Evolutionary Testing: A Case Study

Haifa Verification Conference 2006

Stella Levin and Amiram Yehudai

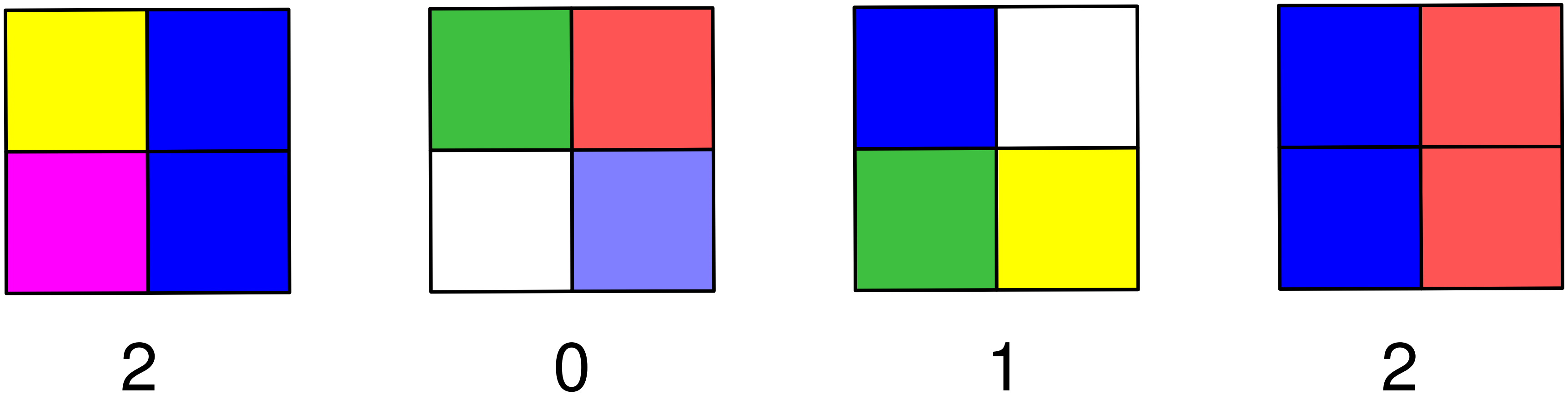
Tel-Aviv University

Contents

- GA introduction
- Testing problem as an optimization problem
- Testing system description
- Experiments
- Conclusions and plans

Genetic Algorithms

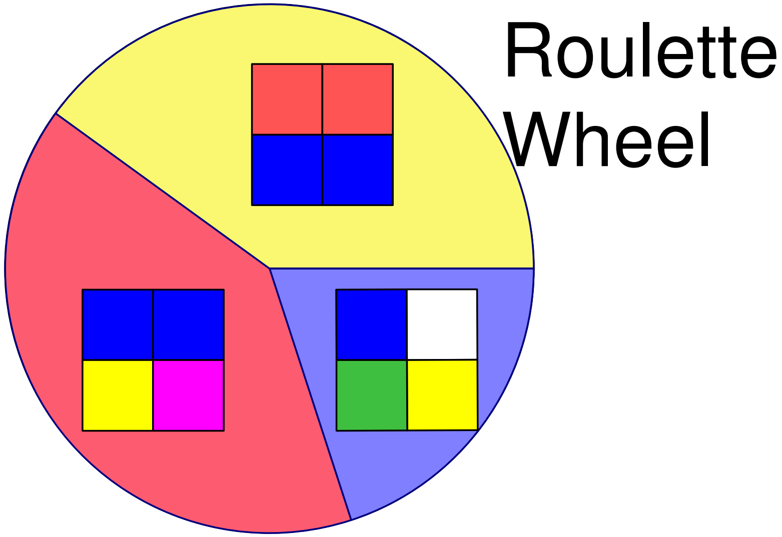
- Start from random population of individuals
  Goal: get an individual with all blue squares



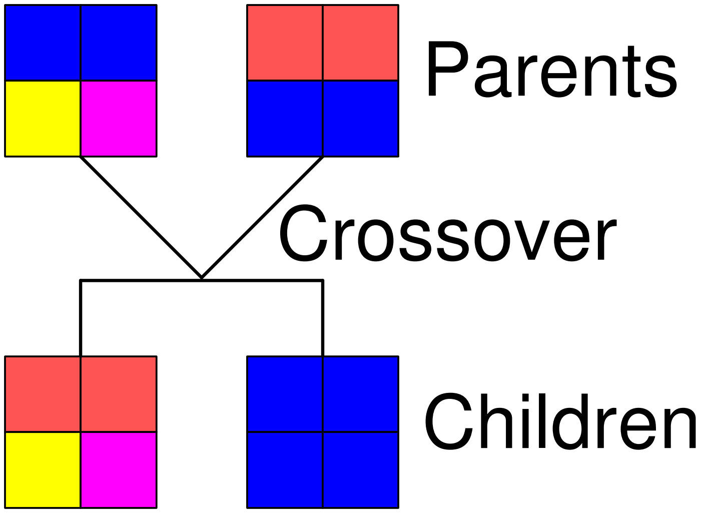
- Evaluate fitness of all individuals: count how many blue squares every individual has

Genetic Algorithms

- Selection: individuals with better fitness have more chance to be selected



- Crossover: switches genes of parents

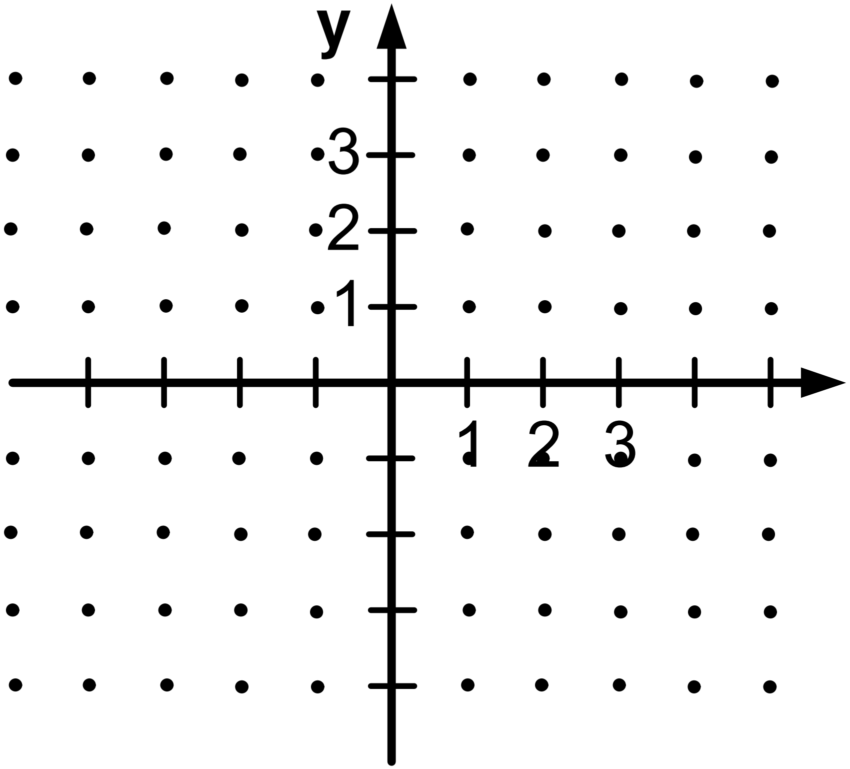


Program Domain

```
int gcd(int a, int b)
{
    while (true) {
          int r = a % b;
          if (r == 0)
                break;
          else {
                a = b;
                b = r;
          }
    }
    return b;
}
```

