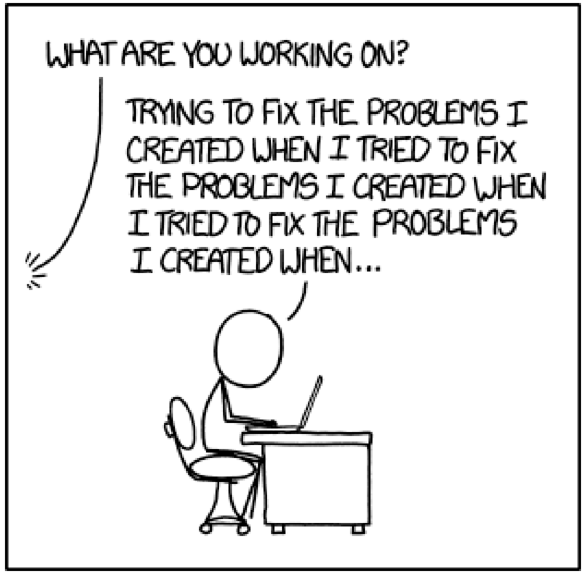
BugSleuth

Ashish Ramrakhiani

Nat Bourassa

Sunayana Sanam

Source: https://xkcd.com/1739/



Problem

- Software practitioners spend a lot of time debugging
- Advent of Automated Fault localization (FL) techniques
- None of the techniques are best, no one-size-fits-all solution
- Research shows combining FL techniques using machine learning improves FL performance [1]
- Supervised learning – a huge labelled dataset required
- Unsupervised techniques – SBIR[2] using RAFL[2] to improve APR – not evaluated for effective FL for practitioners

[1] D.Zou, J.Liang,Y.Xiong,M.D.Ernst, andL.Zhang.Anempirical studyof fault localizationfamiliesandtheircombinations. TSE,2019.
[2] M. Motwani and Y. Brun, “Better automatic program repair by using bug reports and tests together,” in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). IEEE, 2023
RAFL – Rank Aggregation Fault localization

Solution

- We propose, BugSleuth.
- An unsupervised fault localization technique that combines multiple ranked lists of suspicious statements
- Uses Rank Aggregation algorithms -
  a) Genetic Algorithm
  b) Cross Entropy Monte Carlo Algorithm
- Provides an optimal ranked list of suspicious statements thereby improving a practitioner's productivity by detecting bugs in their code

