

On the Value of Sampling and Pruning for Search-Based Software Engineering

PhD Defense

Jianfeng Chen

Timothy Menzies(Advisor)
Min Chi

Emerson Murphy-hill
Xipeng Shen (GSR)

Department of Computer Science
North Carolina State University

May 9, 2019



Find this slides at <http://tiny.cc/jcdefense>

Dissertation Statement

For the optimization of search-based software engineering (SBSE) problems,

- *given a proper configuration selector or comparator built upon **decision space**,*
- **oversampling-and-pruning (OSAP)** *is better than a standard mutation based **evolutionary approach (EVOL)**;*
- *where “better” is measured in terms of runtimes, number of evaluations and value of final results.*

Major content in this talk: Four generations of configuration selector/comparator, i.e. OSAP1, OSAP2,...

Publications List

- [ASE Submitted] **Jianfeng Chen** and Tim Menzies. "On the Benefits of Restrained Mutation: Faster Generation of Smaller Test Suites" Submitted to IEEE/ACM International Conference on Automated Software Engineering (ASE 2019).
 - [TSE'18] **Jianfeng Chen**, Vivek Nair, Rahul Krishna, and Tim Menzies. ""Sampling" as a Baseline Optimizer for Search-based Software Engineering." IEEE Transactions on Software Engineering (2018).
 - [IEEE CLOUD'18] **Jianfeng Chen**, and Tim Menzies. "RIOT: A Stochastic-Based Method for Workflow Scheduling in the Cloud." 2018 IEEE 11th International Conference on Cloud Computing.
 - [IST'17] **Jianfeng Chen**, Vivek Nair, and Tim Menzies. "Beyond evolutionary algorithms for search-based software engineering." Information and Software Technology (2017).
 - [FSE Submitted] **Jianfeng Chen**, Joymallya Chakraborty, Philip Clark, Kevin Haverlock, Snehit Cherian and Tim Menzies. "Predicting Breakdowns in Cloud Services (with SPIKE)". Submitted to ESEC/FSE 2019 - Industry Paper Track
 - [TSE'19] Junjie Wang, *et al.*. "Characterizing Crowds to Better Optimize Worker Recommendation in Crowdsourced Testing ". IEEE Transactions on Software Engineering(2019).
 - [EMSE'18] Tianpei Xia, *et al.*. "Hyperparameter optimization for effort estimation." Empirical Software Engineering (EMSE), 2018
 - [MSR'18] Vivek Nair, *et al.*. "Data-Driven Search-based Software Engineering." The Mining Software Repositories (MSR) 2018.
 - [SSBSE'16] Vivek Nair, *et al.*. "An (accidental) exploration of alternatives to evolutionary algorithms for sbse." In International Symposium on SBSE, 2016.
- * Covered in this talk.

Impact on SE community

- 21 citations per year since 2017, according to the google scholar
- Extended by other researchers in software effort estimation.¹
- Similar insights for space reduction in solving probabilistic constrained simulation optimization problems.[Horng'18]²
- and so on

¹Sarro, Federica et al."Linear programming as a baseline for software effort estimation." ACM transactions on software engineering and methodology (TOSEM) 2018

²Horng, Shih-Cheng, and Shieh-Shing Lin. Embedding Ordinal Optimization into Tree-Seed Algorithm for Solving the Probabilistic Constrained Simulation Optimization Problems. Applied Sciences 8.11 (2018)

Feedback from the Oral Prelim Exam

- To answer: why does oversampling work
- When to use oversampling. Difference among developed methods
- To revisit: previous problem + improved method
- To explore: the testing problem
- Identify specific propriety in software engineering models

This talk ...

- review previous developed algorithms; analysis on their achievements and limitations
- latest oversampling technique
- revisit the old model and
- explore the testing problem.

Contents of this talk

■ Overview

- What is SBSE?
- Motivation of this research

■ Early generations of OSAP

- OSAP1, OSAP2, OSAP3
- Achievements and Limitations ← Why did they work/not work?

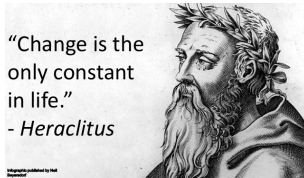
■ Delta-oriented surrogate model embedded OSAP

- OSAP4 ← addressing previous limitations
- Revisiting XOMO & POM3 model ← old problems first
- Test suite generation ← a more challenging problem
- Critics on OSAP4

■ Conclusion and future work

Modeling SE problems

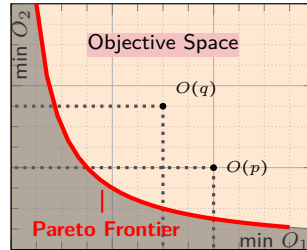
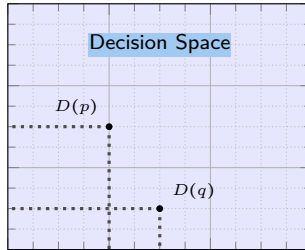
- (Requirement) What feature to include or develop in the project
- (Deployment) How to assign software to cloud environment
- (Test) How to find smaller set of test suite, converging more code



Search-based Software Engineering

- Modeling
- Decision space, objective space
- Search for optimal objective/goal within decision space

Search-based Software Engineering (SBSE)

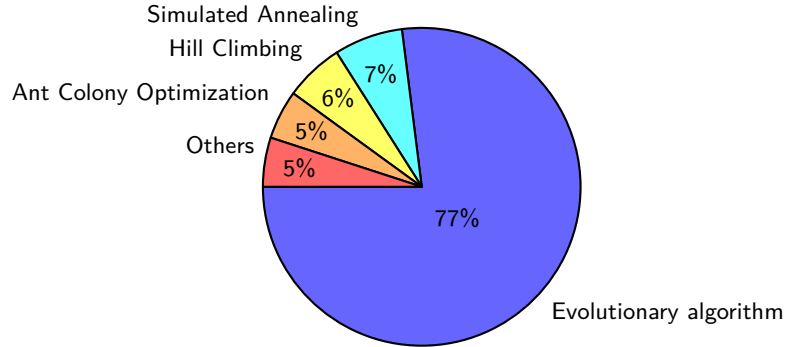


Dominance

p dominance q if and only if

- For every objective, p is no worse than q AND
- Exists at least one objective, p is better than q .

