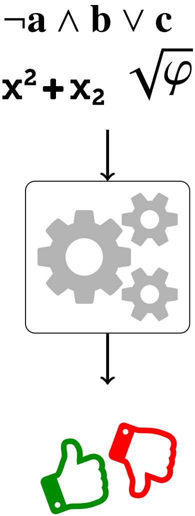
Heuristics in SMT Solving: To Learn or not to Learn?

Erika Ábrahám
with Gereon Kremer

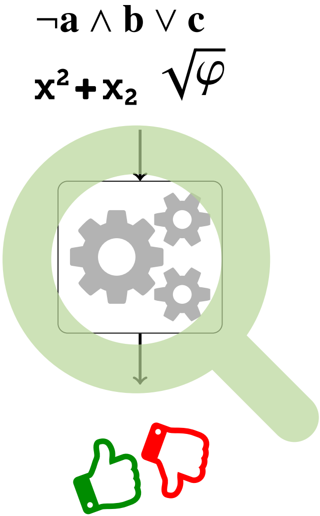
RWTH Aachen University, Germany

ICMS'18, 26 July 2018

What is this talk about?



What is this talk about?



What is the role of heuristics in SMT solving?

Are there potentials for learning?

The satisfiability problem

Propositional logic

Formula: $(a \vee \neg b) \wedge (\neg a \vee b \vee c)$

Satisfying assignment: $a = \textit{true}$, $b = \textit{false}$, $c = \textit{true}$

It is perhaps the most well-known NP-complete problem [Cook'71].

The satisfiability problem

Propositional logic

Formula: $(a \vee \neg b) \wedge (\neg a \vee b \vee c)$

Satisfying assignment: $a = \mathit{true}$, $b = \mathit{false}$, $c = \mathit{true}$

It is perhaps the most well-known NP-complete problem [Cook'71].

Non-linear real algebra (NRA)

Formula: $(x - 2y > 0 \vee x^2 - 2 = 0) \wedge x^4 y + 2x^2 - 4 > 0$

Satisfying assignment: $x = \sqrt{2}$, $y = 2$

There are some hard problem classes... non-linear integer arithmetic is even undecidable.

SAT solving for propositional logic

SAT solvers are full of heuristics, perhaps the two most successful ones being:

- dynamic variable ordering (VSIDS)
- resolution driven by enumeration/propagation search, learning resolvents (CDCL), forgetting learnt clauses

SAT solving for propositional logic

SAT solvers are full of heuristics, perhaps the two most successful ones being:

- dynamic variable ordering (VSIDS)
- resolution driven by enumeration/propagation search, learning resolvents (CDCL), forgetting learnt clauses

Research threads on machine learning in SAT solving: predict satisfiability, variable ordering, determine values for decision variables, clause forgetting,...

SAT solving for propositional logic

SAT solvers are full of heuristics, perhaps the two most successful ones being:

- dynamic variable ordering (VSIDS)
- resolution driven by enumeration/propagation search, learning resolvents (CDCL), forgetting learnt clauses

Research threads on machine learning in SAT solving: predict satisfiability, variable ordering, determine values for decision variables, clause forgetting,...

Problem: how to extract characteristic information for training sets?

(Full/less) lazy SMT solving

(Full/less) lazy SMT solving

φ quantifier-free FO formula

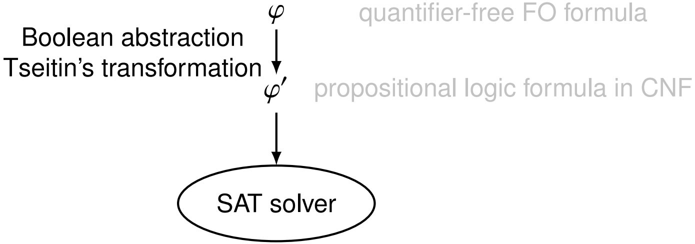
(Full/less) lazy SMT solving

φ quantifier-free FO formula

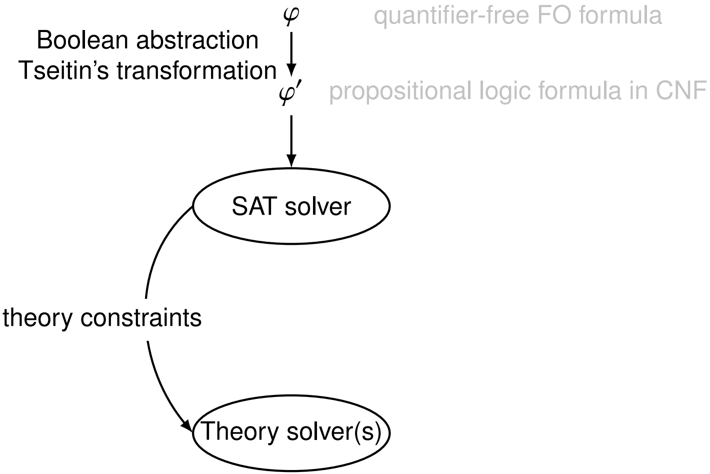
Boolean abstraction
Tseitin's transformation
$\downarrow$

