Searching for Diverse Software Engineering Solutions

Robert Feldt, robert.feldt@chalmers.se

23rd of March 2012, COW18, London

HOSE Lab (Human-fOcused SE)
Division of Software Engineering
Chalmers Univ of Tech, Sweden

How make software less brittle / more robust?

How make software less brittle / more robust?

Engineered Systems
often brittle!

How make software less brittle / more robust?

Engineered Systems
often brittle!

Biological Systems
sometimes robust!

How make software less brittle / more robust?

?

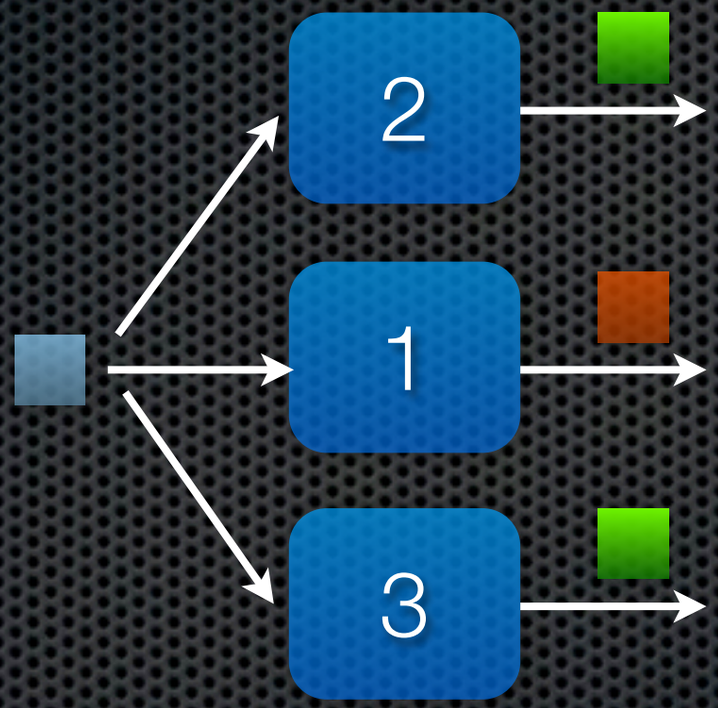
Engineered Systems
often brittle!

Biological Systems
sometimes robust!

