Testing Programs with Random Generated Test Cases

Kr. Manev, *Sofia University*
B. Yovcheva, M. Yankov, P. Petrov
Shumen University

Content

- Random generation of test cases
- Example
- Experiments
- Some results and comments
- Conclusion

Random test generation

- *Automated test data generation* (TDG) is inevitable – both for software industry and programing contests
- *Random TDG* is one of the earliest methods for ATDG which is still used
- In evaluation of programming contests random TDG has a special place, not typical for software testing in industry – checking the *time complexity* of algorithms implemented by students programs
- **RANDOM TDG IS NOT TRIVIAL!**

Random test generation

During an IOI (about 10 years ago) a task *(**formulation is consciously changed**)* on a graph $G(V,\mathrm{E})$ with a weight function c: $V \to N$ on the vertices was proposed. Some other function $f : V \to N$ has to be calculated and a vertex v to be found such that $f(v)=\max f(v_i)$, $v_i \in V$.

Being in shortage of time a contestants submitted trivial solution - a program that prints the vertex *w,* such that $c(w)=\max c(v_i)$, $v_i \in V$, and obtained enormously big amount of points (75). **We could guess that some of the test cases were randomly generated.**

Random test generation

Some years ago were popular **optimization tasks** (reduced versions of which are **NP-complete**), with “relative” evaluation. The programs that give a best result receives 100% of the points and the other programs receive amount of points proportional to the distance between their result and the best one.

We proposed such task in one of Bulgarian national contests. After the grading we decided to **change the random generated test cases with other random generated** and the results of the contestants changed dramatically.

Example

- **Task:** Given a **rooted** tree *T*(*V*,*E*) with *N* vertices, labeled from 1 to *N*. The root of *T* is the vertex labeled with 1. Find a vertex *v* such that the length of the path from 1 to *v* is maximal.
- **Simple Test Generator:**

```
int n, p[MAXN], i;
p[1] = 0; p[2] = 1; //1 is the root
for(i = 3; i <= n; i++)
    p[i] = rand() % (i - 1) + 1;
```

Example

- **Tests:** We decide to test the solutions with 100 randomly generated tests, with *N* = 10, 20, ..., 1000 respectively and assign 1 point for a successful test
- **“Stupid” solution:**

```
int n;
scanf(“%d”,&n);
… //reading other data not necessary
printf(“%d”,n);
```

Example

- **Test generation:** we generated 100000 test sets of 100 test cases as mentioned above. Average result of the stupid solution is 6 points, and the maximal – **16 points**!
- **How to proceed:**
  - Random permutation of the vertices after the generation of tests;
  - Each vertex to be generated as a possible answer as small as possible number of times;
  - Etc.

Experiments

- For the experiments the rooted trees was chosen because their simpleness and existence of characteristics that are easy to observe – height, width and branching
- For our experiments we first generated all RT(n) rooted trees of small number of vertices (n < 9) without excluding the isomorphic cases
- Distributions $(h_1,h_2,...,h_{n-1})$, $(w_1,w_2,...,w_{n-1})$ and $(b_1,b_2,...,b_{n-1})$ where h_i, w_i and b_i are the number of trees of n vertices with height, width and branching equal to i.

First Experiment

- For each n = 3, 4, 5, 6, 7 and 8 RT(n) rooted trees of n vertices were randomly generated (without excluding the isomorphic cases again)
- Corresponding distributions $(H_1,H_2,...,H_{n-1})$, $(W_1,W_2,...,W_{n-1})$ and $(B_1,B_2,...,B_{n-1})$ where H_i, W_i and B_i are the number of random generated trees of n vertices with height, width and branching equal to i.
- Expectations – real and random generated distribution will not coincide (statistically)