Input xi from domain Di

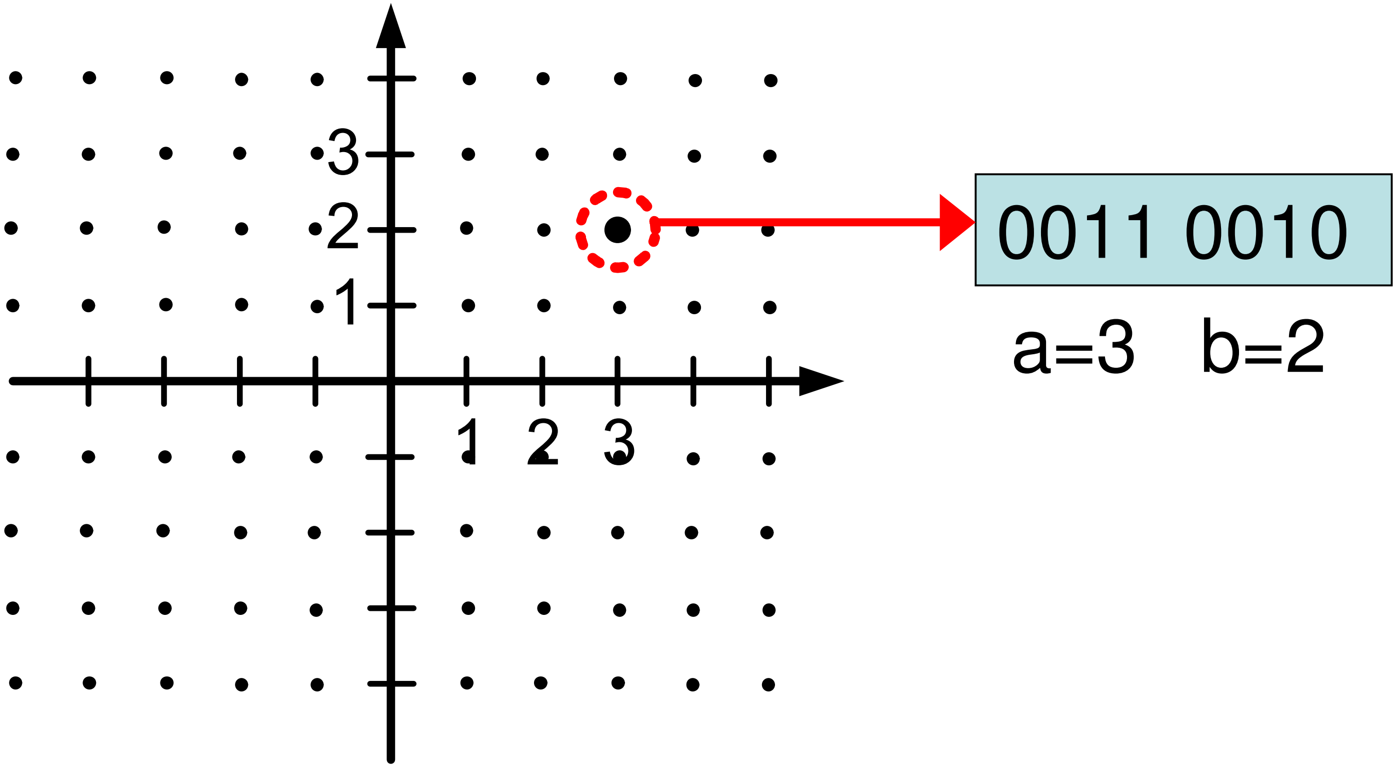
D = D1×D2×…Dn



Domain Encoding

- Encode point in D as an individual in GA

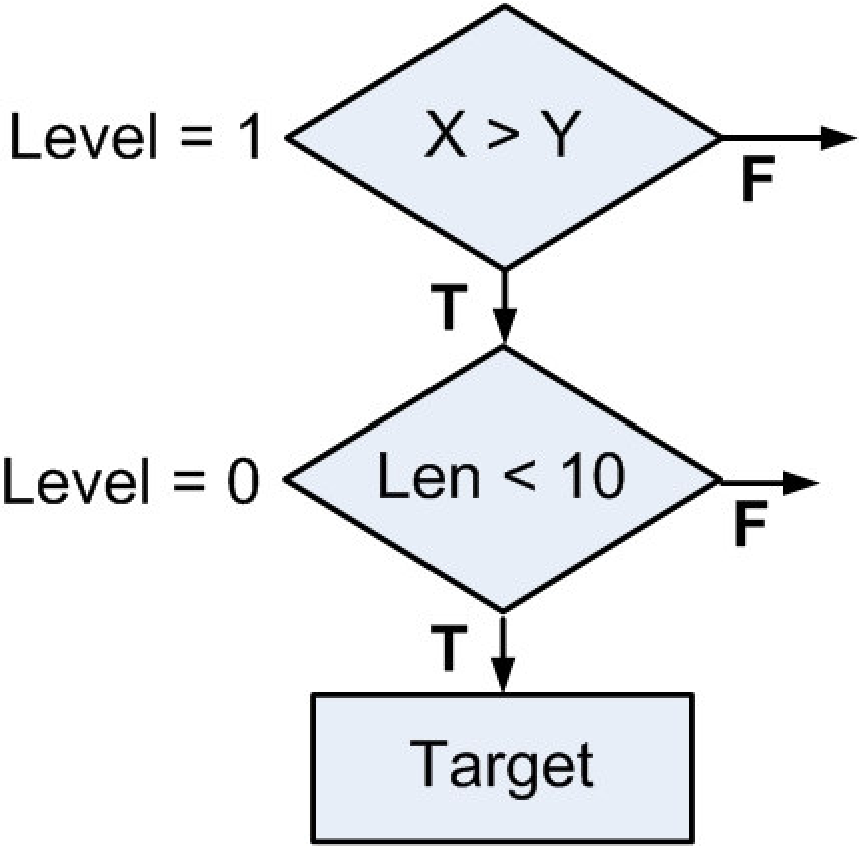
  gcd(int a, int b): a in [0, 15], b in [0, 15]



