Test Scheduling Algorithm Safety Evaluation Framework

Claire Leong

Goal: to make a generic framework for evaluating test scheduling algorithms at scale from the historical record.

Background

- Changelist (CL) = files changed in a code commit
  - A test is affected iff a file being changed is present in the transitive closure of the test dependencies (Regression Test Selection)
- Safety = would skipping this test target miss a transition?
- Transition = a change in target results, either from failing->passing or passing->failing

Safe Targets *skipping this target at CL2 would not miss a transition*

Time →

Changelist	CL1	CL2
Target Result	P	P
Safety	**-**	**Safe**
Transition	-	P->P

** = affected but no pass or fail result*

Safe Targets *skipping this target at CL2 would not miss a transition*

Time →

Changelist	CL1	CL2
Target Result	F	F
Safety	**-**	**Safe**
Transition	-	F->F

** = affected but no pass or fail result*

Safe Targets *skipping this target at CL2 or CL3 would not miss a transition*

Time →

Changelist	CL1	CL2	CL3
Target Result	P	*	P
Safety	**-**	**Safe**	**Safe**
Transition	-	P->P	P->P

We make the simplifying assumption that CL2 does not introduce a problem that is fixed by CL3.

** = affected but no pass or fail result*

Safe Targets *skipping this target at CL2 or CL3 would not miss a transition*

Time →

Changelist	CL1	CL2	CL3
Target Result	F	*	F
Safety	**-**	**Safe**	**Safe**
Transition	-	F->F	F->F

** = affected but no pass or fail result*

Unsafe Targets *skipping this target at CL2 would definitely miss a transition*

Time →

Changelist	CL1	CL2
Target Result	P	F
Safety	**-**	**Unsafe**
Transition	-	P->F

** = affected but no pass or fail result*

Unsafe Targets *skipping this target at CL2 would definitely miss a transition*

Time →

Changelist	CL1	CL2
Target Result	F	P
Safety	**-**	**Unsafe**
Transition	-	F->P

** = affected but no pass or fail result*

Maybe Unsafe Targets *skipping this target at CL2 or CL3 might miss a transition*

Time →

Changelist	CL1	CL2	CL3
Target Result	P	*	F
Safety	**-**	**Maybe unsafe**	**Maybe unsafe**
Transition	-	P->F	P->F

Without the data about whether CL2 passed or failed, we cannot accurately determine which CL introduced the problem.

** = affected but no pass or fail result*

Maybe Unsafe Targets *skipping this target at CL2 or CL3 might miss a transition*

Time →

Changelist	CL1	CL2	CL3
Target Result	F	*	P
Safety	**-**	**Maybe unsafe**	**Maybe unsafe**
Transition	-	F->P	F->P

** = affected but no pass or fail result*

Project Overview

- Implementation:
    1. Determine safety information for historical changelists
    2. Evaluate the safety of test selection algorithms
    3. Implement optimal, pessimal and random test selection algorithms

Project Overview

- Used over 2 datasets:

Small Dataset	Large Dataset
2 days of CL data (6-8 Dec 2017)	1 month of CL data (October 2017)
11k changelists	900k changelists
1k total targets	4m total targets
430k times targets were affected	16b times targets were affected

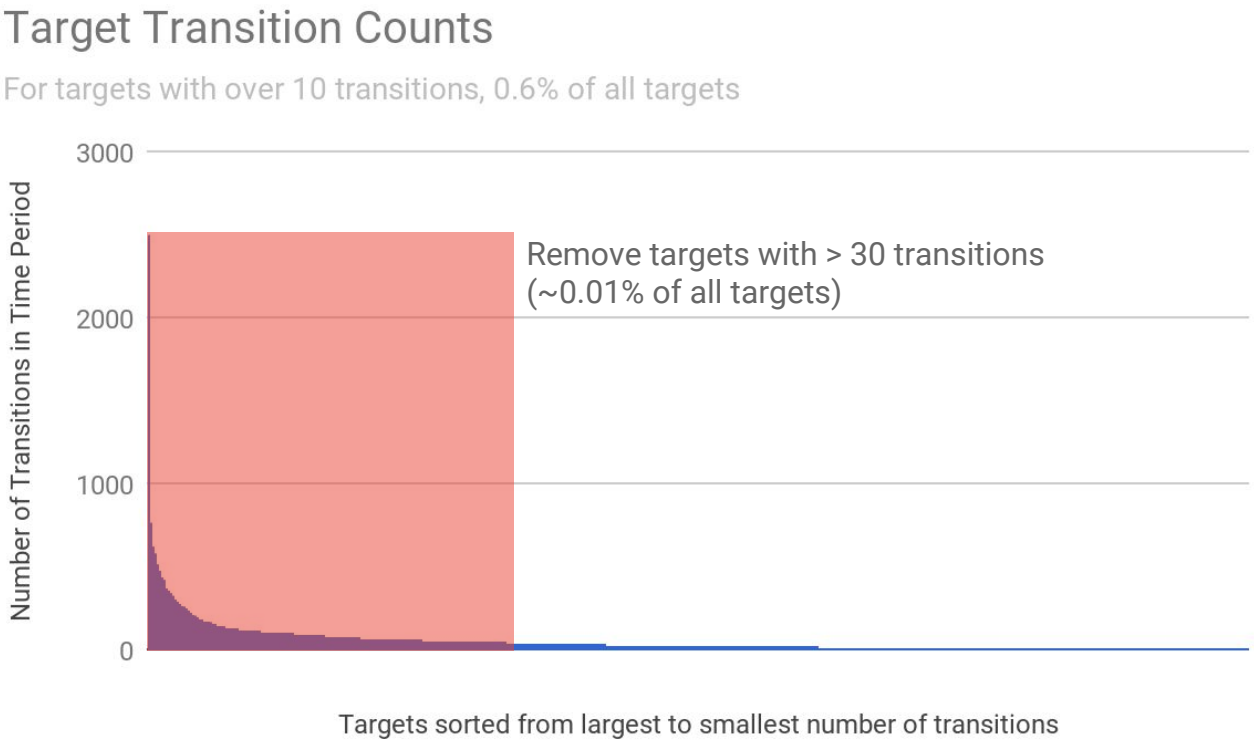
Can we skip targets safely?

- This information is used to determine whether skipping a target would have been safe
- All non-definitive pass or fail results treated as affected
- For a given test scheduling algorithm, we can evaluate it on a scale of skip rates from 0% (skips no targets) to 100% (skips all targets)

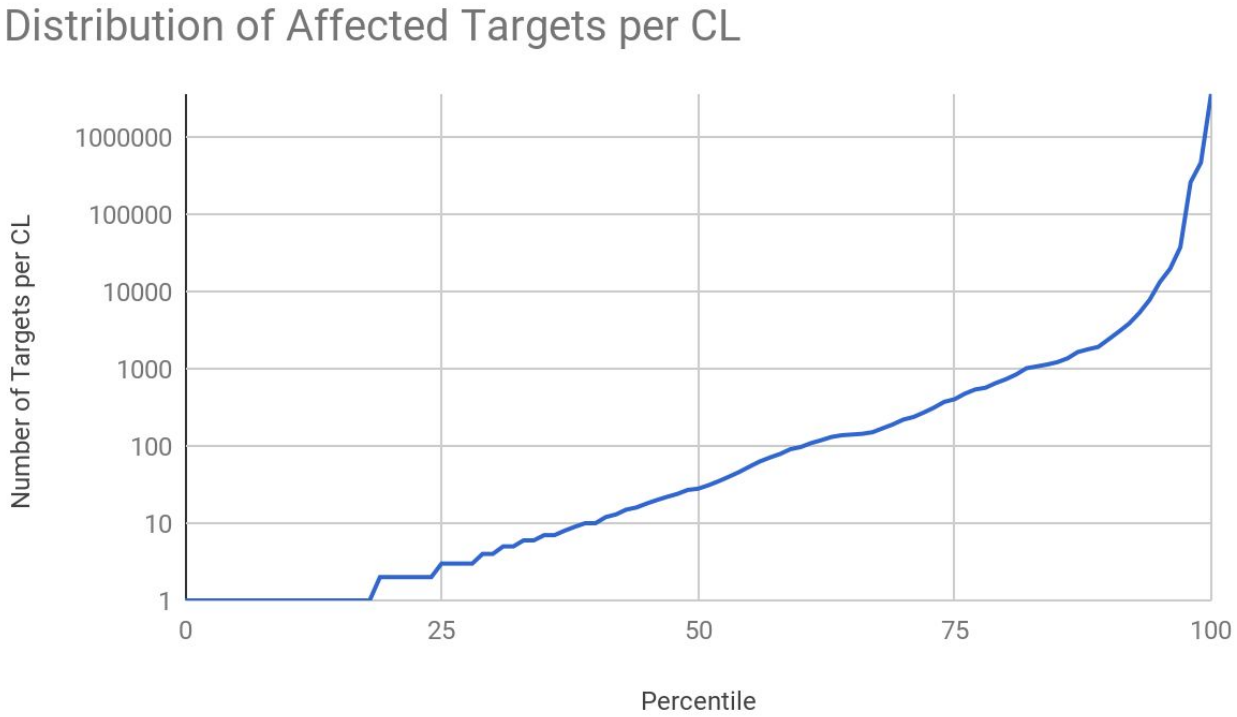
Input data

- Input taken from Google test target databases
- Used 3 methods to eliminate flakes from the data
  - Only take pass and fail results (not broken builds, tool failures etc.)
  - Removing target results identified as flaky by Google’s flakiness finder
  - Removing targets with over X transitions in the time period

