# Analysing the Empirical Time Complexity of High-performance Algorithms for SAT and TSP

MSc Thesis Presentation

**Zongxu Mu**

Department of Computer Science
University of British Columbia
Vancouver, BC, Canada

9 September 2015

Supervisor: Prof. Holger H. Hoos
Second Reader: Prof. David Poole

# Time Complexity

Time complexity is key in theoretical CS and practical applications

- Scaling of running time as function of instance size

# Time Complexity

Time complexity is key in theoretical CS and practical applications

- Scaling of running time as function of instance size

Approaches:

- Theoretical: rigorous combinatorial analysis
  - E.g., worst case time complexity for Quicksort is $O\left(n^2\right)$

# Time Complexity

Time complexity is key in theoretical CS and practical applications

- Scaling of running time as function of instance size

Approaches:

- Theoretical: rigorous combinatorial analysis
  - E.g., worst case time complexity for Quicksort is $O\left(n^2\right)$
- Empirical: well-designed statistical analysis
  - Applicable to sophisticated (heuristic-based) algorithms, including state of the art solvers for important problems

# Time Complexity

Time complexity is key in theoretical CS and practical applications

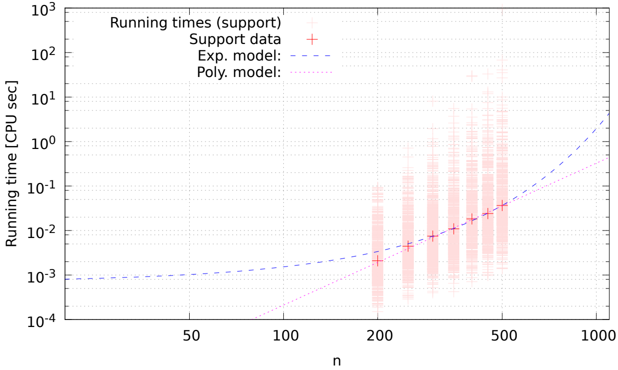
- Scaling of running time as function of instance size

Approaches:

- Theoretical: rigorous combinatorial analysis
  - E.g., worst case time complexity for Quicksort is $O\left(n^2\right)$
- Empirical: well-designed statistical analysis
  - Applicable to sophisticated (heuristic-based) algorithms, including state of the art solvers for important problems
  - Key idea: fit parametric functions on running times

# Time Complexity – Empirical Approach

Key idea: fit parametric functions on running times



# Propositional satisfiability problem (SAT)

Determining existence of interpretation satisfying Boolean formula

# Propositional satisfiability problem (SAT)

Determining existence of interpretation satisfying Boolean formula

- Literal: variable or negation of variable
- Clause: disjunction of literals
- Conjunctive normal form (CNF): conjunction of clauses

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

# Propositional satisfiability problem (SAT)

Determining existence of interpretation satisfying Boolean formula

- Literal: variable or negation of variable
- Clause: disjunction of literals
- Conjunctive normal form (CNF): conjunction of clauses

$$(x_1 \vee x_2 \vee x_3) \wedge (\neg x_2 \vee x_3 \vee \neg x_4) \wedge (\neg x_1 \vee x_2 \vee x_4)$$

First problem proved to be $\mathscr{N}\mathscr{P}$-complete [Cook, 1971]

- Intense academic interest & many practical applications
- Dramatic & sustained progress in SAT solving
  - International SAT Competitions / Challenges

# Travelling salesperson problem (TSP)

Given: cities and pair-wise distances

Objective: shortest roundtrip route to pass through each city exactly once


Figure from http://www.math.uwaterloo.ca/tsp/usa50/

# Travelling salesperson problem (TSP)

Given: cities and pair-wise distances
Objective: shortest roundtrip route to pass through each city exactly once

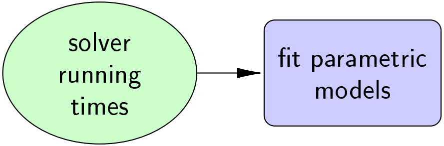
Figure from http://www.math.uwaterloo.ca/tsp/usa50/

Prominent combinatorial optimisation problem

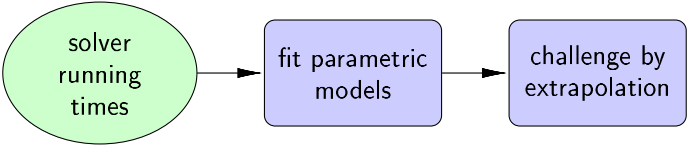
- General TSP [Garey and Johnson, 1979] and Euclidean TSP [Papadimitriou, 1977] are both $\mathcal{NP}$-hard
- Sustained academic and practical interest
- Testbed for new algorithmic ideas in combinatorial optimisation

# Outline

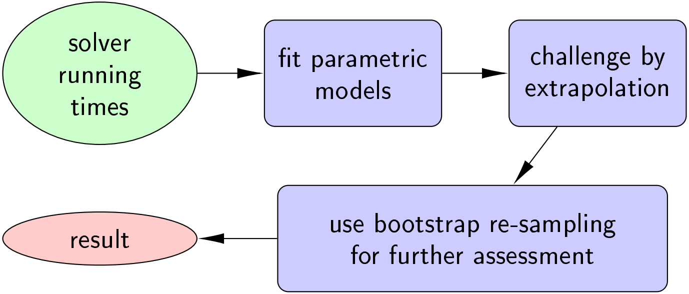
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]



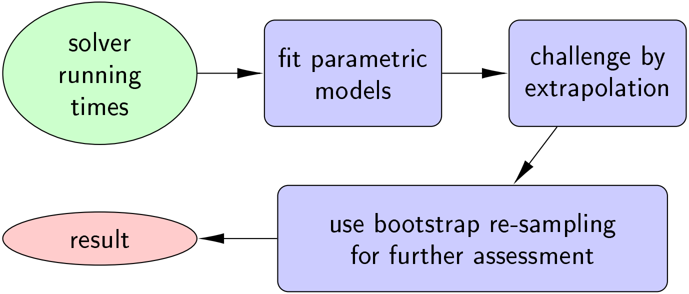
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]



# Methodology [Hoos, 2009, Hoos and Stützle, 2014]



# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

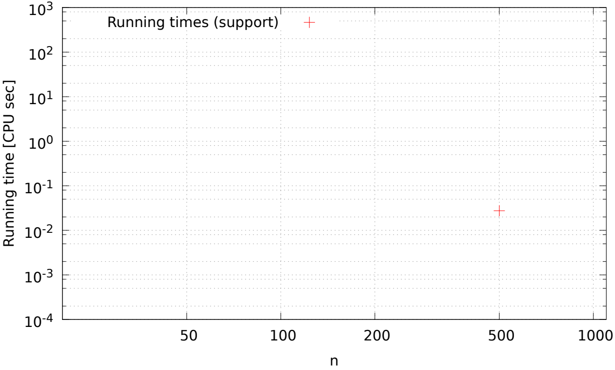



My contributions:

- Use confidence intervals of observed data to assess models
- Compare scaling models of two solvers based on confidence intervals of observed/predicted data