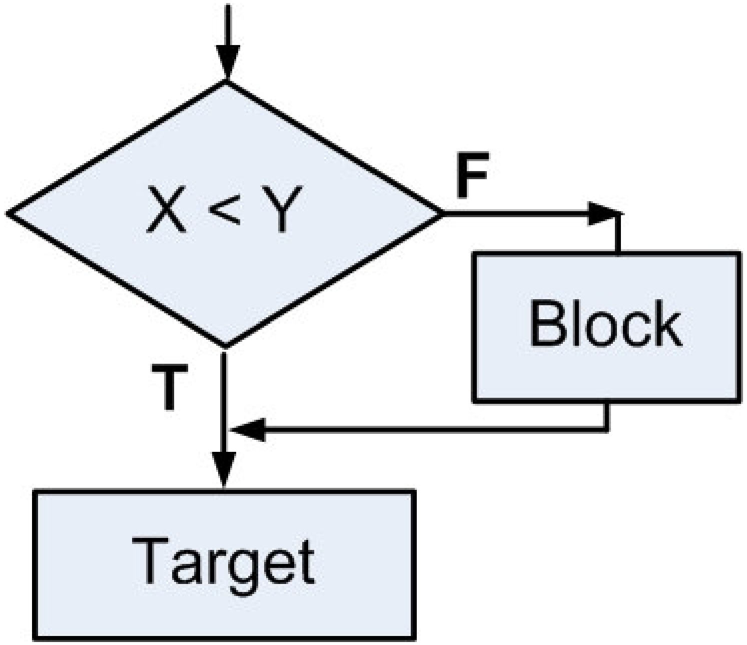
Fitness Function: Approximation level

[McMinn]