Research questions

RQ1: How effective are rank aggregation algorithms for localizing defects

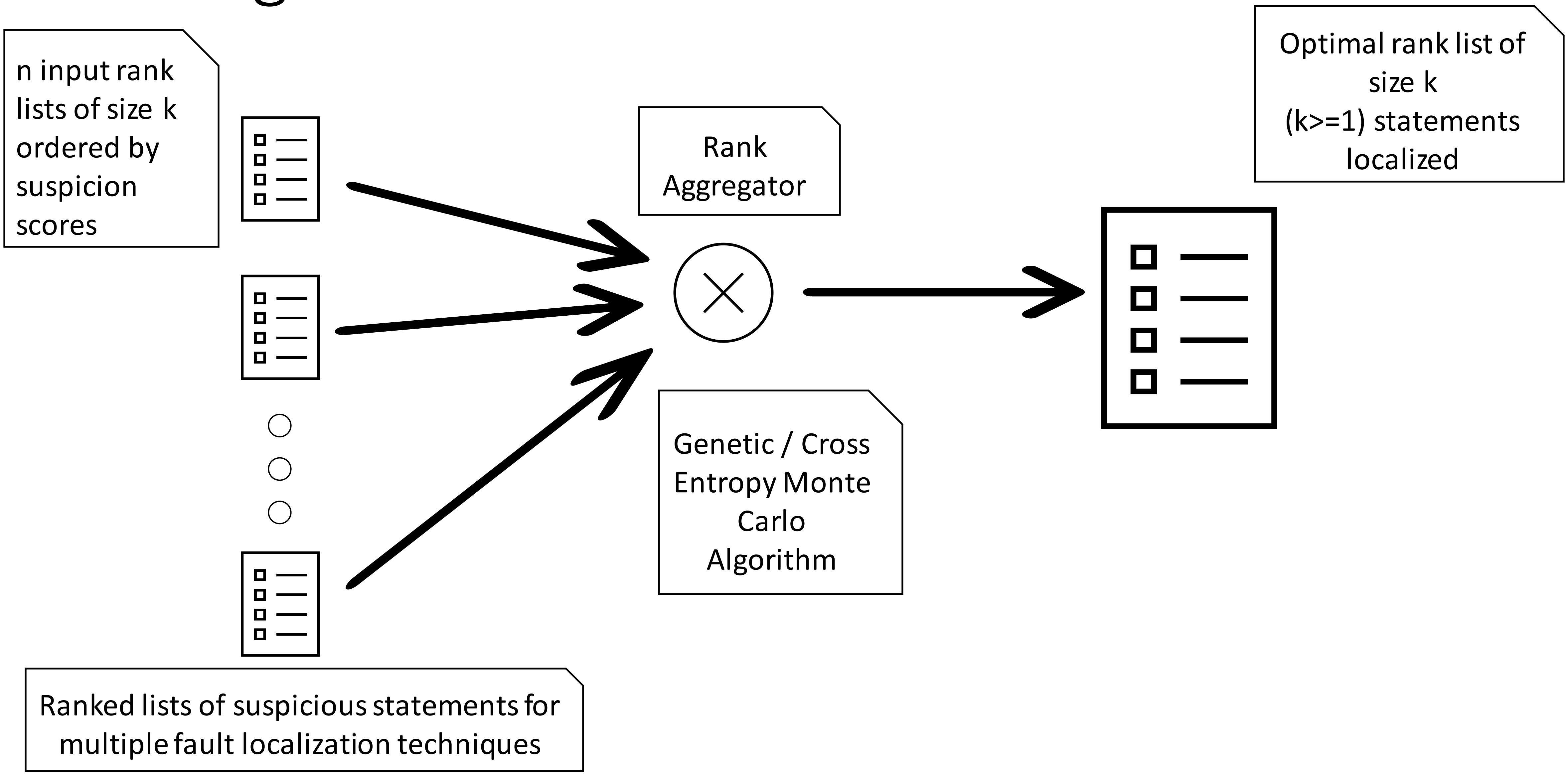
(A) Genetic Algorithm

(B) Cross Entropy Monte Carlo Algorithm

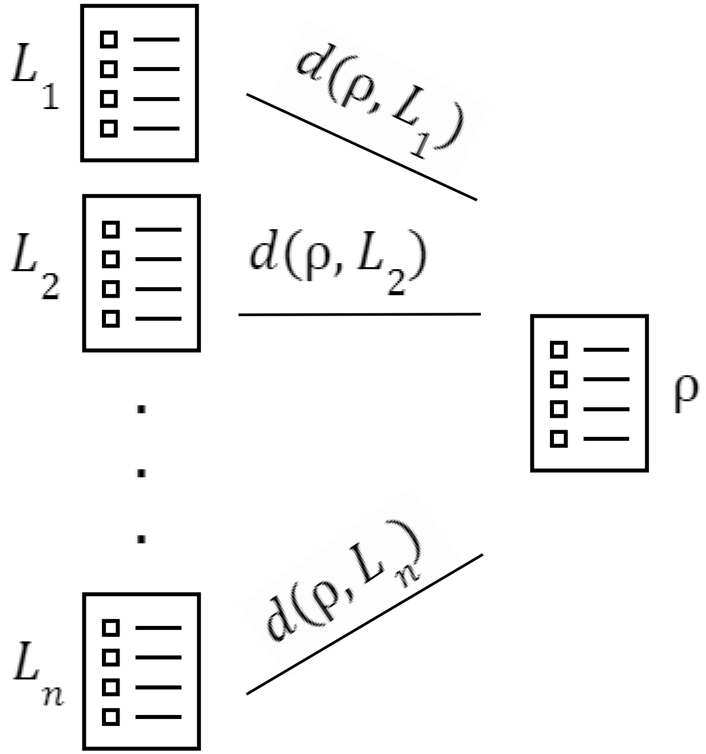
RQ2: How efficient are rank aggregation algorithms for fault localization

RQ3: How do these rank aggregation algorithms compare against state-of-the-art fault localization techniques

Design


n input rank lists of size k ordered by suspicion scores
Rank Aggregator
Genetic / Cross Entropy Monte Carlo Algorithm
Optimal rank list of size k (k>=1) statements localized
Ranked lists of suspicious statements for multiple fault localization techniques

Rank Aggregation as an Optimization problem



$$f(\rho) = \sum_{i=1}^{n} d(\rho, L_i)$$

- f() - objective function
- Rho – combined list
- Li – input lists
- d() - distance function

Optimization:
Choose f(rho) which has smallest value to obtain an optimal list which is closest to all input lists

Algorithms

- Genetic Algorithm (using jenetics library)
  - Selector – Roulette Wheel selection
  - Crossover strategy – Partially matched Crossover
  - Mutation strategy – Swap Mutation
  - Fine tune – Crossover probability, Mutation probability, Convergence iterations, Population size
- Cross-Entropy Monte Carlo Algorithm
  - Own implementation
  - Number of sampled matrices
  - Quantile index – how many sample matrices should be considered
- Number of iterations before convergence
- Distance function
  - Spearman footrule distance.
- Finetuning Dataset
  - 112 Defects of Jackson-Databind project in Defects4J

Evaluation

Dataset

Defects4J (v2.0) benchmark to evaluate our FL technique

Defects4J consists of 835 reproducible defects from 17 large open-source Java projects.

Each defect comes with

(1) one buggy and one developer-repaired version of the project code

(2) a set of developer written tests, all of which pass on the developer-repaired version and at least one of which evidences the defect by failing on the buggy version and

(3) defect information

Evaluation

Metrics

We will use two metrics, common to FL evaluations

Top-N is the number of defects localized in the top-n ranked statements

- It tells us how efficient our FL technique is in localizing bugs
- Top-1, Top-3 and Top-5 utilized

EXAM is the fraction of ranked statements one must inspect before finding a buggy statement.

- EXAM tells us how high the buggy statements are ranked
- saves practitioner's time and resources otherwise spent on manual bug detection

Thank You