Integrated Energy-Directed Test Suite Optimization

Ding Li[1], Yuchen Jin[1], Cagri Sahin[2], James Clause[2], William G. J. Halfond[1]

1. Department of Computer Science
University of Southern California

2. Computer and Information Sciences Department
University of Delaware

This work was supported in part by the National Science Foundation under Grant No. CCF-1321141 to the University of Southern California.

USC Viterbi
School of Engineering

A Background Story

Changing batteries is expensive

Battery supported

Need in situ testing

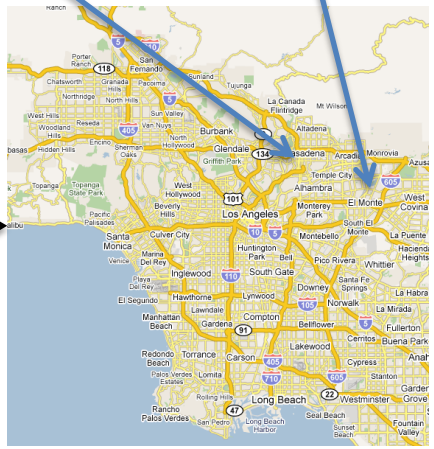
On dams

No stable power supply

In mountains

Alice

Earthquake monitor



LA metro area

USC Viterbi School of Engineering

Alice's Problem

- In situ testing is required
  - But it consumes precious battery energy
- Batteries have to be changed frequently
  - But it is difficult to change them

How to reduce the energy usage of an in situ test suite but still maintain the same effectiveness?

USC Viterbi School of Engineering

Test Suite Minimization

Coverage	Test Cases t_1	t_2	t_3	t_4	t_5
1. public void updateData(Location prior) {	✓	✓	✓	✓	✓
2. Location current = LocationManager.getLocation();	✓	✓	✓	✓	✓
3. SensorData data = getCurrentData();	✓	✓	✓	✓	✓
4. if (prior.distanceTo(current) > MAX_DISTANCE) {	✓	✓	✓	✓	✓
5. data = Sensors.getData(current);		✓		✓	
6. }	✓	✓	✓	✓	✓
7. if (OutOfDate(data)) {	✓	✓	✓	✓	✓
8. data = Sensors.getData(current)			✓		✓
9. }	✓	✓	✓	✓	✓
10. if (TimePassed() > MAX_TIME) {	✓	✓	✓	✓	✓
11. resendMessage(data);	✓			✓	✓
12. }	✓	✓	✓	✓	✓
13. }	✓	✓	✓	✓	✓
Energy Consumption (mJ)	0.5	0.7	0.2	1.3	1.5