- Approximation level is number of critical branches minus one

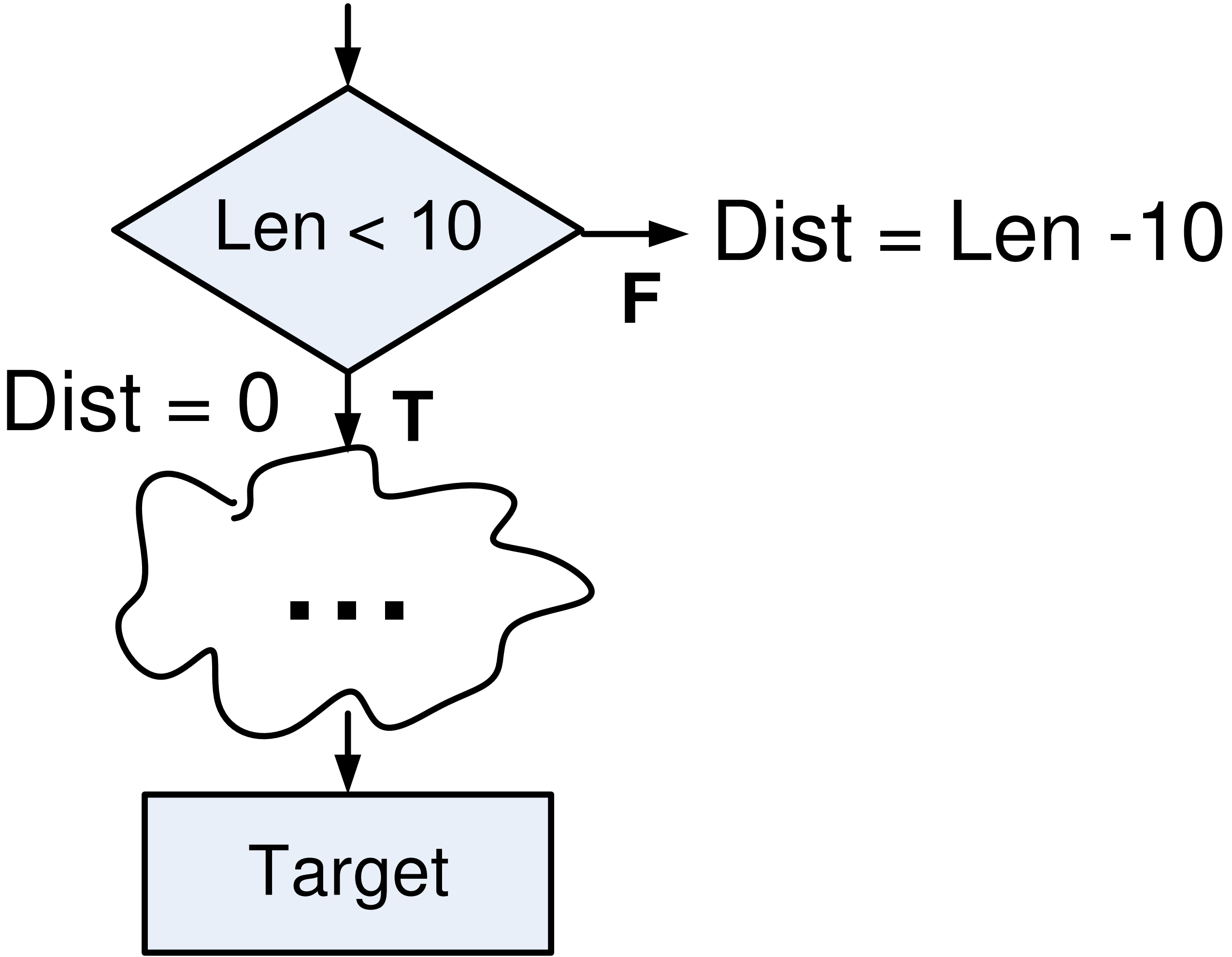


- Non-Critical Branch is not counted



Fitness Function: Branch Distance

- Branch Distance



Stella Levin and Amiram Yehudai, 2006

Fitness Function

- Fitness = Approximation Level

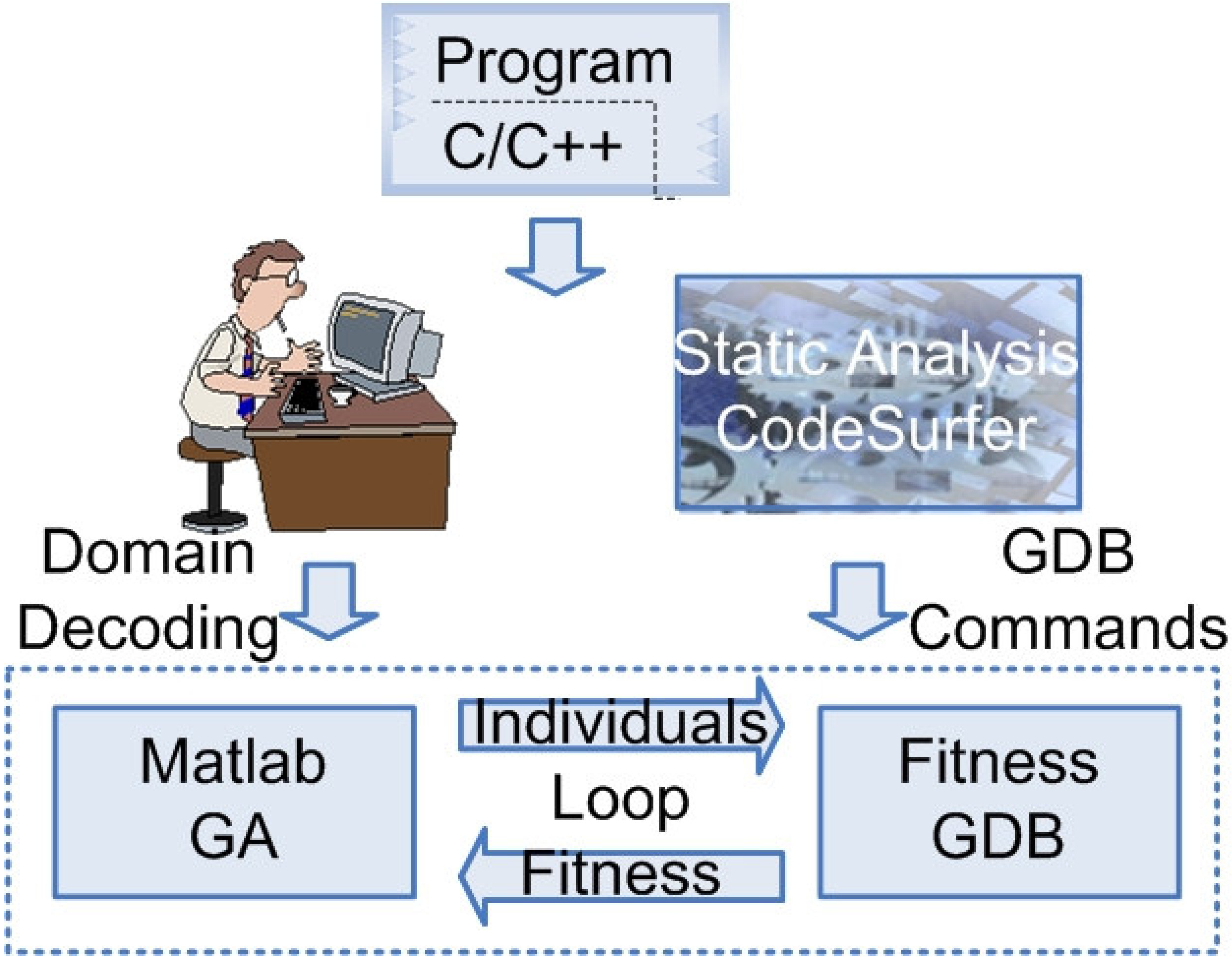
  + Branch Distance [0,1]

Expr	**Branch Distance**
a>b	b-a
a<b	a-b
a=b	abs(a-b)
a≠b	Constant
e1 or e2	min (dist(e1), dist(e2))
e1 and e2	max (dist(e1), dist(e2))

Static Analysis Algorithm

- From Target make DFS backward in control flow
- If get to decision point
  - If the point is new then save Approximation Level and outcome
  - Else update Approximation Level and outcome
- Print GDB file for the target
  - Break at decision point and calculate Approximation Level and Branch Distance

