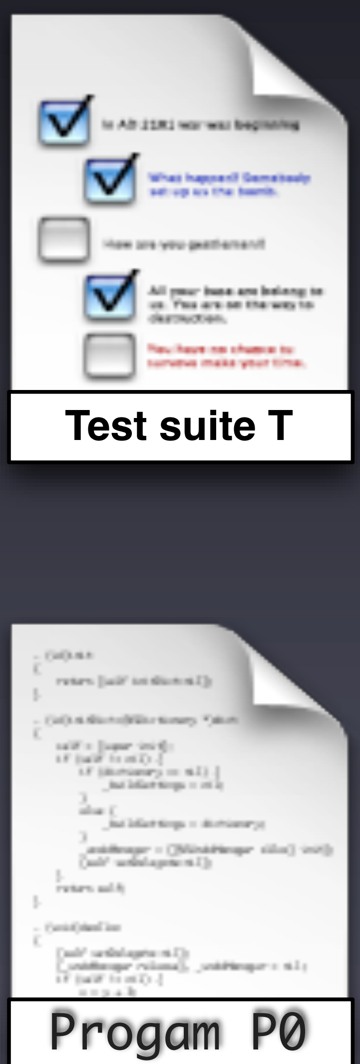
MINTS – A General Framework and Tool for Supporting Test-suite Minimization

Hwa-You Hsu and Alessandro Orso

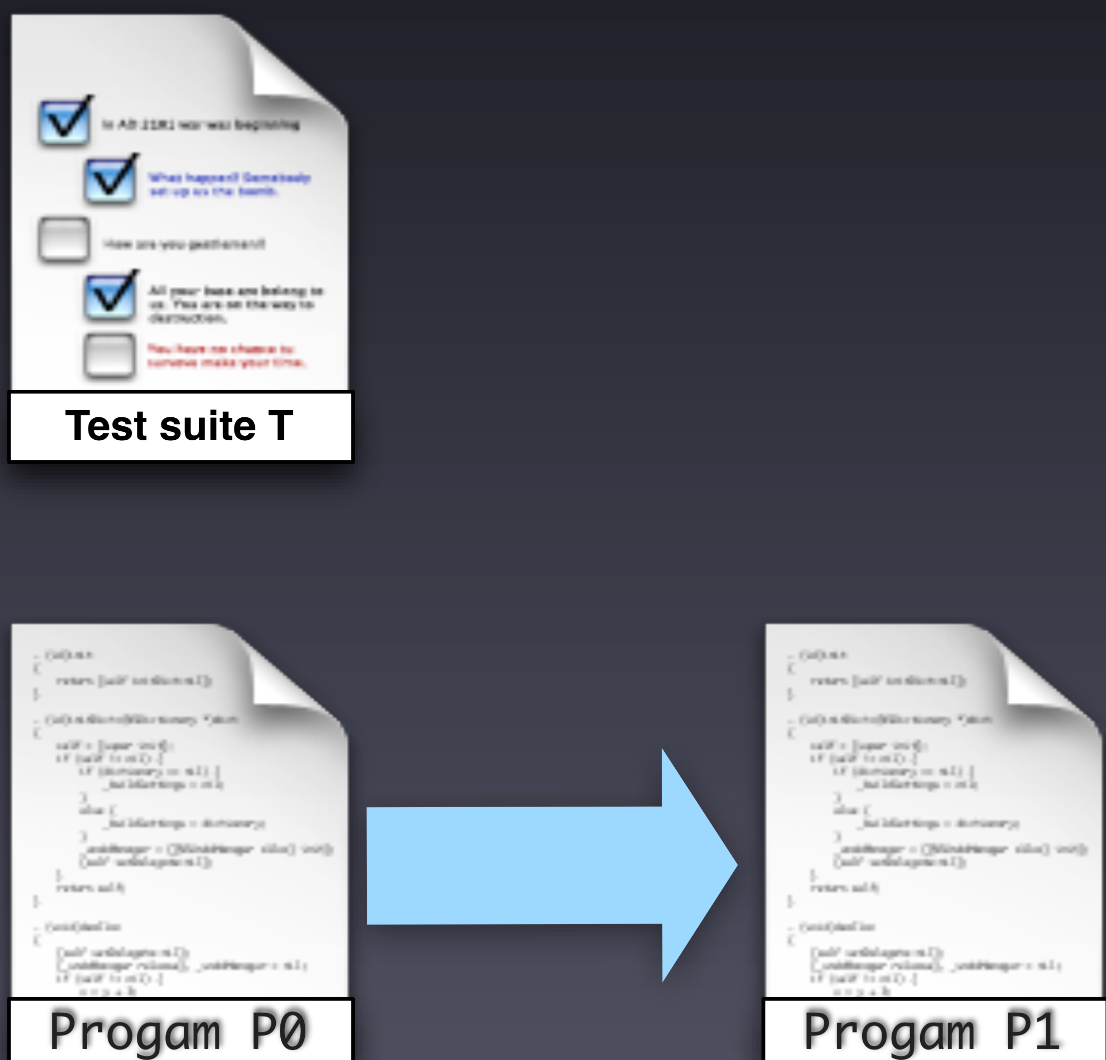
School of CS – College of Computing
Georgia Institute of Technology
http://www.cc.gatech.edu/~orso