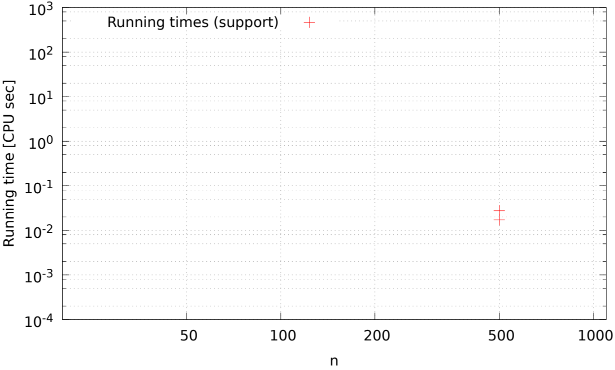
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



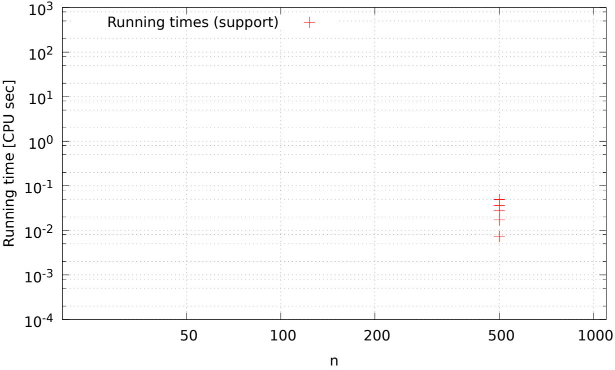
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



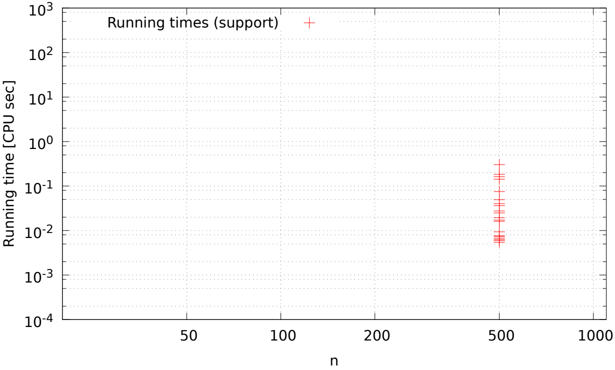
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



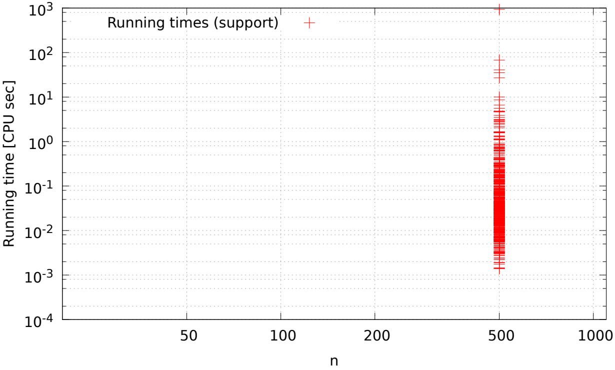
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



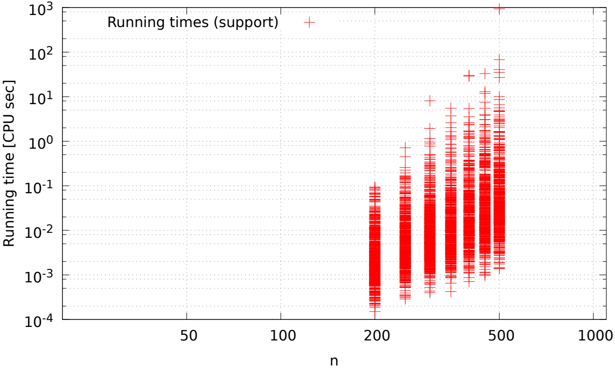
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



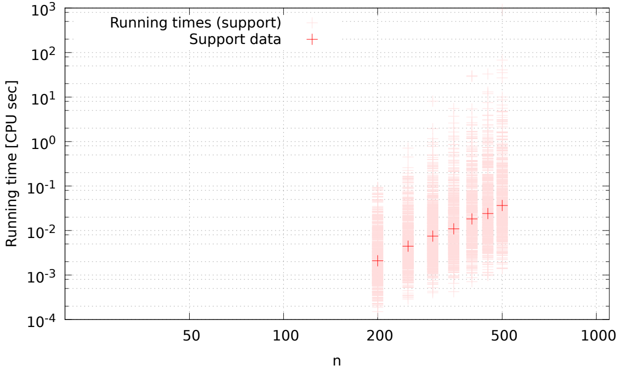
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



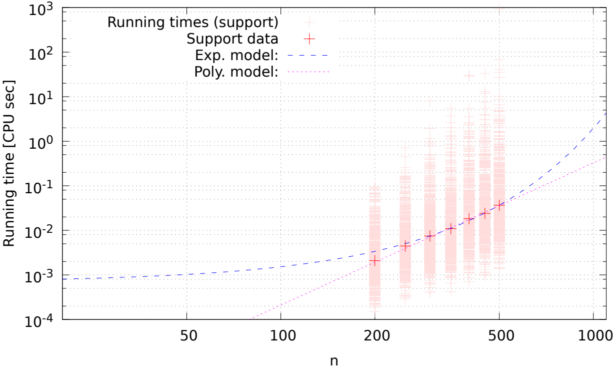
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

1. Solver running times:



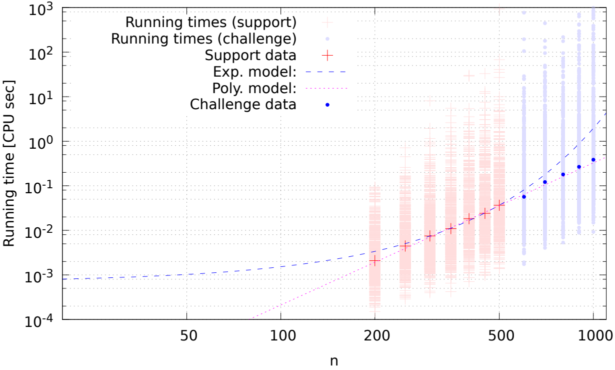
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

2. Fit parametric models:



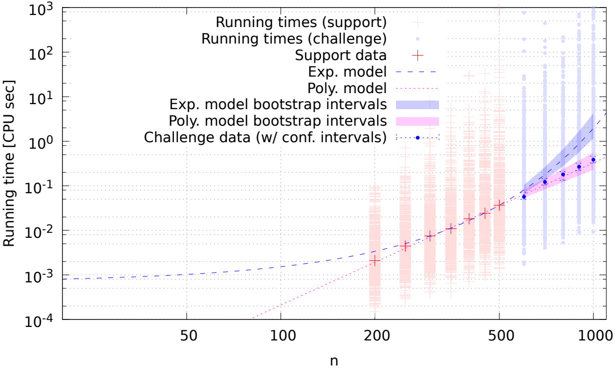
# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

3. Challenge by extrapolation:



# Methodology [Hoos, 2009, Hoos and Stützle, 2014]

4. Use bootstrap re-sampling for further assessment:



# SAT – Scientific Questions

Scientific questions:

# SAT – Scientific Questions

Scientific questions:

- How do running times of high-performance SAT solvers scale?