φ' propositional logic formula in CNF

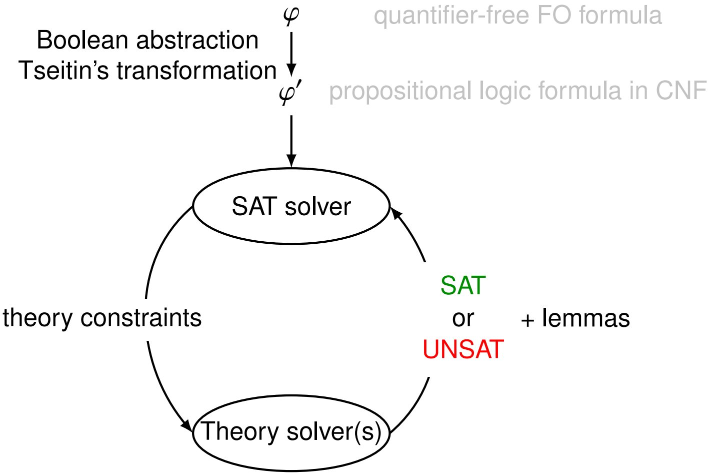
(Full/less) lazy SMT solving



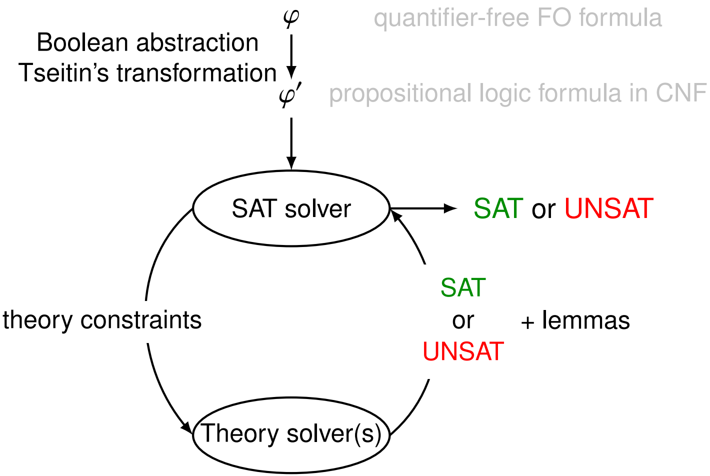
(Full/less) lazy SMT solving



(Full/less) lazy SMT solving



(Full/less) lazy SMT solving



General SMT solving heuristics

General SMT solving heuristics

- A particularly interesting case: variable ordering.

  In SAT solving, VSIDS is very successful, but the variable odering at the Boolean level is not connected to the theory solver.

  Also the variable ordering in arithmetic theory solvers is usually statically determined, independently of the problem at hand.

General SMT solving heuristics

- A particularly interesting case: variable ordering.

  In SAT solving, VSIDS is very successful, but the variable odering at the Boolean level is not connected to the theory solver.

  Also the variable ordering in arithmetic theory solvers is usually statically determined, independently of the problem at hand.

- Other cases: preprocessing, restarts, constraint ordering, value ordering, ...

Some SMT solvers for arithmetic theories

- AProVE (RWTH Aachen University, Germany)
- CVC4 (New York and Iowa, USA)
- MathSAT 5 (FBK, Italy)
- MiniSmt (University of Innsbruck, Austria)
- Boolector (JKU, Austria)
- SMT-RAT (RWTH Aachen University, Germany)
- veriT+Redlog (CNRS Inria, France and MPI Informatics, Germany)
- Z3 (NYU, Microsoft Research, USA)
- Yices 2 (SRI International, USA)
- ...

Our SMT-RAT library SAT'12, SAT'15

SMT solver

Strategic composition of SMT-RAT modules

SMT real-algebraic toolbox

collection of solver modules

CArL

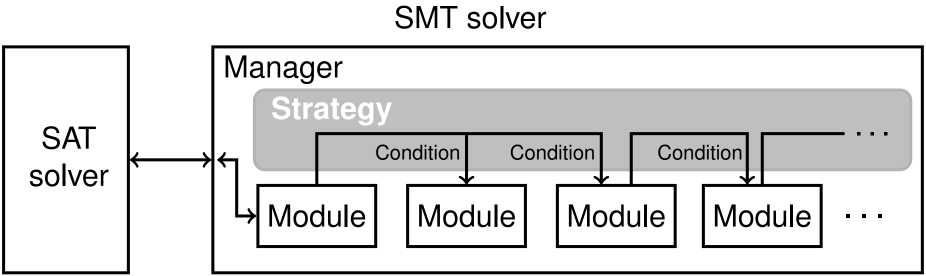
real-arithmetic computations

gmp, Eigen3, boost

- MIT licensed source code: `github.com/smtrat/smtrat`
- Documentation: `smtrat.github.io`

Strategic composition of solver modules in SMT-RAT

- Strategy: directed graph over modules with guarded edges
- Guard: may talk about the formula forwarded to backends
- Backend-calls: passed to all enabled successors → parallelism



