Recent Research on Search Based Software Testing: Part 1

Francisco Chicano

University of Málaga, Spain (assistant professor)

Colorado State University, USA (faculty affiliate)

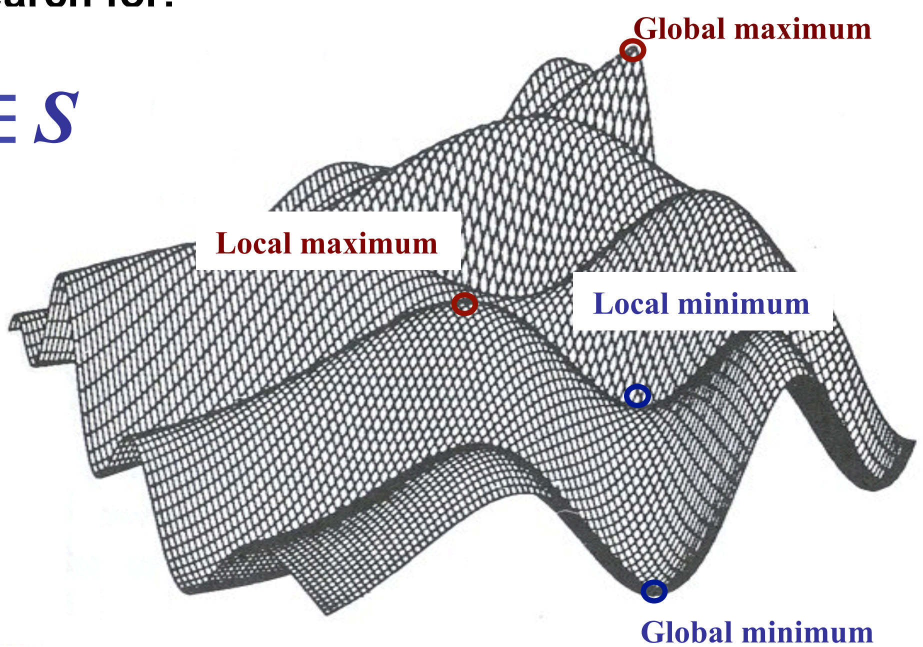
Optimization Problem

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**

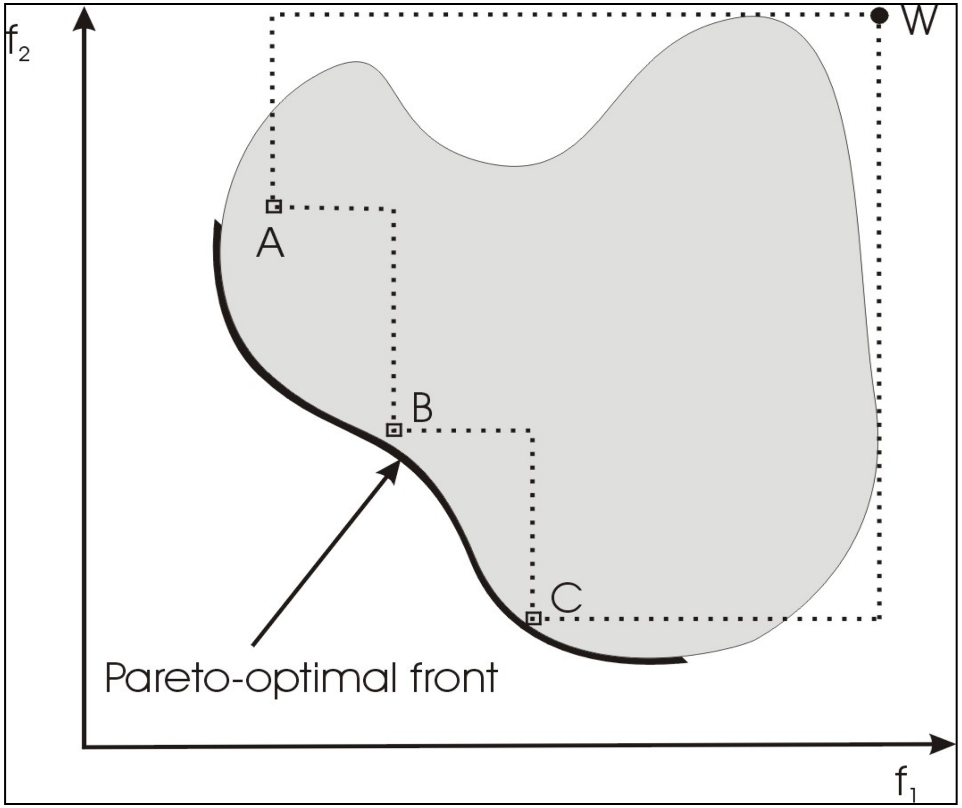
If our goal is to minimize the function we search for:

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

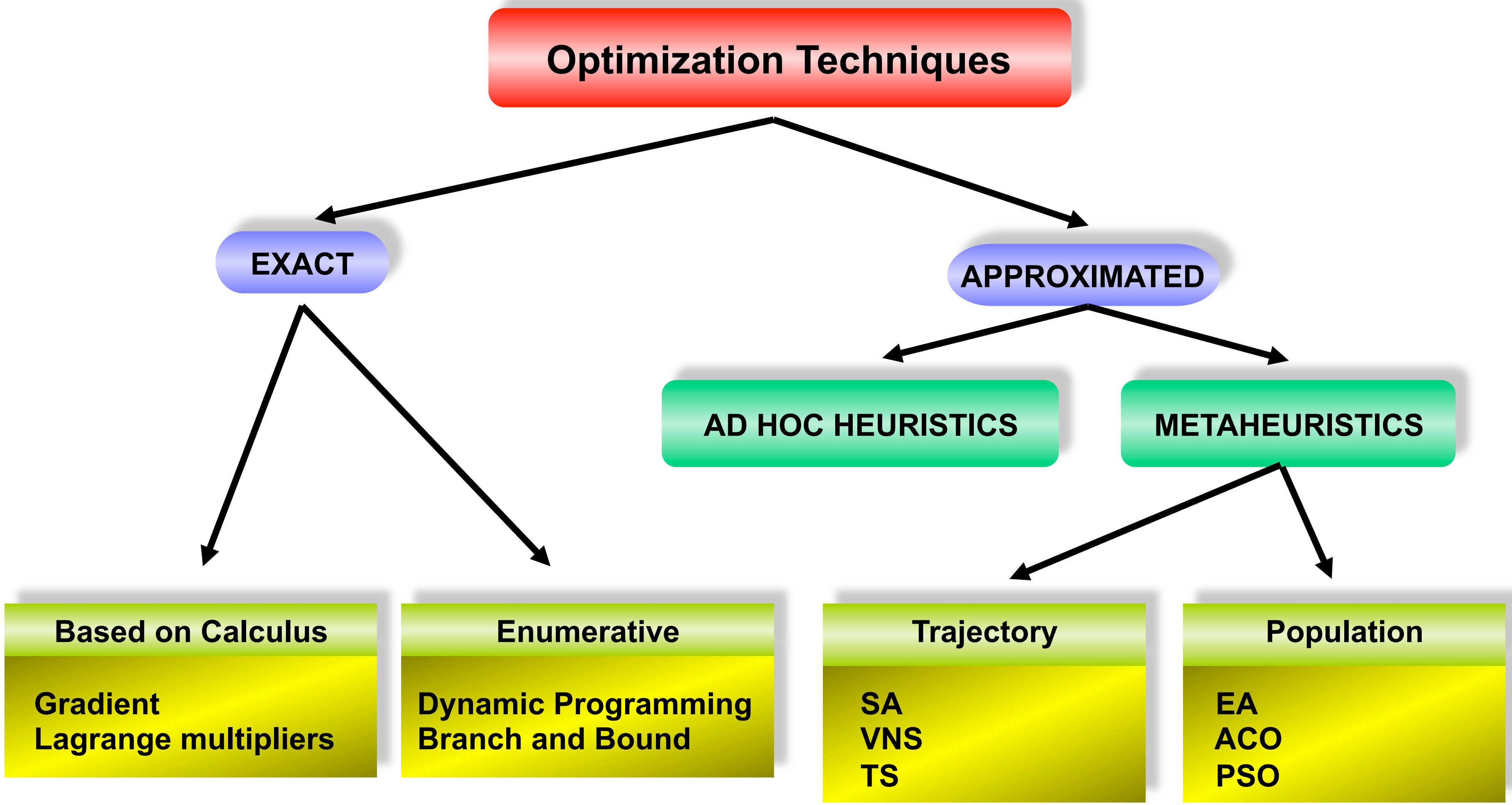


Multi-Objective Optimization Problem

In a MO problem there are several objectives (functions) we want to optimize



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

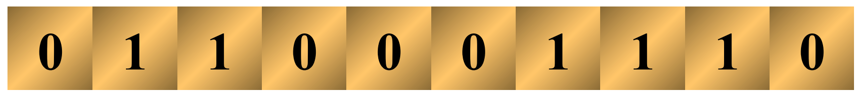
- Variation operators → Make the population to evolve
  - Recombination: exchange of features
  - Mutation: generation of new features

