Recent Research on Search Based Software Testing

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

Francisco Chicano

Optimization Problem

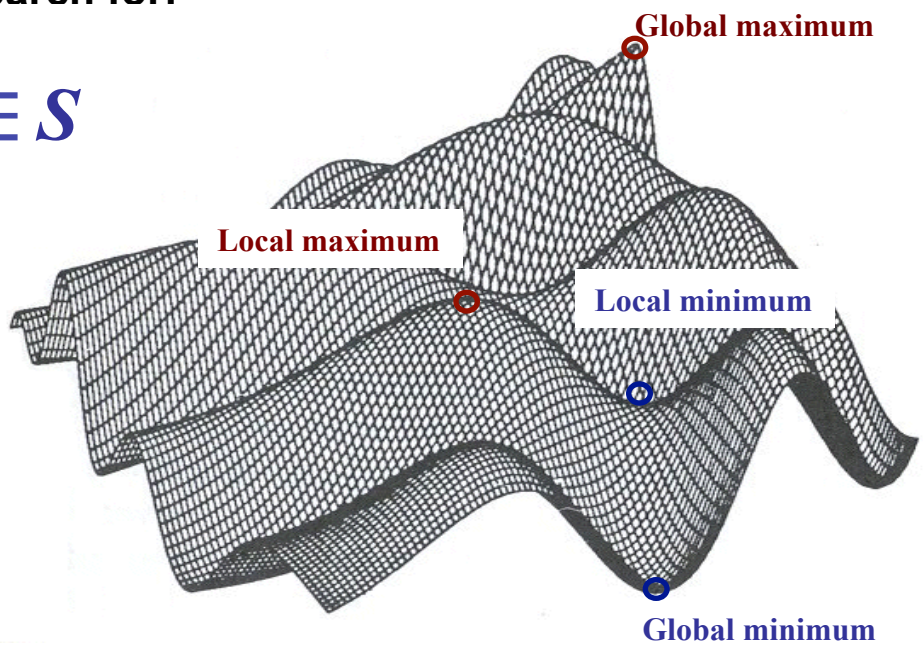
An optimization problem is a pair: $P = (S,f)$ where:

S is a set of solutions (solution or search space)

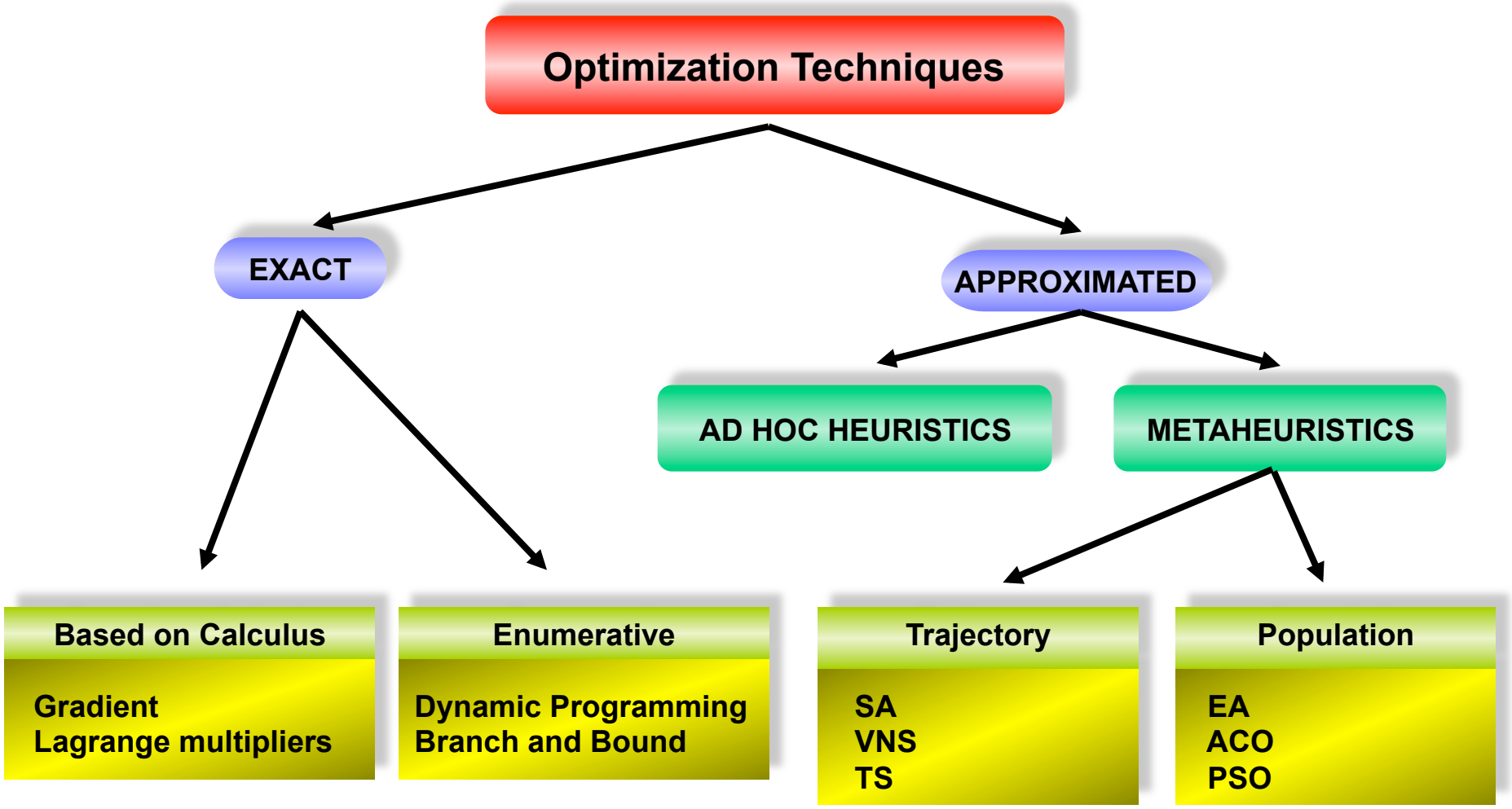
$f: S \rightarrow R$ is an objective function to minimize or maximize

If our goal is to minimize the function we search for:

$$s' \in S \mid f(s') \leq f(s), \forall s \in S$$



Optimization Techniques



Evolutionary Algorithm

Pseudocode of a simple EA

```
P = generateInitialPopulation ();
evaluate (P);
while not stoppingCondition () do
        P´ = selectParents (P);
        P´ = applyVariationOpterators (P´);
        evaluate(P´);
        P = selectNewPopulation (P,P´);
end while
return the best solution found
```

Three main steps: selection, reproduction, replacement

Variation operators → Make the population to evolve

Recombination: exchange of features

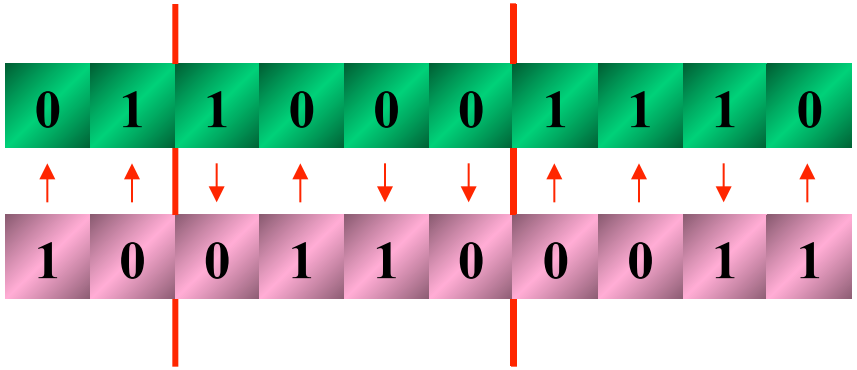
Mutation: generation of new features

Evolutionary Algorithm

Genetic Algorithms

- **Individuals**

- **Recombination**
  - **One point**
  - **Two points**
  - **Uniform**



- **Mutation → bit flips**

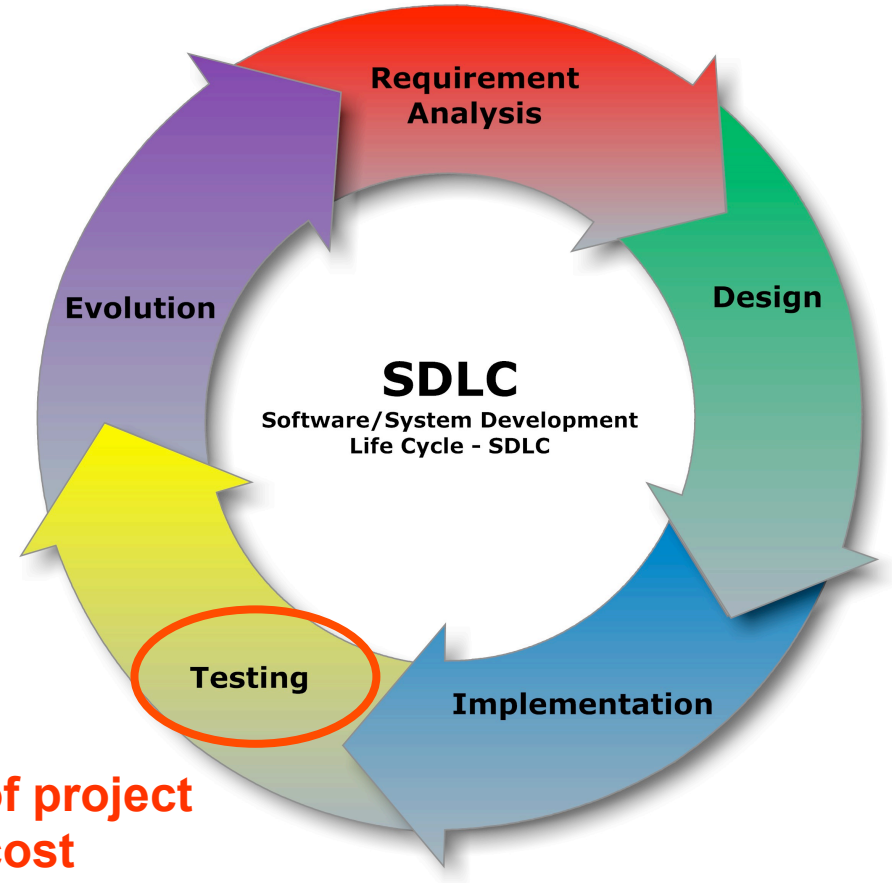
Software Testing: Definition and Goal

- What is software testing?
  - It is the process of running a software product or a portion of it in a controlled environment with a given input followed by the collection and analysis of the output and/or other relevant information of the execution.
- What is the goal of software testing?
  - To find out errors in a portion or the complete software product and/or to assure with a high probability that the software is correct (according to the requirements).

Software Testing: Impact

Software testing is important because…

60.000 M$ annually (0,6% GDP) in USA



60 % of project cost

Software Testing: Classification

Classification of testing techniques (by goal)

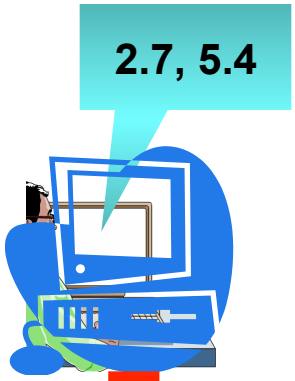
- Unit testing: test one module of the software.
- Integration testing: test the interfaces between different modules in the software.
- System testing: test the complete system.
- Validation testing: test if the software system fulfills the requirements.
- Acceptance testing: the client test whether the system is what s/he wants.
- Regression testing: after a change in the software test whether a new error has been introduced.
- Stress testing: test the system under a high load
- Load testing: test the response of the system under a normal load of work.

Software Testing: Automatization

Test case design

Automatic test case generation

Test case run



nit, Ea
Unit,
Selen...

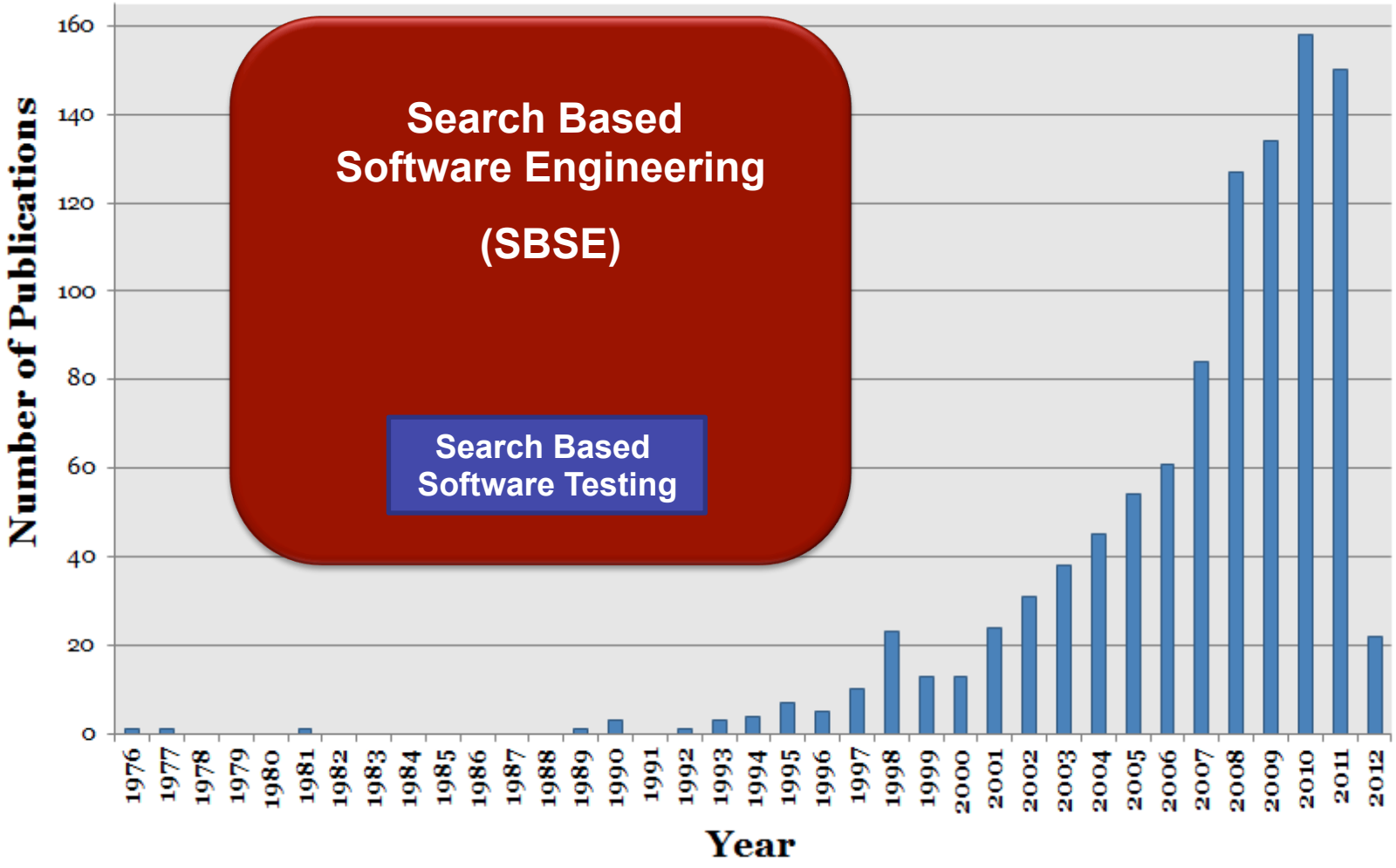
Search Techniques

Check of results

=

Search Based Software Testing

Search Based Software Engineering



Search Based Software Engineering

Special Issue on SBSE in Journal of Systems and Software (JSS)

Deadline: November 15th

Guest editors:

M. Harman & F. Chicano

http://neo.lcc.uma.es/mase/index.php/jss-sbse

Our Research on SBSE

- Software Project Scheduling
- White-box Software Testing
- Testing of Concurrent Systems (based on Model Checking)
- Test Sequences for Functional Testing
- Test Suite Minimization in Regression Testing
- Software Product Lines Testing
- Prioritized Pairwise Combinatorial Interaction Testing
- Testing Complexity

Test Suite Minimization in Regression Testing

Test Suite Minimization

Given:

