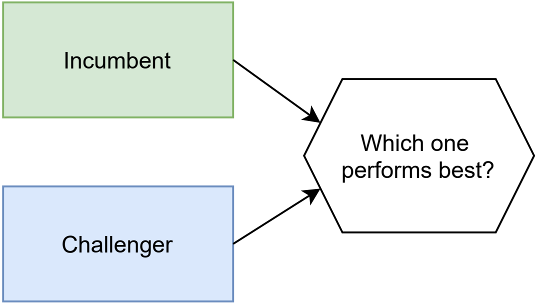
Statistical Comparison of Algorithm Performance Through Instance Selection

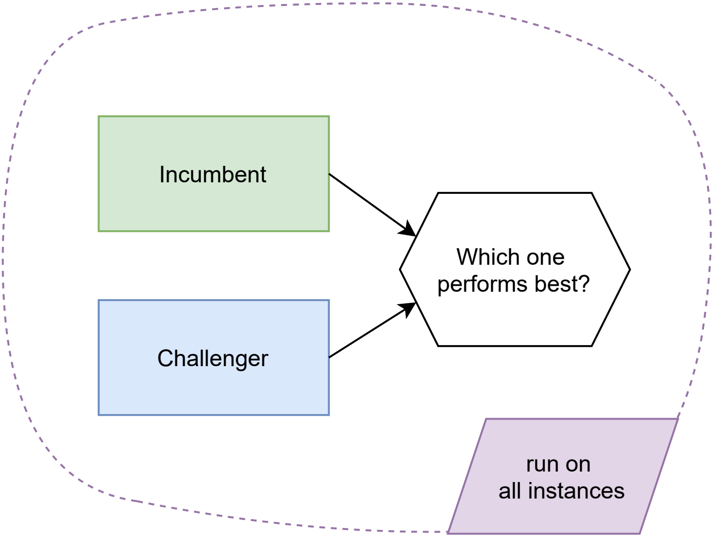
T. Matricon, M. Anastacio, N. Fijalkow, L. Simon and H. H. Hoos