Removing high transition count targets



Distribution of Targets per CL



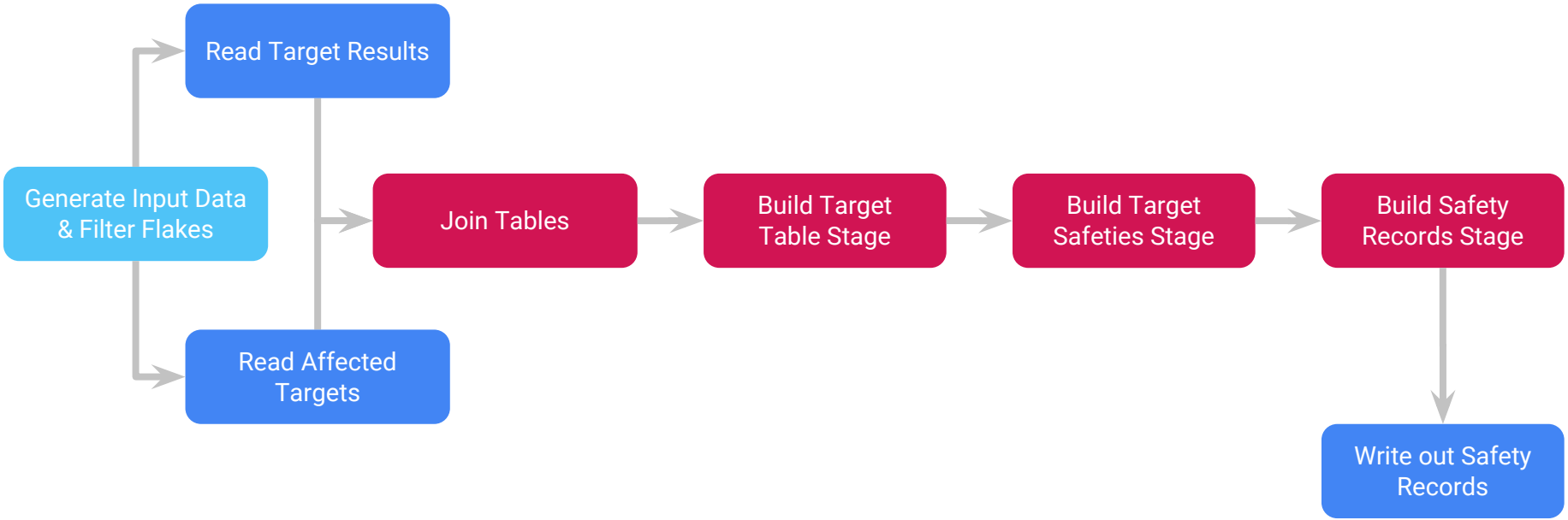
Stats:

- Median 38 tests!
- 90th percentile 2,604
- 95th percentile 4,702
- 99th percentile 55,730

Implementation: Safety Data Builder

This package creates safety data given the historical changelist data as input.

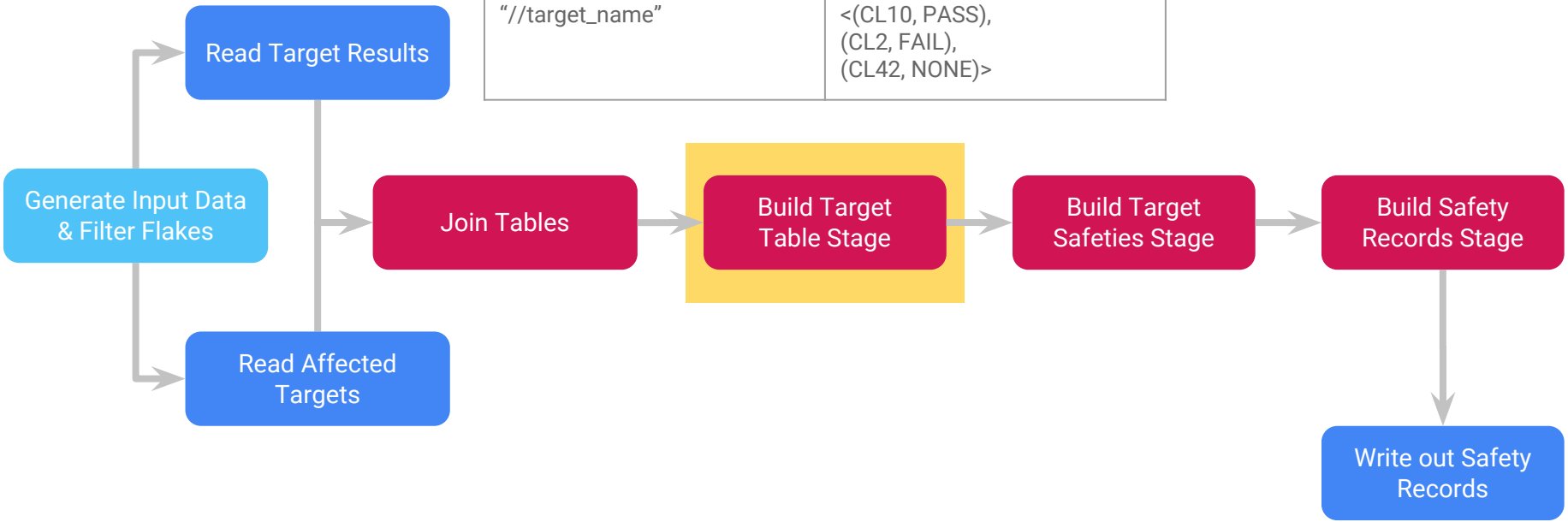
Pipeline



Pipeline

Output: Table<Target, Collection<ClAndResult>>

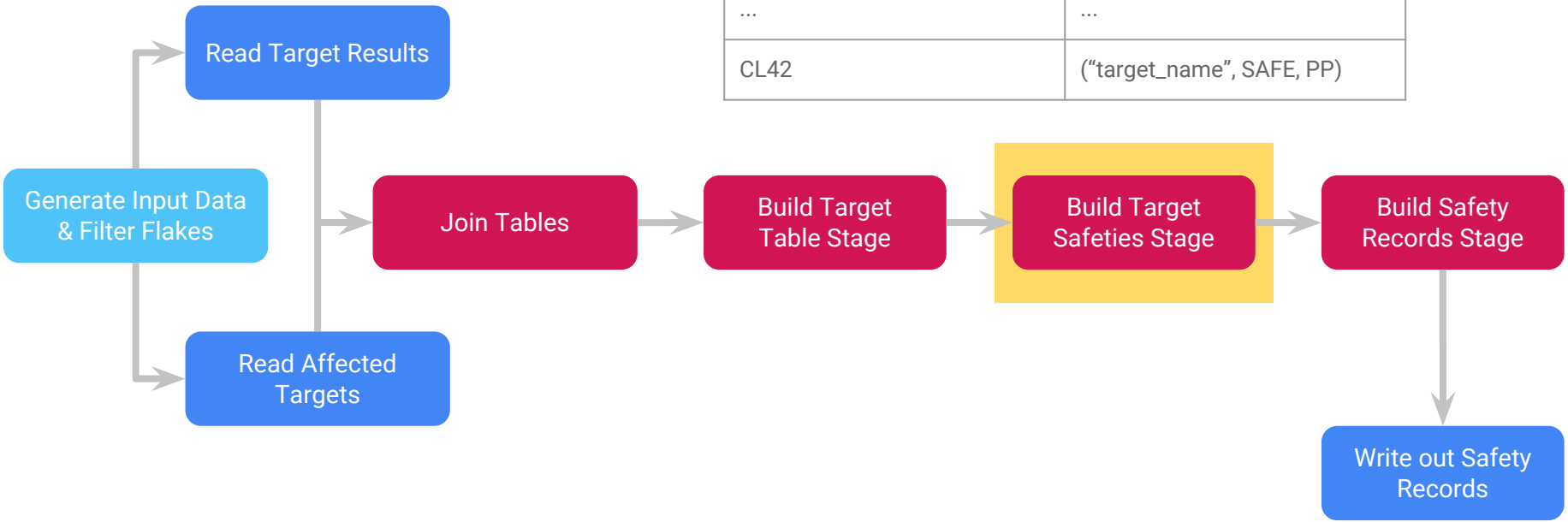
...	...
“//target_name”	<(CL10, PASS), (CL2, FAIL), (CL42, NONE)>



Pipeline

Output: Table<CL, TargetSafety>

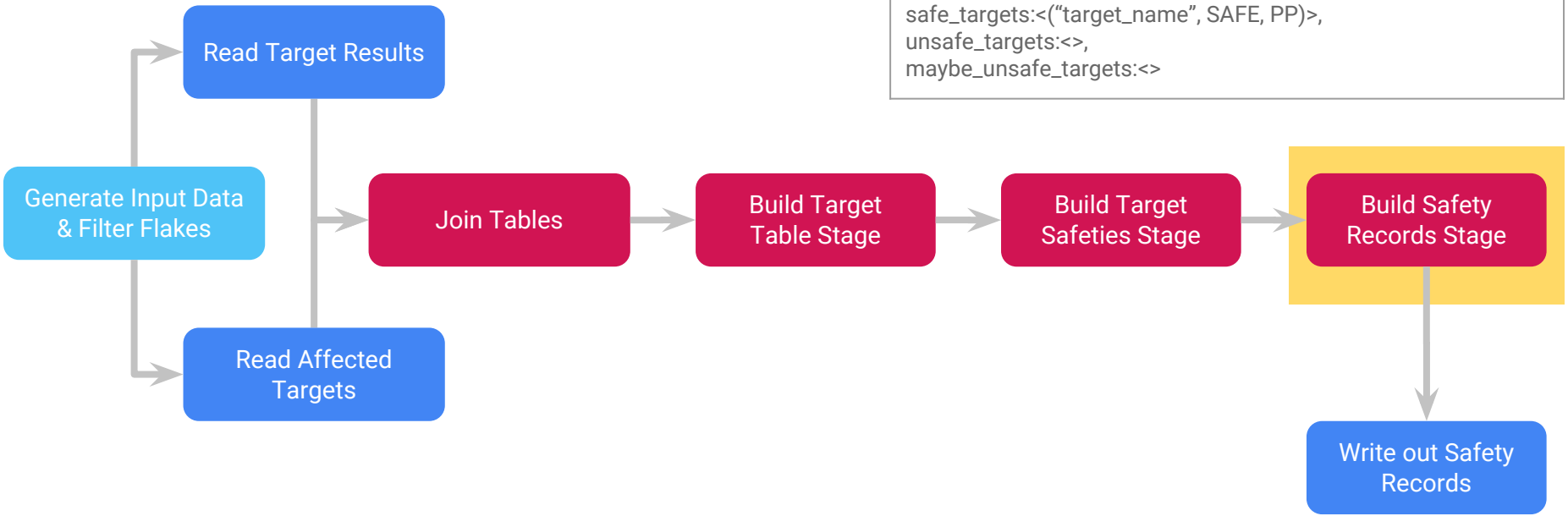
...	...
CL42	(“target_name”, SAFE, PP)