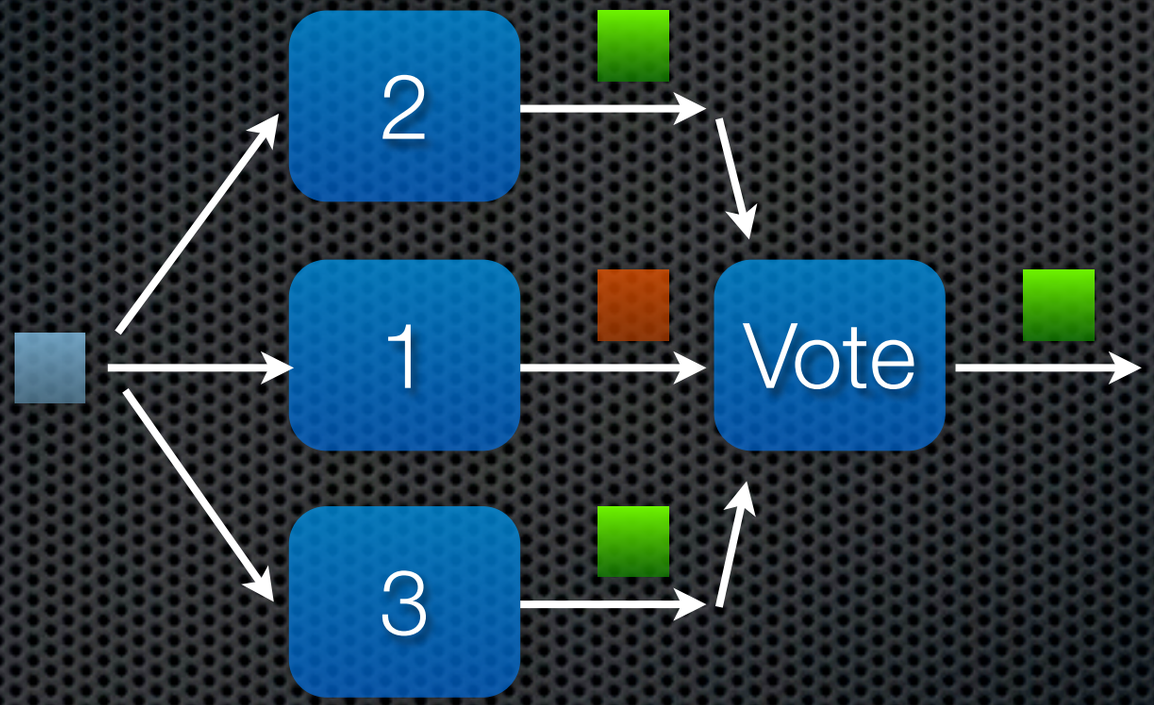
N-Version Programming

N-Version Programming

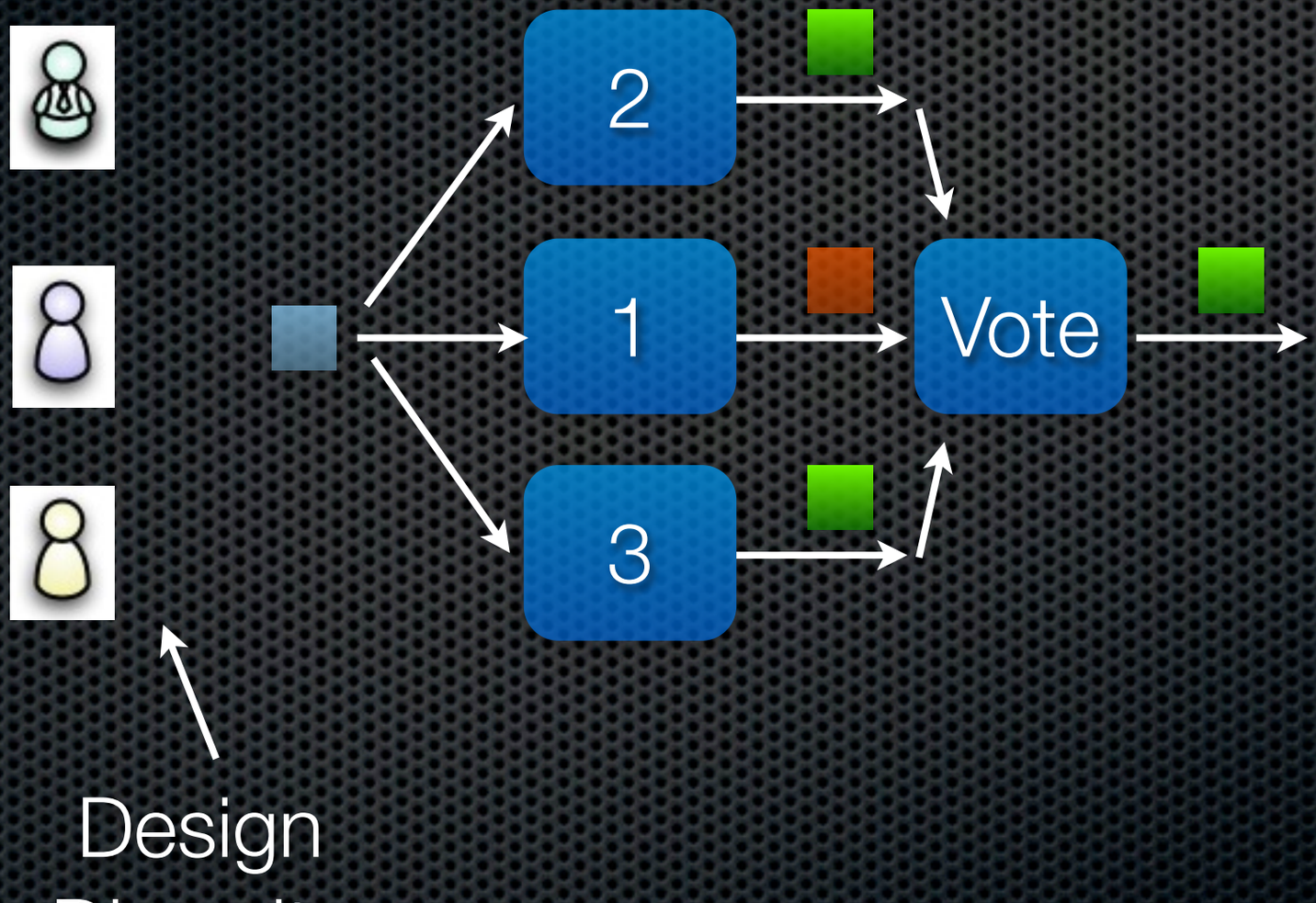
N-Version Programming



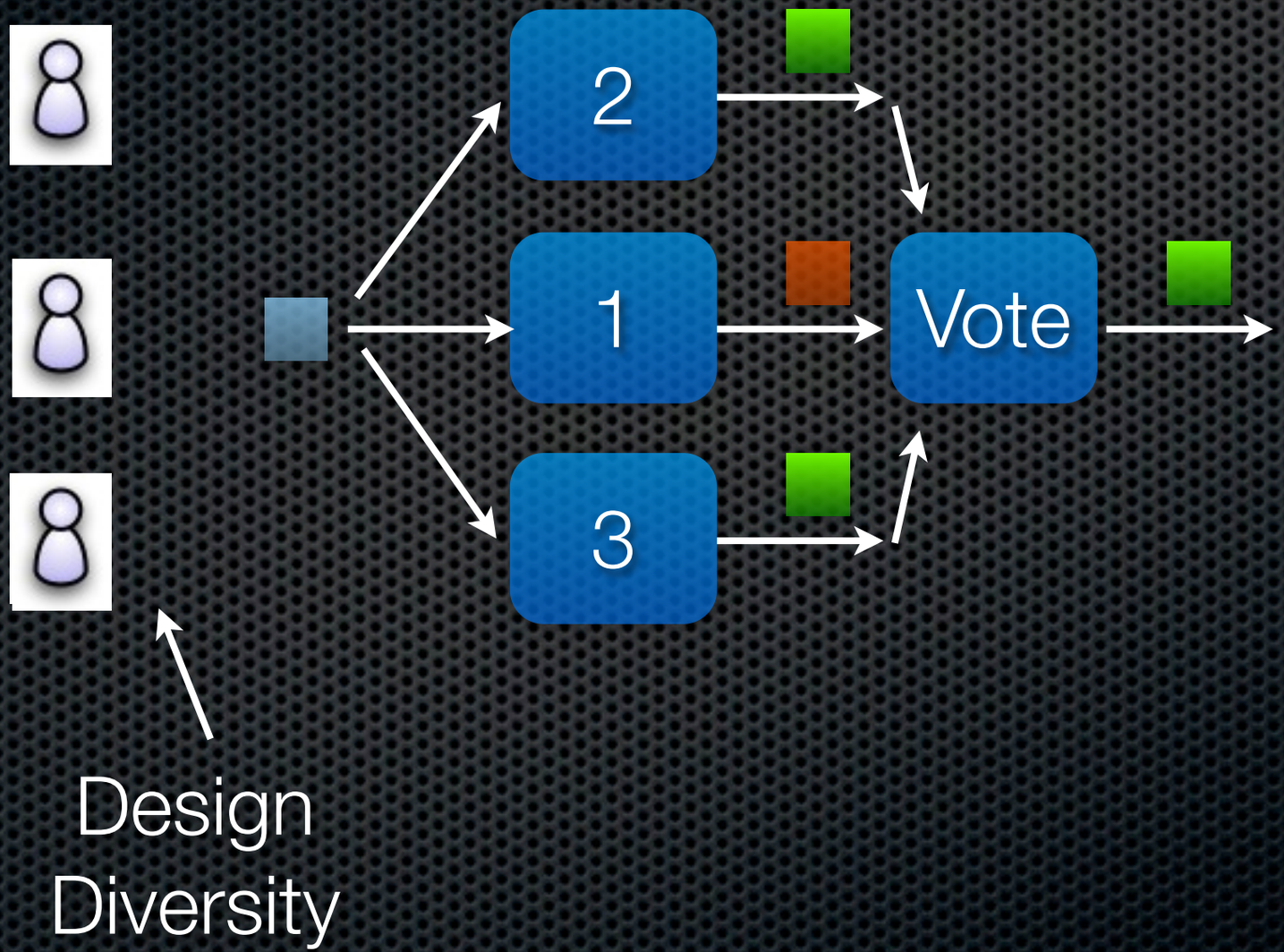
N-Version Programming



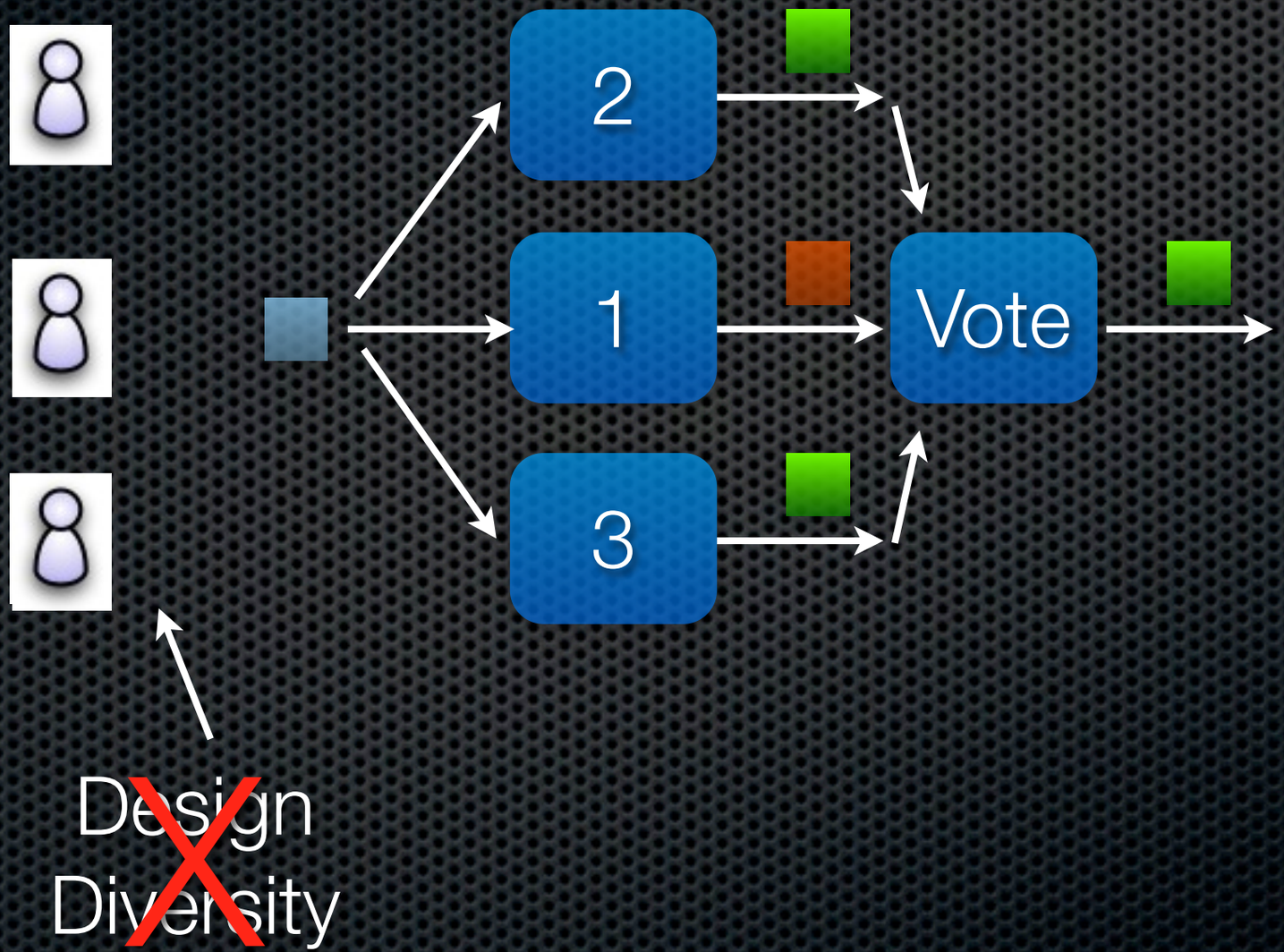
N-Version Programming



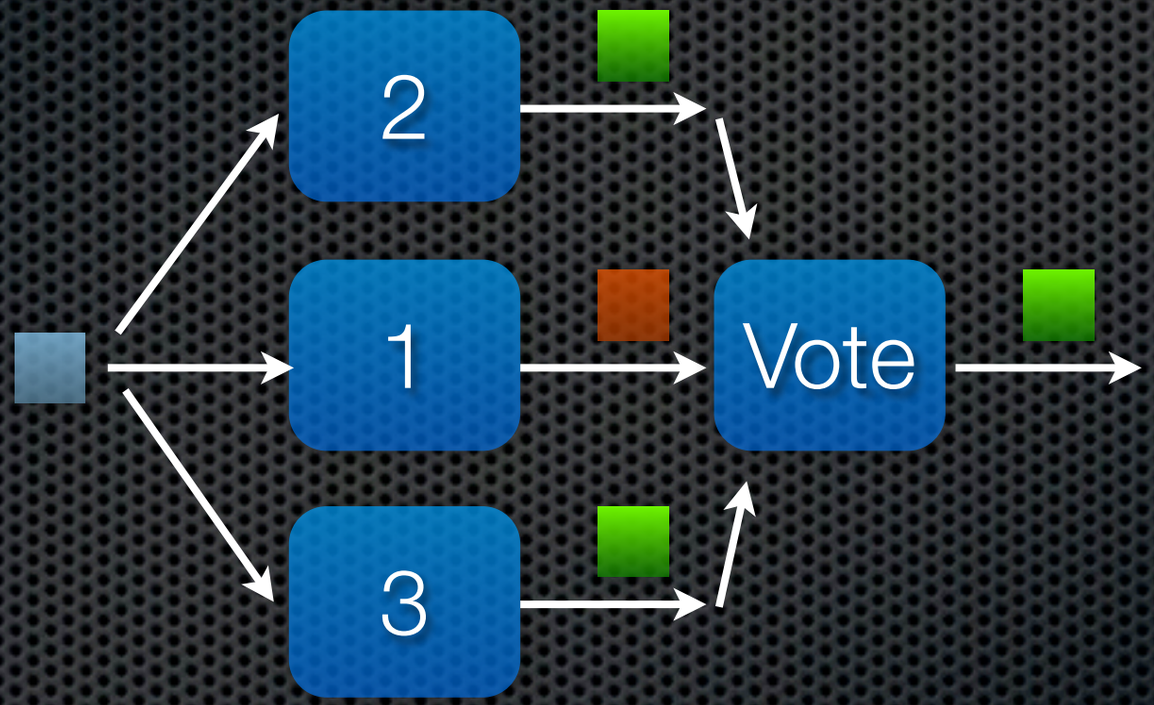
N-Version Programming



N-Version Programming

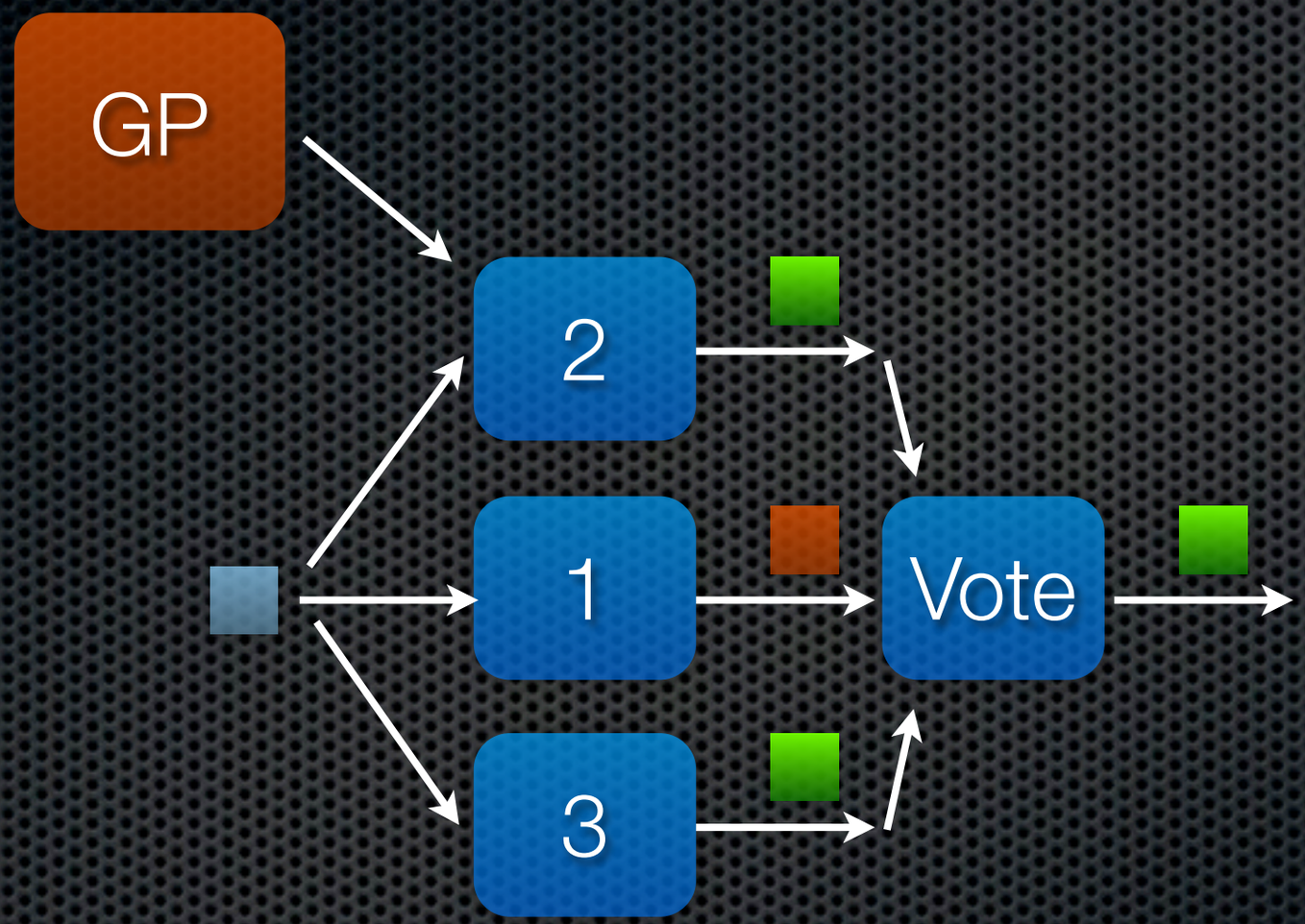


N-Version Programming



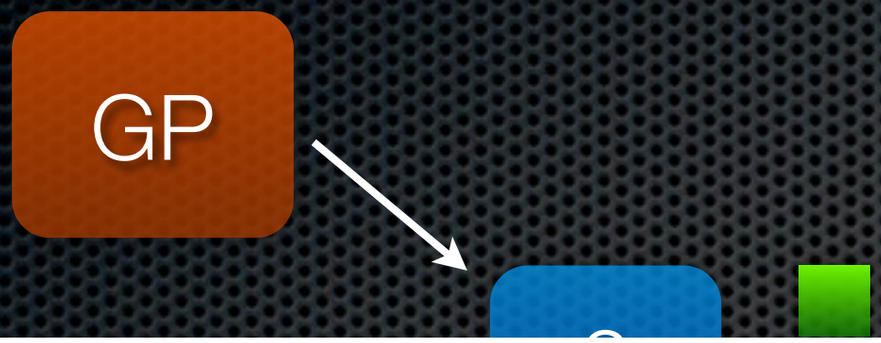
Design Diversity

N-Version Programming



Design Diversity

N-Version Programming



Generating diverse software versions with genetic programming: an experimental study

R. Feldt

Indexing terms: Design diversity, Fault tolerance, Genetic programming

Abstract: Software fault-tolerance schemes often employ multiple software versions developed to meet the same specification. If the versions fail independently of each other, they can be combined to give high levels of reliability. Although design diversity is a means to develop these versions, it has been questioned because it increases development costs and because

common-mode failures, i.e. several versions failing for the same input, and limit the diversity that can be achieved. Experimental research has shown that there are systems for which the independence assumption is not valid [2]. The strength of using design diversity has thus been questioned.

In [3], the term random diversity was proposed to denote the above scenario: generation of diversity is left to chance and arises from differences in the back-