# SAT – Scientific Questions

Scientific questions:

- How do running times of high-performance SAT solvers scale?
- Is scaling difference between solvers significant?
  - Between SLS- and DPLL-based solvers?
  - Between two solvers of same kind?

# SAT – Scientific Questions

Scientific questions:

- How do running times of high-performance SAT solvers scale?
- Is scaling difference between solvers significant?
  - Between SLS- and DPLL-based solvers?
  - Between two solvers of same kind?
- How much faster are complete solvers solving satisfiable instances?

# SAT – Scientific Questions

Scientific questions:

- How do running times of high-performance SAT solvers scale?
- Is scaling difference between solvers significant?
  - Between SLS- and DPLL-based solvers?
  - Between two solvers of same kind?
- How much faster are complete solvers solving satisfiable instances?

Solvers:

- SLS-based: WalkSAT/SKC, BalancedZ, probSAT
- DPLL-based: kcnfs, march_hi, march_br

# SAT – Scientific Questions

Scientific questions:

- How do running times of high-performance SAT solvers scale?
- Is scaling difference between solvers significant?
  - Between SLS- and DPLL-based solvers?
  - Between two solvers of same kind?
- How much faster are complete solvers solving satisfiable instances?

Solvers:

- SLS-based: WalkSAT/SKC, BalancedZ, probSAT
- DPLL-based: kcnfs, march_hi, march_br

Problem instances:

- Phase-transition random 3-SAT [Mu and Hoos, 2015a]

# Results for SAT – Phase Transition

Soluability phase transition: 50% of random instances satisfiable

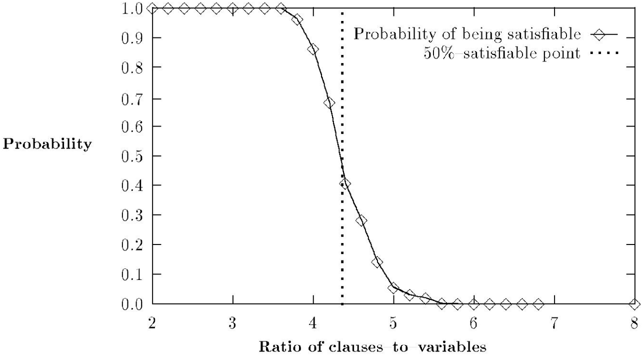



Figure from [Mitchell et al., 1992]

# Results for SAT – Phase Transition

Soluability phase transition: 50% of random instances satisfiable

- Phase transition is sharp [Cheeseman et al., 1991]
- Believed to converge to fixed threshold

# Results for SAT – Phase Transition

Soluability phase transition: 50% of random instances satisfiable

- Phase transition is sharp [Cheeseman et al., 1991]
- Believed to converge to fixed threshold

Widely studied instance distribution

- Prominent model of computational hardness in SAT and beyond
  - For DPLL-based solvers [Mitchell et al., 1992]
  - For SLS-based solvers [Yokoo, 1997]
  - ......

# SAT – Phase Transition

Best previous model [Crawford and Auton, 1996]:

$$m_c = 4.258 \cdot n + 58.26 \cdot n^{-2/3}$$

# SAT – Phase Transition

Best previous model [Crawford and Auton, 1996]:

$$m_c = 4.258 \cdot n + 58.26 \cdot n^{-2/3}$$

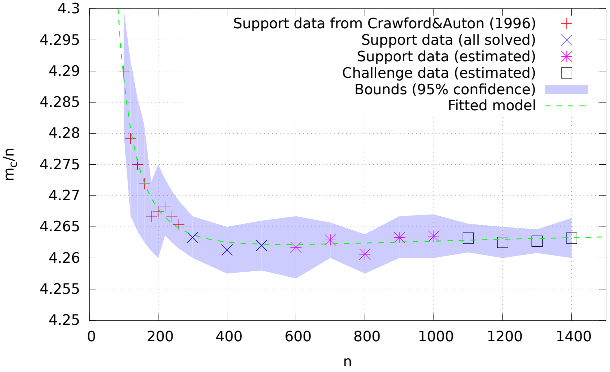
Weaknesses:

- Inconsistent with results from cavity method [Mertens et al., 2006]:

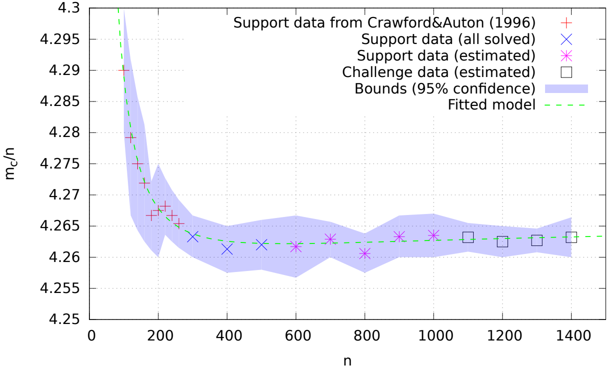
$$\lim_{n \to \infty} m_c / n = 4.26675 \pm 0.00015$$

- Under-estimates $m_c$ for larger $n$

# SAT – Phase Transition



# SAT – Phase Transition




Refined model:

$$m_c = 4.26675 \cdot n + 447.884 \cdot n^{-0.0350967} - 430.232 \cdot n^{-0.0276188}$$

# SAT – Related Work

Work on empirical scaling of:

- SLS-based solvers, e.g., Gent and Walsh [1993], Gent et al. [1997]
- DPLL-based solvers, e.g., Coarfa et al. [2003]

# SAT – Related Work

Work on empirical scaling of:

- SLS-based solvers, e.g., Gent and Walsh [1993], Gent et al. [1997]
- DPLL-based solvers, e.g., Coarfa et al. [2003]

Limitations:

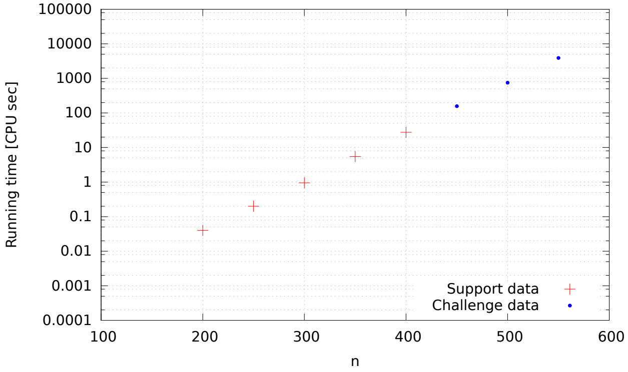
- # variable flips vs. actual running times,
  e.g., Gent and Walsh [1993], Gent et al. [1997]
- Inconclusive results, e.g., Gent and Walsh [1993]
- Simple curve fitting & vague definition of "good fit"