Evolutionary Algorithm

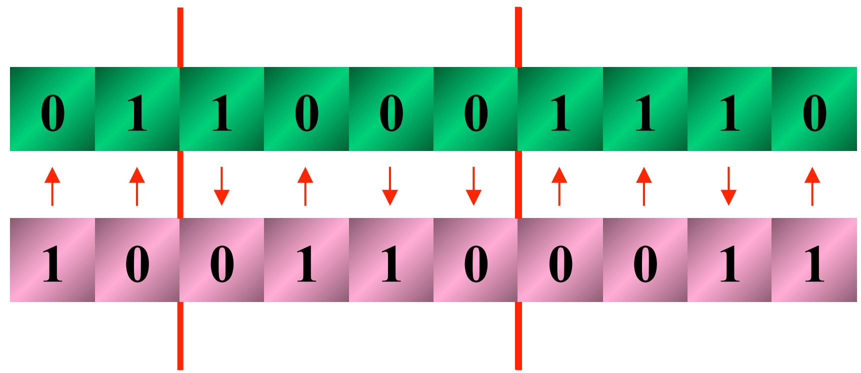
Genetic Algorithms

- **Individuals**

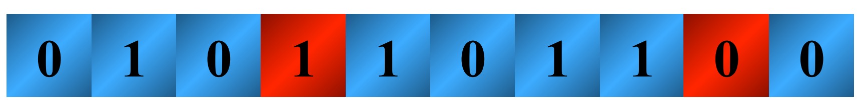
Binary Chromosome



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

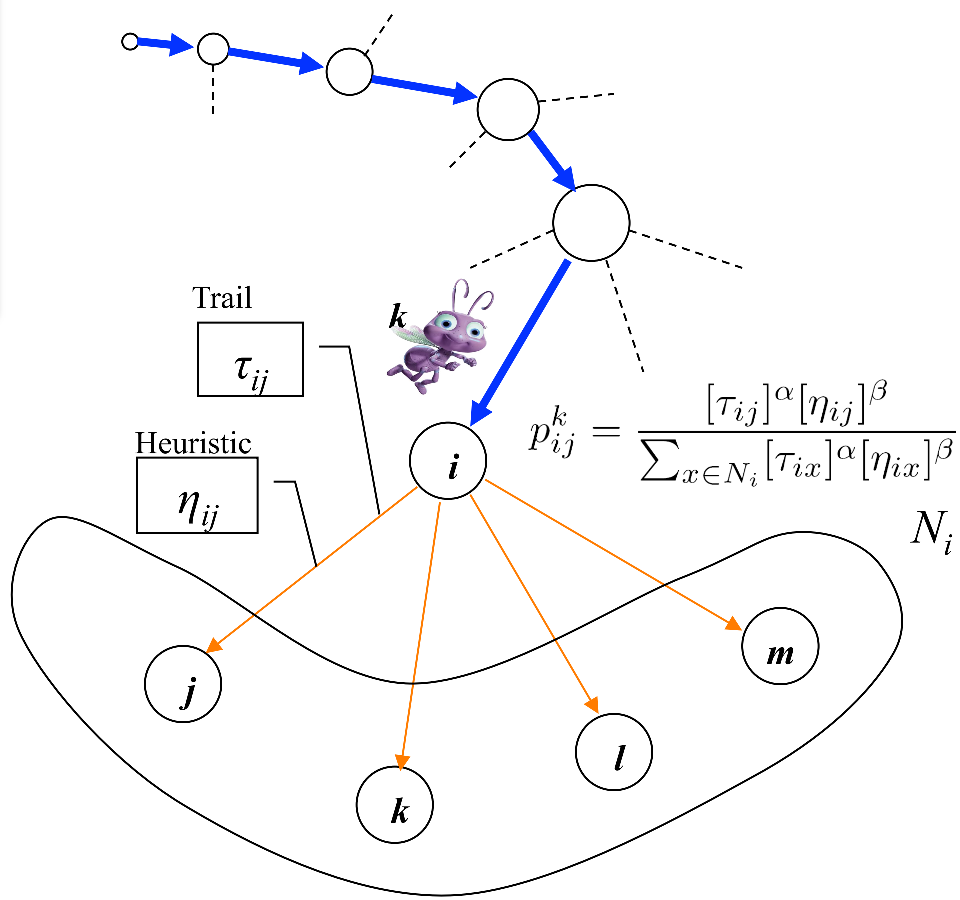


- **Mutation → bit flips**



Ant Colony Optimization

```
procedure ACOMetaheuristic
    ScheduleActivities
        ConstructAntsSolutions
        UpdatePheromones
        DaemonActions // optional
    end ScheduleActivities
end procedure
```



- **The ant selects stochastically its next node**
- **The probability of selecting one node depends on the pheromone trail and the heuristic value (optional) of the edge/node**
- **The ant stops when a complete solution is built**

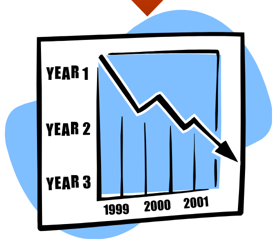
Software Testing: Definition and Goal

- 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).