- A set of test cases $T = \{t_1, t_2, ..., t_n\}$
- A set of program elements to be covered (e.g., branches) $E = \{e_1, e_2, ..., e_k\}$
- A coverage matrix

M=

	e_1	e_2	e_3	...	e_k
t_1	1	0	1	…	1
t_2	0	0	1	…	0
…	…	…	…	…	…
t_n	1	1	0	…	0

$$m_{ij} = \begin{cases} 1 & \text{if element } e_j \text{ is covered by test } t_i \\ 0 & \text{otherwise} \end{cases}$$

Find a subset of tests $X \subseteq T$ maximizing coverage and minimizing the testing cost

$$minimize \quad cost(X) = \sum_{\substack{i=1 \\ t_i \in X}}^{n} c_i$$

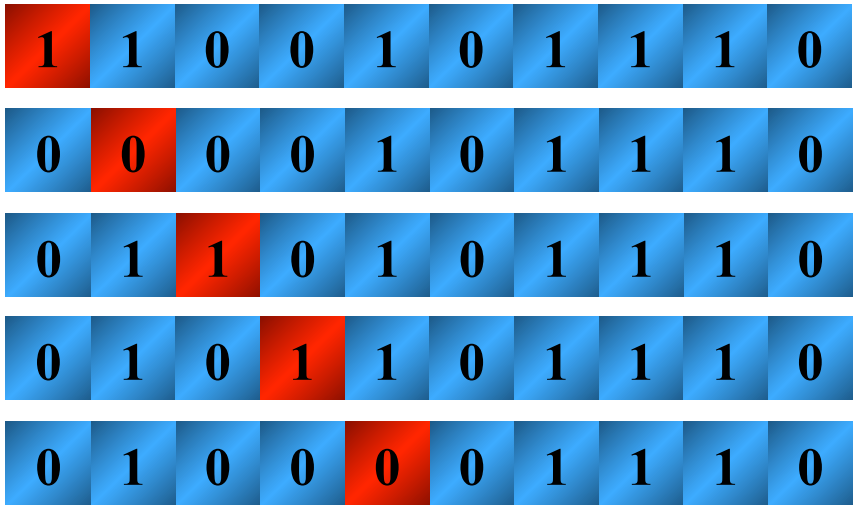
Yoo & Harman

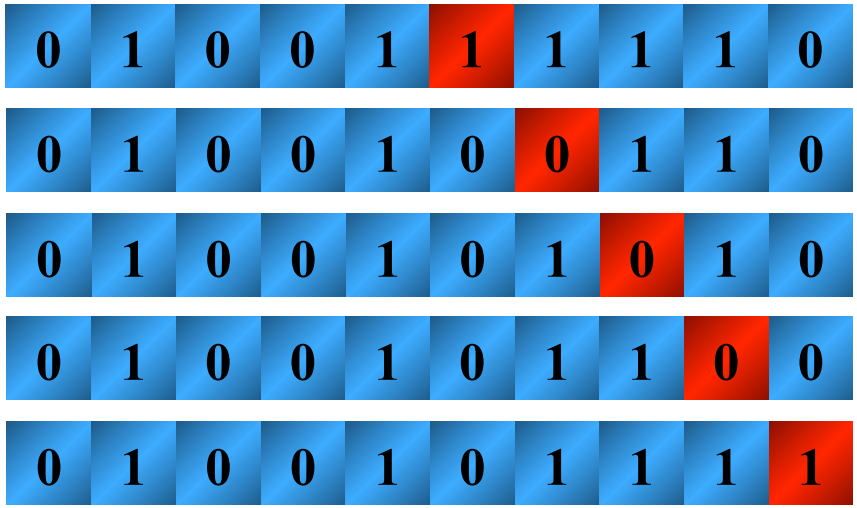
$$maximize \quad cov(X) = |\{e_j \in \mathcal{E} | \exists t_i \in X \text{ with } m_{ij} = 1\}|$$

Binary Search Space

- The set of solutions is the set of binary strings with length *n*

- Neighborhood used: one-change neighborhood
  - Two solutions *x* and *y* are neighbors iff *Hamming(x,y)=1*





Elementary Landscapes: Characterizations

- **An elementary landscape is a landscape for which**

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = \alpha f(x) + \beta \quad \forall x \in X$$

Depend on the problem/instance

Linear relationship

where

$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} \overset{\text{def}}{=} \frac{1}{d} \sum_{y \in N(x)} f(y)$$

$$\bar{f} = \frac{1}{|X|} \sum_{y \in X} f(y)$$

- **Grover's wave equation**

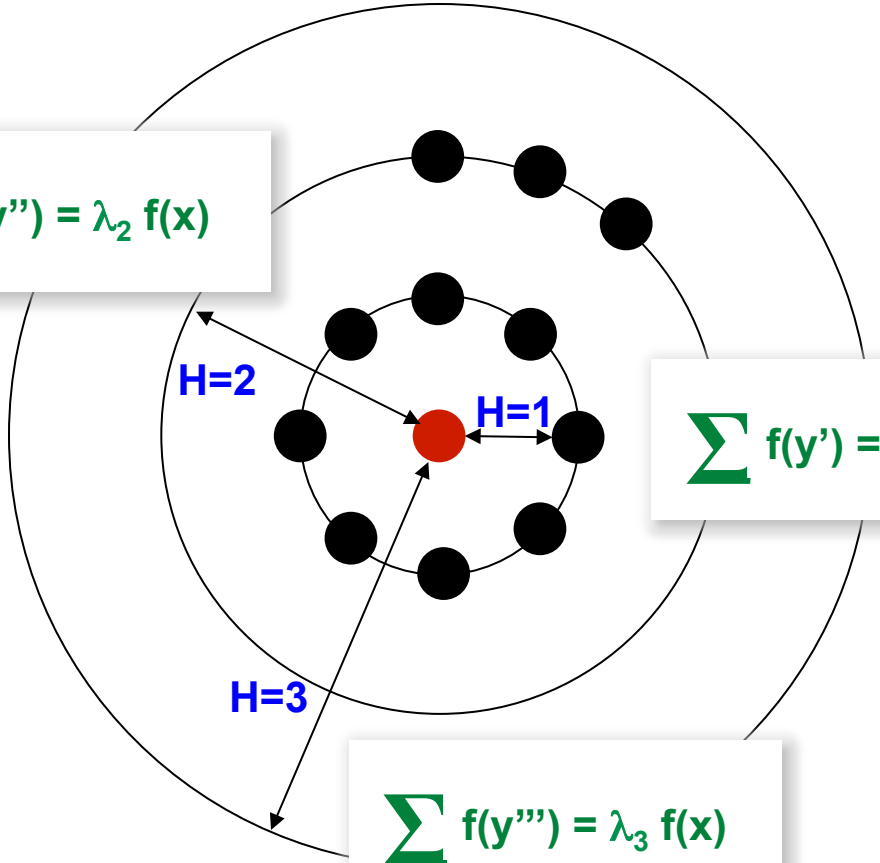
$$\underset{y \in N(x)}{\text{avg}}\{f(y)\} = f(x) + \frac{\lambda}{d}(\bar{f} - f(x))$$

Eigenvalue

$$\alpha = 1 - \frac{\lambda}{d} \qquad \beta = \frac{\lambda}{d}\bar{f}$$

Spheres around a Solution

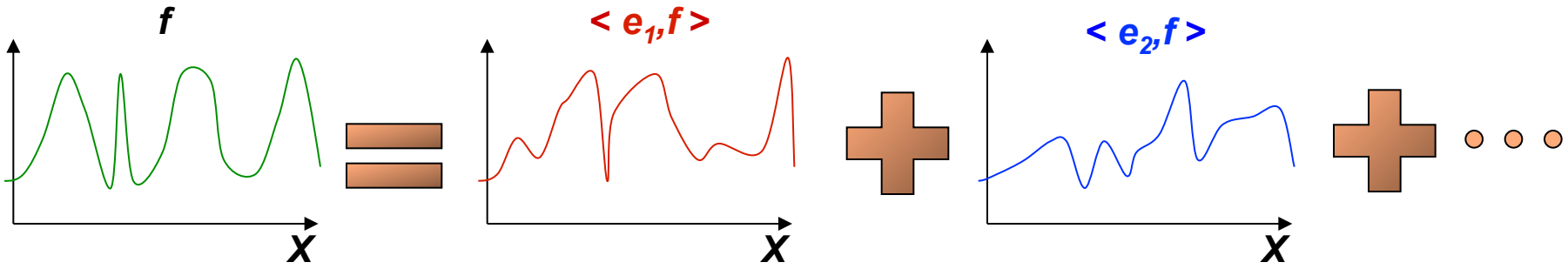
- If *f* is elementary, the average of *f* in any sphere and ball of any size around *x* is a linear expression of f(x)!!!



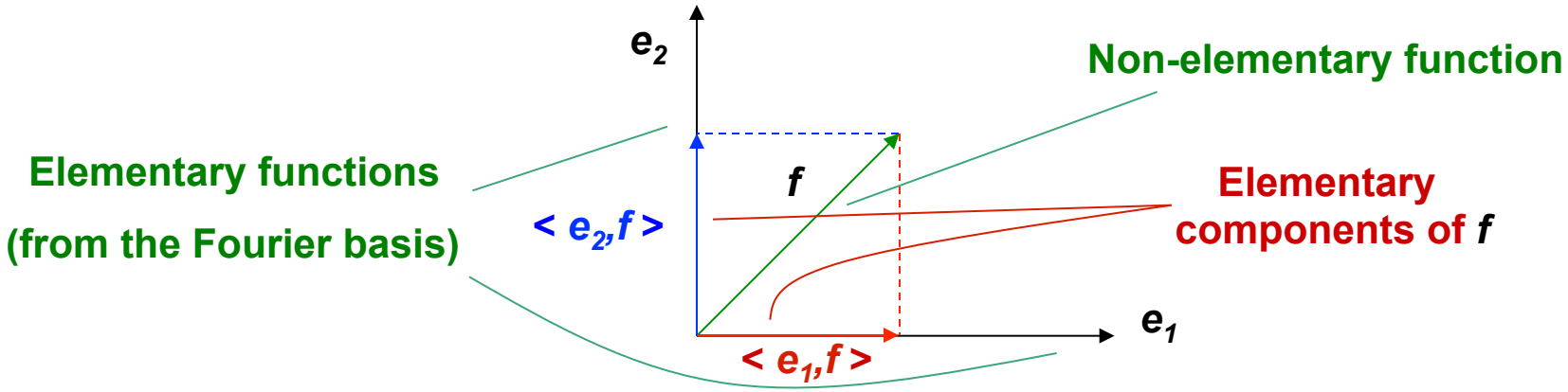
Sutton
Whitley
Langdon

Landscape Decomposition

- What if the landscape is not elementary?
- Any landscape can be written as the sum of elementary landscapes



- There exists a set of eigenfunctions of Δ that form a basis of the function space (Fourier basis)



Elementary Landscape Decomposition of *f*

- **The elementary landscape decomposition of**

$$f(x) = cov(x) - c \cdot cost(x)$$

Computable in O(nk)

is

Tests that cover e_i

$$f^{(0)}(x) = \sum_{i=1}^{k} \left(1 - \frac{1}{2^{|V_i|}}\right) - c \cdot \frac{n}{2}$$

← **constant expression**

$$f^{(1)}(x) = -\sum_{i=1}^{k} \frac{1}{2^{|V_i|}} (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-1,n_1^{(i)}} - c \cdot \left(ones(x) - \frac{n}{2}\right)$$

Krawtchouk matrix

$$f^{(p)}(x) = -\sum_{i=1}^{k} \frac{1}{2^{|V_i|}} (-1)^{n_1^{(i)}} \mathcal{K}^{|V_i|}_{|V_i|-p,n_1^{(i)}} \quad \text{where } 1 < p \leq n$$

Tests in the solution that cover e_i

F. Chicano et al., SSBSE 2011

Elementary Landscape Decomposition of f^2

- **The elementary landscape decomposition of f^2 is**

Computable in $O(nk^2)$

$$(f^2)^{(0)}(x) = \beta^2 + \frac{c^2}{4}n - \sum_{i=1}^{k} \frac{c|V_i| + 2\beta}{2^{|V_i|}} + \sum_{i,i'=1}^{k} \frac{1}{2^{|V_i \cup V_{i'}|}}$$

$$\beta = k - cn/2$$

Number of tests that cover e_i or $e_{i'}$

$$(f^2)^{(p)}(x) = -\sum_{i=1}^{k} \left(\frac{(c|V_i| + 2\beta)(-1)^{n_1^{(i)}}}{2^{|V_i|}} \mathcal{K}^{|V_i|}_{|V_i|-p,n_1^{(i)}} \right) \qquad p > 2$$

$$+ \sum_{i,i'=1}^{k} \left(\frac{(-1)^{n_1^{(i \vee i')}}}{2^{|V_i \cup V_{i'}|}} \mathcal{K}^{|V_i \cup V_{i'}|}_{|V_i \cup V_{i'}|-p,n_1^{(i \vee i')}} \right)$$

Number of tests in the solution that cover e_i or $e_{i'}$

$$- c \sum_{i=1}^{k} \frac{(-1)^{n_1^{(i)}}}{2^{|V_i|}} \mathcal{K}^{|V_i|}_{|V_i|-p+1,n_1^{(i)}} \left(n - 2ones(x) - |V_i| + 2n_1^{(i)} \right)$$

Guarded Local Search

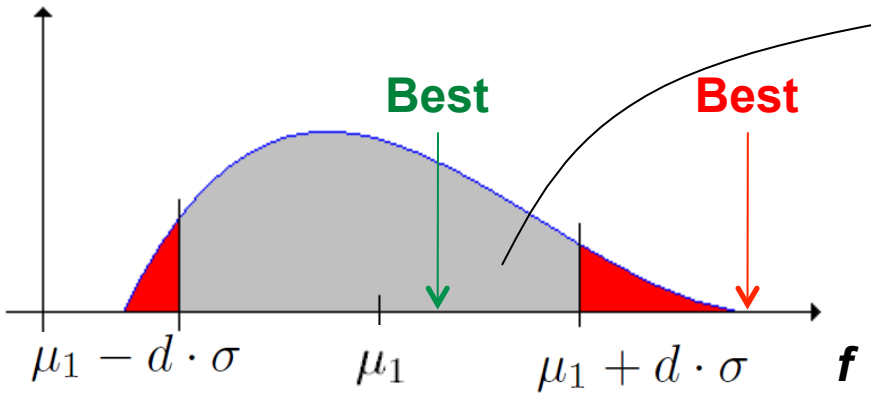
- With the Elementary Landscape Decomposition (ELD) we can compute:

$$\mu_c = \underset{y|\mathcal{H}(y,x)=r}{\text{avg}}\{f^c(y)\} = \binom{n}{r}^{-1} \sum_{p=0}^{n} \mathcal{K}_{r,p}^{(n)} \left(f^c\right)^{(p)}(x)$$

- With the ELD of *f* and f^2 we can compute for any sphere and ball around a solution:

μ_1 : the average $\qquad \sigma = \sqrt{\mu_2 - \mu_1^2}$: the standard deviation

- Distribution of values around the average



Chebyshev inequality

$$100\left(1 - \frac{1}{d^2}\right)\%$$

At least 75% of the samples are in the interval

$$[\mu_1 - 2\sigma, \mu_1 + 2\sigma]$$

Guarded Local Search: Experimental Setting

- **Steady state genetic algorithm**: bit-flip (p=0.01), one-point crossover, elitist replacement
  - **GA** (no local search)
  - **GLSr** (guarded local search up to radius r)
  - **LSr** (always local search in a ball of radius r)

- Instances from the **Software-artifact Infrastructure Repository (SIR)**
  - printtokens
  - printtokens2
  - schedule
  - schedule2
  - totinfo
  - replace

