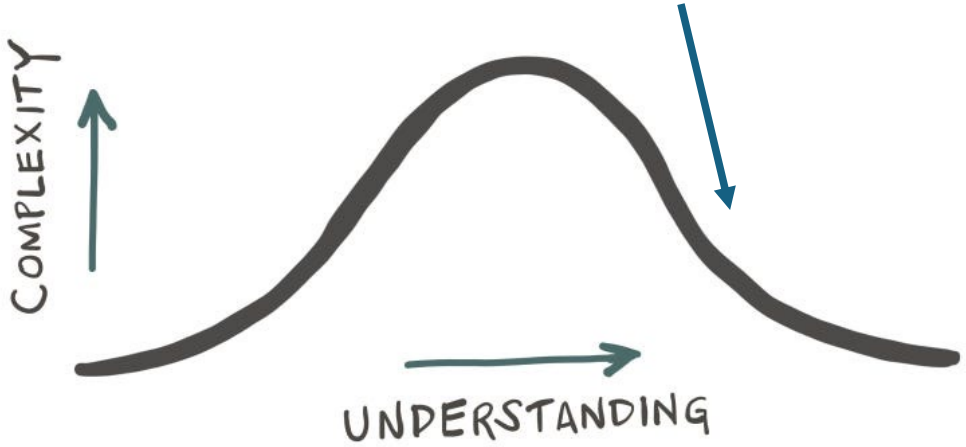
“as a system evolves, its complexity increases unless work is done to reduce it” — Lehman, 1980 **[1]**

if anyone can do it for $1 … can you do it for a cent?

easier AI?



timm@ieee.org

ssbse’26: july 6, 2026

code: ***pip install ezr***
data: ***http://tiny.cc/moot***

Ssbse'16 [2]

An (Accidental) Exploration of Alternatives to Evolutionary Algorithms for SBSE

Vivek Nair(✉), Tim Menzies, and Jianfeng Chen

North Carolina State University, Raleigh, USA
vivekaxl@gmail.com

Abstract. SBSE researchers often use an evolutionary algorithm to solve various software engineering problems. This paper explores an alternate approach of sampling. This approach is called SWAY (Samplying WAY) and finds the (near) optimal solutions to the problem by (i) creating a larger initial population and (ii) intelligently *sampling* the solution space to find the best subspace. Unlike evolutionary algorithms, SWAY does not use mutation or cross-over or multi-generational reasoning to find interesting subspaces but relies on the underlying dimensions of the solution space. Experiments with Software Engineering (SE) models shows that SWAY's performance improvement is competitive with standard MOEAs while, terminating over an order of magnitude faster.

Keywords: Search-based SE · Sampling · Evolutionary algorithms

- 2019:
  Amritanshu Agrawal (Snap);
  Vivek Nair (Meta)
  Jianfeng Chen (Meta)
- 2025:
  Andre Lustosa (Redhat)
- 2027:
  Kishan Ganguly BINGO (FSE'26);
  Amirali Rayegon Explanation (JSS'26)
- 2028:
  Srinath Srinivasan, Neurosymbolic (arxiv)

why easier AI?

how to do easier AI?

why not more of easier AI?

An **ASE'26** workshop
1 page abstract due **July 20**

why easier AI?

how to do easier AI?

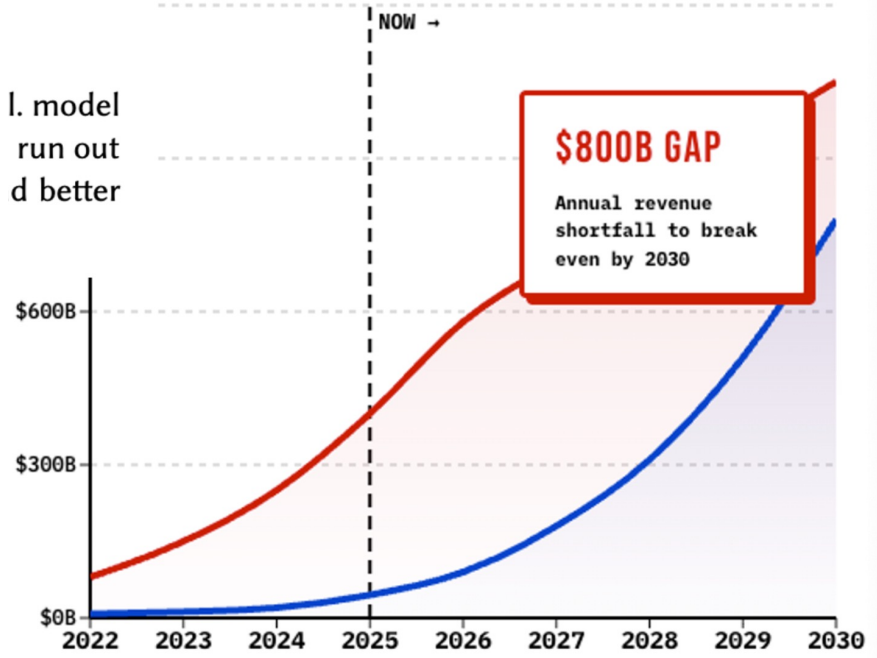
why not more of easier AI?

Q: why easier AI?
A: economics [3] [4]

- Niklaus Wirth:
  - “Software gets **slower** faster than hardware gets faster”

- Even if $/token falls, Jevons paradox: total token burn grows (*10, *100: 2023→2026)

- Bain 2025 forecast:
  - $800B/yr revenue gap by 2030
  - 2026 capex ~$725B; 2027 projected >$1T — gap widening