Pipeline

Output: Collection<SafetyRecord>

...
CL42, safe_targets:<(“target_name”, SAFE, PP)>, unsafe_targets:<>, maybe_unsafe_targets:<>

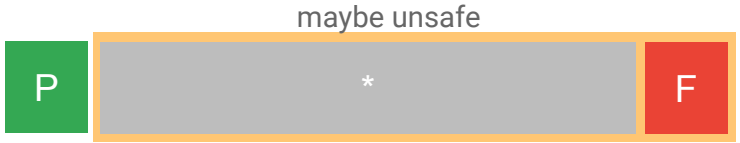


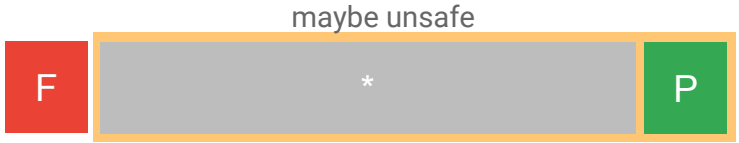
Safety Data Results

	Small Data Set	Large Data Set
Total CLs	10170	891,621
CLs which affected only safe to skip targets	96.4% (9801)	90.2% (804,160)
CLs which affected maybe unsafe to skip targets	3.4% (346)	8.3% (73,897)
CLs which affected unsafe to skip targets	0.2% (25)	1.5% (13,564)
Total target affecteds	428,938	15,931,019,923
Safe target affecteds	99.9% (428,547)	99.98% (15,927,853,638)
Maybe unsafe target affecteds	0.09% (365)	0.019% (3,054,667)
Unsafe target affecteds	0.01% (26)	0.0001% (111,618)

Culprit finding works!

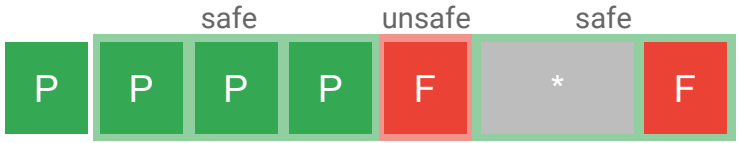
- When a P->F target transition is found with some number of affected CLs in between, **culprit finding** is applied: the target is rerun at the affected CLs to find exactly which CL caused the transition



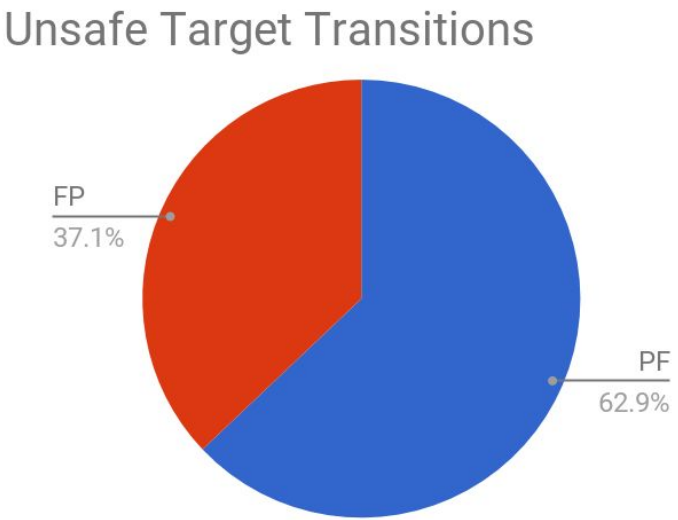


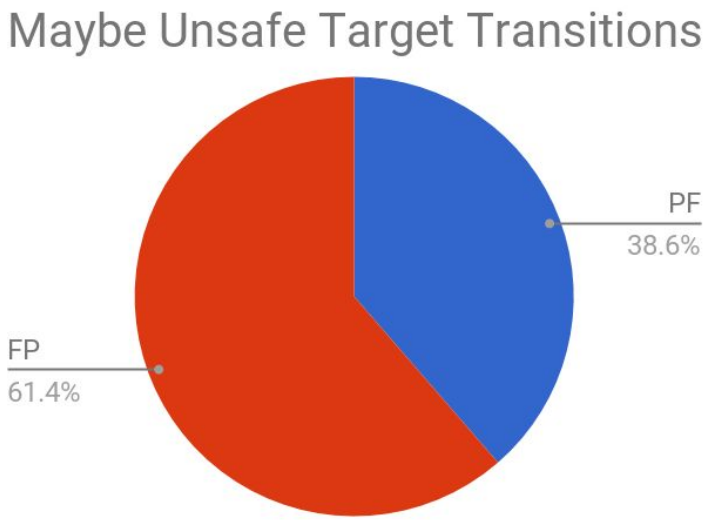
With culprit finding:

We don't do fix finding



Culprit finding works!





Implementation: Algorithm Evaluator

This package evaluates the safety of using an algorithm to select tests to skip for a changelist.

Evaluator Implementation

- For every changelist in the safety data, it will call an algorithm that is asked to skip a percentage of the changelist's affected targets
- Using the targets returned by the algorithm, determines if that selection was safe or not
  - Safe = no unsafe or maybe unsafe tests were skipped
  - Maybe unsafe = maybe unsafe tests were skipped but no unsafe tests
  - Unsafe = unsafe tests were skipped

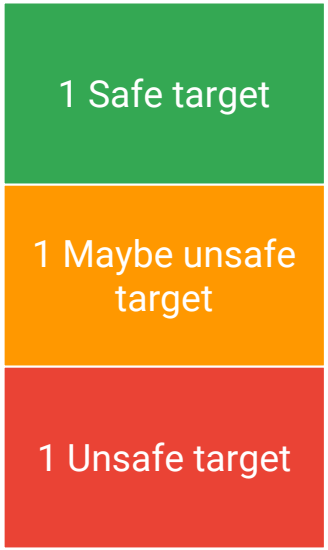
Algorithms

- Current algorithms are artificial algorithms which already know the safety of targets when choosing what to skip
- Algorithms are implementations of the interface TestSelectionAlgorithm which contains the method

```
ImmutableSet<Target> skipTargets(long cl, Iterable<Target> targets, int numToSkip)
```