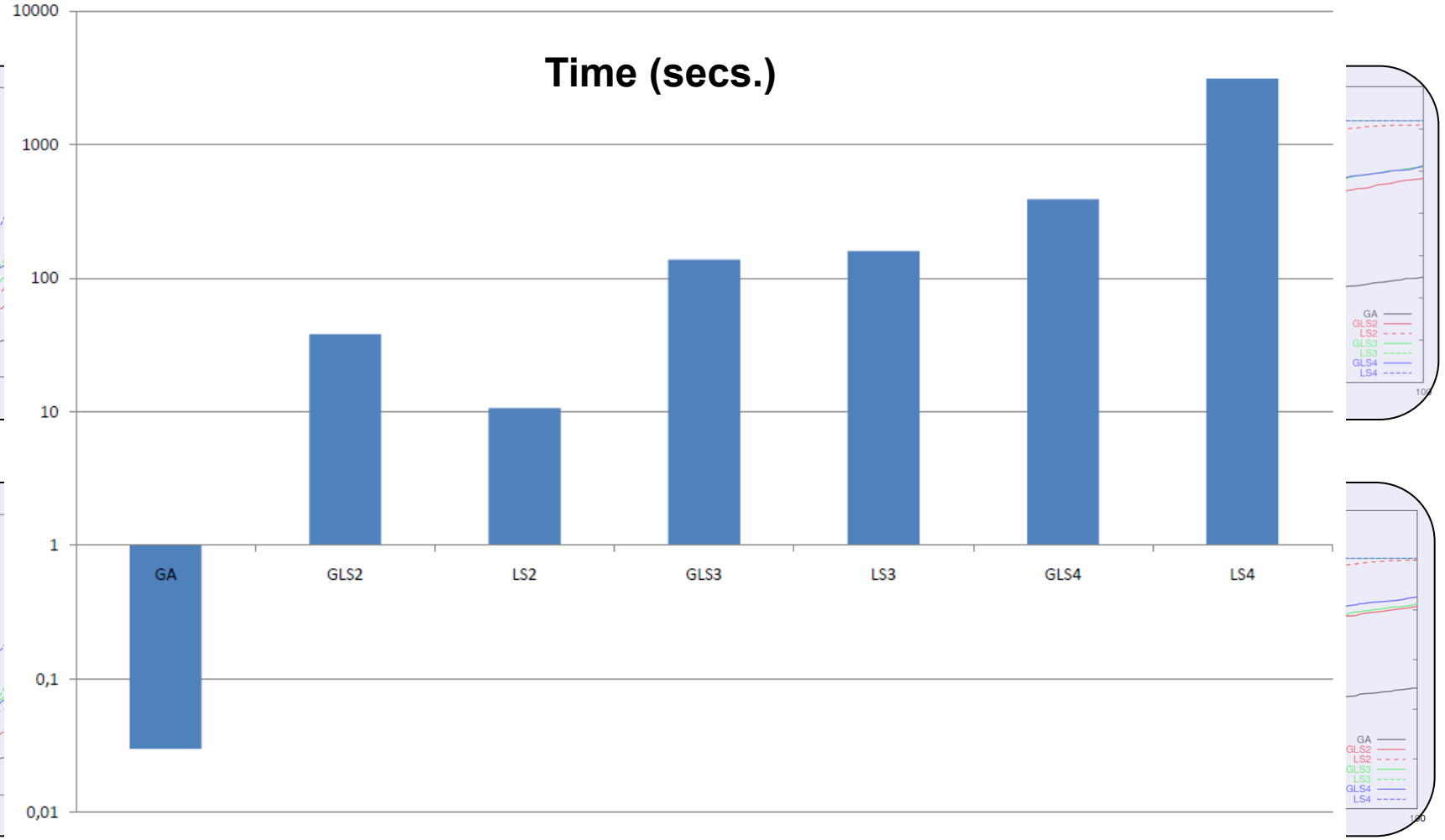
Oracle cost c=1..5

n=100 test cases

k=100-200 items to cover

100 independent runs

Guarded Local Search: Results



NP-hard Problems

In many papers we can read…

"Our optimization problem is NP-hard, and for this reason we use…

- **Metaheuristic techniques**
- **Heuristic algorithms**
- **Stochastic algorithms**

… which do not ensure an optimal solution but they are able to find good solutions in a reasonable time."

As far as we know: no efficient (polynomial time) algorithm exists for solving NP-hard problems

But we know inefficient algorithms (at least exponential time)

The SATisfiability Problem

Can we find an assignment of boolean values (true and false) to the variables such that all the formulas are satisfied?

$$\neg A \wedge (B \vee C)$$
$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$
$$A \vee B$$

The first NP-complete problem (Stephen Cook, 1971)

If it can be solved efficiently (polynomial time) then P=NP

The known algorithms solve this problem in exponential time (worst case)

State-of-the-art algorithms in SAT

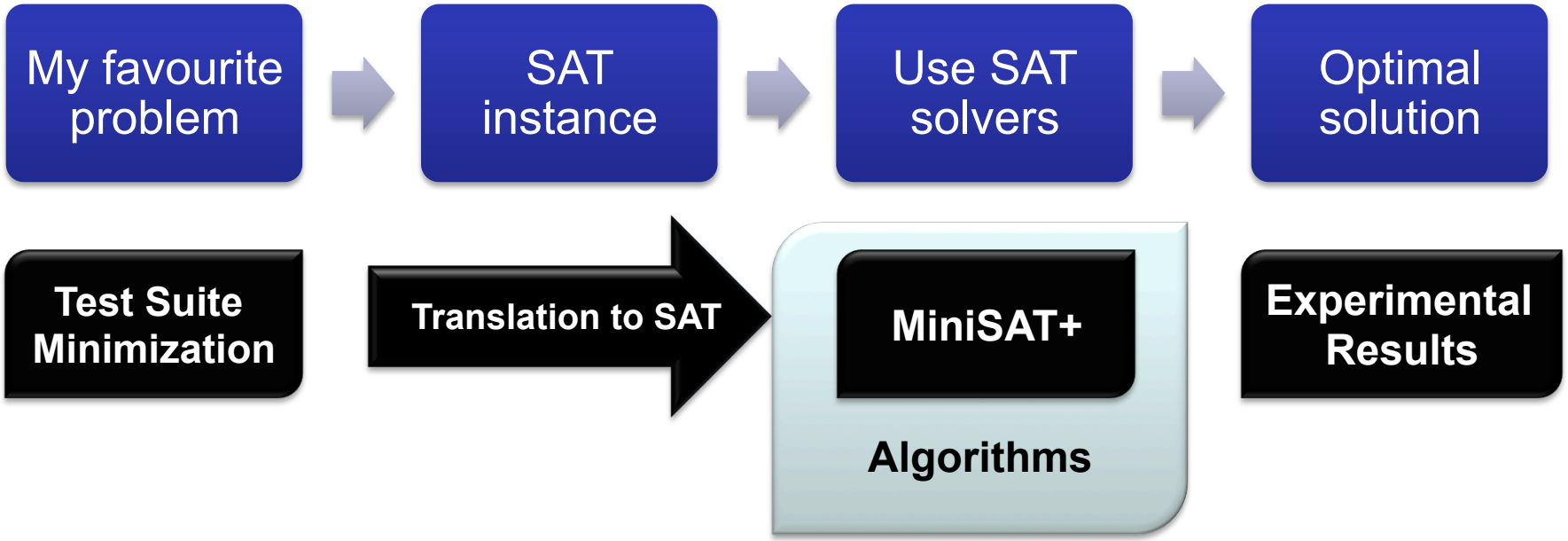
Nowadays, SAT solvers can solve instances with 500 000 boolean variables

This means a search space of $2^{500\,000} \approx 10^{150514}$

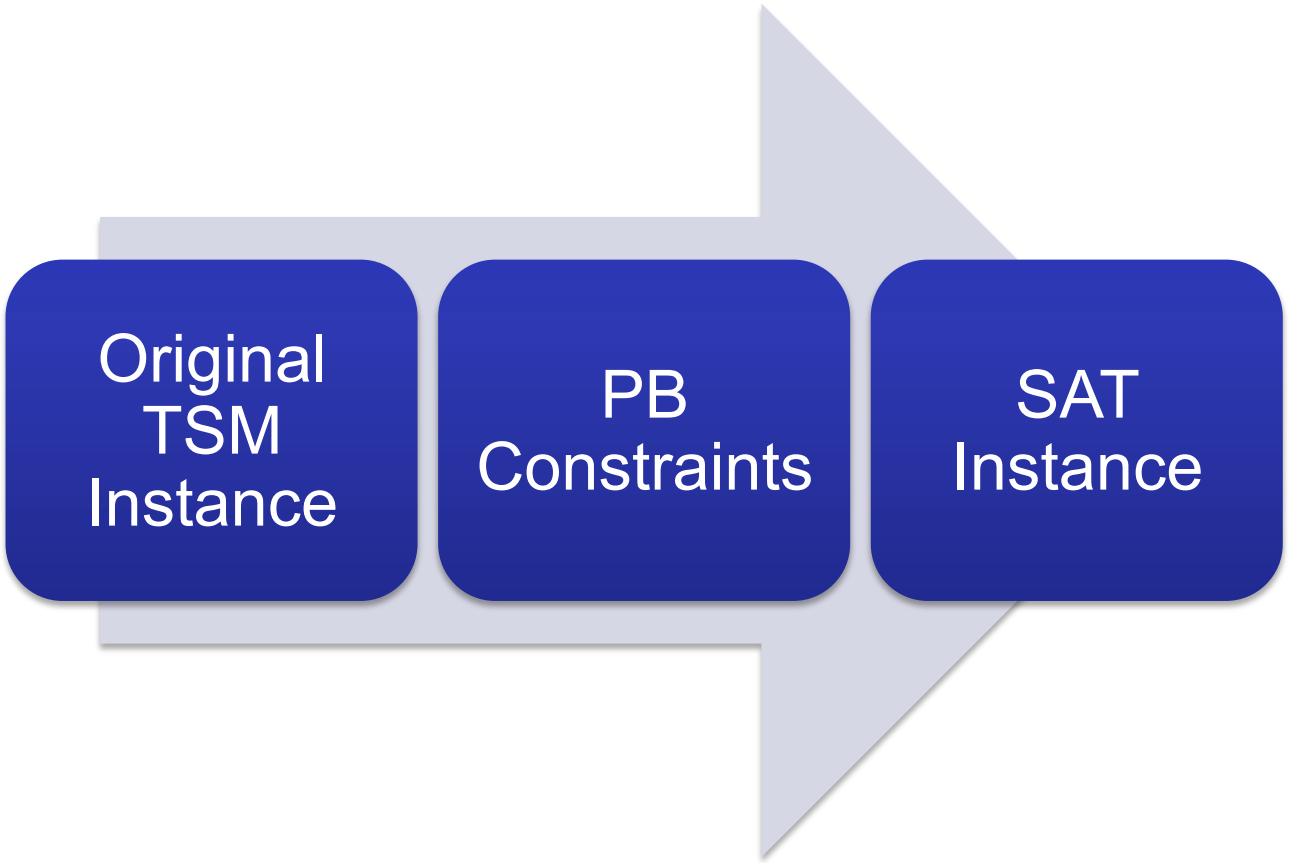
The SATisfiability Problem

Main research question:

Can we use the advances of SAT solvers to solve optimization algorithms up to optimality?



Outline



F. Arito et al., SSBSE 2012

Pseudo-Boolean Constraints

A Pseudo-Boolean (PB) constraint has the form:

$$\sum_{i=1}^{n} a_i x_i \odot B$$

where

$$\odot \in \{<, \leq, =, \neq, >, \geq\}$$

$$a_i, B \in \mathbb{Z} \quad x_i \in \{0, 1\}$$

Can be translated to SAT instances (usually efficient)

Are a higher level formalism to specify a decision problem

Can be the input for MiniSAT+

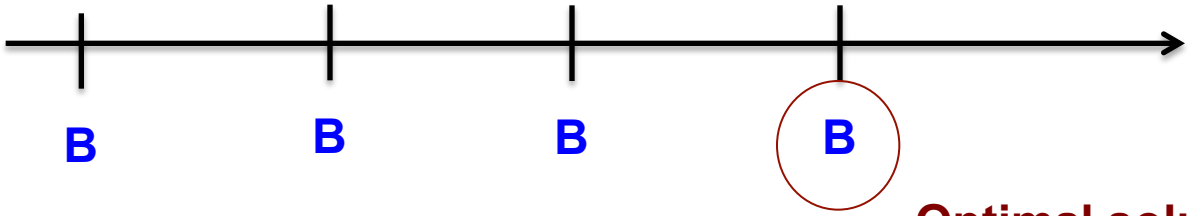
Translating Optimization to Decision Problems

Let us assume we want to minimize f(x)

Check **Check** **Check** **Check**

$$f(x) \leq B \quad f(x) \leq B \quad f(x) \leq B \quad f(x) \leq B$$

no **no** **no** **yes**



Optimal solution found

The same can be done with multi-objective problems, but we need more PB constraints

$$f_1(y) \leq B_1 \qquad f_2(y) \leq B_2 \qquad \cdots \qquad f_m(y) \leq B_m$$

PB Constraints for the TSM Problem

M=

	e_1	e_2	e_3	...	e_m
t_1	1	0	1	…	1
t_2	0	0	1	…	0
…	…	…	…	…	…
t_n	1	1	0	…	0

$$m_{ij} = \begin{cases} 1 & \text{if element } e_j \text{ is covered by test } t_i \\ 0 & \text{otherwise} \end{cases}$$

$$e_j \leq \sum_{i=1}^{n} m_{ij} t_i \leq n \cdot e_j \qquad 1 \leq j \leq m$$

Cost

$$\sum_{i=1}^{n} c_i t_i \leq B$$

Coverage

$$\sum_{j=1}^{m} e_j \geq P$$

Example

	e_1	e_2	e_3	e_4
t_1	1	0	1	0
t_2	1	1	0	0
t_3	0	0	1	0
t_4	1	0	0	0
t_5	1	0	0	1
t_6	0	1	1	0

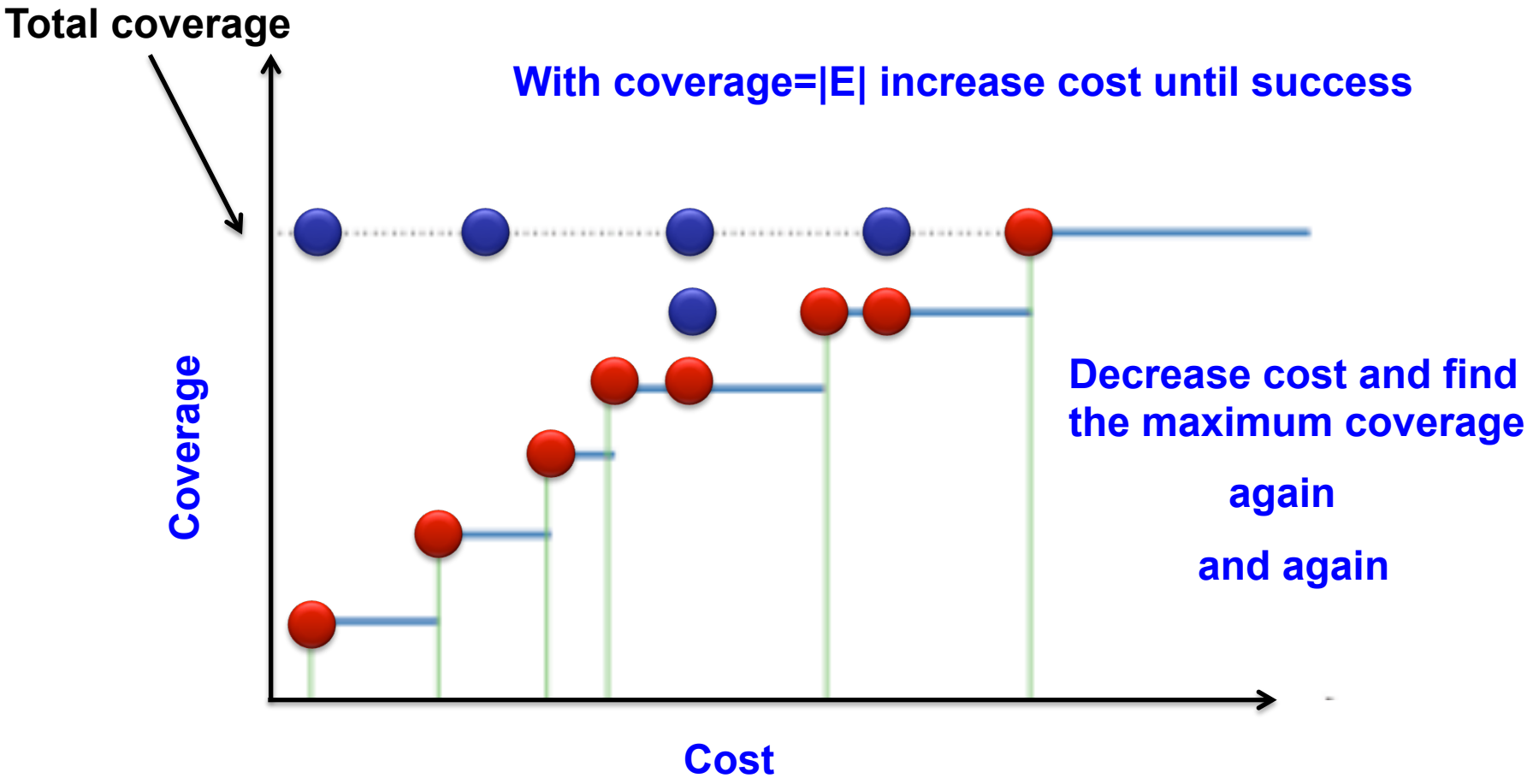
Bi-objective problem

$$\begin{cases} e_1 \leq t_1 + t_2 + t_4 + t_5 \leq 6e_1 \\ e_2 \leq t_2 + t_6 \leq 6e_2 \\ e_3 \leq t_1 + t_3 + t_6 \leq 6e_3 \\ e_4 \leq t_5 \leq 6e_4 \\ t_1 + t_2 + t_3 + t_4 + t_5 + t_6 \leq B \\ e_1 + e_2 + e_3 + e_4 \geq P \end{cases}$$