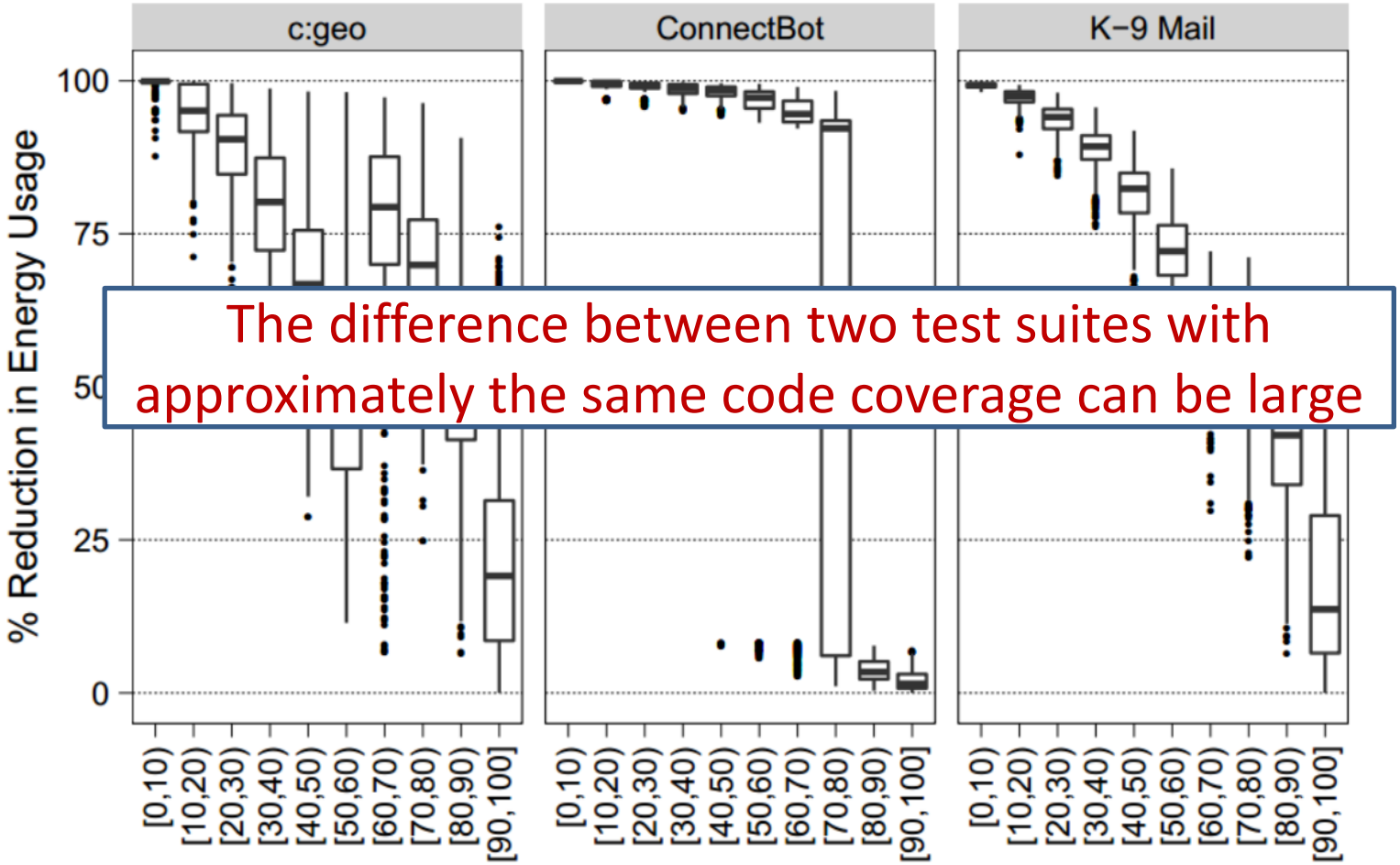
1.4 mJ 2.8 mJ

USC Viterbi School of Engineering

Current Solution

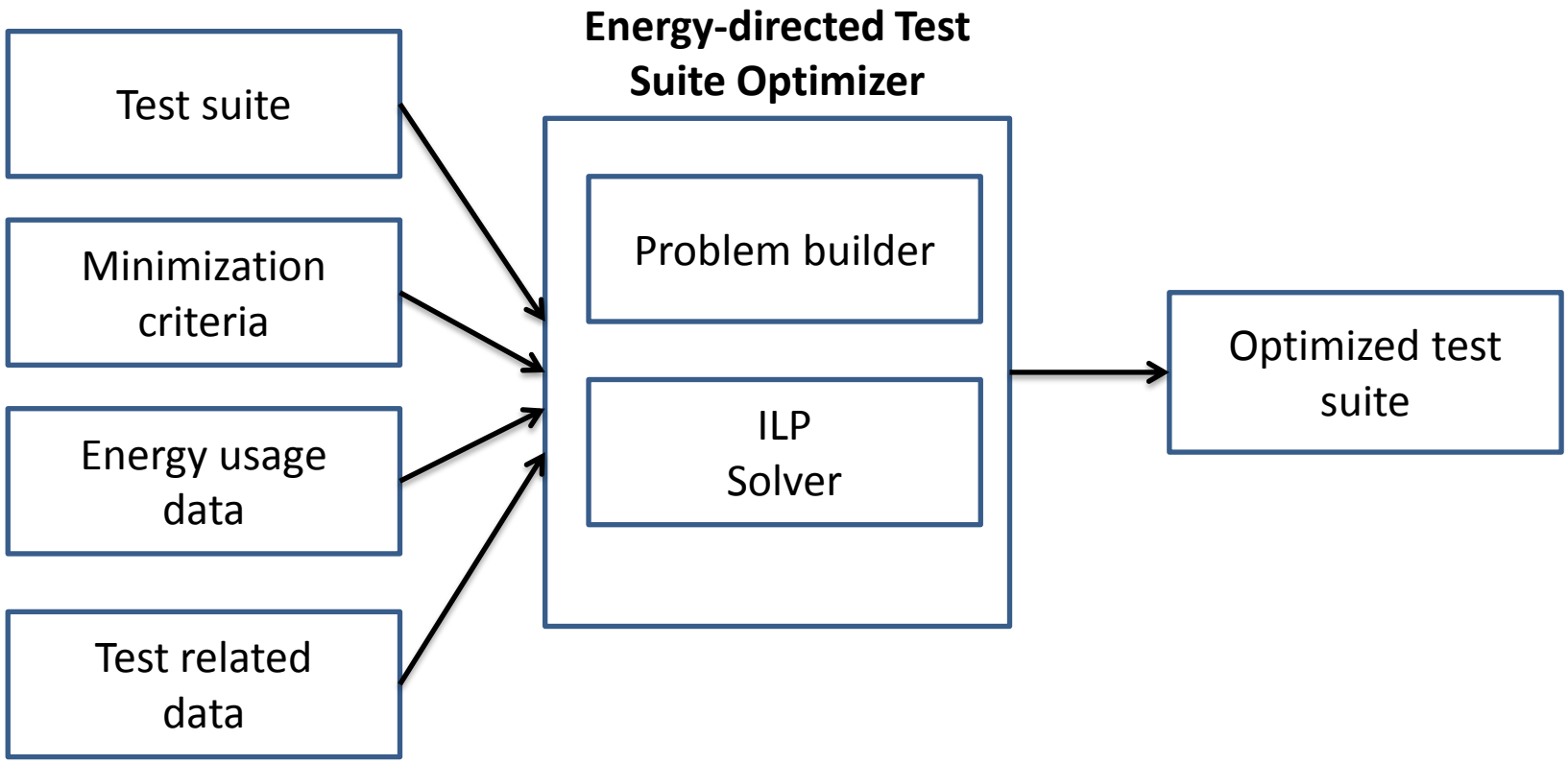
- Energy is not considered [G. Rothermel et al.] [S. Yoo et al.]
  - Optimize the size
  - Optimize the execution time
- These methods may not generate the optimal solution
  - On average, the least energy expensive test suite consumes ≈18 % less energy than the smallest test suite, despite being ≈11 % larger
  - Optimization of time also generates suboptimal solutions with respect to energy