Algorithms - **Random**

Changelist's Affected Targets

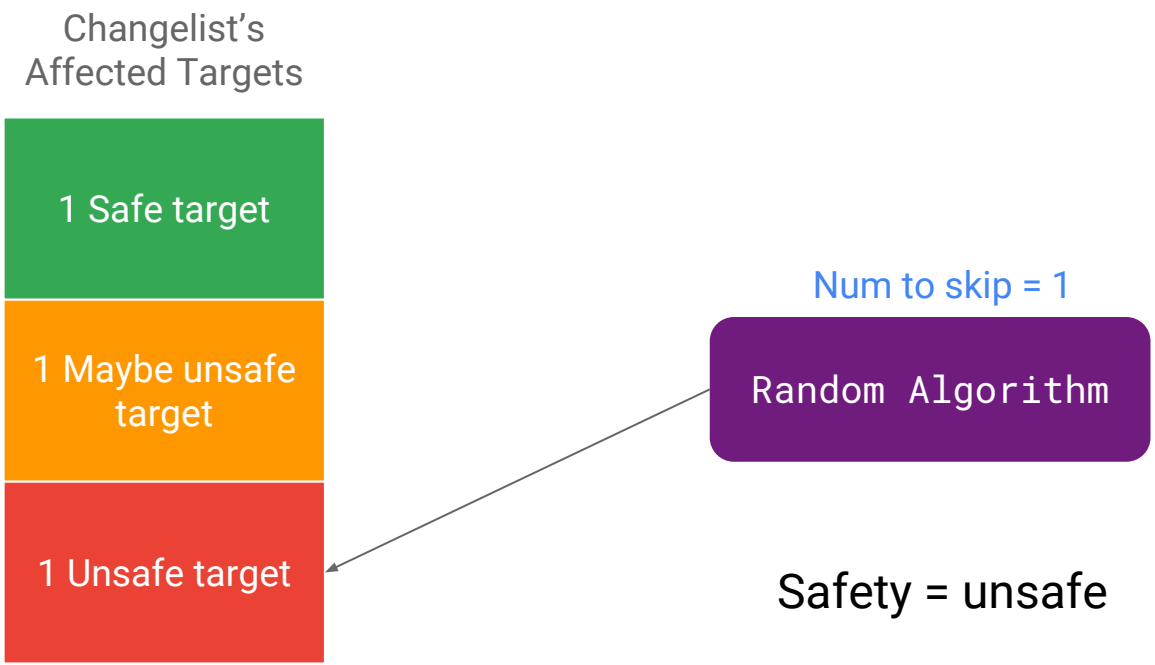


Num to skip = 0

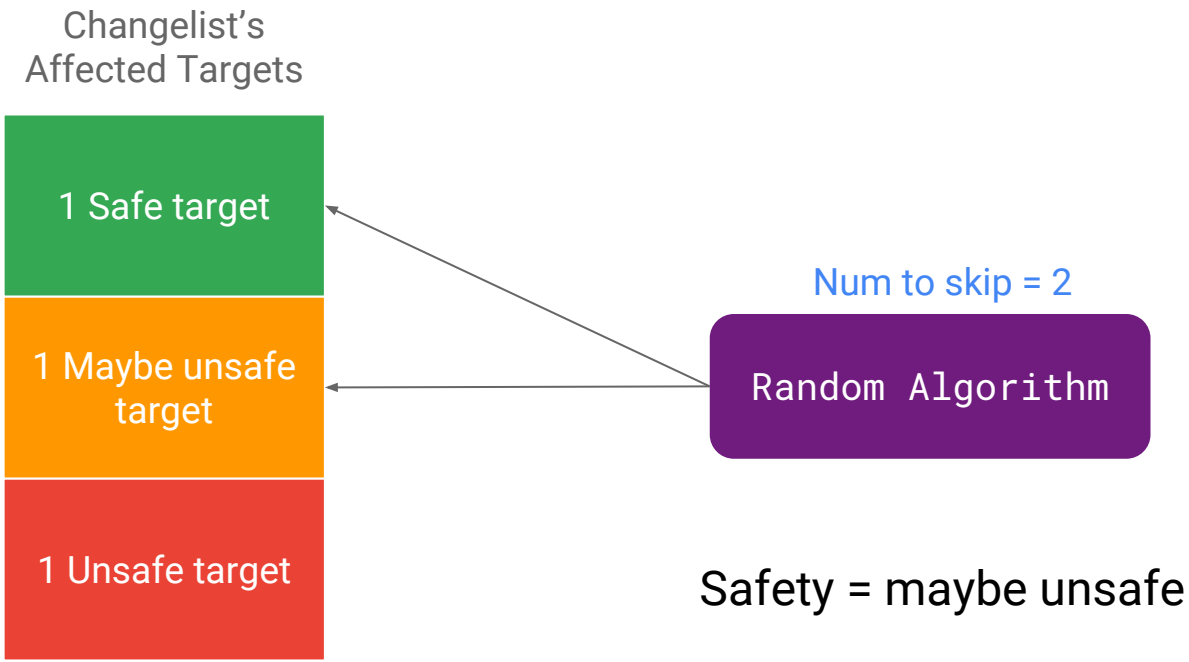
`Random Algorithm`

Safety = safe

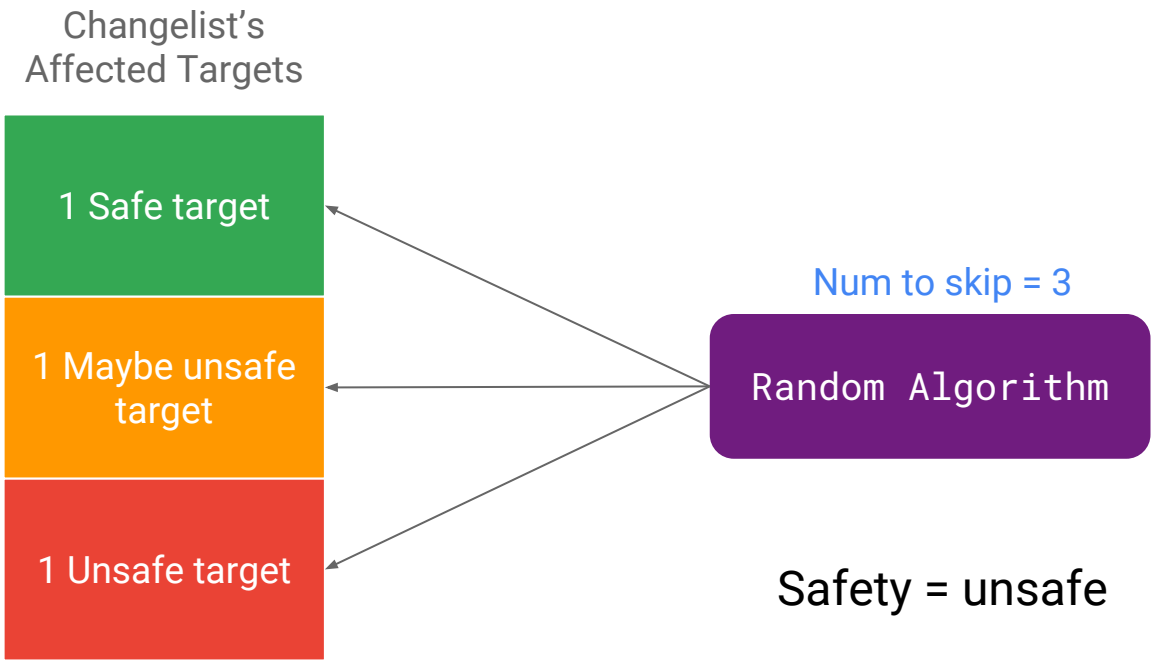
Algorithms - **Random**



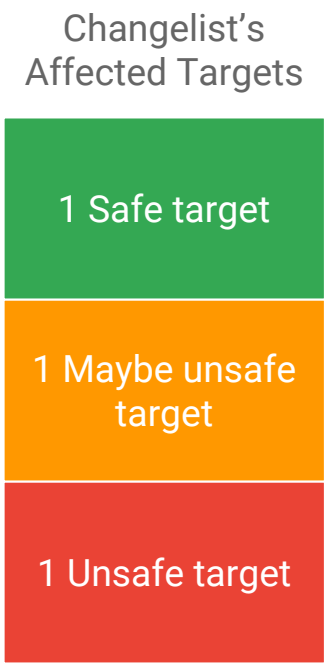
Algorithms - **Random**



Algorithms - **Random**



Algorithms - **Optimal**

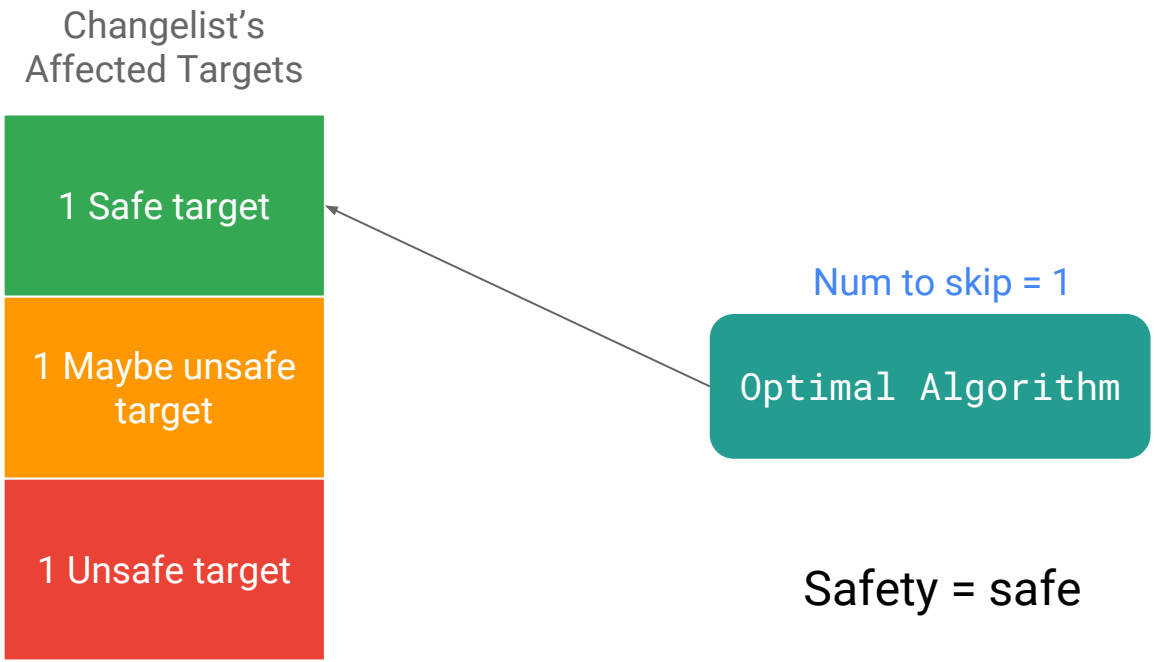


