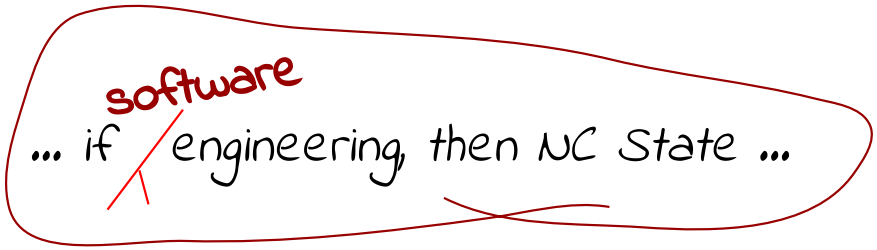
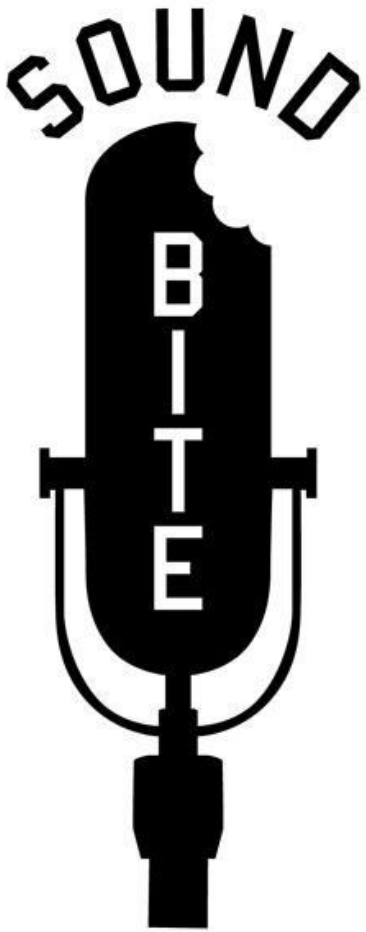


In the age of Software 2.0, what role for software engineers?

Tim Menzies
cs, se, ai, ncstate, usa
timm@ieee.org,

Mar 7, 2019

Association for Computing Machinery



- Programmers > Programming
- Software engineering = the search for feedback
- Engineering AI systems: the next challenge
- D.U.O. = data miners using/used-by optimizers

AI: the new engineering challenge

So many new building blocks for new kinds of software
e.g. this talk: DUO= data miners using/used by optimizers

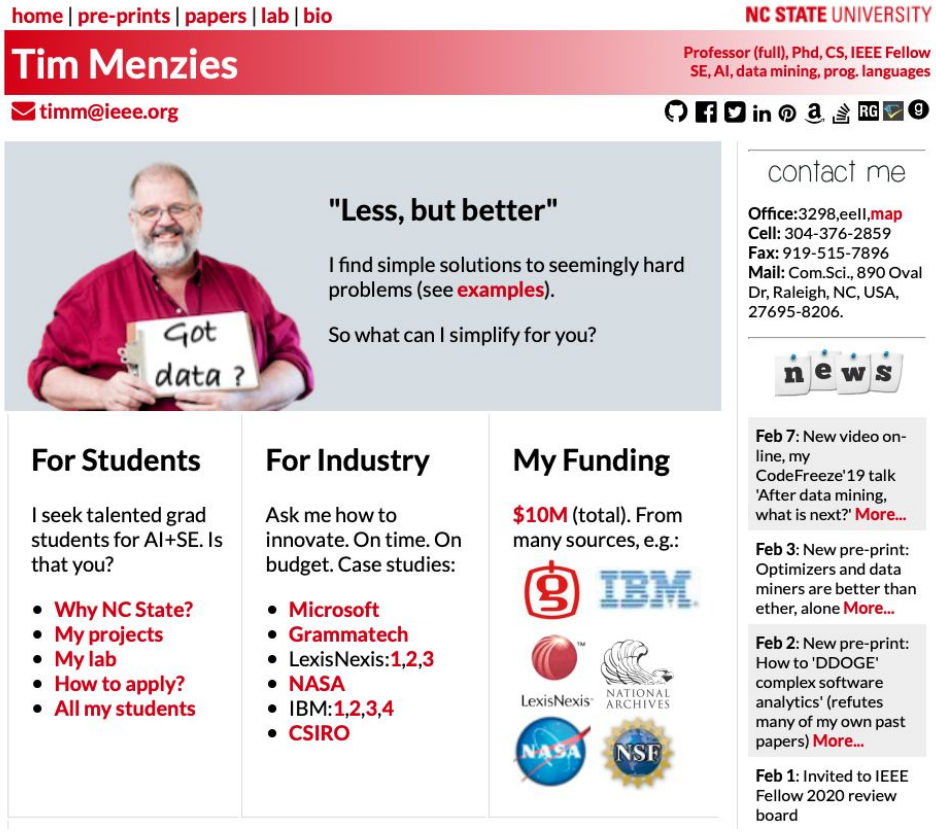
Some can be auto-configured; some need human ingenuity

AI = a large buffet of options. Successful selection, combination of AI tools still requires extensive human expertise

- Many options for combining humans + AI in SE
- Much room for (human) creativity
- e.g. DUO = data miners using/used by optimizers

Me = menzies.us

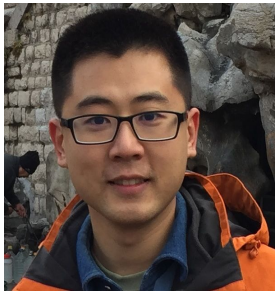
Lab = ai4se.net



Credits, and further reading

Tim Menzies, Leandro Minku, Amritanshu Agrawal, Zhe Yu, Huy Tu, Markus Wagner



Noname manuscript No.
(will be inserted by the editor)

Better Software Analytics via "DUO":
Data Mining Algorithms Using/Used-by Optimizers

Amritanshu Agrawal · Tim Menzies · Leandro L. Minku · Markus Wagner · Zhe Yu

the date of receipt and acceptance should be inserted later

Abstract This paper claims that a new field of empirical software engineering research and practice is emerging: data mining using/used-by optimizers for empirical studies, or DUO. For example, data miners can generate the models that are explored by optimizers. Also, optimizers can advise how to best adjust the control parameters of a data miner. This combined approach acts like an agent leaning over the shoulder of an analyst that advises "ask this question next" or "ignore that problem, it is not relevant to your goals". Further, those agents can help us build "better" predictive models, where "better" can be either greater predictive accuracy, or faster modeling time (which, in turn, enables the exploration of a wider range of options). We also caution that the era of papers that just use data miners is coming to an end. Results obtained from an unoptimized data miner can be quickly refuted, just by applying an optimizer to produce a different (and better performing) model. Our conclusion, hence, is that for software analytics it is possible, useful and necessary to combine data mining and optimization using DUO.

Keywords Software analytics, data mining, optimization, evolutionary algorithms

https://arxiv.org/pdf/1812.01550.pdf

Motivation

Software 2.0: when software writes itself What does that mean, exactly?

Erik Meijer

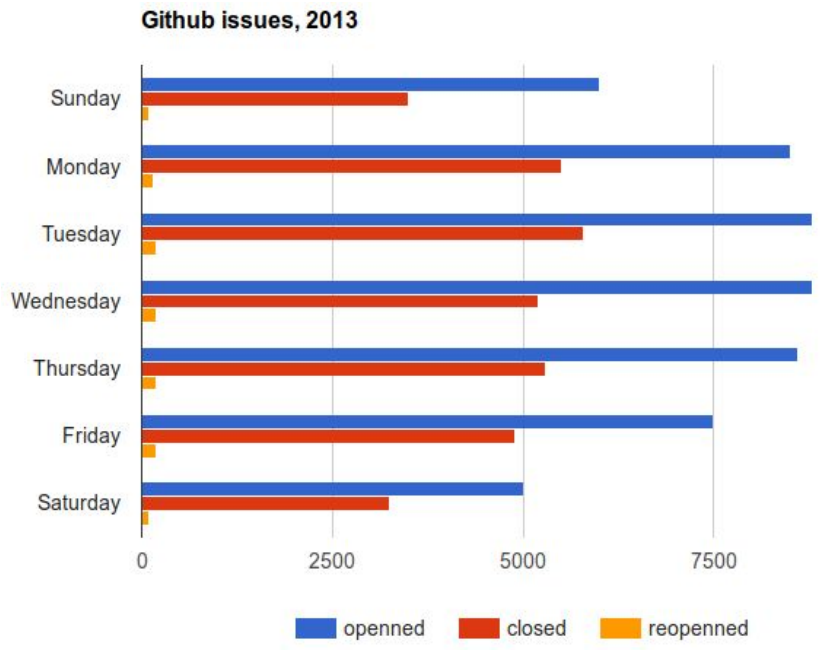
- Behind Every Great Deep Learning Framework Is An Even Greater Programming Languages

- Software 1.0:
  - coffee + human = code

- Software 2.0
  - humans say “what”
  - a agents automatically work out “how”

A counter-view: Programmers > programming

- Still much space for novel human creativity
- Programmers do not just sit in a backroom, just filling in the details to a spec
- Programming is about software engineering
  - The search for feedback on ideas.
  - Half the issues in Github: never closed
  - Programmers spend much time negotiating which, what how, when, who
- Feedback can be handled many ways
  - See below
  - Need humans + automation
    - Not one replacing the other



There are somethings humans just do better than AI

Human factors:

- Convincing you that the system is trustable
- Helping you when the systems fails

Current roles of SE experts
when adopting data mining approaches:

- Problem definition.
- Data collection.
- Data "surrogates" (when the joins fail you)
- Model building.
- **Managing organizational impact**

MINKU, L.L.; MENDES, E.; TURHAN, B. . "Data Mining for SE and Humans in the Loop", *Progress in Artificial Intelligence (PRAI)*, 5(4), 307-314, Nov 2016

Potential roles of SE experts
when adopting data mining approaches :

- Model building involvement.
- Expert domain knowledge.

AI = Not automatic.
A large buffet of options.
Needs skilled human engineers

Can't just throw all problems at huge CPU farms

- Zhe's summer at Google: 3 years of CPU/day
- Wang et al., FSE'13. 15 years of CPU to evaluate software clone configurations
- Truede and Wagner: 30 years of CPU to tune clusters for each corpus

Cannot get humans out of the loop (though some want to try)

- "The notion of user cannot be precisely defined and therefore has no place in CS or SE"
  - Edsger Dijkstra, ICSE 4, 1979

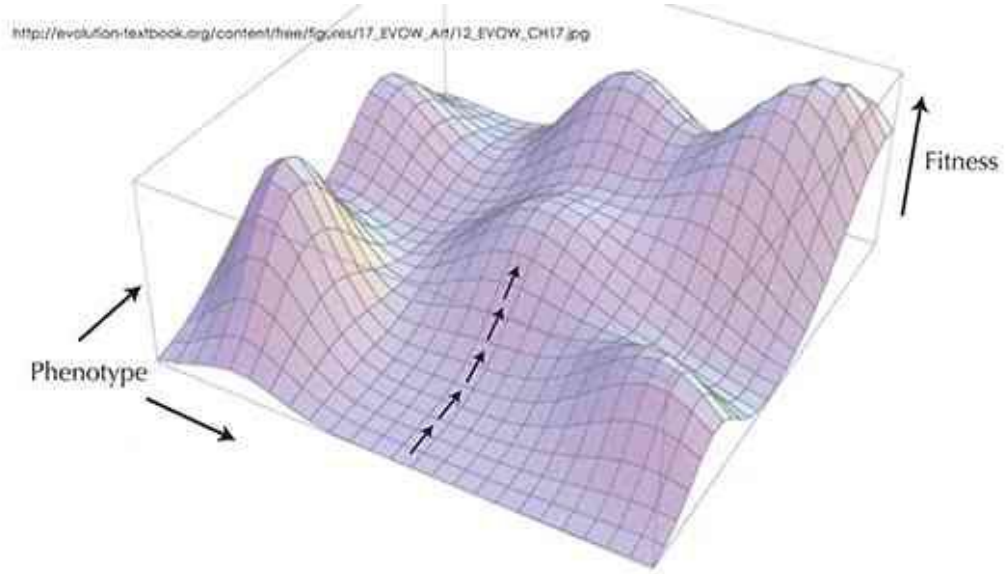
- ".. kill living human experts and resurrect the dead ones"
  - Anonymous machine learning researcher,
  - 1986

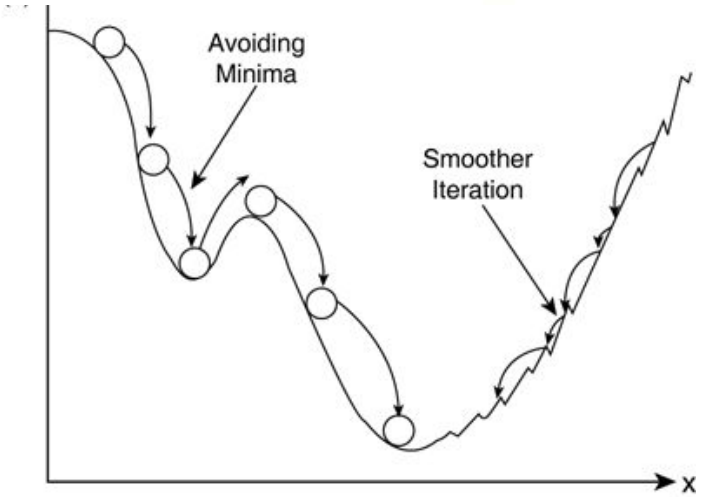
Humans exist. We must work with them

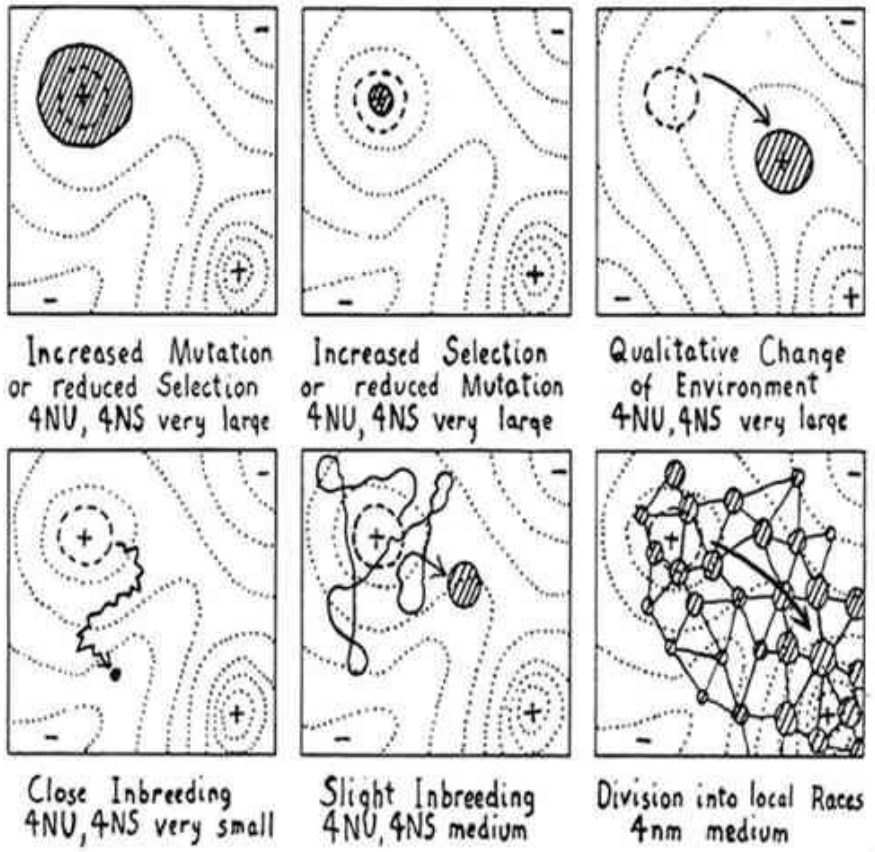
- "The notion of user cannot be precisely defined and therefore has no place in CS or SE"
  - Edsger Dijkstra, ICSE 4, 1979

- Mathematically:
  - "User"= a force that shapes goal space
- Engineering = surfing goal space
- SE = building + using software that surfs
  - Tim Menzies, FSE 2018

The shape of goal space? Surfing?







Brand's favorite landscape, 1932 Rabbits.

