SAT-powered heuristics for very large instances

Satisfiability: Theory, Practice, and Beyond Reunion
Berkeley
June 16th, 2022

André Schidler & Stefan Szeider

ALGORITHMS AND COMPLEXITY GROUP
TU Wien, Vienna, Austria

Der Wissenschaftsfonds.
WIENER WISSENSCHAFTS-, FORSCHUNGS- UND TECHNOLOGIEFONDS

SAT Encodings for Combinatorial Problems

Motivation

- Modern SAT/MaxSMT/PB/SMT solvers can tackle large instances.
- E.g., largest encoding from recent twin-width encoding: 2.5 Mio. variables, 21 Mio. clauses.
- Encodings and/or search-space can become prohibitively large, solutions:
  - A better encoding or more search-space restrictions.
  - A lazy encoding.
  - A heuristic.

Today's topic

SAT-Based Local Improvement (SLIM)

- Improving state-of-the-art heuristics with the help of a SAT solver.
- Successfully used for:
  - Treewidth
  - Branchwidth
  - Treedepth
  - Graph Coloring
  - Bayesian Network Structure Learning
  - Decision Tree Induction
  - ...
- Goal of this talk:
  - Conceptual introduction to SLIM.
  - Use cases showing diversity and effectiveness.

SAT-Based Local Improvement (SLIM)

SAT-Based Local Improvement (SLIM)

General Idea

- Metaheuristic that embeds a SAT solver into a heuristic algorithm.
- Starts from a heuristic solution and repeats the following steps until stopped or all possible options have been tried:
  1. Extract a smaller *local instance*.
  2. Improve this local instance using a SAT solver.
  3. Combine the local solution with the original–global—solution.
- Building blocks
  - Local instance and SAT encoding.
  - Budget limiting local instance size.
  - Search Strategy.

SAT-Based Local Improvement (SLIM)

Local Instance

- Representation of a small part of the original instance.
- Any solution must be integrable with the global solution.
- Improvements in the local instance must transfer to the global solution.
- The local instance must be efficiently encodable in propositional logic.

SAT-Based Local Improvement (SLIM)

Budget

- Measures the size of the local instance.
- Should correlate with solving time.
- Usually very rough estimate that captures most satisfiable cases.
- Other cases are handled via a timeout for the SAT solver.

SAT-Based Local Improvement (SLIM)

Search Strategy

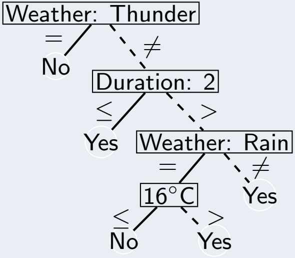
- How to construct local instances: which of the available parts to pick.
- Which local instances to pick, if there are multiple possibilities.
- In which order.
- Apply different parameterizations for different local instances, like different timeouts.

Decision Tree Induction

Decision Tree Induction

Example

°C	Weather	Day	Length	Hike
29.1	Sunny	Sat	3 h	Yes
22.3	Thunder	Mon	2 h	No
21.5	Rain	Thu	1 h	Yes
23.7	Rain	Fri	3 h	Yes
14.3	Rain	Wed	4 h	No
14.7	Sunny	Tue	3 h	Yes

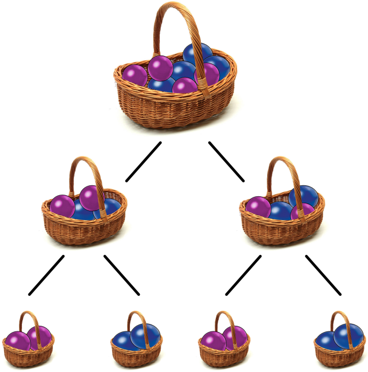


Decision Tree Induction

Problem

Find a decision tree such that:

- The decision tree correctly classifies all the samples.
- The decision tree has minimum depth/size.



Decision Tree Induction

Local Instance

- A sub-tree rooted at an internal node is a decision tree for the associated samples.
- We can therefore select small sub-trees and easily create a new classification instance from it.
- Whenever we can find a smaller sub-tree for the local instance, we can replace it at the sub-tree's root.
- This concept can be extended to sub-graphs, i.e., not complete sub-trees.

Decision Tree Induction

Budget

- The best predictor for solving time is the depth of the sub-tree.
- For each possible depth, the budget defines a maximum number of samples.
- We use a target solving time of a few minutes.

Decision Trees Induction

Search Strategy

- Search strategy for decision trees is straight forward.
- Select sub-tree of large depth/size.
- Prefer sub-trees with fewer associated samples.
- This prefers areas that lead to overfitting.