Existing Research



From CREST Center, UCL ³

³[zhang18] A repository and analysis of authors and research articles on search-based Software Engineering.

How does Evolutionary algorithms (EVOL) work?

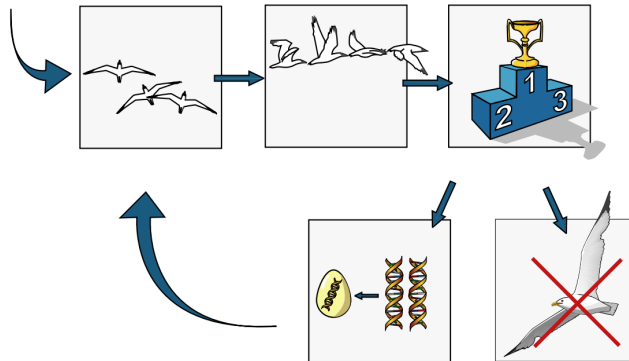


Figure: Framework⁴ of the EVOL algorithms.

⁴Doncieux, Stephane, et al. "The ROBUR project: towards an autonomous flapping-wing animat." Proceedings of the Jounes MicroDrones, Toulouse (2004).

Is EVOL good enough?

- 😊 EVOL - Treats the problem as black-box
- 😊 EVOL - Easy to deploy to new problem
- 😞 Evaluates 1000s, 1,000,000s of configurations
 - Airspace operation model verification – 7 days [Krall'14] ⁵
 - Test suite generation – weeks [Yoo'12] ⁶
 - Software clone evaluation at pc – 15 years [Wang'13] ⁷

Need a faster framework!

- Economic considerations – save computing resources
- Faster response to the environment changes
- As a baseline method – judge the problem before exploration
- Opens up a new research direction

⁵Krall, Joseph, Tim Menzies, and Misty Davies. "Learning the task management space of an aircraft approach model." (2014).

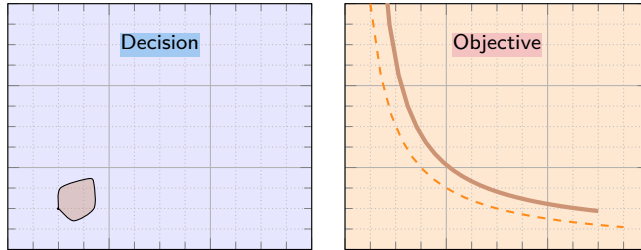
⁶Yoo, Shin, and Mark Harman. "Regression testing minimization, selection and prioritization: a survey." Software Testing, Verification and Reliability

⁷Wang, Tiantian, et al. "Searching for better configurations: a rigorous approach to clone evaluation." Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. ACM, 2013.

Roadmap

- 1 Overview
- 2 Early generations of OSAP
 - OSAP1 - Utilizing "golden" region assumption [SSBSE'16, IST'17]
 - OSAP2 - Utilizing the expert or domain knowledge [TSE'18]
 - OSAP3 - The linear surrogate model [Cloud'18]
- 3 Delta-oriented surrogate model embedded OSAP
- 4 Conclusion and future work

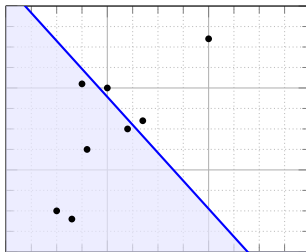
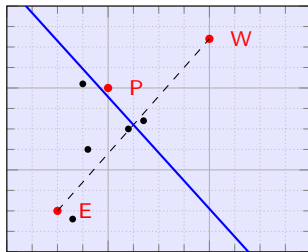
OSAP1 - "Golden" region assumption



Assumption: A small region in the **decision space** covers the majority of the near-optimal configurations.

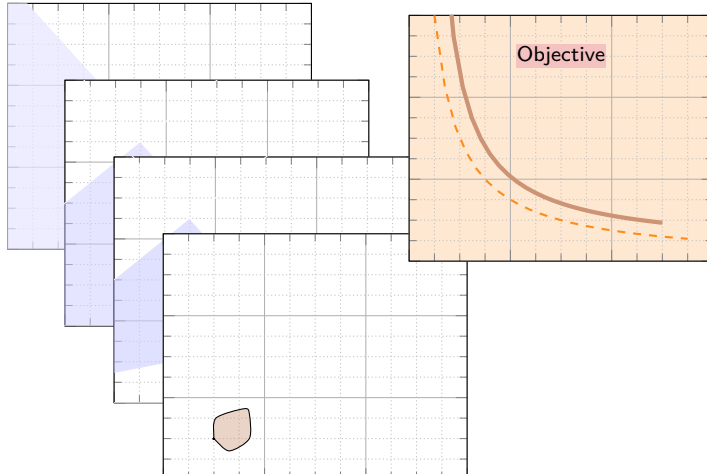
Question: How to figure out such region?
⇒ Similar decisions implies similar objectives