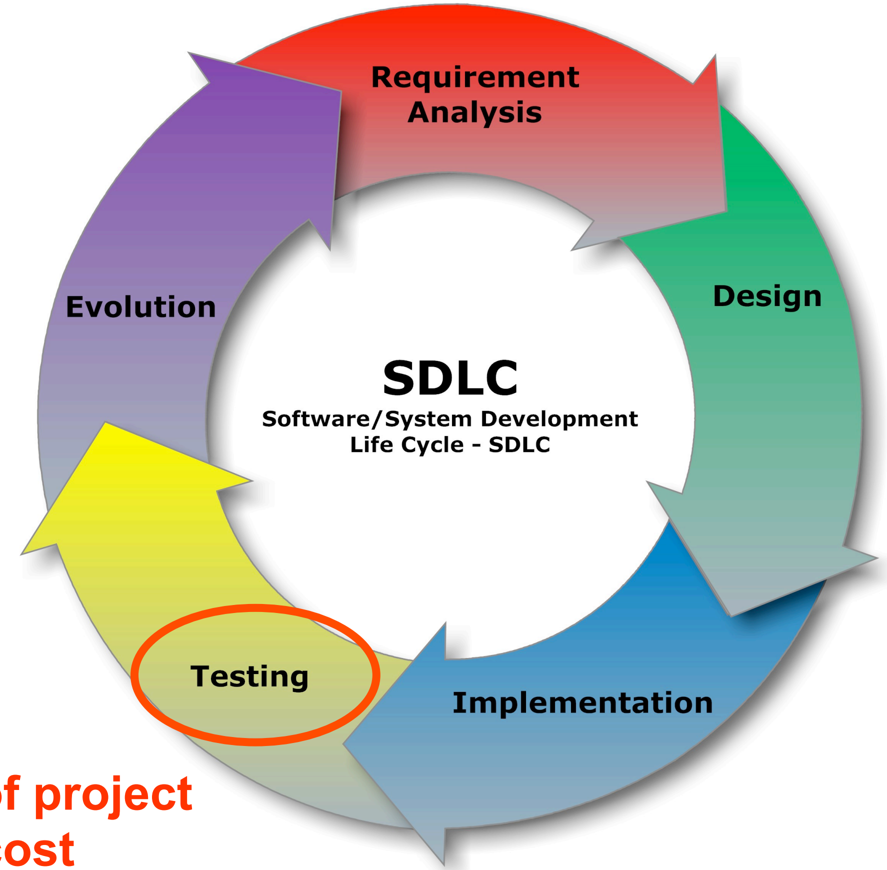
Software Testing: Impact

Software testing is important because…

Software errors



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



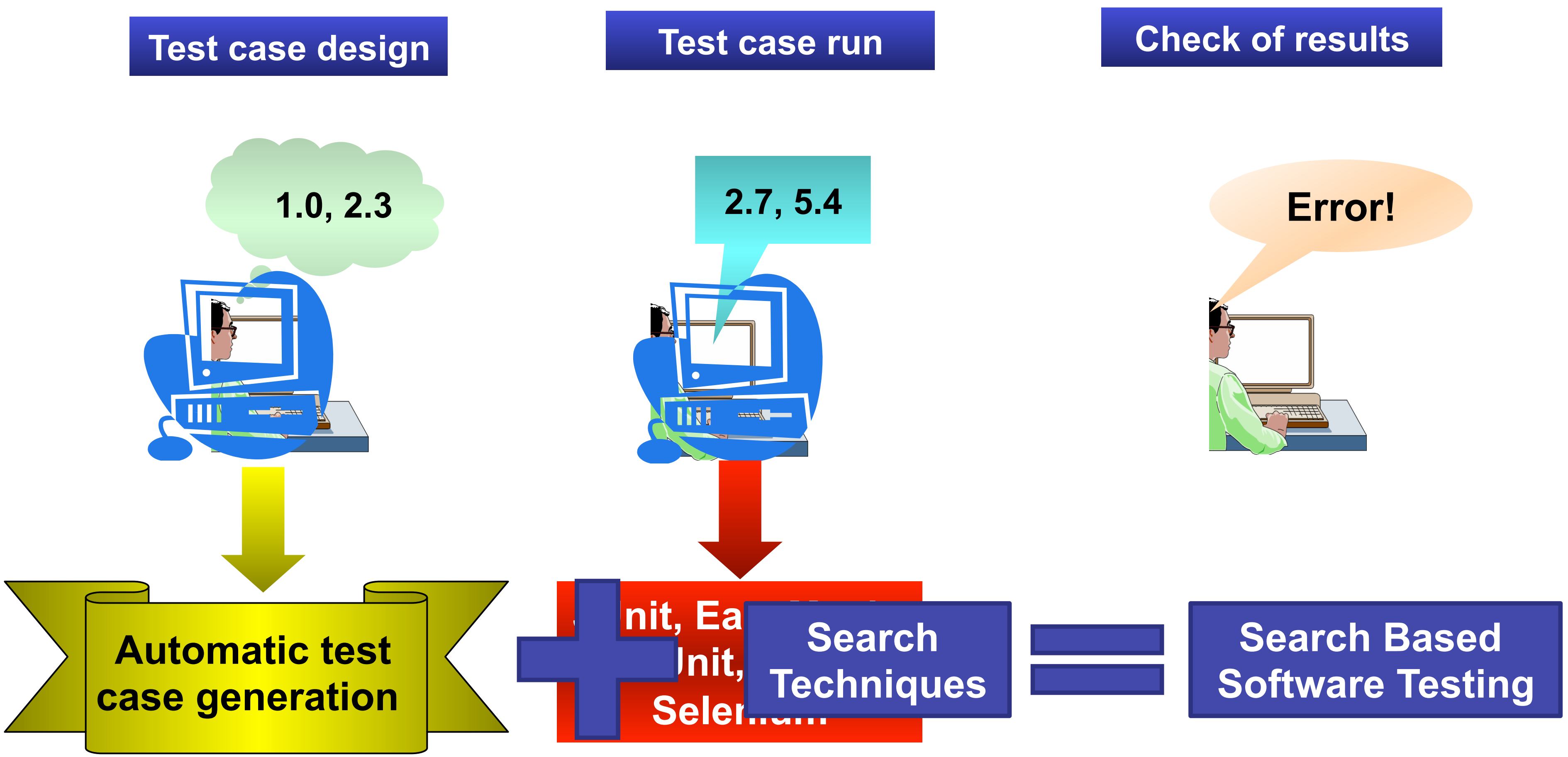
60 % of project cost

Software Testing: Classification

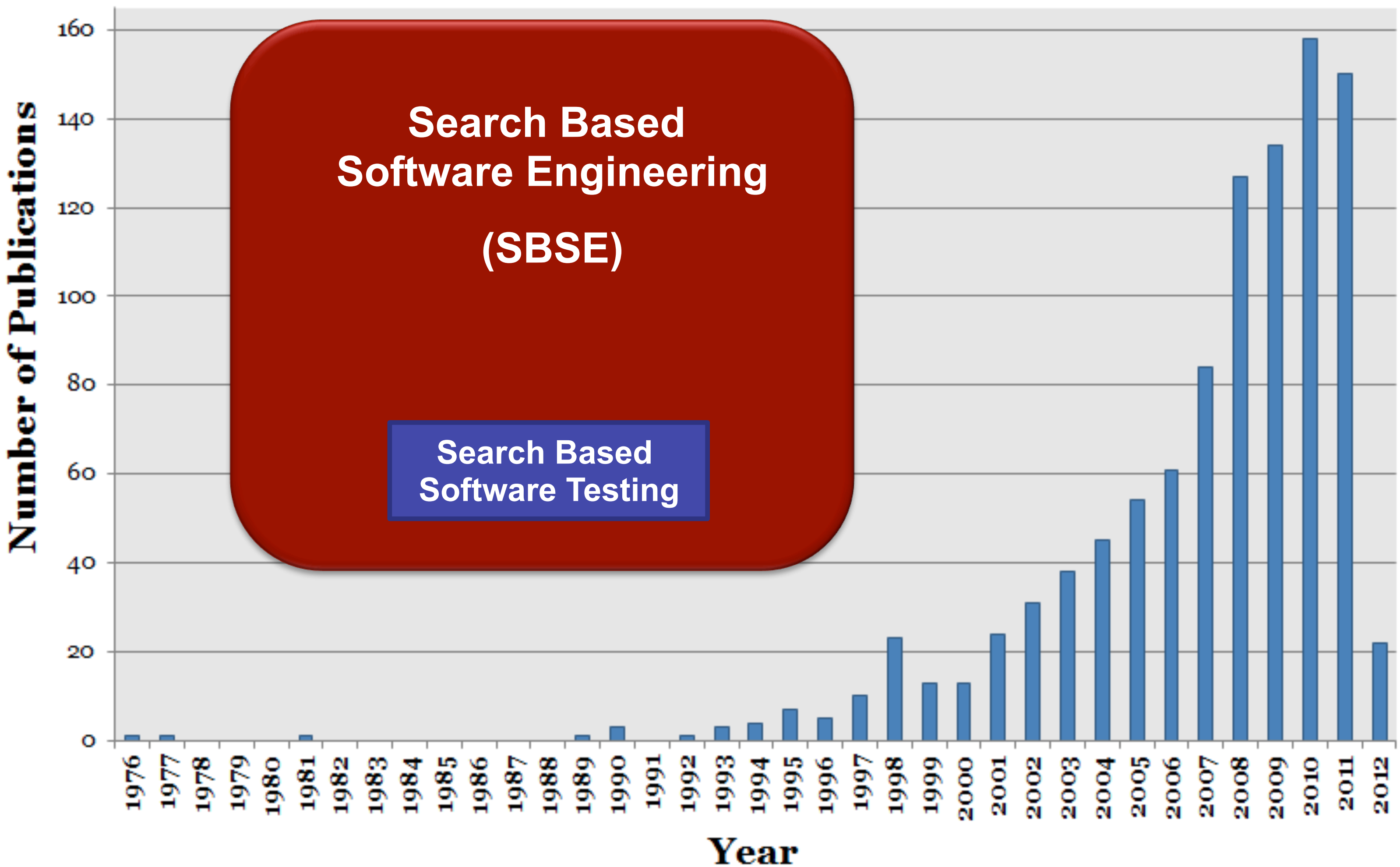
Classification of testing techniques (by goal)

- 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



Search Based Software Engineering



Our Research on SBSE

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

Testing Complexity

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

Motivation

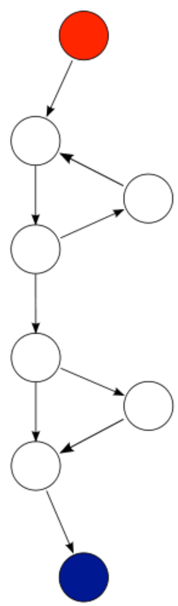
How difficult is to test the Software using automatic test data generation?

Can we estimate the difficulty analyzing the program?

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

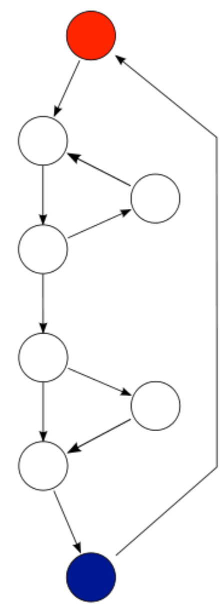
McCabe's Cyclomatic Complexity

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) $= LOCE/LOC$.

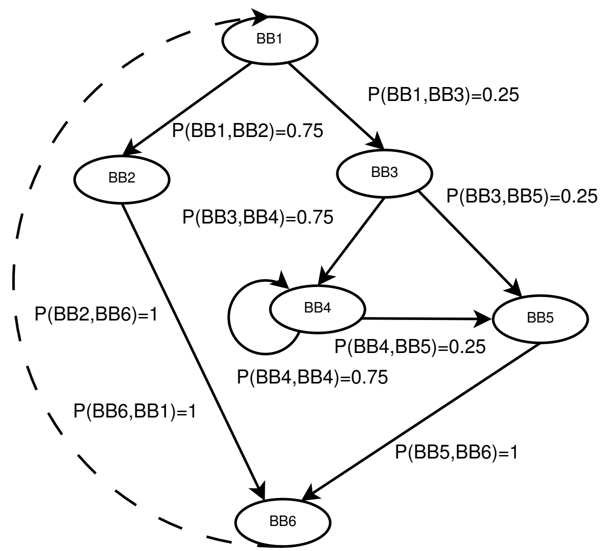
Legend

- *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

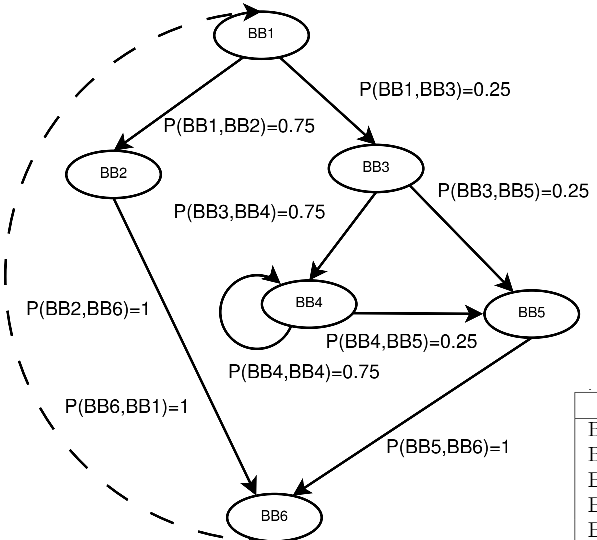
```
/* BB1 */
if (x < 0) || (y < 2)
{
    /* BB2 */
    y=5;
}
else
{
    /* BB3 */
    x=y-3;
    while (y > 5) || (x > 5)
    {
        /* BB4 */
        y=x-5;
    }
    /* BB5 */
    x=x-3;
}
/* BB6 */
```

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



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

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

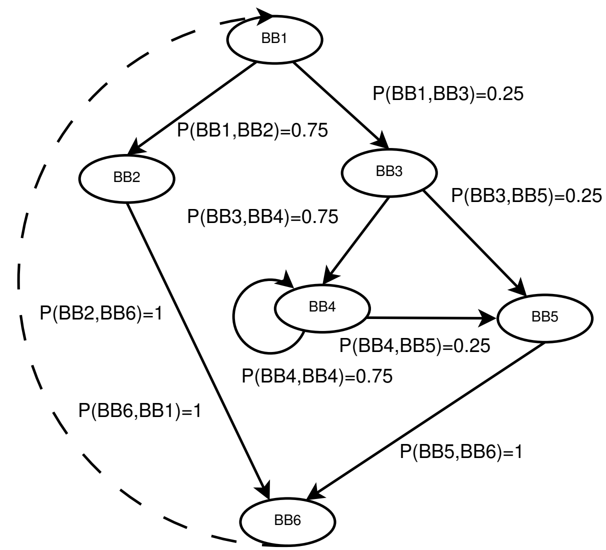
Our Proposal: Branch Coverage Expectation

Most difficult branches to cover

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

Branch Coverage Expectation

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



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

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.

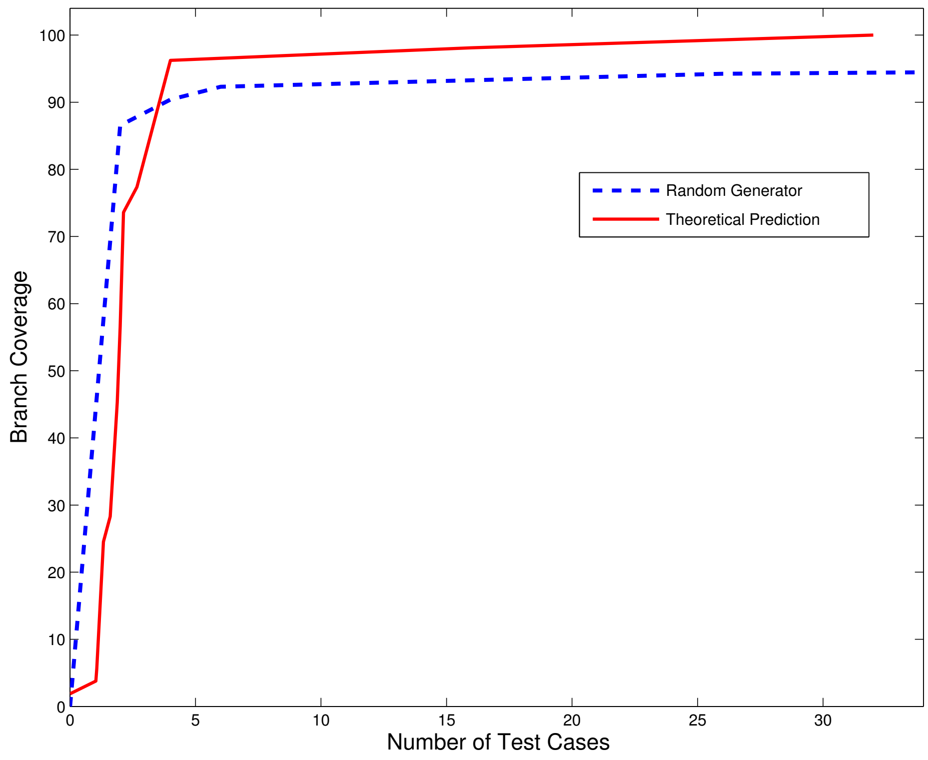
	100%CP			¬100%CP		
	ES	GA	RND	ES	GA	RND
MC	-0.150	-0.226	-0.074	-0.177	-0.168	-0.173
HD	0.070	-0.101	0.077	0.069	0.067	0.079
LOCE	-0.186	-0.251	-0.133	-0.461	-0.452	-0.476
N	-0.543	-0.381	-0.434	-0.563	-0.554	-0.589
DD	-0.439	-0.304	-0.311	-0.476	-0.473	-0.497
DLOCE	-0.504	-0.345	-0.397	-0.577	-0.564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