October, 2021

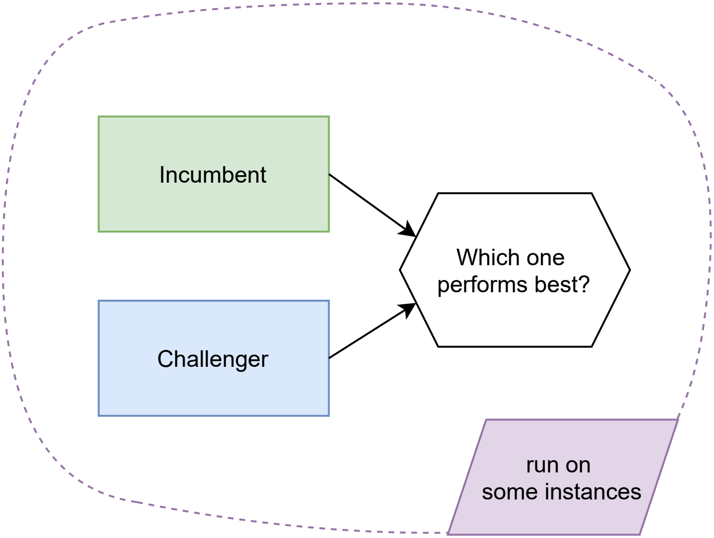
The Problem



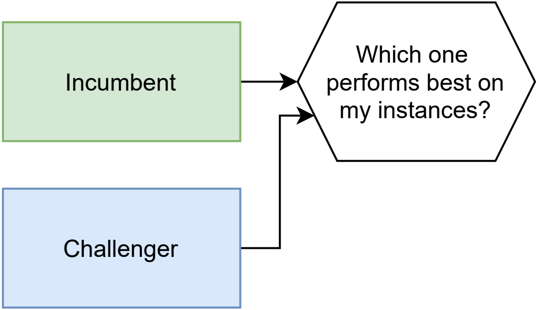
The Problem


Incumbent
Challenger
Which one performs best?
run on all instances

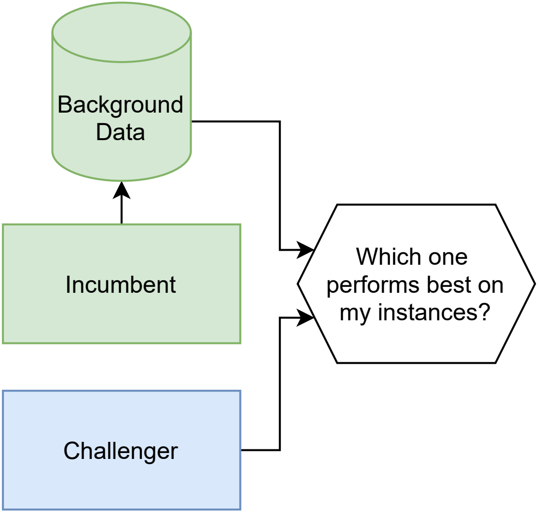
The Problem


Incumbent
Challenger
Which one performs best?
run on some instances

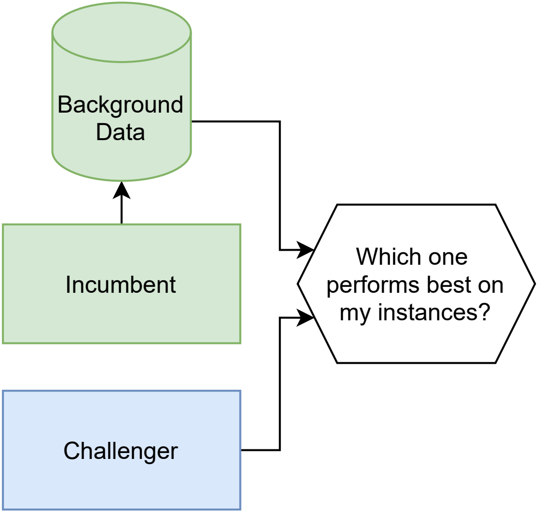
Per Set Efficient Algorithm Selection (PSEAS) Problem



Per Set Efficient Algorithm Selection (PSEAS) Problem



Per Set Efficient Algorithm Selection (PSEAS) Problem



min CPU time
max accuracy

Iterative Strategy

```
compute instances score
```

Iterative Strategy

```
compute instances score
while confidence < threshold do
```

Iterative Strategy

```
compute instances score
while confidence < threshold do
    pick best instance according to score
```

Iterative Strategy

compute instances score
while confidence $<$ threshold **do**
    pick best instance according to score
    run challenger on that instance

Iterative Strategy

```
compute instances score
while confidence < threshold do
    pick best instance according to score
    run challenger on that instance
    update confidence
    update instances score
end while
```

Iterative Strategy

```
compute instances score
while confidence < threshold do
    pick best instance according to score
    run challenger on that instance
    update confidence
    update instances score
end while
return best performing algorithm
```

Iterative Strategy

```
compute instances score
while confidence < threshold do
    pick best instance according to score
    run challenger on that instance
    update confidence
    update instances score
end while
return best performing algorithm
```

Iterative Strategy

```
compute instances score
while confidence < threshold do
    pick best instance according to score
    run challenger on that instance
    update confidence
    update instances score
end while
return best performing algorithm
```

Score & Confidence Methods

Instance scores:

- baseline: random

Score & Confidence Methods

Instance scores:

- baseline: random
- variance-based: $\frac{Var[T_I]}{\mathbb{E}[T_I]}$
- discrimination-based [Gent et al., 2014]

Score & Confidence Methods

Instance scores:

- baseline: random
- variance-based: $\frac{Var[T_I]}{\mathbb{E}[T_I]}$
- discrimination-based [Gent et al., 2014]
- information-based: $\frac{\mathbb{E}[InfoGained[I]]}{\mathbb{E}[T_I]}$
- feature-based