Motivation: Variance in Energy of Test Suites

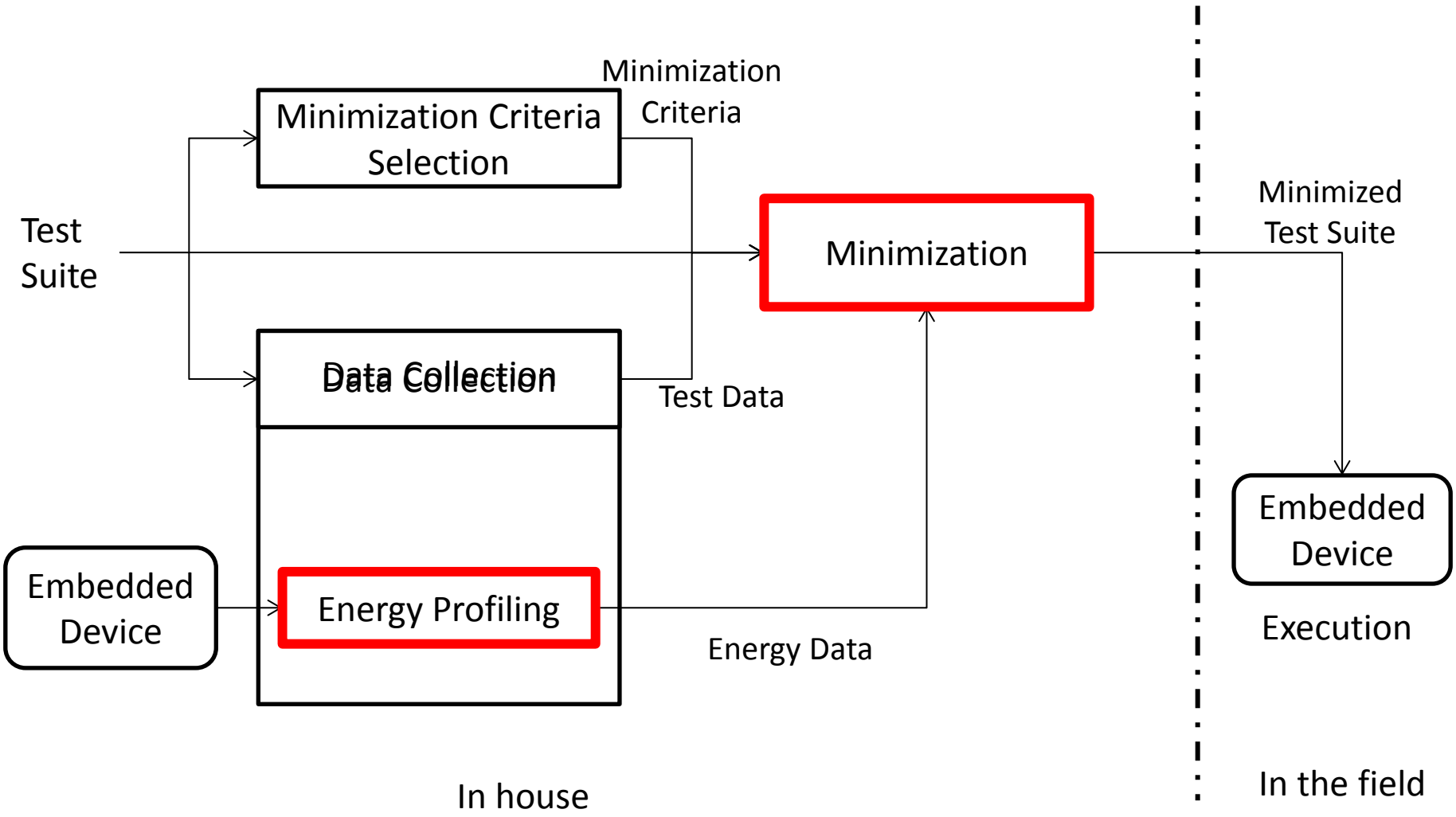


The difference between two test suites with approximately the same code coverage can be large