Single-objective problem (total coverage)

$$\begin{cases} t_1 + t_2 + t_4 + t_5 \geq 1 \\ t_2 + t_6 \geq 1 \\ t_1 + t_3 + t_6 \geq 1 \\ t_5 \geq 1 \\ t_1 + t_2 + t_3 + t_4 + t_5 + t_6 \leq B \end{cases}$$

Algorithm for Solving the 2-obj TSM



TSM Instances

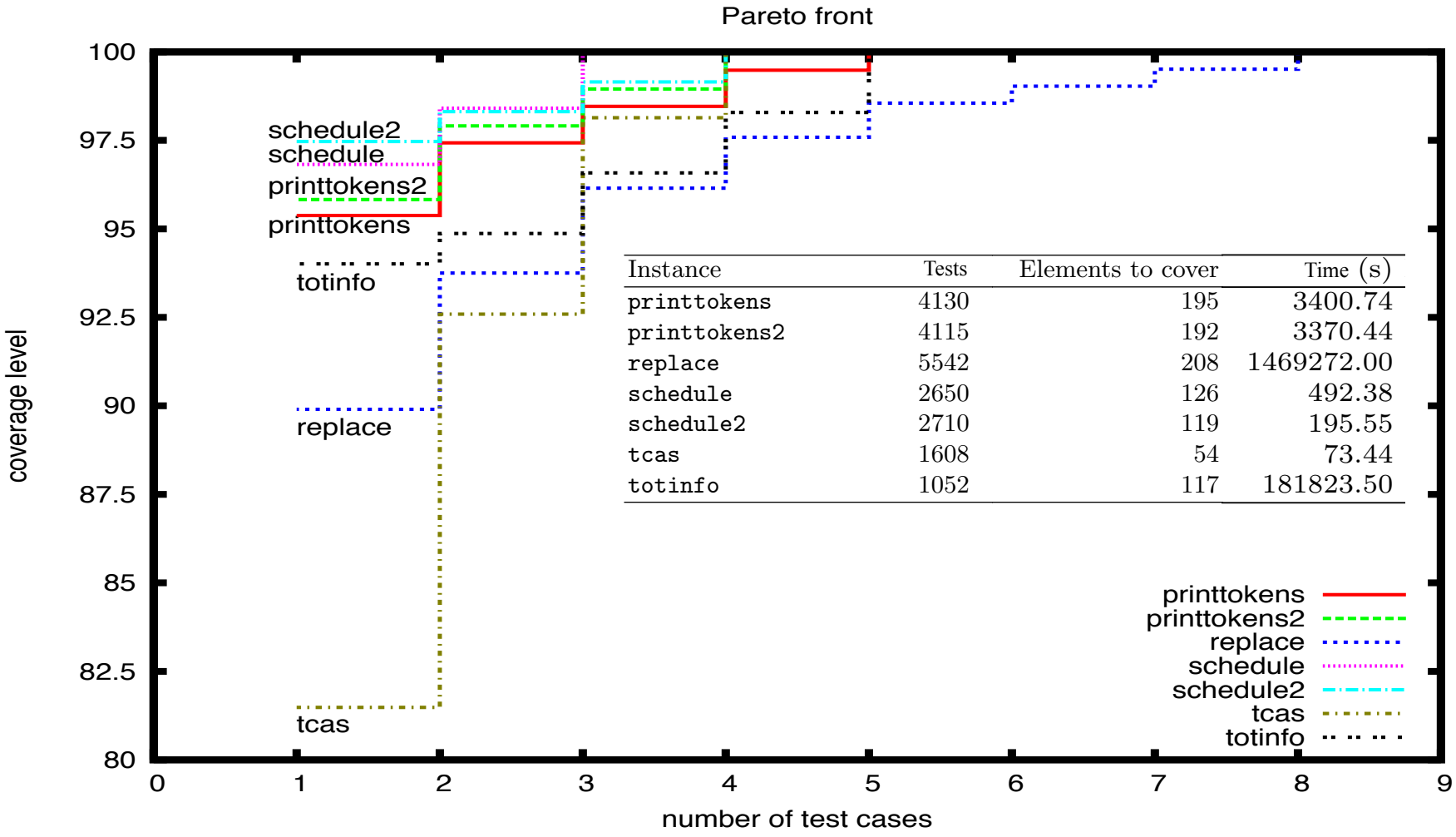
Instances from the Software-artifact Infrastructure Repository (SIR)

`http://sir.unl.edu/portal/index.php`

Instance	Tests	Elements to cover
`printtokens`	4130	195
`printtokens2`	4115	192
`replace`	5542	208
`schedule`	2650	126
`schedule2`	2710	119
`tcas`	1608	54
`totinfo`	1052	117

Cost of each test: 1

Pareto Front



Instance	Tests	Elements to cover	Time (s)
printtokens	4130	195	3400.74
printtokens2	4115	192	3370.44
replace	5542	208	1469272.00
schedule	2650	126	492.38
schedule2	2710	119	195.55
tcas	1608	54	73.44
totinfo	1052	117	181823.50

Pareto Front

Instance	Elements	Tests	Coverage	Solution
printtokens	195	5	100%	$(t_{2222}, t_{2375}, t_{3438}, t_{4100}, t_{4101})$
	194	4	99.48%	$(t_{1908}, t_{2375}, t_{4099}, t_{4101})$
	192	3	98.46%	$(t_{1658}, t_{2363}, t_{4072})$
	190	2	97.43%	(t_{1658}, t_{3669})
	186	1	95.38%	(t_{2597})
printtokens2	192	4	100%	$(t_{2521}, t_{2526}, t_{4085}, t_{4088})$
	190	3	98.95%	$(t_{457}, t_{3717}, t_{4098})$
	188	2	97.91%	(t_{2190}, t_{3282})
	184	1	95.83%	(t_{3717})
replace	208	8	100%	$(t_{306}, t_{410}, t_{653}, t_{1279}, t_{1301}, t_{3134}, t_{4057}, t_{4328})$
	207	7	99.51%	$(t_{309}, t_{358}, t_{653}, t_{776}, t_{1279}, t_{1795}, t_{3248})$
	206	6	99.03%	$(t_{275}, t_{290}, t_{1279}, t_{1938}, t_{2723}, t_{2785})$
	205	5	98.55%	$(t_{426}, t_{1279}, t_{1898}, t_{2875}, t_{3324})$
	203	4	97.59%	$(t_{298}, t_{653}, t_{3324}, t_{5054})$
	200	3	96.15%	$(t_{2723}, t_{2901}, t_{3324})$
	195	2	93.75%	(t_{358}, t_{5387})
	187	1	89.90%	(t_{358})
schedule	126	3	100%	$(t_{1403}, t_{1559}, t_{1564})$
	124	2	98.41%	(t_{1570}, t_{1595})
	122	1	96.82%	(t_{1572})
schedule2	119	4	100%	$(t_{2226}, t_{2458}, t_{2462}, t_{2681})$
	118	3	99.15%	$(t_{101}, t_{1406}, t_{2516})$
	117	2	98.31%	(t_{2461}, t_{2710})
	116	1	97.47%	(t_{1584})
tcas	54	4	100%	$(t_5, t_{1191}, t_{1229}, t_{1608})$
	53	3	98.14%	$(t_{13}, t_{25}, t_{1581})$
	50	2	92.59%	(t_{72}, t_{1584})
	44	1	81.48%	(t_{217})
totinfo	117	5	100%	$(t_{62}, t_{118}, t_{218}, t_{1000}, t_{1038})$
	115	4	98.29%	$(t_{62}, t_{118}, t_{913}, t_{1016})$
	113	3	96.58%	$(t_{65}, t_{216}, t_{913})$
	111	2	94.87%	(t_{65}, t_{919})
	110	1	94.01%	(t_{179})

Reduction in the Number of Test Cases

Since we are considering cost 1 for the tests, we can apply an a priori reduction in the original test suite

	e_1	e_2	e_3	...	e_m
t_1	1	0	0	…	1
t_2	1	0	1	…	1
…	…	…	…	…	…
t_n	1	1	0	…	0

Test t_1 can be removed

Instance	Original Size	Reduced Size	Elements to cover
`printtokens`	4130	40	195
`printtokens2`	4115	28	192
`replace`	5542	215	208
`schedule`	2650	4	126
`schedule2`	2710	13	119
`tcas`	1608	5	54
`totinfo`	1052	21	117

Results with the Reduction

The optimal Pareto Front for the reduced test suite can be found from 200 to 180 000 times faster

	Original (s)	Reduced (s)
printtokens	3400.74	2.17
printtokens2	3370.44	1.43
replace	1469272.00	345.62
schedule	492.38	0.24
schedule2	195.55	0.27
tcas	73.44	0.33
totinfo	181823.50	0.96

Software Product Lines Testing

R. Lopez-Herrejon et al., ICSM 2013

Software Product Lines

A product line is a set of related products developed from a shared set of assets

- **The products have similar characteristics**
- **The products have unique characteristics**

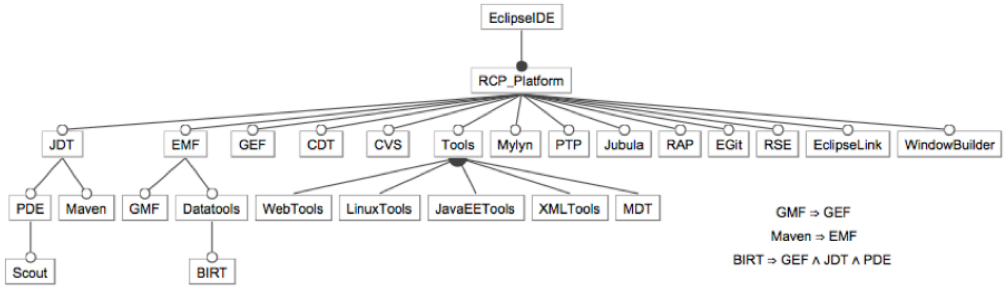
Advantages

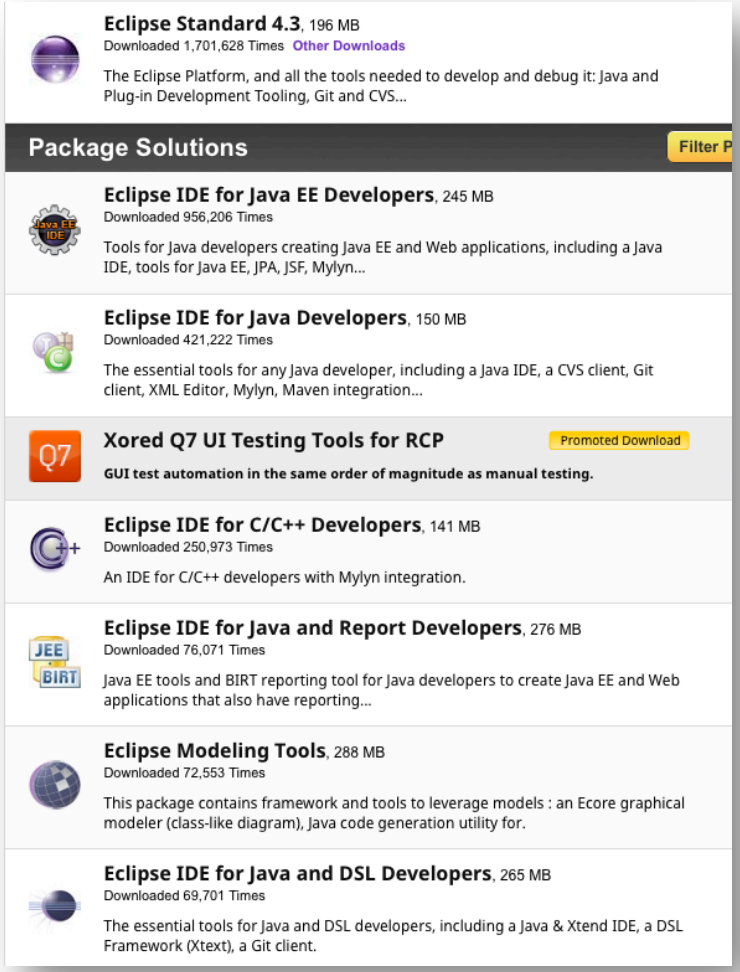
- **Support customization**
- **Improves reuse**
- **Reduce time to market**

Software Product Lines

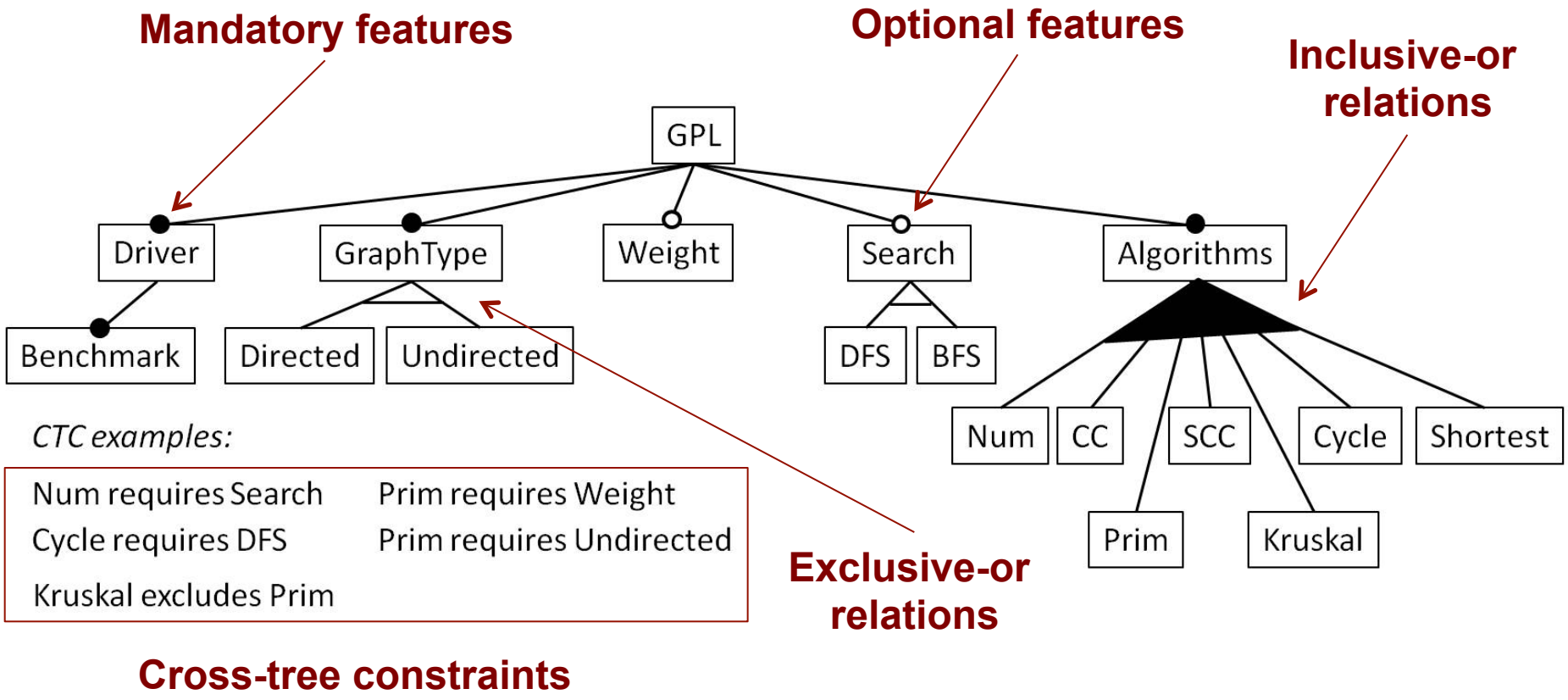
In Software Product Lines the product is Software