Solver modules in SMT-RAT

CArL library for basic arithmetic datatypes and computations [NFM'11, CAI'11, Sapientia'18]

Basic modules

- SAT solver
- CNF converter
- Preprocessing/simplifying modules

Non-algebraic decision procedures

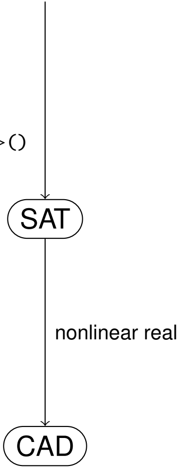
- Bit-vectors
- Bit-blasting
- Equalities and uninterpreted functions
- Pseudo-Boolean formulas
- Interval constraint propagation

Algebraic decision procedures

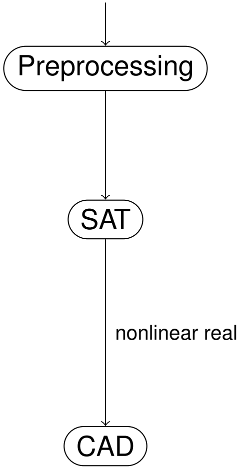
- Fourier-Motzkin variable elimination
- Simplex
- Subtropical satisfiability
- Gröbner bases [CAI'13]
- MCSAT (FM,VS,CAD)
- Cylindrical algebraic decomposition [CADE-24, SC^2'17, PhD Loup, PhD Kremer]
- Virtual substitution [FCT'11, SC^2'17, PhD Corzilius]
- Generalized branch-and-bound [CASC'16]
- Cube tests

SMT-RAT strategies

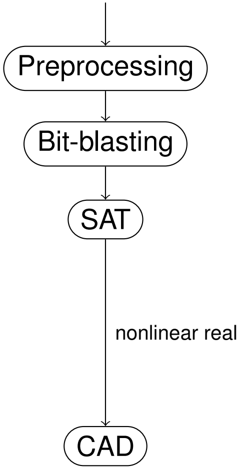
```
class myStrategy: public Manager {
  myStrategy(): Manager() {
    setStrategy(
      addBackend<SATModule<SATSettings>>(
        addBackend<CADModule<CADSettings>>()
      )
    );
  }
};
```