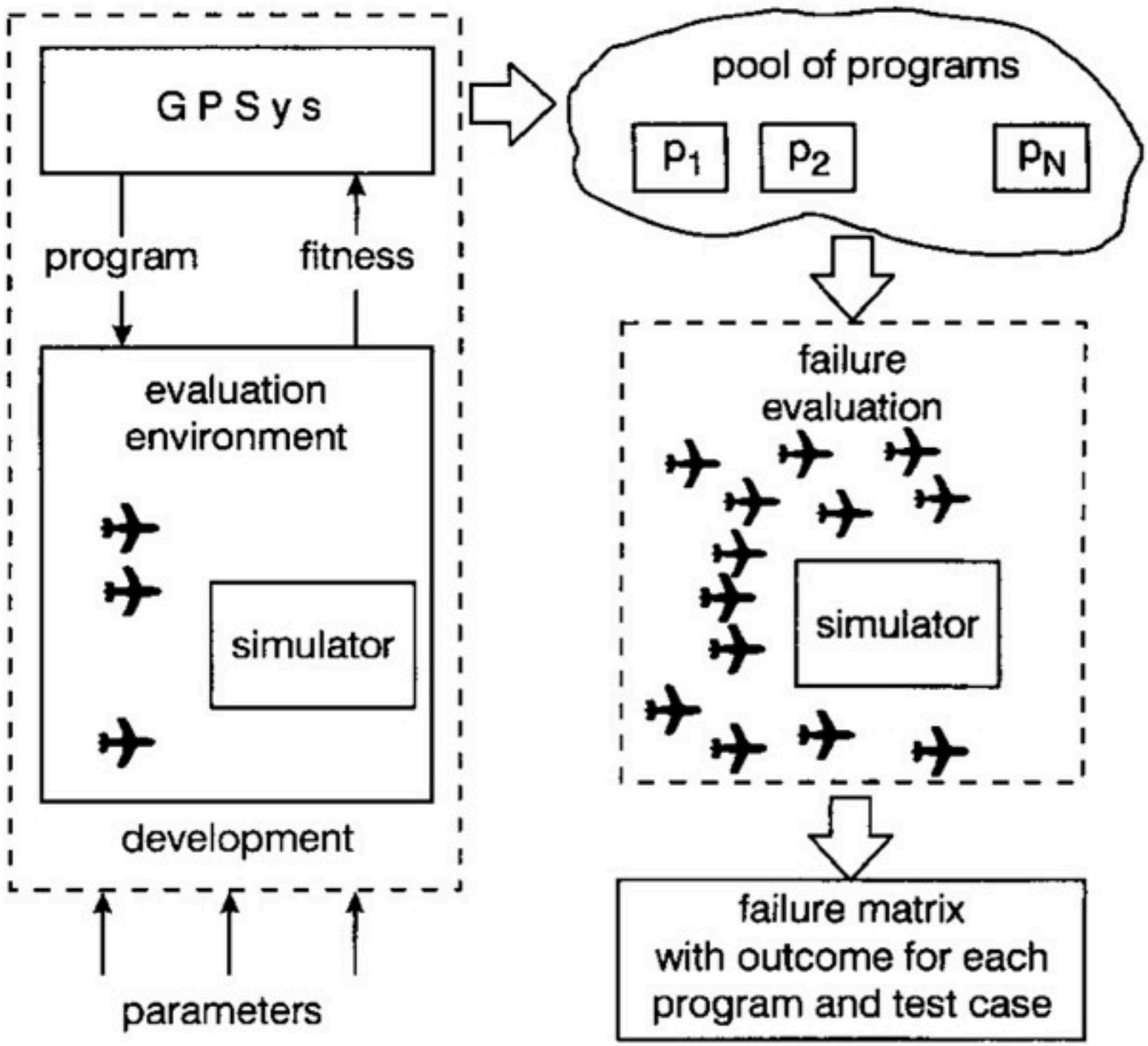


Fig. 1 *Experiment environment for developing and evaluating aircraft arrestment controllers*

15 US Air Force - 99: Military Specification: Aircraft Arresting System BAK-12A/E32A; Portable, Rotary Friction, MIL-A-38202C, Notice 1, US Department of Defense, 1986

Table 2: Description of parameters varied in experiment and their levels

Factors	Levels	Type	Description	Anticipated effect/Motivation
A	−1	PSP	no effect.	for comparison of values during braking
	1		the statement If, and operators LE, And and Not can be used in programs	
B	−1	PSP	no effect.	for oscillatory and/or damping behaviour
	1		the functions Sinus and Exp can be used in the programs	
C	−1	PSP	the average velocity, average retardation and index to current checkpoint can be used in programs	for structural diversity; average velocity and retardation are pre-calculated before they can be used in programs
	1		the angular velocity, current time since start of braking, previous angular velocity and time of previous checkpoint can be used in programs	
D	−1	PSP	programs cannot use any subroutines	For greater program complexity without need for one long program
	1		two subroutines (automatically defined functions) can be used in program; they are evolved in same manner as rest of program	
E	−1	EP	maximum penalty on retardation failure criterion is 1000.0	force programs to find solutions that solve retardation criterion with higher priority than other criteria
	1		maximum penalty on retardation failure criterion is 2000.0.	
F	−1	EP	linear penalties are not used	without linear penalties, fitness only expresses 'amount' of failure; performance on non-failure aspects is not measured
	1		linear penalties are used, and maximum penalty of 30.0 is assigned to each failure criterion	
G	−1	EP	25 test cases uniformly spread over range of possible values for mass and velocity are used to evaluate fitness during evolution	uniform spreading of test cases 'samples' all parts of possible input cases; random spreading can give both easier and more difficult test cases
	1		25 test cases chosen randomly for each run of the GP system are used to evaluate fitness during evolution	
H	−1	SP	probability of mutation is 0.05	initial experiments indicated that high values might be beneficial
	1		probability of mutation is 0.6	

Table 2: Description of parameters varied in experiment and their levels

Factors	Levels	Type	Description	Anticipated effect/Motivation
A	−1	PSP	no effect.	for comparison of values during braking
	1		the statement If, and operators LE, And and Not can be used in programs	
B	−1	PSP	no effect.	for oscillatory and/or damping behaviour
	1		the functions Sinus and Exp can be used in the programs	
C	−1	PSP	the average velocity, average retardation and index to current checkpoint can be used in programs	for structural diversity; average velocity and retardation are pre-calculated before they can be used in programs
	1		the angular velocity, current time since start of braking, previous angular velocity and time of previous checkpoint can be used in programs	
D	−1	PSP	programs cannot use any subroutines	For greater program complexity without need for one long program
	1		two subroutines (automatically defined functions) can be used in program; they are evolved in same manner as rest of program	
E	−1	EP	maximum penalty on retardation failure criterion is 1000.0	force programs to find solutions that solve retardation criterion with higher priority than other criteria
	1		maximum penalty on retardation failure criterion is 2000.0.	
F	−1	EP	linear penalties are not used	without linear penalties, fitness only expresses 'amount' of failure; performance on non-failure aspects is not measured
	1		linear penalties are used, and maximum penalty of 30.0 is assigned to each failure criterion	
G	−1	EP	25 test cases uniformly spread over range of possible values for mass and velocity are used to evaluate fitness during evolution	uniform spreading of test cases 'samples' all parts of possible input cases; random spreading can give both easier and more difficult test cases
	1		25 test cases chosen randomly for each run of the GP system are used to evaluate fitness during evolution	
H	−1	SP	probability of mutation is 0.05	initial experiments indicated that high values might be beneficial
	1		probability of mutation is 0.6	

Failure Diversity

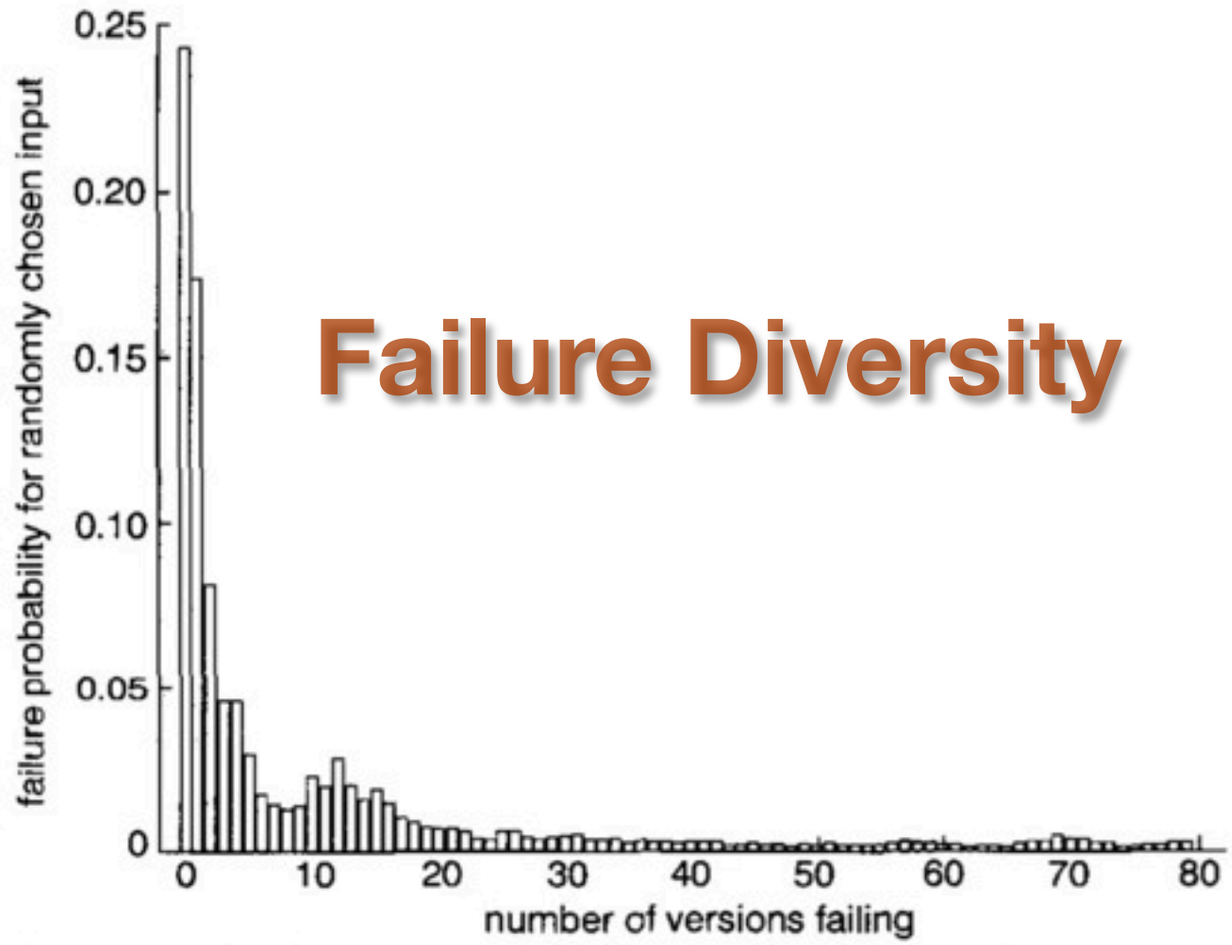


Fig.2 *Probability of failure of n versions for randomly chosen input*

# test cases	# failing programs
0	0
22	79
24	78

Structural Diversity

Size Statistic	Value
Max	459
Average	100.2
Min	17

Diversity at different levels

Diversity at different levels

Pair-wise Failure Diversity between Programs

Statistic	All	Top 10
Min correlation	-0.21	0.55
Max Distinct failures	98.9%	59.7%