Score & Confidence Methods

Instance scores:

- baseline: random
- variance-based: $\frac{Var[T_I]}{\mathbb{E}[T_I]}$
- discrimination-based [Gent et al., 2014]
- information-based: $\frac{\mathbb{E}[InfoGained[I]]}{\mathbb{E}[T_I]}$
- feature-based

Confidence:

- baseline: fixed size subset

Score & Confidence Methods

Instance scores:

- baseline: random
- variance-based: $\frac{Var[T_I]}{\mathbb{E}[T_I]}$
- discrimination-based [Gent et al., 2014]
- information-based: $\frac{\mathbb{E}[InfoGained[I]]}{\mathbb{E}[T_I]}$
- feature-based

Confidence:

- baseline: fixed size subset
- Wilcoxon [Birattari, 2009]

Score & Confidence Methods

Instance scores:

- baseline: random
- variance-based: $\frac{Var[T_I]}{\mathbb{E}[T_I]}$
- discrimination-based [Gent et al., 2014]
- information-based: $\frac{\mathbb{E}[InfoGained[I]]}{\mathbb{E}[T_I]}$
- feature-based

Confidence:

- baseline: fixed size subset
- Wilcoxon [Birattari, 2009]
- distribution-based: $\mathbb{P}(\underbrace{\sum_I \Delta T_I}_{\text{rand. var.}} + \underbrace{\sum_I \Delta t_I}_{\text{constant}} \leq 0)$

Evaluation protocol

- SAT 2020 + 3 ASlib datasets (CSP MiniZinc, BNSL, SAT 2018) [Bischl et al., 2016]

Evaluation protocol

- SAT 2020 + 3 ASlib datasets (CSP MiniZinc, BNSL, SAT 2018) [Bischl et al., 2016]
- All possible ordered pairs of algorithms

Evaluation protocol

- SAT 2020 + 3 ASlib datasets (CSP MiniZinc, BNSL, SAT 2018) [Bischl et al., 2016]
- All possible ordered pairs of algorithms
- Report
  - % of CPU time
  - % of accuracy

Research Questions

- Can our strategies reduce the CPU time required for evaluating a new algorithm?

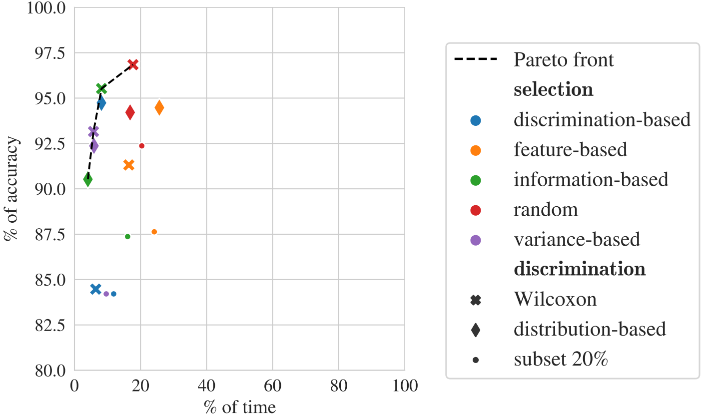
Research Questions

- Can our strategies reduce the CPU time required for evaluating a new algorithm?
- Can our strategies discriminate well between top ranking algorithms?

Research Questions

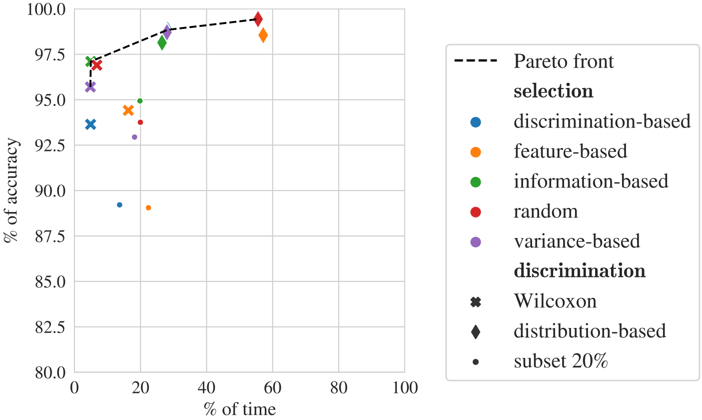
- Can our strategies reduce the CPU time required for evaluating a new algorithm?
- Can our strategies discriminate well between top ranking algorithms?
- How do the selection methods affect the accuracy of the strategies?