money is one worry … trust is another →

Q: why easier AI?
A: easier to trust; easier to watch and understand

My Tesla Was Driving Itself Perfectly —Until It Crashed

The danger of almost-perfect tech

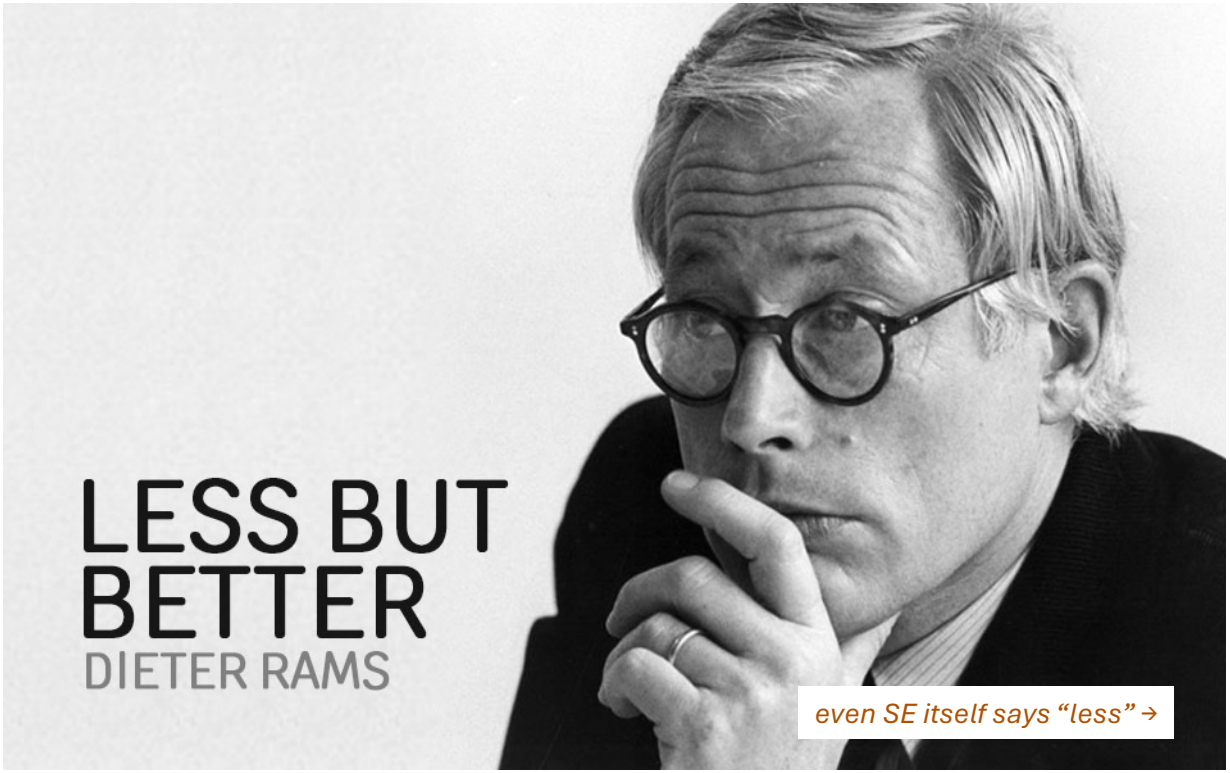
By Raffi Krikorian

THE **ATLANTIC**
MARCH 17, 2026

and some domains have no choice →

Q: why easier AI?
A: cause some domains demand it [5]

- Green AI
- global **south** (countries without e.g. American-scale infrastructure)
- resource-constraint teaching
  - community colleges,
  - K–12 CS,
  - MSIs,
  - thise using **personal** hardware
- privacy-constrained industrial dev
  - **defense**,
  - regulated healthcare,
  - financial services,
  - law firms)
- air-gapped settings;
- **edge** or on-device inference



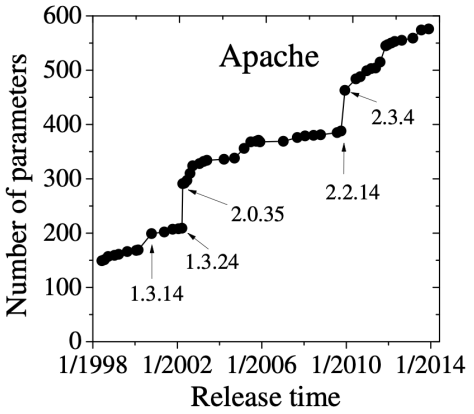
Q: why easier AI?
A: software engineering

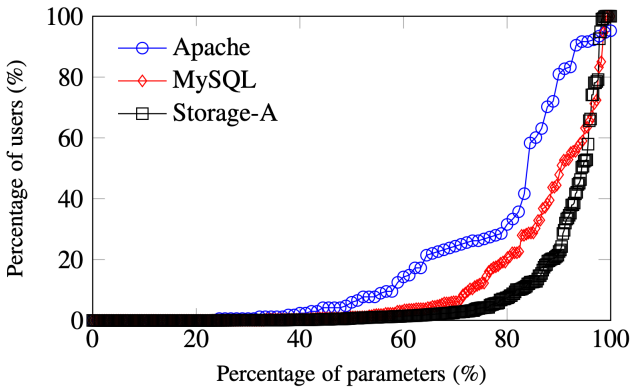
- **security**:
  - modern software = assembly of pre-made parts (> 80%)
  - more reuse == more exposure
  - e.g. left-pad; MOVEit ($12B),
    Falcon ($5B), ByBit (1.4B) $24B (2025);

- **survivability**: e.g. SQLite does not use Berkeley DB.
  - Now used on Mars

- **maintainability + reliability:**
  - more loc = more to maintain, more to go wrong

- **waste**: Most users cannot handle most options **[6]**





and the data for "big" is running out →

Q: why easier AI?
A: cause “big AI” needs too much data [7,10]

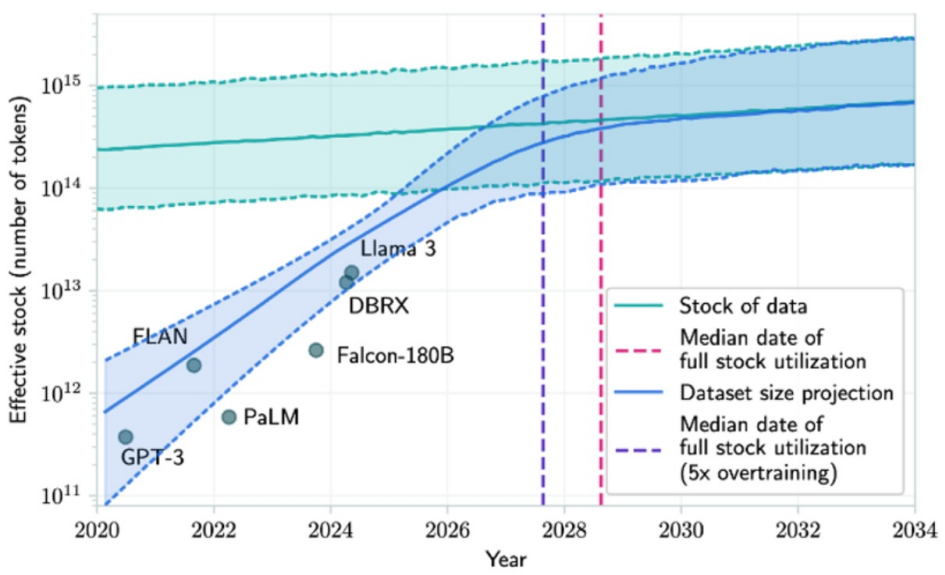


Fig. 4. Median results from the Villalobos et al. model (shown in green) estimate that by 2028, we will run out of new textual data needed to train bigger and better LLMs [75].