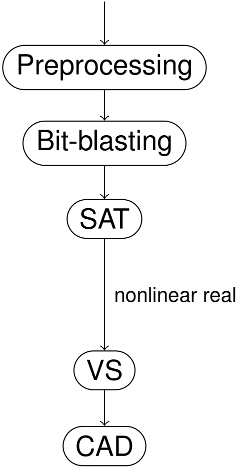
SMT-RAT strategies



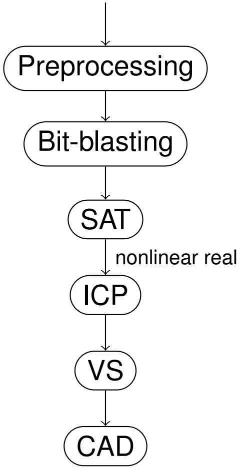
SMT-RAT strategies



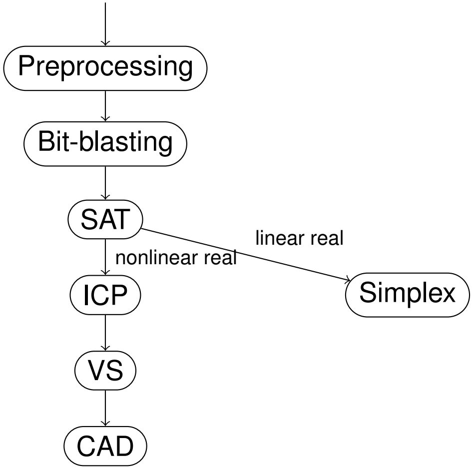
SMT-RAT strategies



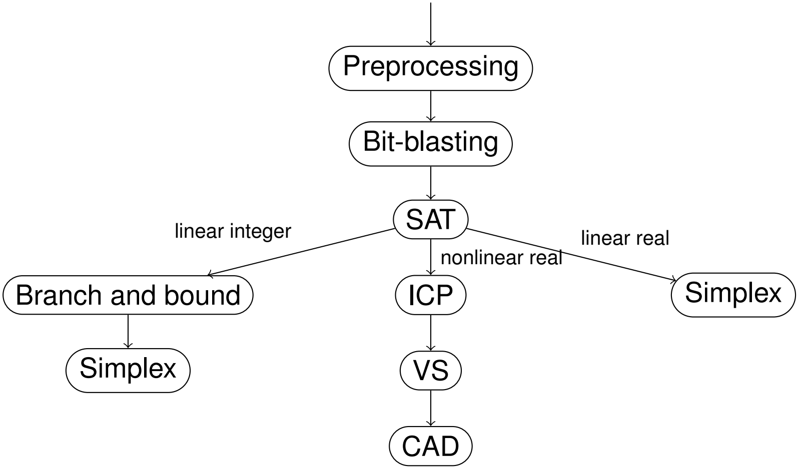
SMT-RAT strategies



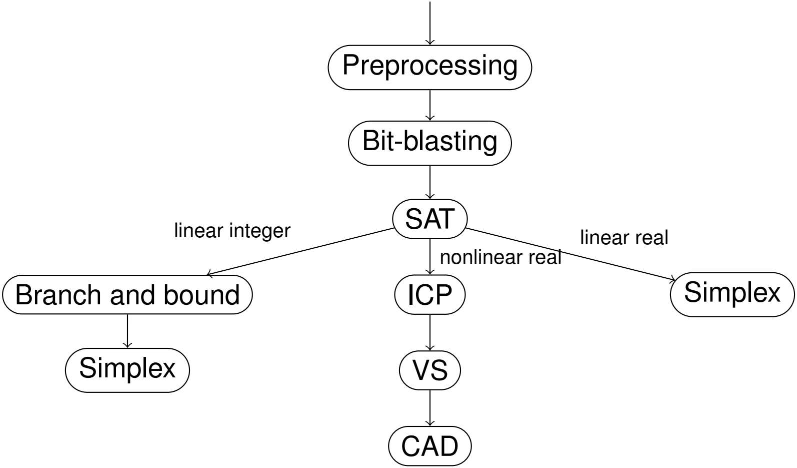
SMT-RAT strategies



SMT-RAT strategies

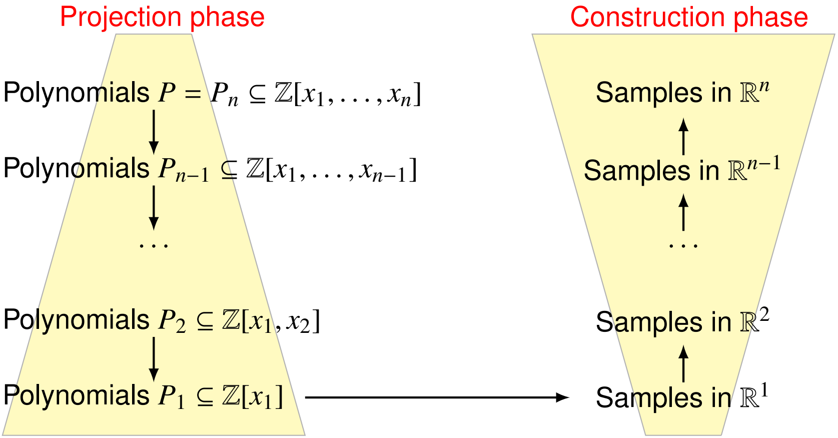


SMT-RAT strategies

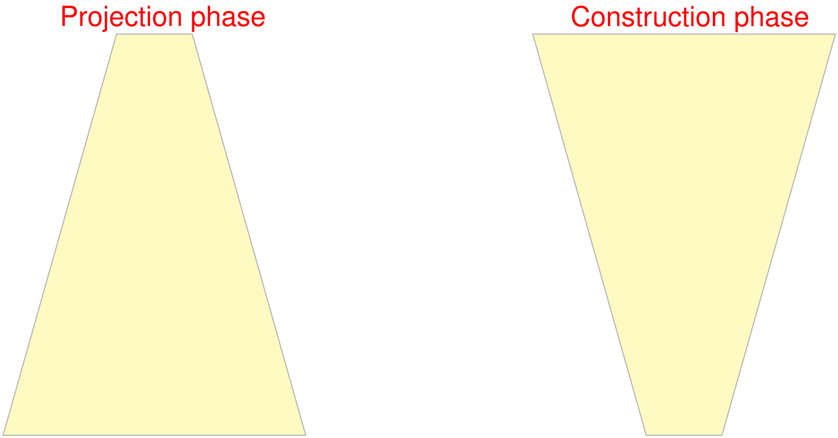


Which strategy to use for which problem?

