Karlsruhe Institute of Technology

Fine-grained Algorithm Design and Engineering

Solver 101
08.11.2024

Overview

- why we do algorithmic challenges
  - how challenges work
- what you will learn
- how to structure a solver
- test instances
- organization

Why Algorithmic Challenges?

- theoretical research focusses on worst case analysis
  - or special cases
- in practice, problems are solvable efficiently
  - even without special cases

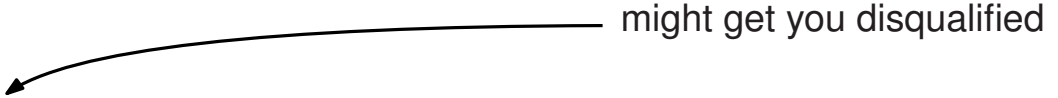
at least more efficiently than suspected

How well do theoretic approaches work?
Can we find new ones?

- better understand engineers
- baseline: beat general purpose solvers!

Challenge Framework

- differences based on specific challenge
- around 10 to 100 participating teams
- challenge announcement, publication of partial test set
- solver submission until deadline, running scoreboard
- evaluation on (additional) private instances
  - similar to public instances
  - scoring based on feasibility and optimization criterion
  - sometimes: points for solver idea



- winning solvers: invitation to associated conference?

In this course, you will ...

- ... work on more real-world problems
- ... learn approaches to tackle general algorithmic problems
- ... differentiate between theoretic results and practical possibilites
- ... engineer a solver from scratch
- ... apply standard algorithmic techniques and learn about their limitations ...
- ... use heuristics and general purpose solvers
- ... learn to evaluate algorithmic approaches
- employers like algorithmic challenges

Solver 101

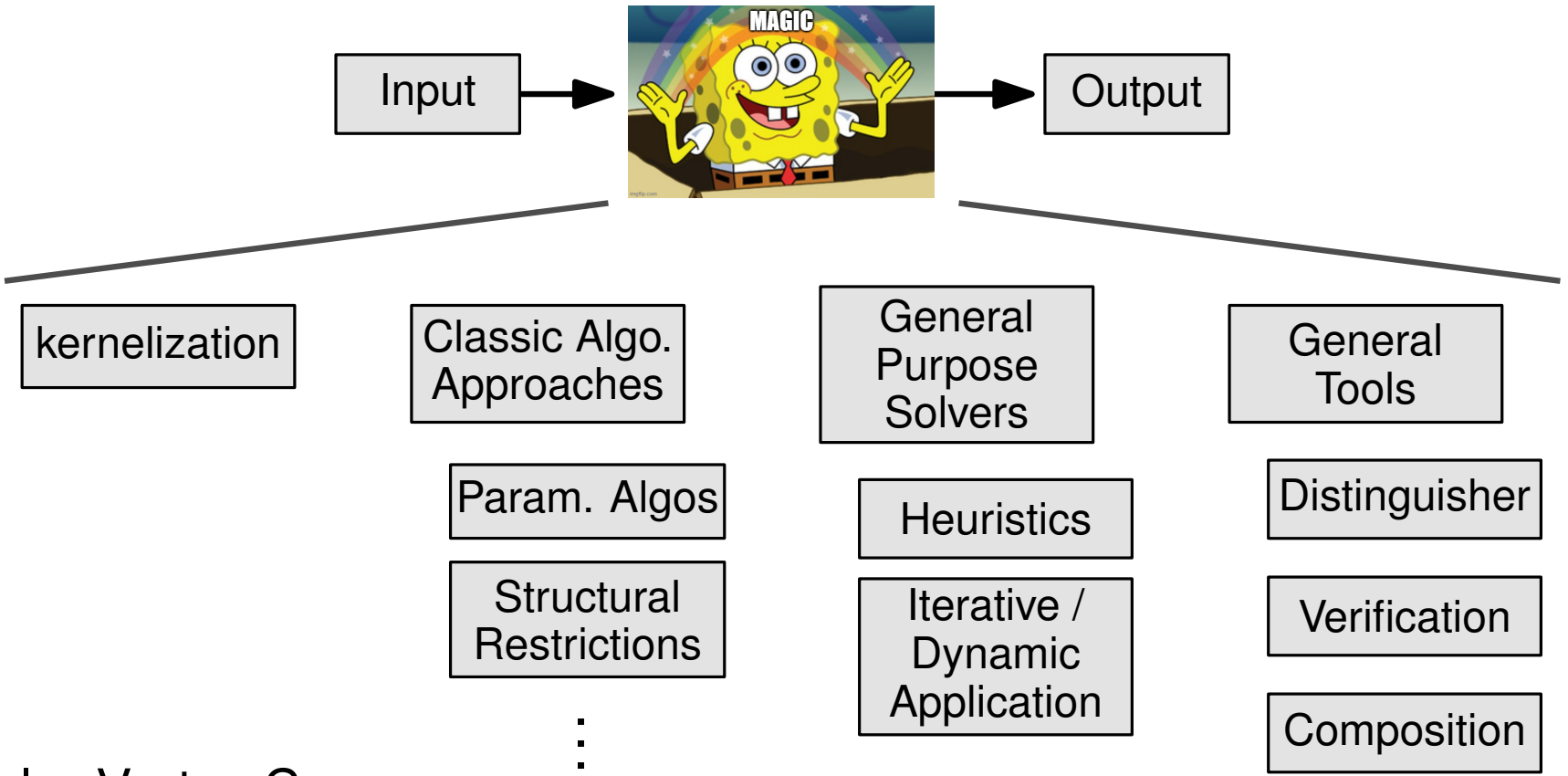
- you will change large portions of your code
  - employ good engineering practises
    - Encapsulation
    - Comments
    - Testing
- we might want to understand your code