Partially supported by: NSF, US Air Force, and IBM

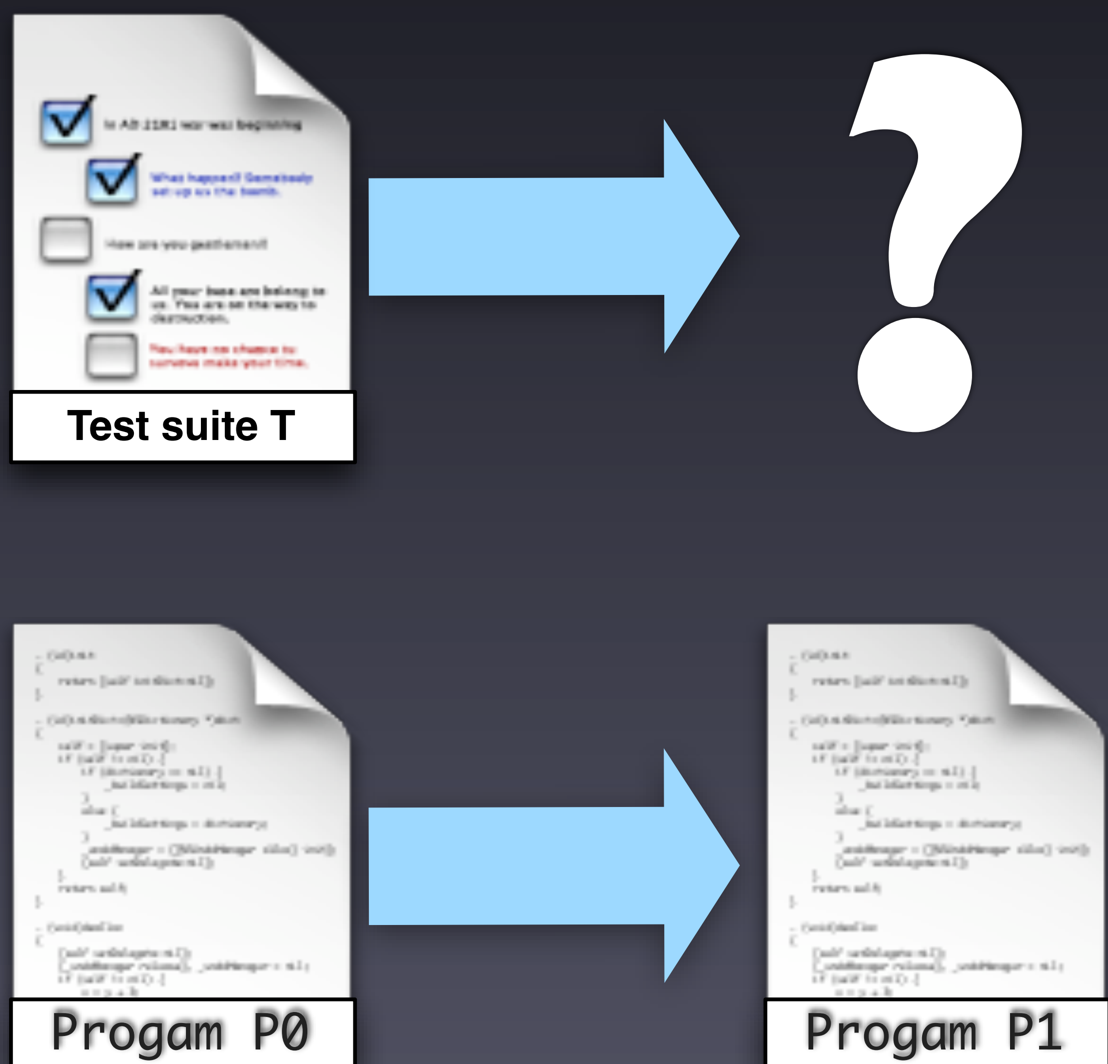
Regression Testing



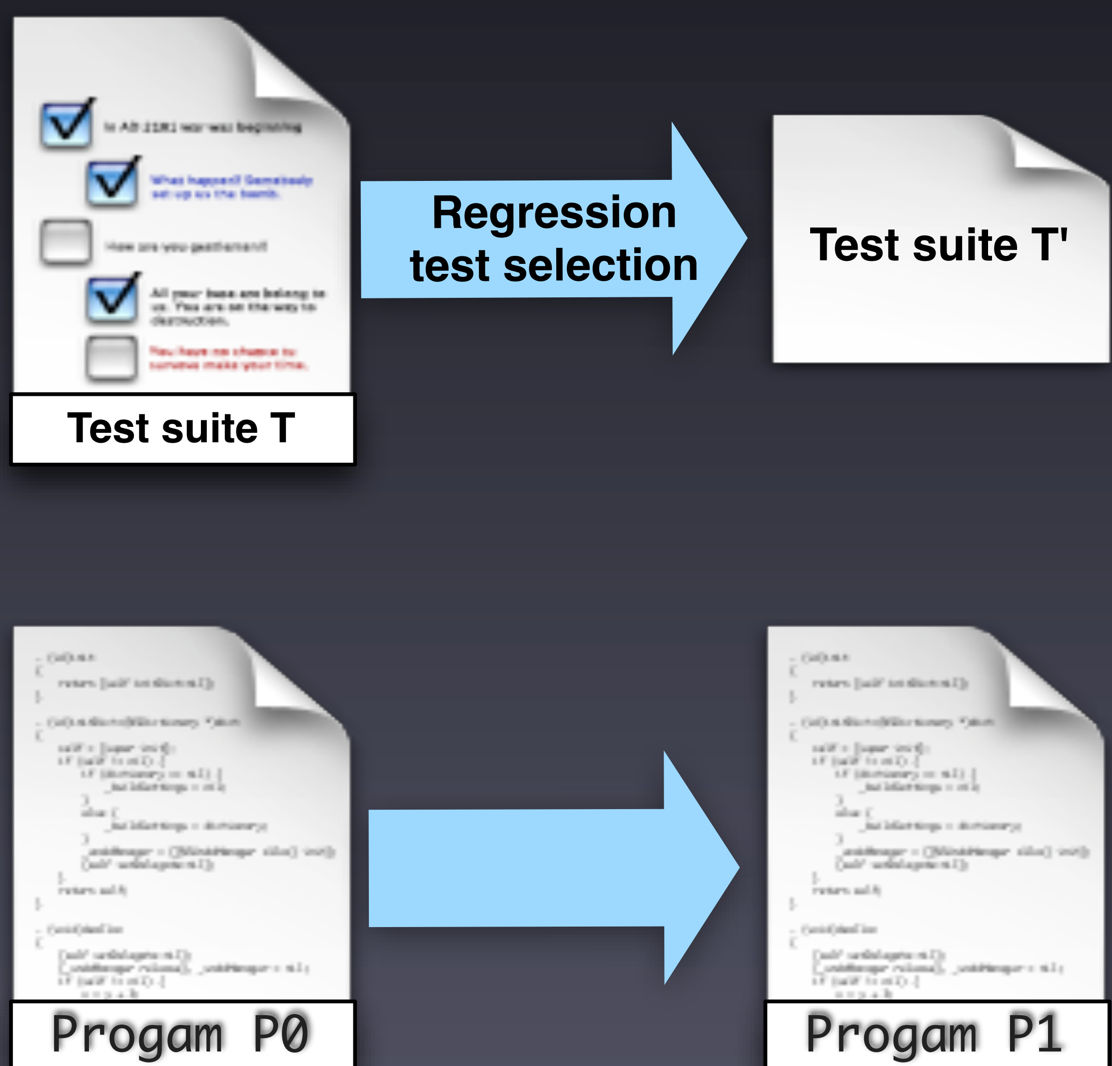
Regression Testing



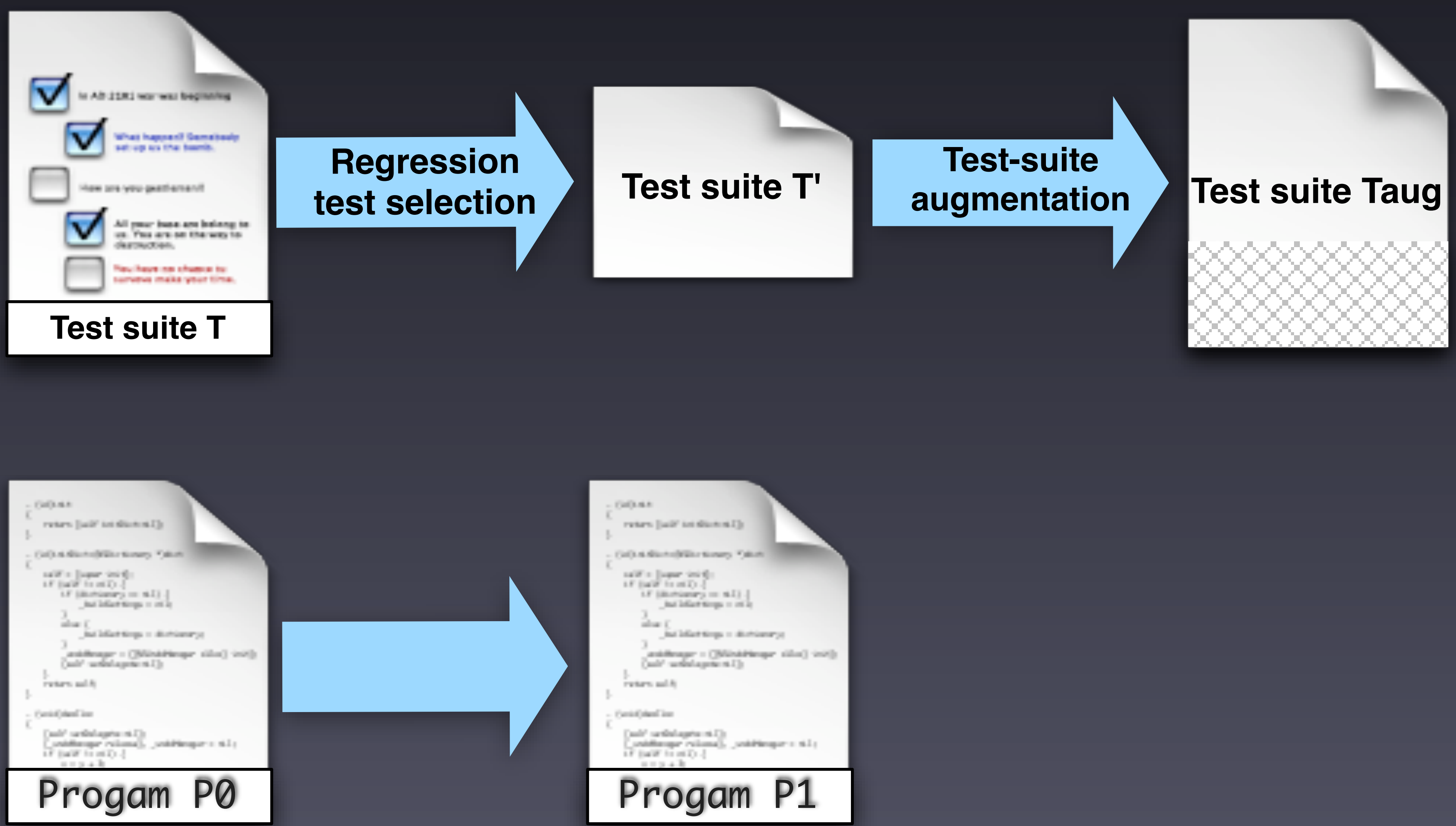
Regression Testing



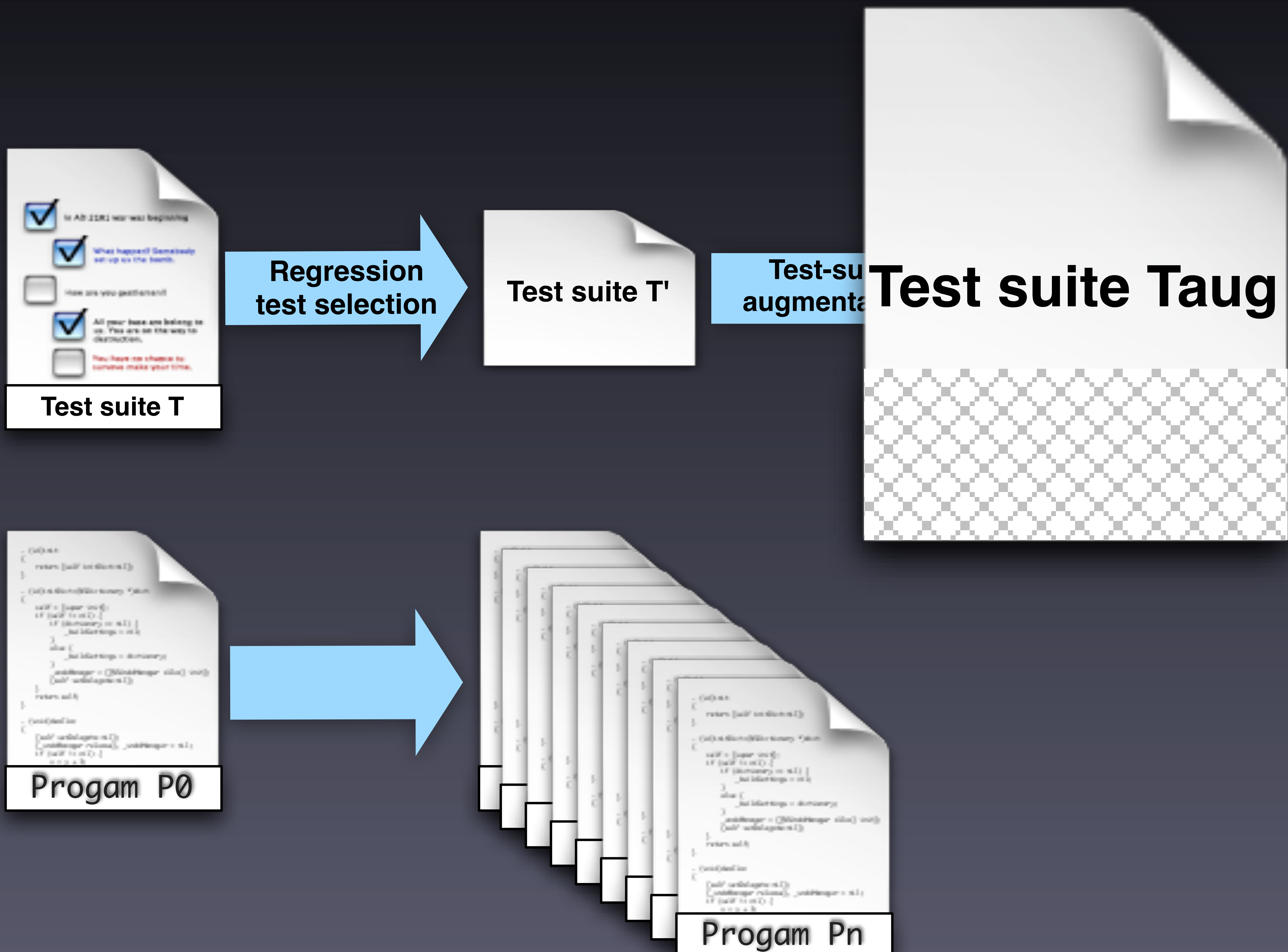
Regression Testing



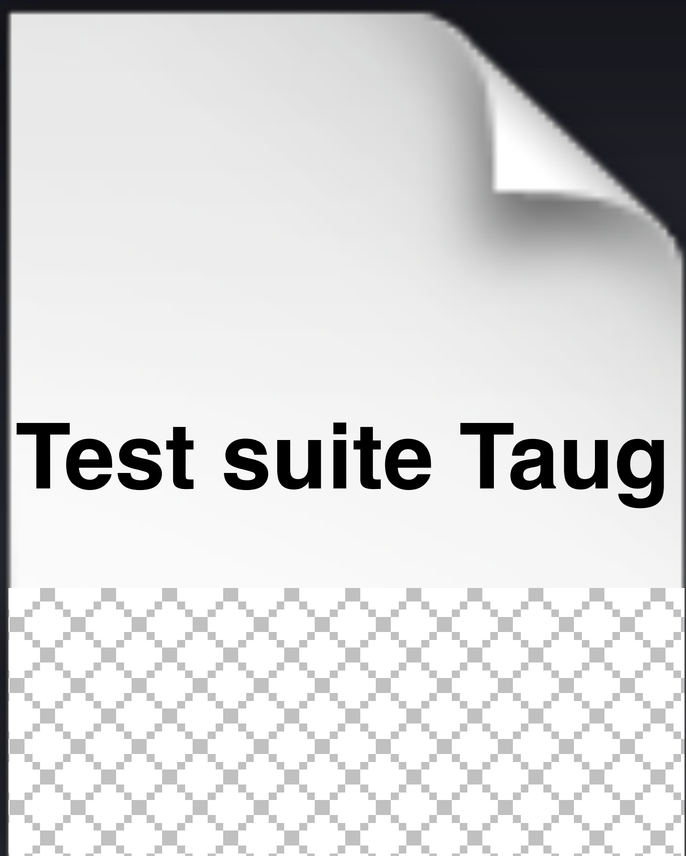