WHERE Geometric Learner



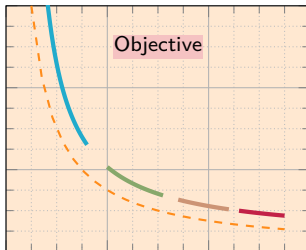
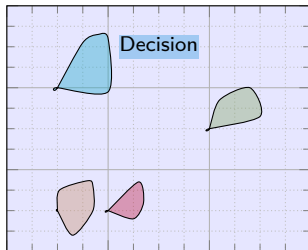
- step 1: get a random configuration, e.g. P
- step 2: find furthest point to P , as E
- step 3: find furthest point to E , as W
- step 4: connect EW . find medium line (hyperplane)
- step 5: compare E and W , select the half-space
- Recursively execute 1 - 5

WHERE Geometric Learner



OSAP2 - Just one "golden" region?

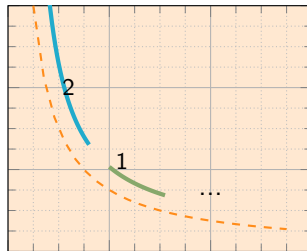
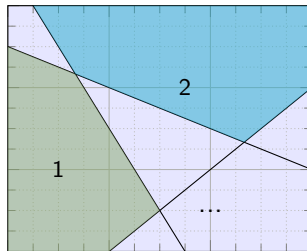
No!



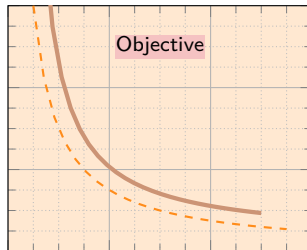
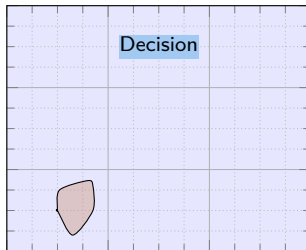
Improvement from OSAP1

OSAP2: utilize the domain or expert knowledge to get the rough sub-space.

OSAP2 - Divide *with domain knowledge*, and conquer



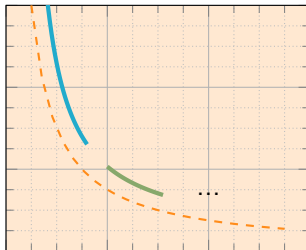
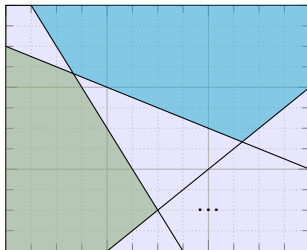
Comments



Achievements of OSAP1

- Oversampling can outperform the mutation based EVOL under some circumstances
- An effective geometric learner

Comments



Achievements of OSAP2

- Fixed OSAP1 via doing the decision space partition first, using the domain or expert knowledge
- Tested in two constrained case studies

Comments

Limitations of OSAP1

- Majority of optimal solutions can be found in one small region
- Similar decisions implies similar objectives

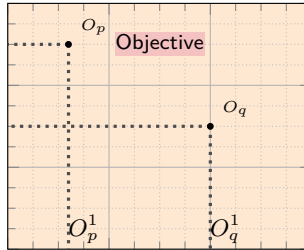
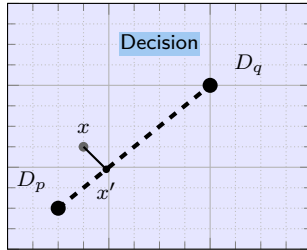
Limitations of OSAP2

- Majority of optimal solutions can be found in several small regions
- Similar decisions implies similar objectives
- Requires the domain or expert knowledge

OSAP3 - Surrogate model

- 😞 Just figure out one (or more) region in the decision space is not enough
- Expecting: given any configurations, determine which one is better/best
- **Surrogate model:** an alternative model to replace the original SE model.
- Simple. fast.
- Estimating the objective is the most directed way
- If SE model has ≥ 2 objectives, build ≥ 2 surrogate models. (one surrogate for each objective)

OSAP3 - Linear surrogate model



$$\frac{|D_p D_q|}{|D_p D_{x'}|} = \frac{O_p^1 - O_q^1}{O_p^1 - O_x^1} = \frac{O_p^2 - O_q^2}{O_p^2 - O_x^2} = \dots$$

$$O_x^1 = O_p^1 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^1 - O_q^1)$$

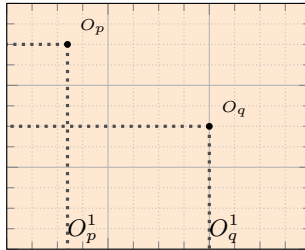
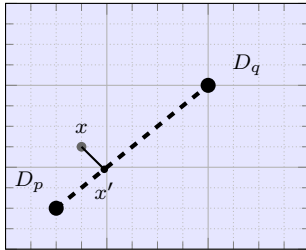
$$O_x^2 = O_p^2 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^2 - O_q^2)$$

OSAP3 - Utilizing the linear surrogate model

- Need a few ≈ 100 evaluated configurations (anchors)
- Three ways to assign the anchors: 1) random , 2) diagonal, 3) 1+2
- Given evaluated anchors, estimate over 10,000 other configurations via surrogate models.
- How to select the p and q ? Nearest and furthest anchors

```
1 Anchors  $\leftarrow n$  evaluated items;  
2 Randoms  $\leftarrow N \gg n$  un-evaluated items;  
3 foreach  $c \in \textit{Randoms}$  do  
4    $A_n \leftarrow$  configurations in Anchors that nearest to  $c$ ;  
5    $A_f \leftarrow$  configurations in Anchors that furthest to  $c$ ;  
6   foreach  $o \in \{o_1, o_2, \dots\}$  do  
7      $\lfloor$  Accessing  $o_c$  using surrogate model;  
8 Collect all items and return all frontiers;
```