besides, more was never always better →

Q: why easier AI?
A: more AI not always better [8][9][10]

- text mining predicting effort within **agile** projects,
  - (SVM and TFIDF) better predictions than deep learners
  - Runs 100 times faster

- tree-based models (Random Forest, XGBoost)
  - consistently **outperform** deep neural networks on tabular datasets,
  - especially those with more than 10,000 rows

- Etc [11-14]

so: how? →

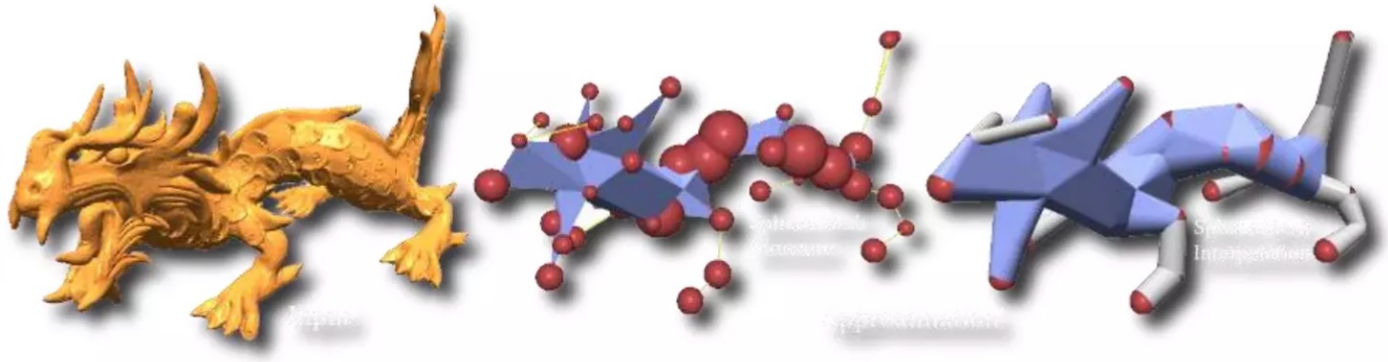
why easier AI?

how to do easier AI?

why not more of easier AI?

Q: how to do easier AI?
A: so many ways [15-28]

- 1300: William of Ockham: “entities must not be multiplied beyond necessity"
- 1902, **PCA**: reduce data to a few principal components
- 1974, **Prototypes**: speed up k-NN by reducing rows to a few exemplars
- 2009, **Active learning**: only use most informative rows
- 2020s, Distillation: compress large LLM models with little performance loss
- etc



of all these, focus on active learning →

Q: how to do easier AI (with active learning)
A: use current model to decide what's next [29]

- E.g. **SMAC3**
  - ensemble of decision trees
  - Seek unlabeled examples where
    - ensemble most disagrees (a.k.a. explore)
    - **ensemble** most agree (a.k.a. exploit)
    - adapt = Slowly shift explore to exploit
- Cost of SMAC3:
  - when new data arrives
  - **rebuilding** the ensemble

- EZR: **pip install ezr**
- "ensemble" = 2 **class** Bayes classifier
  - $|Best|,\ |Rest| = \sqrt{N},\quad N - \sqrt{N},$
  - **next** =
    - closest to *Best*
    - furthest from *Rest*
  - update:
    - add next to *Best*,
    - resort *Best*,
    - **pop** worst *Best* to *Rest*
  - Very fast
    - never sort *Rest*
    - Updates to *Best* and *Rest* are fast.

does it work? first: data →

Q: how to test EZR?
A: read lots of papers, collect their data [30]

- Recent SBSE **papers**: ICSE, FSE, TSE, IST, EMSE, TOSEM, ASE
- 127 data sets
- 3-1044 independent **x attributes**;
- 1 to 8 dependent **y attributes** to be minimized or maximized;
- 93 to 100,000 rows.
- http://**tiny.cc/moot**

#	Category	Focus	File Name(s)	Here, optimizers struggle with issues like...	#y	#x	Rows	Refs
Software Systems & Configuration & Tuning								
24	Soft. Config	**PLE**	SS-[A-X]	Minimize footprint/memory vs. maximizing throughput	2-3	3–88	197–86k	[1]
12	PromiseTune	**Perf.**	7z, LLVM, BDBC	Minimize execution time and energy consumption	1	6–35	864–166k	[2]
1	Cloud compute	**Tuning**	Apache_AllMeas	Balance server response vs. CPU/RAM load patterns	1	9	192	[3]
1	Cloud compute	**Tuning**	SQL_AllMeas	Minimize latency/IO vs. maximizing throughput	1	39	4,654	[4]
1	Cloud compute	**Tuning**	X264_AllMeas	Optimize encoding parameters for PSNR/SSIM quality	1	16	1,153	[1]
2	Testing	**Testing**	test120, 600	Maximize coverage while minimizing execution time	1	9	5,161	–
Project Management & Process Modeling								
35	Proj. Health	**Health**	Health-Closed	Optimize PR rates and minimize developer churn	2-3	5	10,001	[5]
3	Scrum	**Agile**	Scrum[1k-100k]	Maximize velocity within sprint constraints	3	124	1k–100k	[6]
8	Feature Mod.	**Config**	FFM, FM-*	Optimize Clause/Constraint ratio in large spaces	3	128–1k	10,001	[1]
1	nasa93dem	**Cost**	nasa93dem	Minimize effort (person-months) vs. quality	3	26	93	[5]
1	COC1000	**Risk**	coc1000	Minimize risks/schedule slip vs. analyst expertise	5	20	1,001	[7]
4	POM3	**Process**	pom3[a-d]	Balance project idle rates vs. completion costs	3	9	501–20k	[6]
4	XOMO	**Defects**	xomo_[flt,grd]	Minimize defect density vs. total project effort	4	27	10,001	[7]
Interdisciplinary & Behavioral Benchmarks								
4	Behavioral	**Behavior**	all_players, etc	Maximize user retention and predict cohort churn	1-3	26–55	82–17k	[8]
4	Financial	**Finance**	BankChurners	Minimize credit risk vs. optimal pricing models	2-5	19–77	1k–20k	[9]
3	Health	**Health**	COVID19, Life	Maximize accuracy/life vs. readmission likelihood	1-3	20–64	2k–25k	[10]
2	RL	**Control**	A2C_[Acr,Crt]	Maximize reward signals vs. steps to convergence	3-4	9–11	224–318	–
5	Sales	**Market**	Marketing, socks	Maximize ROI vs. minimizing forecasting error	1-8	20–31	247–2k	[11]
3	Misc.	**General**	auto93, Car	Optimize multivariate trade-offs (e.g. MPG vs. HP)	2-5	5–38	205–1k	[1]