Diversity at different levels

Pair-wise Failure Diversity between Programs

Statistic	All	Top 10
Min correlation	-0.21	0.55
Max Distinct failures	98.9%	59.7%

Pair-wise Failure Diversity between Methods

Most are high, 11 below 0.20, and 2 negative (but they are worst)

Diversity at different levels

Pair-wise Failure Diversity between Programs

Statistic	All	Top 10
Min correlation	-0.21	0.55
Max Distinct failures	98.9%	59.7%

Pair-wise Failure Diversity between Methods

Most are high, 11 below 0.20, and 2 negative (but they are worst)

Failure Diversity Inter-Method vs Intra-Method

Inter-Method Diversity higher ($p<10^{-10}$)

Diversity at different levels

Pair-wise Failure Diversity between Programs

Statistic	All	Top 10
Min correlation	-0.21	0.55
Max Distinct failures	98.9%	59.7%

Pair-wise Failure Diversity between Methods

Most are high, 11 below 0.20, and 2 negative (but they are worst)

Failure Diversity Inter-Method vs Intra-Method

Inter-Method Diversity higher ($p<10^{-10}$)

34% of all 3VP voters (worst case) improved, best was -20%

Explanations?

Explanations?

Diversity Prediction Theorem [Page2007]:

Crowd error = Average Individual Error - Diversity among Individuals

Explanations?

Diversity Prediction Theorem [Page2007]:

Crowd error = Average Individual Error - Diversity among Individuals

Diversity trumps Ability [Hong&Page2004]:

Random group of solvers often outperform a group of best solvers

Explanations?

Diversity Prediction Theorem [Page2007]:

Crowd error = Average Individual Error - Diversity among Individuals

Diversity trumps Ability [Hong&Page2004]:

Random group of solvers often outperform a group of best solvers

Logic:

Ideal group = High ability & Diverse

But: Larger pool of solvers => Best solvers more similar =>

Highest-ability solvers are not diverse

Group size matters:

Too small => Random solvers overlap, Too large => Best solvers can differ

Explanations?

Diversity Prediction Theorem [Page2007]:

Crowd error = Average Individual Error - Diversity among Individuals

Diversity trumps Ability [Hong&Page2004]:

Random group of solvers often outperform a group of best solvers

Logic:

Ideal group = High ability & Diverse

But: Larger pool of solvers => Best solvers more similar =>

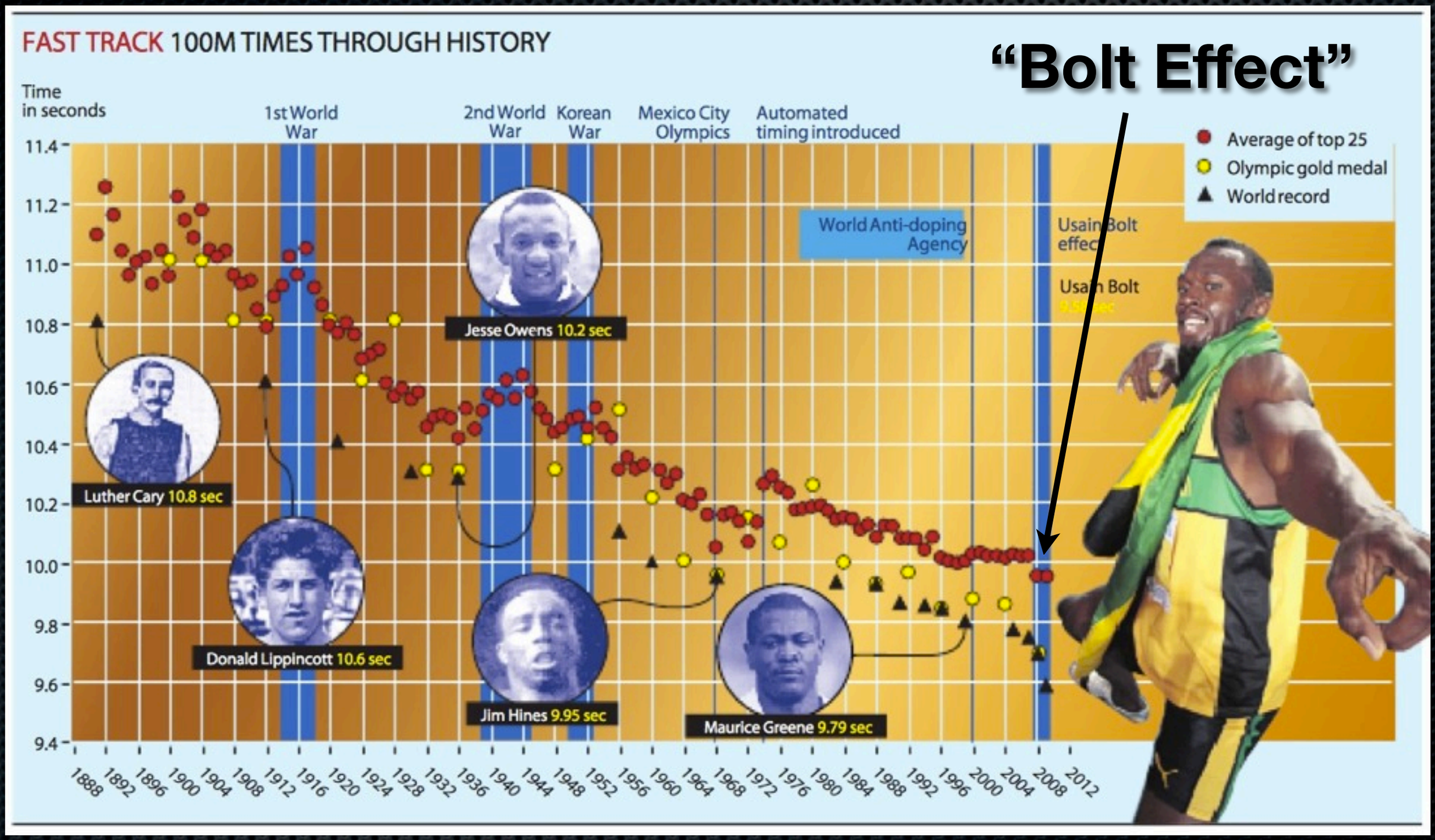
Highest-ability solvers are not diverse

Group size matters:

Too small => Random solvers overlap, Too large => Best solvers can differ

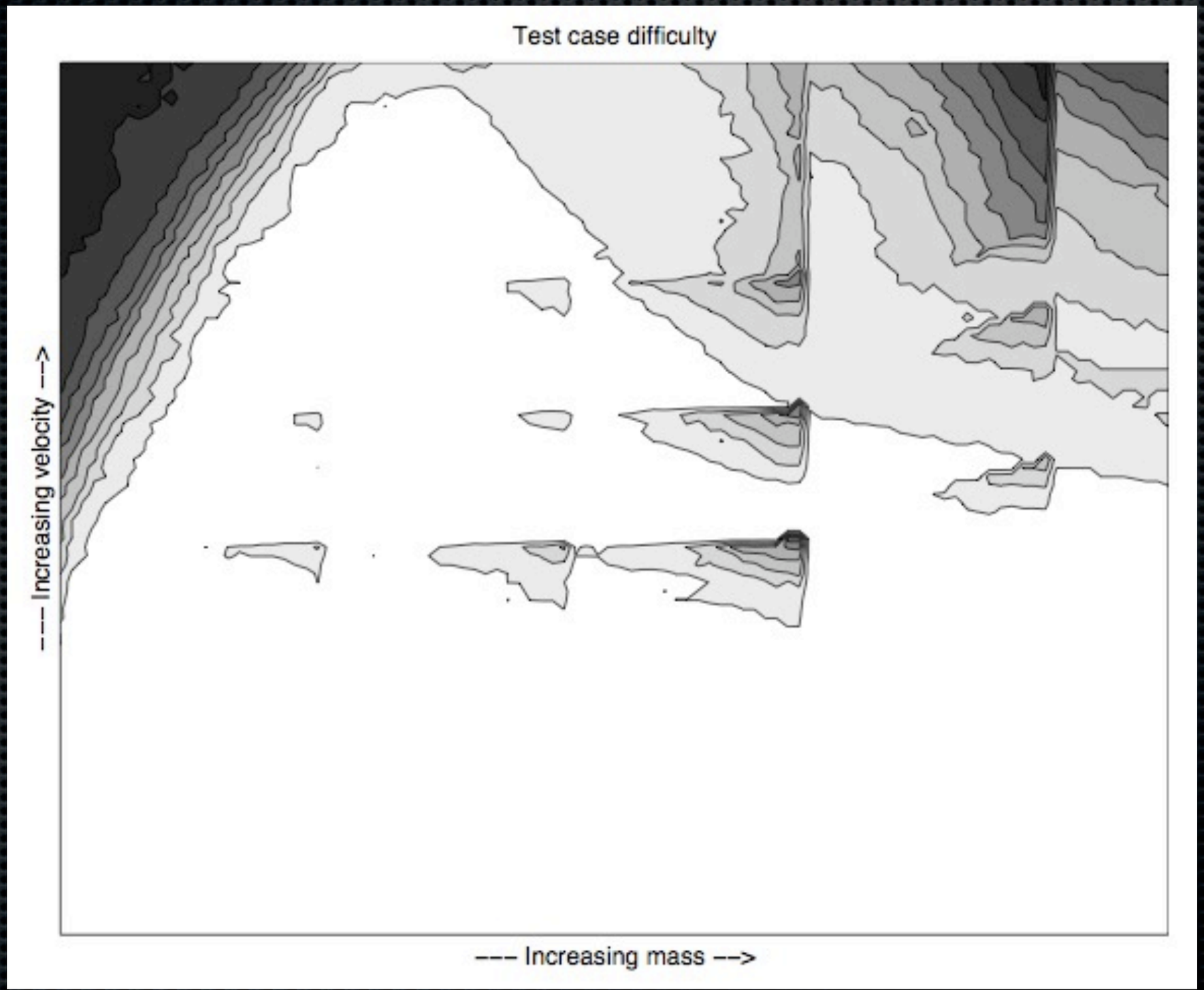
BUT, disregards Communication and Learning

But Diversity is not a simple concept...

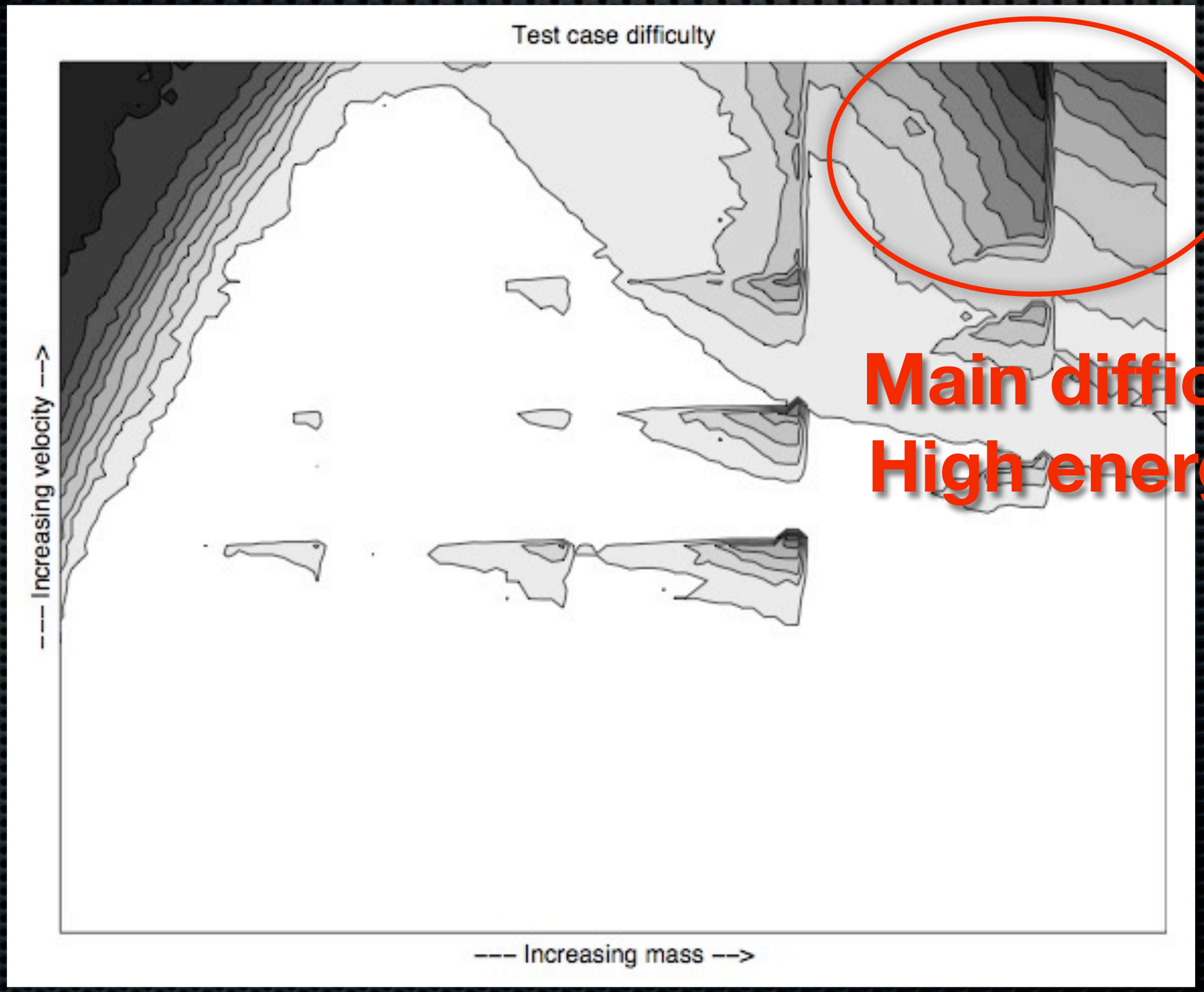


[The Indepedent, March 23 2012]