may use existing solvers

- Disclaimer: Non-exhaustive, personal preference, only basic ideas

- have your own reasonable internal format
  - input/output might change depending on source format, visualization, etc.

Where the Magic Happens



- Example: Vertex Cover
- Want specific material? Ask us!

Techniques

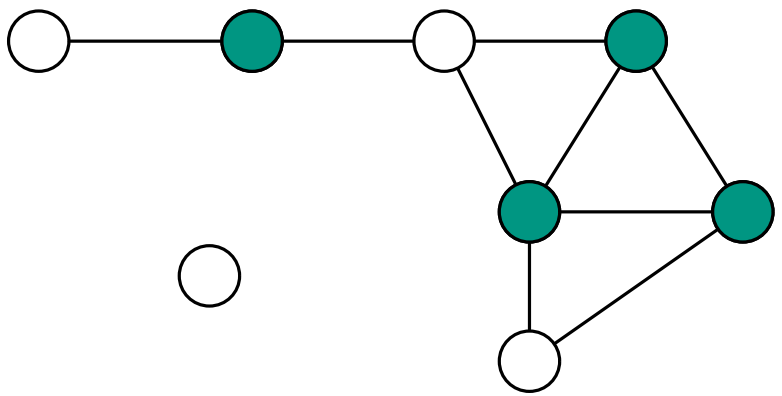
kernelization

Classic Algo. Approaches

General Purpose Solvers

General Tools

- simplify your problem!
  - less vertices, edges, solution size, etc.
  - (try to) stay away from introducing new problems (weights, etc.)



- low vertex degree (0, 1, 2, ...)
- Buss-type arguments
- neighborhood-type arguments
- see Param. Alg. lecture [1]

not necessary for challenge
but for understanding previous solvers

- unreducible instances called kernel
- very important for many solvers

[1] *https://scale.iti.kit.edu/teaching/2024ws/param_algo/*, lecture 1 (german only)

Techniques

Param. Algos

Structural Restrictions

- see param. alg. lecture

https://scale.iti.kit.edu/teaching/2024ws/param_algo/

kernelization

Classic Algo. Approaches

- keep track of your parameters
  - structural parameter (e.g. treewidth, planarity), output size
- runtime exponential in parameter, polynomial in input size
  - parameter small $\Longrightarrow$ fast runtime

General Purpose Solvers

- reduce optimization to decision problem
- selected techniques:
  - branch & bound not only in parameterized setting!
  - treewidth-DP
  - ILP with few variables $\Longrightarrow$ reduction to ILP solver

General Tools

Techniques

kernelization

Classic Algo. Approaches

General Purpose Solvers

General Tools

Iterative / Dynamic Application

Heuristics

EvalMaxSat (github: FlorentAvellaneda)

FindMinHS (github: Felerius, i.a. KIT dev-ed)

GLPK (GNU Linear Programming Kit)