# Empirical Scaling Results – DPLL-based Solvers

Divide instance sets into support and challenge:

| $n$ | 200 | 250 | 300 | 350 | 400 |
|---|---|---|---|---|---|
| median | 0.040 | 0.200 | 0.950 | 5.455 | 27.580 |

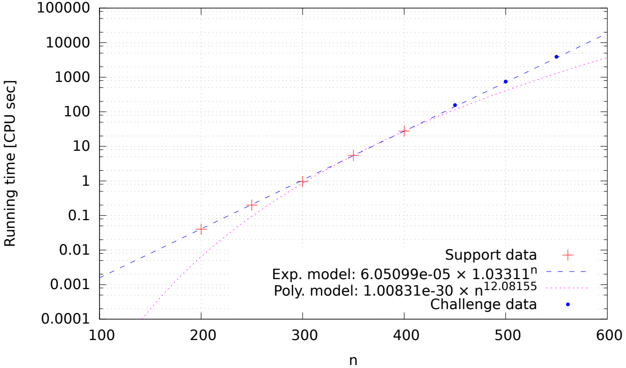
| $n$ | 450 | 500 | 550 |
|---|---|---|---|
| median | 156.480 | 750.510 | 3896.450 |



# Empirical Scaling Results – DPLL-based Solvers

Fit parametric models:

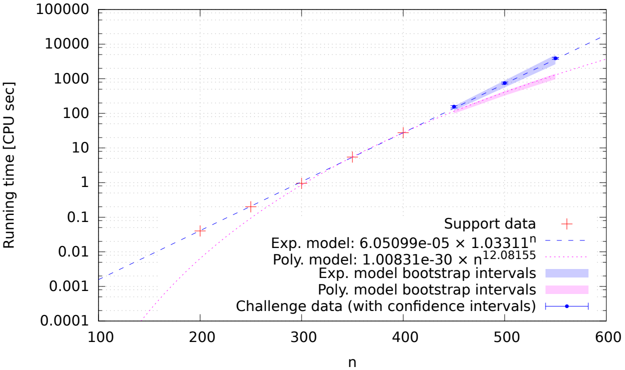
| | | Model | RMSE (support) | RMSE (challenge) |
|---|---|---|---|---|
| kcnfs | **Exp. Model** | $\mathbf{4.30400 \times 10^{-5} \times 1.03411^n}$ | **0.05408** | **143.3** |
| | Poly. Model | $9.40745 \times 10^{-31} \times n^{12.1005}$ | 0.06822 | 1516 |



# Empirical Scaling Results – DPLL-based Solvers

Bootstrap re-sampling:

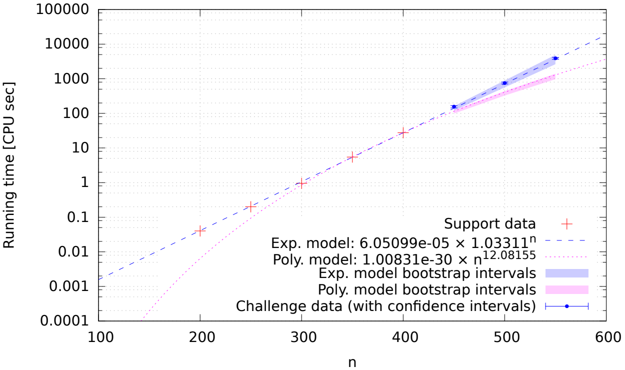
| Solver | $n$ | Predicted confidence intervals | | Observed median run-time (sec) | |
|---|---|---|---|---|---|
| | | Poly. model | Exp. model | Point estimates | Confidence intervals |
| kcnfs | 450 | [98.326, 122.115] | **[120.078, 161.444]** | 156.480 | [143.340, 166.770] |
| | 500 | [327.997, 439.089] | **[561.976, 889.428]*** | 750.510 | [708.290, 806.130] |
| | 550 | [971.862, 1402.255] | **[2622.488, 4901.661]*** | 3896.450 | [3633.630, 4130.915] |



# Empirical Scaling Results – DPLL-based Solvers

Bootstrap re-sampling:

| Solver | Model | Confidence interval of $a$ | Confidence interval of $b$ |
|---|---|---|---|
| kcnfs | Poly. | $[3.33969 \times 10^{-31}, 4.30846 \times 10^{-29}]$ | $[11.4234, 12.2674]$ |
| | Exp. | $[3.33378 \times 10^{-5}, 1.07425 \times 10^{-4}]$ | $[1.03136, 1.03476]$ |

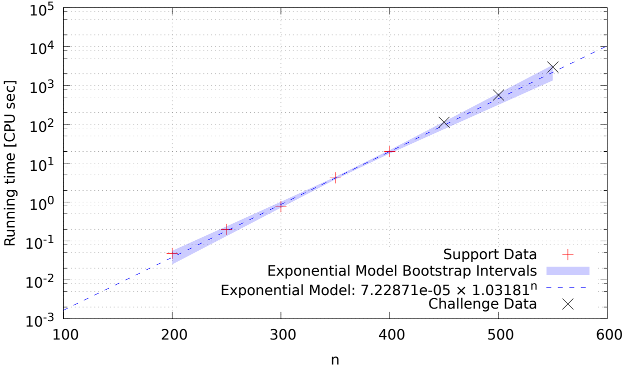


# Empirical Scaling Results – DPLL-based Solvers

Compare scaling models:

- No significant difference between two march-variants
- Two march-variants scale significantly better than kcnfs

Scaling models of march_hi:

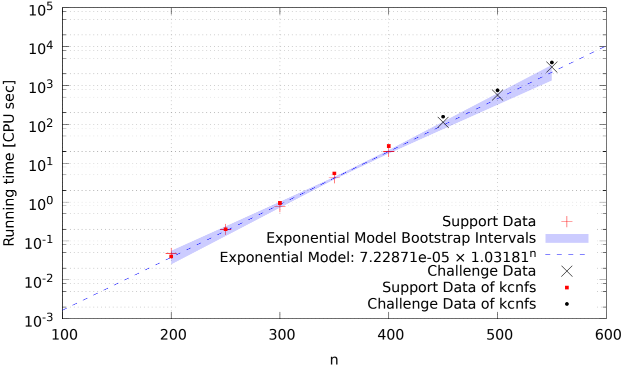


# Empirical Scaling Results – DPLL-based Solvers

Compare scaling models:

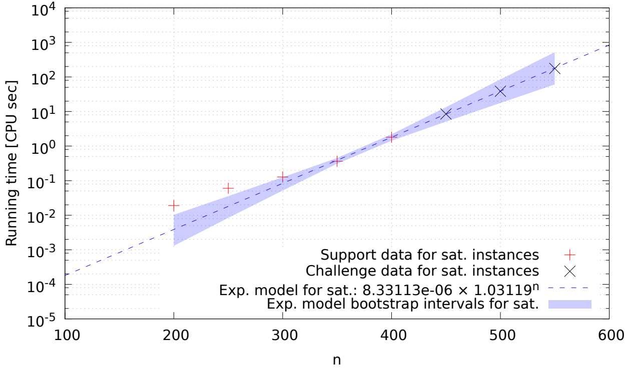
- No significant difference between two march-variants
- Two march-variants scale significantly better than kcnfs

