Solving the Graph Coloring Problem with Cooperative Local Search

Guangping Li – *uzdif@student.kit.edu*

Institute of Theoretical informatics, Algorithmics II

Outline

Outline

ＫＩＴ
Karlsruher Institut für Technologie

The vertex coloring problem

vertex coloring

- The goal is to color the vertices of an undirected graph such that no two adjacent vertices share the same color.
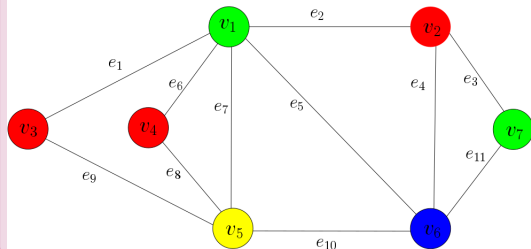
The vertex coloring problem

k-vertex coloring problem

- A k-vertex coloring of a graph is a function $c\colon V \to \{1...k\}$.

Figure: a legal 4-coloring

The vertex coloring problem
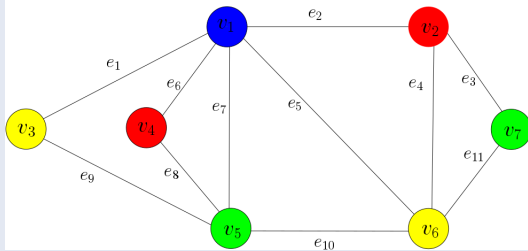
Vertex coloring problem

- The **VCP** is to determine the smallest k, such that the graph can be colored using k colors without conflicts.
- This lower bound k is called the **chromatic number** of G, denoted by $\chi(G)$.

Outline

The Tabucol algorithm
solving k-VCP

- Introduced in 1987 by Hertz and de Werra
- The local search will start from an initial *k*-coloring *c*.
- Changing the color of one vertex to color, is called ***one-step move***.
- the best move with most reduction of conflicts will be reached.
- To avoid short-term cycling, recently performed moves are marked as forbidden moves for a given duration.

The Tabucol algorithm

The pseudo code of *Tabucol*:

Algorithmus 1 : Algorithm Tabucol

---

input : A Graph $G = \{V, E\}$, an integer $k > 0$
parameter : L, α, *Timeout*
output : Coloring c

1 Build a random k-coloring c':
2 $c = c'$;
3 $i = 0$;
4 **while** $(f(c) \neq 0 \land$ *Timeout does not occur*$)$ **do**
5   Evaluate all permitted one-step-moves of c with function Γ;
6   Choose the move [v,i] with maximum $\Gamma(c, v, i)$;
7   Mark the corresponding one-step move [v,i] as a forbidden move with duration $L + \alpha \times f(c)$;
8   Change $c(v) = i$;

Outline

The vertex coloring problem
- k-VCP
- VCP

The Tabucol algorithm

Solving VCP by Tabucol
- Basic scheme
- Data structure
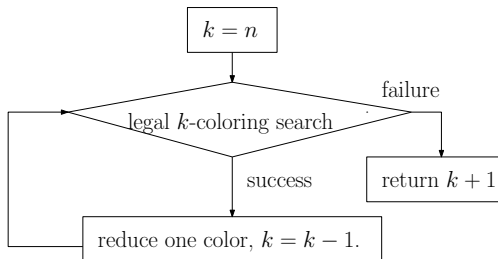- Our improvements

Our Parallel VCP-solver
- Parameter combinations
- Our approaches

Conclusion

Solving VCP by Tabucol
Basic scheme

- Step 1: Initialize Coloring
- Step 2: Solve k-VCP
- Step 3: Reduce a color

$$k = n$$

legal k-coloring search

failure

success

return $k + 1$

reduce one color, $k = k - 1$.