Recap



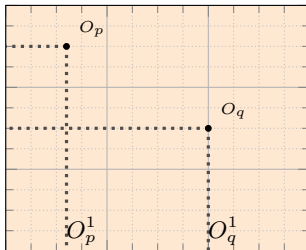
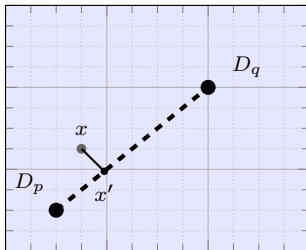
$$O_x^1 = O_p^1 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^1 - O_q^1)$$

$$O_x^2 = O_p^2 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^2 - O_q^2)$$

Achievements of OSAP3

- Replacing previous geometric learners by surrogate model
- Given a small number of configurations evaluated, any configurations' objectives can get estimated
- Successfully found the deployment plan for complex workflows

Recap



$$O_x^1 = O_p^1 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^1 - O_q^1)$$

$$O_x^2 = O_p^2 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^2 - O_q^2)$$

Limitations of OSAP3

OSAP3 is highly relied on the linear surrogate model.

What if the SE does not have linearity kernel, or the linearity inside is weak?

Roadmap

- 1 Overview
- 2 Early generations of OSAP
- 3 **Delta-oriented surrogate model embedded OSAP**
 - OSAP4 - Delta-oriented surrogate model [ASE'19*]
 - Case study I: revisit XOMO & POM3
 - Case study II: test suite generation
 - Summary of OSAP4
- 4 Conclusion and future work

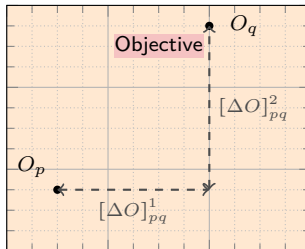
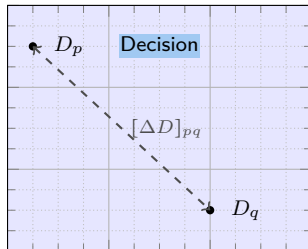
On the surrogate model...

- Ultimate purpose of the surrogate model is to compare or select the better configurations.
- The OSAP3 surrogate model was design to predict the objectives precisely
- Having the objectives, we can do comparisons
- For the purpose of configuration comparisons, is "predicting the objectives" a must?

Delta-oriented surrogate model

- Given any two configurations p, q , predict $[\Delta O]_{pq}$, i.e. $(O_p - O_q)$.
- Predict the $[\Delta O]_{pq}$ from $[\Delta D]_{pq}$ (*again, one predictor for each objective*)
- $[\Delta O]_{pq}$ need not be precise. **Correct sign is good enough.** ($O_p <? O_q$)

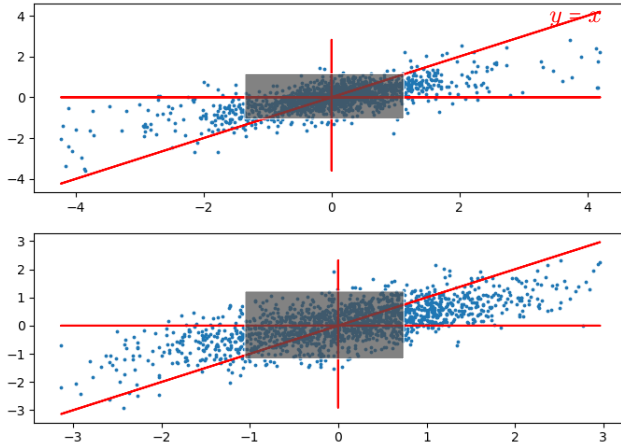
Delta-oriented surrogate model



$[\Delta D]$ (vector)	$[\Delta O]^1$	$[\Delta O]^2$
(pq) ■■■■■	★	●
(pr) ■■■■■	★	●
(uv) ■■■■■	★	●
...

We found that **KNN** is a proper ML learner here.

Delta-oriented surrogate model



- Each chart is a actual $[\Delta O]$ vs. predicted $[\Delta O]$
- Quadrant I, III : FILLED
- **Quadrant II, IV: EMPTY**

Delta-oriented surrogate model

Framework of OSAP4

```
1 Samples  $\leftarrow$  ( $n = 100$ ) evaluated items;  
2 PF  $\leftarrow$  pareto frontier in Samples;  
3 foreach  $x \in PF$  do  
4   Neighbors  $\leftarrow$  Configurations near  $x$  in decision space;  
5   get all  $[\Delta D]_{pq}$  and  $[\Delta O]_{pq}^i$  ( $i = 1, 2, \dots$ ), where  $pq$  are pairs in Neighbors;  
6   train KNN model to predict  $[\Delta O]_{pq}^i$  from  $[\Delta D]_{pq}$  ( $i=1,2,\dots\# \text{of objs}$ );  
7    $y \leftarrow$  random configuration;  
8   predict  $[\Delta O]_{xy}^i$  given  $[\Delta D]_{xy}$ ;  
9   If exists  $i$  such that  $([\Delta O]_{xy}^i \ll 0)$ , evaluate  $y$  using model;  
10  repeat Line 7-9, or Goto 3;  
11 Collect all new evaluated configurations, update Samples;  
12 Goto 2 or Terminate;  
13 Return all pareto frontiers achieved;
```

Roadmap

- 1 Overview
- 2 Early generations of OSAP
- 3 **Delta-oriented surrogate model embedded OSAP**
 - OSAP4 - Delta-oriented surrogate model [ASE'19*]
 - **Case study I: revisit XOMO & POM3**
 - Case study II: test suite generation
 - Summary of OSAP4
- 4 Conclusion and future work

Case study I: revisit XOMO and POM3

Objectives for the XOMO:

- Reduce risk;
- Reduce effort;
- Reduce defects;
- Reduce develop times.