Study over 2600 programs

On real programs the correlation is higher

Approximated Behaviour of RND

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



Approximated number of TCs to cover the branch

Prioritized Pairwise Combinatorial Interaction Testing

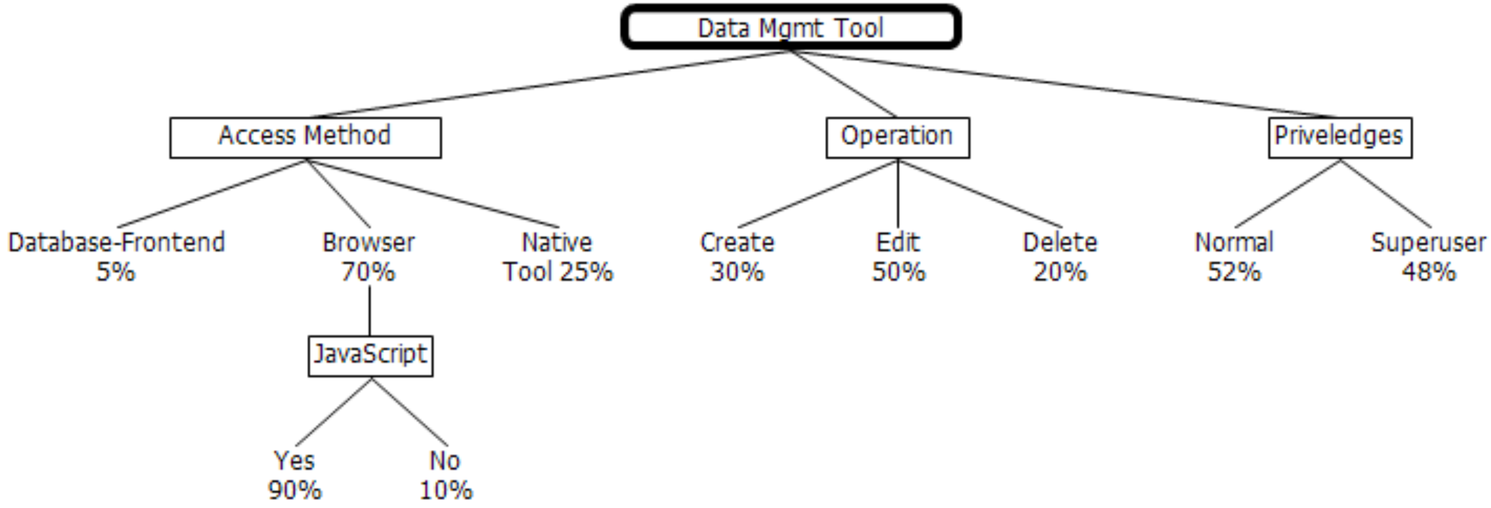
J. Ferrer et al., GECCO 2012

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



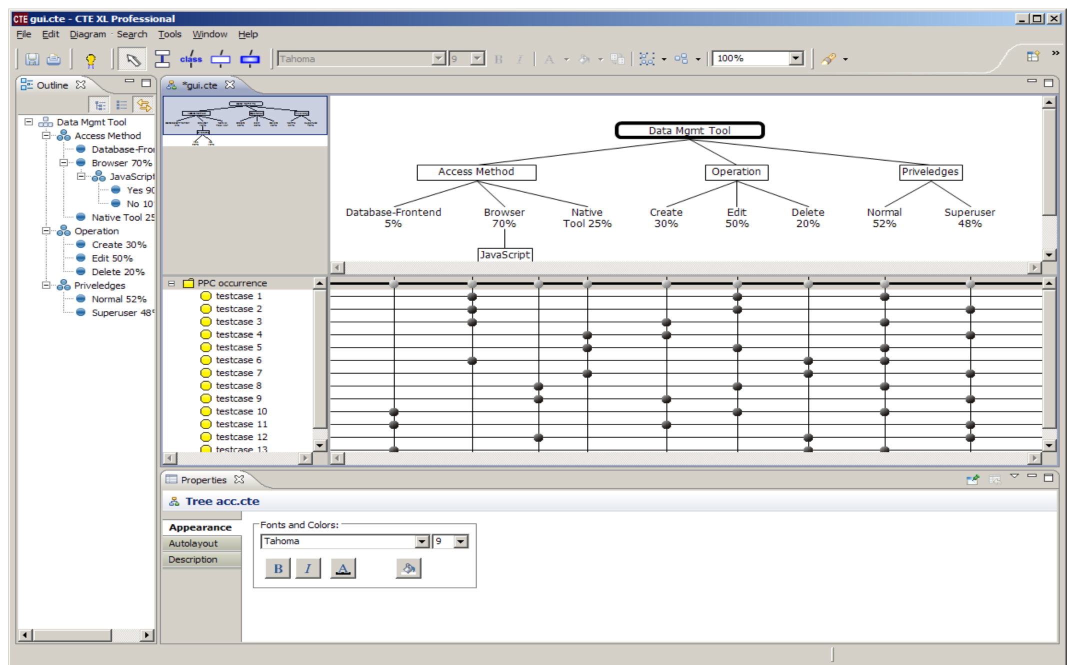
Prioritized Combinatorial Interaction Testing

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

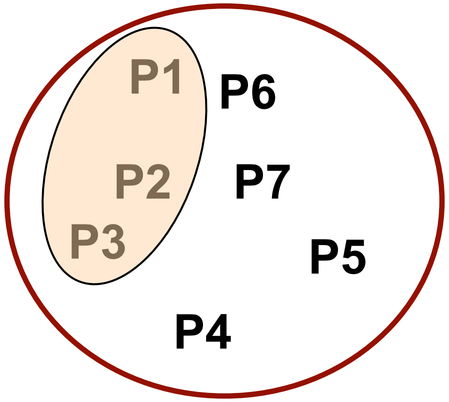


Coverage

Each Used Coverage (EUC)

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

***EUC* = 3 / 7 = 0.43**



Weight Coverage (WC)

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

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

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

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

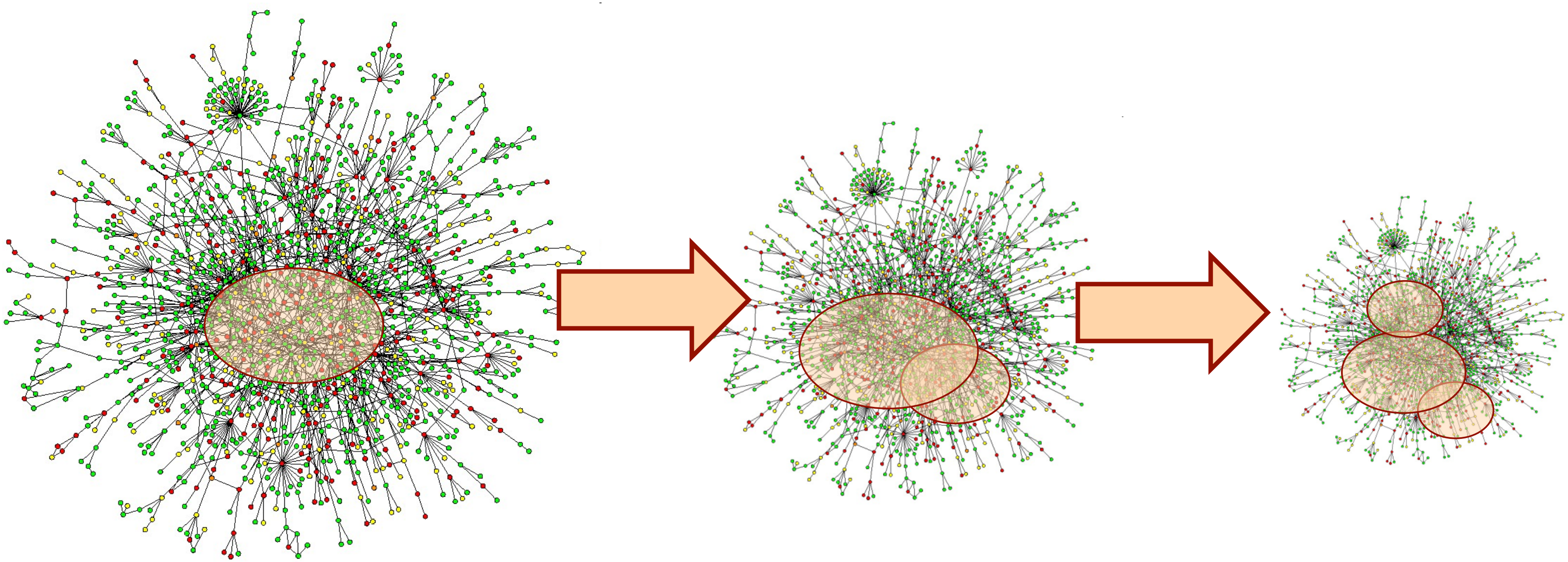
92% weight coverage with just seven test cases