Compare scaling models of kcnfs against march_hi:



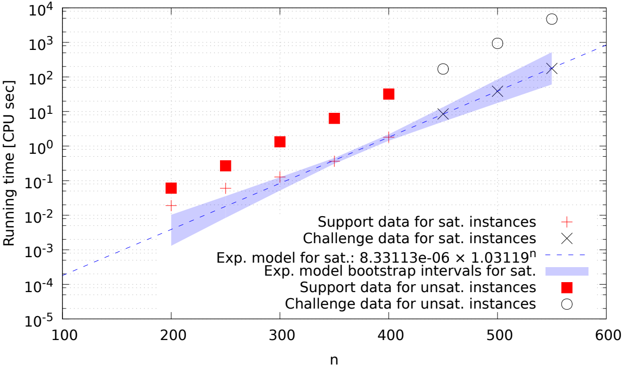
# Empirical Scaling Results – DPLL-based Solvers

Difference in solving satisfiable instances and unsatisfiable instances:



# Empirical Scaling Results – DPLL-based Solvers

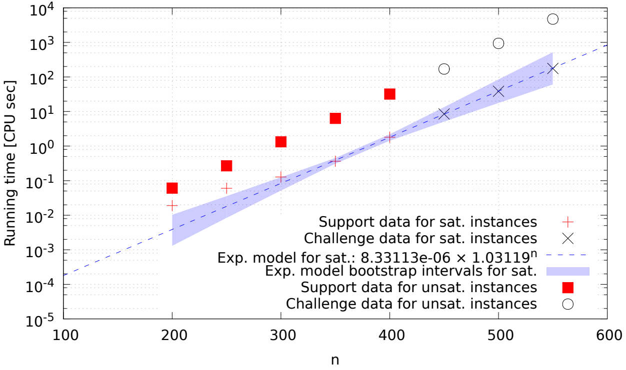
Difference in solving satisfiable instances and unsatisfiable instances:



# Empirical Scaling Results – DPLL-based Solvers

Difference in solving satisfiable instances and unsatisfiable instances:

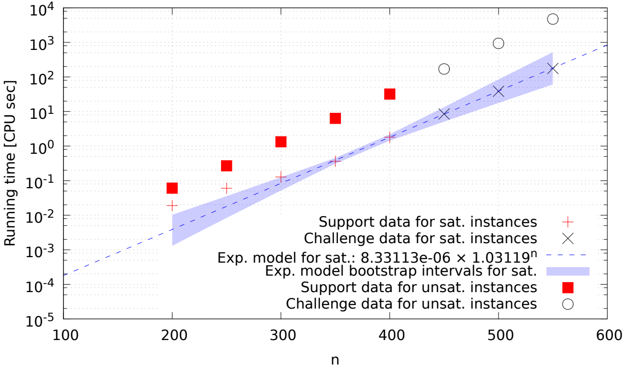
- Is the difference a constant factor?



# Empirical Scaling Results – DPLL-based Solvers

Difference in solving satisfiable instances and unsatisfiable instances:

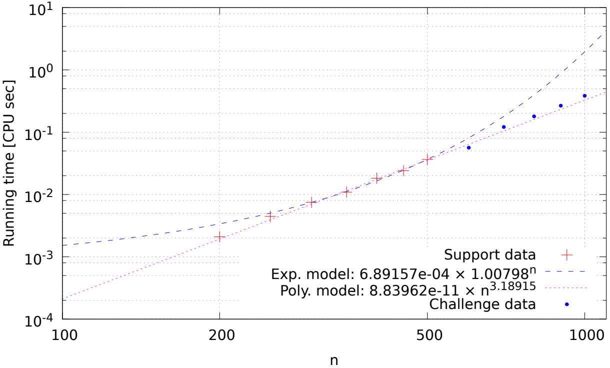
- Is the difference a constant factor?
  - Fit running times of solving unsatisfiable instances with model $a \cdot b_{\mathrm{sat}}^n$
  - Slower in solving unsatisfiable instances by constant factor only



# Empirical Scaling Results – SLS-based Solvers

Fit parametric models:

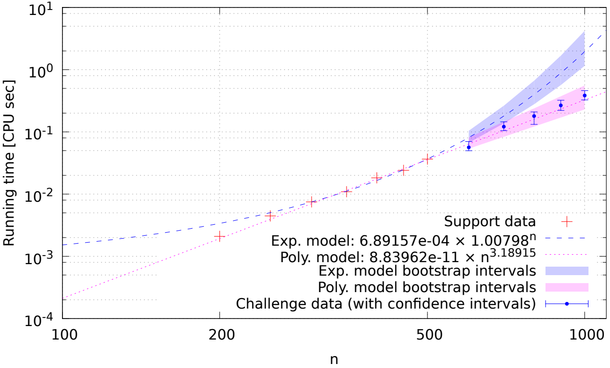
| | | Model | RMSE (support) | RMSE (challenge) |
|---|---|---|---|---|
| WalkSAT/SKC | Exp. Model | $6.89157 \times 10^{-4} \times 1.00798^{n}$ | 0.0008564 | 0.7600 |
| | **Poly. Model** | $\mathbf{8.83962 \times 10^{-11} \times n^{3.18915}}$ | **0.0007433** | **0.03142** |



# Empirical Scaling Results – SLS-based Solvers

Bootstrap re-sampling:

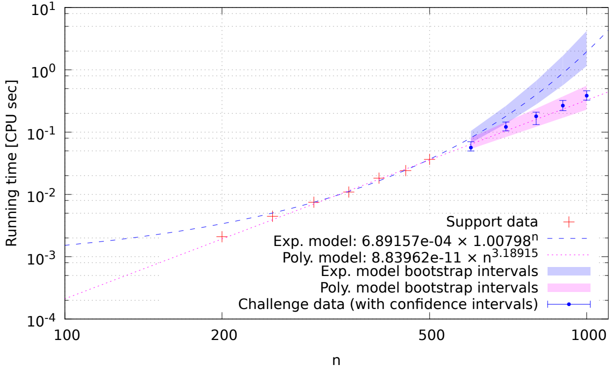
| Solver | $n$ | Predicted confidence intervals | | Observed median run-time (sec) | |
|---|---|---|---|---|---|
| | | Poly. model | Exp. model | Point estimates | Confidence intervals |
| WalkSAT/SKC | 600 | **[0.054, 0.081]** | [0.067, 0.104] | 0.056 | [0.050, 0.070] |
| | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| | 1000 | **[0.229, 0.557]*** | [1.151, 4.200] | 0.385 | [0.327, 0.461] |