They are modelled using Feature Models





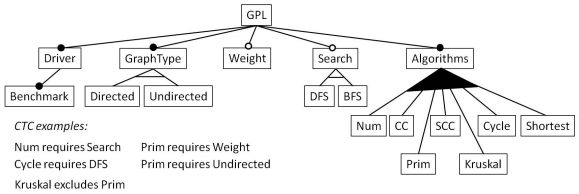
Feature Models



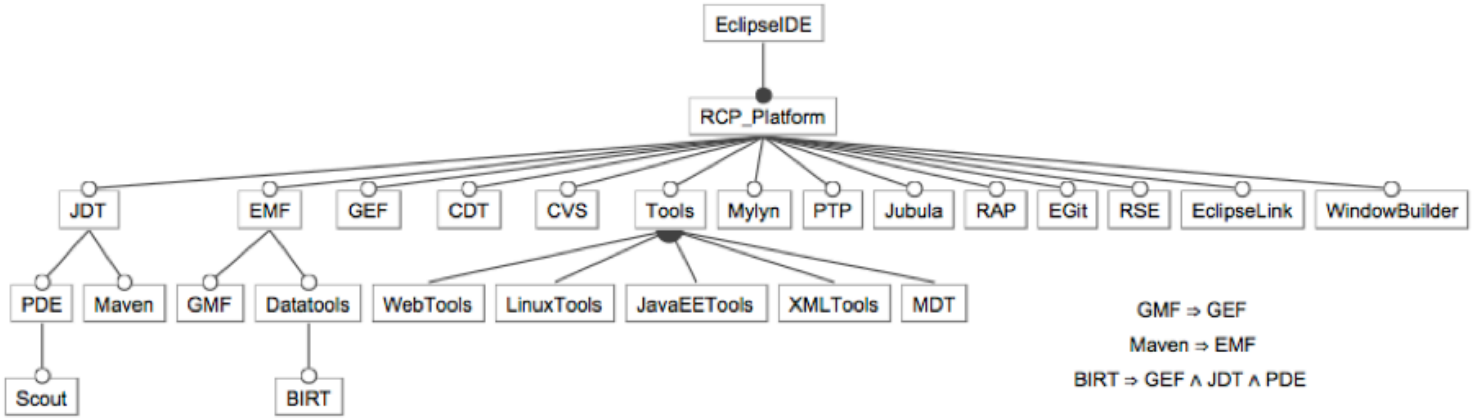
Graph Product Line Feature Model

Testing of Software Product Lines

The GPL Feature Model is small: 73 distinct products



But the number of products grows exponentially with the number of features…



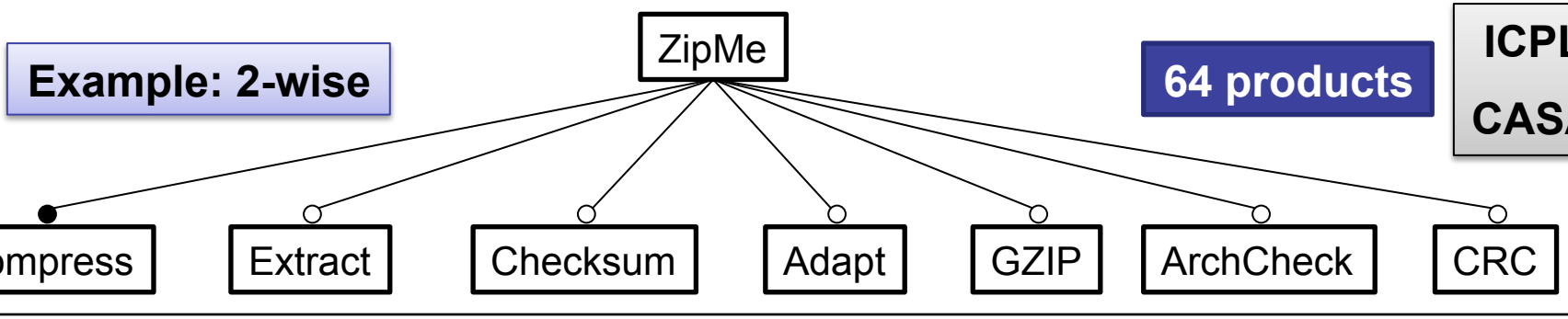
… and testing each particular product is not viable

Testing of SPLs: Combinatorial Interaction Testing

Assuming each features has been tested in isolation, most of the defects come from the interaction between features

Combinatorial Interaction Testing consists in selecting the minimum number of products that covers all *t*-wise interactions (*t*-wise coverage).

Example: 2-wise



64 products

ICPL

CASA

	Compress	Extract	Checksum	Adapt	GZIP	ArchCheck	CRC
1	✔		✔	✔	✔	✔	
2	✔	✔	✔				
3	✔	✔			✔	✔	
4	✔	✔		✔			✔
5	✔					✔	✔
6	✔		✔		✔		✔

Testing of SPLs: Multi-Objective Formulation

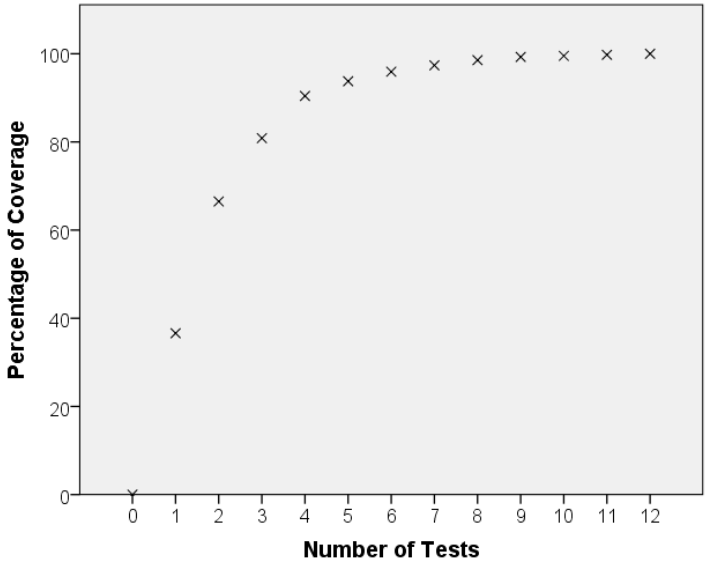
If we don't have the resources to run all the tests, which one to choose?

Multi-objective formulation:

minimize the number of products

maximize the coverage (t-wise interactions)

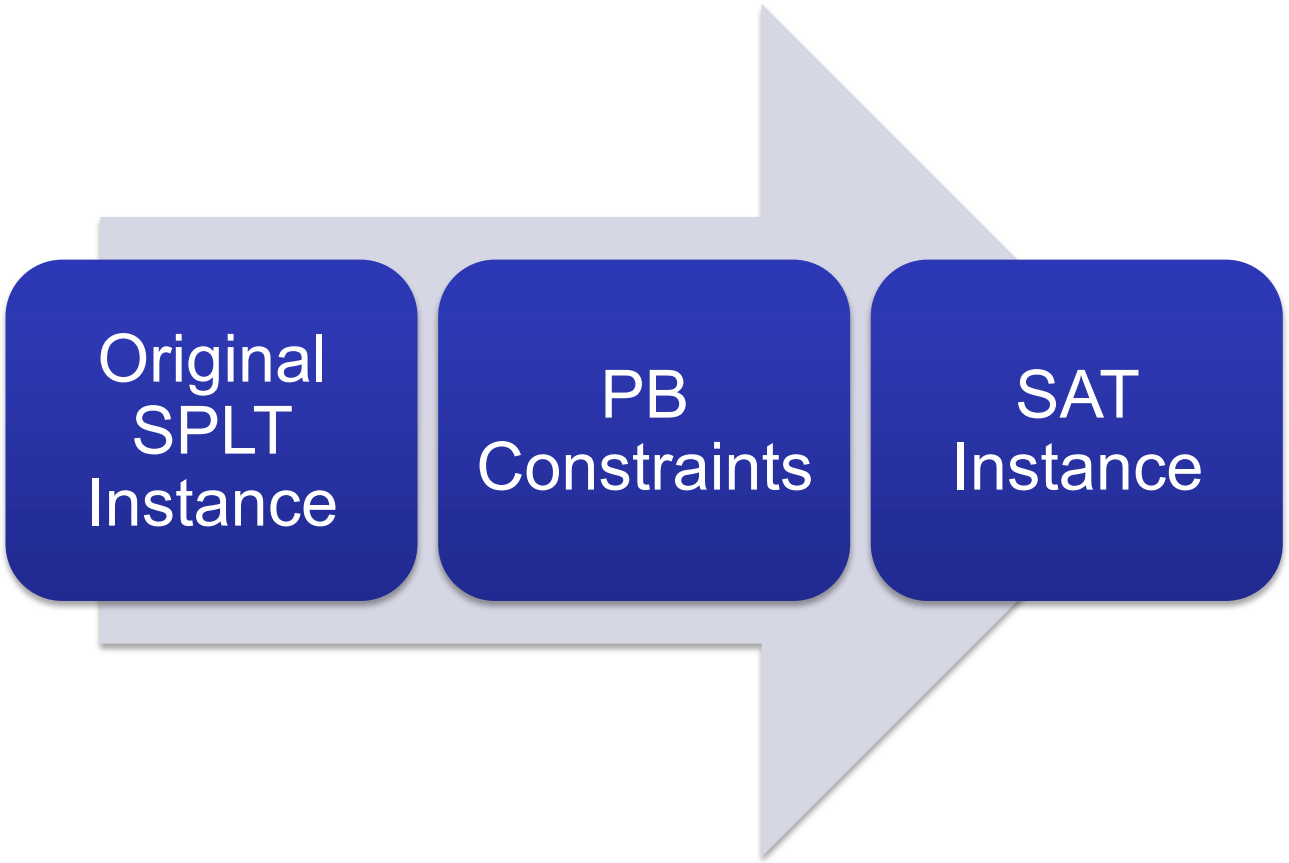
The solution is not anymore a table of products, but a Pareto set



GPL

2-wise interactions

Testing of SPLs: Approach



Testing of SPLs: Approach

Modelling SPLT using PseudoBoolean constraints

Variable	Meaning
$x_{p,i}$	Presence of feature i in product p
$c_{p,i,j,k}$	Product p covers the pair (i,j) with signature k
$d_{i,j,k}$	The pair (i,j) with signature k is covered by some product

k takes values 0, 1, 2 and 3.

All the variables are boolean {0,1}

The values of the signature are:

- **00 (both unselected)**
- **10 (only first selected)**
- **01 (only second selected)**
- **11 (both selected)**

Testing of SPLs: Approach

Equations of the model

- **For each product *p***
  - **Constraints imposed by the Feature Model**
- **For each product *p* and pair of features *i* and *j***

$$2c_{p,i,j,3} \leq x_{p,i} + x_{p,j} \leq 1 + c_{p,i,j,3}$$
$$2c_{p,i,j,2} \leq x_{p,i} + (1 - x_{p,j}) \leq 1 + c_{p,i,j,3}$$
$$2c_{p,i,j,1} \leq (1 - x_{p,i}) + x_{p,j} \leq 1 + c_{p,i,j,3}$$
$$2c_{p,i,j,0} \leq (1 - x_{p,i}) + (1 - x_{p,j}) \leq 1 + c_{p,i,j,3}$$

Testing of SPLs: Approach

Equations of the model (cont.)

- **For each pair of features *i* and *j* and signature *k***

$$d_{i,j,k} \leq \sum_{p} c_{p,i,j,k} \leq n d_{i,j,k}$$

  - ***n*** **is the number of products**
- **Objective: maximize coverage**

$$max : \sum_{i,j,k} d_{i,j,k}$$

Testing of SPLs: Approach

Algorithm 1 Algorithm for obtaining the optimal Pareto set.

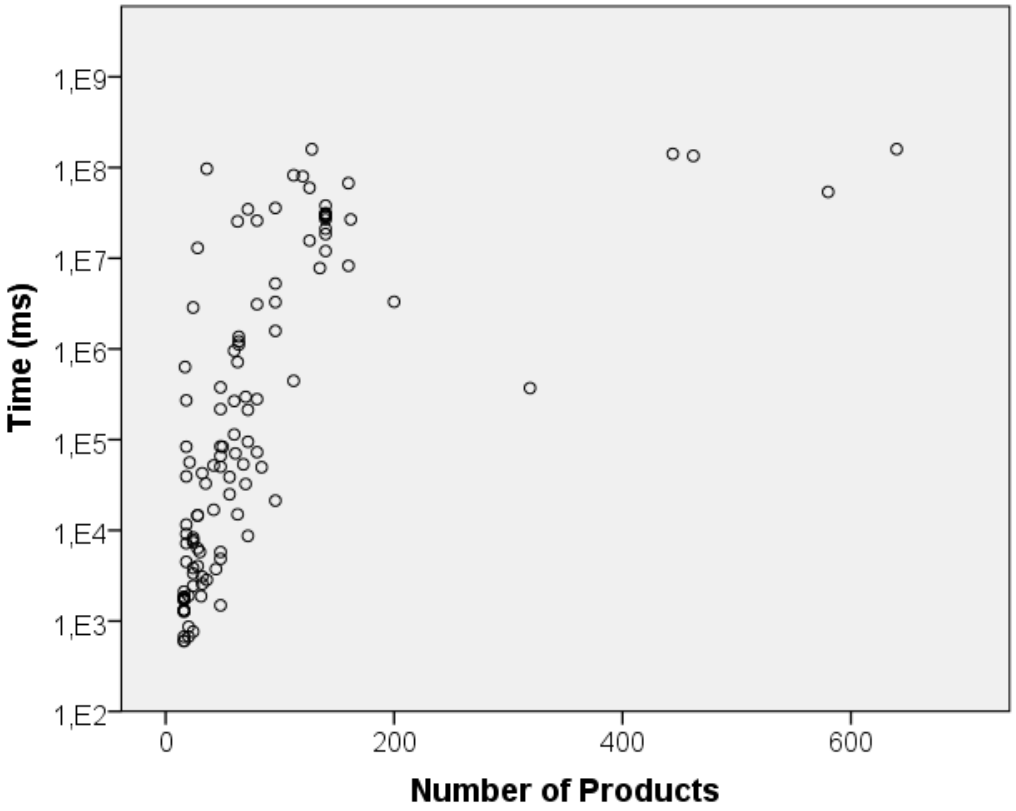
$optimal\_set \leftarrow \{\emptyset\}$;
$cov[0] \leftarrow 0$;
$cov[1] \leftarrow C_2^f$;
$sol \leftarrow$arbitraryValidSolution(fm);
$i \leftarrow 1$;
while $cov[i] \neq cov[i-1]$ **do**
  $optimal\_set \leftarrow optimal\_set \cup \{sol\}$;
  $i \leftarrow i + 1$;
  $m \leftarrow$prepareMathModel(fm,i);
  $sol \leftarrow$solveMathModel(m);
  $cov[i] \leftarrow |sol|$;
end while

Testing of SPLs: Results

Experiments on 118 feature models taken from

SPLOT repository (http://www.splot-research.org)

SPL Conqueror (http://wwwiti.cs.uni-magdeburg.de/~nsiegmun/SPLConqueror/)