The cylindrical algebraic decomposition (CAD) method



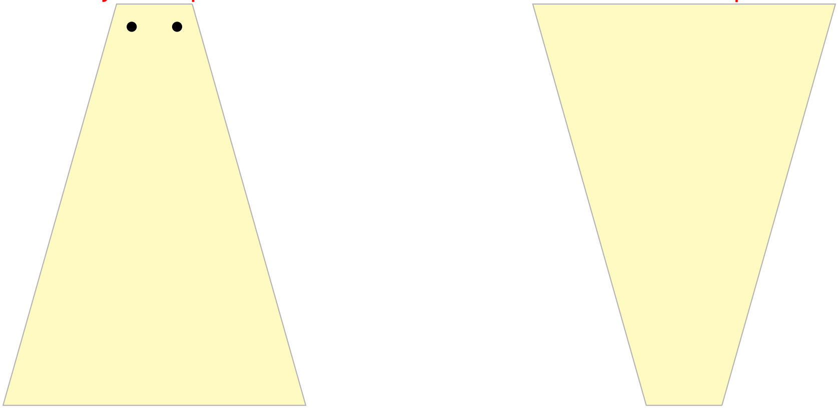
Heuristics in the CAD method



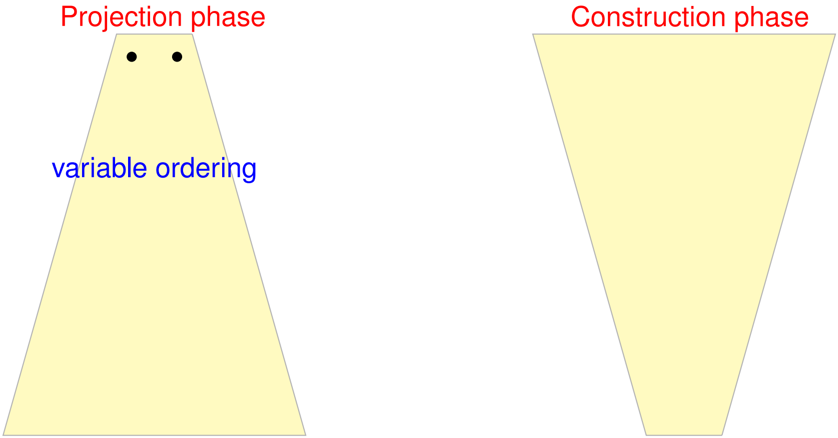
Heuristics in the CAD method

Projection phase

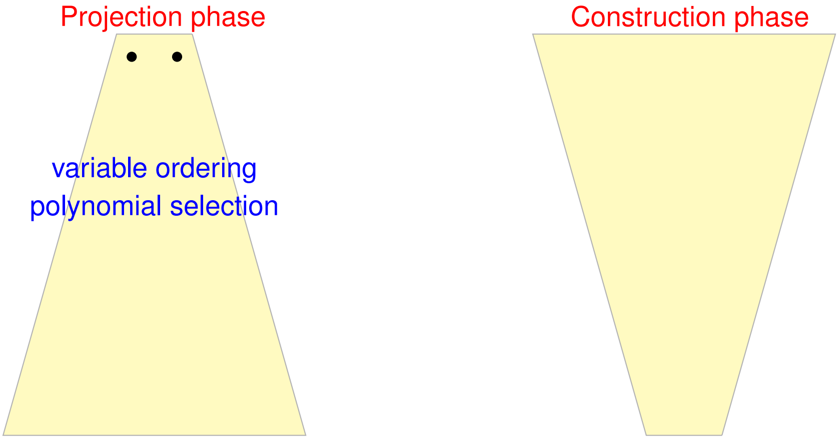
Construction phase



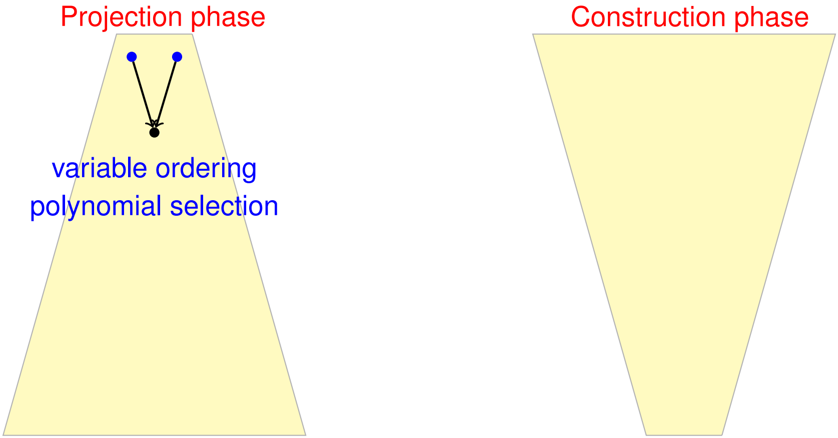
Heuristics in the CAD method



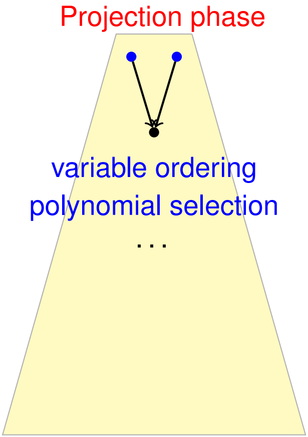
Heuristics in the CAD method

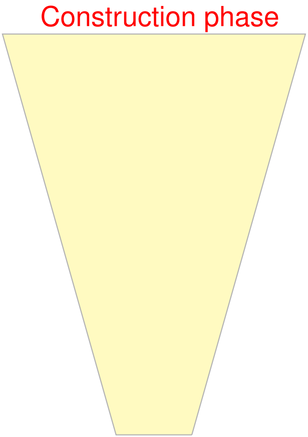


Heuristics in the CAD method

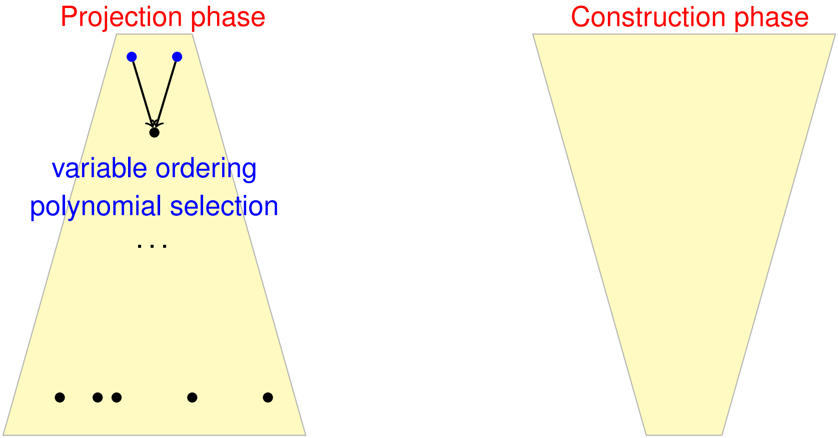


Heuristics in the CAD method

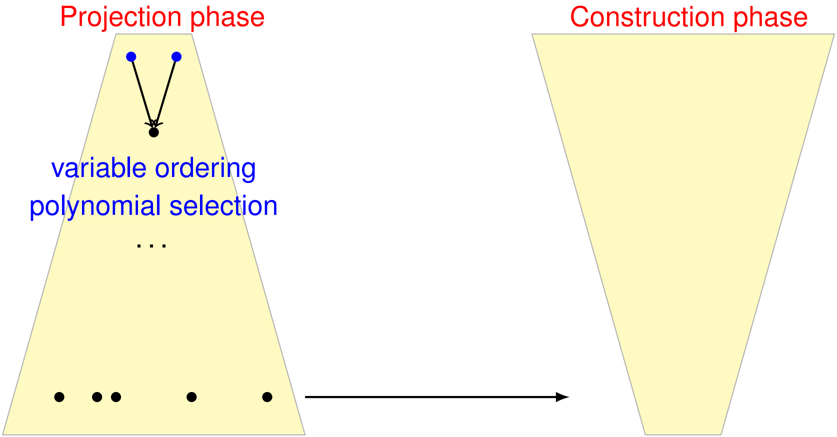




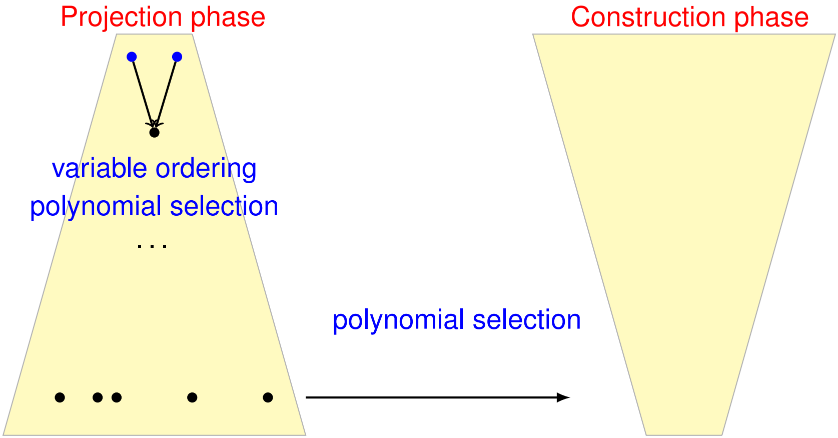
Heuristics in the CAD method



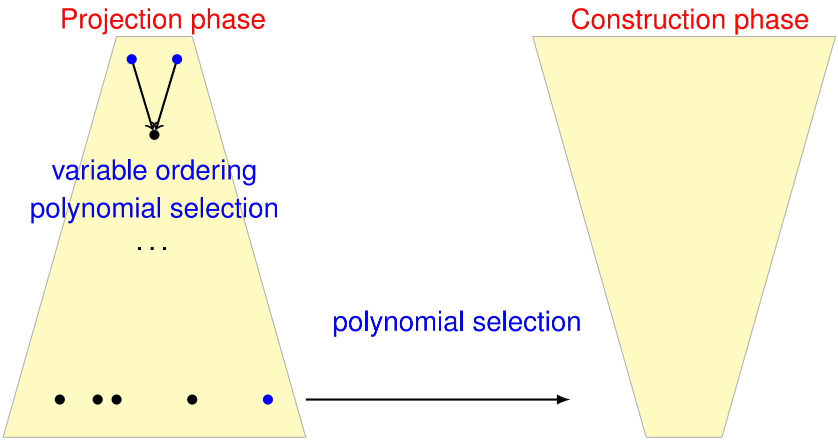
Heuristics in the CAD method