Accuracy over Median Running Time



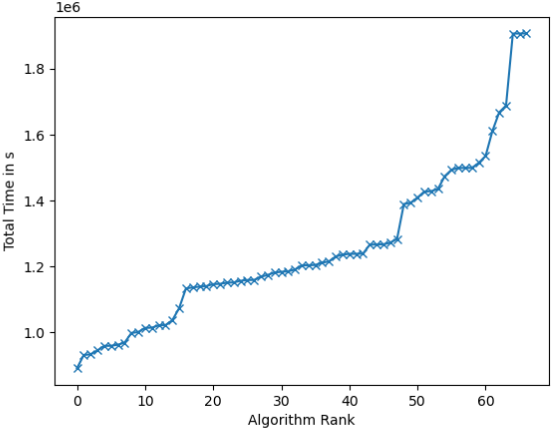
CSP MiniZinc [Stuckey et al., 2014] Confidence > 95%

Accuracy over Median Running Time



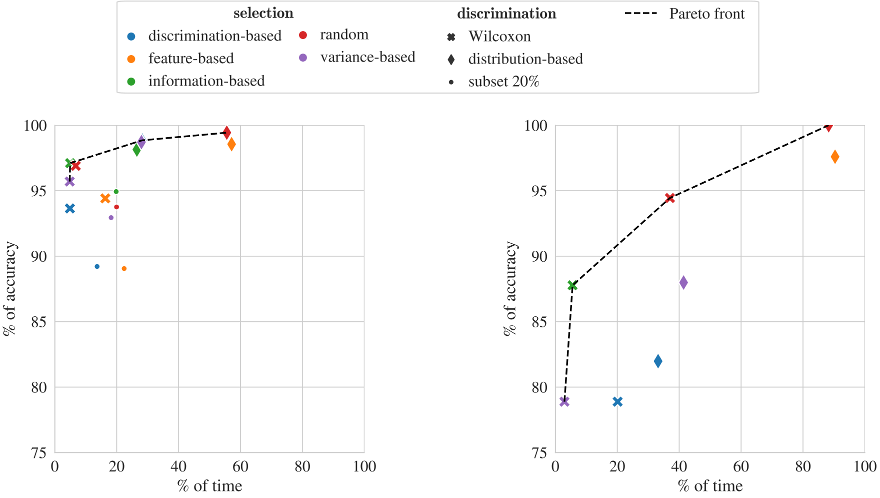
SAT 20 [Balyo et al., 2020]   Confidence > 95%