Decision Trees Induction

Results

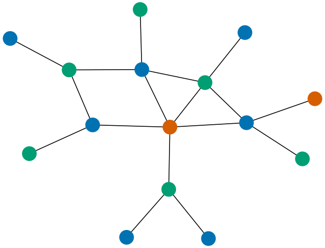
Instance	Samples	Depth start	Depth end	Size start	Size end	Accuracy start	Accuracy end
bank	3617	28	18	664	561	0.87	0.87
ccdefault	24000	65	50	7000	6998	0.73	0.73
hiv schilling	2618	71	26	5240	523	0.86	0.86
ida	60000	30	26	1433	887	0.99	0.99
splice	2552	30	15	977	260	0.91	0.89

Graph Coloring

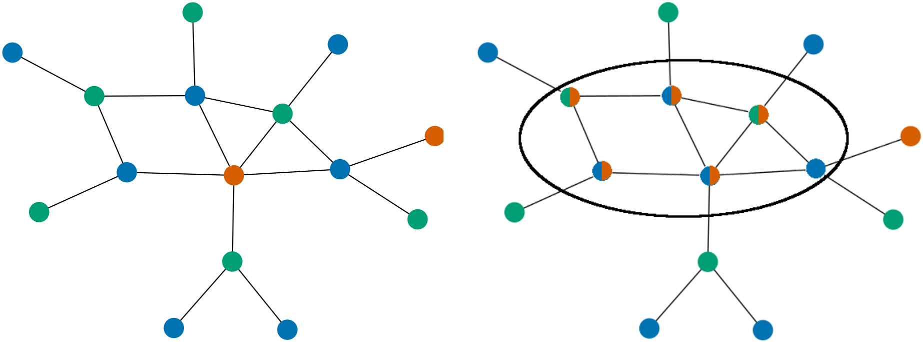
Graph Coloring

Problem

- Given an undirected graph, assign each vertex a color avoiding monochromatic edges.
- Minimize the number of colors.
- This year's SoCG Challenge created new hard instances.



Graph Coloring - Local Instance



Graph Coloring

Budget

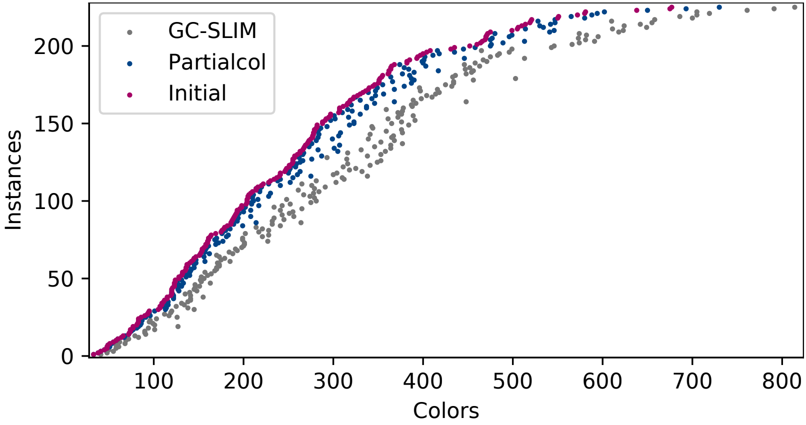
- Limit the number of vertices in the local instance.
- We use a target solving time of a few seconds.
- UNSAT results take comparatively very long.

Graph Coloring

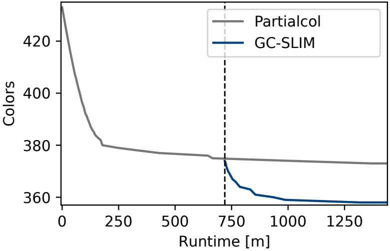
Search Strategy

- Enhances Partialcol [Blöchliger & Zufferey 2008]
- Remove a color and uncolor the corresponding vertices.
- Iteratively color one uncolored vertex.
- Construct the local instance starting with the selected vertex and adding several colors from the neighborhood.
- Iteratively extend the local instance by adding more colors and corresponding vertices.
- The SAT solver now tries to find a (list-)coloring that colors the selected vertex and minimizes the number of uncolored vertices.

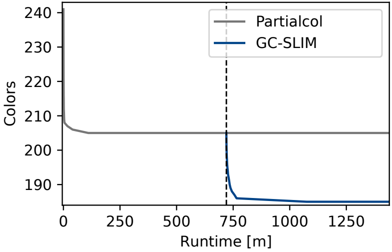
Results Overall



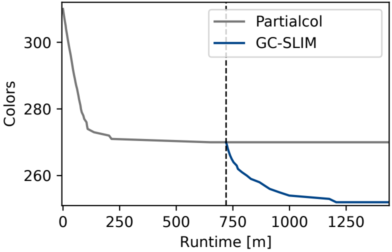
Results - Selected Instances 24-hours



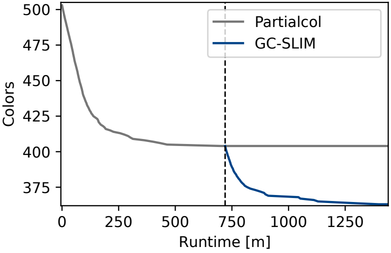
(a) scatterreecn73116



(b) scatterrvispecn17968



(c) scattervisp73369



(d) scattervispecn74166

Conclusion

Summary

- Metaheuristic that can extend known heuristics either as a post-processing step (decision trees) or as an integration (graph coloring).
- Consists local instance definition, budget, and search strategy.
- Works very well, often complementary when used as post-processing.