Num to skip = 0

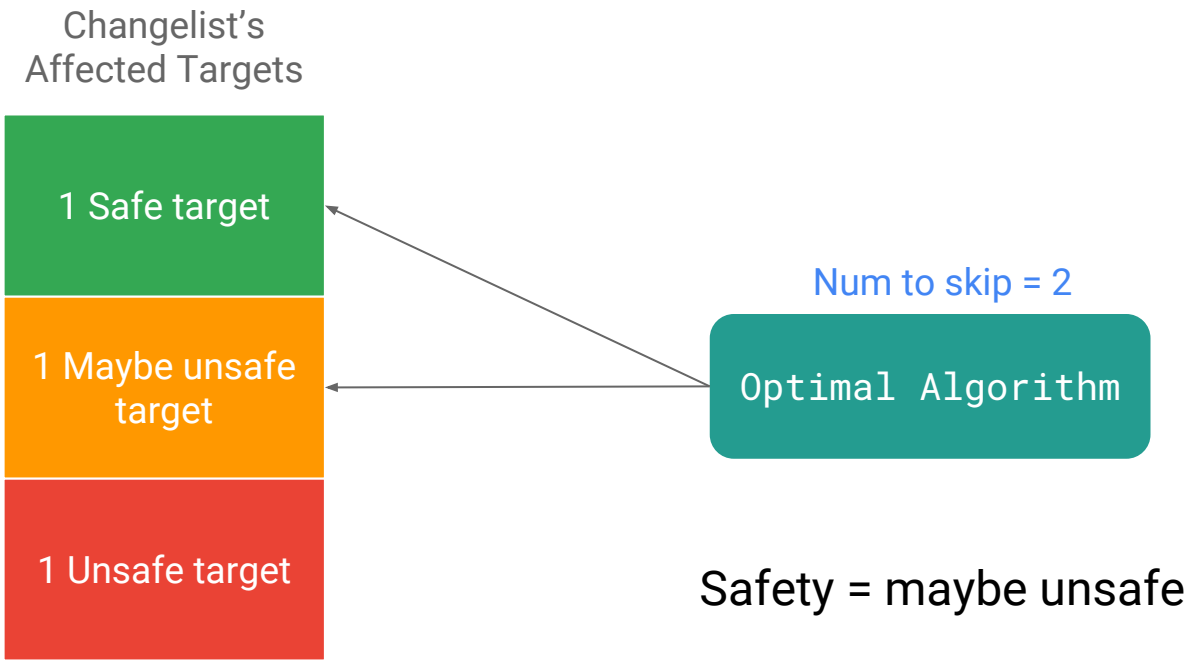
`Optimal Algorithm`

Safety = safe

Algorithms - **Optimal**

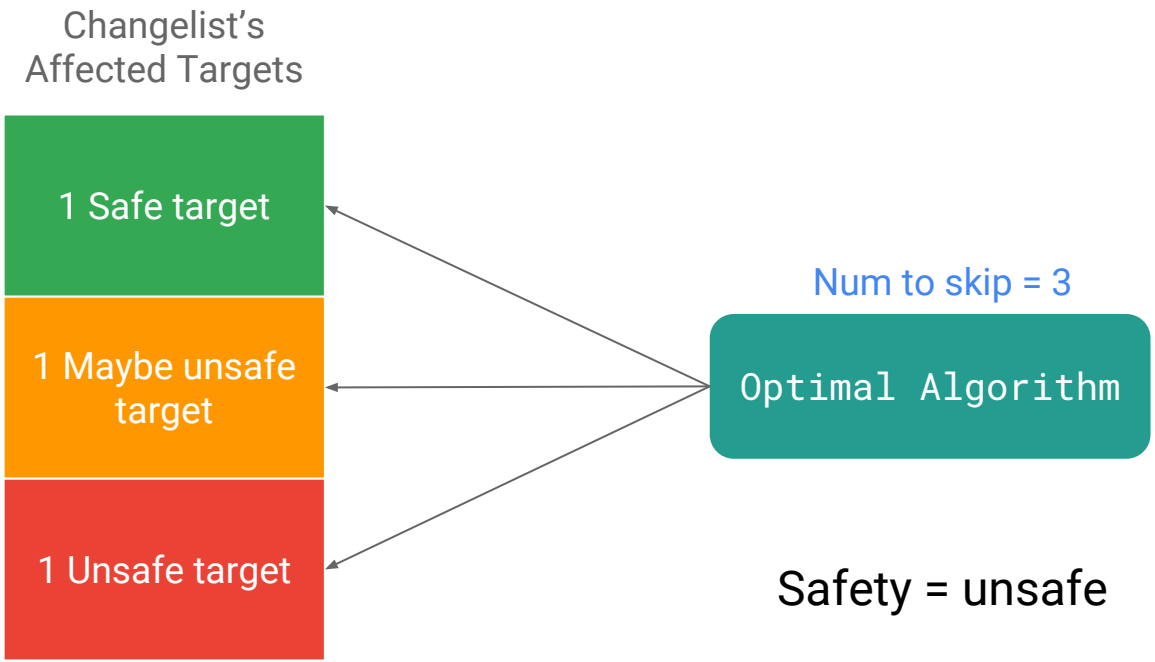


Algorithms - **Optimal**

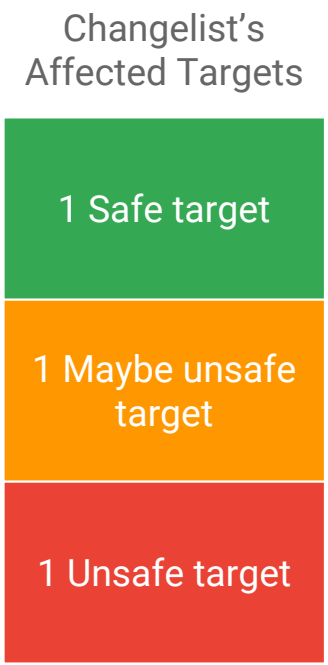


Safety = maybe unsafe

Algorithms - **Optimal**



Algorithms - **Pessimal**

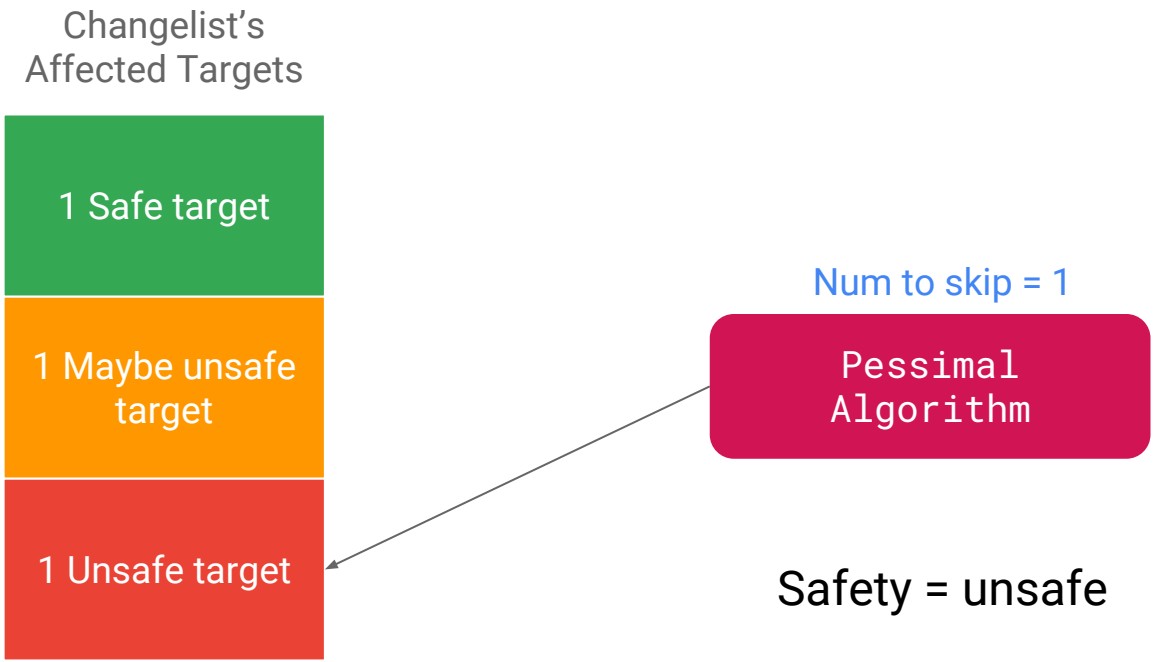


Num to skip = 0

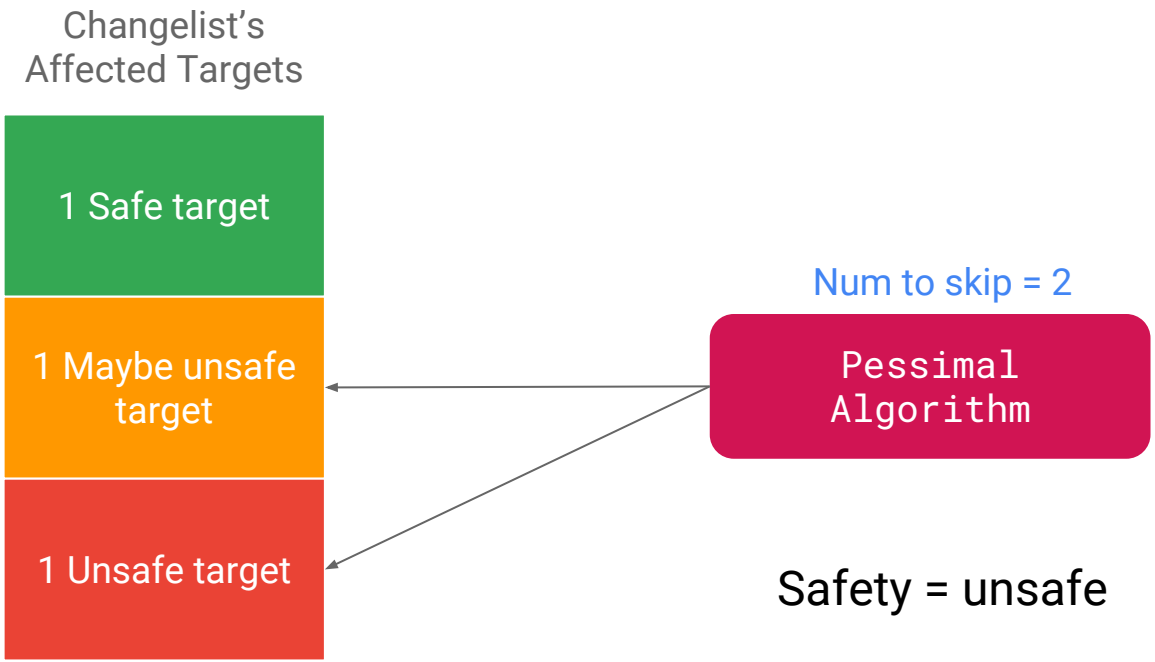