Regression Testing



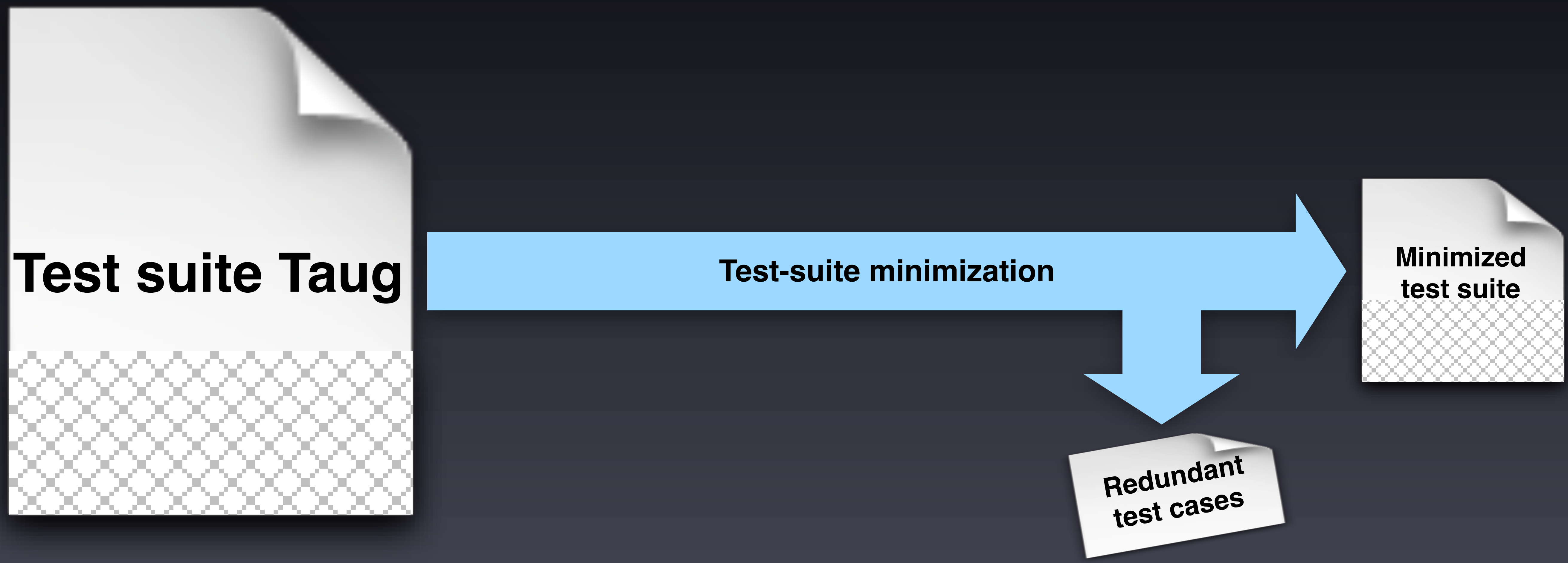
Regression Testing



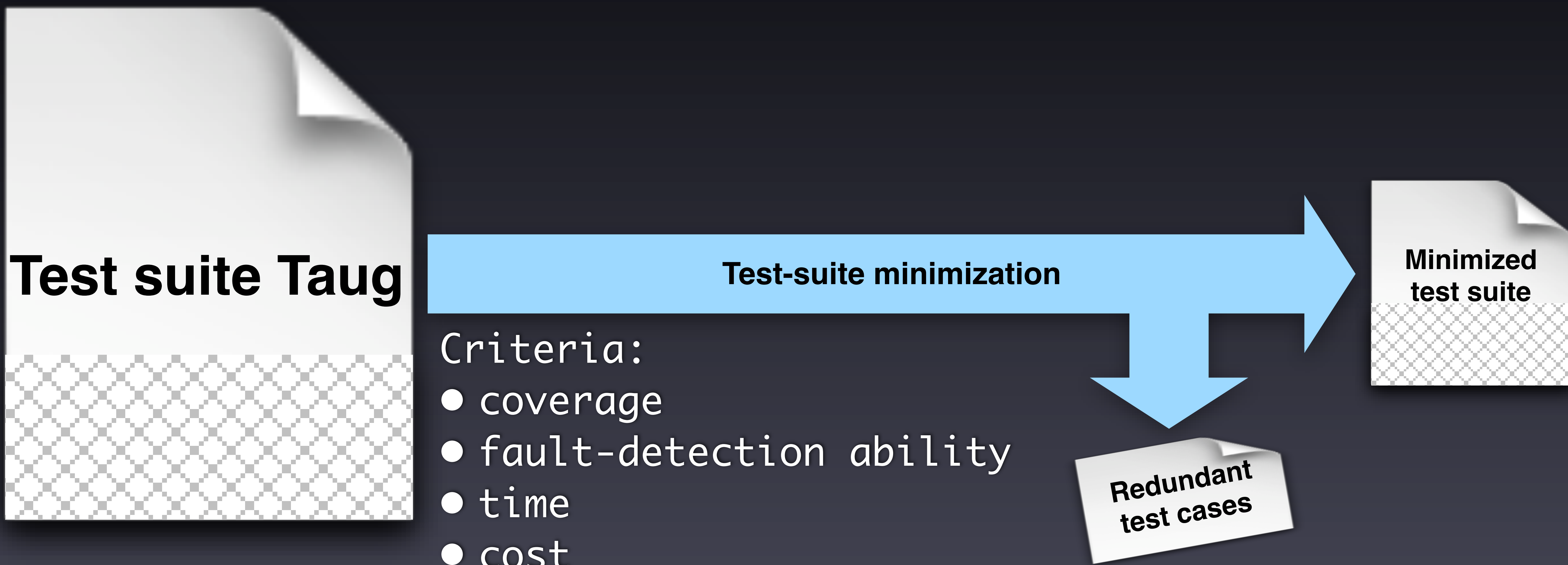
Test Suite Minimization



Test Suite Minimization



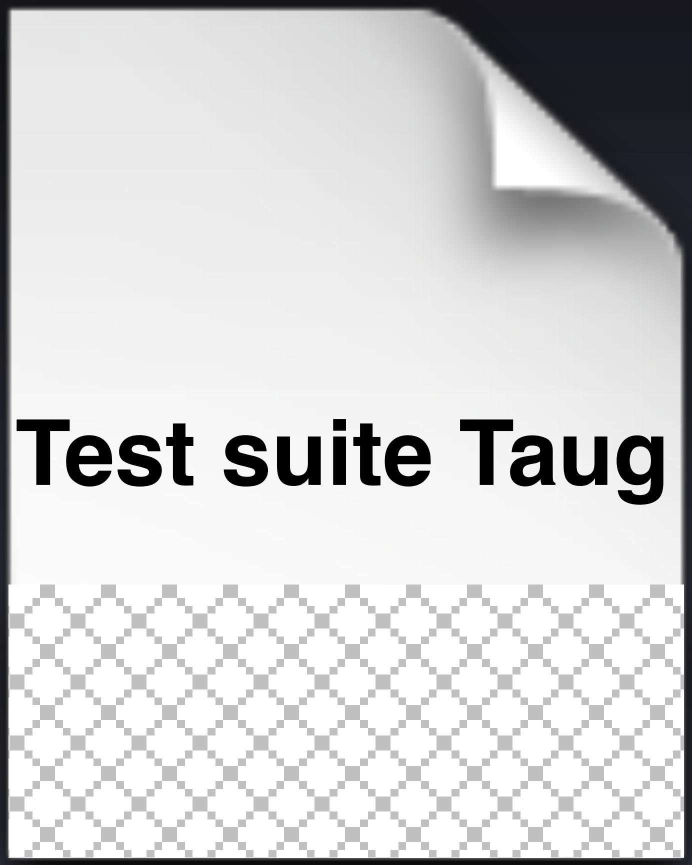
Test Suite Minimization



Criteria:

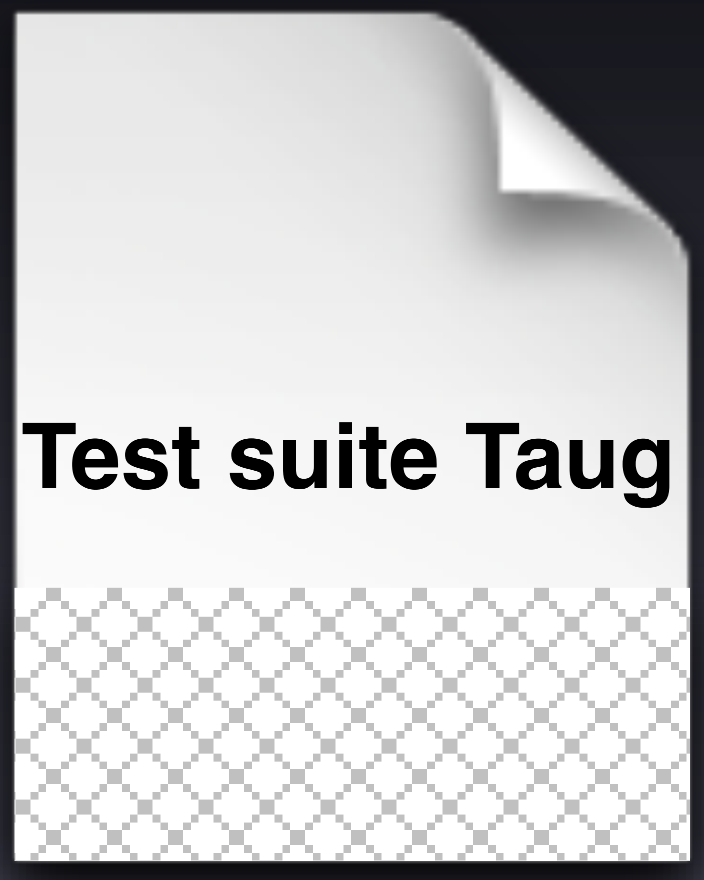
- coverage
- fault-detection ability
- time
- cost
- ...