Data structure

Tabu Map and Tabu Queue

- To avoid using the forbidden candidates, a hash map should provide whether a candidate is a tabu or not.
- A one-step move [v, i] is represented by an ordered pair (v, i).
- to update the forbidden moves stored in the *Tabu Map*.
- The size of the *Tabu Queue* is $L + f(n) \times \alpha$.
- When the color of a vertex v is changed, $(v, c(v))$ is recorded in the *Tabu Map* and also enqueued in the *Tabu Queue*.
- When the *Tabu Queue* is full, extra forbidden moves will be popped and will be deleted in the *Tabu Map*.

Data structure

Solution matrix

- To reuse the calculated results, a matrix M is used to record the information about the neighborhood.
- The matrix M evaluates the candidate moves of the current k-coloring c.
- If $c(v_j) = i$, M_{ij} is the number of conflicting edges incident to v_j in the current solution.
- $\frac{\sum_{j=1}^{n} M_{c(v_j)j}}{2}$ is the conflict number of solution c;
- If $c(v_j) \neq i$, M_{ij} is the number of conflicts involving v_j in a neighbor coloring c' of c with
  $c'(v_j) = i$.
  $c'(v_q) = c(v_q), q \neq i, q \in \{1..n\}$.

Data structure

Solution matrix

- $\Gamma(j, i) = M_{c(v_j)j} - M_{ij}$ evaluates the improvement of the move $[j, i]$ in constant time.
- This matrix is filled at the beginning based on the initial solution and constantly changed.
- If the chosen one-step move is $[i, j]$, which means changing the color of v_i from current color $c(v_i)$ to j, some entities in matrix M must be updated.

Evaluation

- The 68 graphs used in our experiments are from the DIMACS benchmark collection.
- The single-threaded experiments were run on computers that had four AMD(R) Opteron(R) processors 6168 (1.9 Ghz with 12 cores) and 256GB RAM. The computers ran the 64-bit version of Ubuntu 12.04.
- The multi-threaded experiments were run on fat nodes InstitutsClusterII. IC2 is a distributed memory parallel computer with 480 16-way so-called thin compute nodes and 5 32-way so-called fat compute nodes. The thin nodes are equipped with 16 cores, 64 GB main memory, whereas the fat nodes are equipped with 32 cores, 512 GB main memory.

Evaluation

- An advantage plot shows the advantage of an algorithms to another algorithm. The y-axis gives the ordered percentage differences.

Our improvements
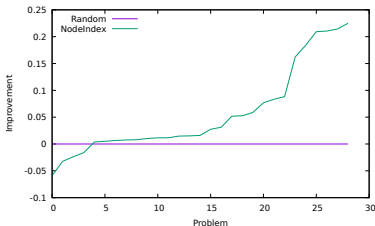
Basic scheme

- Step 1: Initialize Coloring (How to initialize coloring?)
- Step 2: Solve k-VCP (How to improve Tabucol?)
- Step 3: Reduce a color (How to reconstruct coloring?)

Solving VCP by Tabucol

Step 1:How to initialize coloring?

Our Node-index initialization vs Random initialization

- **Node-index initialization** is to use c: $v_i \rightarrow i$ as the initial solution.
- **Random initialization** is to build a coloring randomly.
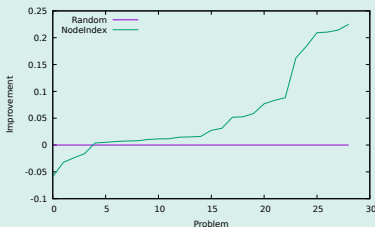
Solving VCP by Tabucol

Step 2: How to improve Tabucol?

Our column-traverse vs row-traverse of solution matrix

- To find next move, the maximum element in the solution matrix must be found.

- If more than one candidate exists, the first found one is chosen as the next step.

- This matrix can be traversed row by row or column by column.
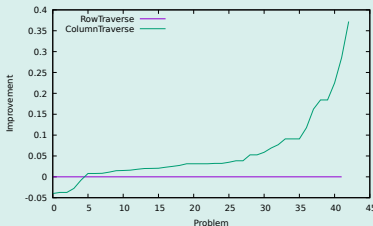
Solving VCP by Tabucol
Step 2: How to improve Tabucol?