The six less important test cases just suppose 8%

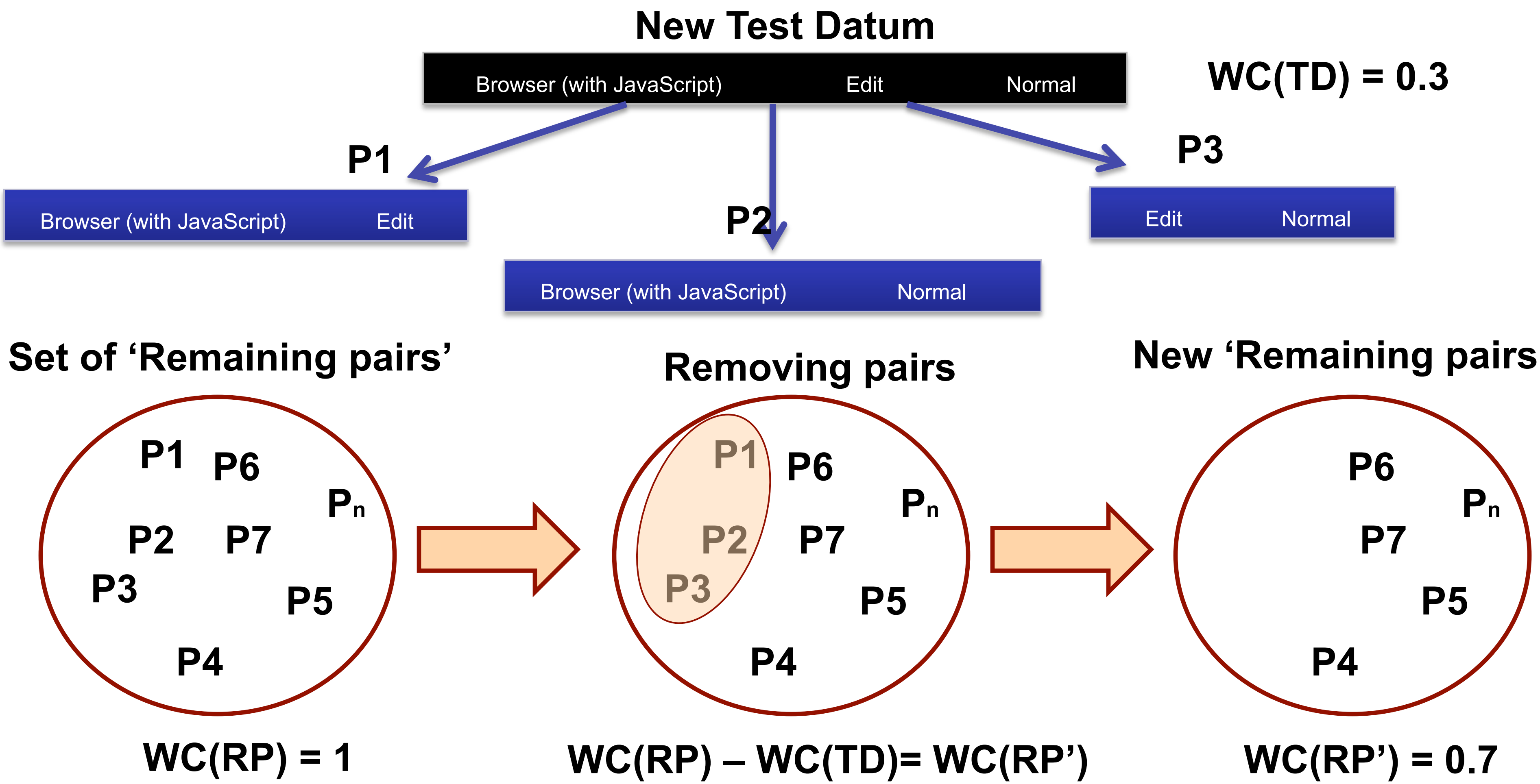
Proposal: Genetic Solver

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

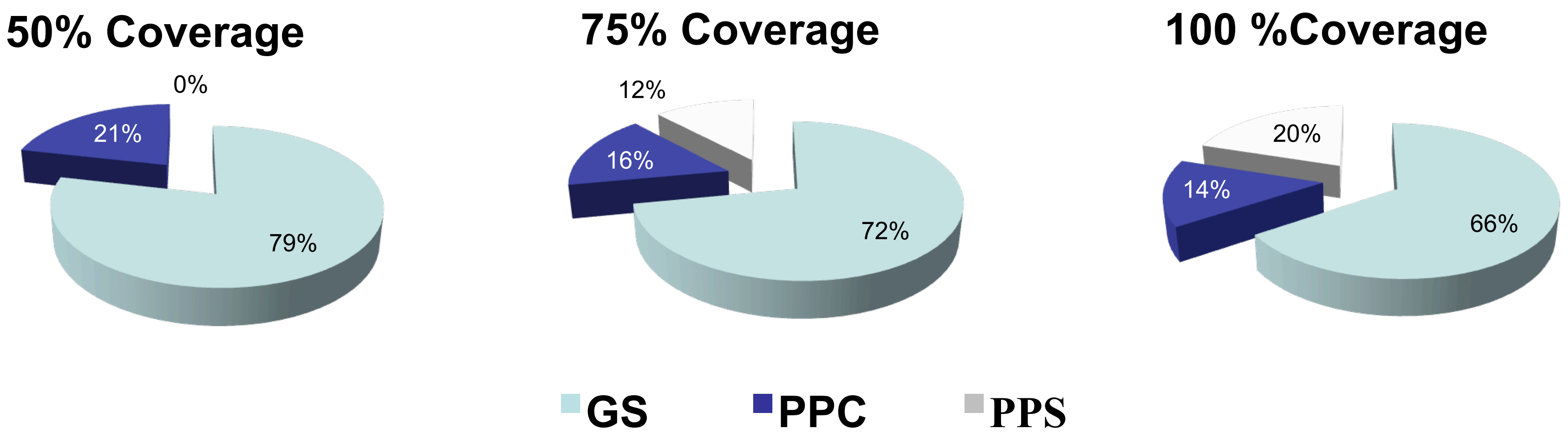
Distribution	PPC		PPS	
D1-GS	28↑	10↓	29↑	8↓
D2-GS	26↑	9↓	42↑	3↓
D3-GS	19↑	10↓	29↑	8↓
D4-GS	22↑	6↓	41↑	4↓
Total	**95↑**	**35↓**	**141↑**	**23↓**

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



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**

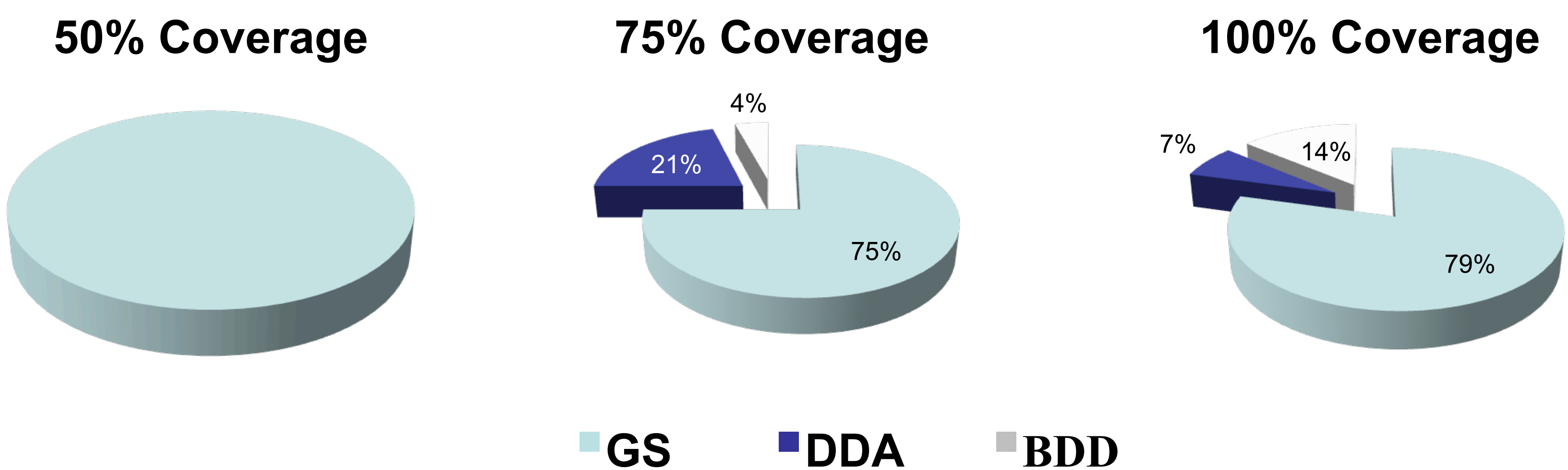
Times there exist significant differences between the algorithms

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.