A Simple Example



	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1

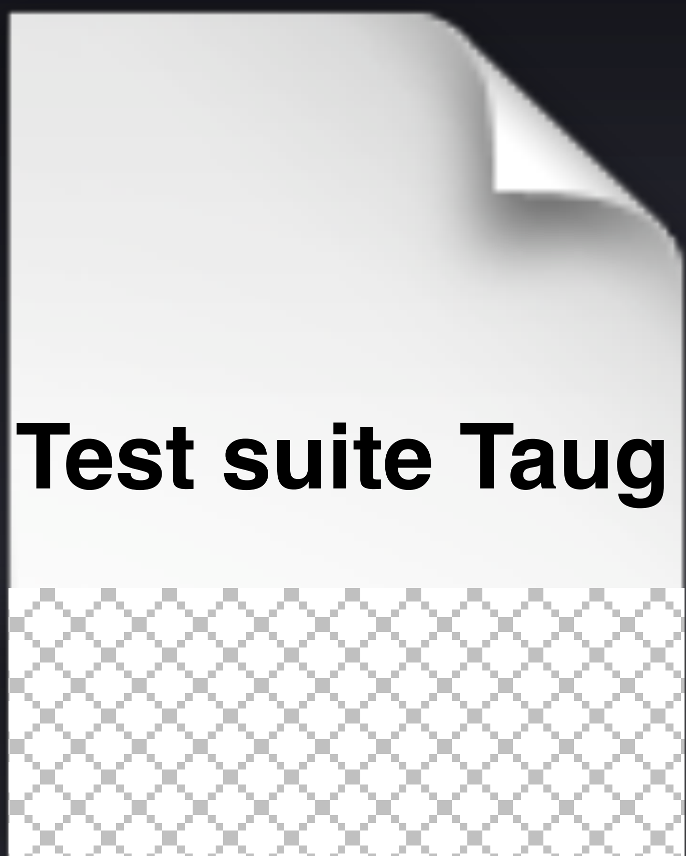
A Simple Example



	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1

Minimize test suite while maintaining the same level of coverage

A Simple Example



	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1

Minimize test suite while maintaining the same level of coverage

A More Realistic Example

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1

Criteria of interest:

C1 – maintain coverage

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1

Criteria of interest:

C1 – maintain coverage

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9

Criteria of interest:

C1 – maintain coverage

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
Fault detection ability	8	4	10	2

Criteria of interest:

C1 – maintain coverage

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
Fault detection ability	8	4	10	2

Criteria of interest:

C1 – maintain coverage

C2 – minimize time to run

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
Fault detection ability	8	4	10	2

Criteria of interest:

C1 – maintain coverage

C2 – minimize time to run

C3 – minimize setup effort

A More Realistic Example

Relevant parameters:

1. Test suite to minimize: T = {t1, t2, t3, t4}
2. Requirements to cover: R = {stmt1, stmt2, stmt3}
3. Test-related data: cost and fault-detection data

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
Fault detection ability	8	4	10	2

Criteria of interest:

C1 – maintain coverage

C2 – minimize time to run

C3 – minimize setup effort

C4 – maximize fault detection

State of the Art

- Several approaches in the literature (e.g., [HGS93],[H99],[MB03],[BMK04],[TG05])
- Two main limitations:
  - Single criterion
    (typically, coverage)
  - Approximated
    (problem is NP-complete)
- Only exception is [BMK04]: two criteria, but still limited in terms of expressiveness

Our Contribution

MINTS – novel technique (and freely-available tool) for test-suite minimization that:

- Lets testers specify a wide range of multi-criteria test-suite minimization problems
- Automatically encodes problems in binary ILP form
- Leverages different ILP solvers to find optimal solutions in a “reasonable” time

Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

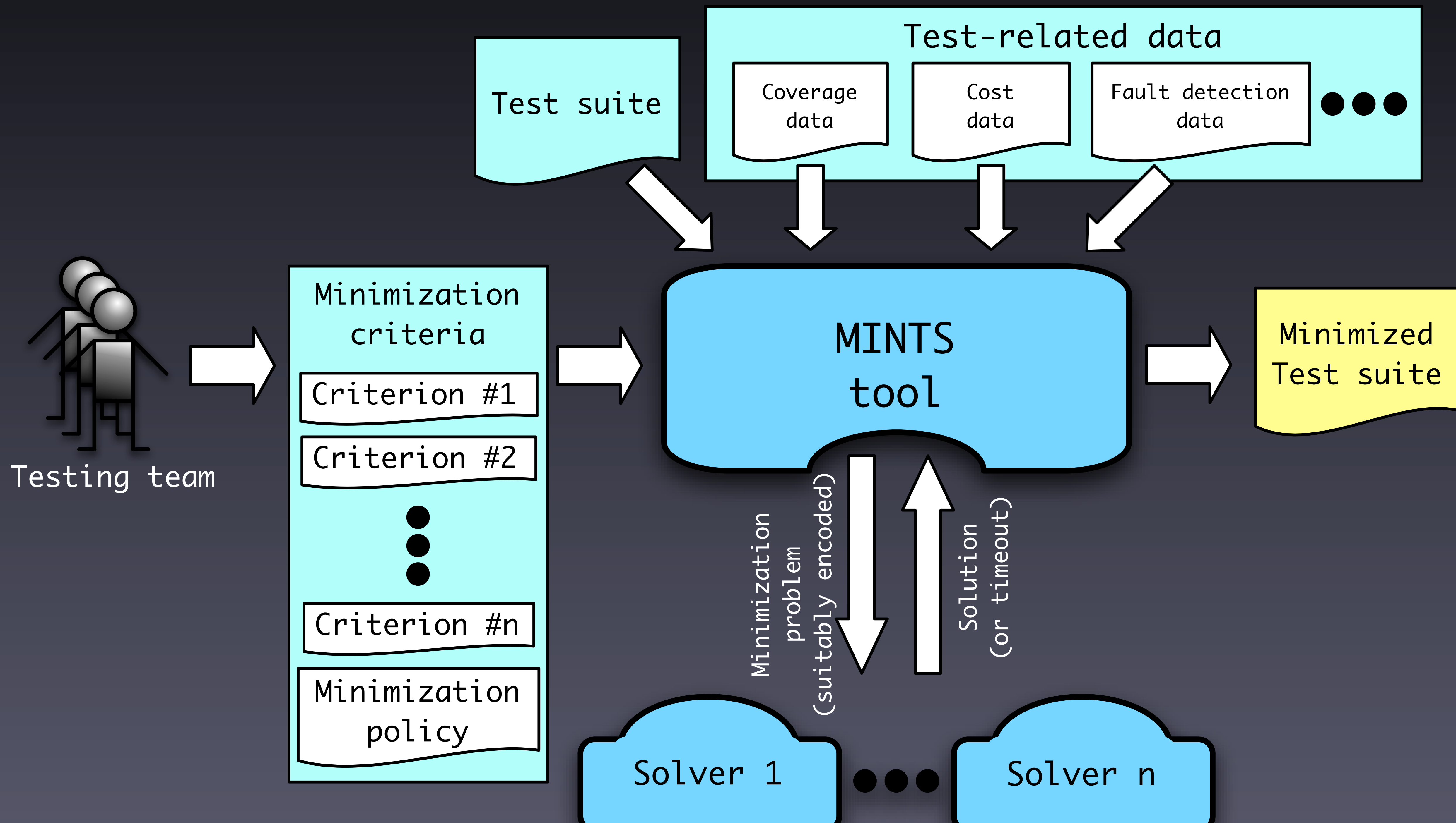
Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

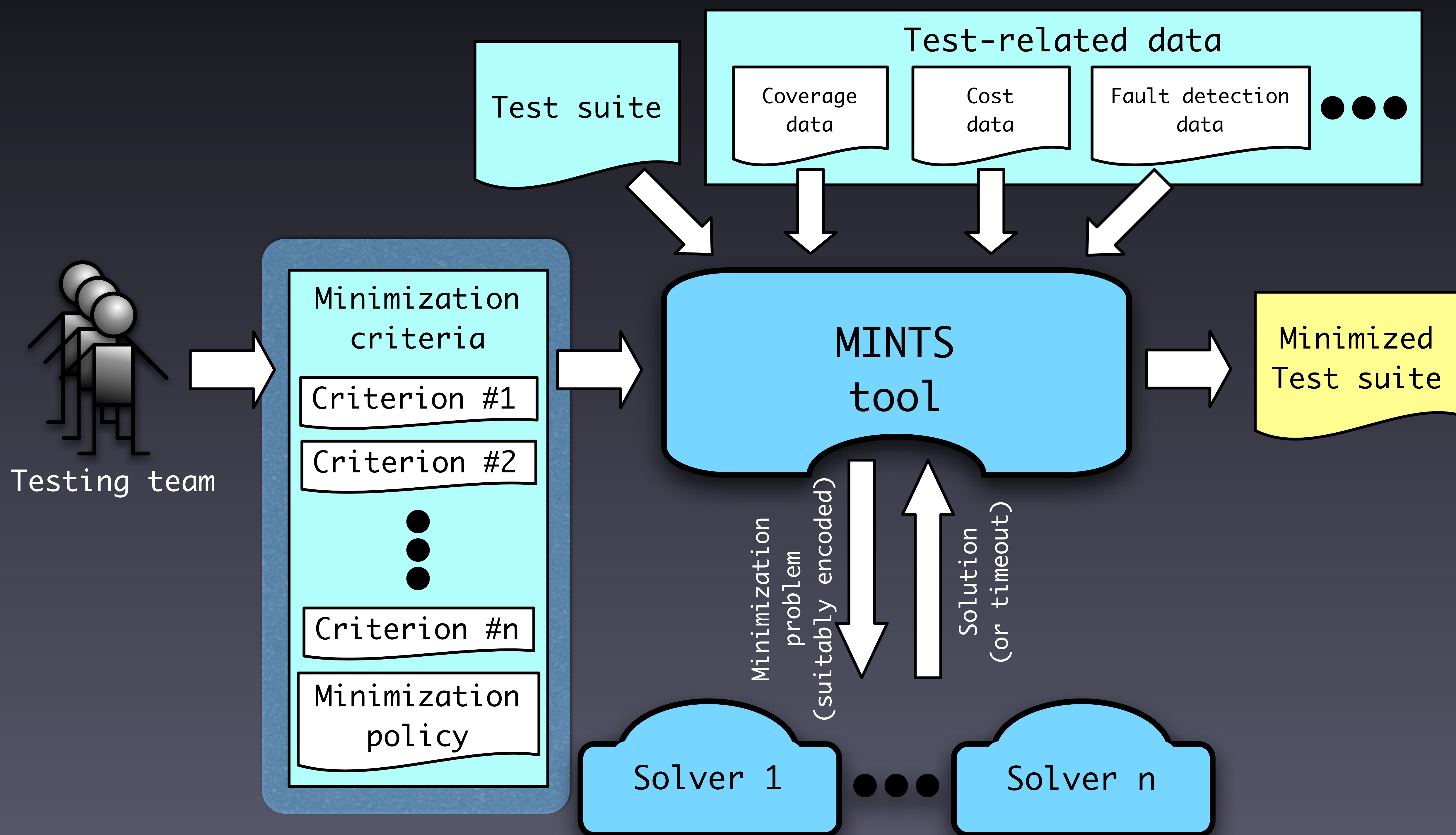
Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