GUROBI OPTIMIZATION

IBM CPLEX

not in solver, ok for evaluation

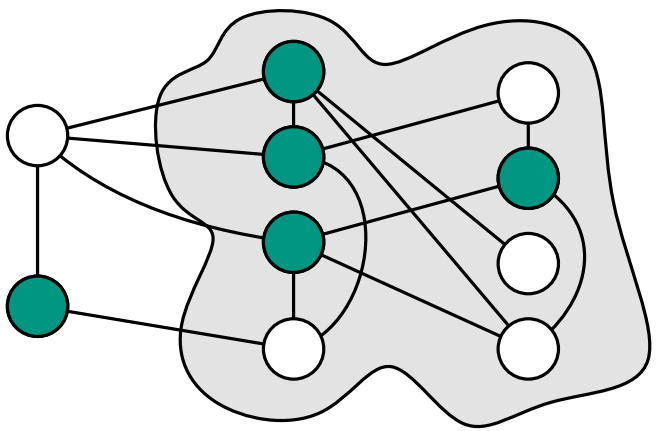
- NP-membership gives reduction to SAT, (M)ILP
  - ridiculously strongly optimized solvers
  - baseline for all practical use cases
- ask for partial solution, dynamically add constraints

LP-solving in P!

smaller instances fast

create heuristics

- Decide what to do? Educated guesses with heuristics!
  - might speed-up computation
- greedy and local search often helpful
- approximate the result
  - prune your search-tree!



Techniques

Verification | Distinguisher | Composition

kernelization

Classic Algo. Approaches

General Purpose Solvers

General Tools

- fast verification allows better use of heuristics
- many instances special enough to be easily solvable
  - what kind of special?
  - distinguishing itself complex, keep track of parameters
    - degree distribution, guess (& bound) your parameters
- runtime depending on interaction, need to iterate procedure
  - example: strong reduction rules, dynamic reduction to ex. solvers, heuristically making decisions for next iteration
- circumvent worst of worst cases: add randomness
- alternative formulation of the problem?

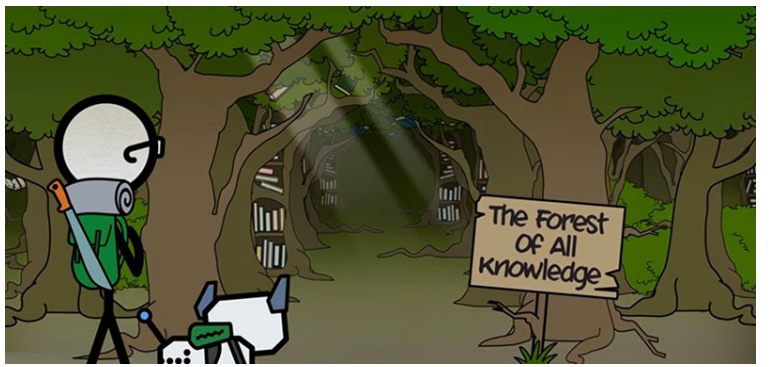
Test Instances

- look at previous write-ups
  - public test instances
  - real-world instances — SNAP project (Stanford Large Network Dataset Collection)
  - randomly generated instances (often hard to solve)
  - other instances harder to predict (and generate)
- course-specific: instances encoding other problems — (fine-grained) reductions?
- persistent output $\Longrightarrow$ file output
  - evaluate your solutions $\Longrightarrow$ feasible?
- visualize your instances! — e.g. graphviz
- scripts to run/evaluate large batches

Miscellaneous

- start small: have your boilerplate code running
  - input/output, validation, testing (metrics or visuals)
- focus on easy instances: solve them fast
  - allow unsolved outputs, look at unsolved parts
  - collect hard instances, investigate them
  - find approaches to next portion of instances
- switch to general solver should not happen too early
  - often tradeoff between simplicity and generality
  - interesting insights often found for specific instances, generalized later
- look at existing algorithmic ideas first

Be Curious!



original by CGP Grey

Organizational

- repository, please invite us
  - all code, test instances and output, visualizations, etc.
  - scripts for building, evaluation, ...

use suitable libraries!

- readme
  - explain usage: how to build and run solver, how to evaluate
  - contribution (bullet points)
    - what was taken from which source, who worked on what part
- tbd: How To Report in December (around 15.12) – announcement via email

Q&A now!