16 to 640 products

Intel Core2 Quad Q9400

2.66 GHz, 4 GB

Prioritized Pairwise Combinatorial Interaction Testing

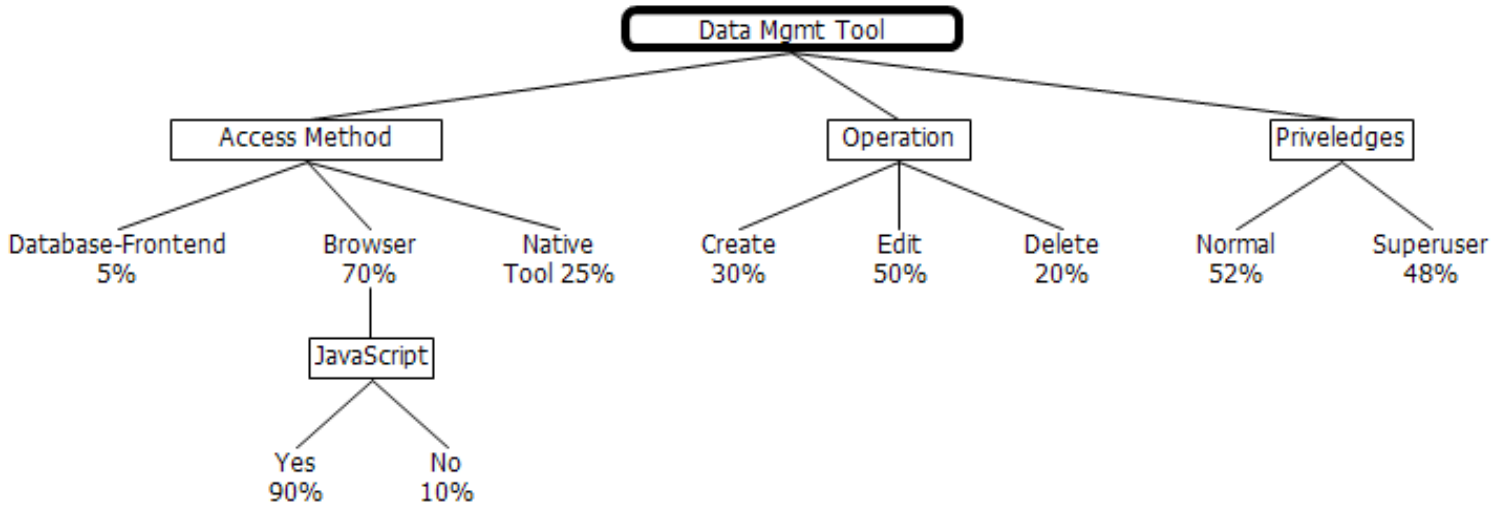
J. Ferrer et al., GECCO 2012

Combinatorial Interaction Testing

The tester identifies the relevant test aspects (*parameters*) and defines corresponding classes (*parameter values*)

A test case is a set of *n* values, one for each parameter

A kind of functional (black-box) testing



Prioritized Combinatorial Interaction Testing

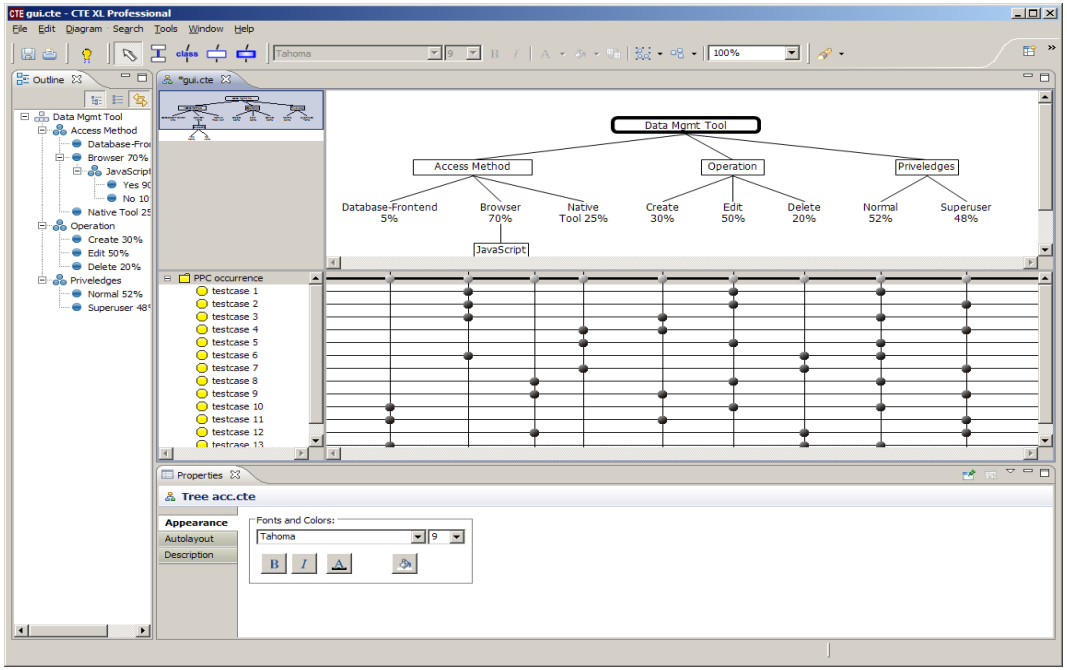
The coverage criterion will determine the degree of parameter interaction

The coverage criterion is defined by its strength *t (t-wise)*

In prioritized CIT, each *t*-tuple has a weight that measures the importance

Tool Support: CTE XL

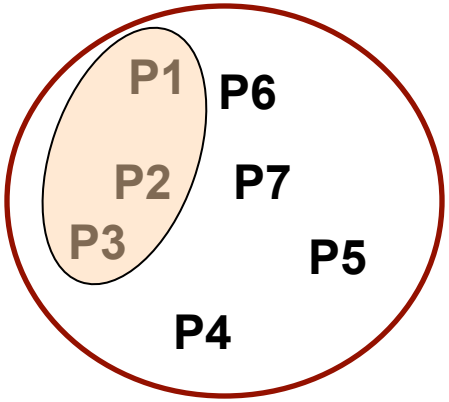
BERNER & MATTNER
AN ASSYSTEM COMPANY



Coverage

Each Used Coverage (EUC)

$$EUC = \frac{number\ of\ covered\ class\ pairs}{number\ of\ coverable\ class\ pairs}$$



EUC = 3 / 7 = 0.43

Weight Coverage (WC)

$$WC = \frac{sum\ of\ weights\ of\ covered\ class\ pairs}{sum\ of\ weights\ of\ all\ coverable\ class\ pairs}$$

Pair	Weight
P1	0.20
P2	0.25
P3	0.15
P4	0.10
P5	0.10
P6	0.05
P7	0.05
$\sum P_i$	0.9

WC = (0.20+0.25+0.15) / 0.9 = 0.66

Coverage: example

#	Access Method	Operation	Priv.	EUC	WC
1	Browser (with JavaScript)	Edit	Normal	0.12	0.30
2	Browser (with JavaScript)	Edit	Superuser	0.19	0.48
3	Browser (with JavaScript)	Create	Normal	0.27	0.60
4	Native Tool	Create	Superuser	0.38	0.71
5	Native Tool	Edit	Normal	0.50	0.80
6	Browser (with JavaScript)	Delete	Normal	0.58	0.88
7	Native Tool	Delete	Superuser	0.62	0.92
8	Browser (no JavaScript)	Edit	Normal	0.69	0.94
9	Browser (no JavaScript)	Create	Superuser	0.77	0.96
10	Database-Frontend	Edit	Normal	0.85	0.98
11	Database-Frontend	Create	Superuser	0.92	0.99
12	Browser (no JavaScript)	Delete	Superuser	0.96	0.99
13	Database-Frontend	Delete	Normal	1.00	1.00

30% weight coverage with one test case

With the weight coverage we cover most important interactions of components in the first test cases

Coverage: example

#	Access Method	Operation	Priv.	EUC	WC
1	Browser (with JavaScript)	Edit	Normal	0.12	0.30
2	Browser (with JavaScript)	Edit	Superuser	0.19	0.48
3	Browser (with JavaScript)	Create	Normal	0.27	0.60
4	Native Tool	Create	Superuser	0.38	0.71
5	Native Tool	Edit	Normal	0.50	0.80
6	Browser (with JavaScript)	Delete	Normal	0.58	0.88
7	Native Tool	Delete	Superuser	0.62	0.92
8	Browser (no JavaScript)	Edit	Normal	0.69	0.94
9	Browser (no JavaScript)	Create	Superuser	0.77	0.96
10	Database-Frontend	Edit	Normal	0.85	0.98
11	Database-Frontend	Create	Superuser	0.92	0.99
12	Browser (no JavaScript)	Delete	Superuser	0.96	0.99
13	Database-Frontend	Delete	Normal	1.00	1.00

60% weight coverage with only three test cases

With the weight coverage we cover most important interactions of components in the first test cases

Coverage: example

#	Access Method	Operation	Priv.	EUC	WC
1	Browser (with JavaScript)	Edit	Normal	0.12	0.30
2	Browser (with JavaScript)	Edit	Superuser	0.19	0.48
3	Browser (with JavaScript)	Create	Normal	0.27	0.60
4	Native Tool	Create	Superuser	0.38	0.71
5	Native Tool	Edit	Normal	0.50	0.80
6	Browser (with JavaScript)	Delete	Normal	0.58	0.88
7	Native Tool	Delete	Superuser	0.62	0.92
8	Browser (no JavaScript)	Edit	Normal	0.69	0.94
9	Browser (no JavaScript)	Create	Superuser	0.77	0.96
10	Database-Frontend	Edit	Normal	0.85	0.98
11	Database-Frontend	Create	Superuser	0.92	0.99
12	Browser (no JavaScript)	Delete	Superuser	0.96	0.99
13	Database-Frontend	Delete	Normal	1.00	1.00

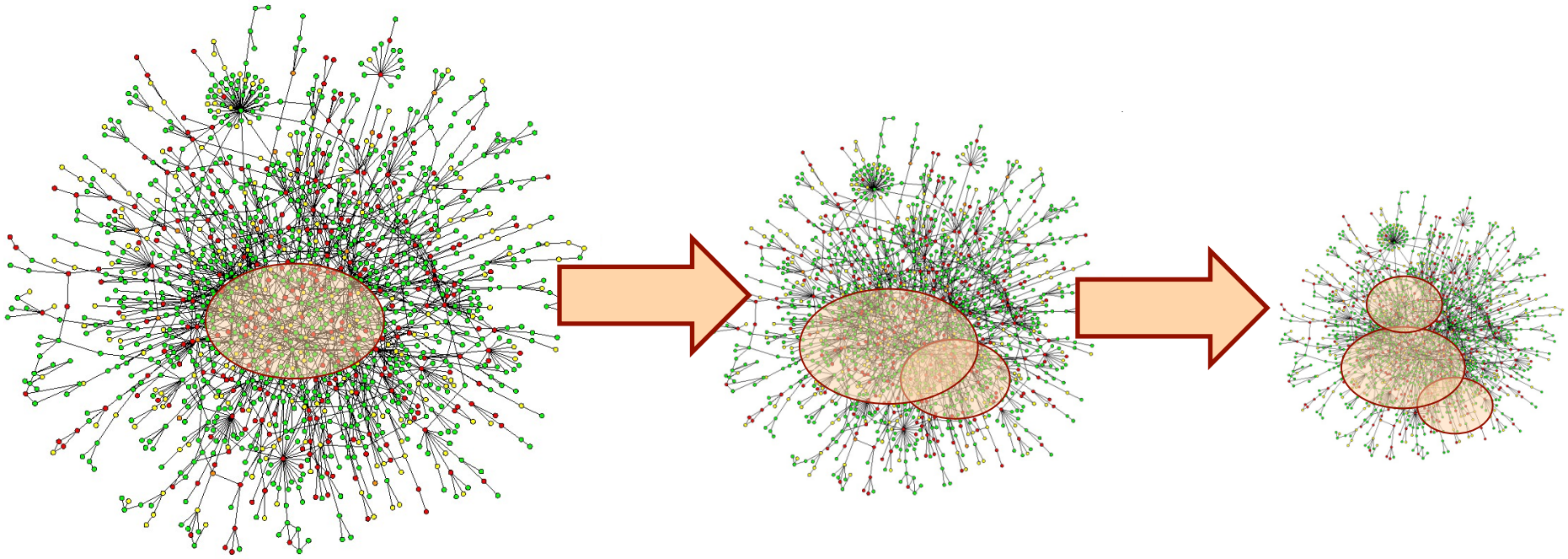
92% weight coverage with just seven test cases

The six less important test cases just suppose 8%

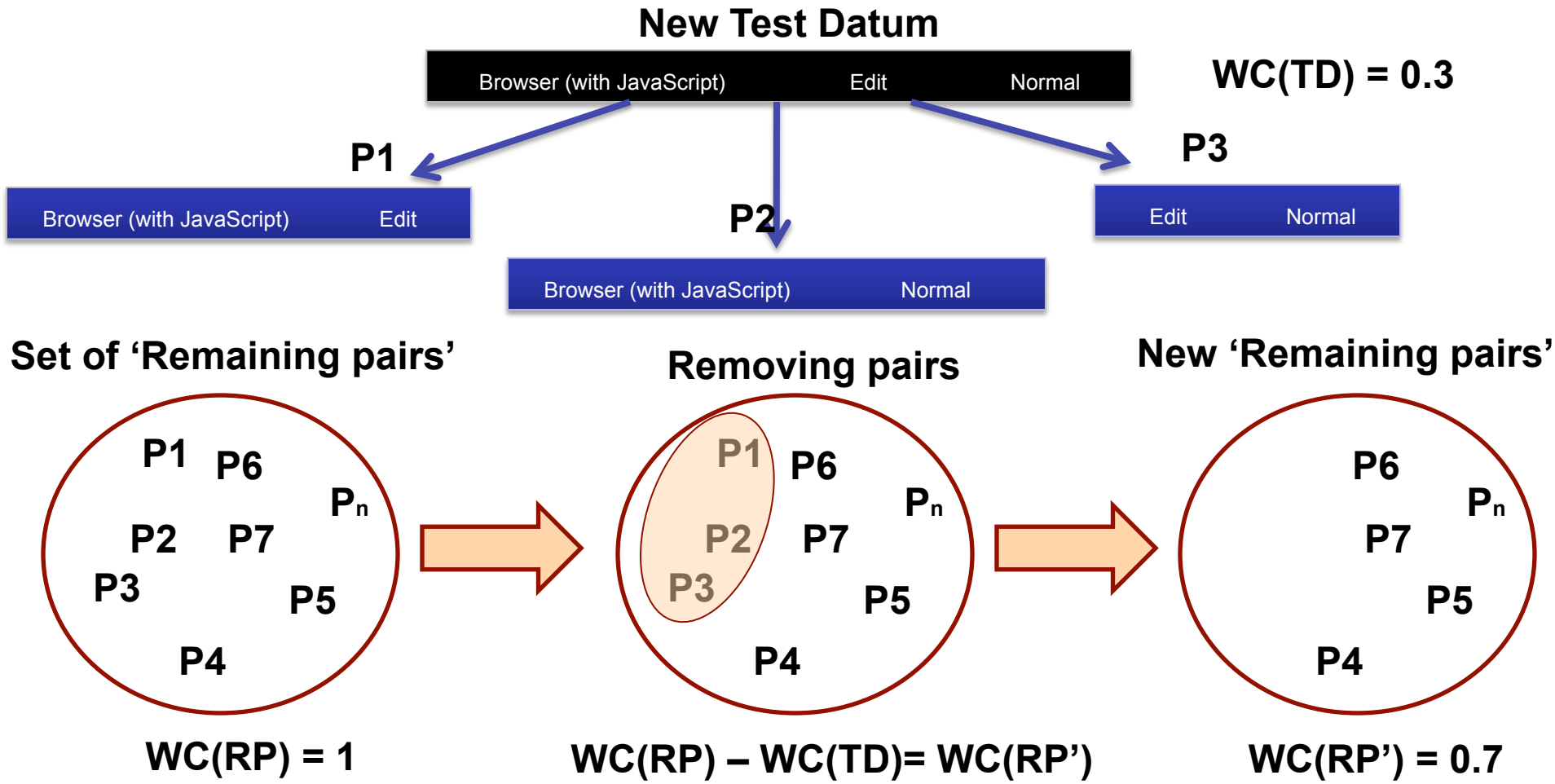
Proposal: Genetic Solver

GS is a constructive algorithm that reduces the problem step by step

It constructs the solution by generating the best test datum at a time



Proposal: Genetic Solver



Results: Experimental Evaluation

Set of benchmarks and distributions proposed by Bryce and Colbourn.