Statistic matrix

- The Tabucol algorithm uses a tabu list to avoid short-term cycling.
- To recognize long-term cycling, a *statistic matrix S* is added.
- The S_{ij} represents how many times a one-step move $[i, j]$ was chosen as the next step.
- The candidate with the smallest statistic value will be chosen in the next step.
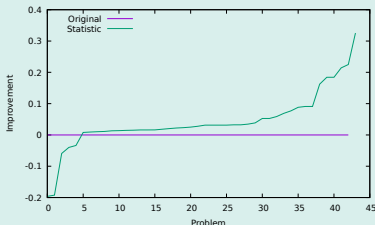
Solving VCP by Tabucol

Step 3: How to reconstruct new coloring?

An observation

It seems that the solution loses its potential in the process of reducing colors iteratively. So it should be helpful to use a new and perhaps more potential coloring.

Replace by a randomly generated solution

We replace the current illegal solution occasionally by a new randomly generated coloring of the same size.
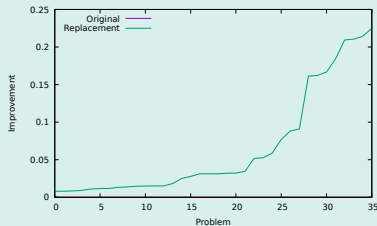
Outline

The vertex coloring problem
- k-VCP
- VCP

The Tabucol algorithm

Solving VCP by Tabucol
- Basic scheme
- Data structure
- Our improvements

Our Parallel GCP-solver
- Parameter combinations
- Our approaches

Conclusion

Guangping Li:
Solving the Graph Coloring Problem with Cooperative Local Search

KIT Department of Informatics
Institute of Theoretical informatics

Our Parallel GCP-solver

Parameter combinations

- Most graphs get better results with the our suggestions.
- Some graphs get better results with the original GCP-solver.
- The agents run with different combinations of the suggestions (L, α, the search directions and whether a statistic matrix is introduced).

Our Parallel GCP-solver

Parameter combinations

Index	L	α	Initialization	Replace	Traverse	Statistic
1	9	0.38	Node-Index	true	Column	true
2	1	0.77	Node-Index	true	Column	true
3	11	0.90	Node-Index	true	Column	true
4	17	0.59	Random	true	Column	true
5	18	0.42	Node-Index	false	Column	false
6	4	0.92	Node-Index	true	Column	true
7	16	0.76	Node-Index	false	Row	false
8	17	0.47	Node-Index	false	Column	false
9	2	0.60	Node-Index	true	Column	false
10	2	0.54	Node-Index	false	Column	true
11	5	0.46	Random	true	Column	true
12	11	0.63	Random	true	Column	true
13	7	0.83	Node-Index	true	Column	true
14	8	0.98	Node-Index	false	Row	true
15	18	0.58	Node-Index	true	Column	false
16	13	0.90	Node-Index	false	Column	true

Our Parallel GCP-solver

Parameter combinations

Index	L	α	Initialization	Replace	Traverse	Statistic
17	20	0.56	Node-Index	true	Column	false
18	10	0.95	Node-Index	true	Column	true
19	15	0.55	Node-Index	true	Row	true
20	17	0.39	Node-Index	true	Column	true
21	18	0.52	Node-Index	false	Column	true
22	11	0.32	Node-Index	true	Column	true
23	15	0.62	Node-Index	false	Column	true
24	6	0.94	Random	true	Column	true
25	9	0.94	Node-Index	false	Column	false
26	12	0.96	Node-Index	true	Column	true
27	16	0.58	Node-Index	false	Column	true
28	9	0.45	Node-Index	false	Column	true
29	19	0.95	Node-Index	true	Column	true
30	18	0.31	Node-Index	true	Column	false
31	6	0.50	Node-Index	false	Column	false
32	15	0.93	Node-Index	false	Column	false