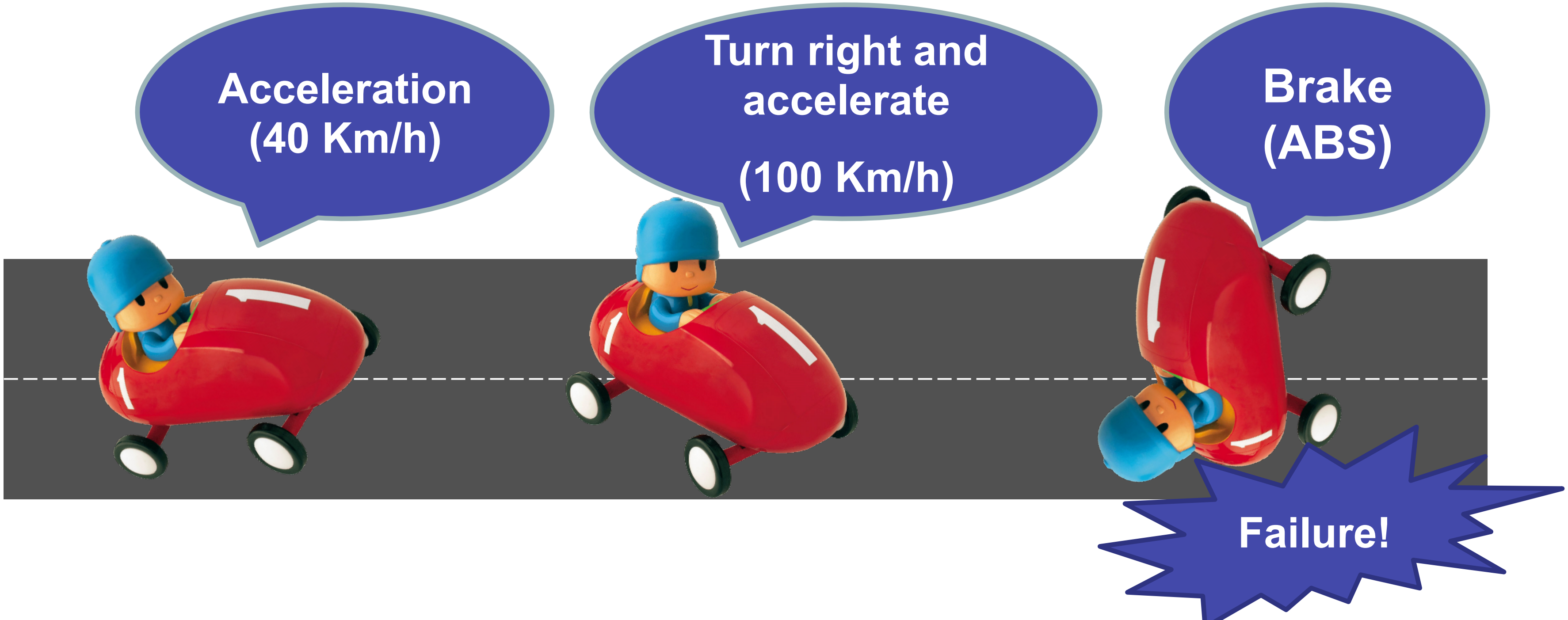


Test Sequence Generation in Functional Testing

J. Ferrer et al., IST 2015

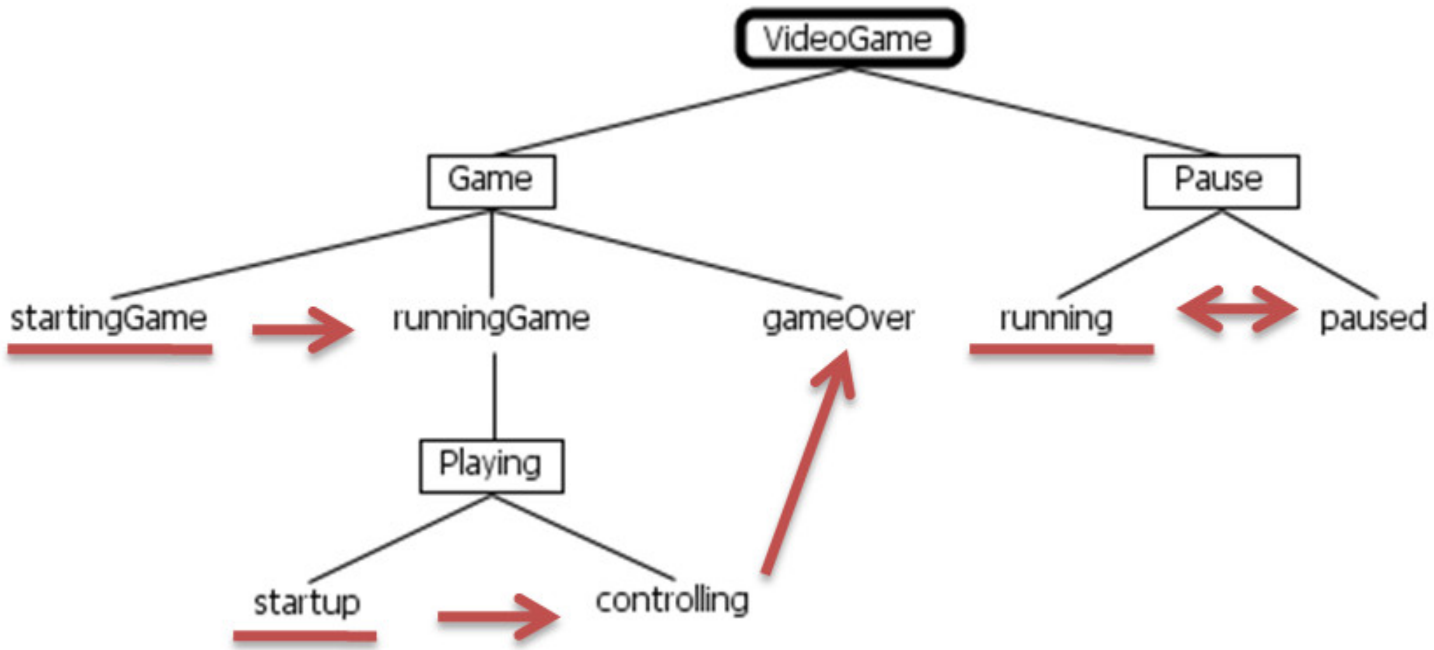
Test Sequences

- Standard tests in combinatorial interaction testing: **independent test cases**
- Test sequences: SUT (*Software Under Test*) state is important



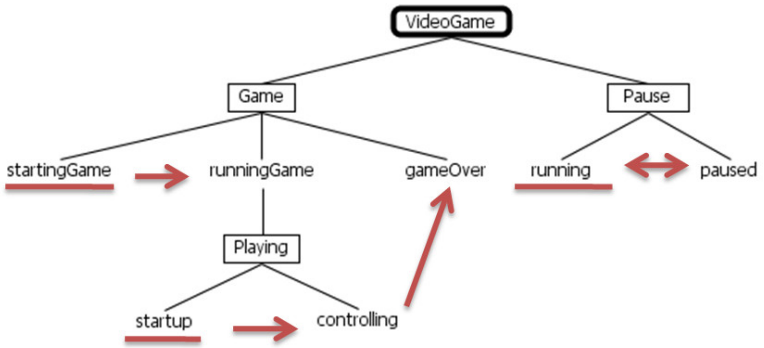
Extended Classification Tree Method

- Classification Tree Method (CTM): is a model to identify states of the software
- Extended CTM: add transitions between classes (states): similar to a hierarchical concurrent state machine



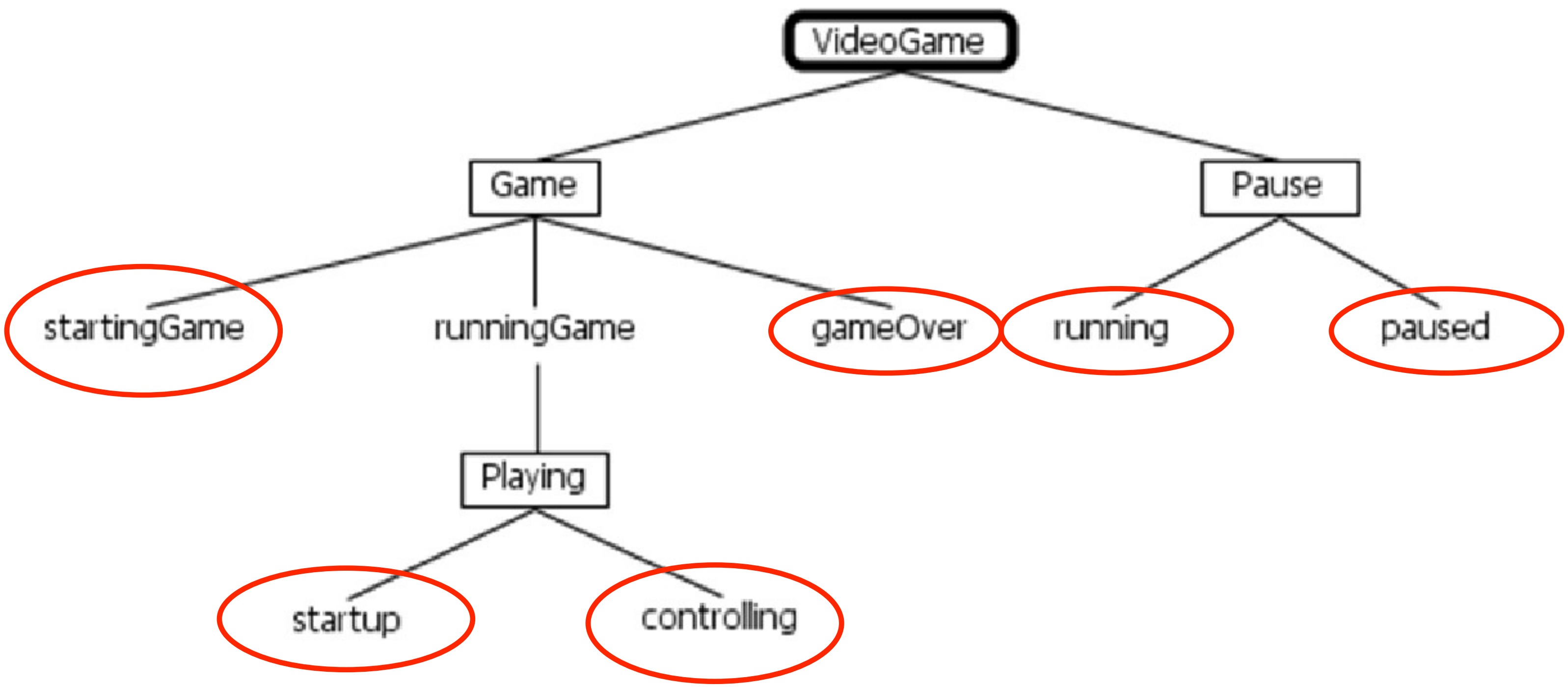
Test Sequence Generation Problem in ECTM

- **Test**: a set of classes that represents the current state of the SUT
- **Test sequence**: a sequence of tests that preserve the transition rules
- **Goal**: find a set of test sequences with the minimum number of tests to fulfill the coverage criterion
- **Coverage criteria**:
  - **Cover all the classes in the ECTM**
  - **Cover all the transitions in the ECTM**