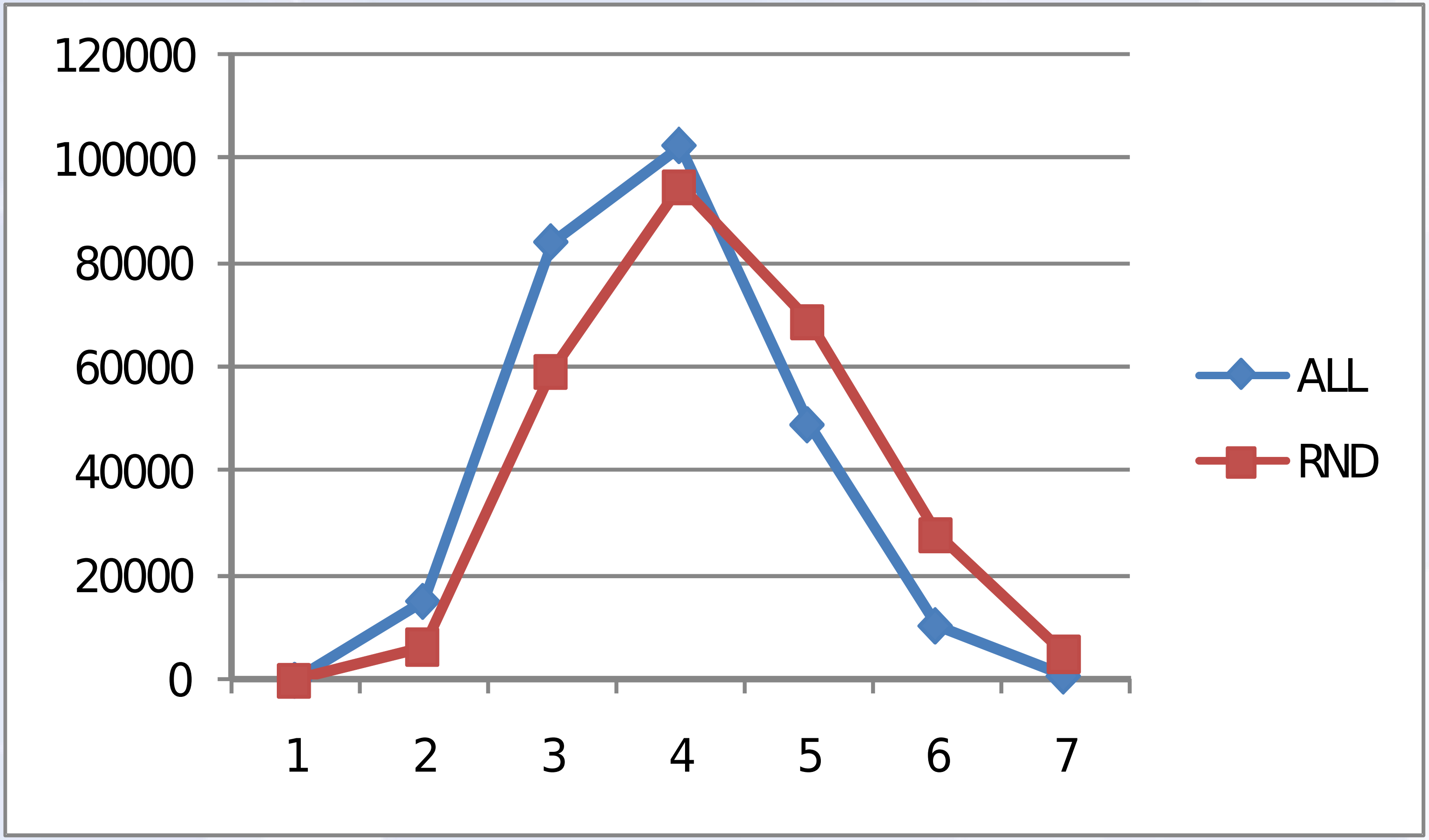
Second Experiment

- To check possibility with a small number of random generated RTs to obtain statistics close enough to the statistics of the full set.
- RT of 7 vertices was used. Random subsets of 20, 30, 40 and 50 such trees were generated.
- Relative distributions (in %) for each random set was calculated and compared with the corresponding, also relative, distribution for the full set.
- Expectations - small subsets of random generated rooted trees will not be able to “cover” statistically the full set.