Complaint: "But that's not SE" Reply "Times change"

- Once upon a time, …
  - SE was not about users (Dijkstra, 79)
  - SE was not about testing (cleanroom, Mills, 1985)
  - SE was not about requirements (Paulk, 1993)
    - "Analysis and allocation of the system requirements is NOT the responsibility of the SE group but is a prerequisite for their work"
  - SE was not about deployment (before CI)
  - SE was not about AI

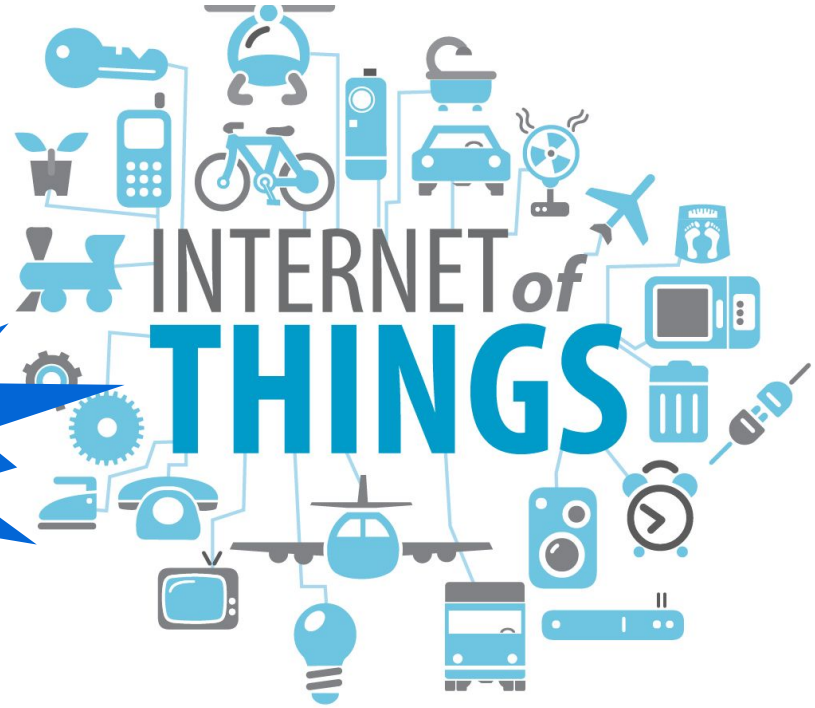
- But AI software is still software
  - needs maintenance, validation, interfacing, usability additions
  - i.e. AI needs software engineers

Software mediates what we see and how we act in the world

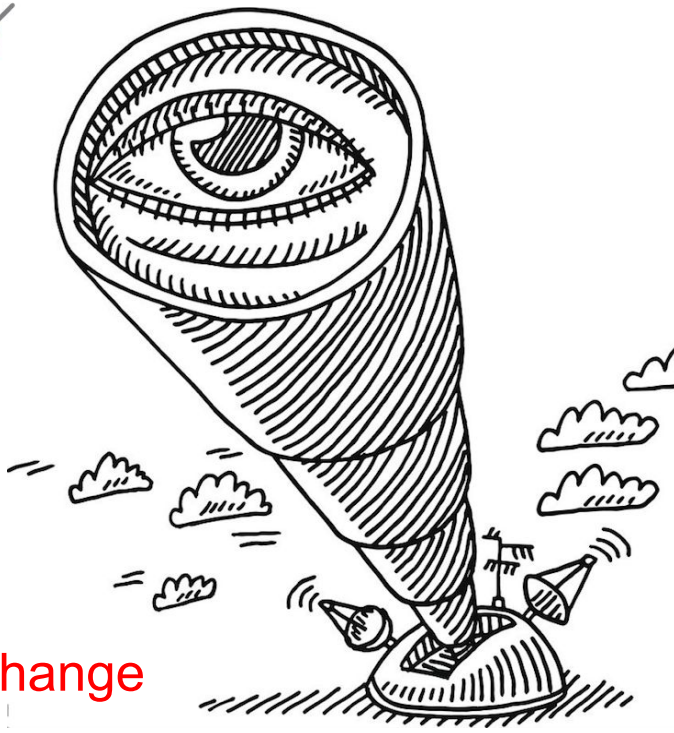
- Silicon valley developers view **every new feature as an experiment**, to be tested within some mash up.
- Chemists win Nobel Prize for **software sims** http://goo.gl/Lwensc
- Engineers use **software to design** optical tweezers, radiation therapy, remote sensing, chip design, http://goo.gl/qBMyIZ
- Web analysts use **software to analyze clickstreams** to improve sales and marketing strategies; http://goo.gl/b26CfY
- Stock traders write software to **simulate trading** strategies http://www.quantopian.com
- Analysts write software to **mine labor statistics** data to review proposed gov policies http://goo.gl/X4kgnc
- Journalists use software to **analyze economic data**, make visualizations of their news stories http://fivethirtyeight.com
- In London or New York, ambulances **wait for your call** at a location determined by a software model http://goo.gl/8SMd1p

Time to think much BIGGER
Time to solve problems (not just implement solutions)

- Internet of **things. that matter**



SE: not just about software anymore?

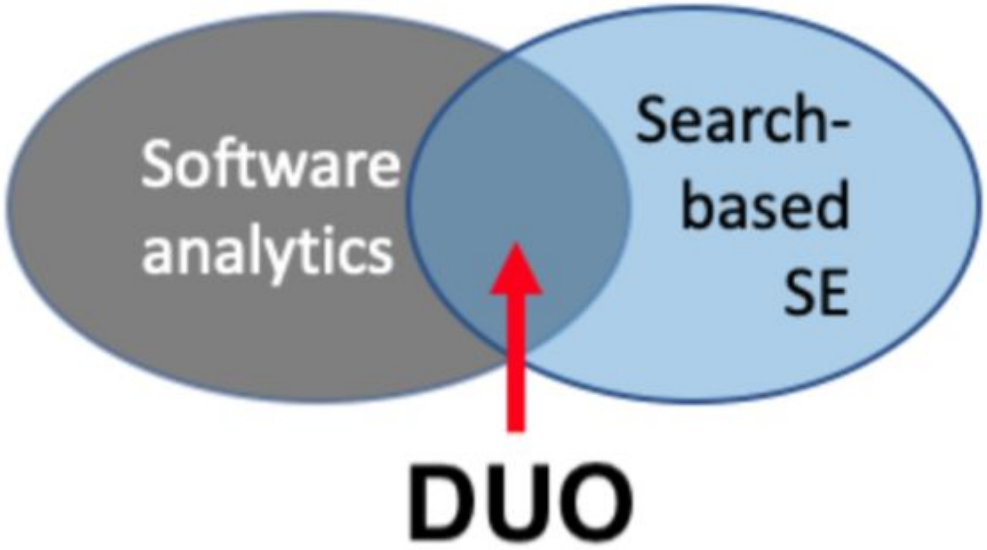


- After "continuous integration" (automate everything)
  - Comes "AI everywhere" (automate automation)

- Old SE: just grind the lens of the telescope
  - New SE: use telescope to look & understand & change

Definitions

DUO = data miners using/used by optimizers


Software analytics
Search-based SE
DUO

Definitions

- Data mining
  - Patterns from data
  - "What is"
  - e.g. learn a decision tree
- Optimization
  - Trends in the data
  - "What to do"
  - e.g. propose a delta across a tree
- Hyperparameter optimization
  - Tuning decisions within a learner
  - e.g. when to stop growing

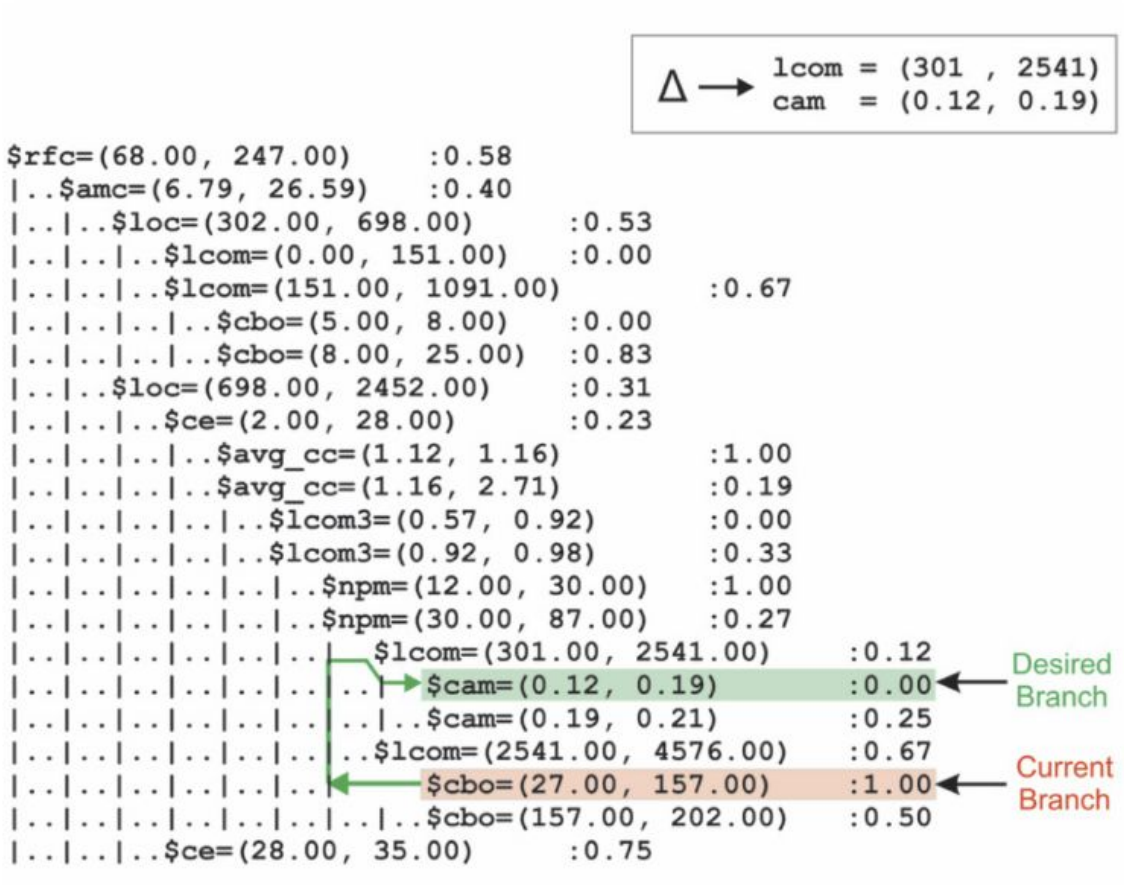


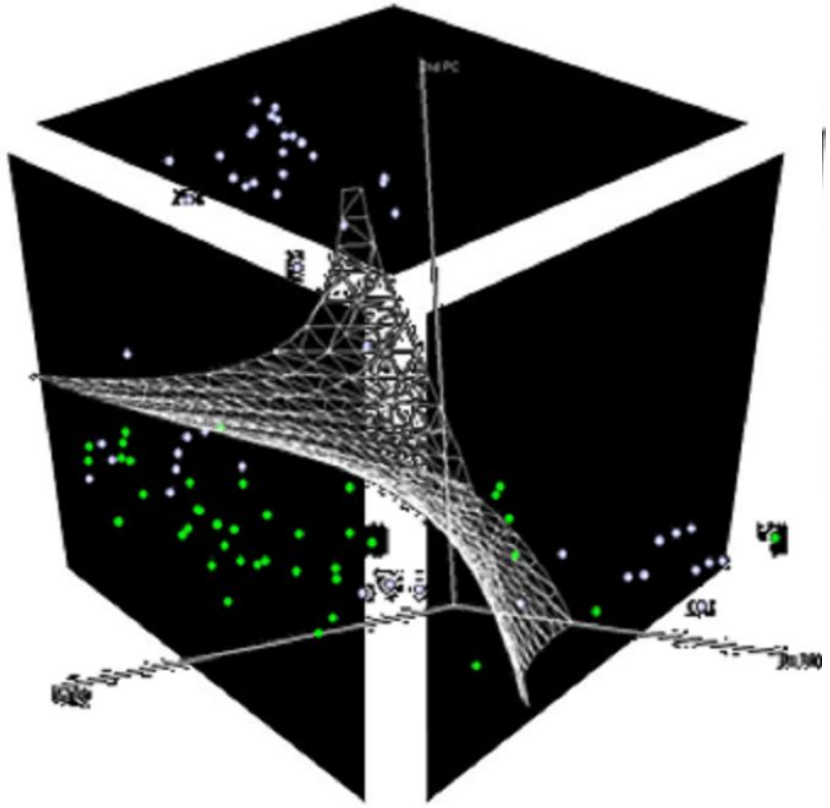
Fig. 1: Generating thresholds using XTREE.

Data Mining ≈ Optimization

What's inside the optimizer?

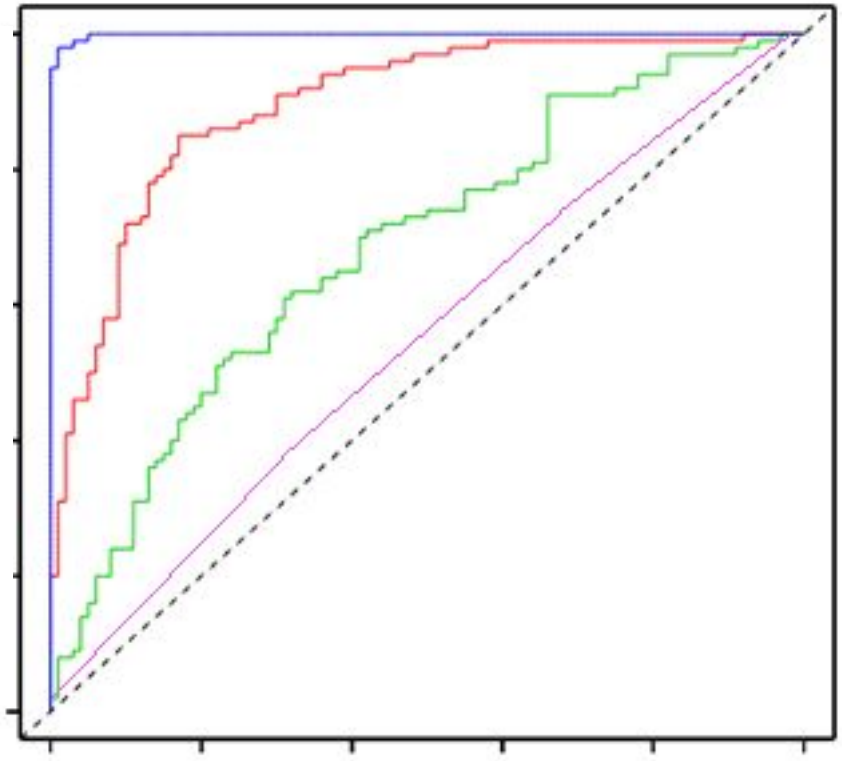
Data Mining ≈ Optimization

What's inside the optimizer?



Data Mining ≈ Optimization

Project Management.

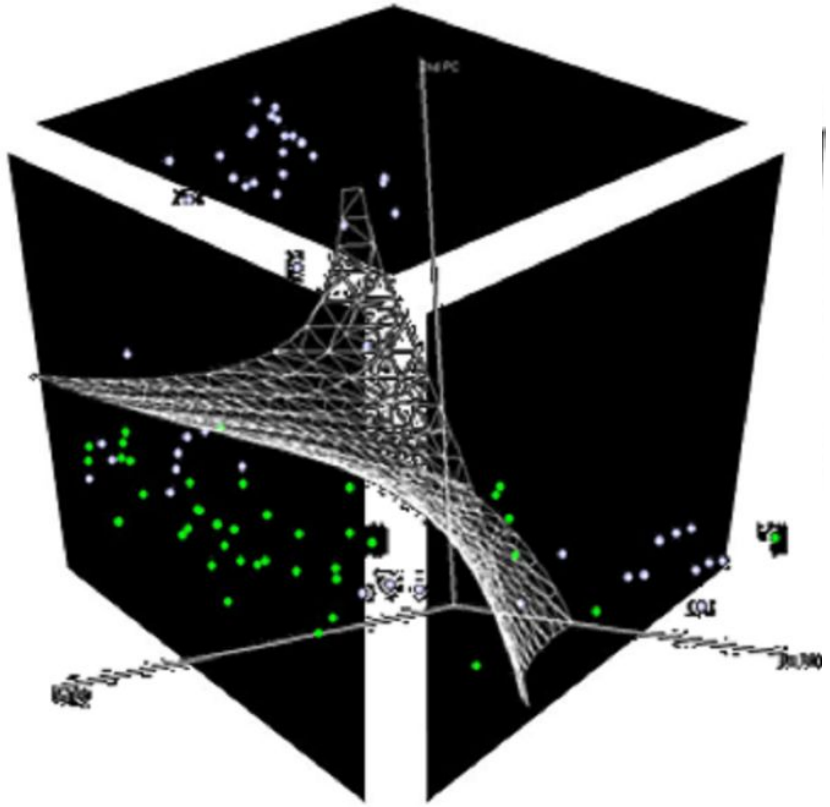


Data Mining ≈ Optimization

What's inside the data miner?