Testing System Description



Program

C/C++

Static Analysis

CodeSurfer

Domain Decoding

GDB Commands

Matlab GA

Individuals

Loop

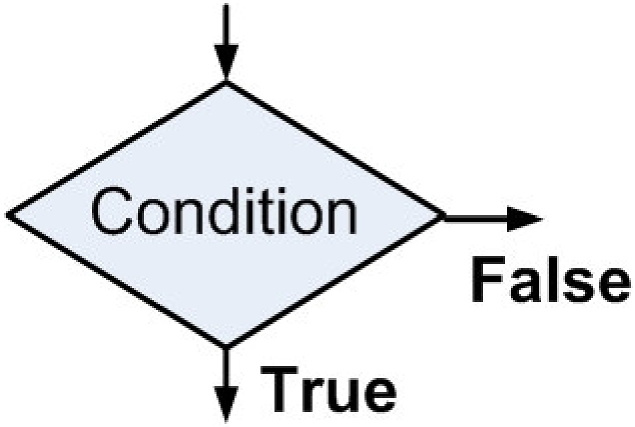
Fitness

Fitness GDB

Stella Levin and Amiram Yehudai, 2006

Our Experiment

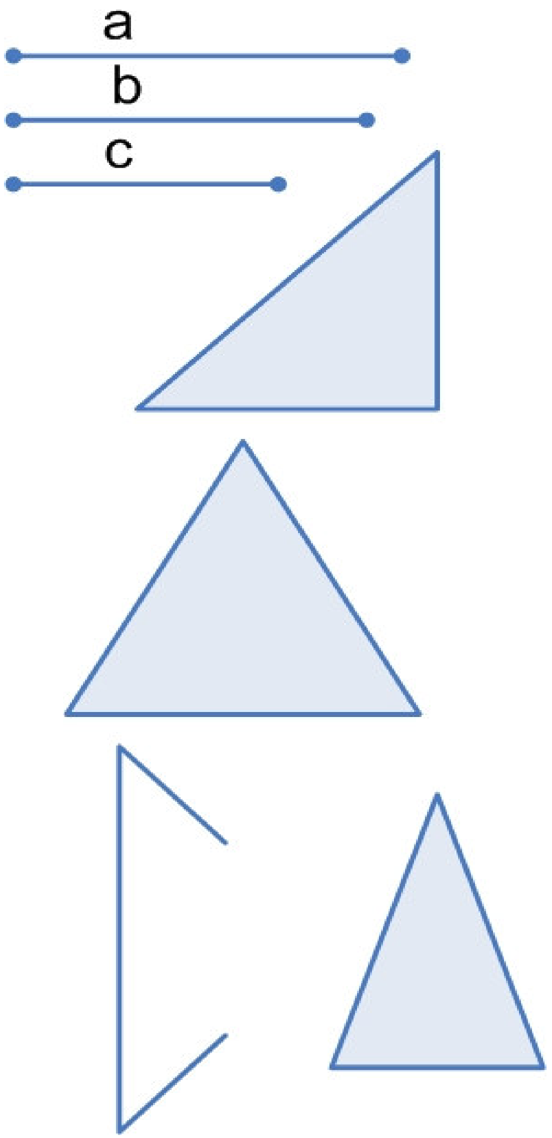
- Branch Coverage

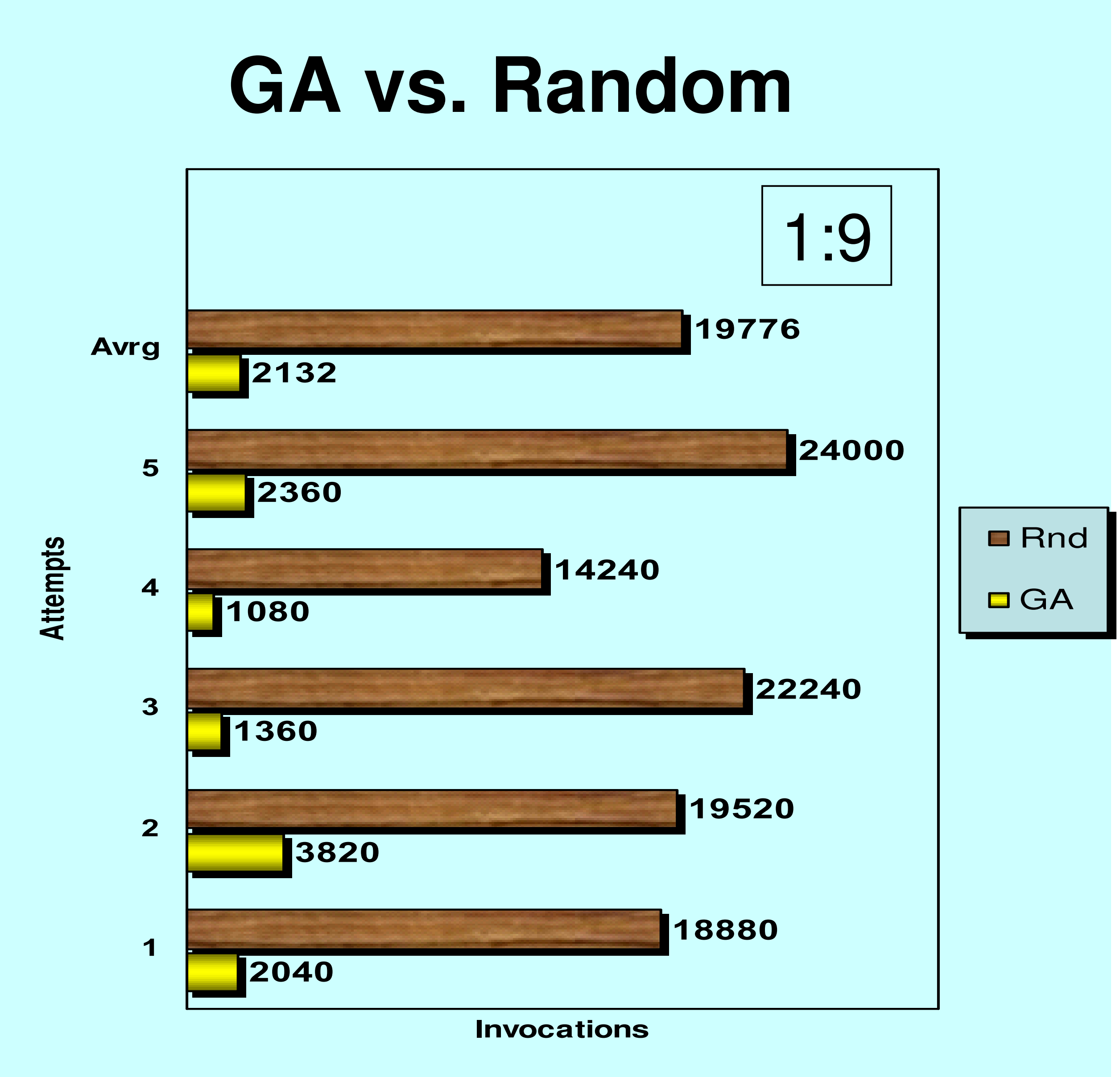


Exercise T/F outcomes of every decision

- Goal: to cover all branches with T/F
- Every Target - separate search
- GA Search
  - 20 individuals
- All attempts get 100% coverage

Triangle Classification





[Schatz et al]

Bubble Sort

GCD

Example:

```
> sort 5 3 4 7 1
  1 3 4 5 7
```

```
> gcd 27 18
  9
```

- Number of invocations is minimal: GA is 20; Random is equal to the number of targets.

[Schatz et al]

String Matching

Example:

```
> strmat abcdefgh def
  3
```

Encode:

8	01234567	3	456
len	string	len	pattern

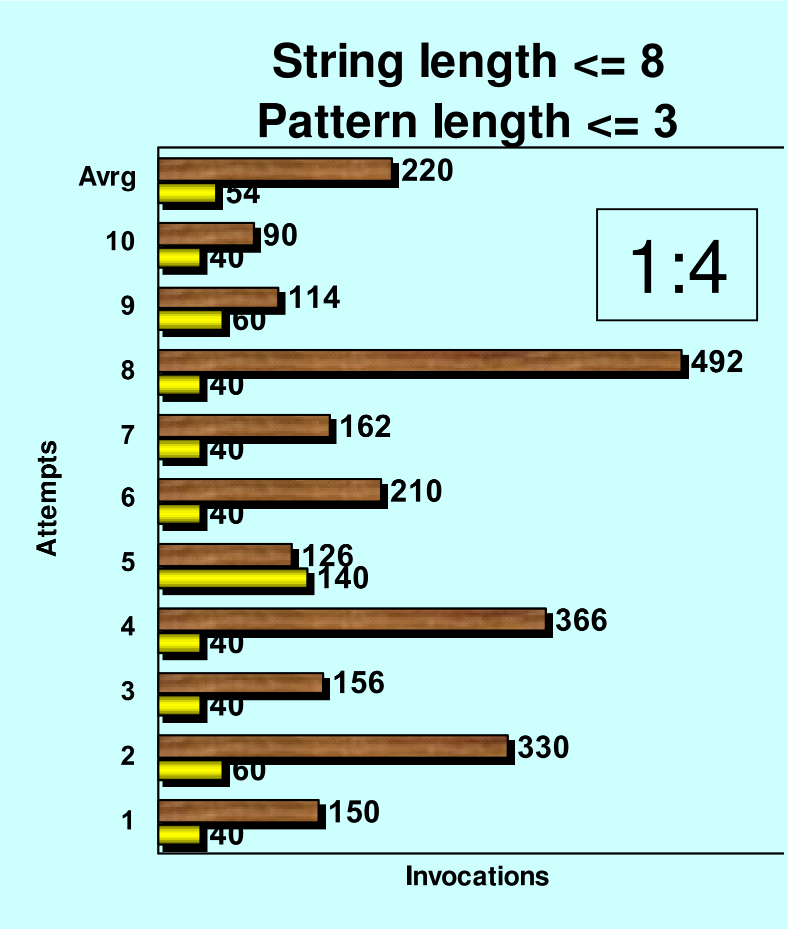
Encode(char) = ascii(char) – ascii(‘a’)