Scenario	# Classes
S1	3^4
S2	10^{20}
S3	3^{100}
S4	$10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1$
S5	$8^2 7^2 6^2 2^4$
S6	$15^1 10^5 5^1 4^1$
S7	$3^{50} 2^{50}$
S8	$20^2 10^2 3^{100}$

Distribution	Description
D1 (equal weights)	All classes have the same weight
D2 (50/50 split)	Half of the weight for each classification are set to 0.9, the other half to 0.1
D3 (1/*vmax*^{2} split)	All weights of classes for a classification are equal to 1/*vmax*^{2}, where *vmax* is the number of classes associated with the classification.
D4 (random)	Weights are randomly distributed

Results: Comparison with PPC and PPS (B&M)

We compare 8 scenarios, 4 distributions, and different coverage values

- **Coverage values: 25%, 50%, 66%, 75%, 90%, 95%, and 99%**

GS is the best in 6 out of 8 scenarios

GS is the best for all distributions

Times one algorithm is better than the others

Scenario	GS	PPC	PPS
S1	0	0	12
S2	8	18	0
S3	9	3	0
S4	14	9	1
S5	13	6	3
S6	24	1	0
S7	5	2	0
S8	19	6	-
Total	**92**	**45**	**19**

Times a significant difference between GS and the others exists

Distribution	PPC		PPS	
D1-GS	28↑	10↓	29↑	8↓
D2-GS	26↑	9↓	42↑	3↓
D3-GS	19↑	10↓	29↑	8↓
D4-GS	22↑	6↓	41↑	4↓
Total	**95↑**	**35↓**	**141↑**	**23↓**

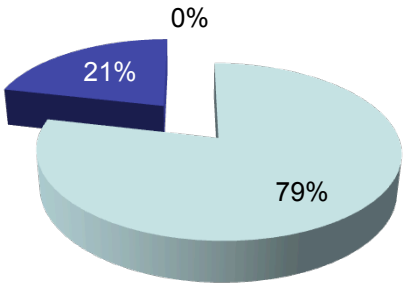
Results: Comparison with PPC and PPS (B&M)

We compared the algorithm focused on different coverage values

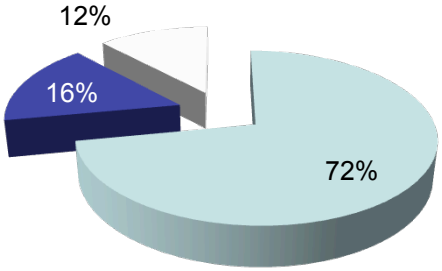
It is important to obtain the best results for intermediate values of coverage

The GS always performs better than the others for these coverage values

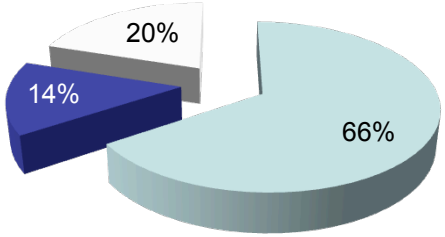
50% Coverage



75% Coverage



100 %Coverage



GS **PPC** **PPS**

Results: Comparison with DDA and BDD

Comparison among GS and the state-of-the-art algorithms:

Deterministic Density Algorithm (DDA): Bryce and Colbourn (2006)

Binary Decision Diagrams (BDD): Salecker et al. (2011)

GS is the best in 7 out of 8 scenarios. It draws on the scenario S1.

GS is the best in 3 out of 4 distributions. It draws in D1 with DDA.

Times one algorithm is better than the others

Scenario	GS	DDA	BDD
S1	2	2	2
S2	11	0	0
S3	6	1	0
S4	8	0	2
S5	7	3	0
S6	11	0	0
S7	3	0	1
S8	3	1	0
Totals	**51**	**7**	**5**

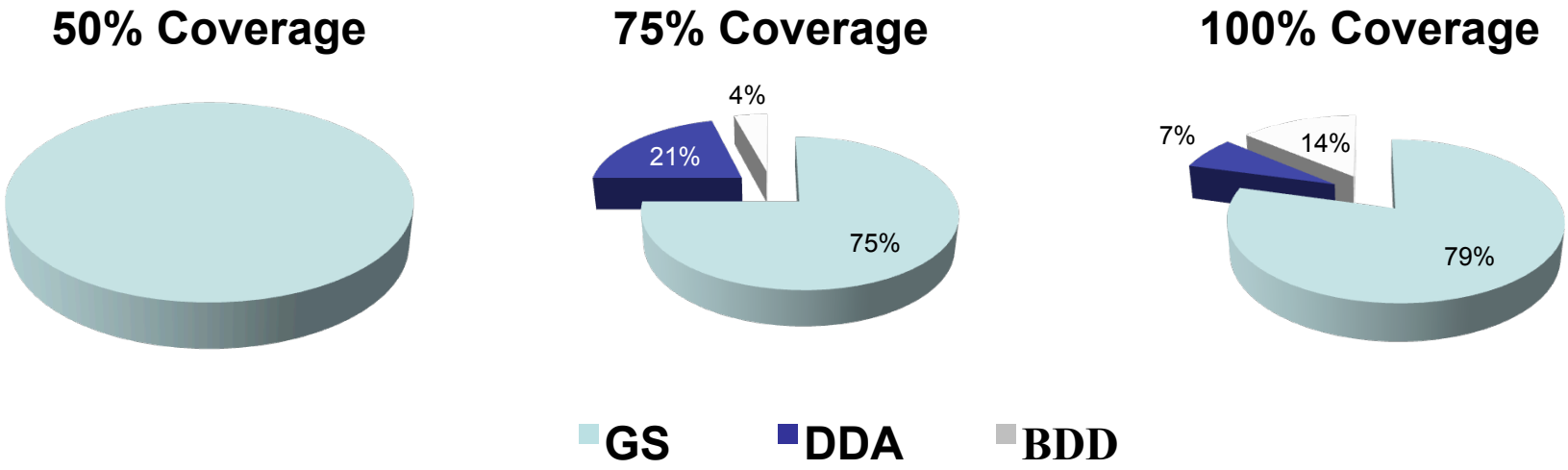
Times there exist significant differences between the algorithms

Distribution	DDA		BDD	
D1-GS	7↑	7↓	15↑	5↓
D2-GS	10↑	1↓	16↑	2↓
D3-GS	16↑	0↓	18↑	1↓
D4-GS	16↑	2↓	22↑	1↓
Totals	**49↑**	**10↓**	**71↑**	**9↓**

Results: Comparison with DDA and BDD

GS always performs better than the state-of-the-art algorithms

It is always better than the other algorithms for all scenarios and distributions for 50% weight coverage.



Testing Complexity

J. Ferrer et al., Inf. & Soft. Tech. 2013

Motivation

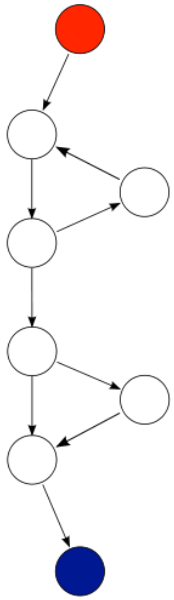
How difficult is to test the Software using automatic test data generation?

Can we estimate the difficulty analyzing the program?

This kind of measure would be useful to estimate the testing costs

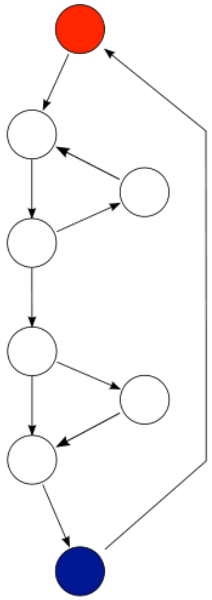
McCabe's Cyclomatic Complexity

v(G)=E-N+2



One entry and exit node

v(G)=E-N+1



Strongly connected graph

What does it mean?

- **Number of linearly independent paths of the graph**
- **Linearly independent paths find errors with high probability**
- **The measure is an estimation of the cost of testing the code**

Other Measures

- Lines of Code (LOC)
- Source Lines of Code ($SLOC$)
- Lines of Code Equivalent ($LOCE$)
- Total Number of Disjunctions ($TNDj$)
- Total Number of Conjunctions ($TNCj$)
- Total Number of Equalities (TNE)
- Total Number of Inequalities (TNI)
- Total Number of Decisions (TND)
- Number of Atomic Conditions per Decision (CpD)
- Nesting Degree (N)
- Halstead's Complexity (HD)
- McCabe's Cyclomatic Complexity (MC)

- Halstead Length (HL): $N = N1 + N2$
- Halstead Vocabulary (HV): $n = n1 + n2$
- Halstead Volume (HVL): $V = N * \log_2 n$
- Halstead Difficulty (HD): $HD = \frac{n_1}{2} * \frac{N_2}{n_2}$
- Halstead Level (HLV): $L = \frac{1}{HD}$
- Halstead Effort (HE): $E = HD * V$
- Halstead Time (HT): $T = \frac{E}{18}$
- Halstead Bugs (HB): $B = \frac{V}{3000}$
- Density of Decisions (DD) $= TND/LOC$.
- Density of LOCE (DLOCE) $= LOCE/LOC$.

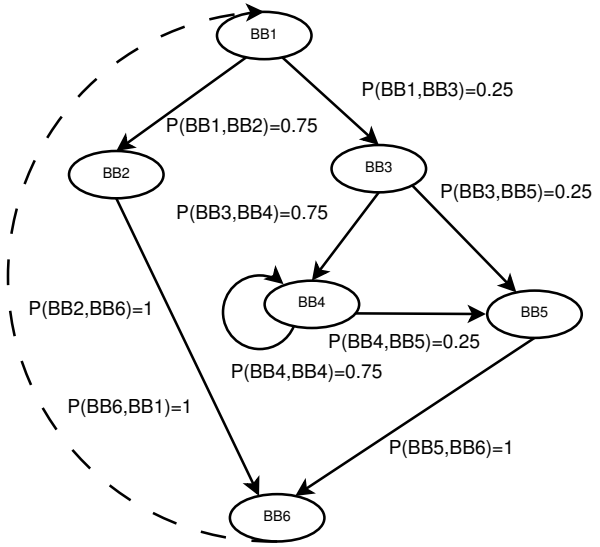
Legend

- $n1$ = the number of distinct operators
- $n2$ = the number of distinct operands
- $N1$ = the total number of operators
- $N2$ = the total number of operands

Our Proposal: Branch Coverage Expectation

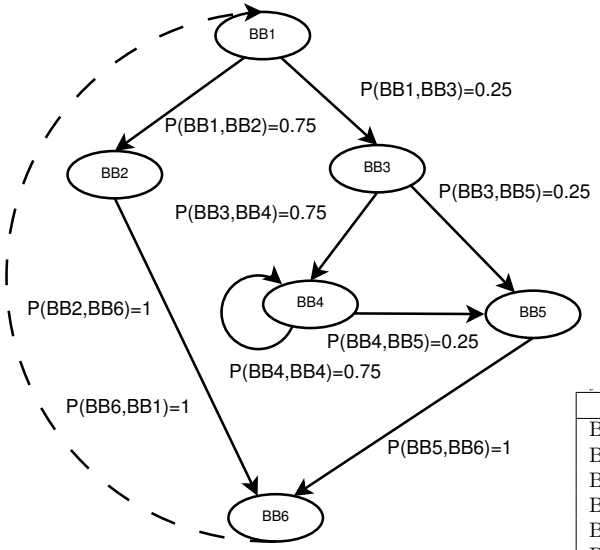
```
/* BB1 */
if (x < 0) || (y < 2)
{
    /* BB2 */
    y=5;
}
else
{
    /* BB3 */
    x=y-3;
    while (y > 5) || (x > 5)
    {
        /* BB4 */
        y=x-5;
    }
    /* BB5 */
    x=x-3;
}
/* BB6 */
```

$$P(c1\&\&c2) = P(c1) * P(c2),$$
$$P(c1||c2) = P(c1) + P(c2) - P(c1) * P(c2),$$
$$P(\neg c1) = 1 - P(c1),$$
$$P(a < b) = \frac{1}{2},$$
$$P(a \leq b) = \frac{1}{2},$$
$$P(a > b) = \frac{1}{2},$$
$$P(a \geq b) = \frac{1}{2},$$
$$P(a == b) = q,$$
$$P(a != b) = 1 - q,$$



Our Proposal: Branch Coverage Expectation

Markov Chain



Compute stationary distribution

$$\pi^T P = \pi^T,$$
$$\pi^T \mathbf{1} = 1.$$

	Stationary Probabilities π_i	Frequency of Appearance $E[BB_i]$
BB1	0.2500	1.00
BB2	0.1875	0.75
BB3	0.0625	0.25
BB4	0.1875	0.75
BB5	0.0625	0.25
BB6	0.2500	1.00

Expected BB executions in 1 run

$$E[BB_i] = \frac{\pi_i}{\pi_1},$$

Expected branch execution in 1 run

$$E[BB_i, BB_j] = E[BB_i] * P_{ij}$$

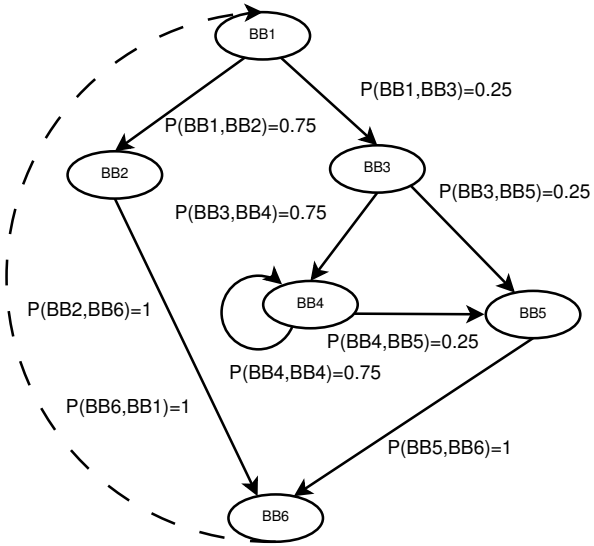
Our Proposal: Branch Coverage Expectation

Most difficult branches to cover

$$A = \{(i,j) | E[BB_i, BB_j] < \frac{1}{2}\}.$$

Branch Coverage Expectation

$$BCE = \frac{1}{|A|} \sum_{(i,j) \in A} E[BB_i, BB_j].$$



$$\text{BCE} = \frac{E[BB_1,BB_3]+E[BB_3,BB_4]+E[BB_3,BB_5]+E[BB_4,BB_5]+E[BB_5,BB_6]}{5} = \frac{\frac{1}{4}+\frac{3}{16}+\frac{1}{16}+\frac{3}{16}+\frac{1}{4}}{5} = \frac{3}{16} = 0.1875.$$

Correlation Study with All the Measures

Table A.10: The correlation coefficients among all the measures analyzed in the benchmark 100%CP