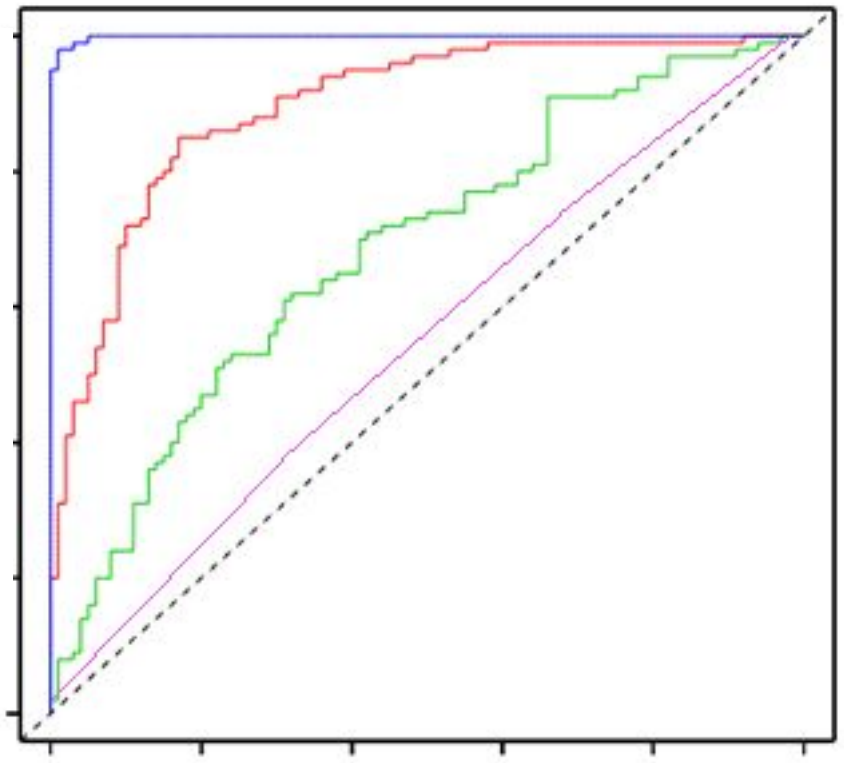
Data Mining ≈ Optimization

What's inside the data miner?



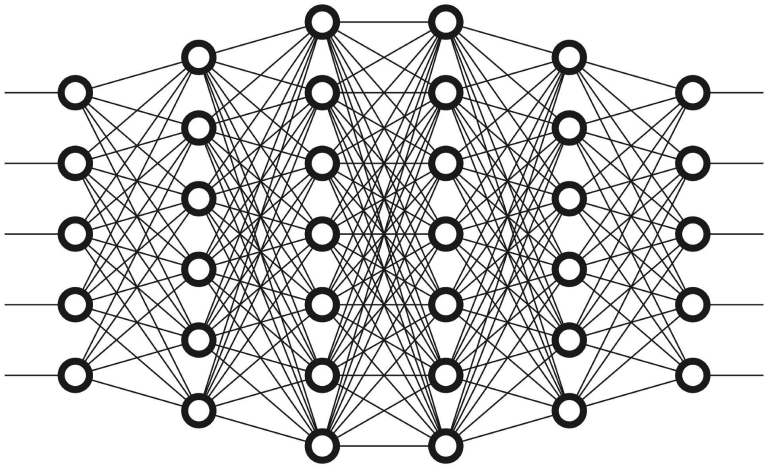
Data Mining ≈ Optimization

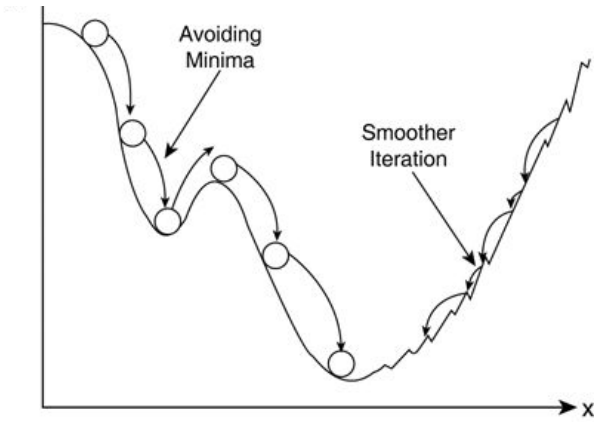
Software quality, defect prediction

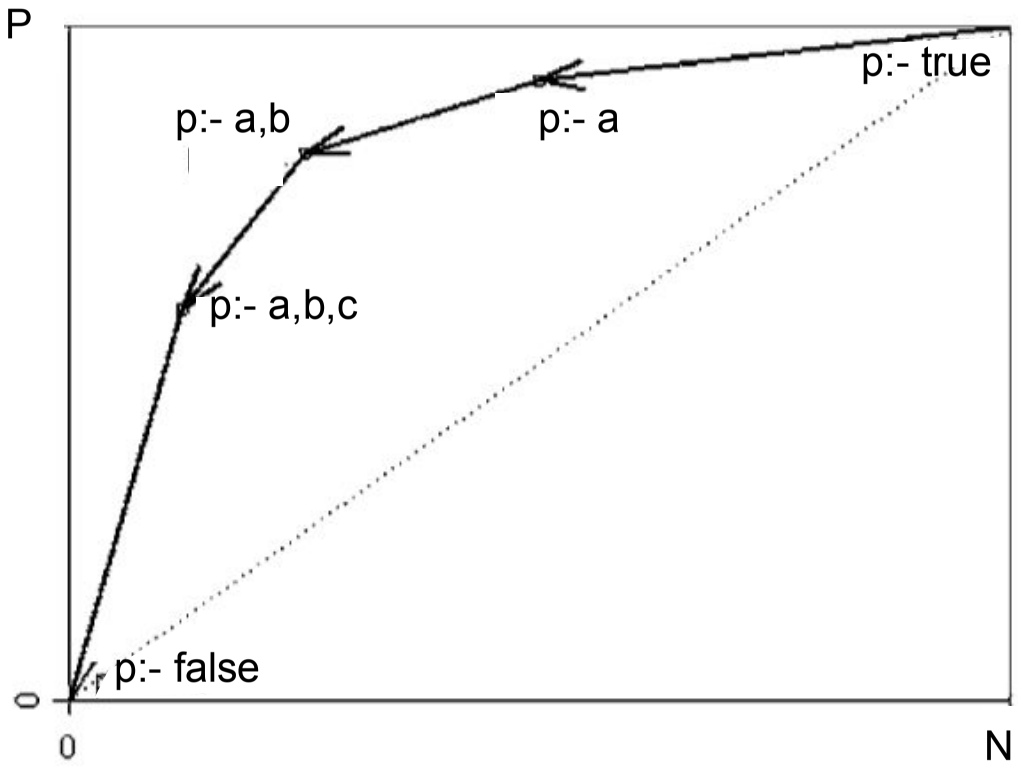


Data Mining ≈ Optimization

Many machine learning algorithms ARE optimization algorithms!







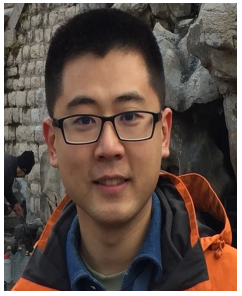
Literature Review

(use of DUO in SE)

Credits, and further reading

Tim Menzies, Leandro Minku, Amritanshu Agrawal, Zhe Yu, Huy Tu, Markus Wagner



Noname manuscript No.
(will be inserted by the editor)

Better Software Analytics via "DUO":
Data Mining Algorithms Using/Used-by Optimizers

Amritanshu Agrawal · Tim Menzies · Leandro L. Minku · Markus Wagner · Zhe Yu

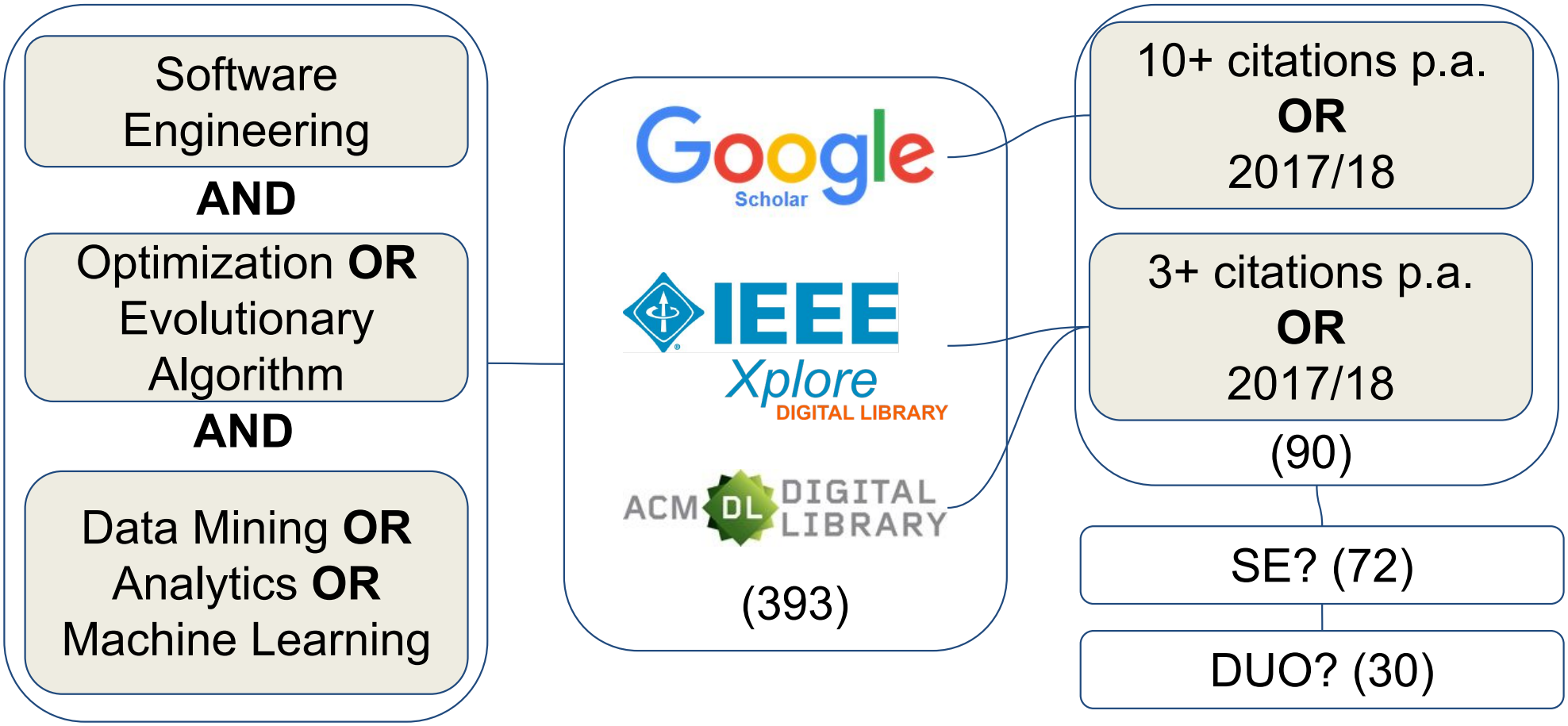
the date of receipt and acceptance should be inserted later

Abstract This paper claims that a new field of empirical software engineering research and practice is emerging: data mining using/used-by optimizers for empirical studies, or DUO. For example, data miners can generate the models that are explored by optimizers. Also, optimizers can advise how to best adjust the control parameters of a data miner. This combined approach acts like an agent leaning over the shoulder of an analyst that advises "ask this question next" or "ignore that problem, it is not relevant to your goals". Further, those agents can help us build "better" predictive models, where "better" can be either greater predictive accuracy, or faster modeling time (which, in turn, enables the exploration of a wider range of options). We also caution that the era of papers that just use data miners is coming to an end. Results obtained from an unoptimized data miner can be quickly refuted, just by applying an optimizer to produce a different (and better performing) model. Our conclusion, hence, is that for software analytics it is possible, useful and necessary to combine data mining and optimization using DUO.

Keywords Software analytics, data mining, optimization, evolutionary algorithms

https://arxiv.org/pdf/1812.01550.pdf

Data mining, optimization, and software engineering



References

1. Sun-Jen Huang and Nan-Hsing Chiu. Optimization of analogy weights by genetic algorithm for software effort estimation. *Information and software technology*, 48(11):1034–1045, 2006.
2. Nan-Hsing Chiu and Sun-Jen Huang. The adjusted analogy-based software effort estimation based on similarity distances. *Journal of Systems and Software*, 80(4):628–640, 2007.
3. Adriano LI Oliveira, Petronio L Braga, Ricardo MF Lima, and Márcio L Cornélio. Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. *information and Software Technology*, 52(11):1155–1166, 2010.
4. Andre B De Carvalho, Aurora Pozo, and Silvia Regina Vergilio. A symbolic fault-prediction model based on multiobjective particle swarm optimization. *Journal of Systems and Software*, 83(5):868–882, 2010.
5. Wasif Afzal and Richard Torkar. On the application of genetic programming for software engineering predictive modeling: A systematic review. *Expert Systems with Applications*, 38(9):11984–11997, 2011.
6. Leandro L Minku and Xin Yao. Software effort estimation as a multiobjective learning problem. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 22(4):35, 2013.
7. Annibale Panichella, Bogdan Dit, Rocco Oliveto, Massimiliano Di Penta, Denys Poshyvanyk, and Andrea De Lucia. How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 522–531. IEEE Press, 2013.
8. Leandro L Minku and Xin Yao. An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation. In *Proceedings of the 9th international conference on predictive models in software engineering*, page 8. ACM, 2013.
9. Federica Sarro, Sergio Di Martino, Filomena Ferrucci, and Carmine Gravino. A further analysis on the use of genetic algorithm to configure support vector machines for inter-release fault prediction. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 1215–1220. ACM, 2012.

10. Xin Du, Xin Yao, Youcong Ni, Leandro L Minku, Peng Ye, and Ruliang Xiao. An evolutionary algorithm for performance optimization at software architecture level. In *Evolutionary Computation (CEC), 2015 IEEE Congress on*, pages 2129–2136. IEEE, 2015.
11. Jianfeng Chen, Vivek Nair, and Tim Menzies. Beyond evolutionary algorithms for search-based software engineering. *Information and Software Technology*, 95:281–294, 2018.
12. Wei Fu, Tim Menzies, and Xipeng Shen. Tuning for software analytics: Is it really necessary? *Information and Software Technology*, 76:135–146, 2016.
13. Federica Sarro, Alessio Petrozziello, and Mark Harman. Multi-objective software effort estimation. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 619–630. IEEE, 2016.
14. Wei Fu and Tim Menzies. Easy over hard: A case study on deep learning. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 49–60. ACM, 2017.
15. Mohammed Hasan Ali, Bahaa Abbas Dawood Al Mohammed, Alyani Ismail, and Mohamad Fadli Zolkipli. A new intrusion detection system based on fast learning network and particle swarm optimization. *IEEE Access*, 6:20255–20261, 2018.
16. Ali Safaa Sadiq, Basem Alkazemi, Seyedali Mirjalili, Noraziah Ahmed, Suleman Khan, Ihsan Ali, Al-Sakib Khan Pathan, and Kayhan Zrar Ghafoor. An efficient ids using hybrid magnetic swarm optimization in wanets. *IEEE Access*, 6:29041–29053, 2018.
17. Amritanshu Agrawal and Tim Menzies. Is better data better than better data miners?: on the benefits of tuning smote for defect prediction. In *Proceedings of the 40th International Conference on Software Engineering*, pages 1050–1061. ACM, 2018.
18. Amritanshu Agrawal, Wei Fu, and Tim Menzies. What is wrong with topic modeling? and how to fix it using search-based software engineering. *Information and Software Technology*, 98:74–88, 2018.
19. Raja Ben Abdessalem, Shiva Nejati, Lionel C. Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, pages 1016–1026, New York, NY, USA, 2018. ACM.