Table: Descriptions of the XOMO decisions.

scale factors (exponentially decrease effort)	prec: have we done this before? flex: development flexibility resl: any risk resolution activities? team: team cohesion pmat: process maturity
upper (linearly decrease effort)	acap: analyst capability pcap: programmer capability pcon: programmer continuity aexp: analyst experience pexp: programmer experience ltex: language and tool experience : ...
lower (linearly increase effort)	rely: required reliability data: 2nd memory requirements cplx: program complexity ruse: software reuse docu: documentation requirements : ... stor: main memory requirements pvol: platform volatility

Case study I: revisit XOMO and POM3

Objectives for the POM3:

- Increase completion rates,
- Reduce idle rates,
- Reduce overall cost.

Table: List of POM3 decisions.

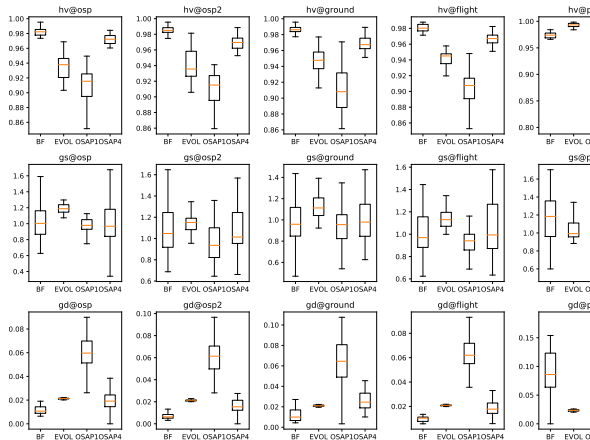
Decision	Description
Culture	Number (%) of requirements that change.
Criticality	Requirements cost effect for safety critical systems.
Criticality Modifier	Number of (%) teams affected by criticality.
Initial Known	Number of (%) initially known requirements.
Inter-Dependency	Number of (%) requirements that have interdependencies to other teams.
Dynamism	Rate of how often new requirements are made.
Size	Number of base requirements in the project.
Plan	Prioritization Strategy: 0= Cost Ascending; 1= Cost Descending; 2= Value Ascending; 3= Value Descending; 4= $\frac{Cost}{Value}$ Ascending.
Team Size	Number of personnel in each team

XOMO and POM3

Benchmark scenarios

- XOMO-OSP : NASA flight guidance system
- XOMO-OSP2: Another NASA flight guidance system
- XOMO-Flight: NASA JPL general flight system
- XOMO-Ground: NASA JPL general ground system
- POM3a: A broad space of project
- POM3b: Critical small project
- POM3c: Highly dynamic large projects

Comparing the effectiveness



- **BF** ("brute force") = randomly sample 10,000 configurations; evaluate all; report the best;
- **EVOL** = evolutionary algorithms, with hyperparameter tuned
- **OSAP1** = previous results, applying WHERE geometric learner
- **OSAP4** = using delta-based surrogate model

Comparing the effectiveness (EVOL vs. OSAPs)

model	Hypervolume		General Spread		Generated distance	
	OSAP1	OSAP4	OSAP1	OSAP4	OSAP1	OSAP4
osp	●	●	●	●	●	●
osp2	●	●	●	●	●	●
ground	●	●	●	●	●	●
flight	●	●	●	●	●	●
pom3a	●	●	●	●	●	●
pom3b	●	●	●	●	●	●
pom3c	●	●	●	●	●	●
same+better	1/7	6/7	4/7	6/7	0/7	5/7

- **Hypervolume:** How large the area the obtained PF can covered?
- **General Spread:** Can PF provide enough choices to the users?
- **Generated distance:** How close the obtained PF to the theoretically-PF?

Observations

- In majority cases, OSAP4 is same or better than EVOL methods;
- OSAP1 is no good enough. Look back the digits, it was worse than EVOL by 27% on average.
- *OSAP1 conclusion not consistent with previous?* Following an updated HV/GS/GD calculation guidance ^a

^aLi, Miqing et al. "A Critical Review of" A Practical Guide to Select Quality Indicators for Assessing Pareto-Based Search Algorithms in Search-Based Software Engineering" 2018 IEEE/ACM 40th International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER)

Comparing the effectiveness (EVOL vs. BF)

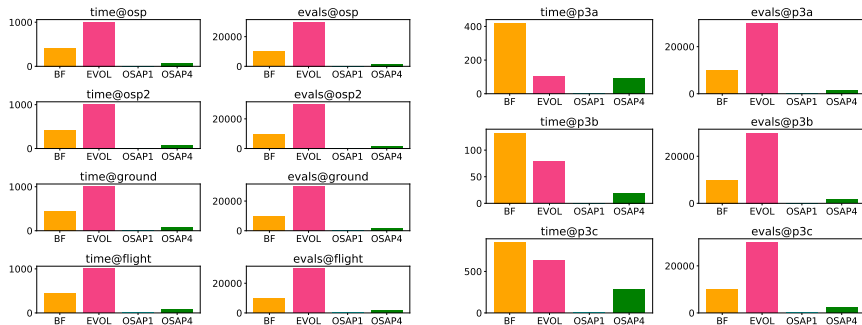
model	Hypervolume	General Spread	Generated Distance
	BF better?	BF better?	BF better?
osp	●	●	●
osp2	●	●	●
ground	●	●	●
flight	●	●	●
pom3a	●	●	●
pom3b	●	●	●
pom3c	●	●	●
better+same	6/7	6/7	5/7

- **Hypervolume:** How large the area the obtained PF can covered?
- **General Spread:** Can PF provide enough choices to the users?
- **Generated distance:** How close the obtained PF to the theoretically-PF?