# Empirical Scaling Results – SLS-based Solvers

## Bootstrap re-sampling:

| Solver | Model | Confidence interval of $a$ | Confidence interval of $b$ |
|---|---|---|---|
| WalkSAT/SKC | Exp. | $[4.05064 \times 10^{-4}, 1.00662 \times 10^{-3}]$ | $[1.00709, 1.00924]$ |
| | Poly. | $[2.58600 \times 10^{-12}, 8.63869 \times 10^{-10}]$ | $[2.80816, 3.76751]$ |



# Empirical Scaling Results – SLS-based Solvers

No significant difference among scaling models for WalkSAT/SKC, BalancedZ & probSAT

# Empirical Scaling Results – SLS-based Solvers

No significant difference among scaling models for WalkSAT/SKC, BalancedZ & probSAT

Higher quantiles:

- Scaling of 0.75- and 0.9-quantile of running times still consistent with polynomial model

# Empirical Scaling Results – SLS-based Solvers

No significant difference among scaling models for WalkSAT/SKC, BalancedZ & probSAT

Higher quantiles:

- Scaling of 0.75- and 0.9-quantile of running times still consistent with polynomial model

Even larger instances:

- Limited experiments on instances of $n \in \{1500, 2000, 5000\}$
- Data consistent with polynomial models

# Empirical Scaling Results – 4-SAT

Refined model for 4-SAT phase transition

# Empirical Scaling Results – 4-SAT

Refined model for 4-SAT phase transition

Phase-transition random 4-SAT

- SLS-based solvers: exponential or root-exponential
- DPLL-based solvers: exponential

# Empirical Scaling Results – 4-SAT

Refined model for 4-SAT phase transition

Phase-transition random 4-SAT

- SLS-based solvers: exponential or root-exponential
- DPLL-based solvers: exponential

Under-constrained instances with $m = 2^{k-1} \cdot n$

- WalkSAT/SKC: polynomial model is a better fit
- kcnfs: root-exponential model is a better fit

# TSP – Scientific Questions

Finding time: time required for finding optimal solutions w/o proving

# TSP – Scientific Questions

Finding time: time required for finding optimal solutions w/o proving

For complete solvers:

- How do finding times scale with instance size?
- How do finding times scale differently from proving times?

# TSP – Scientific Questions

Finding time: time required for finding optimal solutions w/o proving

For complete solvers:

- How do finding times scale with instance size?
- How do finding times scale differently from proving times?

For incomplete solvers:

- How do running times scale with instance size?
- Are incomplete solvers significantly faster from scaling point of view?

# TSP – Scientific Questions

Finding time: time required for finding optimal solutions w/o proving

For complete solvers:

- How do finding times scale with instance size?
- How do finding times scale differently from proving times?

For incomplete solvers:

- How do running times scale with instance size?
- Are incomplete solvers significantly faster from scaling point of view?

Solvers:

- Complete: Concorde [Applegate et al., 2012]
- Incomplete: LKH [Helsgaun, 2009], EAX [Nagata and Kobayashi, 2013]

# TSP – Scientific Questions

Finding time: time required for finding optimal solutions w/o proving

For complete solvers:

- How do finding times scale with instance size?
- How do finding times scale differently from proving times?

For incomplete solvers:

- How do running times scale with instance size?
- Are incomplete solvers significantly faster from scaling point of view?

Solvers:

- Complete: Concorde [Applegate et al., 2012]
- Incomplete: LKH [Helsgaun, 2009], EAX [Nagata and Kobayashi, 2013]

Problem instances: random uniform Euclidean (RUE)

# TSP – Related Work

Work on running time distribution of:

- Concorde: Hoos and Stützle [2014, 2015]
- LKH & EAX: Dubois-Lacoste et al. [2015]

Work on empirical scaling of

- Concorde (proving): Applegate et al. [2006], Hoos and Stützle [2014]
- LKH & EAX: Dubois-Lacoste et al. [2015]

# TSP – Related Work

Work on running time distribution of:

- Concorde: Hoos and Stützle [2014, 2015]
- LKH & EAX: Dubois-Lacoste et al. [2015]

Work on empirical scaling of

- Concorde (proving): Applegate et al. [2006], Hoos and Stützle [2014]
- LKH & EAX: Dubois-Lacoste et al. [2015]

Extensions:

- Empirical scaling of finding times of Concorde
- Comparison of scaling of complete & incomplete algorithms

# Empirical Scaling Results – Concorde

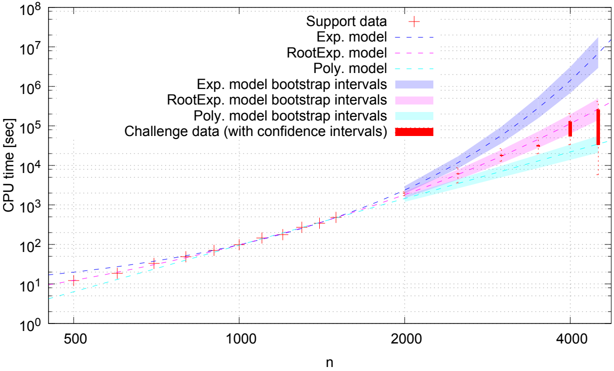
Optimistic & pessimistic treatment of timeout runs

# Empirical Scaling Results – Concorde

Optimistic & pessimistic treatment of timeout runs

Finding times consistent with root-exponential model

- Exponential and polynomial models rejected with high confidence



# Empirical Scaling Results – Concorde

Finding and proving times differ by constant factor:

# Empirical Scaling Results – Concorde

Finding and proving times differ by constant factor:

- Intervals of $b$ in both models:
  - Proving: $[1.2212, 1.2630]$
  - Finding: $[1.2280, 1.2760]$

# Empirical Scaling Results – Concorde

Finding and proving times differ by constant factor:

- Intervals of $b$ in both models:
  - Proving: $[1.2212, 1.2630]$
  - Finding: $[1.2280, 1.2760]$
- Fit model $a \cdot b_{proving}^{\sqrt{n}}$ on finding time
  - Very good fit
  - $a$ for proving vs. finding: 0.21 vs. 0.11

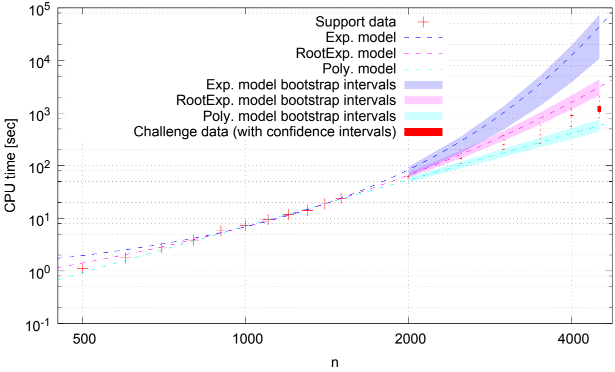
# Empirical Scaling Results – LKH & EAX

Optimistic & pessimistic treatment of instances with unknown optimal

# Empirical Scaling Results – LKH & EAX

Optimistic & pessimistic treatment of instances with unknown optimal

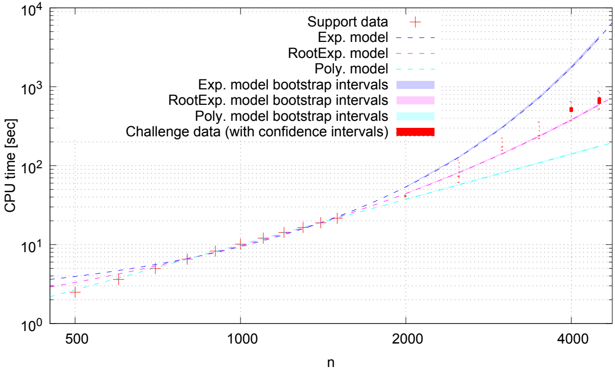
Running times of LKH bounded by polynomial & root-exponential