20. Suvodeep Majumder, Nikhila Balaji, Katie Brey, Wei Fu, and Tim Menzies. 500+ times faster than deep learning (a case study exploring faster methods for text mining stackoverflow). *arXiv preprint arXiv:1802.05319*, 2018.
21. Vivek Nair, Zhe Yu, Tim Menzies, Norbert Siegmund, and Sven Apel. Finding faster configurations using Flash. *arXiv preprint arXiv:1801.02175*, 2018.
22. Christoph Treude and Markus Wagner. Per-corpus configuration of topic modelling for github and stack overflow collections. *arXiv preprint arXiv:1804.04749*, 2018.
23. Cagatay Catal and Banu Diri. Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem. *Information Sciences*, 179(8):1040–1058, 2009.
24. Olivier Vandecruys, David Martens, Bart Baesens, Christophe Mues, Manu De Backer, and Raf Haesen. Mining software repositories for comprehensible software fault prediction models. *Journal of Systems and software*, 81(5):823–839, 2008.
25. Andrea Arcuri and Gordon Fraser. On parameter tuning in search based software engineering. In *International Symposium on Search Based Software Engineering*, pages 33–47. Springer, 2011.
26. Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 492–501. IEEE Press, 2013.
27. Chakkrit Tantithamthavorn, Shane McIntosh, Ahmed E Hassan, and Kenichi Matsumoto. Automated parameter optimization of classification techniques for defect prediction models. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 321–332. IEEE, 2016.
28. Mark Stephenson, Saman Amarasinghe, Martin Martin, and Una-May O'Reilly. Meta optimization: improving compiler heuristics with machine learning. In *ACM SIGPLAN Notices*, volume 38, pages 77–90. ACM, 2003.
29. Alaa F Sheta. Estimation of the cocomo model parameters using genetic algorithms for nasa software projects. *Journal of Computer Science*, 2(2):118–123, 2006.
30. Eugenia Díaz, Javier Tuya, Raquel Blanco, and José Javier Dolado. A tabu search algorithm for structural software testing. *Computers & Operations Research*, 35(10):3052–3072, 2008.

Systematic Literature Review

Optimizer?

Data Miner?

Domain?

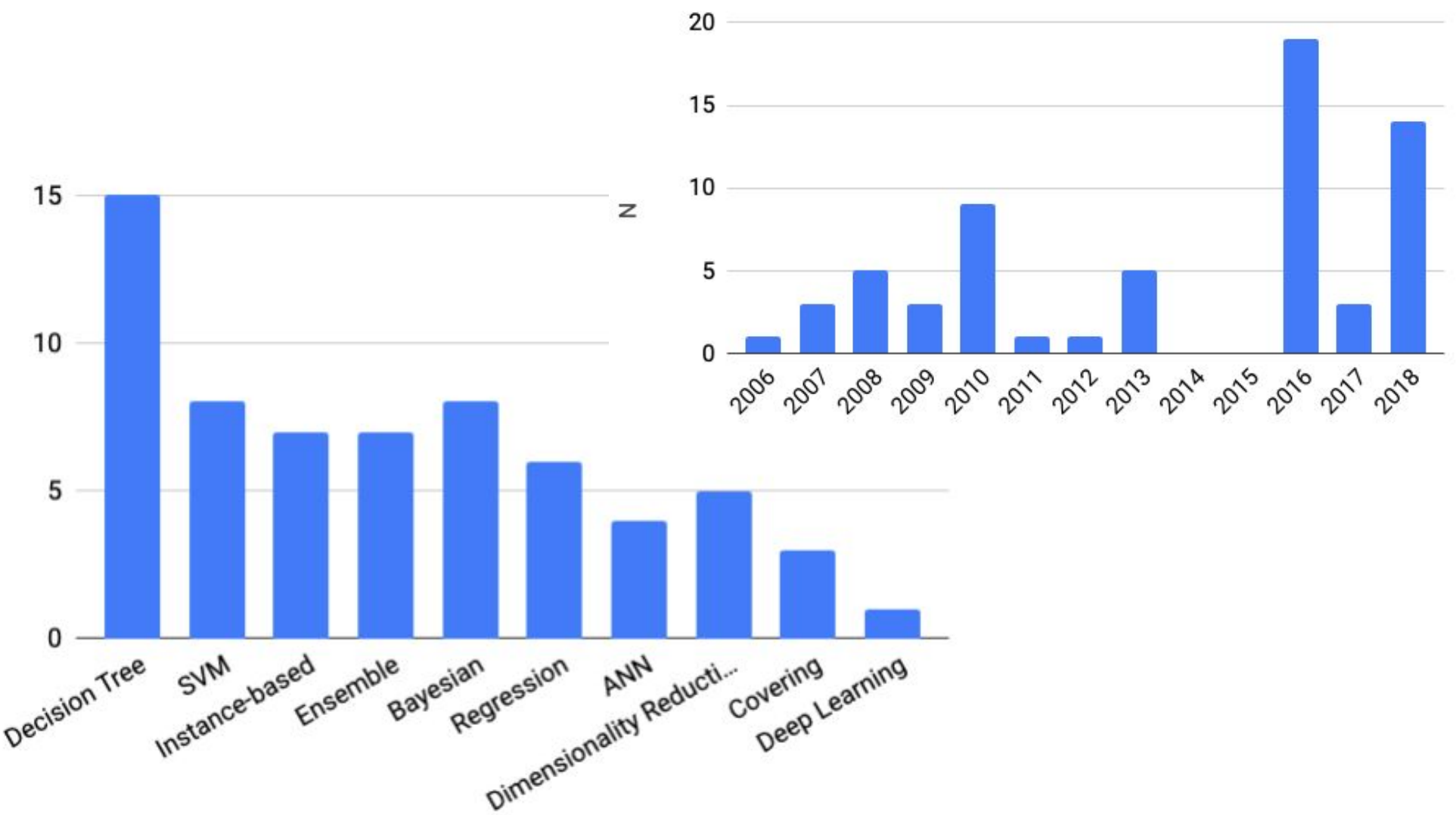
How?

Problem?

Take-home message

Domain	Problem
Project management	Effort estimation, dramatic reductions in errors
Requirements	Faster navigation of complex stakeholder conflicts, assessing, software architecture decisions (faster)
Design	Extraction of products from very large product lines
Security	Configuring intrusion detection
Software quality	Detect prediction, test case generation
Software configuration	10^5 options explored using < 100 samples
Text mining and topic modeling: StackOverflow	Improved classification performance using stabler topics.
Text mining and topic modeling: Defect Reports	Stabler topics using LDADE
Text mining and topic modeling: Software Artifact Labeling	Using active learner to label commit data

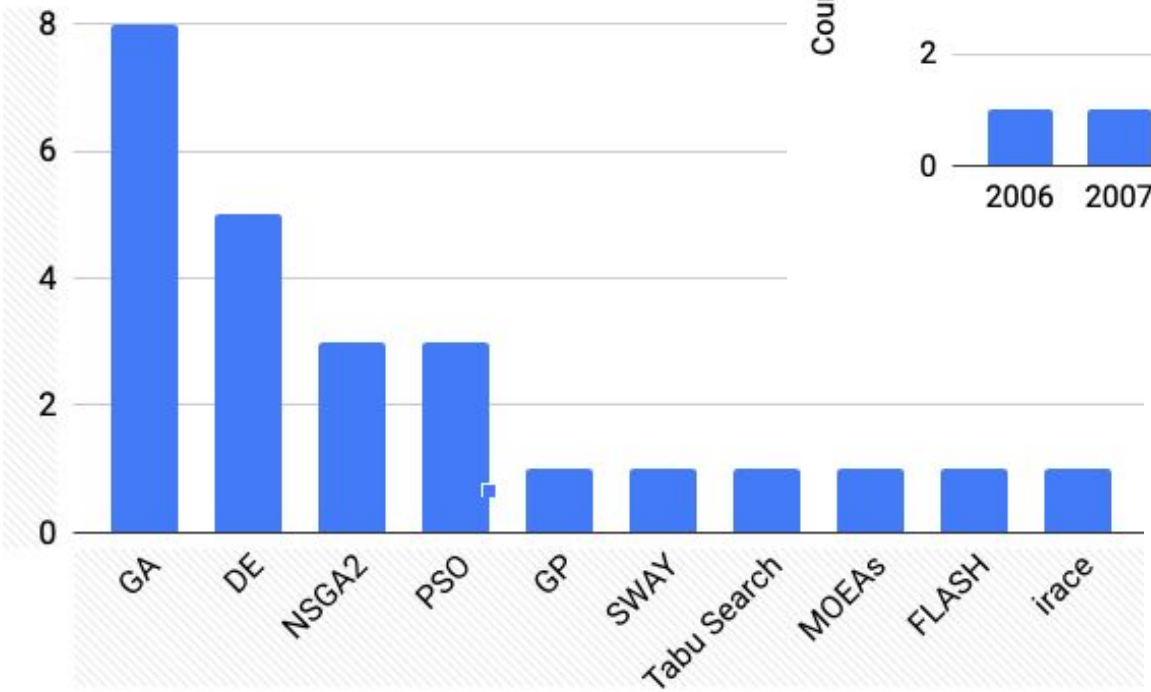
DUO: Data miners in our corpus

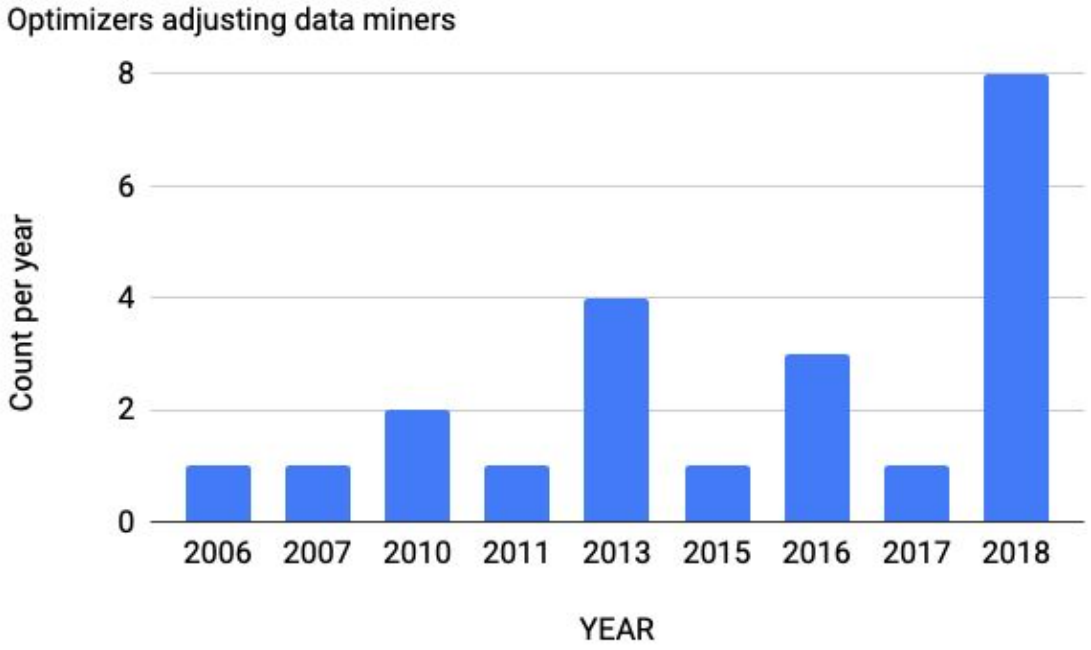


DUO: Optimizers controlling data miners (in our corpus)

some acronyms:

GA= genetic algorithms
DE = differential evolution
PSO=particle swarm optimization





Four flavors of DUO

Four flavors of DUO

Data Miner **is** the Optimizer, e.g.,

- Krall et al. TSE 2015

Optimizer **controls** data miner,

- Tantithamthavorn et al ICSE 2016
- Agrawal et al. ICSE 18
- Fu et al. IST 2016

Data Miner **controls** optimizer, e.g.,

- Majumder et al. MSR 18

Dancing with both, e.g.,

- Abdessalem et al. 2018
- Data Driven SBSE, Nair et al. MSR 2018

Four flavors of DUO

Data Miner **is** the Optimizer, e.g.,

- Krall et al. TSE 2015

Optimizer **controls** data miner,

- Tantithamthavorn et al ICSE 2016
- Agrawal et al. ICSE 18
- Fu et al. IST 2016

Data Miner controls optimizer, e.g.,

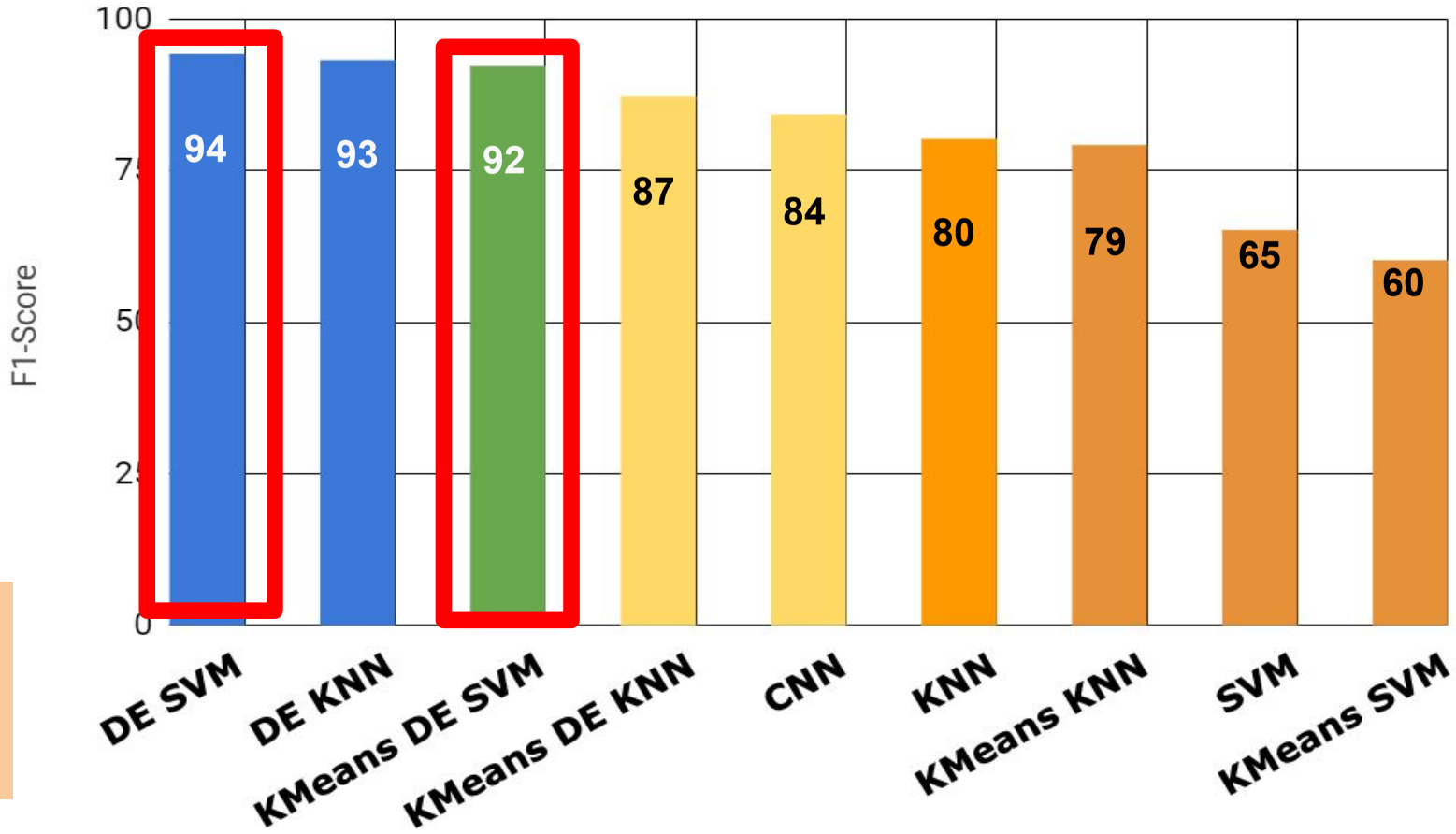
- **Majumder et al. MSR 18**

Dancing with both, e.g.,

- Abdessalem et al. 2018
- Data Driven SBSE, Nair et al. MSR 2018

Data Miner Helping Optimizer

- Problem = finding similar questions in Stackoverflow
- Build local clusters with Data Miner (KMeans), run optimizer (DE) and data miner (SVM) on those local clusters.
- Performing as good as using data miner with optimizer.
  - Majumder et al. MSR 2018 DE SVM ≅ KMeans DE SVM.