Our Solution: EDTSO



The Minimization Workflow



EDTSO: Energy Profiling

1. Instrument test cases to collect start and end timestamps
   – Can be fully automated for JUnit or TestNG
2. Execute the test suits on an Energy Measurement Platform (EMP)
   – Many ways to implement EMP [Singh et al.] [Hao et al.]
3. Map the logged time stamps to the energy measurements

EDTSO: Energy Minimization

- Convert the energy minimization problem to an ILP problem [Hsu et al.]
  - Each test case is represented as a binary variable
  - Requirements are represented as constraints
  - Energy is coded as minimization target
  - Generate a set of values of all binary variables that meets all constraints but has minimal energy consumption

EDTSO: Energy Minimization

Coverage	Test Cases				
	t_1	t_2	t_3	t_4	t_5
1. public void updateData(Location prior) {	✓	✓	✓	✓	✓
2. Location current = LocationManager.getLocation();	✓	✓	✓	✓	✓
3. SensorData data = getCurrentData();	✓	✓	✓	✓	✓
4. if (prior.distanceTo(current) > MAX_DISTANCE) {	✓	✓	✓	✓	✓
5. data = Sensors.getData(current);		✓		✓	
6. }	✓	✓	✓	✓	✓
7. if (OutOfDate(data)) {	✓	✓	✓	✓	✓
8. data = Sensors.getData(current)			✓		✓
9. }	✓	✓	✓	✓	✓
10. if (TimePassed() > MAX_TIME) {	✓	✓	✓	✓	✓
11. resendMessage(data);	✓			✓	✓
12. }	✓	✓	✓	✓	✓
13. }	✓	✓	✓	✓	✓
Energy Consumption (mJ)	0.5	0.7	0.2	1.3	1.5

USC Viterbi School of Engineering

EDTSO: Energy Minimization

```
 1. bin: b1, b2, b3, b4, b5;

 2. min: 0.5 b1 + 0.7 b2 + 0.2 b3 + 1.3 b4 + 1.5 b5;

 3.  s1: b1 + b2 + b3 + b4 + b5 >= 1;
 4.  s2: b1 + b2 + b3 + b4 + b5 >= 1;
 5.  s3: b1 + b2 + b3 + b4 + b5 >= 1;
 6.  s4: b1 + b2 + b3 + b4 + b5 >= 1;
 7.  s5:      b2 +      b4 + b5 >= 1;
 8.  s6: b1 + b2 + b3 + b4 + b5 >= 1;
 9.  s7: b1 + b2 + b3 + b4 + b5 >= 1;
10.  s8:           b3 +      b5 >= 1;
11.  s9: b1 + b2 + b3 + b4 + b5 >= 1;
12. s10: b1 + b2 + b3 + b4 + b5 >= 1;
13. s11: b1 +           b4      >= 1;
14. s12: b1 + b2 + b3 + b4 + b5 >= 1;
15. s13: b1 + b2 + b3 + b4 + b5 >= 1;
```

USC Viterbi School of Engineering

EDTSO: Energy Minimization

Coverage	Test Cases t_1	t_2	t_3	t_4	t_5
1. public void updateData(Location prior) {	✓	✓	✓	✓	✓
2. Location current = LocationManager.getLocation();	✓	✓	✓	✓	✓
3. SensorData data = getCurrentData();	✓	✓	✓	✓	✓
4. if (prior.distanceTo(current) > MAX_DISTANCE) {	✓	✓	✓	✓	✓
5. data = Sensors.getData(current);		✓		✓	
6. }	✓	✓	✓	✓	✓
7. if (OutOfDate(data)) {	✓	✓	✓	✓	✓
8. data = Sensors.getData(current)			✓		✓
9. }	✓	✓	✓	✓	✓
10. if (TimePassed() > MAX_TIME) {	✓	✓	✓	✓	✓
11. resendMessage(data);	✓			✓	✓
12. }	✓	✓	✓	✓	✓
13. }	✓	✓	✓	✓	✓
Energy Consumption (mJ)	0.5	0.7	0.2	1.3	1.5