configuration, performance tuning, product lines, project health, defect prediction, testing, cost estimation, cross-domain generalization, and text mining

... and a score →

Q: how to score EZR?
A: win = 1 - normalize(regret) [31]

- Distance to **heaven** (aggregate n goals to one)

$$d2h(y) = \sqrt{\sum_{i=1}^{k} y_i^2} \,/\, \sqrt{k},$$

- Regret = (**d2h** – lo) / (mu – lo)
- Win = int(100 * (1 – regret))
  - Win ≤ 0 ➔ optimizer just finds mu
    - **Failure**
  - Win=100 ➔ found reference optimum
    - **Success**
- Build regression tree on **labeled** egs
  - Splitting rows to minimize σ of win
  - Reports what makes you good or bad

win	n	Lbs-	Acc+	Mpg+	
8	**43**	**2334.2**	**16.3**	**27.4**	
41	26	2038.7	17.1	30.0	Volume <= 111
71	10	1871.7	18.1	32.0	| Volume <= 83
✅ **85**	**4**	**1831.7**	**19.0**	**32.5**	**| | Model <= 73**
62	6	1898.3	17.4	31.7	| | Model > 73
22	16	2143.0	16.5	28.7	| Volume > 83
36	7	2098.1	18.0	28.6	| | Model <= 73
12	9	2177.9	15.3	28.9	| | Model > 73
26	4	2210.0	15.7	30.0	| | | Volume > 97
0	5	2152.2	15.0	28.0	| | | Volume <= 97
-41	17	2786.1	15.1	23.5	Volume > 111
-9	5	2279.8	14.6	28.0	| Volume <= 116
-54	12	2997.1	15.3	21.7	| Volume > 116
-44	8	2921.1	15.8	22.5	| | Model > 73
-38	4	2563.0	14.0	25.0	| | | Clndrs <= 4
-50	4	3279.2	17.4	20.0	| | | Clndrs > 4
❌ **-75**	**4**	**3149.0**	**14.4**	**20.0**	**| | Model <= 73**

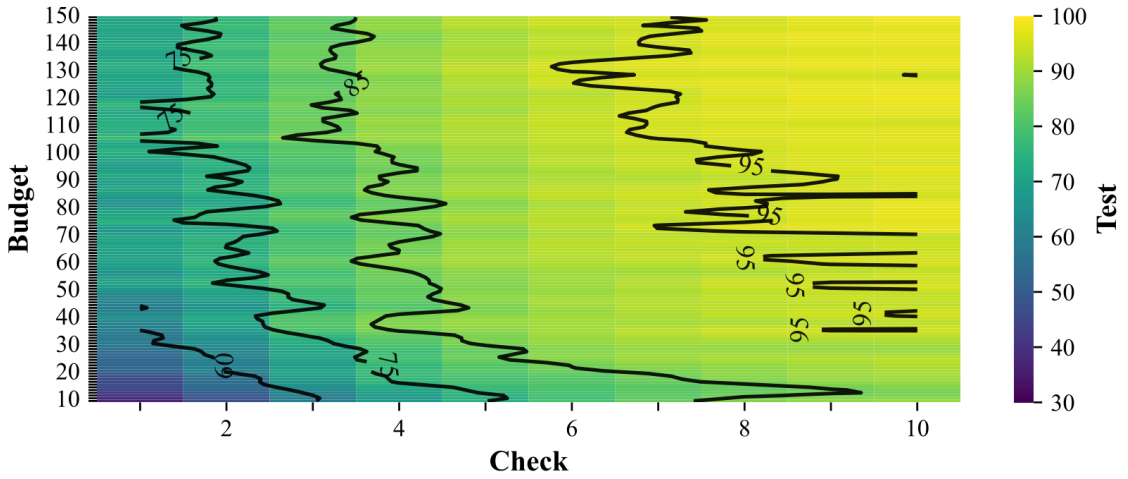
configuration, performance tuning, product lines, project health, defect prediction, testing, cost estimation, cross-domain generalization, and text mining

so how well does it generalize? →

Q: how well does EZR generalize?
A: sort holdout via tree, check first few [31]

Win = int(100 * (1 – regret))

- **100,000** times
  - pick any MOOT data set
  - Train:
    - Active learning, labeling y rows with a **limited** budget.
    - Build tree from labels
  - Test:
    - Use tree to sort holdout
    - **Check** first few
- Active learning, very effective
  - Label 50 rows
  - Check 5
  - **Achieve** 85% of best



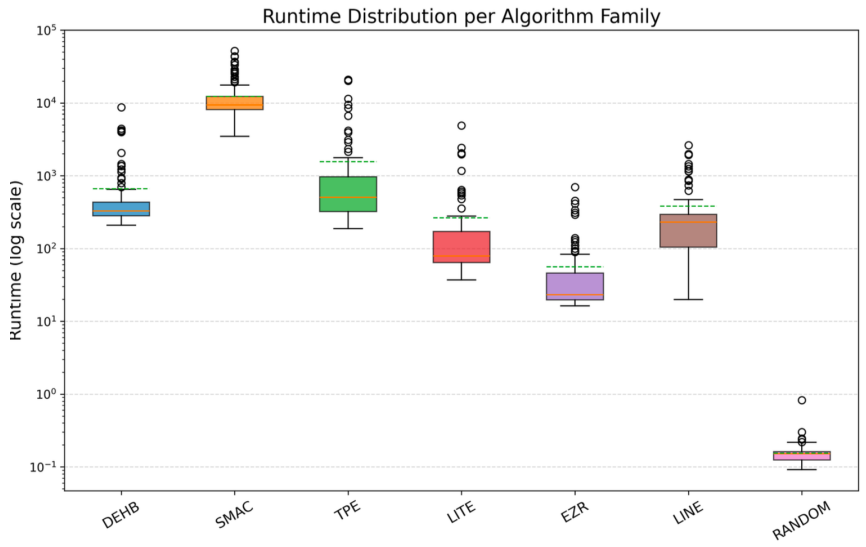
This entire 100,000-experiment study: one laptop, 2 hours, no cluster, no cloud bill.