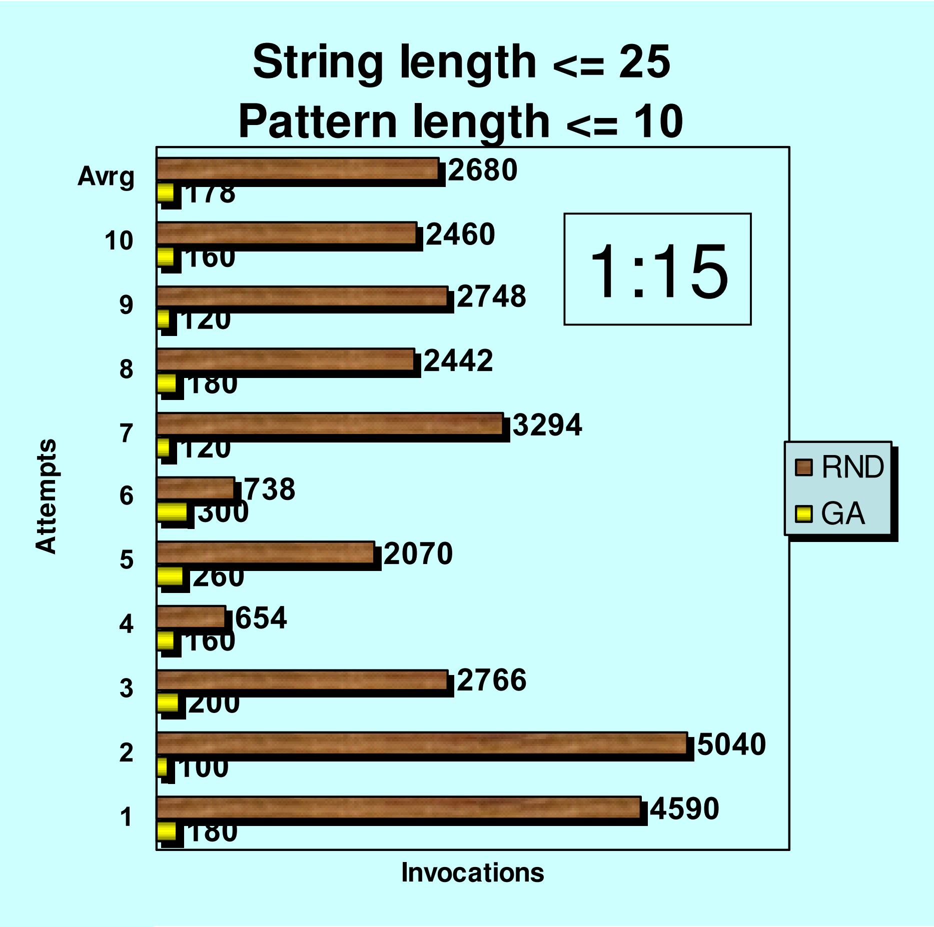
Encode(‘a’) = 0

Encode(‘b’) = 1

[Rad]

String Matching



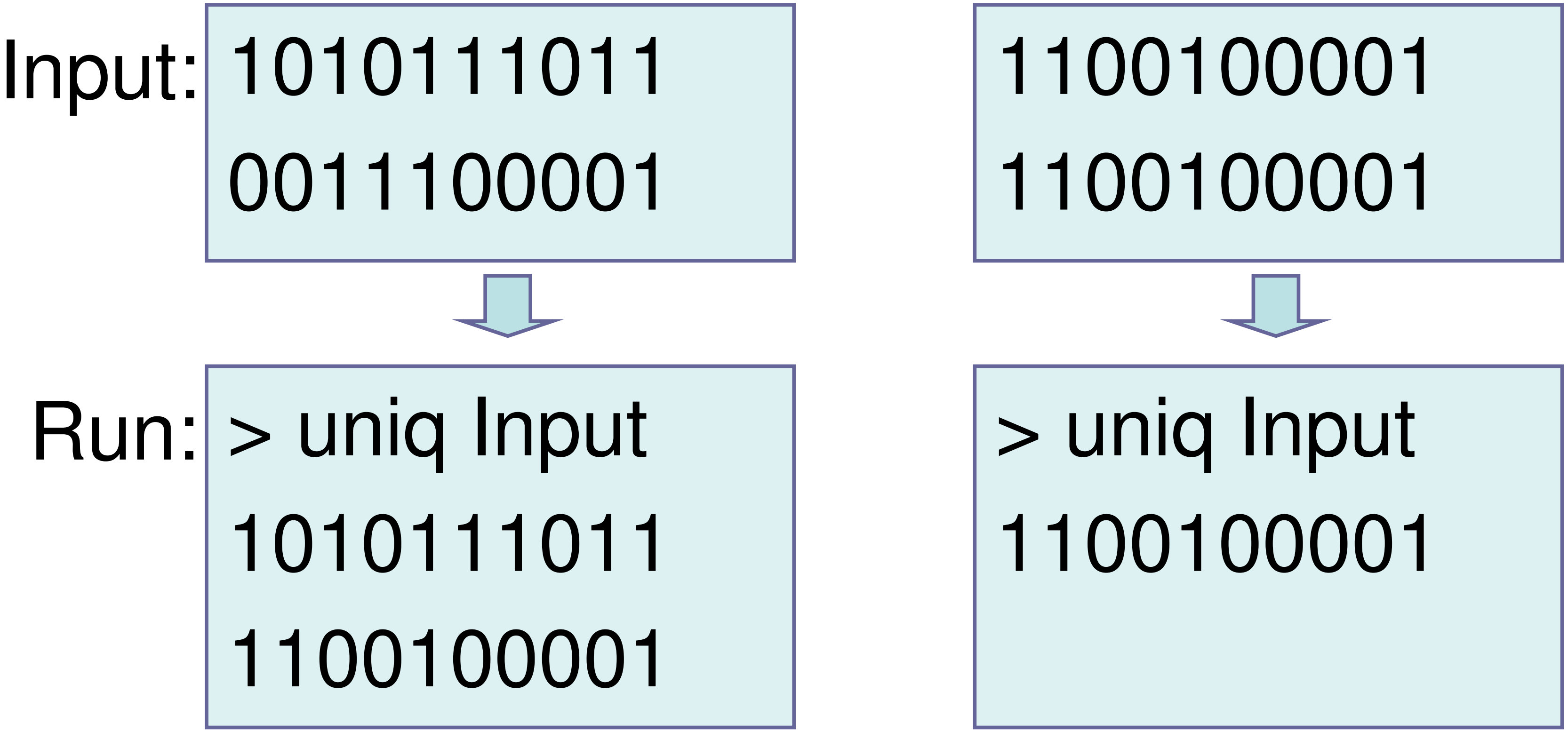


GA vs. Random

Stella Levin and Amiram Yehudai, 2006

Uniq UNIX Utility

- Uniq is a Unix utility which merges identical sequential lines in text file



Input:

```
1010111011
0011100001
```

```
1100100001
1100100001
```

Run:

```
> uniq Input
1010111011
1100100001
```

```
> uniq Input
1100100001
```