USC Viterbi School of Engineering

EDTSO: Energy Minimization

```
 1. bin: b1, b2, b3, b4, b5;

 2. min: 0.5 b1 + 0.7 b2 + 0.2 b3 + 1.3 b4 + 1.5 b5;

 3.  s1: b1 + b2 + b3 + b4 + b5 >= 1;
 4.  s2: b1 + b2 + b3 + b4 + b5 >= 1;
 5.  s3: b1 + b2 + b3 + b4 + b5 >= 1;
 6.  s4: b1 + b2 + b3 + b4 + b5 >= 1;
 7.  s5:      b2 +       b4 + b5 >= 1;
 8.  s6: b1 + b2 + b3 + b4 + b5 >= 1;
 9.  s7: b1 + b2 + b3 + b4 + b5 >= 1;
10.  s8:           b3 +       b5 >= 1;
11.  s9: b1 + b2 + b3 + b4 + b5 >= 1;
12. s10: b1 + b2 + b3 + b4 + b5 >= 1;
13. s11: b1 +            b4      >= 1;
14. s12: b1 + b2 + b3 + b4 + b5 >= 1;
15. s13: b1 + b2 + b3 + b4 + b5 >= 1;
```

USC Viterbi School of Engineering

EDTSO: Energy Minimization

Coverage	Test Cases				
	t_1	t_2	t_3	t_4	t_5
1. public void updateData(Location prior) {	✓	✓	✓	✓	✓
2. Location current = LocationManager.getLocation();	✓	✓	✓	✓	✓
3. SensorData data = getCurrentData();	✓	✓	✓	✓	✓
4. if (prior.distanceTo(current) > MAX_DISTANCE) {	✓	✓	✓	✓	✓
5. data = Sensors.getData(current);		✓		✓	
6. }	✓	✓	✓	✓	✓
7. if (OutOfDate(data)) {	✓	✓	✓	✓	✓
8. data = Sensors.getData(current)			✓		✓
9. }	✓	✓	✓	✓	✓
10. if (TimePassed() > MAX_TIME) {	✓	✓	✓	✓	✓
11. resendMessage(data);	✓			✓	✓
12. }	✓	✓	✓	✓	✓
13. }	✓	✓	✓	✓	✓
Energy Consumption (mJ)	0.5	0.7	0.2	1.3	1.5

USC Viterbi School of Engineering

EDTSO: Energy Minimization

```
 1. bin: b1, b2, b3, b4, b5;

 2. min: 0.5 b1 + 0.7 b2 + 0.2 b3 + 1.3 b4 + 1.5 b5;

 3.  s1: b1 + b2 + b3 + b4 + b5 >= 1;
 4.  s2: b1 + b2 + b3 + b4 + b5 >= 1;
 5.  s3: b1 + b2 + b3 + b4 + b5 >= 1;
 6.  s4: b1 + b2 + b3 + b4 + b5 >= 1;
 7.  s5:      b2 +       b4 + b5 >= 1;
 8.  s6: b1 + b2 + b3 + b4 + b5 >= 1;
 9.  s7: b1 + b2 + b3 + b4 + b5 >= 1;
10.  s8:           b3 +       b5 >= 1;
11.  s9: b1 + b2 + b3 + b4 + b5 >= 1;
12. s10: b1 + b2 + b3 + b4 + b5 >= 1;
13. s11: b1 +            b4      >= 1;
14. s12: b1 + b2 + b3 + b4 + b5 >= 1;
15. s13: b1 + b2 + b3 + b4 + b5 >= 1;
```

USC Viterbi School of Engineering

EDTSO: Energy Minimization

Coverage		Test Cases				
		t_1	t_2	t_3	t_4	t_5
1.	`public void updateData(Location prior) {`	✓	✓	✓	✓	✓
2.	`  Location current = LocationManager.getLocation();`	✓	✓	✓	✓	✓
3.	`  SensorData data = getCurrentData();`	✓	✓	✓	✓	✓
4.	`  if (prior.distanceTo(current) > MAX_DISTANCE) {`	✓	✓	✓	✓	✓
5.	`  data = Sensors.getData(current);`		✓		✓	
6.	`  }`	✓	✓	✓	✓	✓
7.	`  if (OutOfDate(data)) {`	✓	✓	✓	✓	✓
8.	`  data = Sensors.getData(current)`			✓		✓
9.	`  }`	✓	✓	✓	✓	✓
10.	`  if (TimePassed() > MAX_TIME) {`	✓	✓	✓	✓	✓
11.	`  resendMessage(data);`	✓			✓	✓
12.	`  }`	✓	✓	✓	✓	✓
13.	`}`	✓	✓	✓	✓	✓
Energy Consumption (mJ)		0.5	0.7	0.2	1.3	1.5