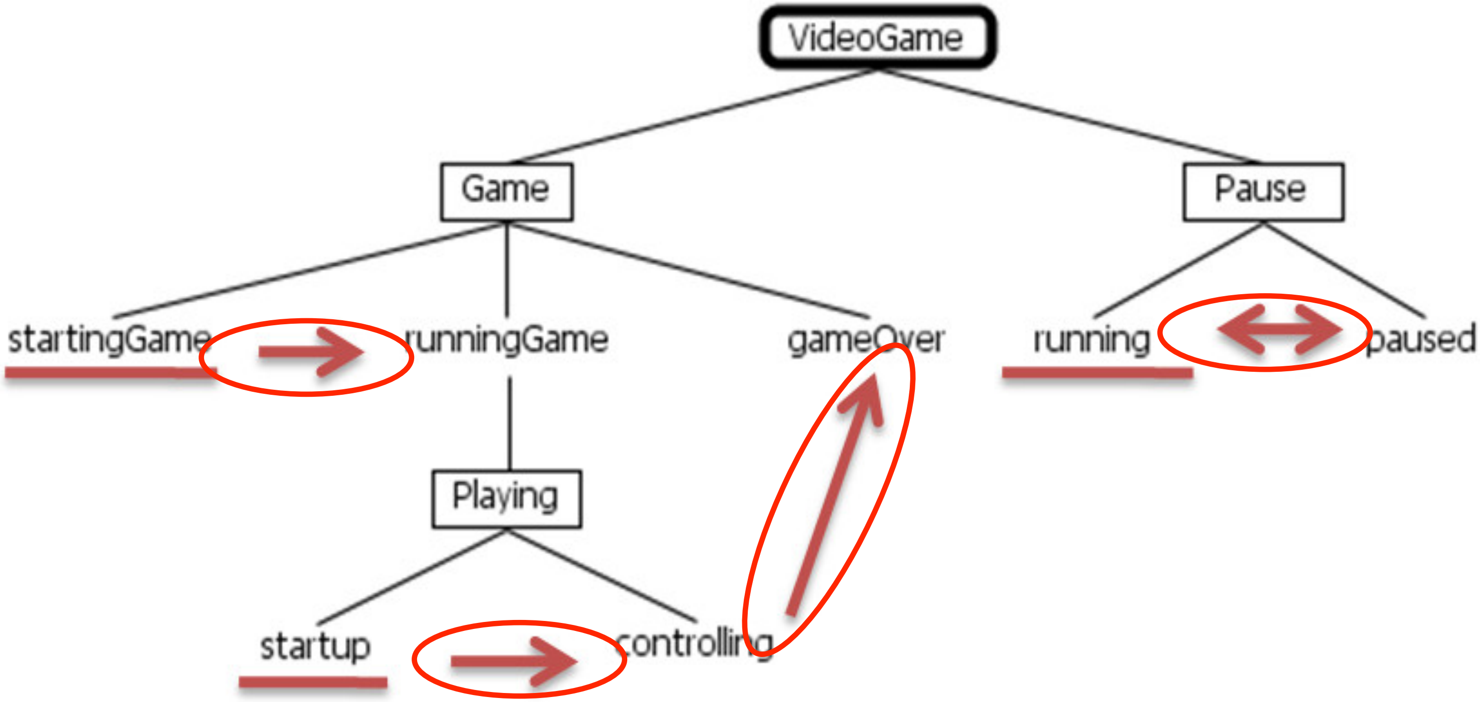
Coverage Criteria

- **Class coverage**: all classes must appear in the test sequence



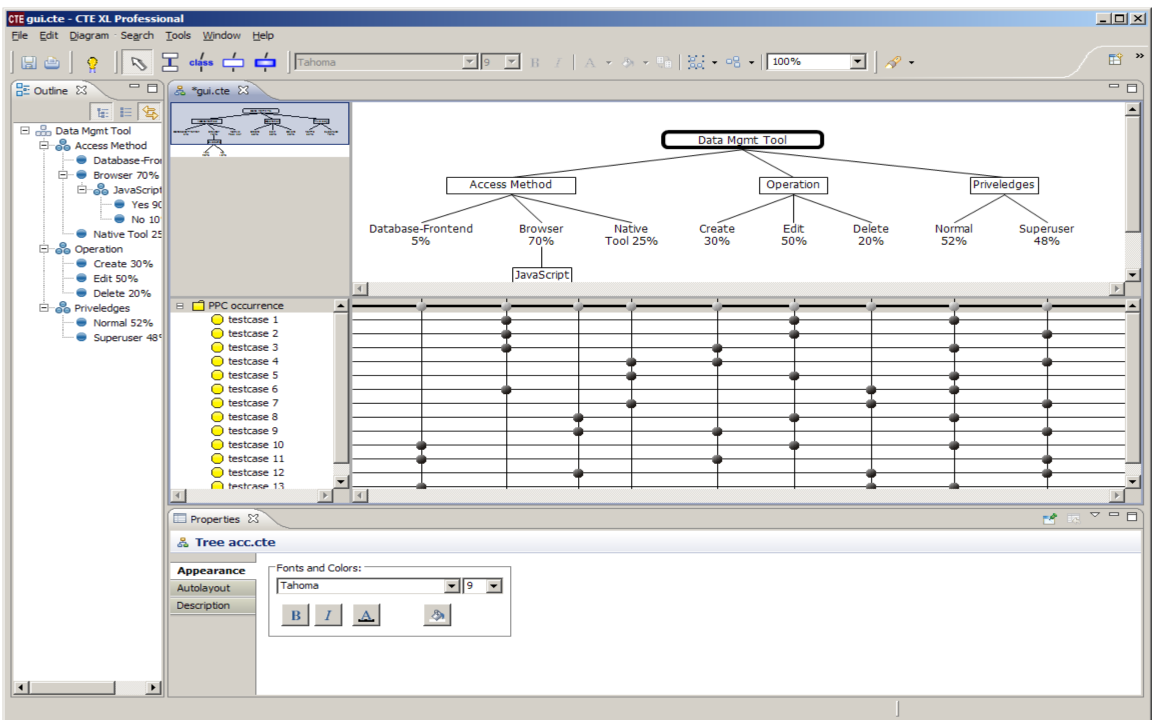
Coverage Criteria

- **Transition coverage**: all transitions must be taken in the test sequence



Algorithms

- **We developed two algorithms based on Genetic Algorithms and Ant Colony Optimization**
- **Integrated in CTE XL Professional**



BERNER & MATTNER
AN ASSYSTEM COMPANY

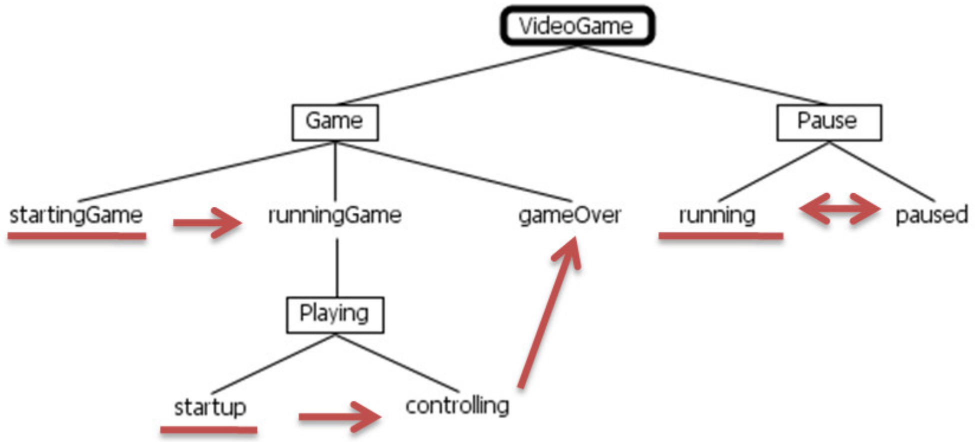
SSTQB
Spanish Software Testing Qualifications Board

Algorithms: Genetic Test Sequence Generator

- **Solution**: a sequence of integers representing the outgoing transition of a class

1	1	1	2	2	5	2	1	3	1

- **Evaluation**:
  - It starts in the initial state (*startingGame*, *running*)
  - Then, it consumes the transition vector (one number per leaf class)



Algorithms: Genetic Test Sequence Generator

- **No recombination**
- **Mutation**: position-based probability in range *[m1, m2]*

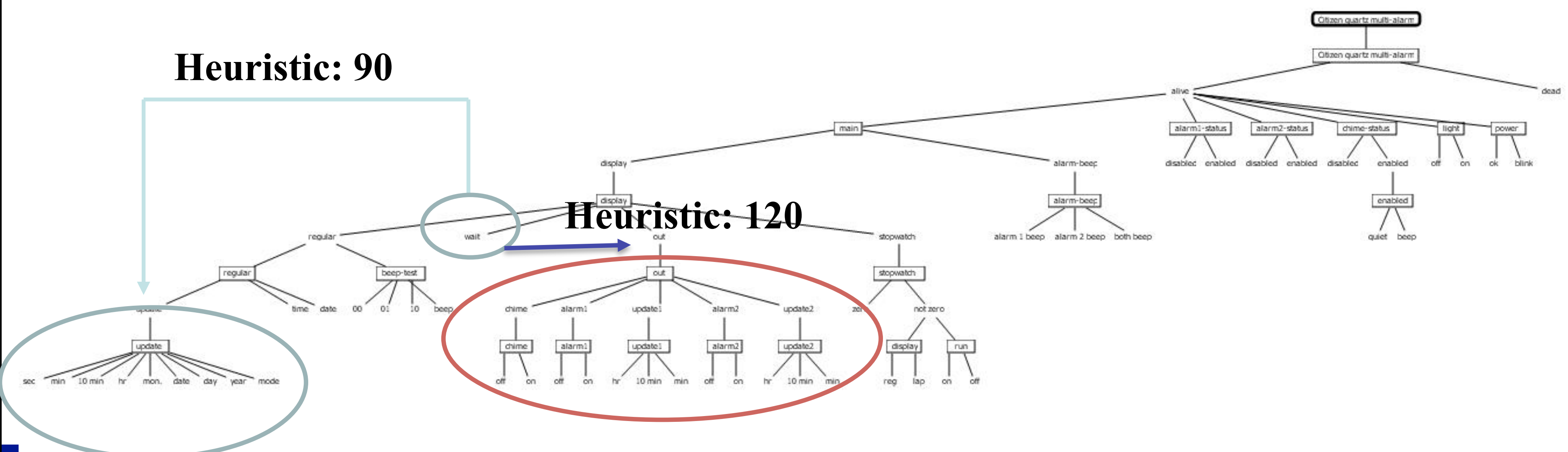
m1=0,05 → **m2=0,25**

1	1	1	2	2	5	2	1	3	1

- A change in the first positions could be a hard perturbation of the solution

Algorithms: ACO for Test Sequences

- Based on ACOhg (ACO for Huge Graphs)
- Two changes over ACOhg:
  - The goal is to reach maximum coverage, instead of shortest paths
  - There are no final nodes
- Heuristic function:
  - Designed to guide the search to unexplored regions