Text mining

Four flavors of DUO

Data Miner **is** the Optimizer, e.g.,

- Krall et al. TSE 2015

Optimizer controls data miner,

- **Tantithamthavorn et al ICSE 2016**
- **Agrawal et al. ICSE 18**
- **Fu et al. IST 2016**

Data Miner **controls** optimizer, e.g.,

- Majumder et al. MSR 18

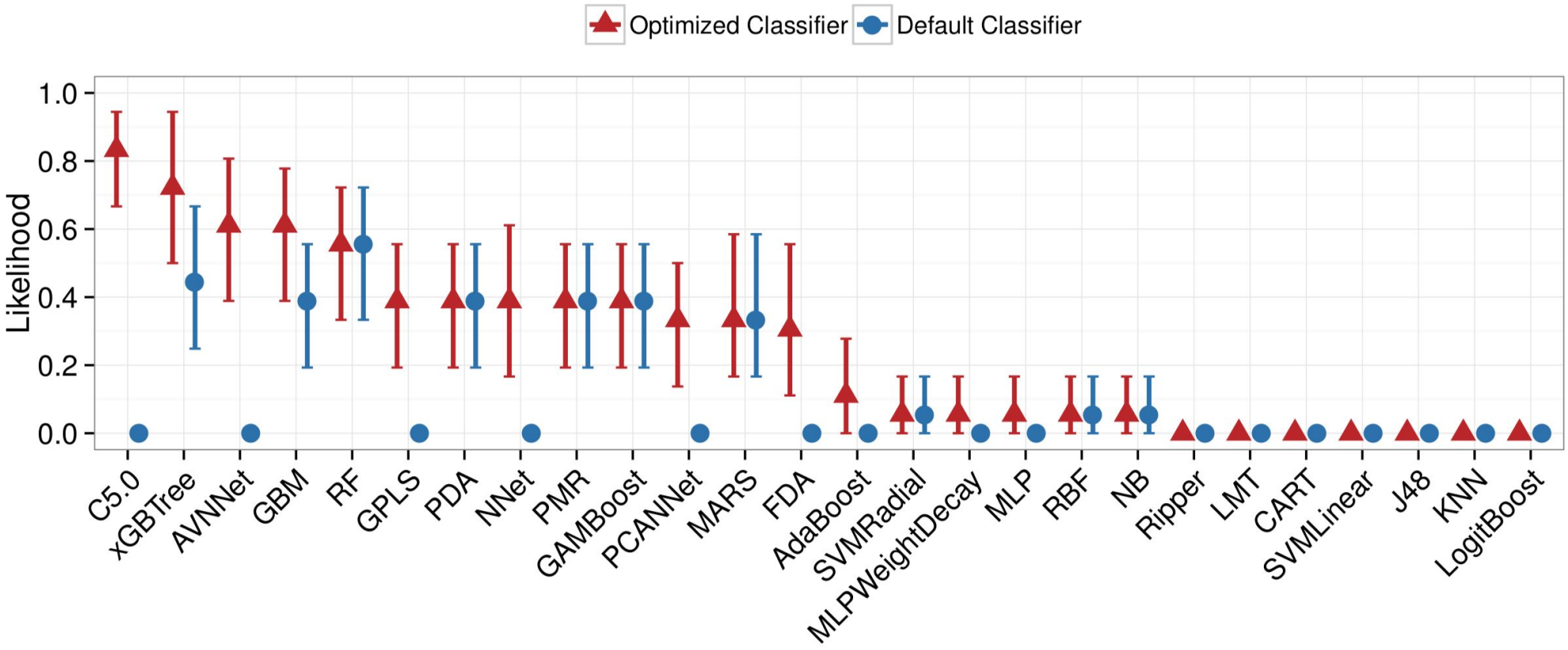
Dancing with both, e.g.,

- Abdessalem et al. 2018
- Data Driven SBSE, Nair et al. MSR 2018

Optimizer improving Data Mining

- Multiple data miners improved via optimizer Tantithamthavorn et al ICSE 2016.

Software quality, defect prediction



Four flavors of DUO

Data Miner **is** the Optimizer, e.g.,

- Krall et al. TSE 2015

Optimizer **controls** data miner,

- Tantithamthavorn et al ICSE 2016
- Agrawal et al. ICSE 18
- Fu et al. IST 2016

Data Miner **controls** optimizer, e.g.,

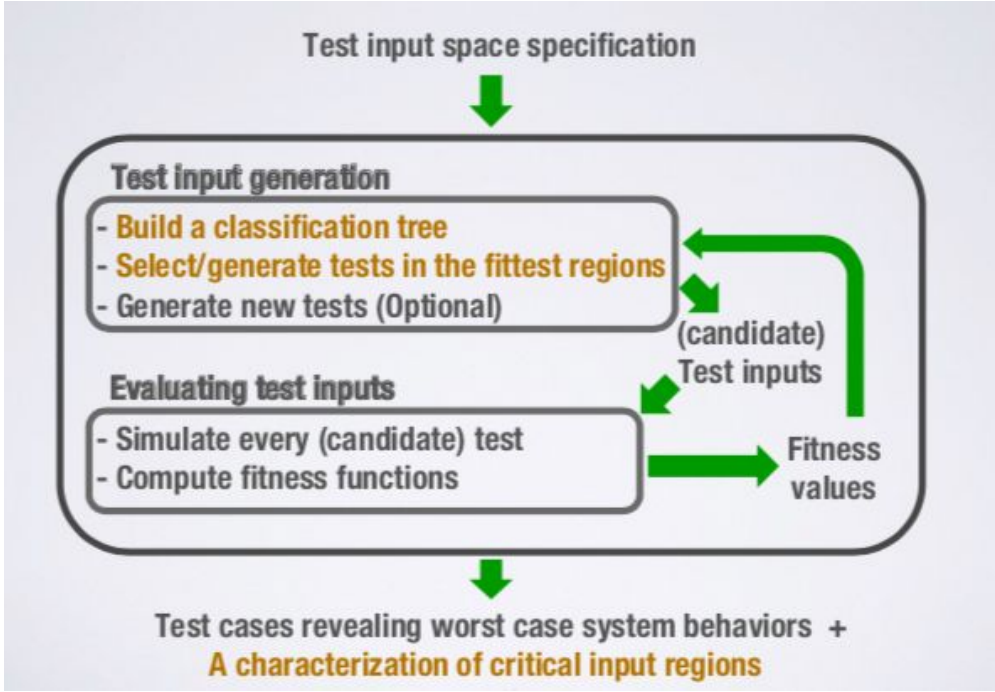
- Majumder et al. MSR 18

Dancing with both, e.g.,

- **Abdessalem et al. 2018**
- **Data Driven SBSE, Nair et al. MSR 2018**

So many innovative and useful ways to combine optimizers and data miners

- **Testing Vision-Based Control Systems Using Learnable Evolutionary Algorithms**
  - ICSE'18
  - Abdessalem, Nejati, Briand, Stifter
- Vision, autonomy, self-driving cars
- Complicated by complex and multidimensional input spaces
- Machine learning and Darwinian genetic operators to generate test scenarios



Software quality, Test case generation

Four flavors of DUO

Data Miner is the Optimizer, e.g.,

- **Krall et al. TSE 2015**

Optimizer **controls** data miner,

- Tantithamthavorn et al ICSE 2016
- Agrawal et al. ICSE 18
- Fu et al. IST 2016

Data Miner **controls** optimizer, e.g.,

- Majumder et al. MSR 18

Dancing with both, e.g.,

- Abdessalem et al. 2018
- Data Driven SBSE, Nair et al. MSR 2018

GALE = configuration = Data mining? optimizer?

Approximate the space

- k=2 divisive clustering

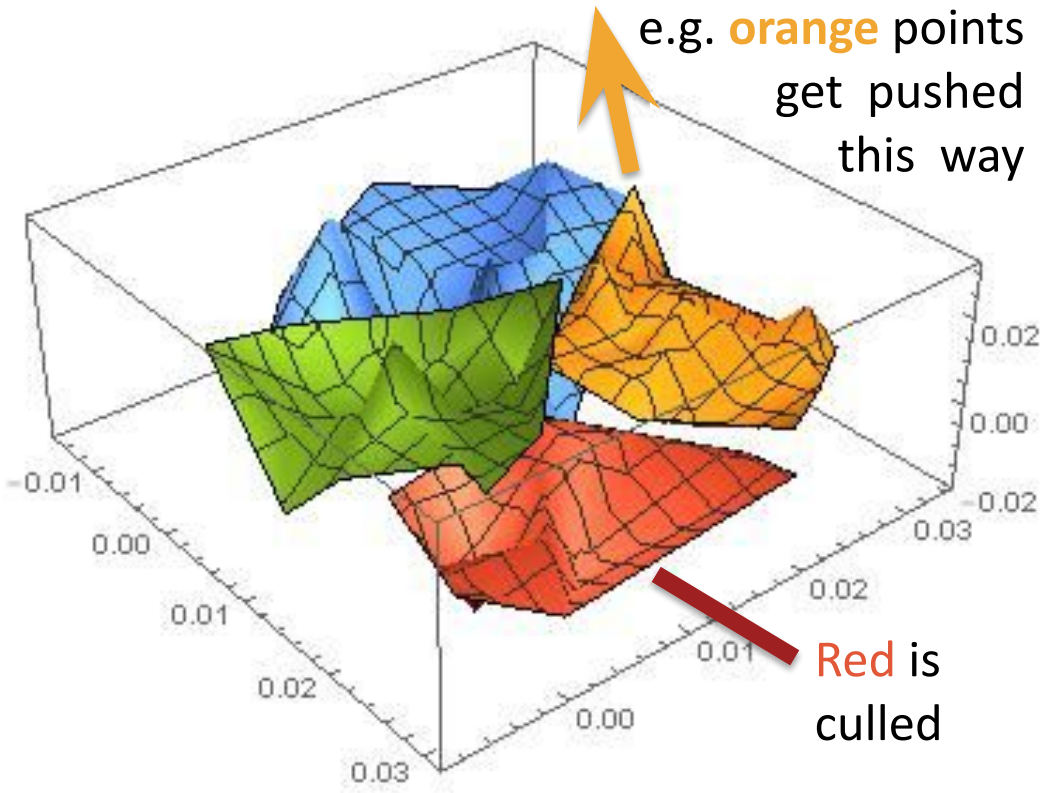
(X,Y)= 2 very distant points in O(2N)

Evaluate only (X,Y)

If better(X,Y)

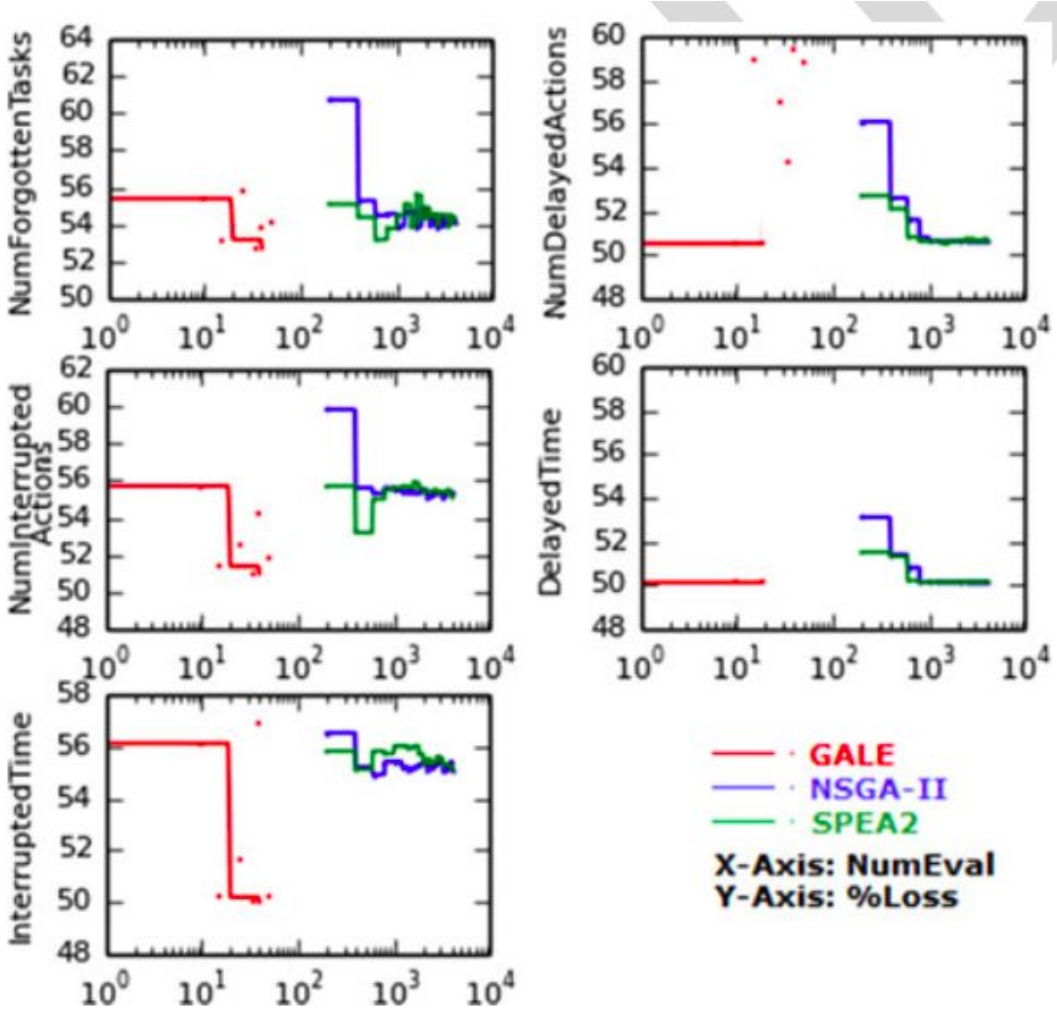
- If size(cluster) > sqrt(N)
  - Split, recurse on better half
  - E.g. cull red
- Else, push points towards X
  - E.g. push orange

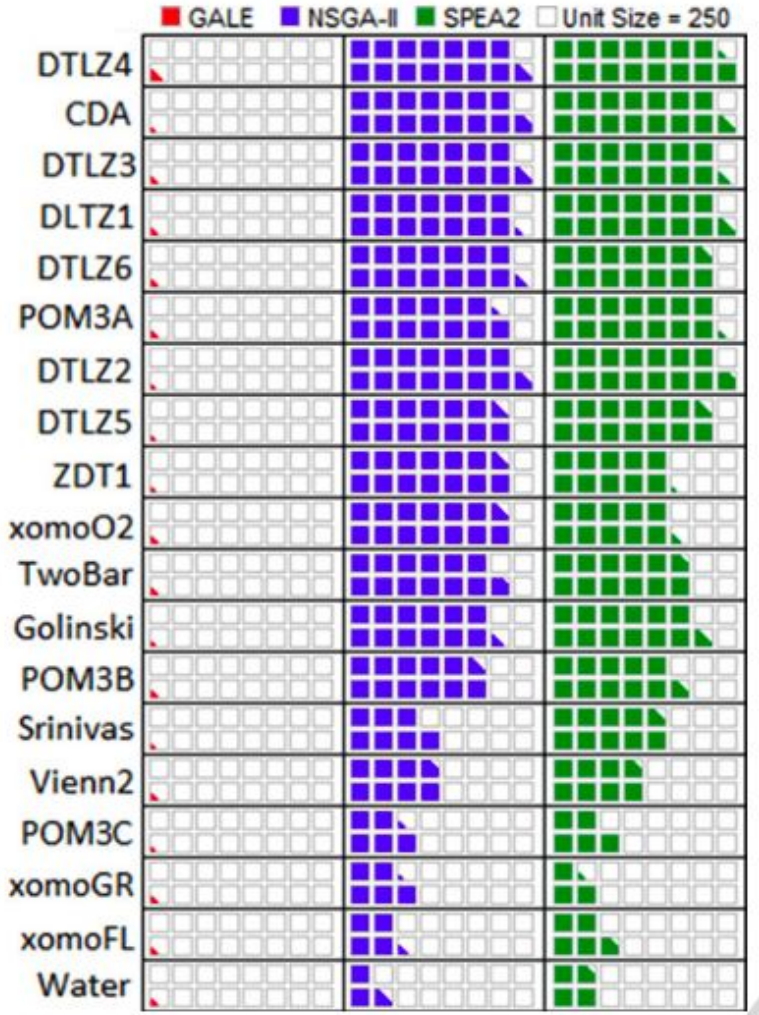
Software configuration, Options explored