# Empirical Scaling Results – LKH & EAX

Optimistic & pessimistic treatment of instances with unknown optimal

Running times of EAX consistent with root-exponential



# Empirical Scaling Results – LKH & EAX

LKH & EAX scale significantly better than Concorde

# Empirical Scaling Results – LKH & EAX

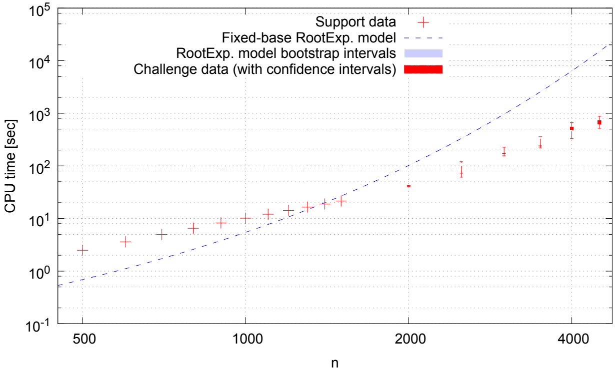
LKH & EAX scale significantly better than Concorde

- Comparison of intervals of $b$'s

# Empirical Scaling Results – LKH & EAX

LKH & EAX scale significantly better than Concorde

- Comparison of intervals of $b$'s
- Fit model $a \cdot b_{Concorde}^{\sqrt{n}}$ on running times of LKH & EAX



# Empirical Scaling Results – EAX Configuration

Configuration experiments:

- SMAC [Hutter et al., 2011]
- 2 parameters: population size & restarting iterations
- 25 parallel runs

# Empirical Scaling Results – EAX Configuration

Configuration experiments:

- SMAC [Hutter et al., 2011]
- 2 parameters: population size & restarting iterations
- 25 parallel runs

Effect on scaling models:

- $b$ in root-exponential model: from $\approx 1.14$ to $\approx 1.12$

# Empirical Scaling Results – EAX Configuration

Configuration experiments:

- SMAC [Hutter et al., 2011]
- 2 parameters: population size & restarting iterations
- 25 parallel runs

Effect on scaling models:

- $b$ in root-exponential model: from $\approx 1.14$ to $\approx 1.12$

Effect of varying population size:

- Population size $\propto$ instance size
- Best fit: polynomial instead of root-exponential

# Empirical Scaling Analyser (ESA)

Automated tool for empirical scaling analysis [Mu and Hoos, 2015b]

- Available as web service or command-line tool
  - `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`

# Empirical Scaling Analyser (ESA)

Automated tool for empirical scaling analysis [Mu and Hoos, 2015b]

- Available as web service or command-line tool
  - `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`
- One algorithm
- Multiple models

# Empirical Scaling Analyser (ESA)

Automated tool for empirical scaling analysis [Mu and Hoos, 2015b]

- Available as web service or command-line tool
  - `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`
- One algorithm
- Multiple models
- Input of running times
  - Timeout or crashed runs
  - Unknown running times

# Empirical Scaling Analyser (ESA)

Automated tool for empirical scaling analysis [Mu and Hoos, 2015b]

- Available as web service or command-line tool
  - `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`
- One algorithm
- Multiple models
- Input of running times
  - Timeout or crashed runs
  - Unknown running times
- Output as technical report
  - Figures of fitted models
  - Tables of fitted models and bootstrap intervals
  - Automatically generated interpretations

# ESA – Example Input

```
# instance name, size, datum (running time)
portgen-500-1000.tsp,500,2.3
portgen-500-100.tsp,500,2.58
portgen-500-101.tsp,500,2.36
portgen-500-102.tsp,500,2.51
portgen-500-103.tsp,500,2.63
portgen-500-104.tsp,500,2.84
portgen-500-105.tsp,500,2.62
portgen-500-106.tsp,500,3
...
portgen-600-1000.tsp,600,3.42
...
portgen-4500-10.tsp,4500,727.68
portgen-4500-11.tsp,4500,inf
...
#instances,4000,100
#instances,4500,100
```

# ESA – Example Output

On the empirical scaling of running time of EAX for solving RUE instances

Empirical Scaling Analyser

23rd June 2015

## 1 Introduction

This is the automatically generated report on the empirical scaling of the running time of EAX for solving RUE instances.

## 2 Methodology

For our scaling analysis, we considered the following parametric models:

- $Exp\,[a,b]\,(n) = a \times b^{x}$ (2-parameter Exp)
- $RootExp\,[a,b]\,(n) = a \times b^{\sqrt{x}}$ (2-parameter RootExp)
- $Poly\,[a,b]\,(n) = a \times x^{b}$ (2-parameter Poly)

Note that the approach could be easily extended to other scaling models. For fitting parametric scaling models to observed data, we used the non-linear least-squares Levenberg-Marquardt algorithm.

Models were fitted to performance observations in the form of medians of the distributions of running times over sets of instances for given $n$, the instance size. To assess the fit of a given scaling model to observed data, we used root-mean-square error (RMSE).

Due to the instances for which the running times are unknown, there is uncertainty about the precise location of the medians of the running time distributions at each such $n$, and we can only provide bounds on those medians instead. Closely following [Dubois-Lacoste et al.(2015)Dubois-Lacoste, Hoos, and Stützle], we calculate these bounds based on the best and worst case scenarios, in which all instances with unknown running times are easiest or hardest, respectively. We note that these are not confidence intervals, since we can guarantee the actual median running times to lie within them. We also calculate RMSEs and confidence intervals based on these bounds. In the following, we say that a scaling model is in- consistent with observed data of the interval formed by the bounds on the observed medians if it is disjoint from the bootstrap confidence interval for predicted running times; we say that a scaling model is con- sistent with observed data, if the interval for observed medians overlaps with, but is not fully contained within the bootstrap confidence interval for predicted running times; we say that a scaling model is strongly consistent with observed data, if the interval for observed medians is fully contained within the bootstrap confidence interval for predicted running times.

Closely following [Hoos(2009), Hoos and Stützle(2014)], we computed 95% bootstrap confidence intervals for the performance predictions obtained from our scaling models, based on 1000 bootstrap samples per instance set and 1000 automatically fitted variants of each scaling model.