Experiments: benchmark

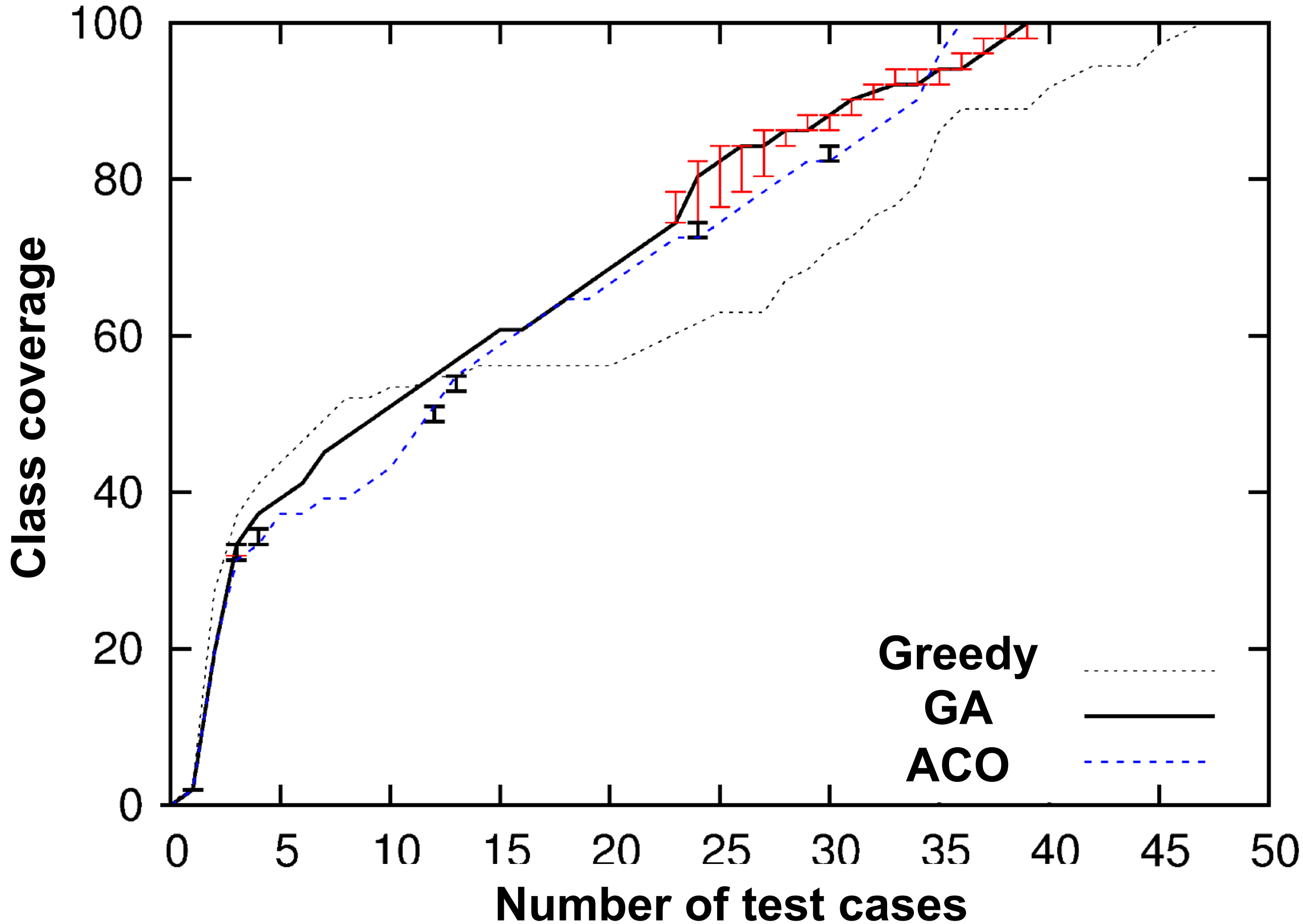
Program	Classes	Transitions	Minimal	Complete
Keyboard	5	8	2	4
Microwave	19	23	7	56
Autoradio	20	35	11	66
Citizen	62	74	31	3121
Coffee Machine	21	28	9	81
Communication	10	12	7	7
Elevator	13	18	5	80
Tetris	11	18	10	10
Mealy Moore	5	11	5	5
Fuel Control	5	27	5	600
Transmission	7	12	4	12
Aircraft	24	20	5	625

Experiments: Results (Class Coverage)

Program	GA	ACO	Greedy
Keyboard	2	2	2
Microwave	**8***	**8***	9
Autoradio	13,30*	14	**13***
Citizen	39,47*	**36****	47
Coffee Machine	9	9	9
Communication	7	7	7
Elevator	6	6	6
Tetris	**12***	**12***	15
Mealy Moore	5	5	5
Fuel Control	5	5	5
Transmission	4	4	4
Aircraft	4 (86,20%)	4 (86,20%)	4 (86,20%)

***Statistically significant difference with the worst algorithm**
****Statistically significant difference with the other algorithms**

Experiments: Results (Class Coverage)

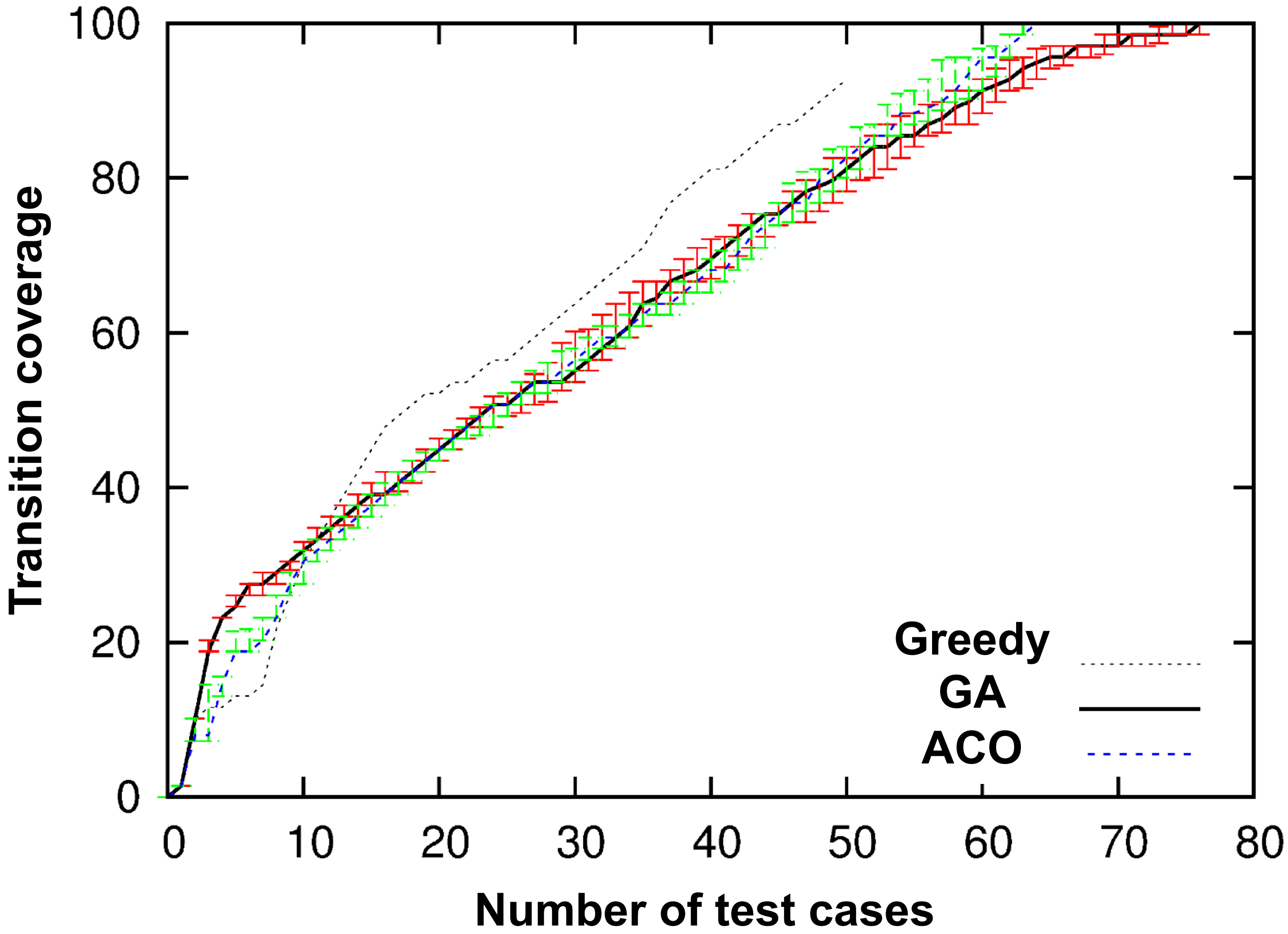


Experiments: Results (Transition Coverage)

Program	GA	ACO	Greedy
Keyboard	5	5	5
Microwave	17	17	17
Autoradio	36,30	**36**	**36**
Citizen	75,27* (99,90%)	**64,17****	51(92,70%)
Coffee Machine	19	19	**18****
Communication	**16***	**16***	17
Elevator	9	9	9
Tetris	31	31	31
Mealy Moore	24	24	24
Fuel Control	**11***	**11***	12
Transmission	9	9	9
Aircraft	7 (2)	7 (2)	7 (2)

***Statistically significant difference with the worst algorithm**
****Statistically significant difference with the other algorithms**

Experiments: Results (Transition Coverage)



Acknowledgements

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, CULTURA Y DEPORTE

GOBIERNO DE ESPAÑA
MINISTERIO DE CIENCIA E INNOVACIÓN

Colorado State University

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
UNITED STATES AIR FORCE

FULBRIGHT España

ANDALUCÍA TECH
Campus de Excelencia Internacional

VŠB 1849 OSTRAVA TECHNICKÁ UNIVERZITA

UNIVERSIDAD DE MÁLAGA

SCHLOSS DAGSTUHL
Leibniz-Zentrum für Informatik

Thanks for your attention!!!