[Krall, Menzies, TSE, 2015]

GALE = Explore complex avionics requirements models





GALE 4 mins vs NSGA-II 8 hours

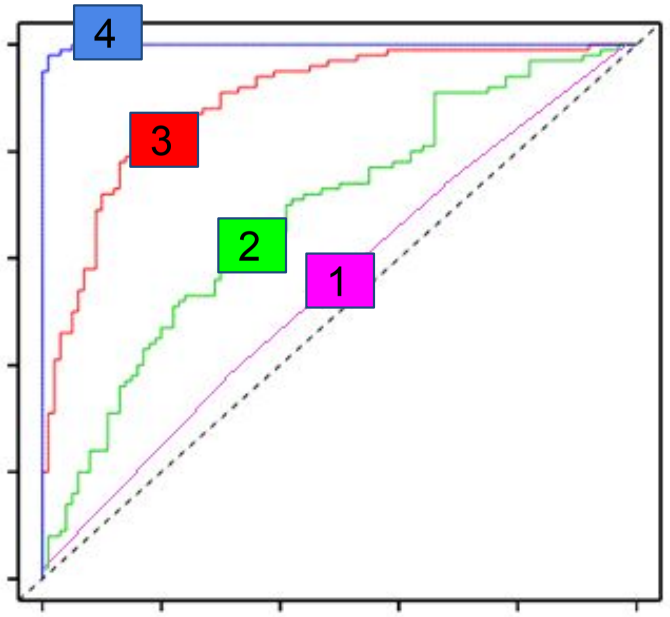
Bad smell: Data Mining without Optimization

- How to get a paper rejected (in 2020):
  - Publish data mining results **without** optimizater doing the tuning
- Coming to the end of "merely mining"
  - See debates on "unsupervised learning"
    - Too easy to just chase precision, recall etc
- Complex problems need complex inference
  - e.g.minimizing #false alarms before first defect [Huang et al.ICSME'17]
  - Needed to reply to (e.g.) [Parnin, Orso, Issta'11]

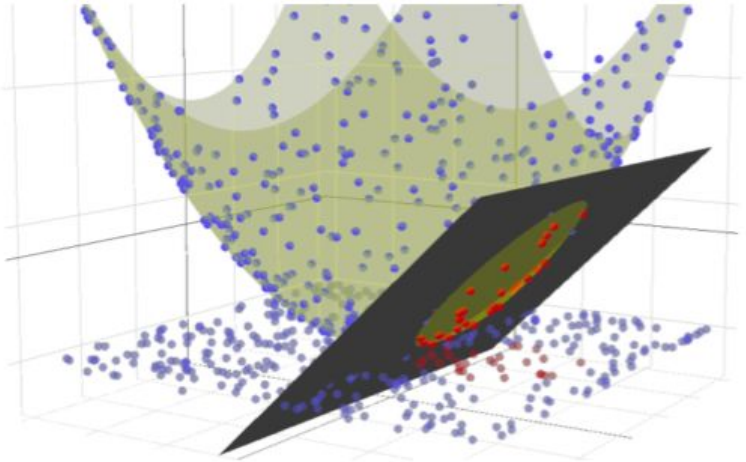
Advantages of DUO

Why add optimizers to data miners?

- Because optimization technology knows more about goals:
  - **Multiple competing goals**
    - pareto frontier,
    - domination
  - **Wider range of goals**
    - Data miners have goals “baked in” (e.g. minimize entropy)
    - But optimisers accept arbitrary goals as input



Why add data miners to optimizers?



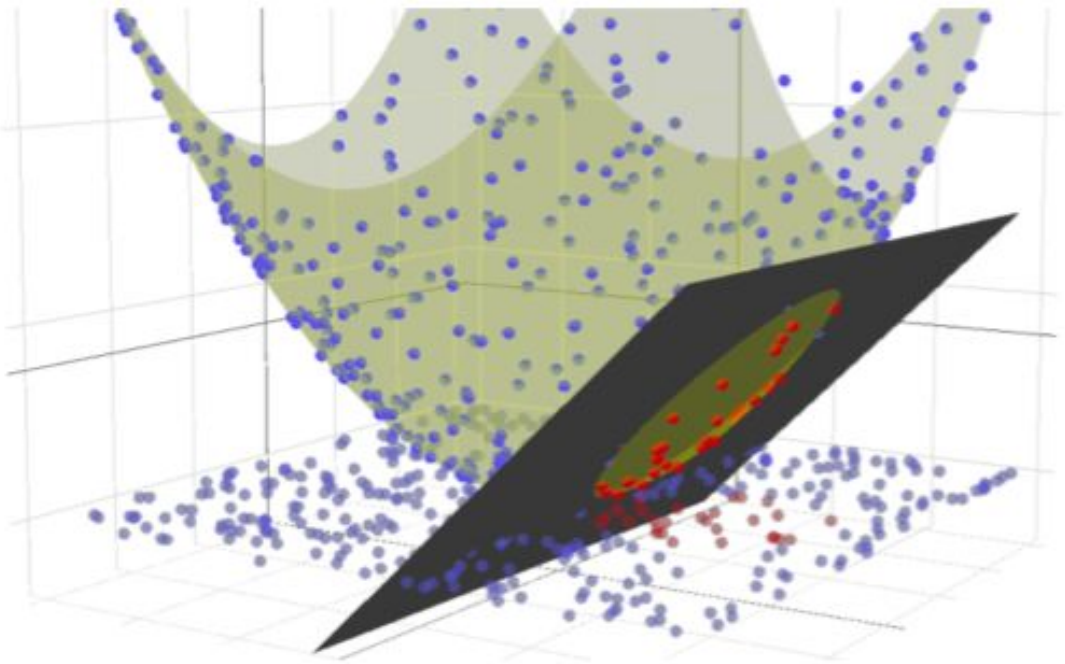
```
sccp=0
|    print_used_types=0
|    |    ipsccp=0 (6.5)
|    |    ipsccp=1
|    |    |    x[10]=0 (12)
|    |    |    x[10]=1 (19)
|    print_used_types=1
|    |    ipsccp=0 (14)
|    |    ipsccp=1
|    |    |    time_passes=0   (24)
|    |    |    time_passes=1
|    |    |    |    jump_threading=0 (28)
|    |    |    |    jump_threading=1 (31)
sccp=1
|    ipsccp=0 (1.5)
|    ipsccp=1 (7.5)
```

- **Speed.** Data miners built for massive scale up
- **Generalizations** over multiple examples
  - Not brittle point solutions
- **Human readable** models (if you use the right learner)
  - explanation
  - humans+AI, more **trust**

Advantages of DUO: More trust than standard methods

- One rule for forensic accounting
  - Fraud is more likely if no one can audit a process

- Standard optimiers: 100 to 1000 individuals, 100 to 1000 generations



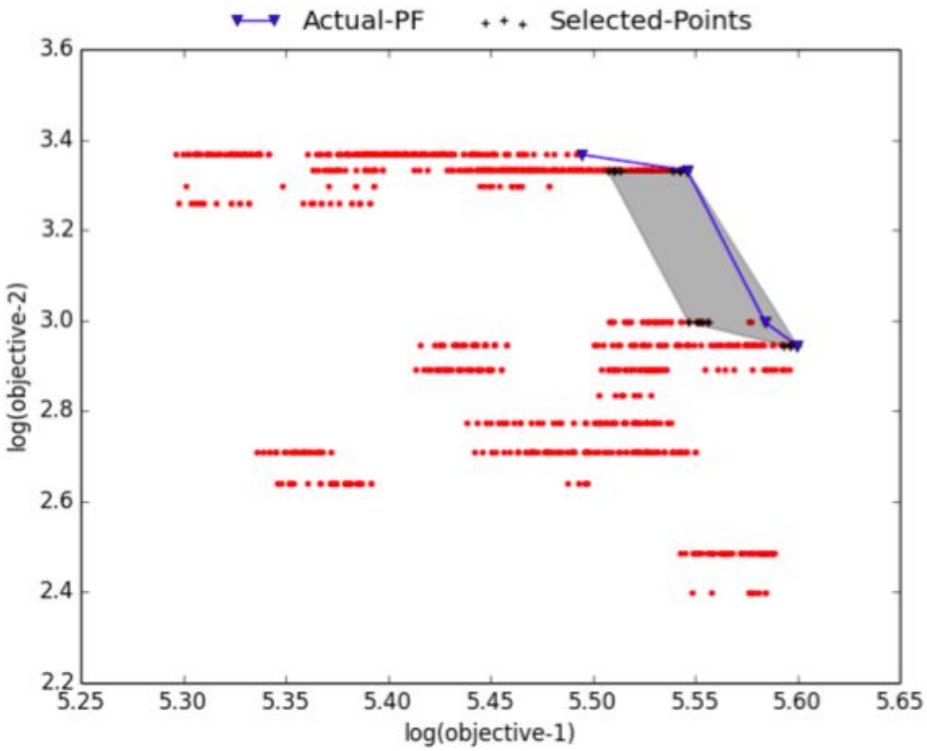
Advantages of DUO: Human-level summaries of complex AI

DUO: < 100 individuals evaluated.

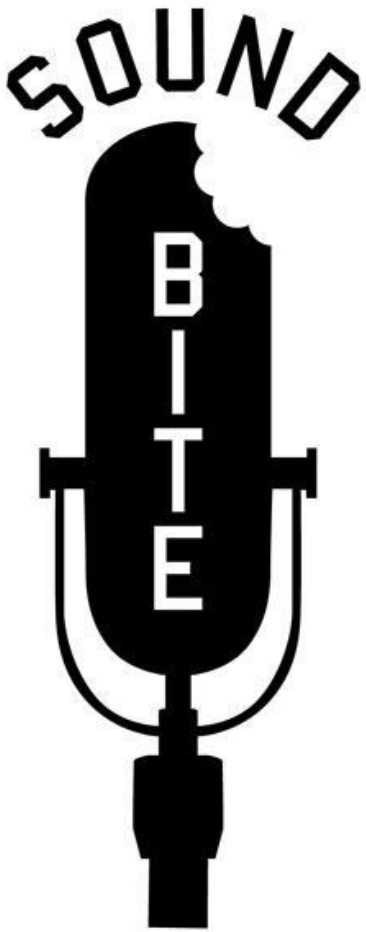
Software configuration, Options explored

Models built from these small samples are readable, even get debug hints

```
sccp=0
|    print_used_types=0
|    |    ipsccp=0 (6.5)
|    |    ipsccp=1
|    |    |    x[10]=0 (12)
|    |    |    x[10]=1 (19)
|    print_used_types=1
|    |    ipsccp=0 (14)
|    |    ipsccp=1
|    |    |    time_passes=0  (24)
|    |    |    time_passes=1
|    |    |    |    jump_threading=0 (28)
|    |    |    |    jump_threading=1 (31)
sccp=1
|    ipsccp=0 (1.5)
|    ipsccp=1 (7.5)
```



Conclusion



- Programmers > Programming
- Software engineering = the search for feedback
- Engineering AI systems: the next challenge
- D.U.O. = data miners using/used-by optimizers

AI: the new engineering challenge

So many new building blocks for new kinds of software
e.g. this talk: DUO= data miners using/used by optimizers

Some can be auto-configured; some need human ingenuity

Domain	Problem
Project management	Effort estimation, dramatic reductions in errors
Requirements	Faster navigation of complex stakeholder conflicts, assessing, software architecture decisions (faster)
Design	Extraction of products from very large product lines
Security	Configuring intrusion detection
Software quality	Detect prediction, test case generation
Software configuration	10^5 options explored using < 100 samples
Text mining and topic modeling: StackOverflow	Improved classification performance using stabler topics.
Text mining and topic modeling: Defect Reports	Stabler topics using LDADE
Text mining and topic modeling: Software Artifact Labeling	Using active learner to label commit data

AI = a large buffet of options. Successful selection, combination of AI tools still requires extensive human expertise

- Many options for combining humans + AI in SE
- Much room for (human) creativity
- e.g. DUO = data miners using/used by optimizers

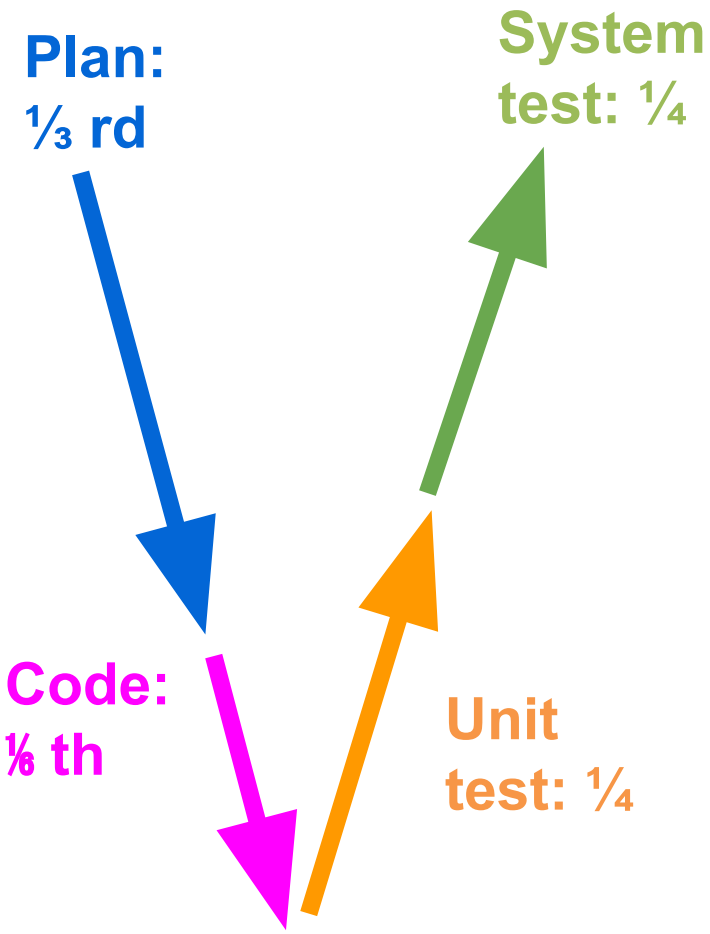
AI = a large buffet of options. Successful selection, combination of AI tools still requires extensive human expertise

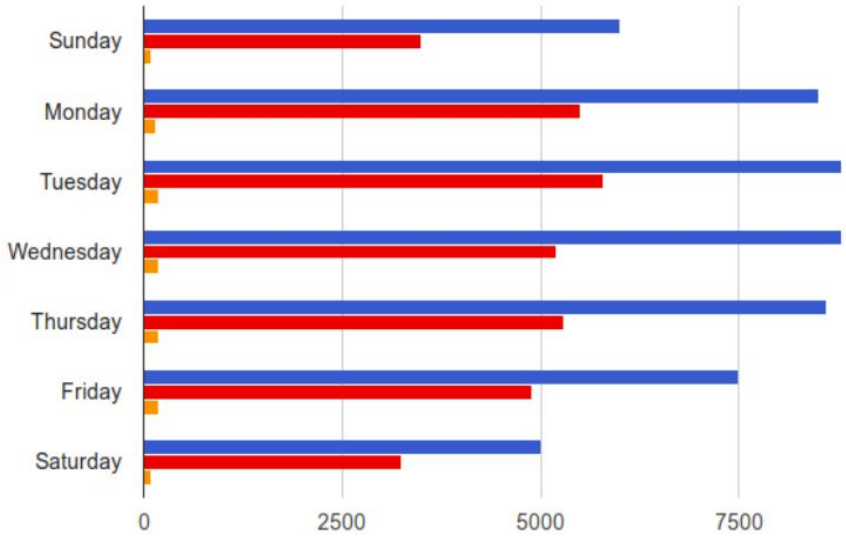
- Many options for combining humans + AI in SE
- Much room for (human) creative
- Much has been done
- Much remains to do
- Care to join the fun?

... if ~~engineering~~ software, then NC State ...