SAT 2020 Running Times



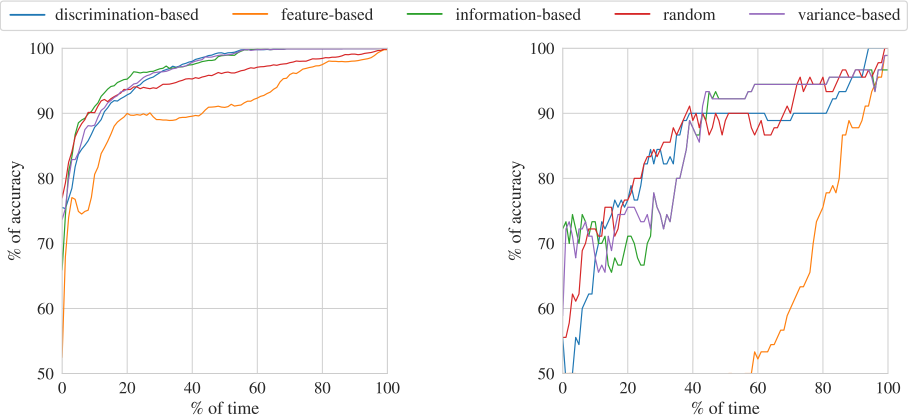
SAT 20 Algorithm Ranking

Top Ranking Algorithms



SAT 20 Confidence > 95% SAT 20, top-10

Selection Methods Accuracy for Wilcoxon



SAT 20

SAT 20, top-10

Conclusion

- Comparing algorithm on subset = faster result
  Statistical test = confidence in result

Conclusion

- Comparing algorithm on subset = faster result
  Statistical test = confidence in result

- How to select instances?
  - Low running time
  - High discrimination power

Conclusion

- Comparing algorithm on subset = faster result
  Statistical test = confidence in result
- How to select instances?
  - Low running time
  - High discrimination power
- Future applications
  - Faster development
  - Lower cost of experiments / competitions
  - Faster algorithm configuration

Conclusion

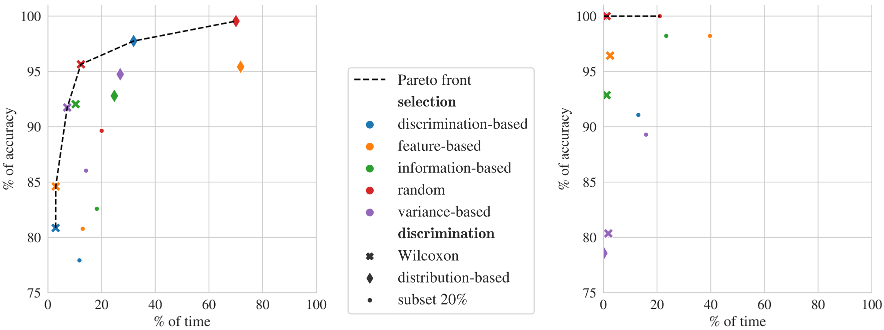
- Comparing algorithm on subset = faster result
  Statistical test = confidence in result

- How to select instances?
  - Low running time
  - High discrimination power

- Future applications
  - Faster development
  - Lower cost of experiments / competitions
  - Faster algorithm configuration

`github.com/Theomat/PSEAS`

Accuracy over median running time



SAT 18

BNSL

T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors. *Proceedings of SAT Competition 2020: Solver and Benchmark Descriptions*. Department of Computer Science Report Series B. Department of Computer Science, University of Helsinki, Finland, 2020. URL `http://hdl.handle.net/10138/318450`.

M. Birattari. *Tuning Metaheuristics: A Machine Learning Perspective*. Springer Publishing Company, Incorporated, 1st ed. 2005. 2nd printing edition, 2009. ISBN 3642004822. doi: 10.1007/978-3-642-00483-4.

B. Bischl, P. Kerschke, L. Kotthoff, M. Lindauer, Y. Malitsky, A. Fréchette, H. Hoos, F. Hutter, K. Leyton-Brown, K. Tierney, and J. Vanschoren. Aslib: A benchmark library for algorithm selection. *Artificial Intelligence*, 237:41–58, 2016. ISSN 0004-3702. doi: 10.1016/j.artint.2016.04.003.

I. P. Gent, B. S. Hussain, C. Jefferson, L. Kotthoff, I. Miguel, G. F. Nightingale, and P. Nightingale. Discriminating instance generation for automated constraint model selection. In B. O'Sullivan, editor, *Principles and Practice of Constraint Programming*, pages 356–365, Cham, 2014. Springer International Publishing. doi: 10.1007/978-3-319-10428-7_27.

P. Stuckey, T. Feydy, A. Schutt, G. Tack, and J. Fischer. The minizinc challenge 2008-2013. *AI Magazine*, 35(2):55–60, 2014. ISSN 0738-4602. doi: 10.1609/aimag.v35i2.2539.