and how fast? →

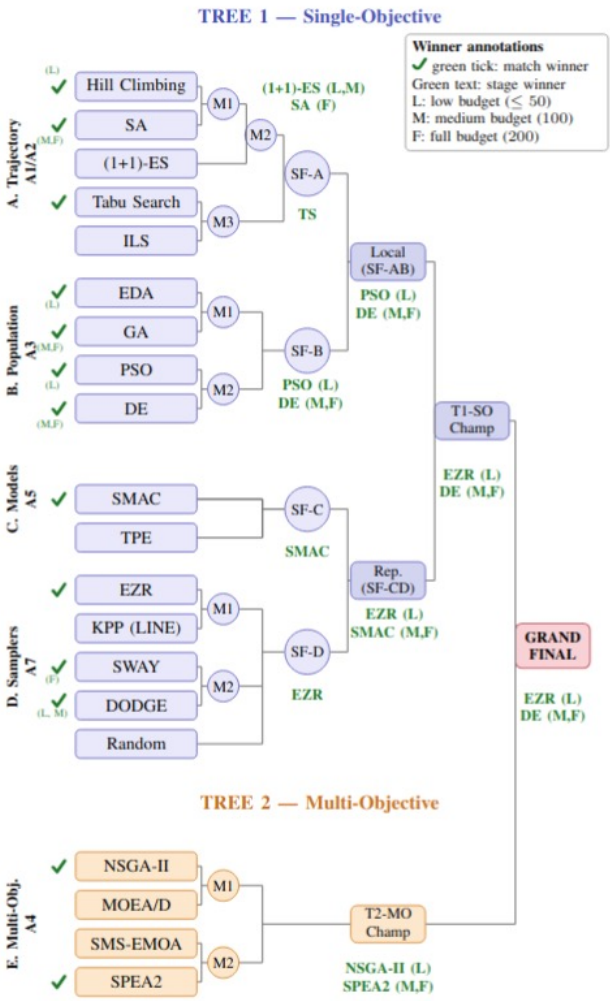
Q: how fast and effective are EZR's trees?

A: very and very [32]

ms



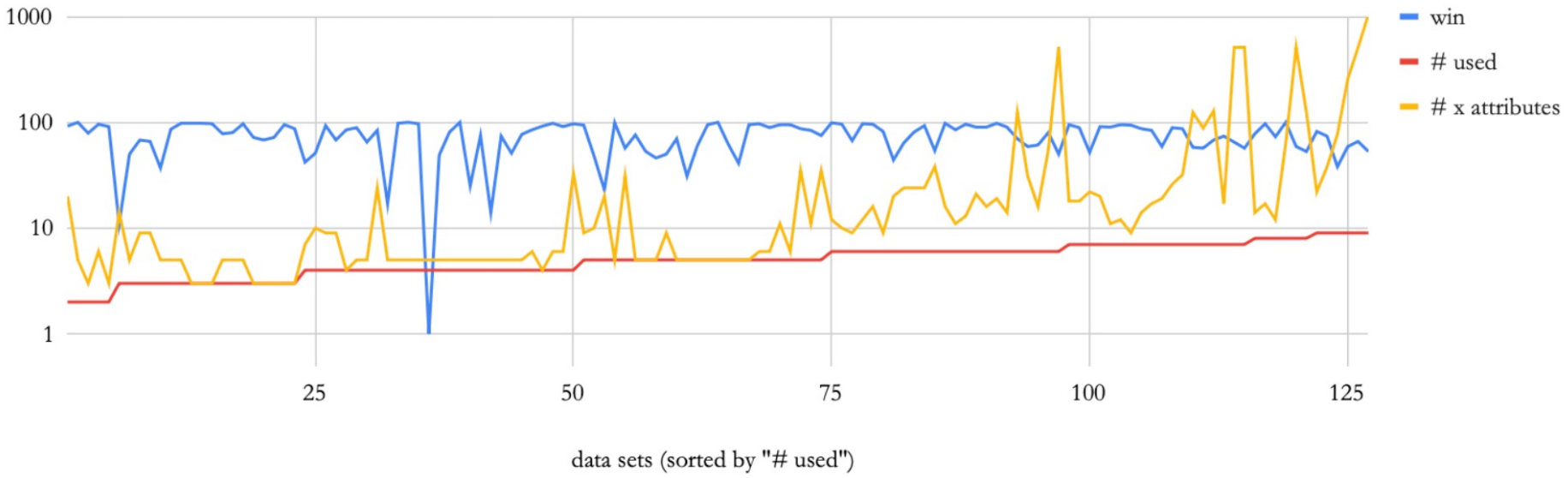
500x less CPU = 500x less energy. every run.



e.g. NSGA-II needs 1000 samples for what EZR reaches in 50

and are the models small enough to explain? →

Q: how big are EZR's models?
A: surprisingly small [33]



and are the models small enough to explain? →

Q: so is EZR useful for explanation?
A: EZR's trees competitive with SOTA [33]

- Build trees using N **attributes**
- N= #attributes used in EZR's tree
- Using other tools: select **top** N attributes

- White = tied best
- Blue = within **top** 90%
- Yellow = within top 75%
- EZR goes a long way
  - E.g. last row
  - For a **3 goal** task
    - EZR uses 7 / 1044 attributes
    - EZR uses 150 rows
    - Arguably competitive