`Pessimal Algorithm`

Safety = safe

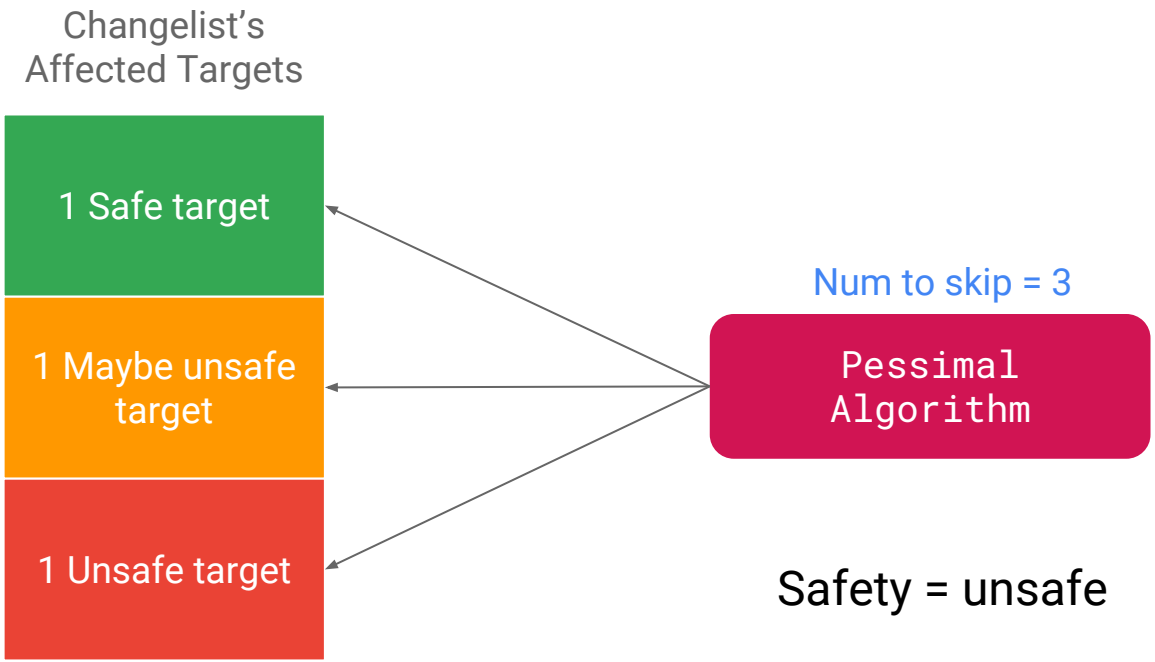
Algorithms - **Pessimal**



Algorithms - **Pessimal**



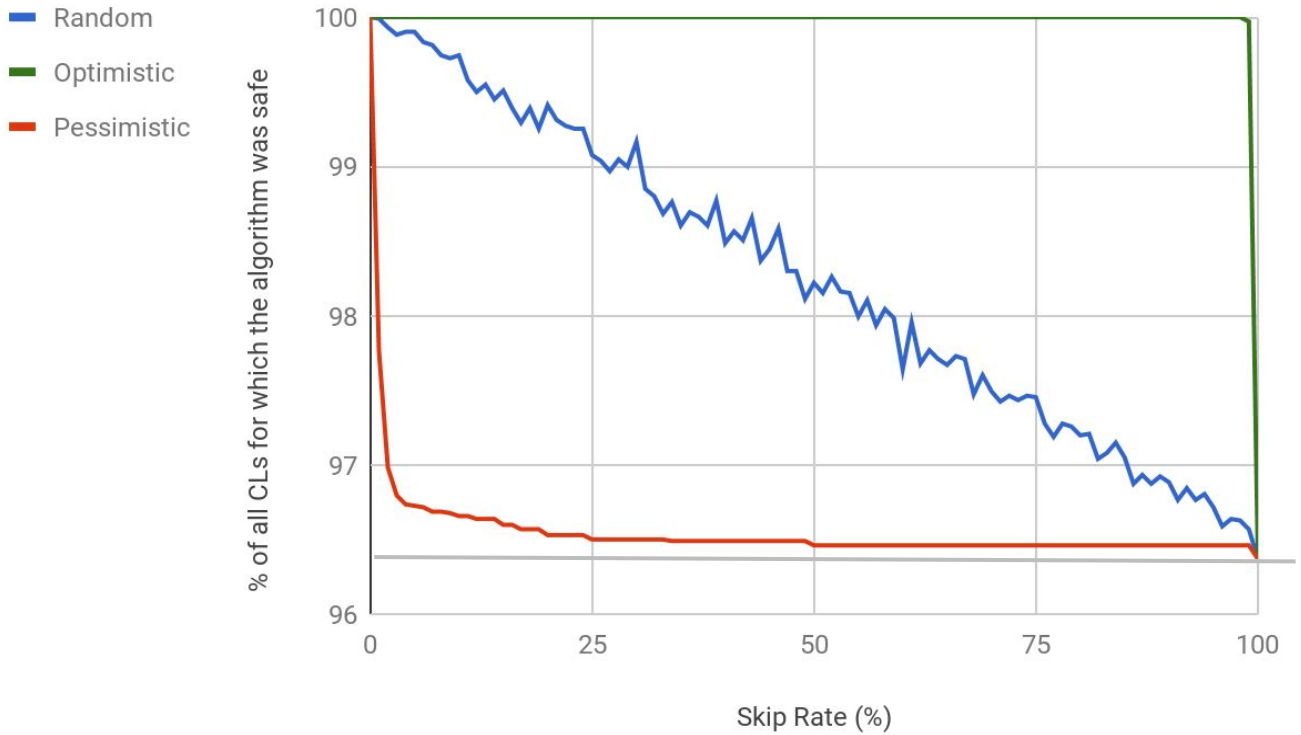
Algorithms - **Pessimal**



Pipeline performance

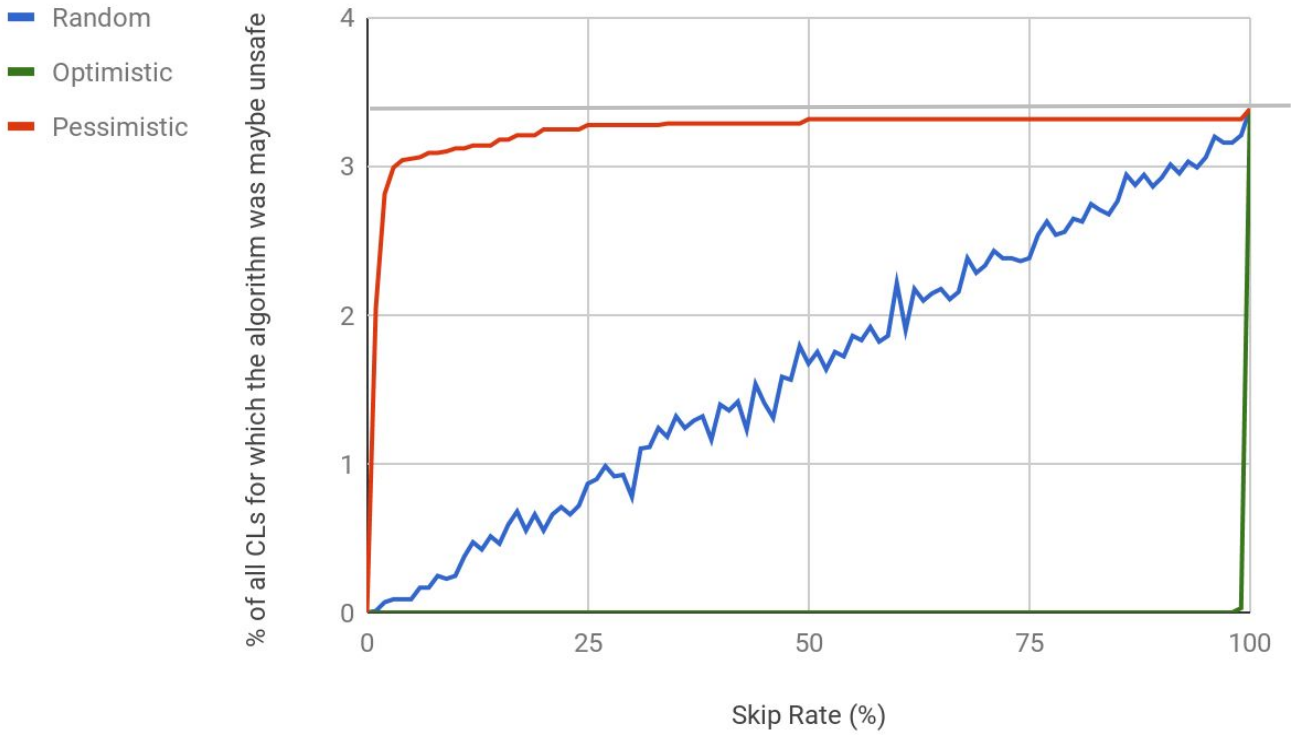
- Safety data builder ran in 35 mins
- Algorithm evaluator
  - Optimal ran in 2h 40m
  - Pessimal ran in 3h 5m
  - Random ran in 4h 40m