Analysis of Failures of GP programs

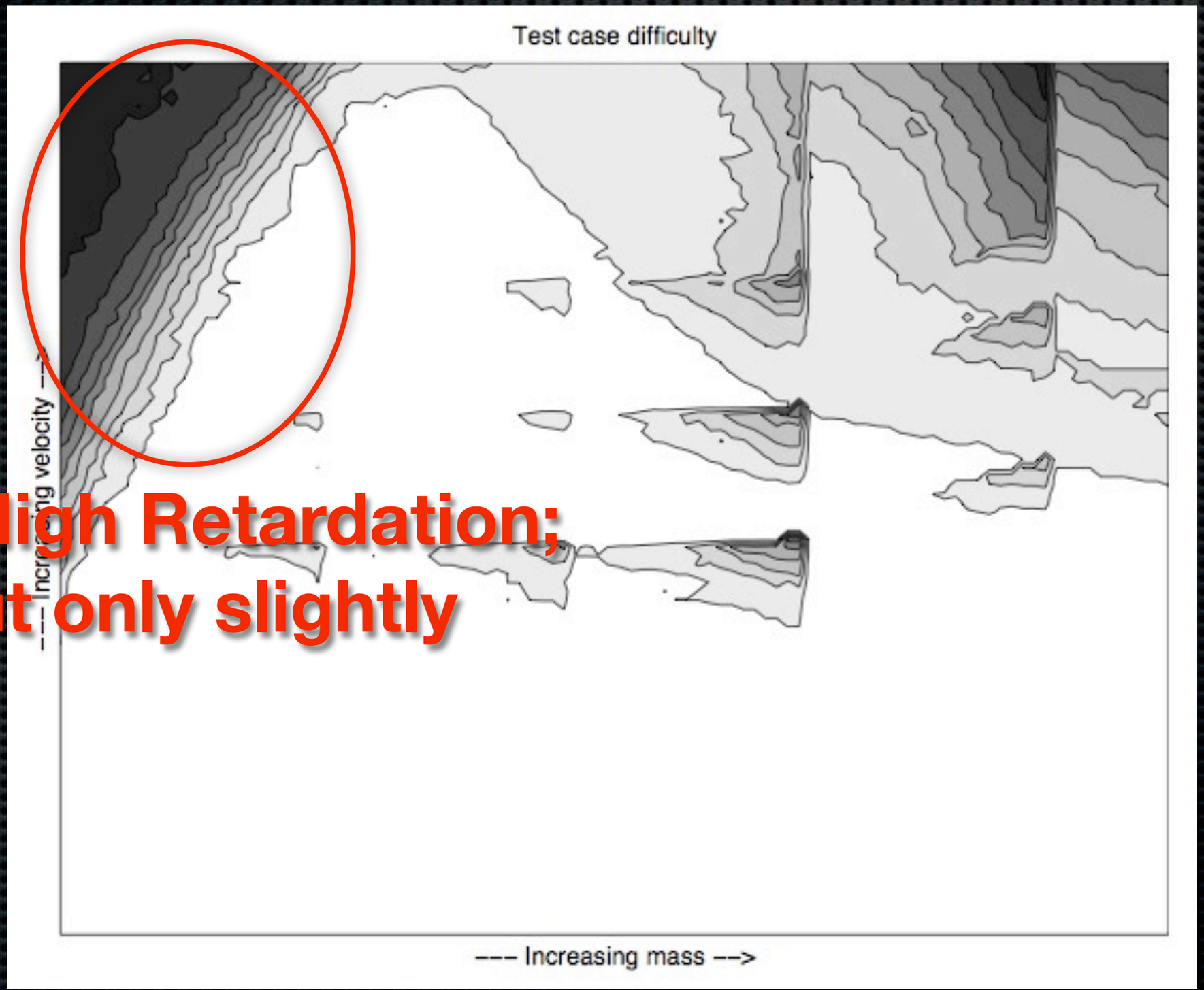


Analysis of Failures of GP programs



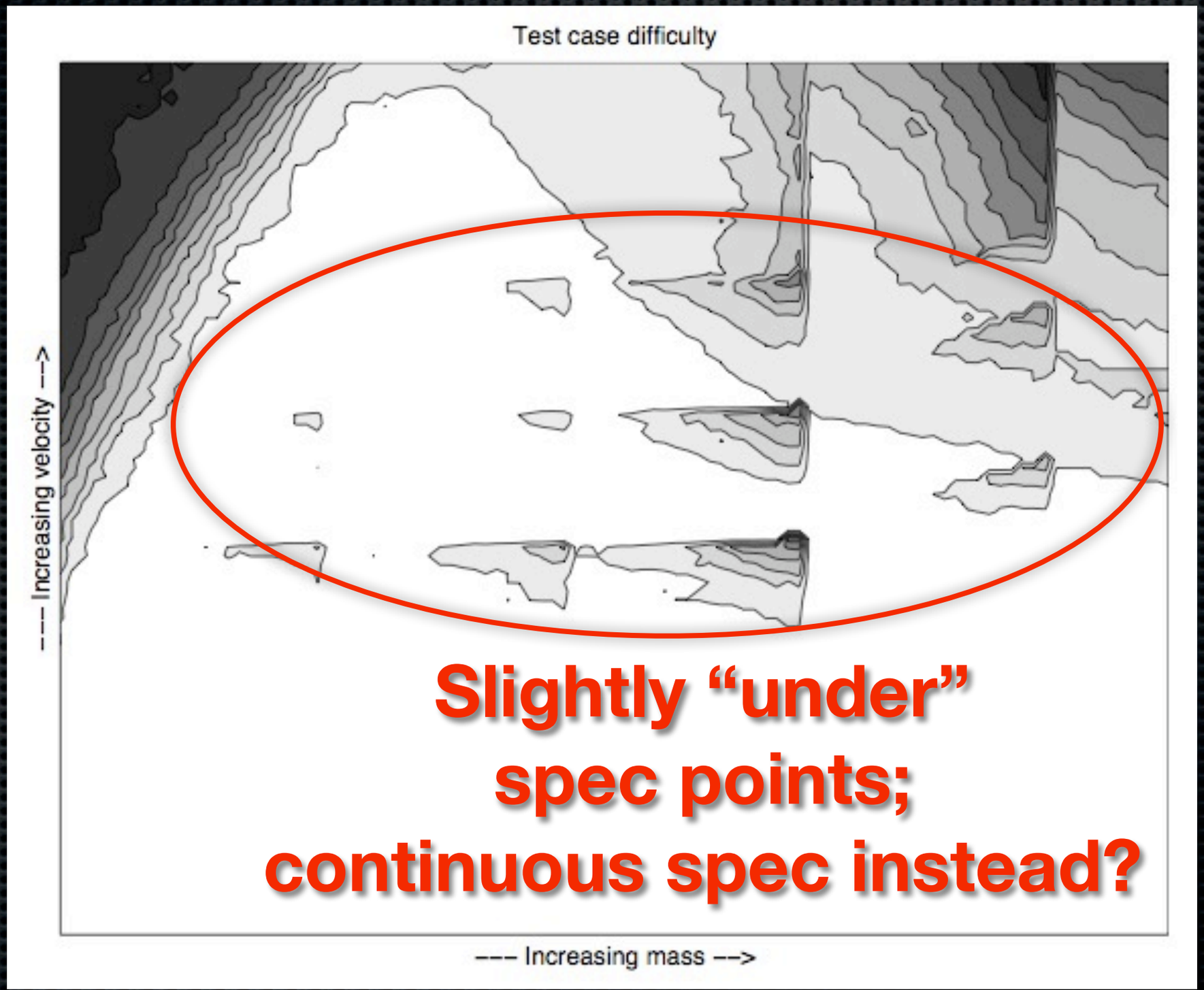
Main difficulty: High energies!