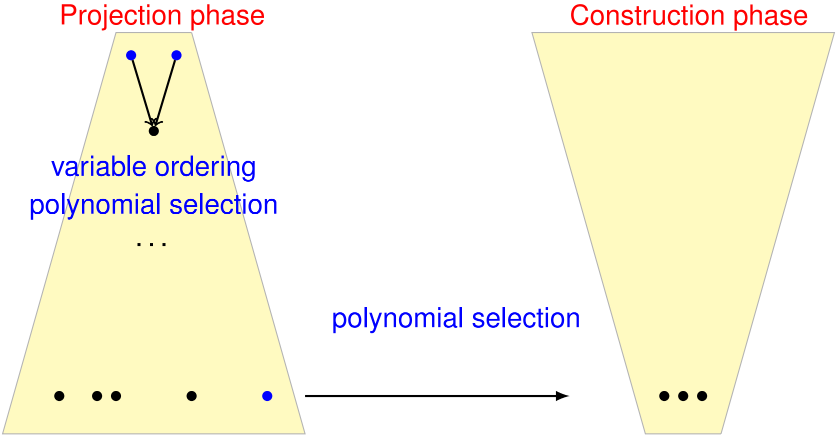
Heuristics in the CAD method



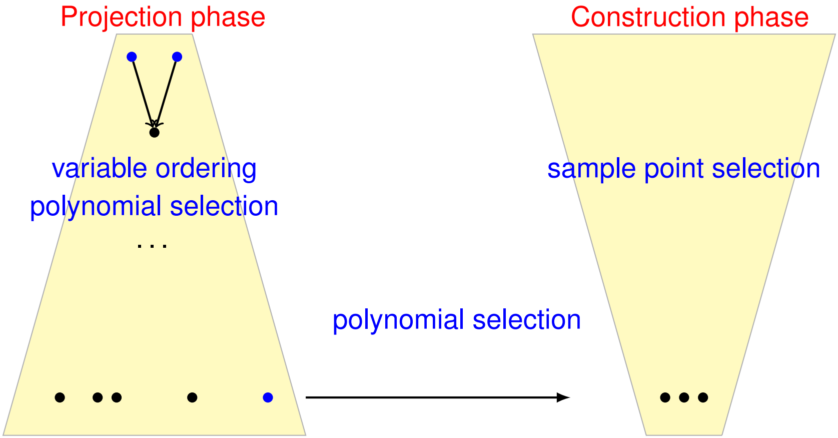
Heuristics in the CAD method



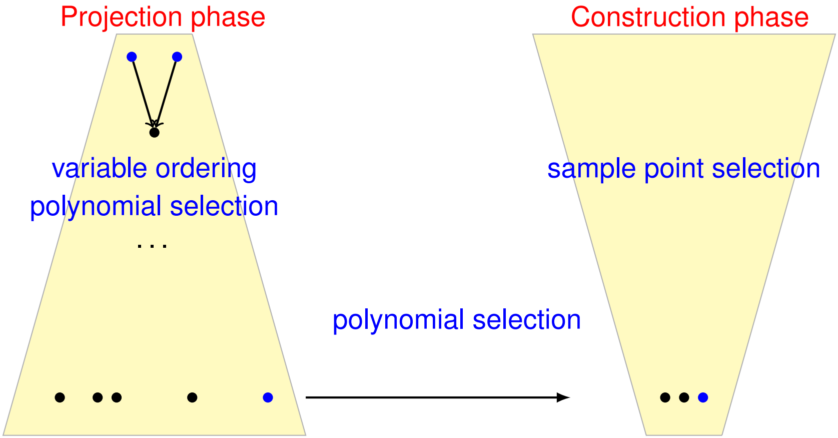
Heuristics in the CAD method



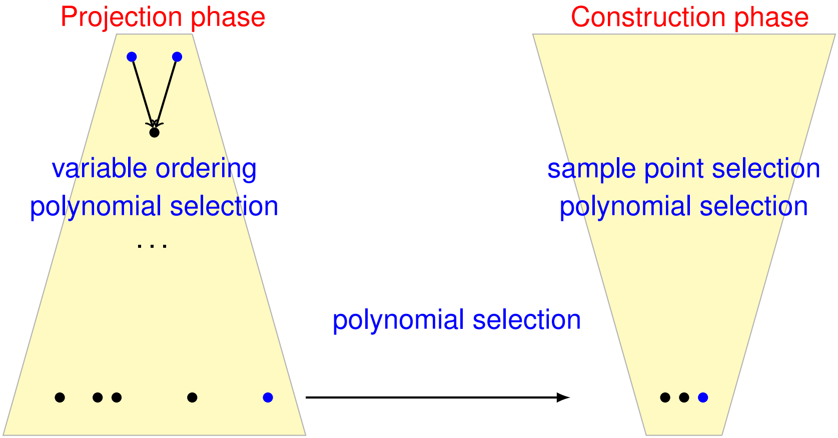
Heuristics in the CAD method



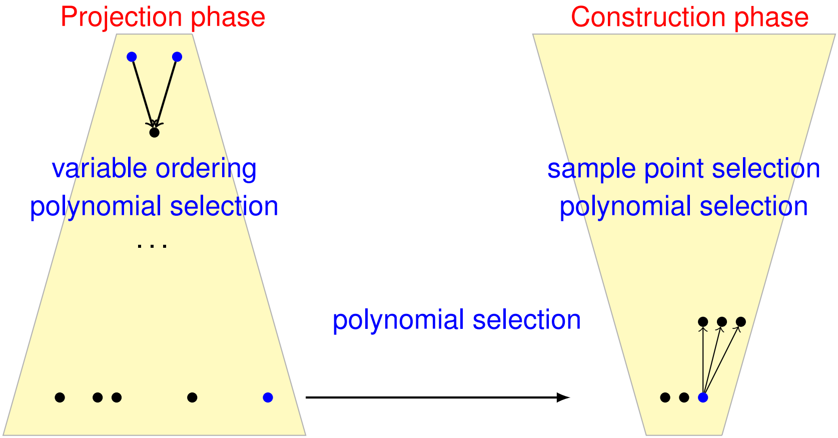
Heuristics in the CAD method



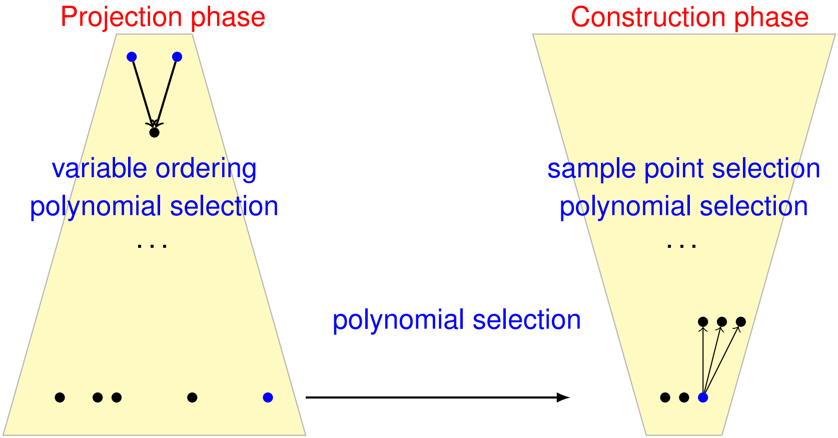
Heuristics in the CAD method



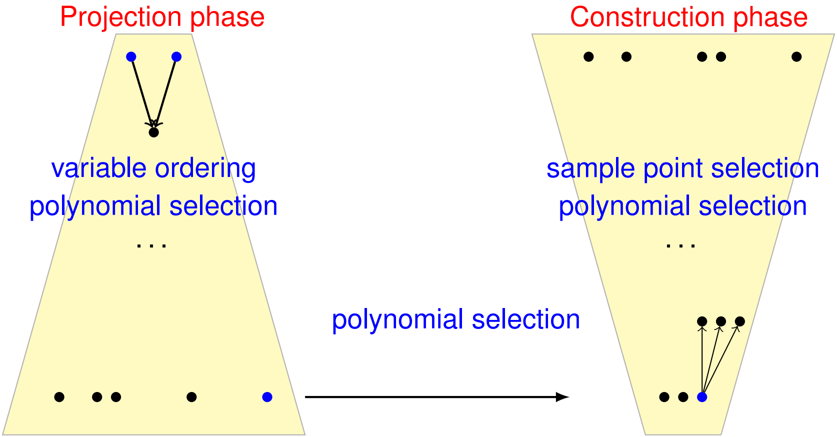
Heuristics in the CAD method



Heuristics in the CAD method



Heuristics in the CAD method



Experimental results: Projection order

11354 `QF_NRA` benchmarks, TO: 60 secs

- C: complexity
- S: single projection
- P: paired projection
- L_i: level in increasing order