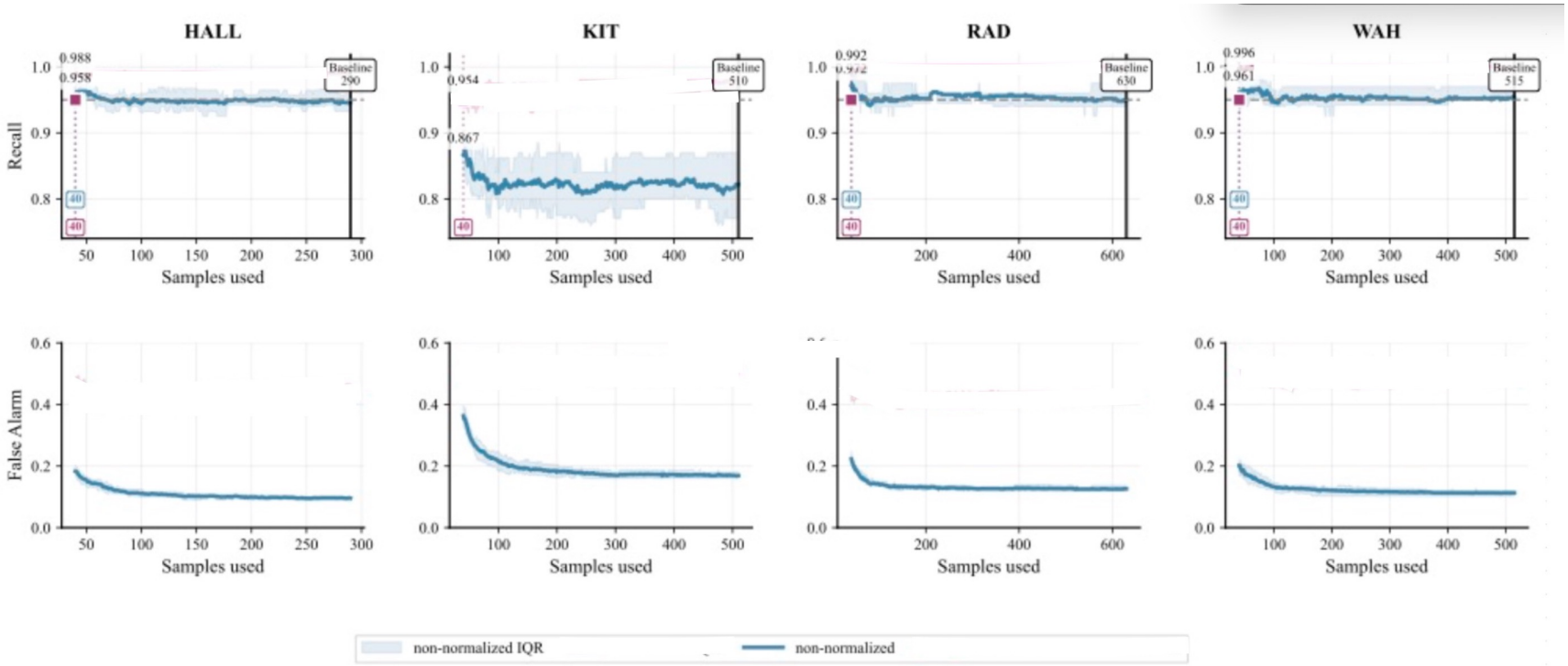
- **across 127 datasets: trees use**
- **4–9 of up to 1044 attributes**

RLF	SHAP	EZR	anova	all	data	x*	x	y	budget	rows
95	100	100	100	100	HSMGP	6	14	1	150	3457
87	81	82	82	85	SS-R	7	14	2	150	3009
94	95	63	57	96	SS-V	4	16	2	150	6841
74	74	59	74	99	SS-W	7	16	2	150	65537
100	100	100	100	100	X264-AM	6	16	1	150	1153
100	100	100	99	100	SS-M	7	17	3	150	865
81	81	81	81	81	SS-N	8	17	2	150	53663
59	62	53	52	58	coc1000	9	20	5	150	1001
95	96	96	93	100	SS-U	7	21	2	150	4609
98	98	93	98	98	xomo_flight	7	27	4	150	10001
92	92	90	90	92	xomo_grnd	7	27	4	150	10001
94	95	95	95	95	xomo_osp	7	27	4	150	10001
87	88	82	82	88	xomo_osp2	8	27	4	150	10001
77	75	71	64	85	SQL-AM	10	39	1	150	4654
87	91	92	74	94	billing10k	7	88	3	150	10001
75	75	81	75	97	Scrum100k	7	124	3	150	100001
82	88	92	88	97	Scrum10k	8	124	3	150	10001
83	89	86	83	89	Scrum1k	7	124	3	150	1001
81	99	93	81	99	FFM-125	9	128	3	150	10001
81	97	89	97	97	FFM-250	8	256	3	150	10001
95	93	93	95	95	FFM-500	8	510	3	150	10001
81	96	87	96	96	FM-500-1	8	511	3	150	10001
86	91	93	91	97	FM-500-2	7	511	3	150	10001
95	95	95	87	95	FM-500-3	9	513	3	150	10001
97	97	93	96	97	FM-500-4	9	517	3	150	10001
82	94	94	71	100	FFM-1000	7	1044	3	150	10001

Any way to use all this to help LLMs? →

Q: what does EZR say about LLMs?
A: LLMs rarely tested against simpler tool [34]

- Search google scholar for **relevant** research



why easier AI?

how to do easier AI?

why not more of easier AI?

Q: if so good, why not more easier AI?
A: (weak) empirical standards [34] [35]

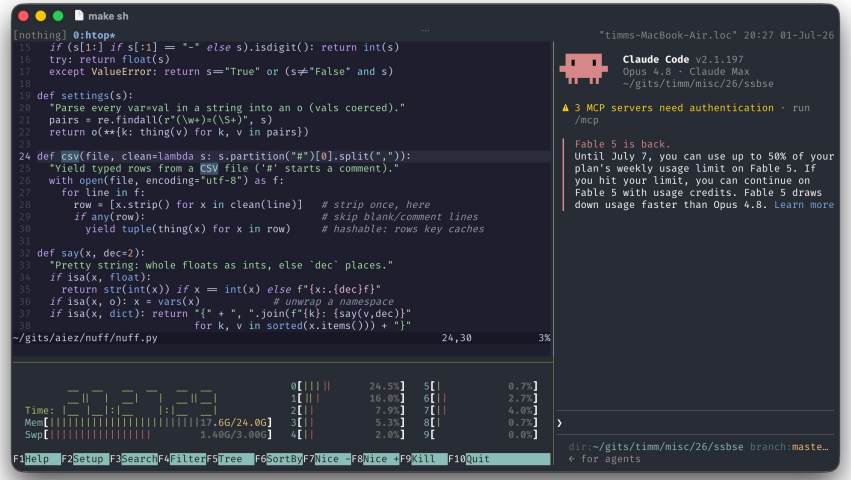
- In a survey of 229 SE papers on **LLMs in SE**
  - Only 5% compared to simpler approaches
- **Methodological** error
  - Other methods can produce results that are better and/or **faster**
    - (See above)
- My career:
  - Step 1: compare ”it” vs a **stupider** straw man
  - Step 2: throw “it” away, use the straw man

also, the new toys are seductive →

Q: if so good, why not more easier AI?
A: cause “big” sells; prompts/loops are cool [36]

- Welcome to my data center
  - ribbon on the door the mayor can cut
  - **such** a spectacle!