	HD	MC	LOCE	N	DD	DLOCE	BCE	LOC	SLOC	TNDj	TNCj	TNE	TNI	TND	CpD	HL	HV	HVL	HLV	HE	HT	HB	ES	GA	RND
HD	-	0.796	0.786	-0.108	0.052	-0.035	0.285	0.932	0.853	0.742	0.731	0.644	0.639	0.799	0.454	0.870	0.842	0.864	1.0	0.920	0.920	0.864	0.070	-0.101	0.077
MC	0.796	-	0.965	0.266	0.519	0.408	0.025	0.805	0.962	0.925	0.934	0.829	0.811	0.985	0.524	0.976	0.969	0.977	-0.796	0.954	0.954	0.977	-0.150	-0.226	-0.074
LOCE	0.786	0.965	-	0.344	0.515	0.474	-0.038	0.796	0.974	0.884	0.882	0.822	0.789	0.976	0.501	0.945	0.938	0.945	-0.786	0.921	0.921	0.945	-0.186	-0.251	-0.133
N	-0.108	0.266	0.344	-	0.765	0.877	-0.540	-0.207	0.180	0.235	0.240	0.311	0.234	0.276	0.136	0.138	0.127	0.139	0.108	0.089	0.089	0.139	-0.543	-0.381	-0.434
DD	0.052	0.519	0.515	0.765	-	0.912	-0.377	-0.043	0.405	0.449	0.489	0.485	0.437	0.538	0.283	0.368	0.367	0.372	-0.052	0.302	0.302	0.372	-0.439	-0.304	-0.311
DLOCE	-0.035	0.408	0.474	0.877	0.912	-	-0.485	-0.132	0.336	0.352	0.380	0.410	0.353	0.418	0.217	0.270	0.258	0.271	0.035	0.208	0.208	0.271	-0.504	-0.345	-0.397
BCE	0.285	0.025	-0.038	-0.540	-0.377	-0.485	-	0.307	0.081	0.065	0.008	-0.124	0.009	0.017	0.078	0.121	0.129	0.120	-0.285	0.159	0.159	0.120	0.510	0.375	0.534
LOC	0.932	0.805	0.796	-0.207	-0.043	-0.132	0.307	-	0.879	0.753	0.730	0.634	0.646	0.810	0.419	0.891	0.892	0.890	-0.932	0.910	0.910	0.890	0.136	-0.053	0.120
SLOC	0.853	0.962	0.974	0.180	0.405	0.336	0.081	0.879	-	0.884	0.878	0.794	0.778	0.973	0.492	0.975	0.970	0.975	-0.853	0.960	0.960	0.975	-0.091	-0.194	-0.050
TNDj	0.742	0.925	0.884	0.235	0.449	0.352	0.065	0.753	0.884	-	0.773	0.813	0.719	0.897	0.515	0.919	0.908	0.919	-0.742	0.900	0.900	0.919	-0.119	-0.175	-0.036
TNCj	0.731	0.934	0.882	0.240	0.489	0.380	0.008	0.730	0.878	0.773	-	0.734	0.806	0.905	0.497	0.913	0.901	0.913	-0.731	0.895	0.895	0.913	-0.158	-0.235	-0.072
TNE	0.644	0.829	0.822	0.311	0.485	0.410	-0.124	0.634	0.794	0.813	0.734	-	0.618	0.822	0.435	0.798	0.785	0.797	-0.644	0.779	0.779	0.797	-0.272	-0.279	-0.207
TNI	0.639	0.811	0.789	0.234	0.437	0.353	0.009	0.646	0.778	0.719	0.806	0.618	-	0.799	0.439	0.794	0.791	0.795	-0.639	0.774	0.774	0.795	-0.121	-0.201	-0.095
TND	0.799	0.985	0.976	0.276	0.538	0.418	0.017	0.810	0.973	0.897	0.905	0.822	0.799	-	0.503	0.961	0.959	0.962	-0.799	0.935	0.935	0.962	-0.147	-0.226	-0.082
CpD	0.454	0.524	0.501	0.136	0.283	0.217	0.078	0.419	0.492	0.515	0.497	0.435	0.439	0.503	-	0.524	0.518	0.523	-0.454	0.514	0.514	0.523	-0.089	-0.132	0.035
HL	0.870	0.976	0.945	0.138	0.368	0.270	0.121	0.891	0.975	0.919	0.913	0.798	0.794	0.961	0.524	-	0.991	1.0	-0.870	0.989	0.989	1.0	-0.071	-0.180	-0.012
HV	0.842	0.969	0.938	0.127	0.367	0.258	0.129	0.892	0.970	0.908	0.901	0.785	0.791	0.959	0.518	0.991	-	0.994	-0.842	0.971	0.971	0.994	-0.061	-0.172	-0.003
HVL	0.864	0.977	0.945	0.139	0.372	0.271	0.120	0.890	0.975	0.919	0.913	0.797	0.795	0.962	0.523	1.0	0.994	-	-0.864	0.987	0.987	1.0	-0.072	-0.181	-0.011
HLV	-1.0	-0.796	-0.786	0.108	-0.052	0.035	-0.285	-0.932	-0.853	-0.742	-0.731	-0.644	-0.639	-0.799	-0.454	-0.870	-0.842	-0.864	-	-0.920	-0.920	-0.864	-0.070	0.101	-0.077
HE	0.920	0.954	0.921	0.089	0.302	0.208	0.159	0.910	0.960	0.900	0.895	0.779	0.774	0.935	0.514	0.989	0.971	0.987	-0.920	-	1.0	0.987	-0.046	-0.168	0.006
HT	0.920	0.954	0.921	0.089	0.302	0.208	0.159	0.910	0.960	0.900	0.895	0.779	0.774	0.935	0.514	0.989	0.971	0.987	-0.920	1.0	-	0.987	-0.046	-0.168	0.006
HB	0.864	0.977	0.945	0.139	0.372	0.271	0.120	0.890	0.975	0.919	0.913	0.797	0.795	0.962	0.523	1.0	0.994	1.0	-0.864	0.987	0.987	-	-0.072	-0.181	-0.011
ES	0.070	-0.150	-0.186	-0.543	-0.439	-0.504	0.510	0.136	-0.091	-0.119	-0.158	-0.272	-0.121	-0.147	-0.089	-0.071	-0.061	-0.072	-0.070	-0.046	-0.046	-0.072	-	0.365	0.445
GA	-0.101	-0.226	-0.251	-0.381	-0.304	-0.345	0.375	-0.053	-0.194	-0.175	-0.235	-0.279	-0.201	-0.226	-0.132	-0.180	-0.172	-0.181	0.101	-0.168	-0.168	-0.181	0.365	-	0.403
RND	0.077	-0.074	-0.133	-0.434	-0.311	-0.397	0.534	0.120	-0.050	-0.036	-0.072	-0.207	-0.095	-0.082	0.035	-0.012	-0.003	-0.011	-0.077	0.006	0.006	-0.011	0.445	0.403	-

Study over 2600 programs

Table A.11: The correlation coefficients among all the measures analyzed in the benchmark ¬100%CP

	HD	MC	LOCE	N	DD	DLOCE	BCE	LOC	SLOC	TNDj	TNCj	TNE	TNI	TND	CpD	HL	HV	HVL	HLV	HE	HT	HB	ES	GA	RND
HD	-	0.698	0.359	-0.062	0.023	0.014	0.051	0.664	0.648	0.653	0.651	0.557	0.569	0.463	0.441	0.764	0.576	0.747	-1.0	0.872	0.872	0.747	0.069	0.067	0.079
MC	0.698	-	0.571	0.257	0.432	0.351	-0.142	0.472	0.667	0.936	0.937	0.803	0.827	0.718	0.671	0.782	0.762	0.786	-0.698	0.803	0.803	0.786	-0.177	-0.168	-0.173
LOCE	0.359	0.571	-	0.692	0.590	0.833	-0.461	0.414	0.717	0.435	0.432	0.479	0.485	0.814	0.086	0.564	0.503	0.560	-0.359	0.524	0.524	0.560	-0.461	-0.452	-0.476
N	-0.062	0.257	0.692	-	0.708	0.870	-0.575	-0.160	0.190	0.163	0.161	0.229	0.220	0.502	-0.031	0.020	0.009	0.019	0.062	-0.007	-0.007	0.019	-0.563	-0.554	-0.589
DD	0.023	0.432	0.590	0.708	-	0.774	-0.426	-0.178	0.280	0.306	0.304	0.385	0.372	0.723	0.026	0.089	0.056	0.087	-0.023	0.070	0.070	0.087	-0.476	-0.473	-0.497
DLOCE	0.014	0.351	0.833	0.870	0.774	-	-0.556	-0.113	0.284	0.247	0.243	0.308	0.291	0.593	0.013	0.096	0.076	0.095	-0.014	0.073	0.073	0.095	-0.577	-0.564	-0.602
BCE	0.051	-0.142	-0.461	-0.575	-0.426	-0.556	-	0.075	-0.143	-0.078	-0.079	-0.200	-0.138	-0.318	0.080	-0.021	-0.006	-0.020	-0.051	0.001	0.001	-0.020	0.714	0.698	0.732
LOC	0.664	0.472	0.414	-0.160	-0.178	-0.113	0.075	-	0.857	0.398	0.397	0.386	0.406	0.494	0.144	0.906	0.821	0.901	-0.664	0.874	0.874	0.901	0.102	0.099	0.116
SLOC	0.648	0.667	0.717	0.190	0.280	0.284	-0.143	0.857	-	0.533	0.532	0.549	0.572	0.834	0.152	0.916	0.813	0.910	-0.648	0.875	0.875	0.910	-0.137	-0.137	-0.137
TNDj	0.653	0.936	0.435	0.163	0.306	0.247	-0.078	0.398	0.533	-	0.849	0.753	0.781	0.555	0.747	0.702	0.697	0.707	-0.653	0.731	0.731	0.707	-0.110	-0.101	-0.102
TNCj	0.651	0.937	0.432	0.161	0.304	0.243	-0.079	0.397	0.532	0.849	-	0.753	0.771	0.551	0.746	0.702	0.697	0.707	-0.651	0.731	0.731	0.707	-0.116	-0.107	-0.111
TNE	0.557	0.803	0.479	0.229	0.385	0.308	-0.200	0.386	0.549	0.753	0.753	-	0.623	0.600	0.544	0.633	0.619	0.636	-0.557	0.646	0.646	0.636	-0.278	-0.270	-0.270
TNI	0.569	0.827	0.485	0.220	0.372	0.291	-0.138	0.406	0.572	0.781	0.771	0.623	-	0.619	0.559	0.658	0.645	0.662	-0.569	0.671	0.671	0.662	-0.207	-0.198	-0.204
TND	0.463	0.718	0.814	0.502	0.723	0.593	-0.318	0.494	0.834	0.555	0.551	0.600	0.619	-	0.132	0.688	0.605	0.683	-0.463	0.648	0.648	0.683	-0.338	-0.336	-0.348
CpD	0.441	0.671	0.086	-0.031	0.026	0.013	0.080	0.144	0.152	0.747	0.746	0.544	0.559	0.132	-	0.394	0.436	0.402	-0.441	0.437	0.437	0.402	0.026	0.026	0.031
HL	0.764	0.782	0.564	0.020	0.089	0.096	-0.021	0.906	0.916	0.702	0.702	0.633	0.658	0.688	0.394	-	0.932	0.999	-0.764	0.980	0.980	0.999	-0.021	-0.018	-0.010
HV	0.576	0.762	0.503	0.009	0.056	0.076	-0.006	0.821	0.813	0.697	0.697	0.619	0.645	0.605	0.436	0.932	-	0.946	-0.576	0.874	0.874	0.946	-0.040	-0.030	-0.022
HVL	0.747	0.786	0.560	0.019	0.087	0.095	-0.020	0.901	0.910	0.707	0.707	0.636	0.662	0.683	0.402	0.999	0.946	-	-0.747	0.974	0.974	1.0	-0.023	-0.020	-0.011
HLV	-1.0	-0.698	-0.359	0.062	-0.023	-0.014	-0.051	-0.664	-0.648	-0.653	-0.651	-0.557	-0.569	-0.463	-0.441	-0.764	-0.576	-0.747	-	-0.872	-0.872	-0.747	-0.069	-0.067	-0.079
HE	0.872	0.803	0.524	-0.007	0.070	0.073	0.001	0.874	0.875	0.731	0.731	0.646	0.671	0.648	0.437	0.980	0.874	0.974	-0.872	-	1.0	0.974	0.004	0.005	0.016
HT	0.872	0.803	0.524	-0.007	0.070	0.073	0.001	0.874	0.875	0.731	0.731	0.646	0.671	0.648	0.437	0.980	0.874	0.974	-0.872	1.0	-	0.974	0.004	0.005	0.016
HB	0.747	0.786	0.560	0.019	0.087	0.095	-0.020	0.901	0.910	0.707	0.707	0.636	0.662	0.683	0.402	0.999	0.946	1.0	-0.747	0.974	0.974	1.0	-0.023	-0.020	-0.011
ES	0.069	-0.177	-0.461	-0.563	-0.476	-0.577	0.714	0.102	-0.137	-0.110	-0.116	-0.278	-0.207	-0.338	0.026	-0.021	-0.040	-0.023	-0.069	0.004	0.004	-0.023	-	0.954	0.940
GA	0.067	-0.168	-0.452	-0.554	-0.473	-0.564	0.698	0.099	-0.137	-0.101	-0.107	-0.270	-0.198	-0.336	0.026	-0.018	-0.030	-0.020	-0.067	0.005	0.005	-0.020	0.954	-	0.950
RND	0.079	-0.173	-0.476	-0.589	-0.497	-0.602	0.732	0.116	-0.137	-0.102	-0.111	-0.270	-0.204	-0.348	0.031	-0.010	-0.022	-0.011	-0.079	0.016	0.016	-0.011	0.940	0.950	-

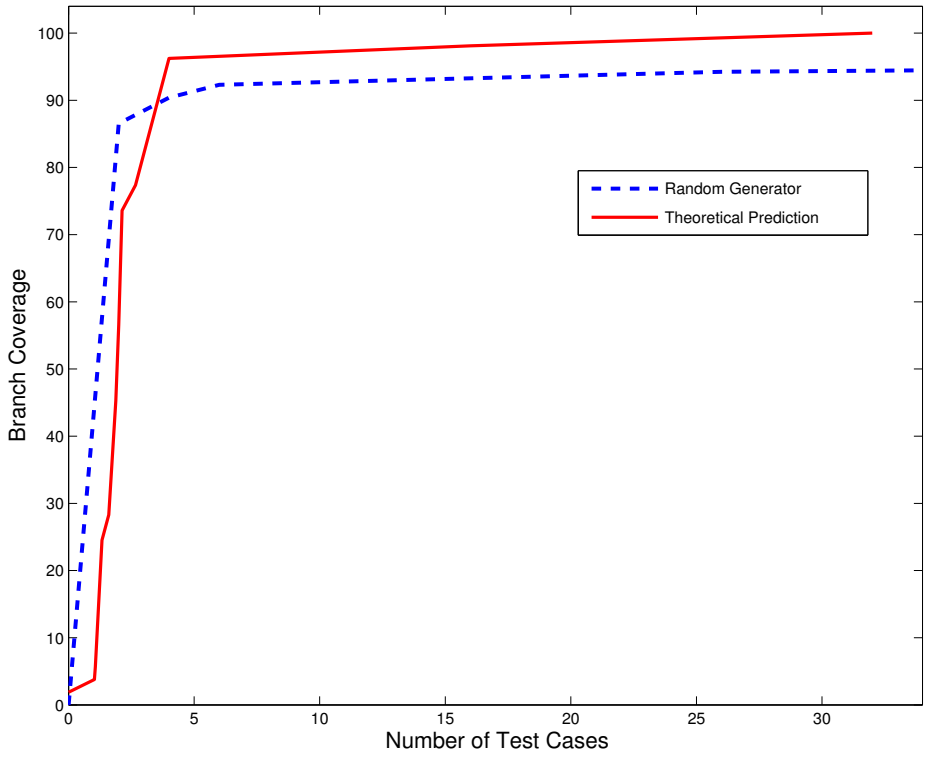
Correlation with Cov. of an Automatic TD Gen.

Study over 2600 programs

	$\widetilde{100\%CP}$			$\neg 100\%CP$		
	ES	GA	RND	ES	GA	RND
MC	-0.150	-0.226	-0.074	-0.177	-0.168	-0.173
HD	0.070	-0.101	0.077	0.069	0.067	0.079
LOCE	-0.186	-0.251	-0.133	-0.461	-0.452	-0.476
N	-0.543	-0.381	-0.434	-0.563	-0.554	-0.589
DD	-0.439	-0.304	-0.311	-0.476	-0.473	-0.497
DLOCE	-0.504	-0.345	-0.397	-0.577	-0.564	-0.602
BCE	0.510	0.375	0.534	0.714	0.698	0.732

Approximated Behaviour of RND

$$f(x) = \left| \left\{ (i,j) \left| \frac{1}{E[BB_i, BB_j]} < x \right. \right\} \right| .$$



Approximated number of TCs to cover the branch

Recent Research on Search Based software Testing

Thanks for your attention !!!