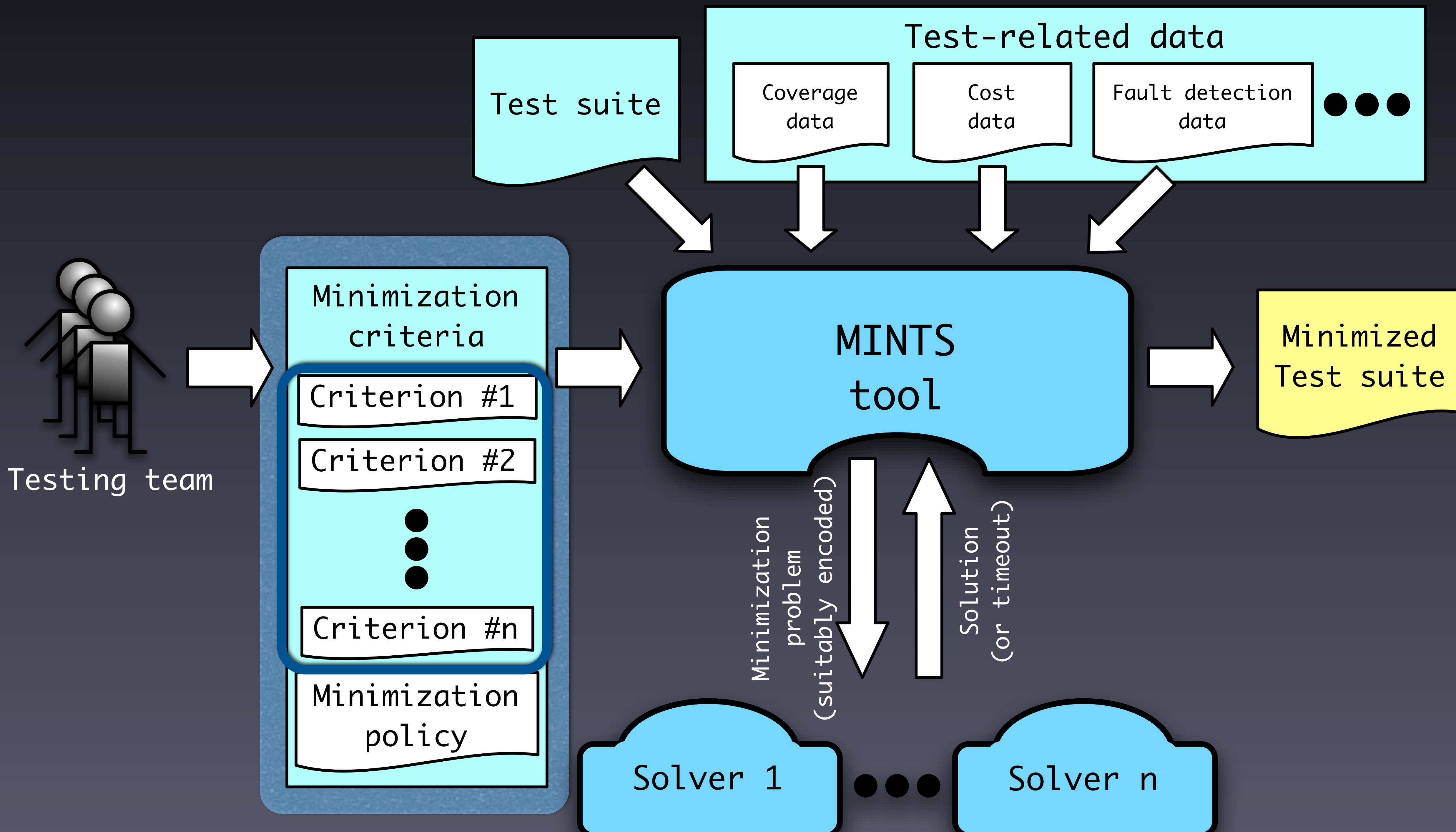
Overview of MINTS



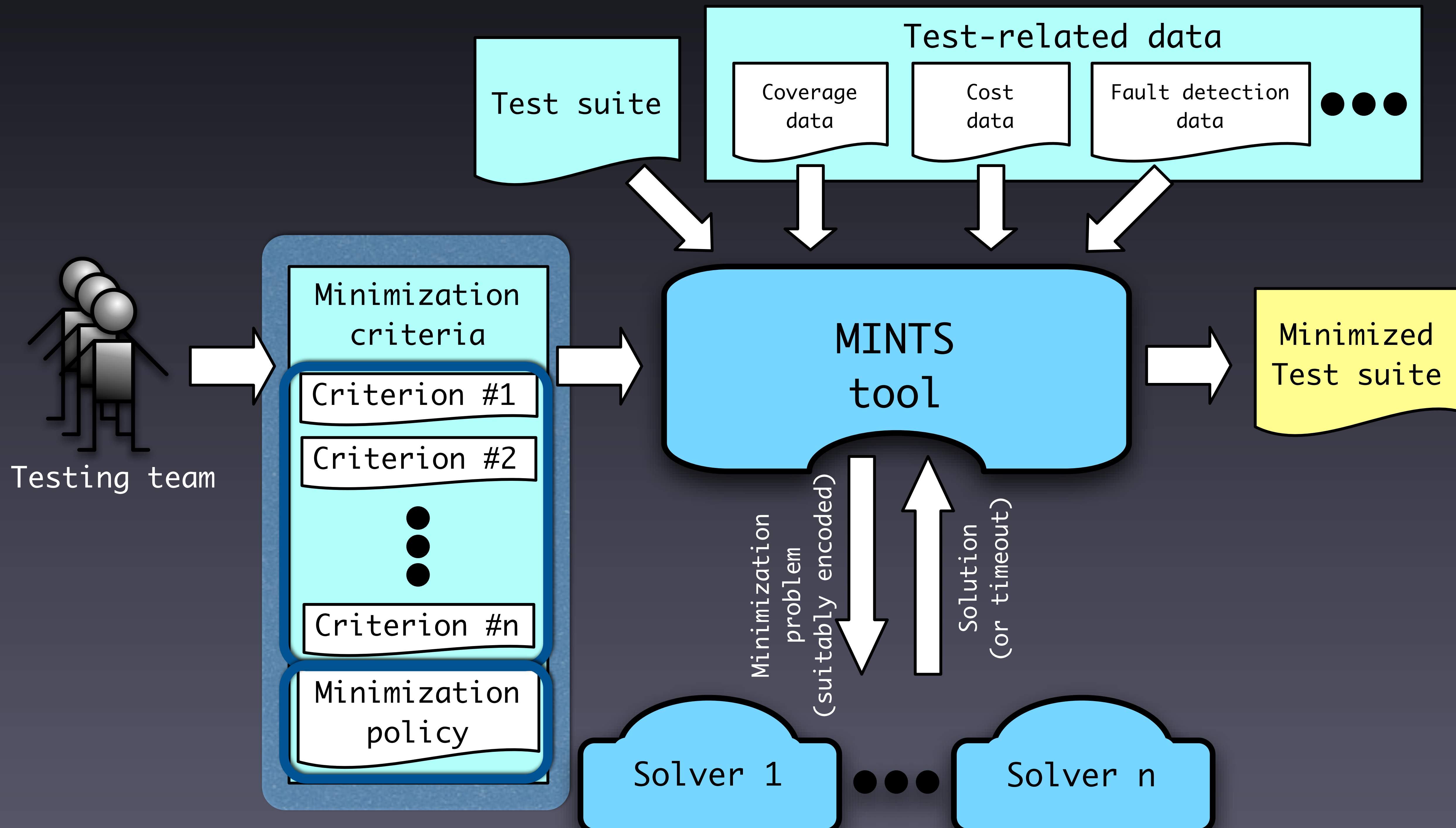
Overview of MINTS



Overview of MINTS



Overview of MINTS



Minimization Criteria

- Absolute criteria
  - Introduce a <u>constraint</u>
  - Example: C1 – Maintain statement coverage
- Relative criteria
  - Introduce an <u>objective</u>
  - Example: C2 – Minimize time to run
- Note: the same set of data can be used for either type of criteria

Minimization Policy

- Defines how to combine different objectives
- Weighted
- Prioritized
- Hybrid

Minimization Policy

- Defines how to combine different objectives
- Weighted
  - Testers associate a weight to each objective
  - Weights indicate relative importance
  - Example: very limited man power:
    C2 – minimize time to run ➡ 0.1
    C3 – minimize setup effort ➡ 0.8
    C4 – maximize fault detection ➡ 0.1
- Prioritized
- Hybrid

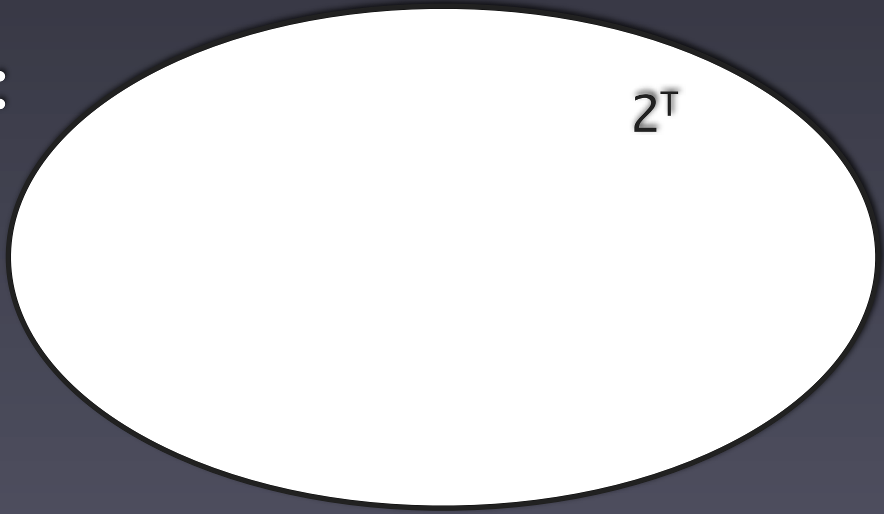
Minimization Policy

- Defines how to combine different objectives
- Weighted
- Prioritized
  - Testers specify a priority order for each objective
  - Priorities indicate order of processing
  - Example: C3 ➡ 1, C2 ➡ 2, C4 ➡ 3:
    $S1 \subseteq 2^T$ = min setup effort
    $S2 \subseteq S1$ = min testing time
    $S3 \subseteq S2$ = max fault detection
- Hybrid

C2 – minimize time to run
C3 – minimize setup effort
C4 – maximize fault detection

Minimization Policy

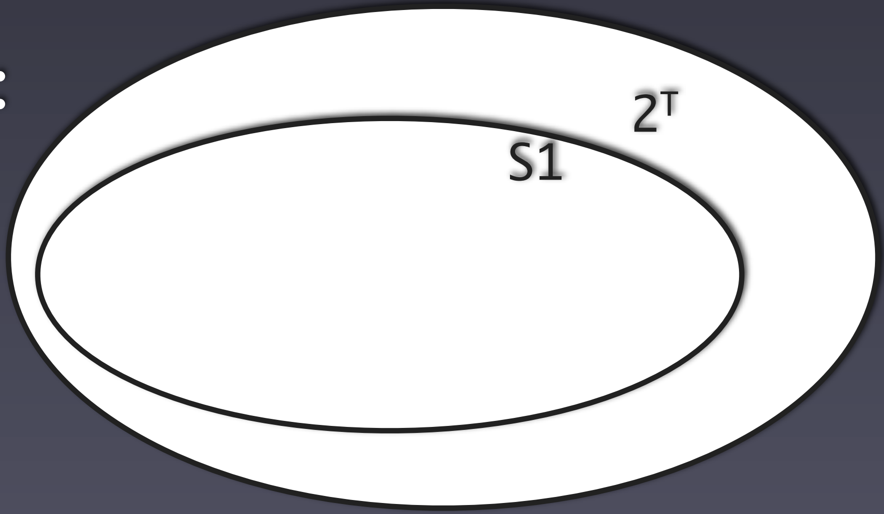
- Defines how to combine different objectives
- Weighted
- Prioritized
  - Testers specify a priority order for each objective
  - Priorities indicate order of processing
  - Example: C3 ➡ 1, C2 ➡ 2, C4 ➡ 3:
    $S1 \subseteq 2^T$ = min setup effort
    $S2 \subseteq S1$ = min testing time
    $S3 \subseteq S2$ = max fault detection
- Hybrid



C2 – minimize time to run
C3 – minimize setup effort
C4 – maximize fault detection