Analysis of Failures of GP programs

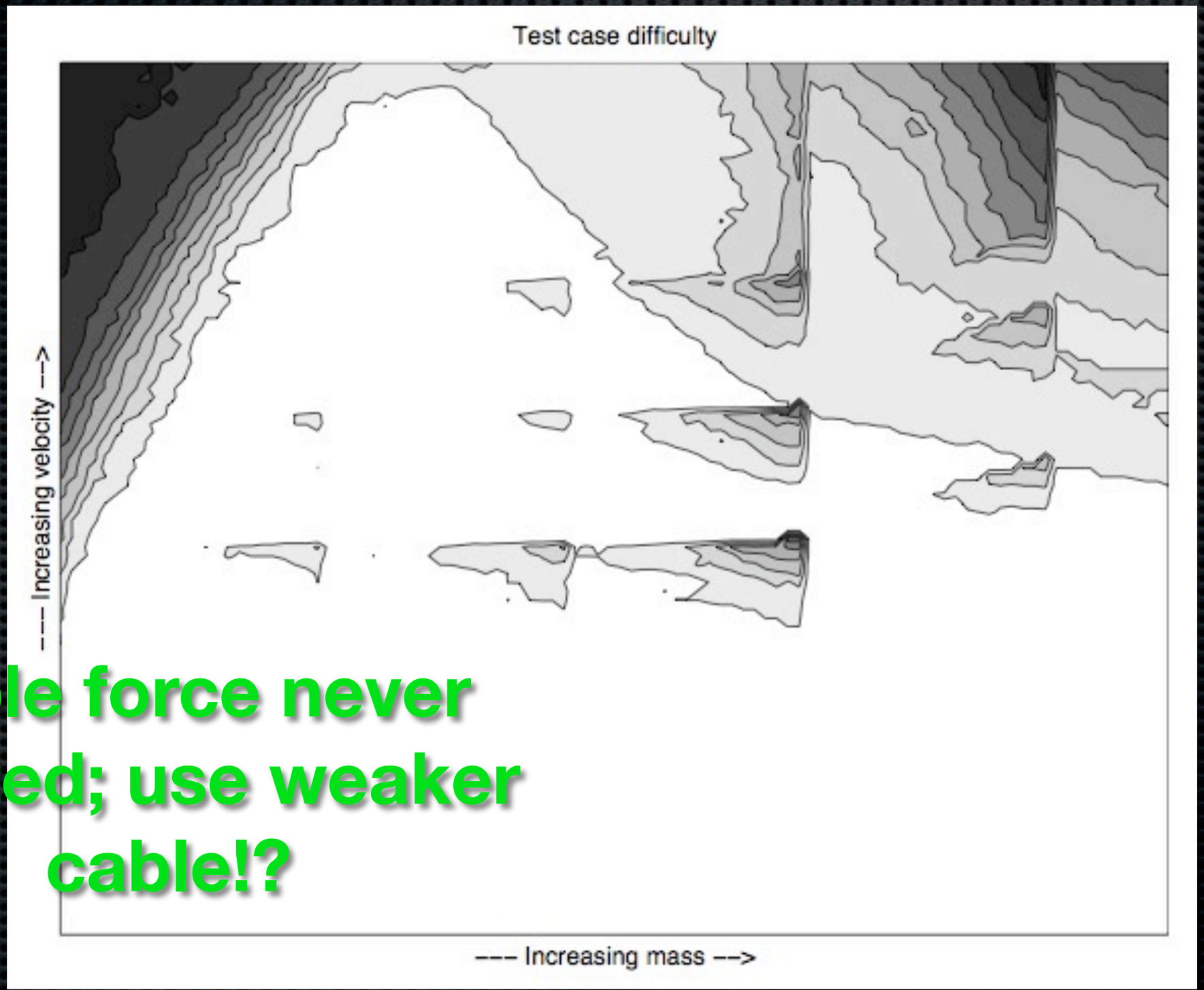


Too High Retardation; but only slightly

Analysis of Failures of GP programs

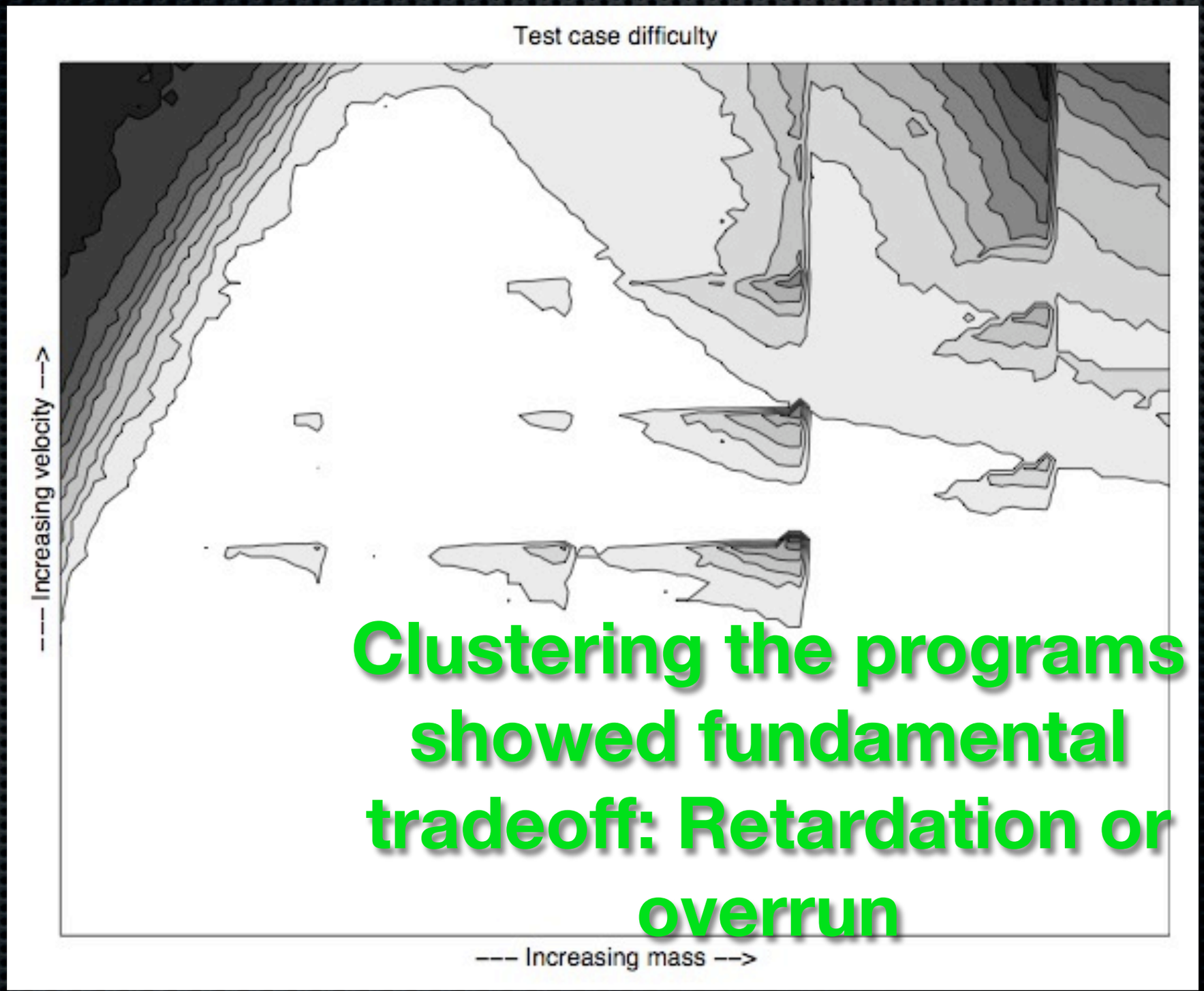


Analysis of Failures of GP programs



Cable force never violated; use weaker cable!?

Analysis of Failures of GP programs



Clustering the programs showed fundamental tradeoff: Retardation or overrun

Many Limitations

Many Limitations

- Small target application

Many Limitations

- Small target application
- Few requirements

Many Limitations

- Small target application
- Few requirements
- Low-dimensional input space

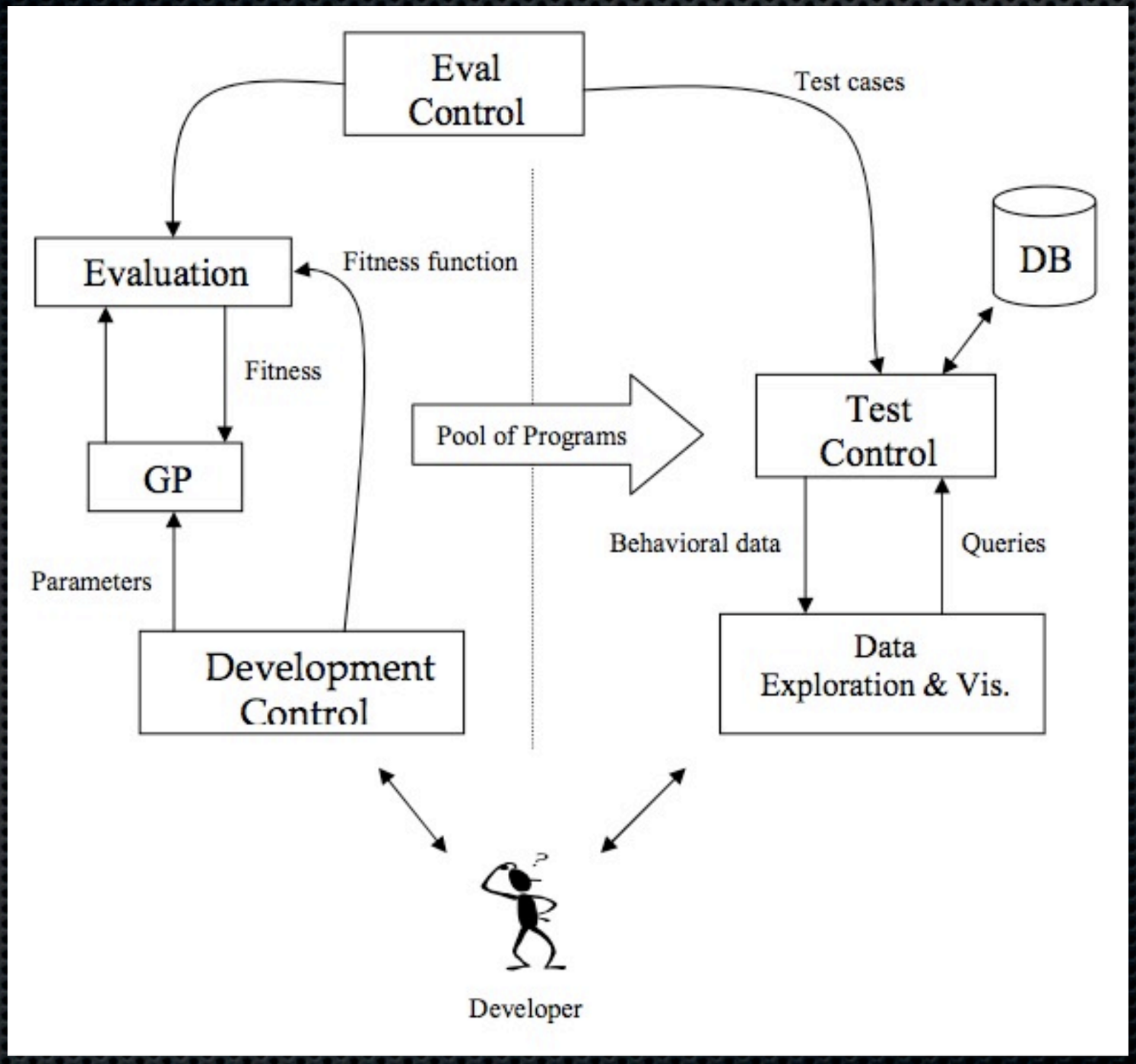
Many Limitations

- Small target application
- Few requirements
- Low-dimensional input space
- Existing simulator; typically not available in early phases

Many Limitations

- Small target application
- Few requirements
- Low-dimensional input space
- Existing simulator; typically not available in early phases
- Fundamental assumption: SB AutoProgramming fail in similar ways to human programmers

Generalization: Search-Based SW Prg Exploration



Questions?

robert.feldt@chalmers.se

http://www.cse.chalmers.se/~feldt/