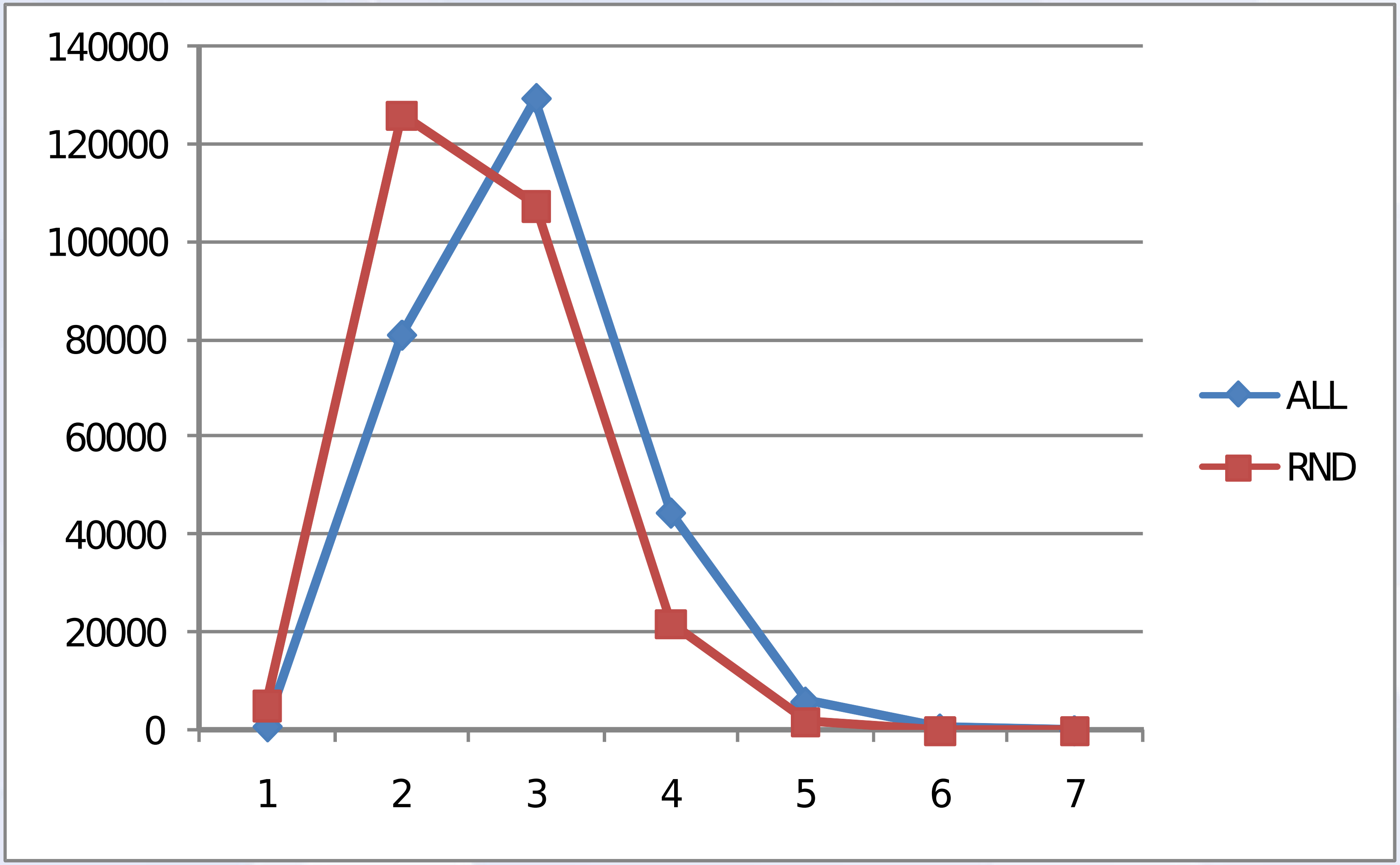
Exp.1. Some results (n=8)

k	With height *k*		With width *k*		With branching *k*	
	ALL	RND	ALL	RND	ALL	RND
1	19	1	857	5040	857	5040
2	15166	6321	80976	126000	111394	163170
3	84067	59472	129403	107520	114421	80850
4	102563	94710	44581	21875	30459	12005
5	49016	68880	5897	1659	4583	1029
6	10456	27720	411	49	411	49
7	857	5040	19	1	19	1