USC Viterbi School of Engineering

EDTSO: Energy Minimization

```
 1. bin: b1, b2, b3, b4, b5;

 2. min: 0.5 b1 + 0.7 b2 + 0.2 b3 + 1.3 b4 + 1.5 b5;

 3.  s1: b1 + b2 + b3 + b4 + b5 >= 1;
 4.  s2: b1 + b2 + b3 + b4 + b5 >= 1;
 5.  s3: b1 + b2 + b3 + b4 + b5 >= 1;
 6.  s4: b1 + b2 + b3 + b4 + b5 >= 1;
 7.  s5:      b2 +       b4 + b5 >= 1;
 8.  s6: b1 + b2 + b3 + b4 + b5 >= 1;
 9.  s7: b1 + b2 + b3 + b4 + b5 >= 1;
10.  s8:           b3 +       b5 >= 1;
11.  s9: b1 + b2 + b3 + b4 + b5 >= 1;
12. s10: b1 + b2 + b3 + b4 + b5 >= 1;
13. s11: b1 +            b4      >= 1;
14. s12: b1 + b2 + b3 + b4 + b5 >= 1;
15. s13: b1 + b2 + b3 + b4 + b5 >= 1;
```

USC Viterbi School of Engineering

EDTSO: Energy Minimization

Coverage	Test Cases t_1	t_2	t_3	t_4	t_5
1. public void updateData(Location prior) {	✓	✓	✓	✓	✓
2. Location current = LocationManager.getLocation();	✓	✓	✓	✓	✓
3. SensorData data = getCurrentData();	✓	✓	✓	✓	✓
4. if (prior.distanceTo(current) > MAX_DISTANCE) {	✓	✓	✓	✓	✓
5. data = Sensors.getData(current);		✓		✓	
6. }	✓	✓	✓	✓	✓
7. if (OutOfDate(data)) {	✓	✓	✓	✓	✓
8. data = Sensors.getData(current)			✓		✓
9. }	✓	✓	✓	✓	✓
10. if (TimePassed() > MAX_TIME) {	✓	✓	✓	✓	✓
11. resendMessage(data);	✓			✓	✓
12. }	✓	✓	✓	✓	✓
13. }	✓	✓	✓	✓	✓
Energy Consumption (mJ)	0.5	0.7	0.2	1.3	1.5

USC Viterbi School of Engineering

EDTSO: Energy Minimization

```
 1. bin: b1, b2, b3, b4, b5;

 2. min: 0.5 b1 + 0.7 b2 + 0.2 b3 + 1.3 b4 + 1.5 b5;

 3.   s1: b1 + b2 + b3 + b4 + b5 >= 1;
 4.   s2: b1 + b2 + b3 + b4 + b5 >= 1;
 5.   s3: b1 + b2 + b3 + b4 + b5 >= 1;
 6.   s4: b1 + b2 + b3 + b4 + b5 >= 1;
 7.   s5:      b2 +      b4 + b5 >= 1;
 8.   s6: b1 + b2 + b3 + b4 + b5 >= 1;
 9.   s7: b1 + b2 + b3 + b4 + b5 >= 1;
10.   s8:           b3 +      b5 >= 1;
11.   s9: b1 + b2 + b3 + b4 + b5 >= 1;
12.  s10: b1 + b2 + b3 + b4 + b5 >= 1;
13.  s11: b1 +           b4      >= 1;
14.  s12: b1 + b2 + b3 + b4 + b5 >= 1;
15.  s13: b1 + b2 + b3 + b4 + b5 >= 1;
```

Evaluation

- **RQ 1: Effectiveness.** How much energy can EDTSO save?
- **RQ 2: Minimization time.** How much time does EDTSO need?
- **RQ 3: Proxy Measures.** How is EDTSO's performance impacted by using proxy measures?

Evaluation: Protocol

- Hardware platform: LEAP
  - An embedded x86 platform based on an ATOM N550 processo
- Structural
  - Techniqu Larus
- Energy me
  - Connected to the LEAP
  - Profiles the energy consumption at 10 kHz
- Time measurement: instrumented time stamps

Evaluation: Protocol

- To generate more test suites
  - We randomly selected 10,000 subsets of each test suite
  - We grouped the generated subsets into 10 groups based on coverage level wrt original test suite (i.e. 0%-10%, ... 90%-100%)