Small dataset results - Safe



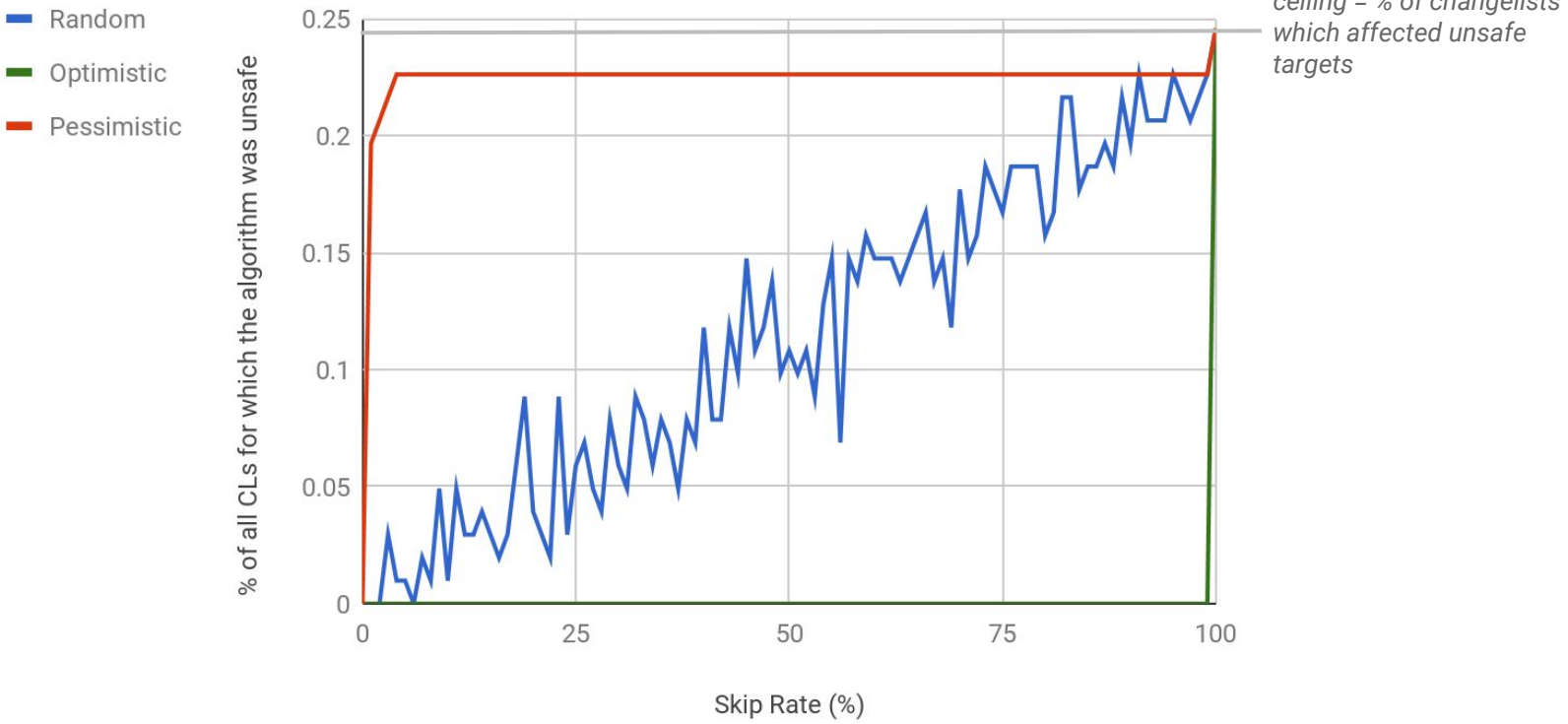
floor = % of changelists which only affected safe to skip targets

Small dataset results - Maybe Unsafe

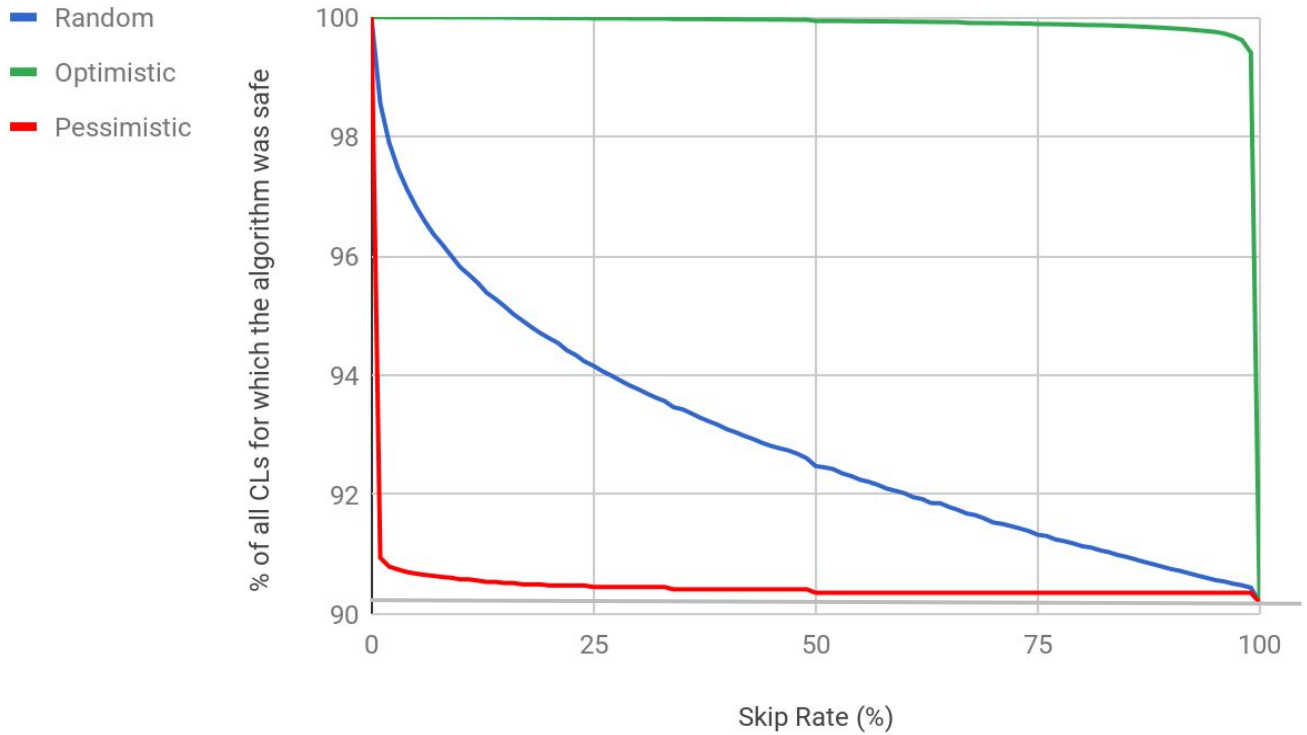


ceiling = % of changelists which affected maybe unsafe targets but no unsafe targets

Small dataset results - Unsafe



Large dataset results - Safe



floor = % of changelists which only affected safe to skip targets

Why is random a curve?

- Previously we had predicted a straight line for random
- Small data set has a straight line

Probability Distribution

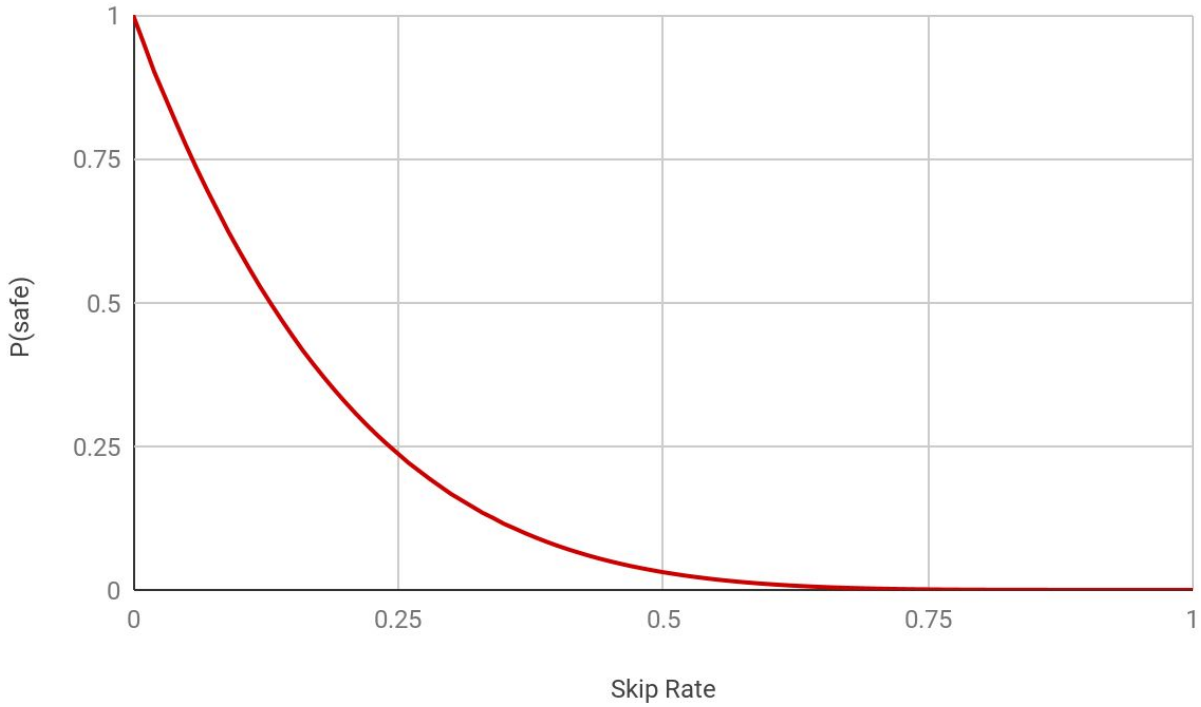
$$\binom{n}{k} = \text{number of ways to select k items from n total items}$$

$$\text{P(select k only safe targets)} = \frac{\text{number of ways to select k safe targets}}{\text{number of ways to select any k targets}}$$

