Result: Multi Coloring Problem

- ABGC tied with SWO based algorithms in all instances.
Result: Multi Coloring Problem

- ABGC tied with SWO based algorithms in all instances.
- There are no results from the Local Search & Constraint Propagation based algorithms for the Multi Coloring problem.
Result: Bandwidth Multi Coloring Problem

<table>
<thead>
<tr>
<th>Instance</th>
<th>SATURN</th>
<th>SWO/TS</th>
<th>ABGC</th>
<th></th>
<th>SATURN</th>
<th>SWO/TS</th>
<th>ABGC</th>
</tr>
</thead>
<tbody>
<tr>
<td>geom20</td>
<td>159</td>
<td>149</td>
<td>149</td>
<td>geom80</td>
<td>–</td>
<td>383</td>
<td>382</td>
</tr>
<tr>
<td>geom20a</td>
<td>175</td>
<td>169</td>
<td>169</td>
<td>geom80a</td>
<td>–</td>
<td>379</td>
<td>367</td>
</tr>
<tr>
<td>geom30</td>
<td>168</td>
<td>160</td>
<td>160</td>
<td>geom80b</td>
<td>152</td>
<td>141</td>
<td>139</td>
</tr>
<tr>
<td>geom30a</td>
<td>235</td>
<td>209</td>
<td>210</td>
<td>geom90</td>
<td>–</td>
<td>332</td>
<td>332</td>
</tr>
<tr>
<td>geom30b</td>
<td>79</td>
<td>77</td>
<td>77</td>
<td>geom90a</td>
<td>–</td>
<td>377</td>
<td>378</td>
</tr>
<tr>
<td>geom40</td>
<td>189</td>
<td>167</td>
<td>167</td>
<td>geom90b</td>
<td>–</td>
<td>157</td>
<td>150</td>
</tr>
<tr>
<td>geom40a</td>
<td>260</td>
<td>213</td>
<td>214</td>
<td>geom100</td>
<td>–</td>
<td>404</td>
<td>405</td>
</tr>
<tr>
<td>geom40b</td>
<td>80</td>
<td>74</td>
<td>74</td>
<td>geom100a</td>
<td>–</td>
<td>459</td>
<td>440</td>
</tr>
<tr>
<td>geom50</td>
<td>257</td>
<td>224</td>
<td>224</td>
<td>geom100b</td>
<td>–</td>
<td>170</td>
<td>164</td>
</tr>
<tr>
<td>geom50a</td>
<td>395</td>
<td>318</td>
<td>317</td>
<td>geom110</td>
<td>–</td>
<td>383</td>
<td>378</td>
</tr>
<tr>
<td>geom50b</td>
<td>89</td>
<td>87</td>
<td>85</td>
<td>geom110a</td>
<td>–</td>
<td>494</td>
<td>487</td>
</tr>
<tr>
<td>geom60</td>
<td>279</td>
<td>258</td>
<td>258</td>
<td>geom110b</td>
<td>–</td>
<td>206</td>
<td>208</td>
</tr>
<tr>
<td>geom60a</td>
<td>–</td>
<td>358</td>
<td>357</td>
<td>geom120</td>
<td>–</td>
<td>402</td>
<td>398</td>
</tr>
<tr>
<td>geom60b</td>
<td>128</td>
<td>116</td>
<td>117</td>
<td>geom120a</td>
<td>–</td>
<td>556</td>
<td>548</td>
</tr>
<tr>
<td>geom70</td>
<td>310</td>
<td>273</td>
<td>267</td>
<td>geom120b</td>
<td>–</td>
<td>199</td>
<td>198</td>
</tr>
<tr>
<td>geom70a</td>
<td>–</td>
<td>469</td>
<td>470</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>geom70b</td>
<td>133</td>
<td>121</td>
<td>121</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. Problems Definition
2. ABGC Algorithm
3. Results
4. Conclusion
Conclusion

Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc) produced competitive results.

Generality: applicable to the classic Graph Coloring problem as well as its (three) generalizations.

Future Work

Use pheromone

Explore parallel implementation
ABGC

- Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc)
Conclusion

ABGC

- Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc)
- Produced competitive results
Conclusion

ABGC

- Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc)
- Produced competitive results
- **Generality**: applicable to the classic Graph Coloring problem as well as its (three) generalizations.
Conclusion

ABGC

- Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc)
- Produced competitive results
- **Generality**: applicable to the classic Graph Coloring problem as well as its (three) generalizations.
Conclusion

ABGC

- Agent based, hybrids with many other techniques (Tabu List, Greedy Local optimization, etc)
- Produced competitive results
- **Generality**: applicable to the classic Graph Coloring problem as well as its (three) generalizations.

Future Work

- Use pheromone
- Explore parallel implementation