- Compare that to **timm inc**.
  - No special software to sell
    - you can build it yourself
  - No special hardware to sell
    - our algorithms are so fast, they don’t **need** it
  - IPO : low evaluation

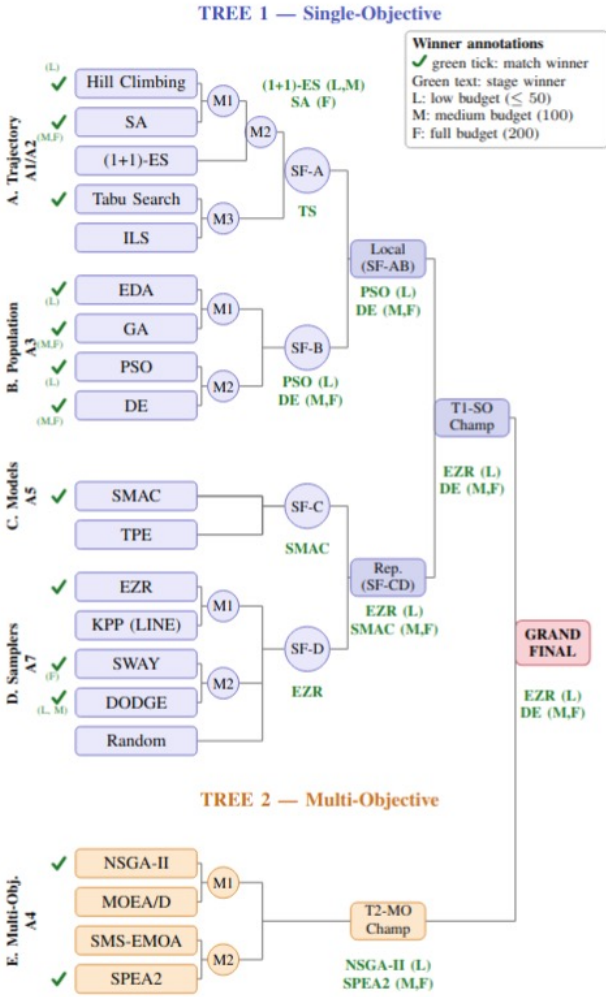


and simplicity is genuinely hard →

Q: if so good, why not easier AI?
A: the simplicity paradox

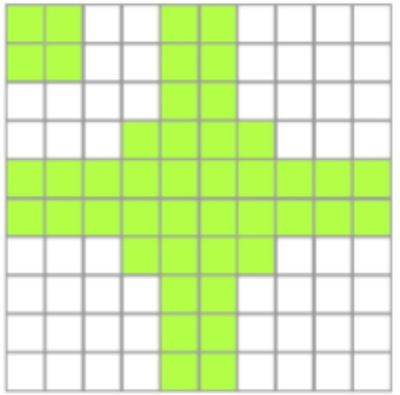
- Simplicity needs **experience**.
  - "Experience is the name every one gives to their **mistakes**."
    — *Lady Windermere's Fan* (1892)

- Simplicity is complicated to certify
  - You have to build and/or **run** simpler "it" and complex
  - Needs lot of CPU; e.g. 20 months of CPU



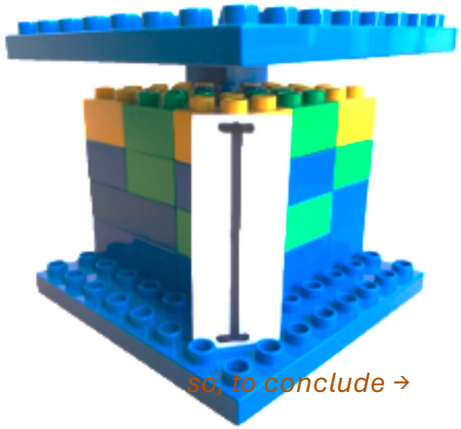
and our brains fight us →

Q: if so good, why not more easier AI?
A: human cognitive bias [37]



- Our simpler methods have not been **previously** explored
  - since other researchers **were not looking for them**.
- Why?

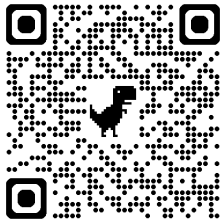
- Humans are biased to bloat
  - In **studies** of 100s of human subjects
    - Humans favor additive to subtractive changes
    - 1,155 to 297 (about 4:1)



so, to conclude →

TL;DR

Lehman's **2nd** law [1]



- "as a system evolves, its complexity **increases** unless work is done to maintain or reduce it"

- Q: why / how can we **reduce** the complexity of AI?
- A: we must / we can

- **do**: baseline the simple:
  pip install ezr + tiny.cc/moot

- **review**:
  - ask "compared to what simpler?"

- **teach**:
  - easier AI first; scale only when simple fails

An **ASE'26** workshop
1 page abstract due **July 20**

/ (question | comment)* /

code: ***pip install ezr***
data: ***http://tiny.cc/moot***

References

[1] Lehman, Meir M. "Programs, Life Cycles, and Laws of Software Evolution." Proceedings of the IEEE, vol. 68, no. 9, 1980, pp. 1060-1076.

[2] Nair V, **Menzies** T., Chen J. An (Accidental) Exploration of Alternatives to Evolutionary Algorithms for SBSE" presented at SSBSE 2016

[3] Wirth, Niklaus. "A Plea for Lean Software." Computer, vol. 28, no. 2, 1995, pp. 64-68.

[4] Bain & Company. Global Technology Report 2025. Bain & Company, Sept. 2025.

[5] Schwartz, Roy, et al. "Green AI." Communications of the ACM, vol. 63, no. 12, 2020, pp. 54-63.