Minimization Policy

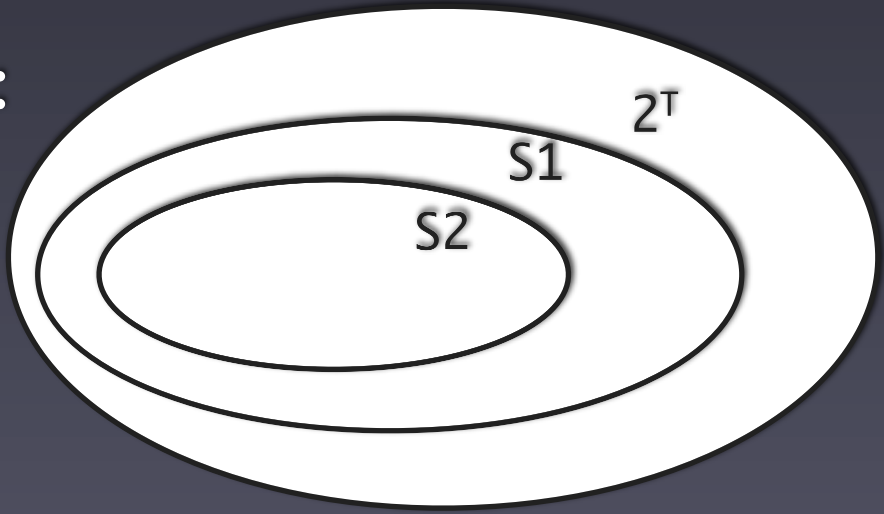
- Defines how to combine different objectives
- Weighted
- Prioritized
  - Testers specify a priority order for each objective
  - Priorities indicate order of processing
  - Example: C3 ➡ 1, C2 ➡ 2, C4 ➡ 3:
    $S1 \subseteq 2^T$ = min setup effort
    $S2 \subseteq S1$ = min testing time
    $S3 \subseteq S2$ = max fault detection
- Hybrid



C2 – minimize time to run
C3 – minimize setup effort
C4 – maximize fault detection

Minimization Policy

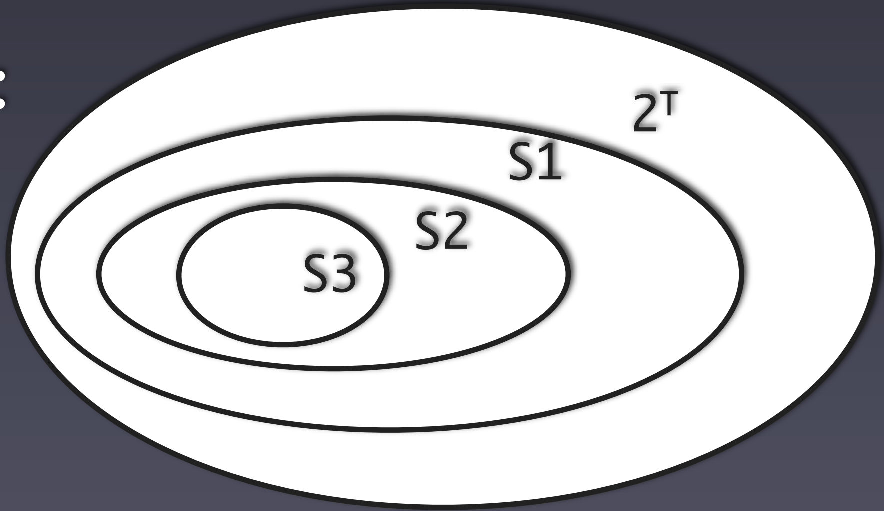
- Defines how to combine different objectives
- Weighted
- Prioritized
  - Testers specify a priority order for each objective
  - Priorities indicate order of processing
  - Example: C3 ➡ 1, C2 ➡ 2, C4 ➡ 3:
    $S1 \subseteq 2^T$ = min setup effort
    $S2 \subseteq S1$ = min testing time
    $S3 \subseteq S2$ = max fault detection
- Hybrid



C2 – minimize time to run
C3 – minimize setup effort
C4 – maximize fault detection

Minimization Policy

- Defines how to combine different objectives
- Weighted
- Prioritized
  - Testers specify a priority order for each objective
  - Priorities indicate order of processing
  - Example: C3 ➡ 1, C2 ➡ 2, C4 ➡ 3:
    $S1 \subseteq 2^T$ = min setup effort
    $S2 \subseteq S1$ = min testing time
    $S3 \subseteq S2$ = max fault detection



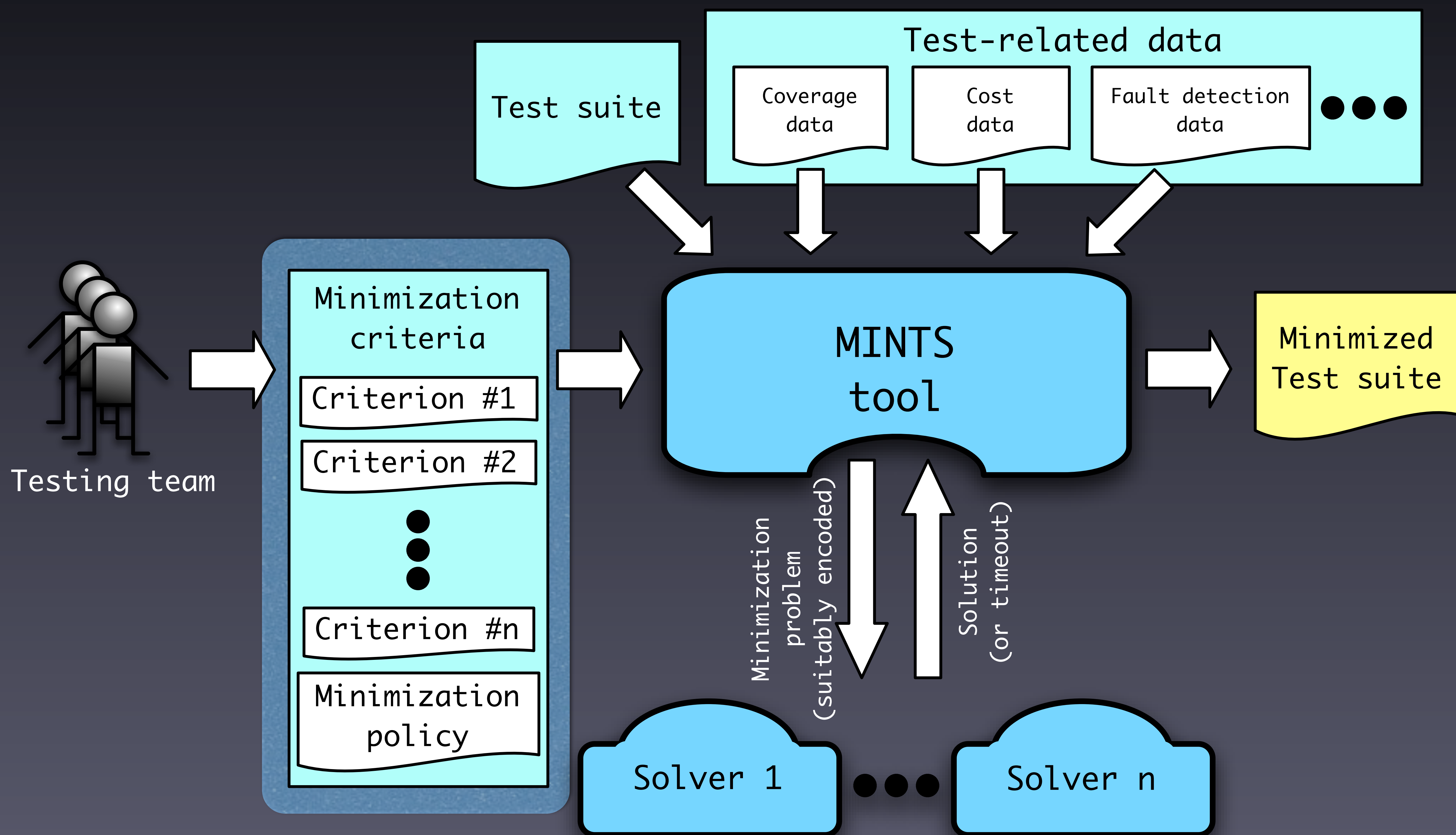
- Hybrid

C2 – minimize time to run
C3 – minimize setup effort
C4 – maximize fault detection

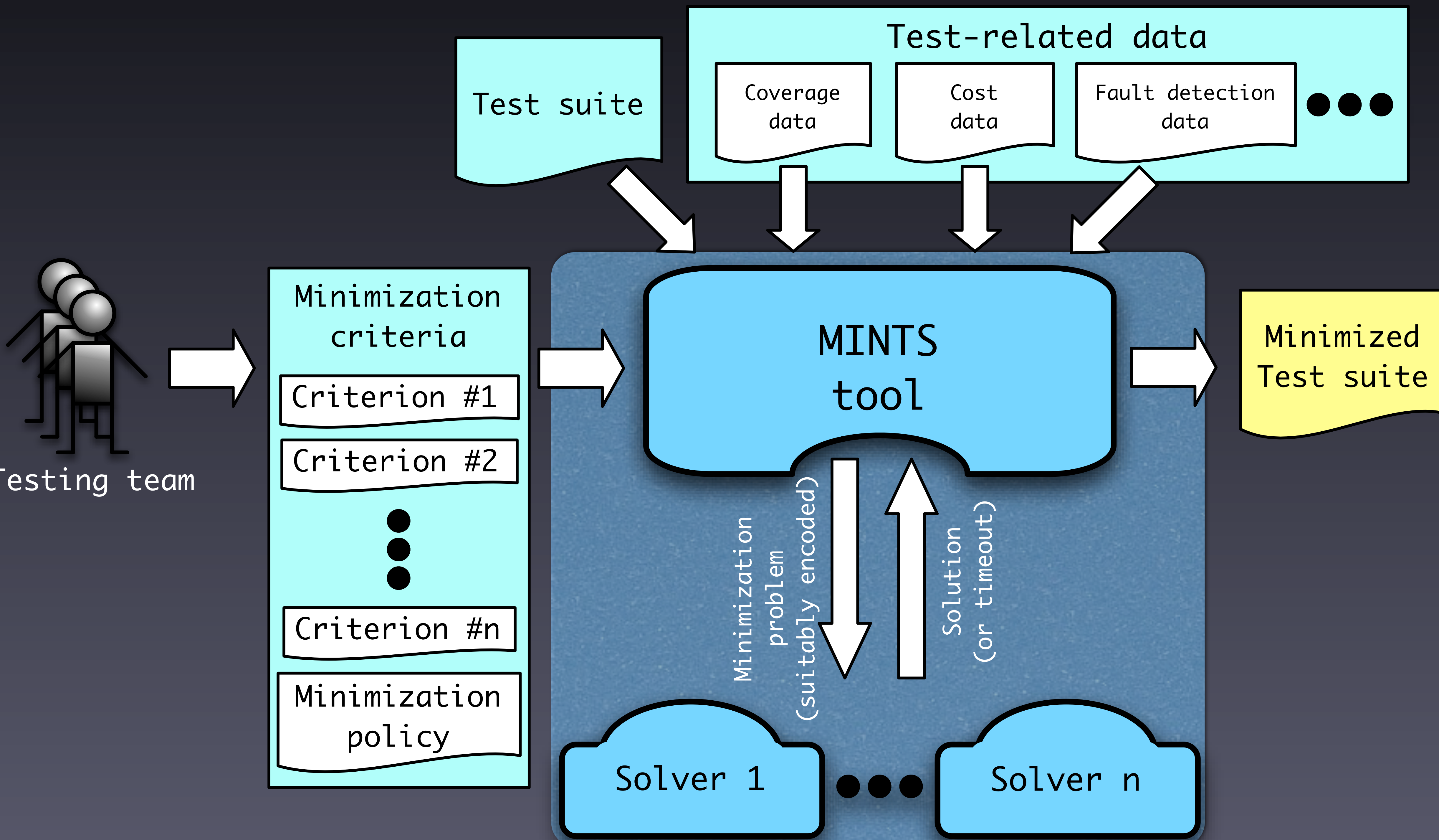
Minimization Policy

- Defines how to combine different objectives
- Weighted
- Prioritized
- Hybrid
  - Testers cluster objectives into groups and
    - assign weights to objects within group
    - assign priorities to groups