Observations

- BF is good enough in majority cases
- If time permits, randomly selecting and evaluating large amount of candidates is a good strategy.
Simple! Effective!
- Is the crossover, mutation in evolutionary algorithms really helpful in SBSE?

Comparing the efficiency (EVOL vs. OSAPs)



- 4 color bars, left to right: BF, EVOL, OSAP1, OSAP4
- Column 1-4: time@XOMOs, eval@XOMOs, time@POM3s, eval@POM3s
- **OSAP1 is always extremely fast.**
- **OSAP4 is frugal.**

Roadmap

- 1 Overview
- 2 Early generations of OSAP
- 3 **Delta-oriented surrogate model embedded OSAP**
 - OSAP4 - Delta-oriented surrogate model [ASE'19*]
 - Case study I: revisit XOMO & POM3
 - **Case study II: test suite generation**
 - Summary of OSAP4
- 4 Conclusion and future work

Case study II: test suite generation

Get diverse solutions(models) to a 3-SAT problems could be helpful to in software testing.

```
1 int mid(int x, int y, int z) {  
2   if (x < y) {  
3     if (y < z) return y;  
4     else if (x < z) return z;  
5     else return x;  
6   } else if (x < z) return x;  
7   else if (y < z) return z;  
8   else return y;  
9 }
```

- path 1: [C1: $x < y < z$] L2→L3
- path 2: [C2: $x < z < y$] L2→L3→L4
- path 3...

- $\vee C_i$ (Disjunction form, meet any of formula)
- $\Rightarrow \wedge C'_j$ (Conjunction form, meet all formulas)
- Model checking tools transform a program to CNF (conjunctive normal form)
- A valid assignment to **CNF** \leftrightarrow a test case
- A test suite with enough diverse \leftarrow figure out enough amount of valid solutions meet the **CNF**
- NP-Complete – Easy to verify, hard to solve
- **Decision space:** 2^v ($v = \#$ of variables) \rightarrow valid configurations
- **Objective space:** not really interesting. Enough valid solution to guarantee diversity is more important.

Test suite generation::state-of-the-art⁸

Efficient Sampling of SAT Solutions for Testing

- Introduced by Dutra *et al.* in ICSE 2018
- Open sourced. Compared to former STOA
- Assert to be better than old STOA
- To achieve diversity, generates huge amount samples (> 2 millions)
- New samples fetched from crossover, or some mutations ~ EVOL
- **Limitations:**
 - long execution time ≈ 3 hrs
 - samples are not verified. (may be invalid)
 - too many samples. Hard to test all suite

⁸Dutra, Rafael, et al. "Efficient sampling of SAT solutions for testing." 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE). IEEE, 2018.

Test suite generation::adapting OSAP4

1. $Samples \leftarrow (n = 100)$ evaluated items
 2. $PF \leftarrow$ pareto frontier in $Samples$
 3. foreach $x \in PF$
 - 3.1 $Neighbors \leftarrow$ Configurations near x in decision space
 - 3.2 train delta-oriented surrogate model
 - 3.3 $y \leftarrow$ random configuration
 - 3.4 predict $[\Delta O]^{xy}$
 - 3.5 if desired, evaluate y
 - 3.6 repeat from 3.3, or Goto 3
 4. Collect all new evaluated configurations, update $Samples$
 5. Goto 2 or Terminate
 6. Return all pareto frontiers achieved
- **No PF here:** k -means. centers of cluster
 - $\Delta D = p \oplus q$, exclusive-or
 - **Local neighbors?** To improve diversity, use global pairwise delta from samples
 - **Predict ΔO via ΔD** \rightarrow applying a ΔD to x , is it still valid?
 - Surrogate model: answers $\uparrow\uparrow$
 - Learn pairwise ΔD from the valid samples. Some ΔD are more common

Test suite generation::adapting OSAP4

1. $Samples \leftarrow (n = 100)$ valid items
2. $PF \leftarrow$ center of k -means clusters
3. Get the frequency of unique deltas among all pairs in $Samples$ as the surrogate model
4. foreach $x \in PF$
 - 4.1 pick one or more $[\Delta D]$, with high frequency ones in priority
 - 4.2 verify $x \oplus [\Delta D]$; fix by SAT solvers
 - 4.3 repeat from 5.1 or Goto 5
5. Collect all valid configurations, update $Samples$
6. Goto 2 or Terminate
7. Return all valid samples achieved