Guangping Li:
Solving the Graph Coloring Problem with Cooperative Local Search

KIT Department of Informatics
Institute of Theoretical Informatics
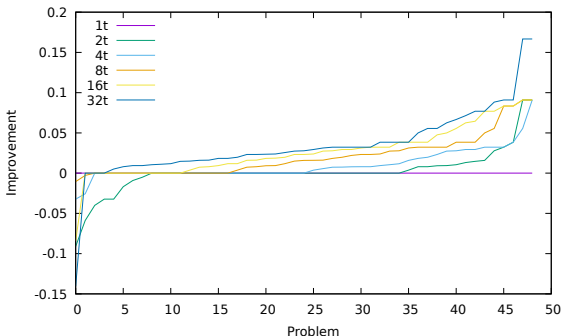
Our Parallel GCP-solver
1st Approach: The pure portfolio approach

- The agents run the GCP solver with different parameter combinations.
- After collecting the solutions found by each agent, the search takes the coloring of the minimum size as the result.

Our Parallel GCP-solver
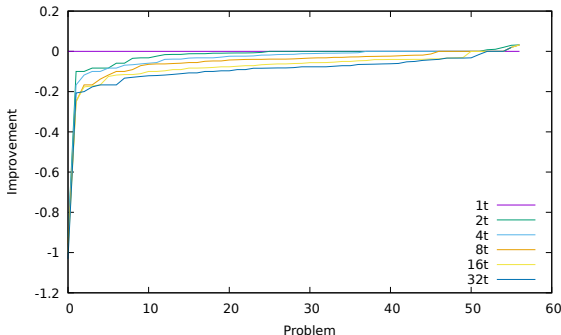
2nd Approach: Forced color reducing

- This approach is based on the pure portfolio approach.
- The agents share the minimum size.
- One agent has already found a k-coloring and broadcasts it.
- With this notification, all agents search for a legal $k - 1$ coloring.

Guangping Li:
Solving the Graph Coloring Problem with Cooperative Local Search

KIT Department of Informatics
Institute of Theoretical informatics
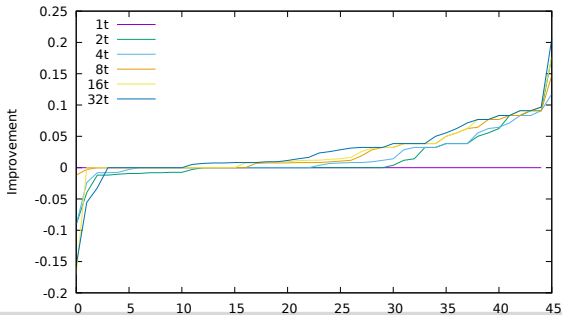
Our Parallel GCP-solver
3rd Approach: Tabu sharing

- A tabu list records the search path to avoid short-term cycling.
- The agents share the "traps" of local search loops.

Guangping Li:
Solving the Graph Coloring Problem with Cooperative Local Search

KIT Department of Informatics
Institute of Theoretical informatics

Our Parallel GCP-solver

4th Approach: Statistic sharing

- To recognize long-term cycling, a *statistic matrix S* is added.
- The S_{ij} represents how many times a one-step move $[v_i, j]$ was chosen as the next step.
- The candidate with the smallest statistic value will be chosen in the next step.
- The agents use one common statistic matrix.

Outline

The vertex coloring problem
- k-VCP
- VCP

The Tabucol algorithm

Solving VCP by Tabucol
- Basic scheme
- Data structure
- Our improvements

Our Parallel GCP-solver
- Parameter combinations
- Our approaches

Conclusion

Conclusion

Our improvement:

- An algorithm solves the VCP with parallel Tabucol searches.
- The statistic matrix recognizes long-term cycling and brings improvement to our algorithm.
- Certain information exchange (minimum size, statistic matrix) can improve the performance of the parallel search.

Comparison of our GCP-solver with DSATUR, PASS, TRICK:

- 50 of 68 (73%) benchmark graphs get best results with our solver.
- 20 of 68 (29%) benchmark graphs get unique best results with our solver.

Further work

- Using different search strategies
- Using different cooperation strategies
- Using different algorithms in agents

Guangping Li:
Solving the Graph Coloring Problem with Cooperative Local Search