$$= \frac{\binom{n}{Np}}{\binom{N}{Np}}$$

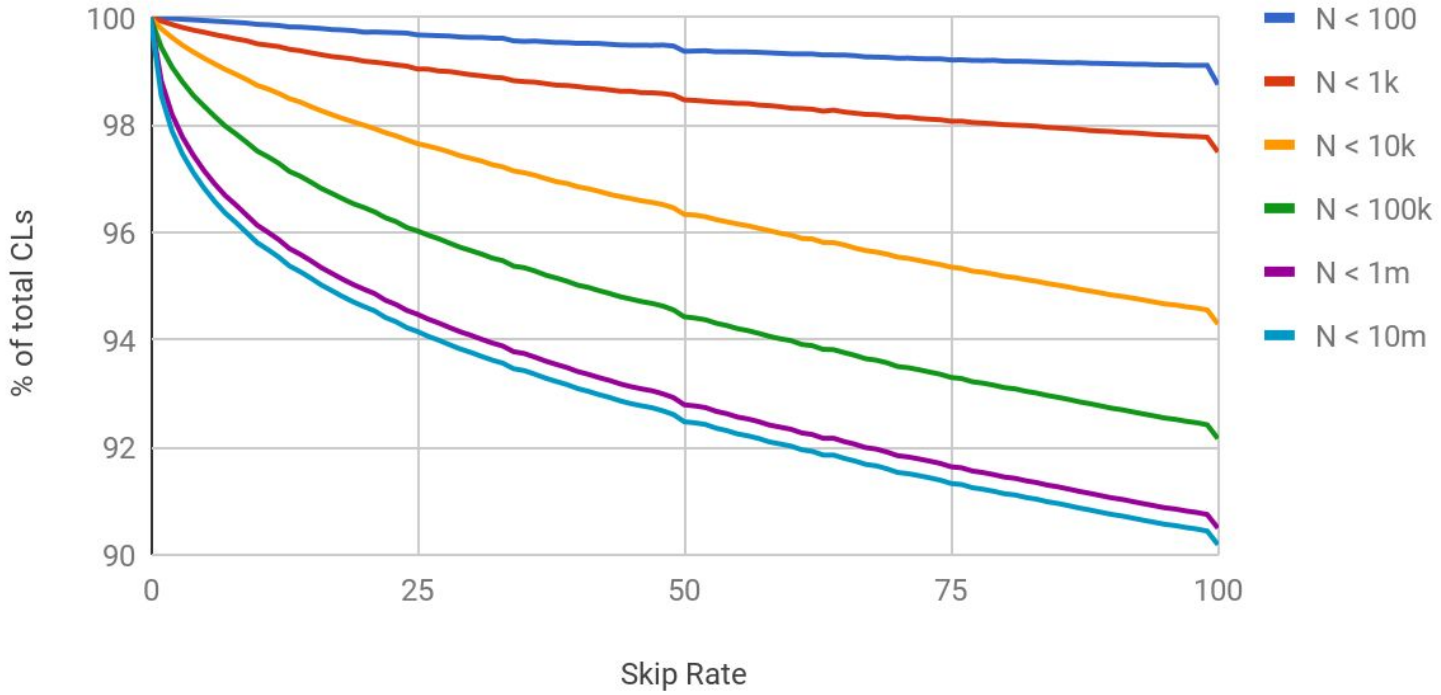
Where n = number of safe affected targets
N = total number of affected targets
p = % of targets being selected

Probability Distribution where N = 1000, n = 995

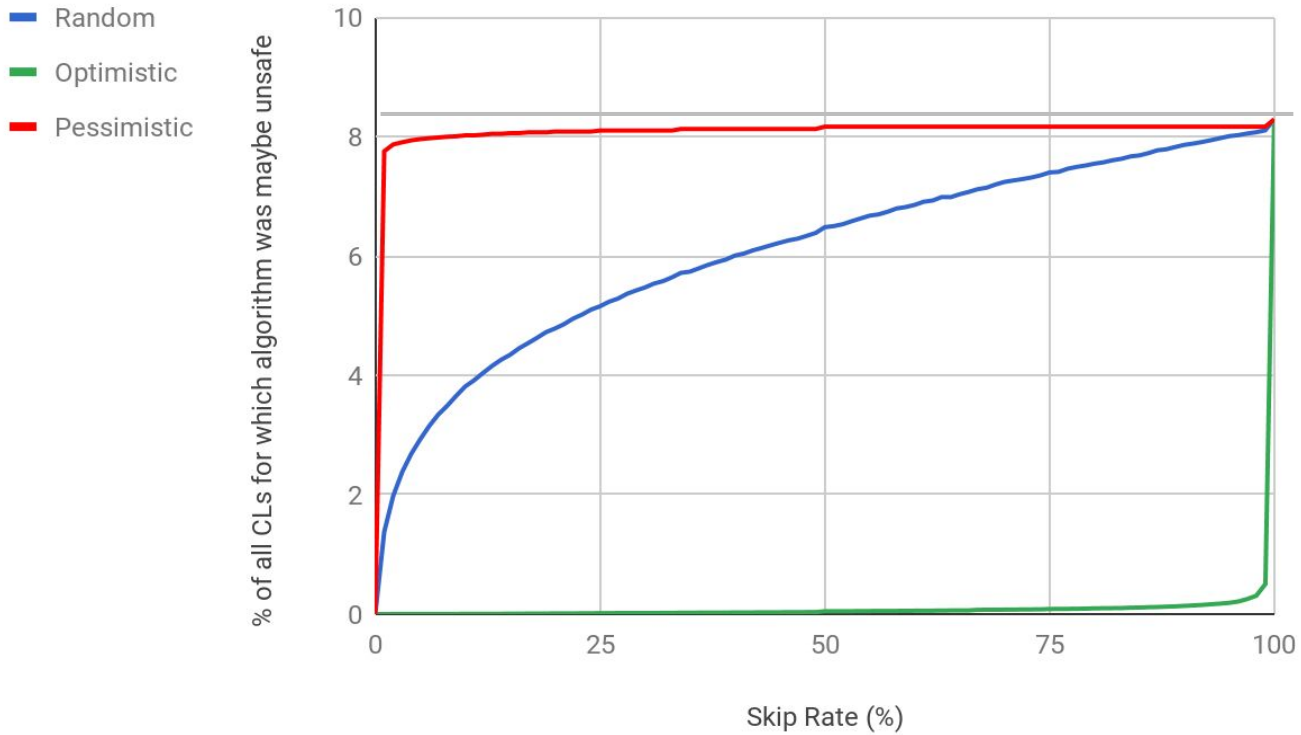


Random Algorithm Safe Changelists

where N = number of CL's affected targets

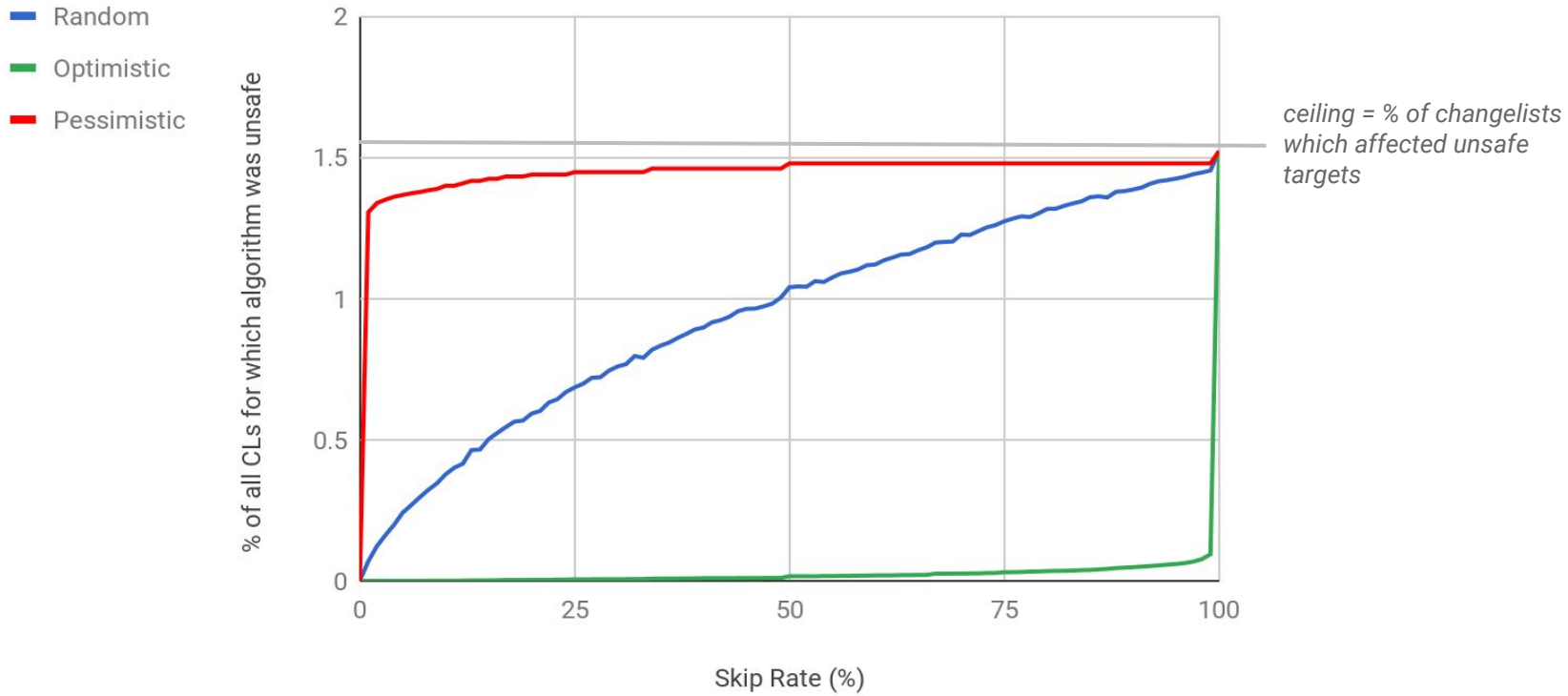


Large dataset results - Maybe Unsafe



ceiling = % of changelists which affected maybe unsafe targets but no unsafe targets

Large dataset results - Unsafe



Conclusions

- The project was completed!
- We now have an offline method to evaluate test scheduling algorithms and a baseline for future comparison

Continuing the project

- Better flake exclusion
  - Filter using ratio transitions:results
  - Find the point where Google’s flake detection software doesn’t identify the target as flaky
- Rerunning Elbaum experiments
  - An algorithm which prioritizes targets based on the number of transitions in some previous window of time
- Evaluating Efficacy machine learning model

Questions?