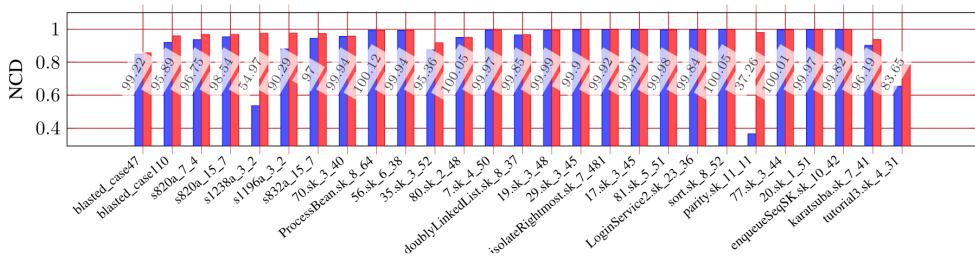
Test suite generation::experiments

Benchmarks	Vars
blasted_case47	118
blasted_case110	287
s820a_7_4	616
s820a_15_7	685
s1238a_3_2	685
...	
35.sk_3_52	4894
80.sk_2_48	4963
7.sk_4_50	6674
doublyLinkedList.sk_8_37	6889
19.sk_3_48	6984
29.sk_3_45	8857
isolateRightmost.sk_7_481	10024
...	
LoginService2.sk_23_36	11510
sort.sk_8_52	12124
...	
enqueueSeqSK.sk_10_42	16465
karatsuba.sk_7_41	19593
tutorial3.sk_4_31	486193

Research questions

- RQ1 - can delta-oriented sampling (OSAP4) return a diverse test suite?
- RQ2 - can OSAP4 return the test suite with less test cases?
- RQ3 - is the sampling procedure fast?

Test suite generation::RQ1 - got enough diversity?



- BLUE: OSAP4. RED: QuickSampler(STOA)
- NCD is the **diversity metrics** for this problem.
- Termination rule: NCD got improved by less than 5% within 10 minutes.
- Except in 2 benchmarks, **OSAP4 achieved more than 95% of the diverse of STOA.**

Test suite generation::RQ2 - less test cases?

Table: Number of unique cases in the test suite.

Benchmarks	OSAP4 O	QuickSampler Q	Q/O
blasted_case47	2799	71	0.00
blasted_case110	174	2386	13.71
s820a_7_4	37363	124457	3.30
80.sk_2_48	553	54440	98.44
...
doublyLinkedList.sk_8_37	178	12042	67.65
19.sk_3_48	104	200	1.90
29.sk_3_45	125	660	5.28
isolateRightmost.sk_7_481	15380	7510	0.49
7.sk_4_50	158	18090	114.49
doublyLinkedList.sk_8_37	178	12042	67.65
...
77.sk_3_44	145	33858	233.50
karatsuba.sk_7_41	39	4210	107.94
tutorial3.sk_4_31	236	2953	12.51

Observations

- Q/O is 91x (in average), 14x (in medium).
- That is, sharing the similar diverse, compared to QuickSampler's, running the test suites from OSAP4 can save > 90% testing times.

Test suite generation::RQ3 - sampling faster?

Table: Termination time (sorted by speedup)

Model	OSAP4	QuickSampler	Speedup
7.sk_4_50	2.47	1833.04	739.92
17.sk_3_45	2.18	1503.44	687.05
35.sk_3_52	1.85	966.40	520.44
81.sk_5_51	2.06	421.63	204.13
ProcessBean.sk_8_64	115.62	9296.81	80.40
20.sk_1_51	32.63	2595.68	79.54
...			
LoginService2.sk_23_36	75.35	99.3716	1.32
19.sk_3_48	29.84	23.43	0.79
isolateRightmost.sk_7_481	4031.86	1675.66	0.42
s832a_15_7	7193.96	1465.93	0.20
70.sk_3_40	2605.32	288.56	0.11

On average, it is 53X speedup.

Test suite generation::results

Summary

Comparing to the state-of-the-art QuickSampler, in majority benchmarks, the OSAP4

- finds test suite with similar diversity
- returns the test suite with much less cases
- terminates in much shorter time

Recap

Achievements of OSAP4

- No linearity dependence. Learning or transferring the deltas
- The learning model is not necessary to be accurate
- The initial sample size can be smaller than previous versions of OSAP

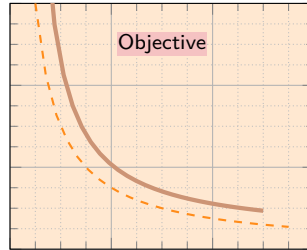
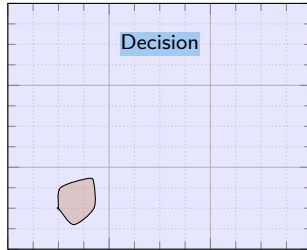
Limitations of OSAP4

- More model evaluations than previous versions (*more uncertainty*)
- Other surrogate model kernel (in addition to KNN, or the frequency) needs to be explored
- Local monotonic?

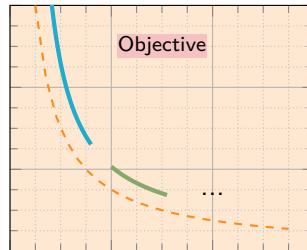
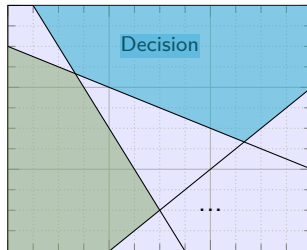
Roadmap

- 1 Overview
- 2 Early generations of OSAP
- 3 Delta-oriented surrogate model embedded OSAP
- 4 **Conclusion and future work**
 - Reviewing OSAP
 - Executive summary
 - Future work

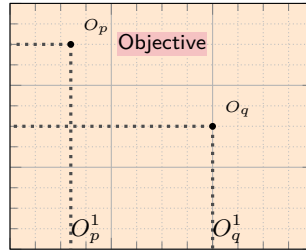
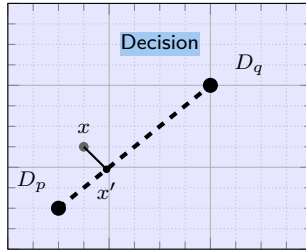
OSAP1