Evaluation: Benchmarks

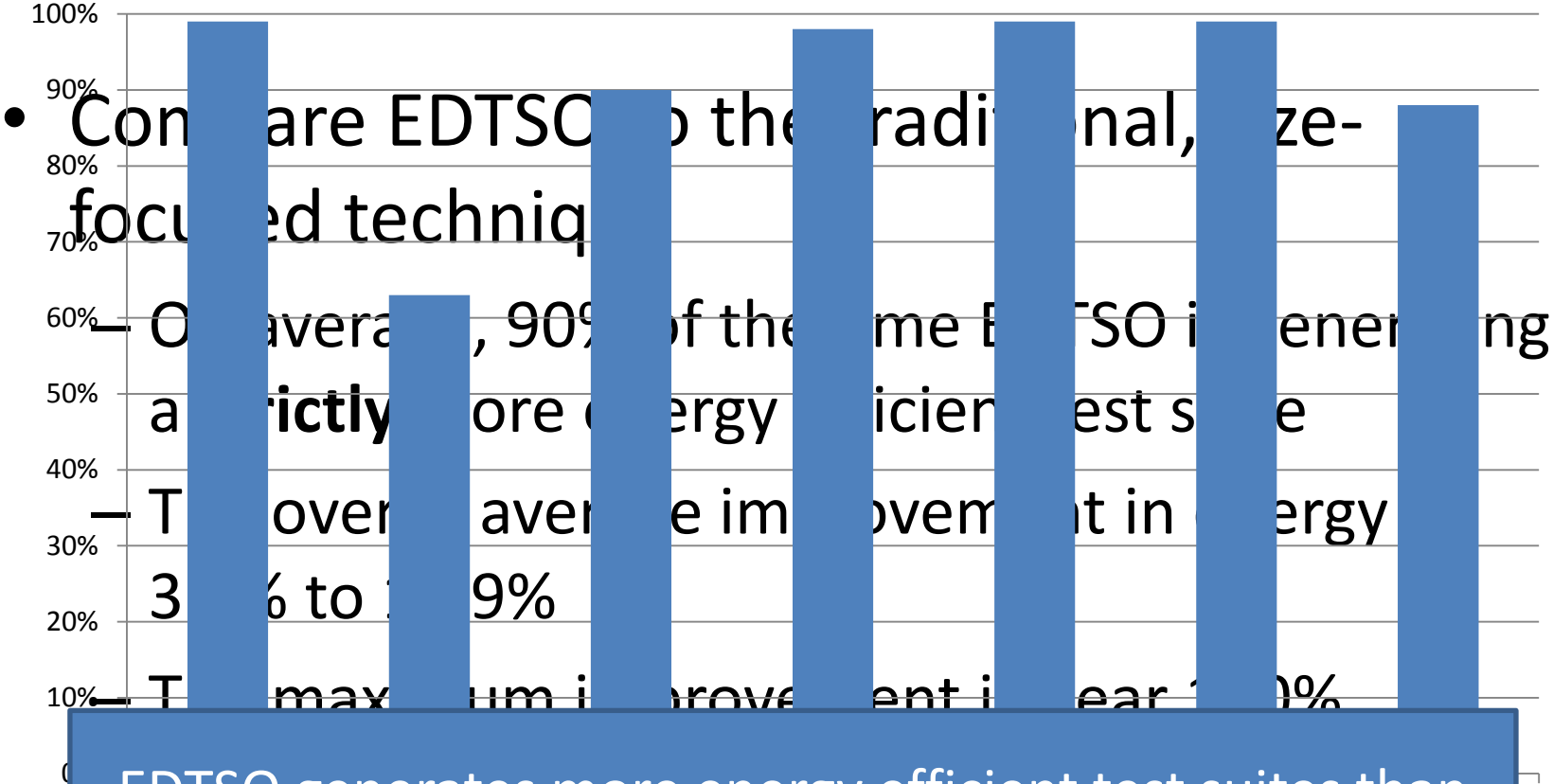
Subject	Description	LOC	#Tests
c:geo	Geocaching client	51,451	176
ConnectBot	SSH client	50,581	22
K-9 Mail	Email Client	71,816	38
MobileOrg	Org-mode file manipulation	14,819	87
MyTracks	Track user paths	35,039	310
Sky Map	Astronomical map	21,121	134
Yaaic	IRC client	19,293	28

They are real-world open source apps

USC Viterbi School of Engineering

RQ 1: Effectiveness

- Com are EDTSO o the radi nal, ze-focu ed techniq
  - O avera , 90% f the me E TSO i ener ng a rictly ore ergy icien est s e
  - T over aver e im ovem t in ergy 3 % to 9%
  - T max um i rove ent i ear 0%



EDTSO generates more energy efficient test suites than size-focused approaches

Frequency of EDTSO generating a more energy efficient test suite than traditional size focused approach