[6] Xu, Tianyin, et al. "Hey, You Have Given Me Too Many Knobs! Understanding and Dealing with Over-Designed Configuration in System Software." Proceedings of the 10th Joint Meeting on Foundations of Software Engineering (FSE), ACM, 2015, pp. 307-319.

[7] Villalobos, Pablo, et al. "Will We Run Out of Data? Limits of LLM Scaling Based on Human-Generated Data." Proceedings of the 41st International Conference on Machine Learning (ICML), 2024.

[8] Tawosi, Vali, Rebecca Moussa, and Federica Sarro. "Agile Effort Estimation: Have We Solved the Problem Yet? Insights from a Replication Study." IEEE Transactions on Software Engineering, vol. 49, no. 4, 2023, pp. 2677-2697.

[9] Majumder, Suvodeep, Tim **Menzies** et al. "500+ Times Faster than Deep Learning: A Case Study Exploring Faster Methods for Text Mining StackOverflow." Proceedings of the 15th International Conference on Mining Software Repositories (MSR), ACM, 2018.

[10] Ling, Xiao, Tim **Menzies** et al. "Trading Off Scalability, Privacy, and Performance in Data Synthesis." IEEE Access, vol. 12, 2024, pp. 26642-26654.

[11] Fu, Wei, and Tim **Menzies**. "Easy over Hard: A Case Study on Deep Learning." Proceedings of the 11th Joint Meeting on Foundations of Software Engineering (ESEC/FSE), ACM, 2017, pp. 49-60.

[12] Fu, Wei, Tim **Menzies**, and Xipeng Shen. "Tuning for Software Analytics: Is It Really Necessary?" Information and Software Technology, vol. 76, 2016, pp. 135-146.

[13] Somvanshi, Shriyank, et al. "A Survey on Deep Tabular Learning." arXiv preprint arXiv:2410.12034, 2024.

References (cont.)

[14] Grinsztajn, Leo, Edouard Oyallon, and Gael Varoquaux. "Why Do Tree-Based Models Still Outperform Deep Learning on Typical Tabular Data?" Proceedings of NeurIPS Datasets and Benchmarks Track, 2022.

[15] Pearson, Karl. "On Lines and Planes of Closest Fit to Systems of Points in Space." The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, vol. 2, no. 11, 1901, pp. 559-572.

[16] Amarel, Saul. "Program Synthesis as a Theory Formation Task: Problem Representations and Solution Methods." Machine Learning: An Artificial Intelligence Approach, Morgan Kaufmann, 1986.

[17] Chang, Chin-Liang. "Finding Prototypes for Nearest Neighbor Classifiers." IEEE Transactions on Computers, vol. C-23, no. 11, 1974, pp. 1179-1184.

[18] Johnson, William B., and Joram Lindenstrauss. "Extensions of Lipschitz Mappings into a Hilbert Space." Contemporary Mathematics, vol. 26, 1984, pp. 189-206.

[19] De Kleer, Johan. "An Assumption-Based TMS." Artificial Intelligence, vol. 28, no. 2, 1986, pp. 127-162.

[20] Crawford, James M., and Andrew B. Baker. "Experimental Results on the Application of Satisfiability Algorithms to Scheduling Problems." Proceedings of the 12th National Conference on Artificial Intelligence (AAAI), 1994, pp. 1092-1097.

[21] Olshausen, Bruno A., and David J. Field. "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images." Nature, vol. 381, 1996, pp. 607-609.

[22] Kohavi, Ron, and George H. John. "Wrappers for Feature Subset Selection." Artificial Intelligence, vol. 97, no. 1-2, 1997, pp. 273-324.

[23] Williams, Ryan, Carla P. Gomes, and Bart Selman. "Backdoors to Typical Case Complexity." Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2003, pp. 1173-1178.

[24] Zhu, Xiaojin. "Semi-Supervised Learning Literature Survey." Computer Sciences Technical Report 1530, University of Wisconsin-Madison, 2005.

[25] Settles, Burr. "Active Learning Literature Survey." Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009.

[26] **Menzies**, Tim. "Shockingly Simple: 'Keys' for Better AI for SE." IEEE Software, vol. 38, no. 2, 2021, pp. 114-118.

References (cont.)

[27] Nair, Vivek, Tim **Menzies**, et al. "Finding Faster Configurations Using FLASH." IEEE Transactions on Software Engineering, vol. 46, no. 7, 2018, pp. 794-811.

[28] Yang, Chuanpeng, et al. "Survey on Knowledge Distillation for Large Language Models: Methods, Evaluation, and Application." ACM Transactions on Intelligent Systems and Technology, 2024.

[29] Lindauer, Marius, et al. "SMAC3: A Versatile Bayesian Optimization Package for Hyperparameter Optimization." Journal of Machine Learning Research, vol. 23, no. 54, 2022, pp. 1-9.

[30] **Menzies**, Tim, et al. "MOOT: A Repository of Many Multi-Objective Optimization Tasks." Proceedings of the International Conference on Mining Software Repositories (MSR), ACM, 2026.

[31] Ganguly, Kishan Kumar, and Tim **Menzies**. "How Low Can You Go? The Data-Light SE Challenge." Proceedings of the ACM on Software Engineering, vol. 3, no. FSE, Article FSE185, 2026.

[32] Ganguly, Kishan Kumar, and Tim **Menzies**. "Which Optimizer, At What Budget? A Tournament of Optimizers for Search-Based SE." 2026, under review.

[33] Rayegan, Amirali, and Tim **Menzies**. "Minimal Data, Maximum Clarity: A Heuristic for Explaining Optimization." Journal of Systems and Software, vol. 238, 2026, 112897.

[34] **Menzies**, Tim, and Srinath Srinivasan. "Can AI Be Easy? Lessons Learned from the EZR.py Toolkit." arXiv preprint arXiv:2606.03640, 2026.

[35] Senthilkumar, Lohith, and Tim **Menzies**. "Can Large Language Models Improve SE Active Learning via Warm-Starts?" ACM Transactions on Software Engineering and Methodology, 2026.

[36] Srinivasan, Srinath, and Tim **Menzies**. "Better Together, in the Right Order: Classical-then-LLM Optimization for SE." arXiv preprint, 2026.

[37] Adams, Gabrielle S., et al. "People Systematically Overlook Subtractive Changes." Nature, vol. 592, 2021, pp. 258-261.