Overview of MINTS



Overview of MINTS



Multi-criteria minimization as a binary ILP problem: Encoding

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$

For example:

Criterion #1: (maintain coverage)

$$\sum_{j=1..4} d_{1,j}\ o_j = o_1 + o_3 \ge 1$$
$$\sum_{j=1..4} d_{2,j}\ o_j = o_1 + o_2 \ge 1$$
$$\sum_{j=1..4} d_{3,j}\ o_j = o_3 + o_4 \ge 1$$

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$
- Relative criteria (type x): min/max $\sum_{j=1..|T|} norm(d_{x,j}) o_j$ $(\sum_{j=1..|T|} norm(d_{,j}) = 1)$

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus \text{const}$
- Relative criteria (type x): $\min/\max \sum_{j=1..|T|} \text{norm}(d_{x,j}) o_j$
  $(\sum_{j=1..|T|} \text{norm}(d_{,j}) = 1)$

$\oplus$ = <, <=, =, >=, or >

For example:

Criterion #2 (minimize time to run):
$\min \sum_{j=1..4} \text{norm}(d_{3,j}) o_j = .5o_1 + .1o_2 + .36o_3 + .04o_4$

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$
- Relative criteria (type x): min/max $\sum_{j=1..|T|} norm(d_{x,j}) o_j$
  $(\sum_{j=1..|T|} norm(d_{,j}) = 1)$
- Minimization policies

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$
- Relative criteria (type x): min/max $\sum_{j=1..|T|} norm(d_{x,j}) o_j$ $(\sum_{j=1..|T|} norm(d_{,j}) = 1)$
- Minimization policies
  - Weighted: $\{\alpha_j\}$, $1 \le j \le$ #relative criteria

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Encoding

- Minimized test suite $MT=\{o_i\}$, $1 \le i \le |T|$, $o_i=1$ iff $t_i \in MT$
- Test-related data (types 1..n) $d_{all}=\{d_{i,j}\}$, $1 \le i \le |n|, 1 \le j \le |T|$
- Test-related data (type x) $d_x=\{d_{x,j}\}$, $1 \le j \le |T|$
- Absolute criteria (type x): $\sum_{j=1..|T|} d_{x,j} o_j \oplus const$
- Relative criteria (type x): min/max $\sum_{j=1..|T|} norm(d_{x,j}) o_j$ $(\sum_{j=1..|T|} norm(d_{,j}) = 1)$
- Minimization policies
  - Weighted: $\{\alpha_j\}$, $1 \le j \le$ #relative criteria
  - Prioritized: criterion ⇒ integer

$\oplus$ = <, <=, =, >=, or >

	t1	t2	t3	t4
stmt1	1		1	
stmt2	1	1		
stmt3			1	1
Time to run	22	4	16	2
Setup effort	3	0	11	9
F. detection	8	4	10	2

Multi-criteria minimization as a binary ILP problem: Weighted policy

Given

- n relative criteria involving test data d_{x1}, ..., d_{xn}
- m absolute criteria involving test data d_{y1}, ..., d_{ym}
- A weighted policy with weights α_1, ..., α_n

Multi-criteria minimization as a binary ILP problem: Weighted policy

Given

- n relative criteria involving test data $d_{x1}, \ldots, d_{xn}$
- m absolute criteria involving test data $d_{y1}, \ldots, d_{ym}$
- A weighted policy with weights $\alpha_1, \ldots, \alpha_n$

MINTS encode the minimization problem as

minimize

$\sum_{i=1..n} \alpha_i \sum_{j=1..|T|} norm(d_{xi,j}) o_j$

under the constraints

$\sum_{j=1..|T|} d_{y1,j}\ o_j \oplus const_1$

...

$\sum_{j=1..|T|} d_{y1,j}\ o_j \oplus const_1$

Multi-criteria minimization as a binary ILP problem: Weighted policy

Given

- n relative criteria involving test data $d_{x1}, \ldots, d_{xn}$
- m absolute criteria involving test data $d_{y1}, \ldots, d_{ym}$
- A weighted policy with weights $\alpha_1, \ldots, \alpha_n$

MINTS encode the minimization problem as

minimize

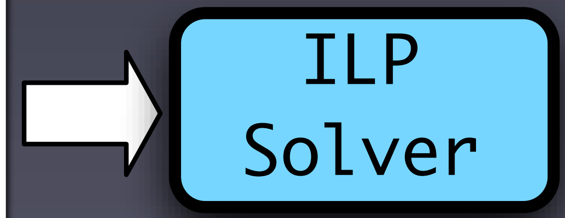
$$\sum_{i=1..n} \alpha_i \sum_{j=1..|T|} norm(d_{xi,j}) o_j$$

under the constraints

$$\sum_{j=1..|T|} d_{y1,j} \; o_j \; \oplus \; const_1$$

...

$$\sum_{j=1..|T|} d_{y1,j} \; o_j \; \oplus \; const_1$$



Multi-criteria minimization as a binary ILP problem: Weighted policy

Given

- n relative criteria involving test data $d_{x1}, \ldots, d_{xn}$
- m absolute criteria involving test data $d_{y1}, \ldots, d_{ym}$
- A weighted policy with weights $\alpha_1, \ldots, \alpha_n$

MINTS encode the minimization problem as

minimize

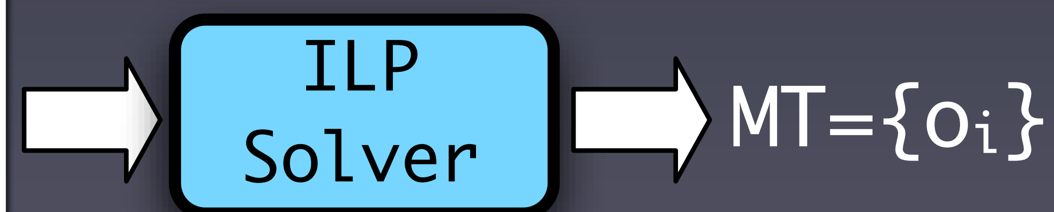
$\sum_{i=1..n} \alpha_i \sum_{j=1..|T|} norm(d_{xi,j}) o_j$

under the constraints

$\sum_{j=1..|T|} d_{y1,j}\ o_j \oplus const_1$

...

$\sum_{j=1..|T|} d_{y1,j}\ o_j \oplus const_1$



Multi-criteria minimization as a binary ILP problem: Weighted policy

Given

- n relative criteria involving test data $d_{x1}, \ldots, d_{xn}$
- m absolute criteria involving test data $d_{y1}, \ldots, d_{ym}$
- A weighted policy with weights $\alpha_1, \ldots, \alpha_n$

MINTS encode the minimization problem as

Minimize

$0.1(.5o_1+.1o_2+.36o_3+.04o_4) + 0.8(.13o_1+.48o_3+.39o_4) - 0.1(.3o_1+.17o_2+.42o_3+.08o_4)$

Under the constraints

$o_1 + o_3 \geq 1,\ o_1 + o_2 \geq 1,\ o_3 + o_4 \geq 1$

$\Rightarrow MT = \{0,1,1,0\}$

Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

Empirical Evaluation

- Goal: assess usefulness and practicality of the approach
- RQ1: How often can MINTS find an optimal solution “quickly”?
- RQ2: How does MINTS compare with a heuristic approach?
- RQ3: How does the use of a specific solver affect MINTS’s performance?

Empirical Evaluation

- Goal: assess usefulness and practicality of the approach
- RQ1: How often can MINTS find an optimal solution “quickly”?
- RQ2: How does MINTS compare with a heuristic approach?
- RQ3: How does the use of a specific solver affect MINTS’s performance?

Experimental Subjects and Solvers Considered

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

Experimental Subjects and Solvers Considered

- Subjects:

Subject	LOC	COV	#Test Cases	#Versions
tcas	173	72	1608	5
schedule2	307	146	2700	5
tot_info	406	136	1052	5
schedule	412	166	2650	5
replace	562	263	5542	5
print_tokens	563	194	4130	5
print_tokens2	570	197	4115	5
flex	12,421	567	548	5
LogicBlox	570,595	29204	393	5
Eclipse	1,892,226	35903	3621	5

- Solvers:
  Four SAT-based pseudo-Boolean and two pure ILP solvers

RQ1: How often can MINTS find an optimal solution quickly? (setup)

RQ1: How often can MINTS find an optimal solution quickly? (setup)

Test-related data

- Code coverage (gcov, cobertura)
- Running time (UNIX's time utility)
- Fault-detection ability (#faults detected in previous version)

RQ1: How often can MINTS find an optimal solution quickly? (setup)

Test-related data

- Code coverage (gcov, cobertura)
- Running time (UNIX's time utility)
- Fault-detection ability (#faults detected in previous version)

Minimization criteria

- One absolute: maintain statement coverage
- Three relatives: min size test suite, min execution time, max fault-detection capability

RQ1: How often can MINTS find an optimal solution quickly? (setup)

Test-related data

- Code coverage (gcov, cobertura)
- Running time (UNIX's time utility)
- Fault-detection ability (#faults detected in previous version)

Minimization criteria

- One absolute: maintain statement coverage
- Three relatives: min size test suite, min execution time, max fault-detection capability

Minimization policies

- Seven weighted: same weight; 0.6, 0.3, 0.1 (all combinations)
- One prioritized: (1) min size test suite, (2) min execution time, (3) max fault-detection capability

RQ1: How often can MINTS find an optimal solution quickly? (setup)

Test-related data

- Code coverage (gcov, cobertura)
- Running time (UNIX's time utility)
- Fault-detection ability (#faults detected in previous version)

Minimization criteria

- One absolute: maintain statement coverage
- Three relatives: min size test suite, min execution time, max fault-detection capability

Minimization policies

- Seven weighted: same weight; 0.6, 0.3, 0.1 (all combinations)
- One prioritized: (1) min size test suite, (2) min execution time, (3) max fault-detection capability