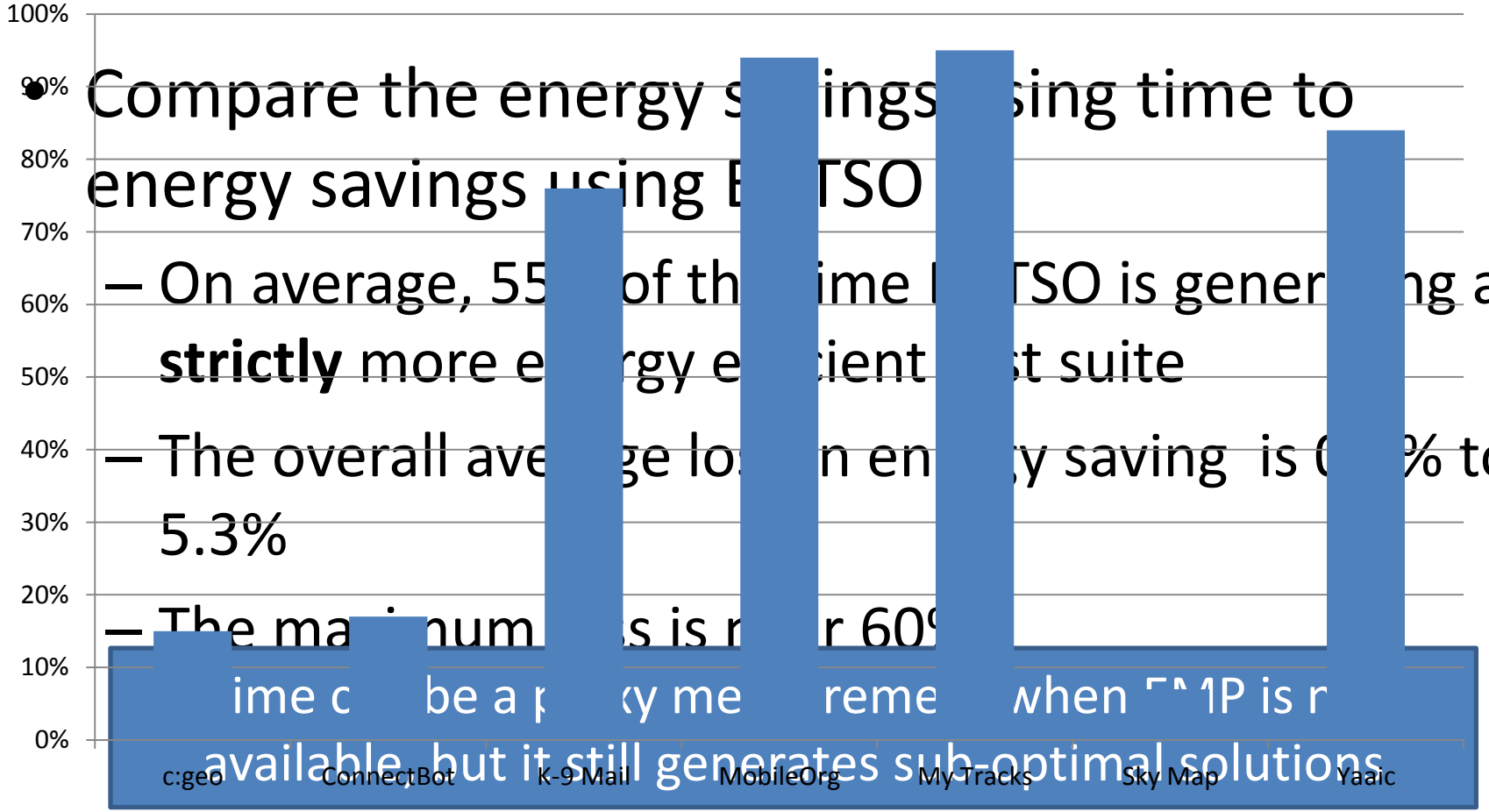
RQ 2: Minimization Time

- We measure the absolute time of EDTSO
  - All 70,000 generated subsets are minimized in less than 1 second
- We compare the minimization time of EDTSO to traditional size-focused approach
  - Only Yaaic has a significant difference ($p > 0.05$)

There is no significant difference in terms of minimization time

USC Viterbi School of Engineering

RQ 3: Proxy Measurement



- Compare the energy s ings sing time to energy savings using E TSO
  - On average, 55 of th ime l TSO is gener ng a **strictly** more e rgy e cient st suite
  - The overall ave ge lo n en y saving is 0 % to 5.3%
  - The ma imum s is r r 60
- ime c be a p y me reme when EDTSP is r available, but it still generates sub-optimal solutions

Frequency of EDTSO generating a more energy efficient test suite than using time as a proxy measurement `

Conclusion

- EDTSO minimizes the energy of in situ test suites
- EDTSO can be integrated into current test suite minimization workflows
- We evaluated EDTSO on realistic market apps
  - Saved up to 95% energy compared to traditional size focused approach
  - Using time as a proxy measurement may lose up to 60% of energy savings
  - EDTSO was as fast as traditional approaches

USC Viterbi School of Engineering