Back-up Slides

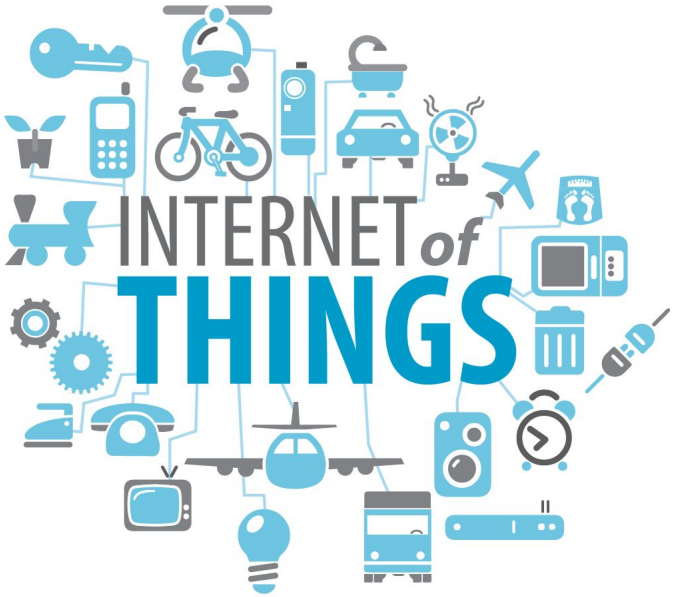
Context: What is SE?
A1: more than programming
A2: negotiation on what2use, what2do next





In the 21st century Many More Models Matter

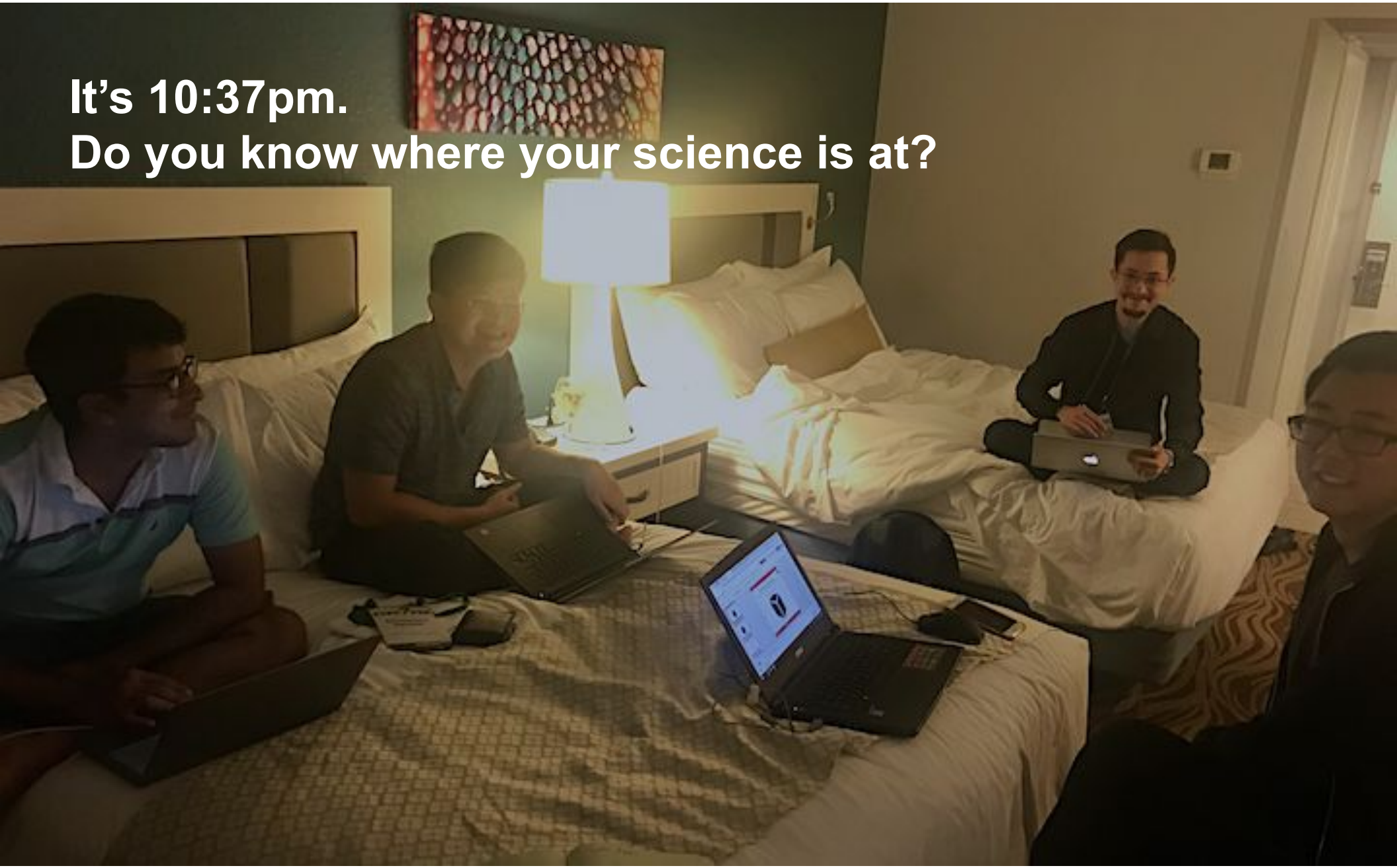
- Software, **matters**
  - In London or New York,
    - time for ambulance to reach patient controlled by models
  - If you cross the border Arizona to Mexico,
    - Models determine if you are taken away for extra security measures
  - If you default on a car loans
    - Models determine when (or if) someone repossesses your car
  - Autonomous cars



- Software **models** of "X":
  - can execute to find options that matter
  - faster than real time

search-based SE

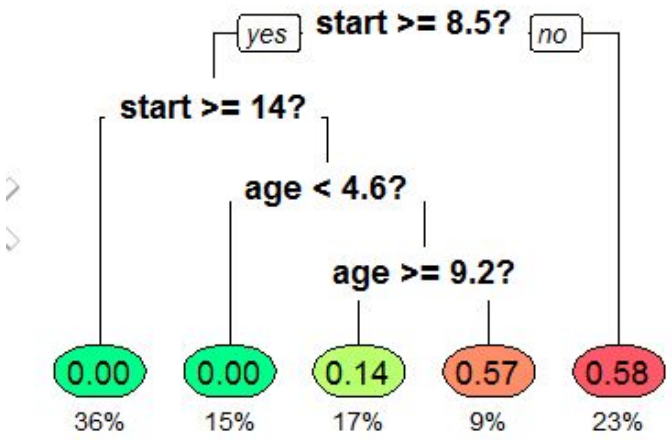
- **Many** models matter:
  - **More** and **more**, SE will be model-based
  - Using, building, maintaining these models that matter
  - Worthwhile to explore model-based reasoning

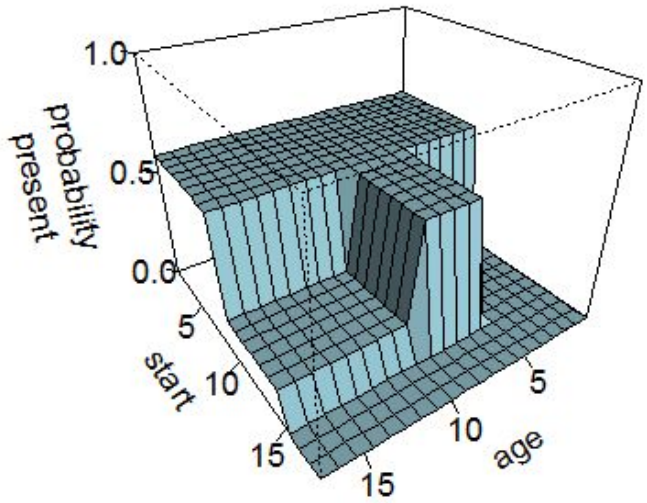

It’s 10:37pm.
Do you know where your science is at?

BTW, how to optimize over discrete and continuous attributes

Base optimizer on learners that know how to handle continuous/discrete values

E.g. CART decision tree learner

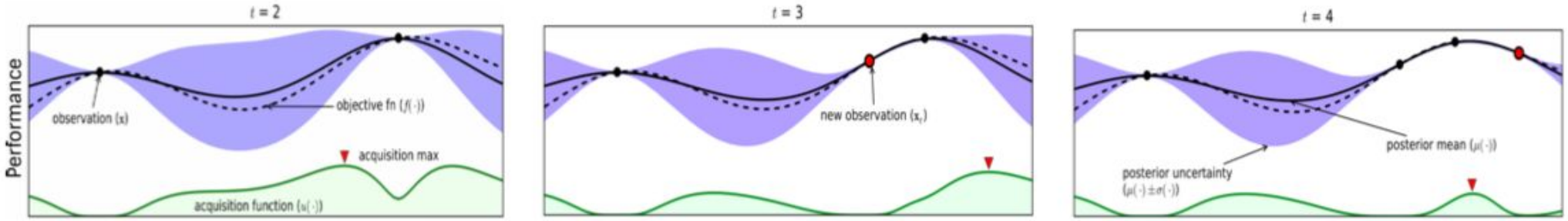




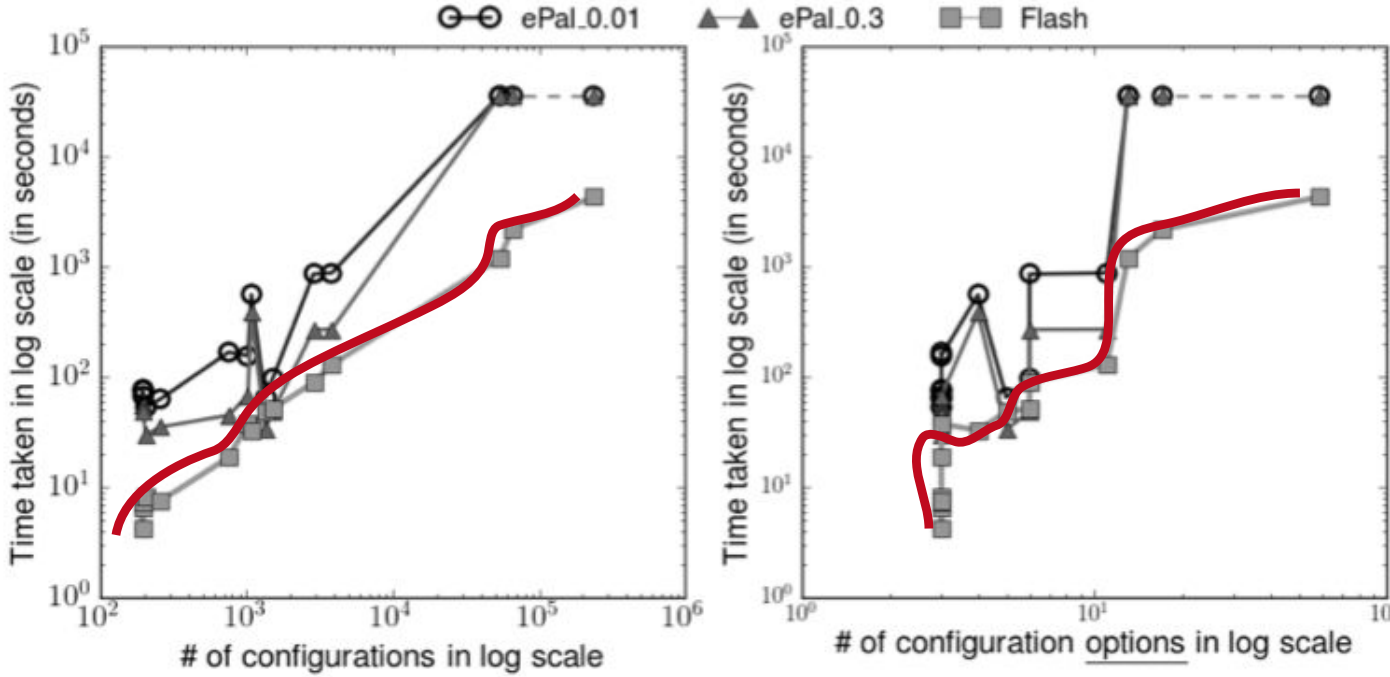
(train) = 20 random examples

1. Eval everything in train
2. Built one CART tree per goal
3. Use trees to guess values for 1000 random examples
    a. Find the “interesting” examples per goal (e.g. max value)
    b. Eval “interesting”, add to train
    c. If most “interesting”s are dull, then break.
    d. Else go to 2

"CART" is now "bayesian parameter optimization" (but it scales, and you can read the models)



Software configuration, Options explored



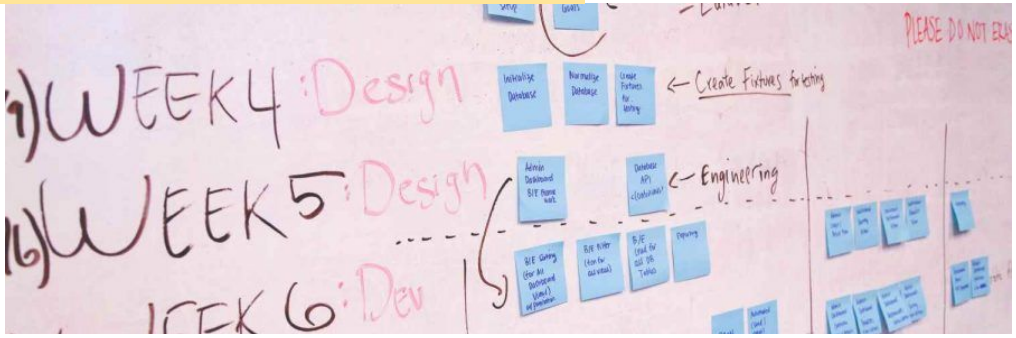
Observations

- Standard data miners are not enough
  - Not clever enough for all nuanced goals pursued in SE
- Standard optimizers are not enough
  - Often, poor on explanation, generalization
  - Also, often very CPU expensive
- But together… much better than apart