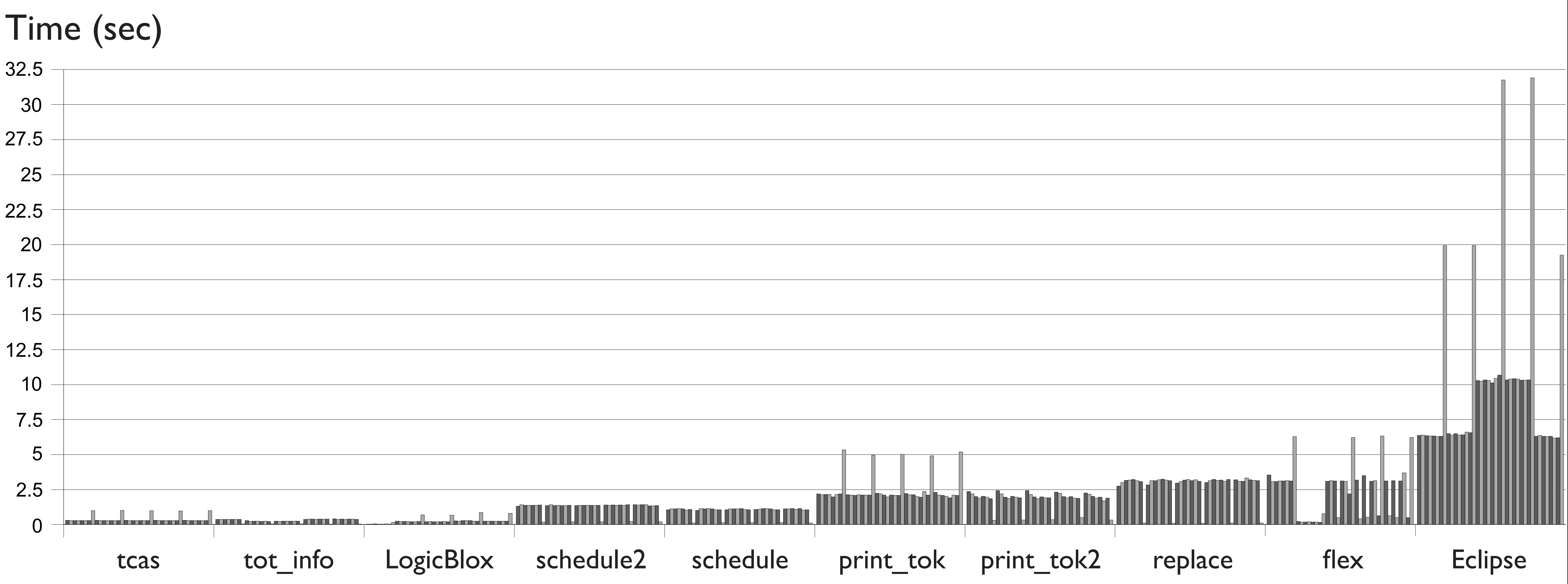
Overall, 400 minimization problems covering a wide spectrum

RQ1: How often can MINTS find an optimal solution quickly? (Process and results)

MINTS encoded each problem, submitted it to all solvers, and measured the time required to get the first solution

RQ1: How often can MINTS find an optimal solution quickly? (Process and results)

MINTS encoded each problem, submitted it to all solvers, and measured the time required to get the first solution

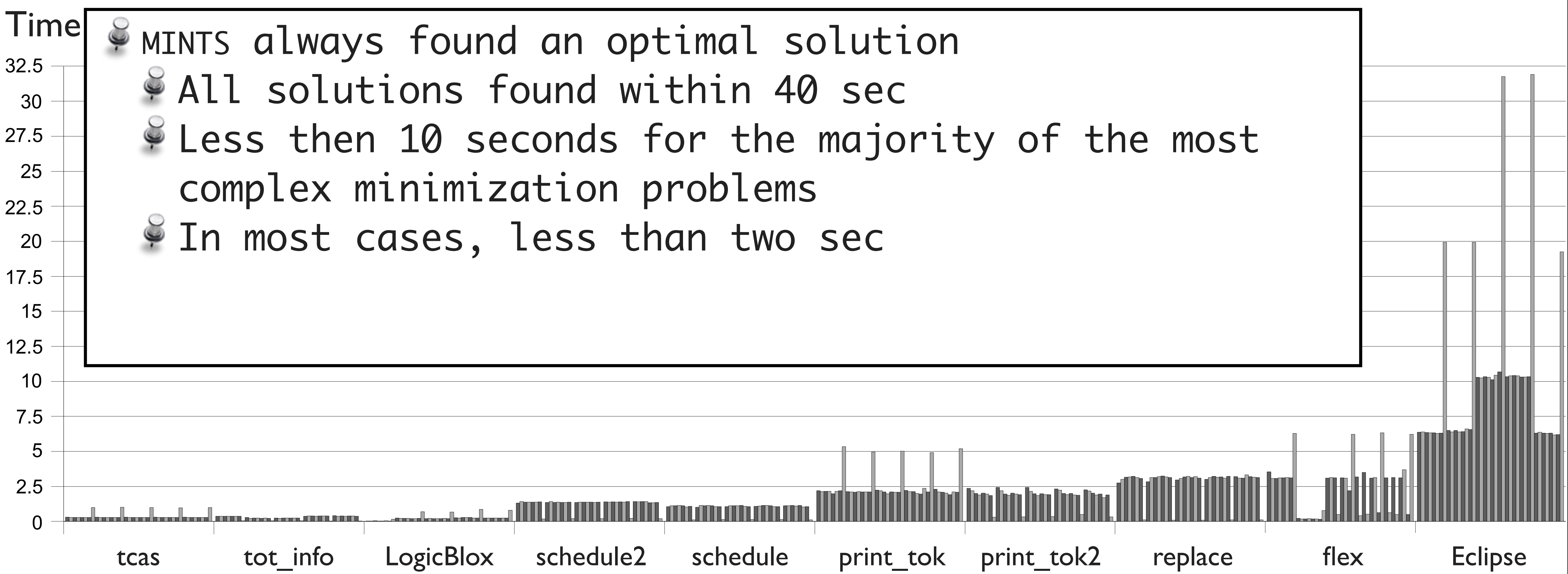


Ordered by complexity indicator – size of the subject x # test cases

RQ1: How often can MINTS find an optimal solution quickly? (Process and results)

MINTS encoded each problem, submitted it to all solvers, and measured the time required to get the first solution

- MINTS always found an optimal solution
  - All solutions found within 40 sec
  - Less then 10 seconds for the majority of the most complex minimization problems
  - In most cases, less than two sec

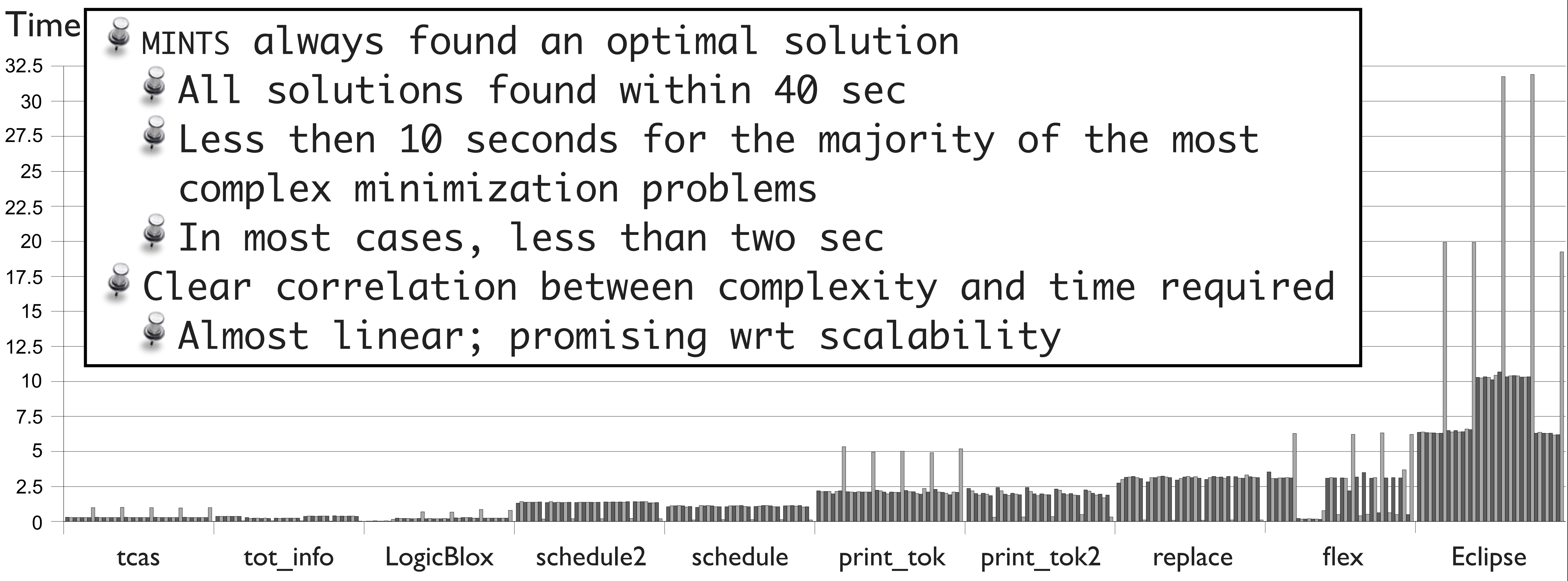


Ordered by complexity indicator – size of the subject x # test cases

RQ1: How often can MINTS find an optimal solution quickly? (Process and results)

MINTS encoded each problem, submitted it to all solvers, and measured the time required to get the first solution

- MINTS always found an optimal solution
  - All solutions found within 40 sec
  - Less then 10 seconds for the majority of the most complex minimization problems
  - In most cases, less than two sec
- Clear correlation between complexity and time required
  - Almost linear; promising wrt scalability



Ordered by complexity indicator – size of the subject x # test cases

RQ2: How does MINTS compare with a heuristic approach?

RQ2: How does MINTS compare with a heuristic approach?

Process

1. Single criterion: maintain statement coverage
2. Implemented HGS [HGS93] – well known, simple
3. Measured
    1. time to solve minimization problems
    2. size of resulting test suite

RQ2: How does MINTS compare with a heuristic approach?

Process

1. Single criterion: maintain statement coverage
2. Implemented HGS [HGS93] – well known, simple
3. Measured
    1. time to solve minimization problems
    2. size of resulting test suite

Results

- Both found solutions to all problems in a few seconds
- MINTS sometimes faster than HGS
- Minimized test suites of the same size for the Siemens programs and flex, of similar size for LogicBlox, and fairly different for Eclipse

RQ2: How does MINTS compare with a heuristic approach?

Eclipse version	Original T's size	HGS	MINTS	Difference
3.0.1	2460	656	418	238 (36%)
3.0.2	2467	651	423	228 (35%)
3.1	3621	851	553	298 (35%)
3.1.1	3681	833	532	301 (36%)
3.1.2	3681	656	406	250 (38%)

Results

- Both found solutions to all problems in a few seconds
- MINTS sometimes faster than HGS
- Minimized test suites of the same size for the Siemens programs and flex, of similar size for LogicBlox, and fairly different for Eclipse

Outline

- Introduction
- Our technique
- Empirical evaluation
- Conclusion and future work

Outline

- Introduction
- Our technique
- Empirical evaluation
- **Conclusion and future work**

Conclusion and Future Work

Conclusion and Future Work

Summary

- MINTS is a technique and tool for test suite minimization that
  - Allows for specifying a wide range of multi-criteria minimization problems
  - Computes (when successful) optimal solutions
- Empirical results show usefulness and applicability of the approach

Conclusion and Future Work

Summary

- MINTS is a technique and tool for test suite minimization that
  - Allows for specifying a wide range of multi-criteria minimization problems
  - Computes (when successful) optimal solutions
- Empirical results show usefulness and applicability of the approach

Future work

- Additional experimentation
- Study solvers' performance to go beyond the black box
- Extension of MINTS to include prioritization

Thank You!