Uniq: Input Encode

Input: 10 bit line

1010111011
1100100001

Encode: 20 bit vector

10101110111100100001

line1 *line2*

Input: 20 bit line

Encode: 40 bit vector

Input: 10 alphabet

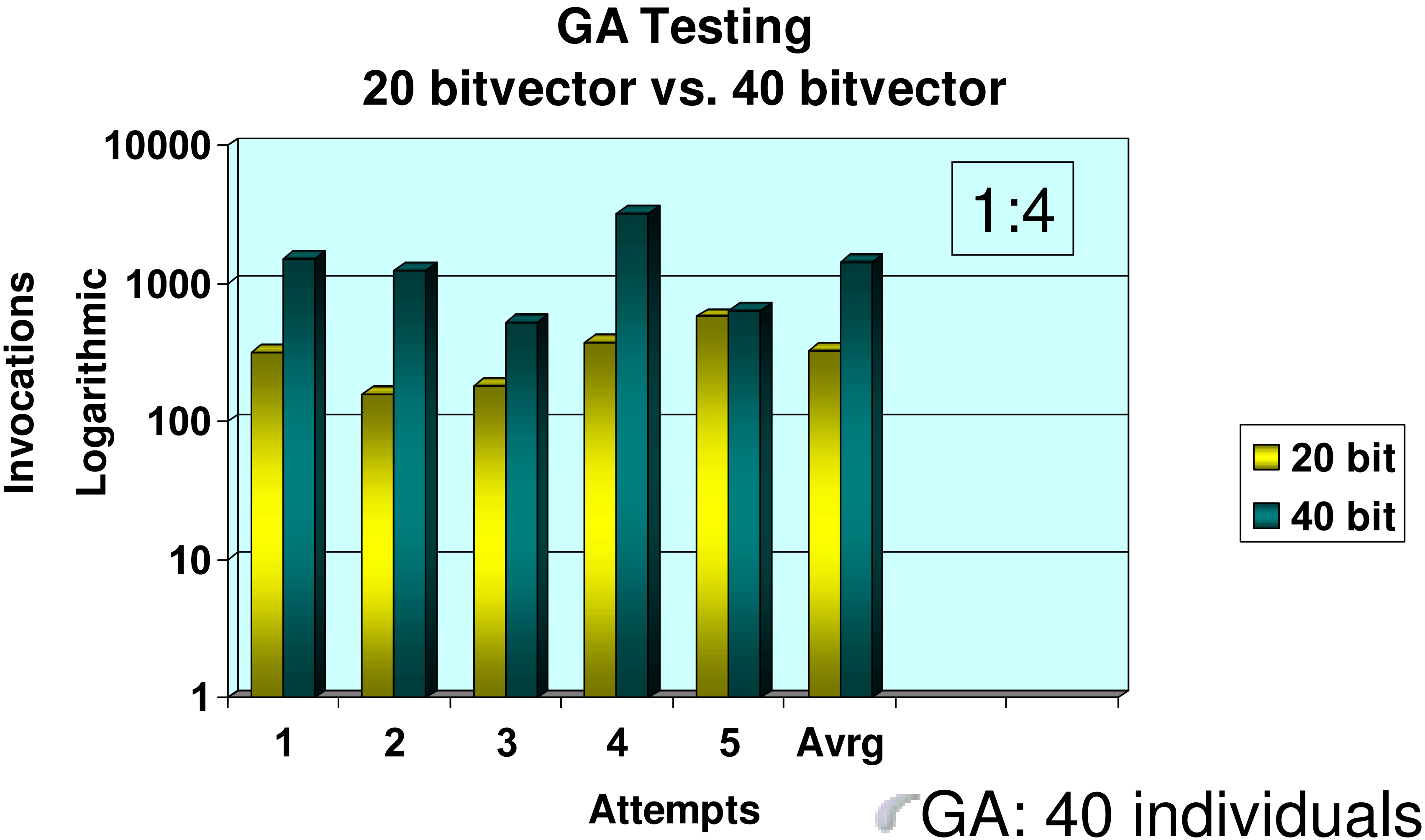
```
aabbccddee
ffgghhiijj
```

Encode: 20 int vector

00112233445566778899

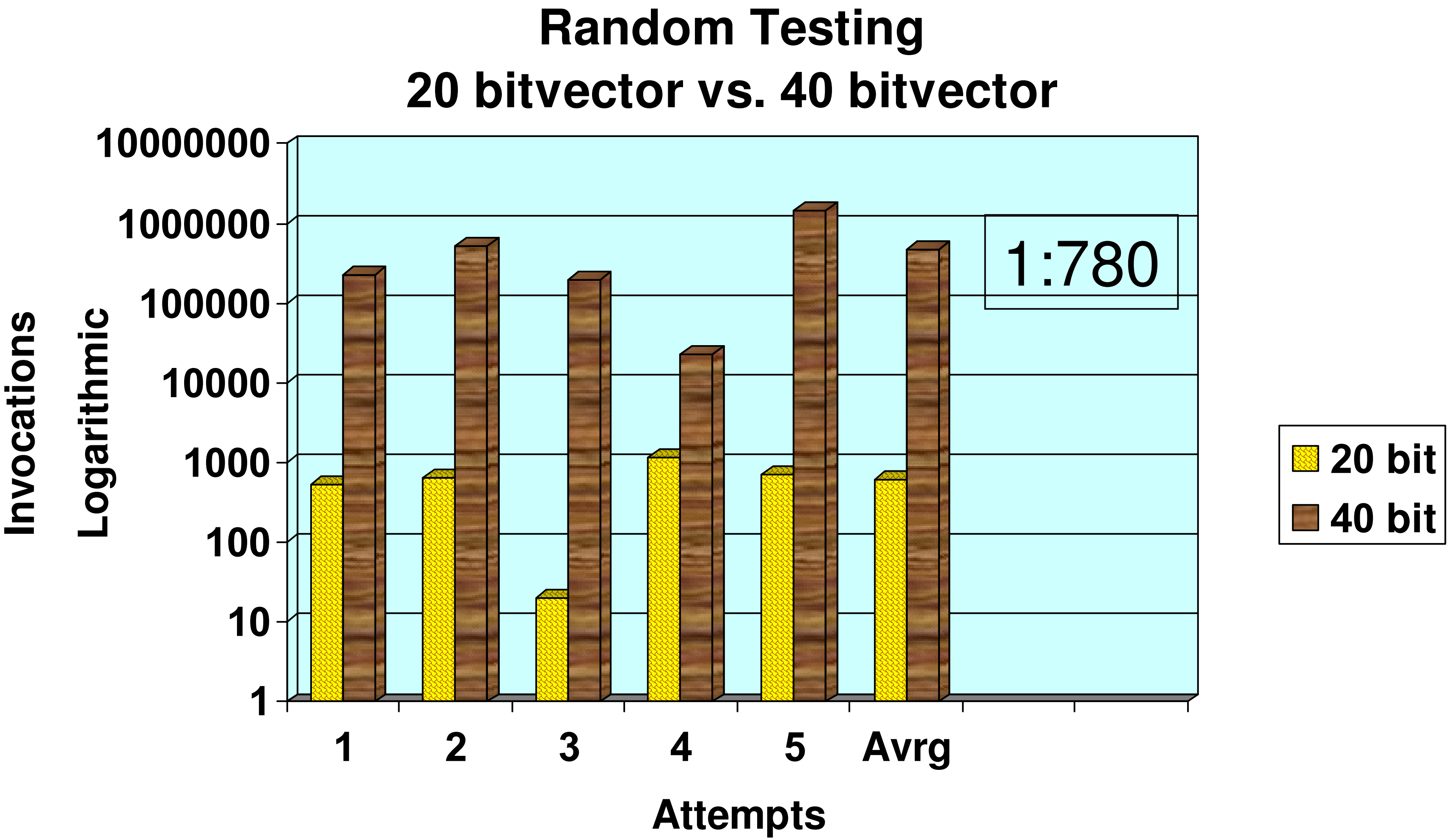
line1 *line2*

Uniq: GA Testing



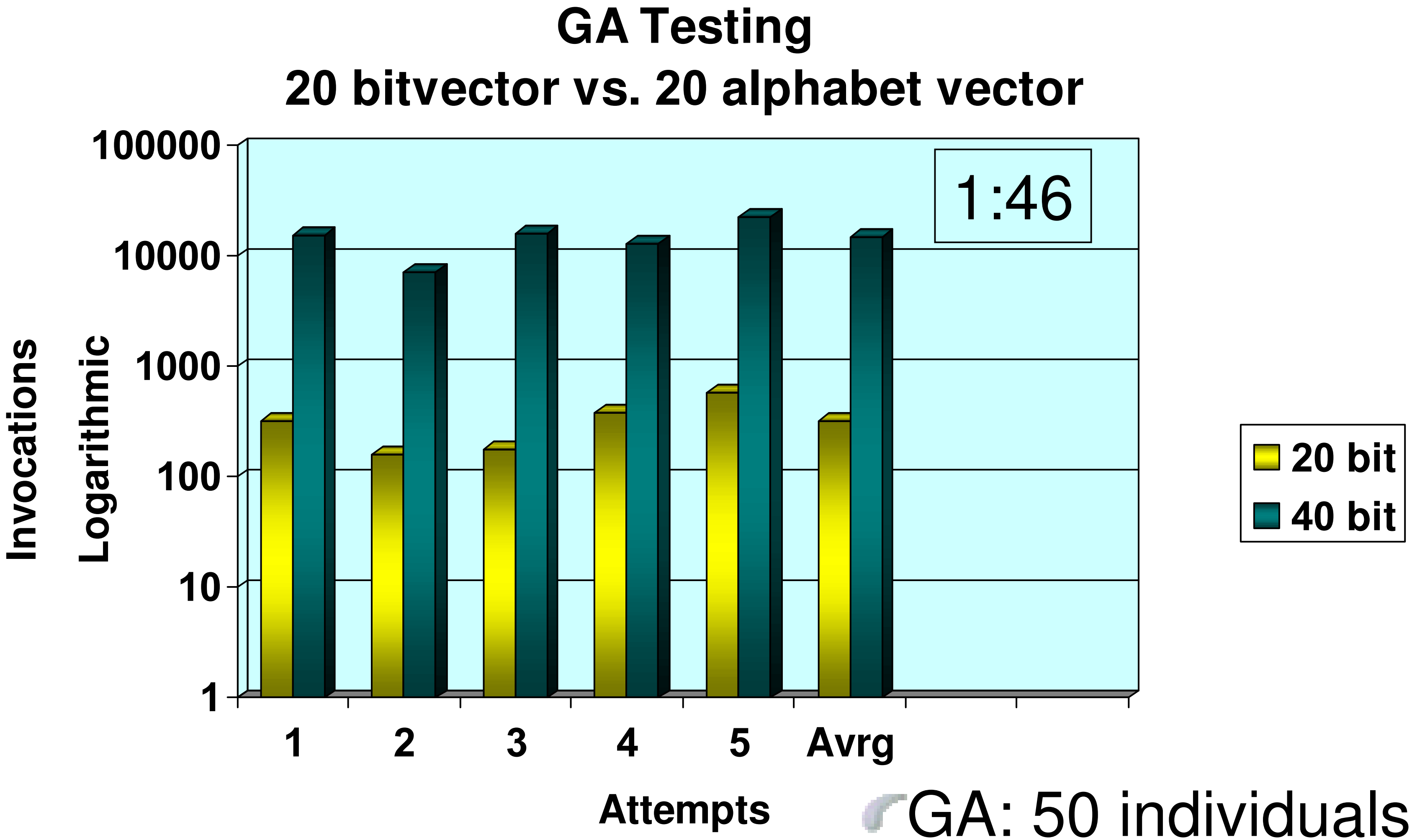
- GA: 40 individuals

Uniq: Random Testing



Stella Levin and Amiram Yehudai, 2006

Uniq: GA Testing



- GA: 50 individuals

Conclusions

- When GA Testing is better than Random?
  - For simple programs both testing systems work fine
  - As the complexity of the program or input domain grows, GA significantly outperforms Random testing system
- How much work is it to test a new program?

- Determine sub-domain of inputs

- Define encode/decode function

- Run the Testing System

Stella Levin and Amiram Yehudai, 2006

Future Plans

- Goal: better classify where the GA testing is superior
  - Examine more types of programs and analyze results
  - Examine larger programs

References

- P. McMinn. Search-based software testing: A survey. *Software Testing, Verification and Reliability*, 14(2): 105-156, 2004/6.
- André Baresel, David Binkley, Mark Harman, Bogdan Korel, Evolutionary Testing in the Presence of Loop-assigned Flags: A Testability Transformation Approach. ISSTA 2004: 108-118
- Goldberg “Genetic Algorithms”
- G. McGraw, C. Michael, and M. Schatz. Generating software test data by evolution. *IEEE Transactions on Software Engineering*, 27(12):1085-1110, 2001.
- David E. Goldberg: Genetic *Algorithms in Search, Optimization and Machine Learning* Kluwer Academic Publishers, Boston, MA, 1989.
- Soroush Karimi Rad. *Can structural test adequacy criteria be used to predict the quality of generated invariants?* MSc thesis, University of Antwerp, 2005