OSAP2

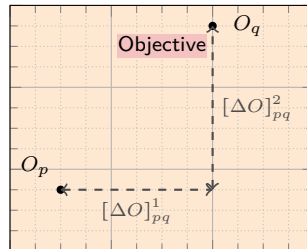
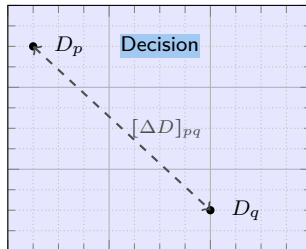


OSAP3



$$O_x^1 = O_p^1 - \frac{|D_p D_{x'}|}{|D_p D_{x'}|} (O_p^1 - O_q^1)$$

OSAP4



$[\Delta \mathbf{D}]$ (vector)	$[\Delta \mathbf{O}]^1$	$[\Delta \mathbf{O}]^2$
(pq) ■ ■ ■ ■ ■	★	●
(pr) ■ ■ ■ ■ ■	★	●
(uv) ■ ■ ■ ■ ■	★	●
...

OSAP generations

Gen	Assuming	Decision space	Objective space	Study cases	Constraint exists	Surrogate model
I	A “golden” region	numeric	numeric	XOMO POM3	✗	✗
II	n “golden” regions	boolean, discrete	numeric	SPL NRP	✓	✗
III	Linearity of the model	discrete	numeric	Workflow	✗	✓
IV	Local monotonic	numeric, discrete	numeric	XOMO POM3 Testing	✓	✓

Executive summary

- Try OSAP before the EVOL
- Always OSAP1 first. Simple, fast! Can use that as baseline method
- For the constraint model, which is not easy to get large amount of samples, OSAP4 could be helpful. (N samples can get $O(N^2)$ deltas)
- If the model is known to have some linearity features, OSAP3 is a good choice.
- “No free lunch theorem”⁹. No simple optimizer is the best for all problems.

⁹Wolpert, et al. "No free lunch theorems for optimization." IEEE transactions on evolutionary computation 1.1 (1997): 67-82.

Future work

- **Ensemble Learning** • random forest • hyperparameter tuning • ...
- **Incremental Sampling** • regression testing • dynamic cloud deployment • ...
- **More on the constraint models** • weighted sampling and counting¹⁰ • AI applications • ...
- **Not just SBSE** • boosting stochastic gradient descent • feature reduction • ...

¹⁰Chakraborty, Supratik, et al. "Distribution-aware sampling and weighted model counting for SAT." Twenty-Eighth AAAI Conference on Artificial Intelligence. 2014.

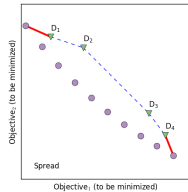
Questions?



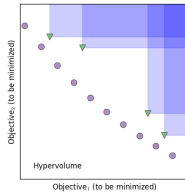
Backup slides

XOMO and POM3::Metrics

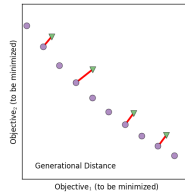
How to measure the results? What is a good pareto frontier?



Generated Spread (GS)



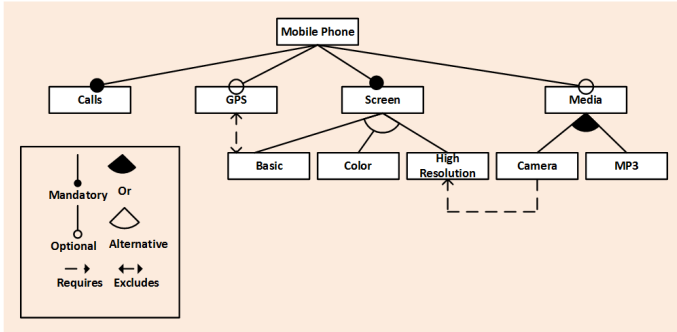
Hypervolume (HV)



Generational distance (GD)

GS, GD: Less is better
HV: Higher is better

Case study(review): Software Product Line



- Constrained model. Initial configurations given from SAT solver.
- Divided via the number of features → small?, medium product? ...
- OSAP2 is **effective, and fast**, compared to [Henard'15] ¹¹

¹¹Henard, Christopher, et al. "Combining multi-objective search and constraint solving for configuring large software product lines." Software Engineering (ICSE), 2015

Case study(review): Next Release Problem

- Which requirements should be implemented for the next version?
 - Subject to: customer satisfaction, budget, precedence constraints
 - Objective: higher customer satisfaction + less development time + less cost
-
- Group (divide) the configurations via $WL(\mathbf{y}) = \|\{y_i < P/2\}\|$
 - i.e. how many features are scheduled in the first half of the plan
 - Compared to the EVOL, OSAP2 was **effective and fast**.

Case study(review): Workflow deployment

- A workflow is the combination of sub computing tasks
- Expressed as directed acyclic graph (DAG)
- For each task, what's the best AWS EC2 instance?
- Two objectives to minimize
 1. Time to complete the whole workflow
 2. \$\$\$ spending
- More than 50 AWS EC2 types. (8 adopted in experiment)
- Experiment outputs:
 - (Efficiency) OSAP3 was 11 to 39 times faster than a state-of-the-art approach (EVOL based).
 - (Effectiveness) In the five largest workflows, OSAP3's results were better among 13/15 (87%) of all the quality indicators.