1

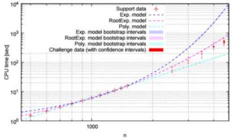



Figure 1: Fitted models of the medians of the running times. Both models are fitted with the medians of the running times of EAX solving the SAT instances from the set of RUE instances of $500 \leq n \leq 1500$ variables, and are challenged by the medians of the running times of $2500 \leq n \leq 4500$ variables.

| Solver | Model | Confidence interval of $a$ | Confidence interval of $b$ |
|---|---|---|---|
| | Exp. | $[0.80556, 0.83992]$ | $[1.002, 1.002]$ |
| EAX | RootExp. | $[0.083217, 0.090067]$ | $[1.1424, 1.1451]$ |
| | Poly. | $[1.0646 \times 10^{-6}, 1.4288 \times 10^{-6}]$ | $[2.2129, 2.2551]$ |

Table 4: Bootstrap intervals of model parameters for the medians of the running time

## References


[Dubois-Lacoste et al.(2015)Dubois-Lacoste, Hoos, and Stützle] Jérémie Dubois-Lacoste, Holger H. Hoos, and Thomas Stützle. On the empirical scaling behaviour of state-of-the-art local search algorithms for the euclidean tsp. In *GECCO*. ACM, 2015.

[Hoos(2009)] Holger H Hoos. A bootstrap approach to analysing the scaling of empirical run-time data with problem size. Technical report, Technical Report TR-2009-16, University of British Columbia, 2009.

[Hoos and Stützle(2014)] Holger H Hoos and Thomas Stützle. On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. *European Journal of Operational Research*, 238(1):87–94, 2014.


4

# Contributions

Empirical scaling results for:

- Phase-transition random 3-SAT:
  - SLS-based solvers: polynomially; DPLL-based solvers: exponentially
  - DPLL-based: faster by constant factor for solving satisfiable instances
- Phase-transition & under-constrained random 4-SAT

# Contributions

Empirical scaling results for:

- Phase-transition random 3-SAT:
  - SLS-based solvers: polynomially; DPLL-based solvers: exponentially
  - DPLL-based: faster by constant factor for solving satisfiable instances
- Phase-transition & under-constrained random 4-SAT
- Euclidean TSP:
  - All solvers: root-exponentially
  - Concorde: finding & proving times differ by constant factor
  - LKH & EAX: scale significantly better than Concorde finding times

# Contributions

Empirical scaling results for:

- Phase-transition random 3-SAT:
  - SLS-based solvers: polynomially; DPLL-based solvers: exponentially
  - DPLL-based: faster by constant factor for solving satisfiable instances
- Phase-transition & under-constrained random 4-SAT
- Euclidean TSP:
  - All solvers: root-exponentially
  - Concorde: finding & proving times differ by constant factor
  - LKH & EAX: scale significantly better than Concorde finding times

Methodology refinements and extensions:

- Extended use of conf. intervals for model assessment & comparison
- ESA: automated tool for scaling analysis [Mu and Hoos, 2015b]
  - `www.cs.ubc.ca/labs/beta/Projects/ESA/esa-online.html`

# Future Work

Potential methodology improvements:

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling
- Automatic model selection from large library of models

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling
- Automatic model selection from large library of models
- More sophisticated models with lower-order terms

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling
- Automatic model selection from large library of models
- More sophisticated models with lower-order terms
- Analysis when no instance generator available
  - One or a few instances at each size
  - Fewer instances overall
  - Outlier detection

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling
- Automatic model selection from large library of models
- More sophisticated models with lower-order terms
- Analysis when no instance generator available
  - One or a few instances at each size
  - Fewer instances overall
  - Outlier detection

Apply to other problem domains

- Planning, scheduling, MIP, etc.

# Future Work

Potential methodology improvements:

- Nested bootstrap re-sampling
- Automatic model selection from large library of models
- More sophisticated models with lower-order terms
- Analysis when no instance generator available
  - One or a few instances at each size
  - Fewer instances overall
  - Outlier detection

Apply to other problem domains

- Planning, scheduling, MIP, etc.

Explore use beyond time complexity of algorithms

- E.g., learning curves of ML algorithms

# References I


David L Applegate, Robert E Bixby, Vasek Chvátal, and William J Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

David L Applegate, Robert E Bixby, Vasek Chvátal, and William J Cook. The traveling salesman problem, concorde TSP solver. *http://www. tsp. gatech. edu/concorde*, 2012.

P. Cheeseman, B. Kanefsky, and W.M. Taylor. Where the really hard problems are. In *IJCAI*, pages 331–337, 1991.

Cristian Coarfa, Demetrios D Demopoulos, Alfonso San Miguel Aguirre, Devika Subramanian, and Moshe Y Vardi. Random 3-SAT: The plot thickens. *Constraints*, 8(3):243–261, 2003.

Stephen A Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.

James M Crawford and Larry D Auton. Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81(1):31–57, 1996.

Jérémie Dubois-Lacoste, Holger H Hoos, and Thomas Stützle. On the empirical scaling behaviour of state-of-the-art local search algorithms for the Euclidean TSP. In *GECCO*, pages 377–384. ACM, 2015.

Michael R Garey and David S Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: Freeman, 1979.

Ian P Gent and Toby Walsh. Towards an understanding of hill-climbing procedures for SAT. In *AAAI*, volume 93, pages 28–33, 1993.

Ian P Gent, Ewan MacIntyre, Patrick Prosser, and Toby Walsh. The scaling of search cost. In *AAAI/IAAI*, pages 315–320, 1997.

Keld Helsgaun. General k-opt submoves for the lin–kernighan tsp heuristic. *Mathematical Programming Computation*, 1(2-3):119–163, 2009.

Holger H. Hoos. A bootstrap approach to analysing the scaling of empirical run-time data with problem size. Technical report, Technical Report TR-2009-16, Department of Computer Science, University of British Columbia, 2009.

Holger H. Hoos and Thomas Stützle. On the empirical scaling of run-time for finding optimal solutions to the travelling salesman problem. *European Journal of Operational Research*, 238(1):87–94, 2014.

# References II


Holger H Hoos and Thomas Stützle. On the empirical time complexity of finding optimal solutions vs proving optimality for Euclidean TSP instances. *Optimization Letters*, 2015. doi: 0.1007/s11590-014-0828-5.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization*, pages 507–523. Springer, 2011.

Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-SAT from the cavity method. *Random Structures and Algorithms*, 28(3):340–373, 2006.

David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of SAT problems. In *AAAI*, volume 92, pages 459–465, 1992.

Zongxu Mu and Holger H. Hoos. On the empirical time complexity of random 3-SAT at the phase transition. *IJCAI*, 2015a.

Zongxu Mu and Holger H Hoos. Empirical scaling analyser: An automated system for empirical analysis of performance scaling. In *GECCO Companion*, pages 771–772, 2015b.

Yuichi Nagata and Shigenobu Kobayashi. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *Informs journal on computing*, 25(2):346–363, 2013.

Christos H Papadimitriou. The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

Andrew J Parkes and Joachim P Walser. Tuning local search for satisfiability testing. In *AAAI/IAAI*, volume 1, pages 356–362, 1996.

Makoto Yokoo. Why adding more constraints makes a problem easier for hill-climbing algorithms: Analyzing landscapes of CSPs. In *CP97*, pages 356–370. Springer, 1997.