**PROBLEM**

Mutation testing can be too slow for practical use because of the large number of mutants.

---

**NEW IDEA**

A reduction method that tries to select the most “interesting” mutants; that is, mutants that will survive mutation testing.

---

**RESULTS**

74 percent of the time, our method chose a greater number of interesting mutants than random reduction did.

---

### The old solution: reduce the mutants at random.

**Source files**

```
  □ □ □ □ □ □
```

**Mutants**

```
  □ □ □ □ □ □
```

**Random reduction**

```
  □ □ □ □ □ □
```

**Two mutants killed by tests**

```
  □ □ □ □ □ □
```

---

The surviving mutants indicate possible weaknesses in the test suite. They are therefore more interesting to developers than the killed mutants. Our goal was to reduce the number of mutants in a way that preserves these interesting ones.

---

We reduced the set of mutants to the **faulty-file mutants**, or mutants made by altering a file that has contained many faults in the past. We hypothesized that faulty file mutants are more likely to survive; that is, they are the more interesting mutants.

**Source files**

```
  □ □ □ □ □ □
```

**Mutants**

```
  □ □ □ □ □ □
```

**Non-random reduction**

```
  □ □ □ □ □ □
```

**One mutant killed by tests**

```
  □ □ □ □ □ □
```

---

We ran 6000 test suites on Apache POI’s faulty file mutants and on an equal number of randomly selected mutants. Our hypothesis was correct, but the results varied with suite size, ranging from 59% for size 100 suites to 97% for size 1000 suites.

---

**Comments**