- L_d: level in decreasing order

Solver	solved		runtime
CAD C	8075	71.1 %	1.13
CAD SC	8074	71.1 %	1.13
CAD PC	8076	71.1 %	1.12
CAD L_iC	**8135**	71.6 %	1.28
CAD L_dC	**8135**	71.6 %	1.18

Experimental results: Sampling

11354 QF_NRA benchmarks, TO: 60 secs

Solver	solved		average runtime
CAD midpoint	8147	71.8 %	1.21
CAD int closest to midpoint	8155	71.8 %	1.19
CAD smallest int	**8158**	71.9 %	1.22
CAD largest int	8144	71.7 %	1.20
CAD int close to 0	8154	71.8 %	1.20
CAD int far from 0	8146	71.7 %	1.21

Experimental results: Lifting

11354 QF_NRA benchmarks, TO: 60 secs

T: type (integer, rational, algebraic)

S: size

A: absolute value

L: level

Solver	solved		average runtime
CAD TSA	8118	71.5 %	1.21
CAD S	8121	71.5 %	1.22
CAD T	8138	71.7 %	1.20
CAD LTS	8143	71.7 %	1.22
CAD LT	**8144**	71.7 %	1.20

An attempt for a conclusion

- Can we draw the conclusion that these heuristics perform equally?

An attempt for a conclusion

- Can we draw the conclusion that these heuristics perform equally?

  Perhaps... but perhaps they perform differently on different problem types.

  In the latter case learning could strengthen these methods.

An attempt for a conclusion

- Can we draw the conclusion that these heuristics perform equally?

  Perhaps... but perhaps they perform differently on different problem types.

  In the latter case learning could strengthen these methods.

There are probably great potentials in learning heuristics, but a number of problems need to be solved before we can explore these possibilities.