DUO = Data miners Used-by/using Optimisers

Case study: software configuration Models getting too complicated

Design, extraction of products



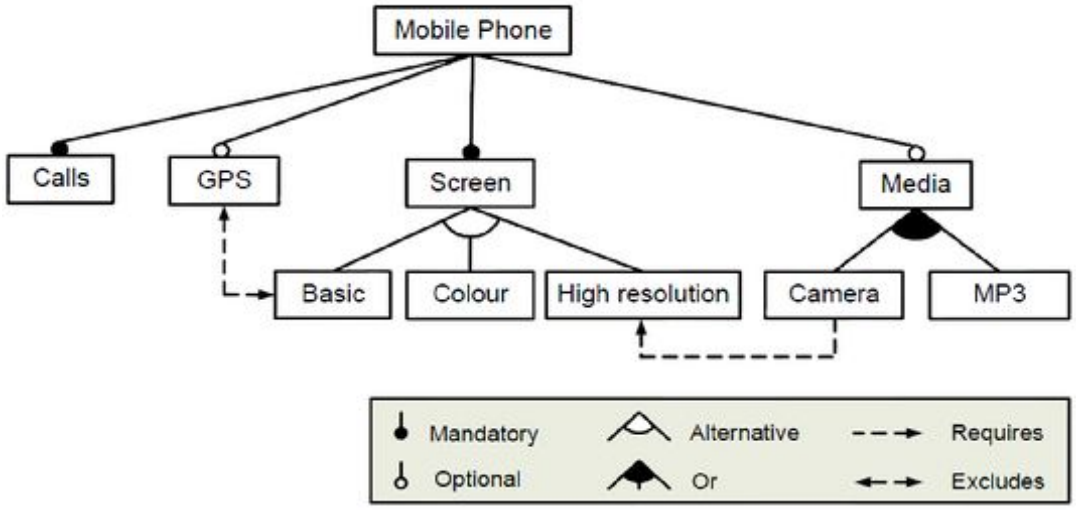


Figure 1: A sample feature model

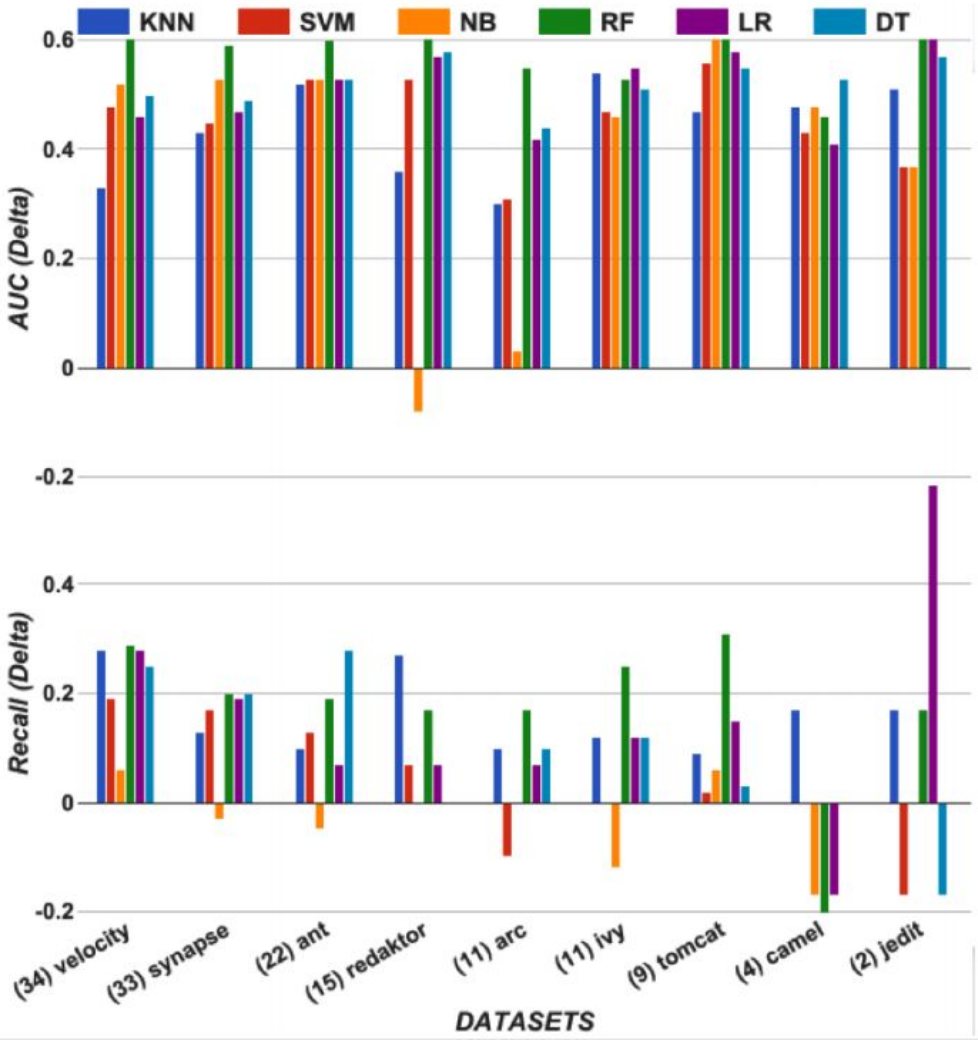
¬Mobile Phone ∨ Calls
Mobile Phone ∨ ¬Calls
¬Mobile Phone ∨ Screen
Mobile Phone ∨ ¬Screen
Mobile Phone ∨ ¬GPS
Mobile Phone ∨ ¬Media
Media ∨ ¬Camera
Media ∨ ¬MP3
¬Media ∨ Camera ∨ MP3
Screen ∨ ¬Basic
Screen ∨ ¬Color
Screen ∨ ¬High resolution
¬Screen ∨ Basic ∨ Color ∨ High resolution
¬Basic ∨ ¬Color ∨ ¬High resolution
Basic ∨ ¬Color ∨ ¬High resolution
¬Basic ∨ Color ∨ ¬High resolution
¬Basic ∨ ¬Color ∨ High resolution
¬GPS ∨ ¬Basic
¬Camera ∨ High resolution

BTW: Linux kernel:
- 7000 terms
- 350,000 constraints

No excuse not to optimize hyperparameters

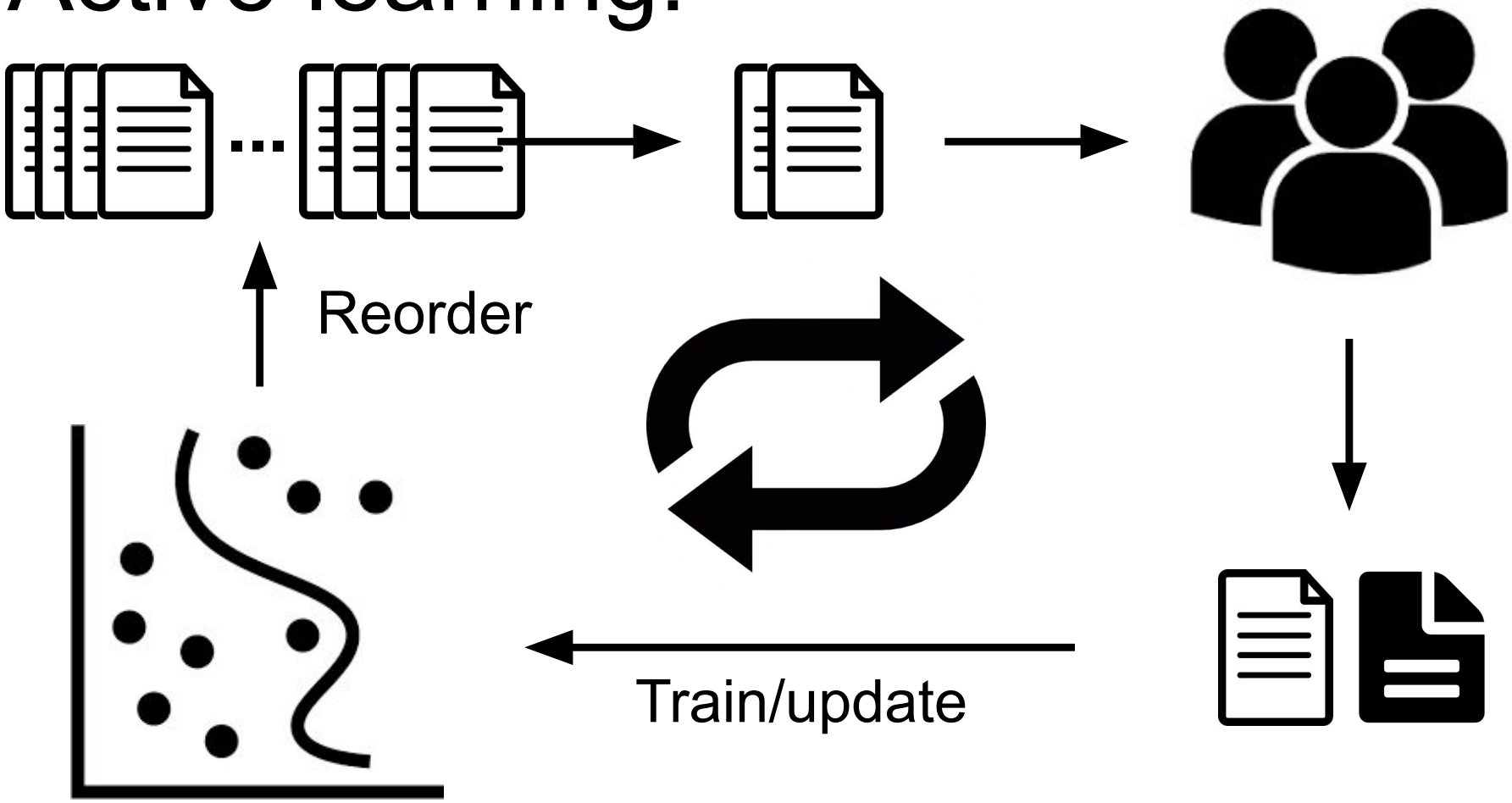
- Large improvements observed with hyperparameter optimization
  - e.g. DE on SMOTE:
- Tricks like DE, BPO with CART are fast,
  - use so few evaluations
  - < 100

Software quality, defect prediction

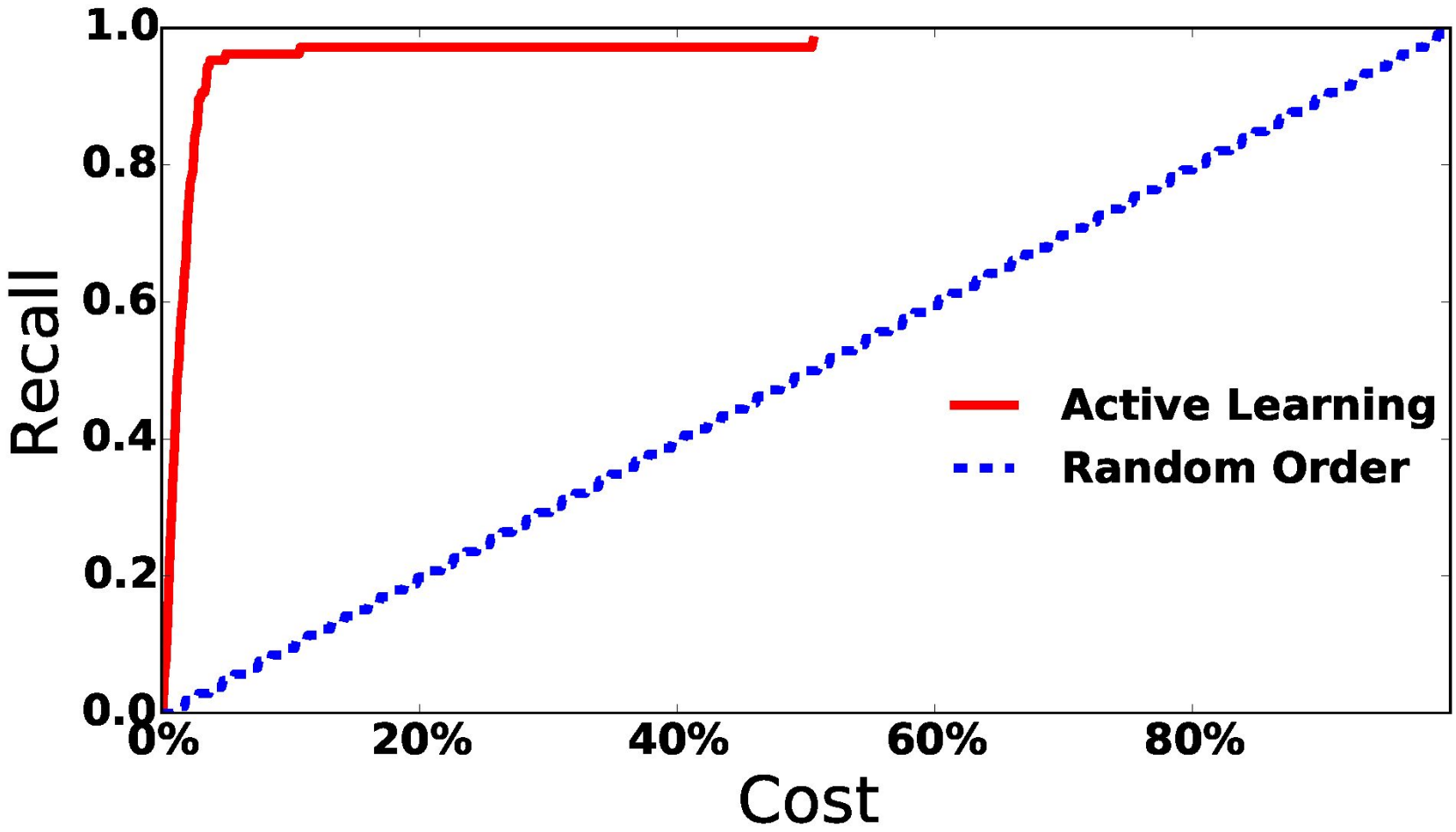


Humans rescuing Broken AI
Incremental support vector machines

Active learning:



Humans rescuing Broken AI
Incremental support vector machines



Humans rescuing Broken AI
Incremental support vector machines

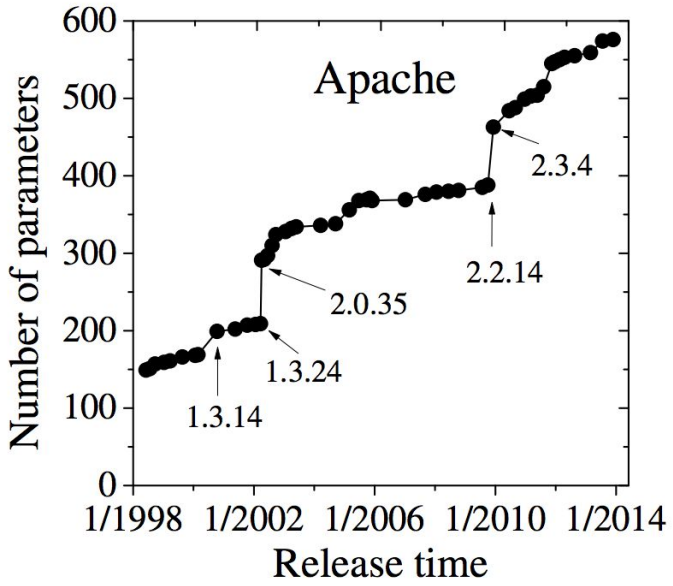
- 50,000 commits, 8 Ph.D. students, 8 hours
- Reviewing decisions “buggy/not-buggy” commits
- Labor intensive
- Faster, using an incremental SVM to learn the “inspector”.
- Very effective (compared to traditional “keyword” method)

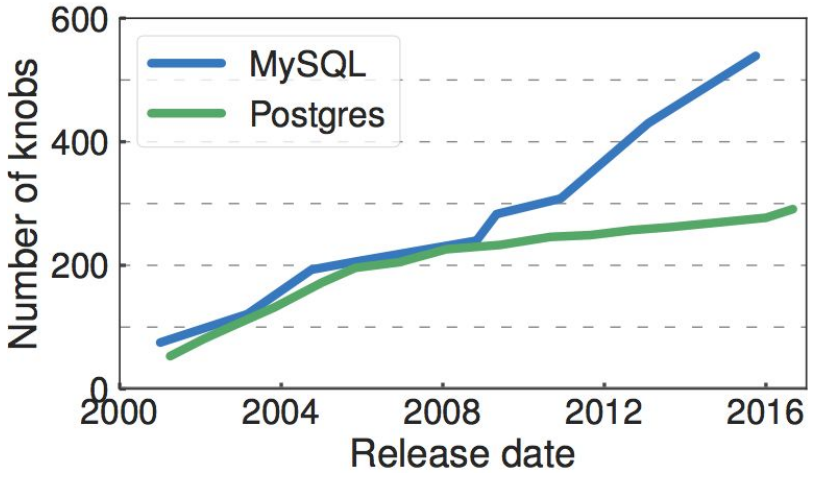
Software quality, defect prediction

Dataset	Metrics			
	Precision		F-Measure	
	keywords	fastread	keywords	fastread
MDANALYSIS	51.43%	41.44%	31.28%	57.60%
ABINIT	58.94%	72.63%	71.20%	79.51%
LIBMESH	43%	49.89%	59.12%	64.20%
LAMMPS	13.38%	23.85%	23.28%	34.27%

Too many, poorly made, choices

- Necessary
- Default is bad
- Expensive





- Default **MySQL** configuration in 2016 assumes that machine **has only 160 MB of RAM**
- Rule-of-thumb settings for WordCount (in **Hadoop**) gave one of its **worst execution times**

- Evaluation of single instance of software/hardware co-design problem can take **weeks**
- Rolling Sort use-case required 21 days, within a total experimental time of about **2.5 months**
- Test suite generation using Evolutionary Algorithm can take **weeks**
- Image recognition workload and speech recognition workload, jobs ran for **many hours or days**

Search-based SE

1. Requirements	Menzies, Feather, Bagnall, Mansouri, Zhang
2. Transformation	Cooper, Ryan, Schielke, Subramanian, Fatiregun, Williams
3.Effort prediction	Aguilar-Ruiz, Burgess, Dolado, Lefley, Shepperd
4. Management	Alba, Antoniol, Chicano, Di Pentam Greer, Ruhe
5. Heap allocation	Cohen, Kooi, Srisa-an
6. Regression test	Li, Yoo, Elbaum, Rothermel, Walcott, Soffa, Kampfhamer
7. SOA	Canfora, Di Penta, Esposito, Villani
8. Refactoring	Antoniol, Briand, Cinneide, O'Keeffe, Merlo, Seng, Tratt
9. Test Generation	Alba, Binkley, Bottaci, Briand, Chicano, Clark, Cohen, Gutjahr, Harrold, Holcombe, Jones, Korel, Pargass, Reformat, Roper, McMinn, Michael, Sthamer, Tracy, Tonella,Xanthakis, Xiao, Wegener, Wilkins
10. Maintenance	Antoniol, Lutz, Di Penta, Madhavi, Mancoridis, Mitchell, Swift
11. Model checking	Alba, Chicano, Godefroid
12. Probing	Cohen, Elbaum
13. UIOs	Derderian, Guo, Hierons
14. Comprehension	Gold, Li, Mahdavi
15. Protocols	Alba, Clark, Jacob, Troya
16. Component sel	Baker, Skaliotis, Steinhofel, Yoo
17. Agent Oriented	Haas, Peysakhov, Sinclair, Shami, Mancoridis