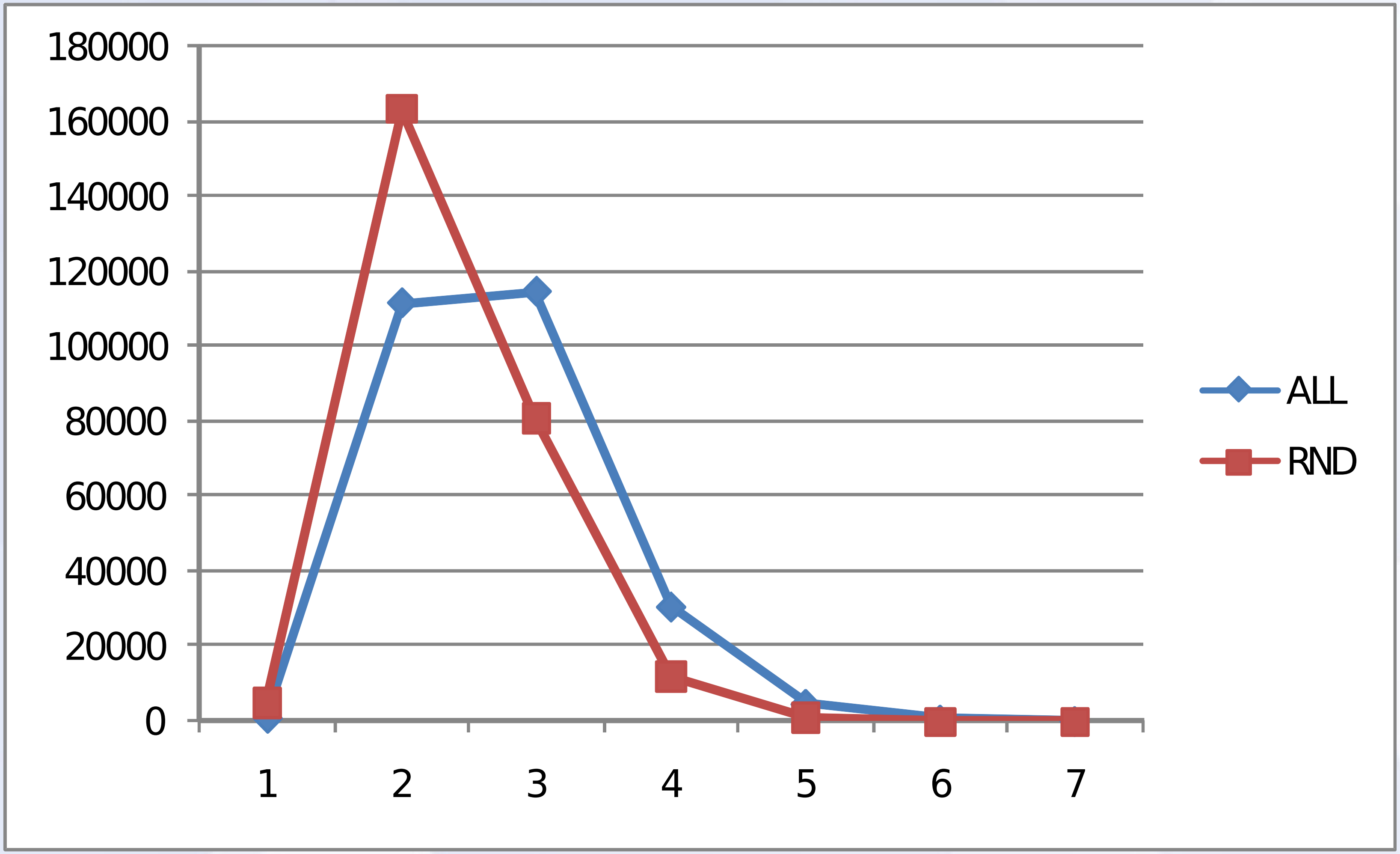
Exp.1. Height (n=8)



Exp.1. Width (n=8)



Exp.1. Branching (n=8)



Exp. 2. Height (n=7)

h	%RALL	%RND	%R20	%R30	%R40	%R50
1	0.01%	0.08%	0.00%	0.00%	0.00%	0.00%
2	6.28%	10.88%	10.00%	16.67%	12.50%	4.00%
3	33.02%	40.57%	30.00%	43.33%	30.00%	28.00%
4	37.13%	36.07%	45.00%	26.67%	50.00%	54.00%
5	19.28%	11.14%	45.00%	13.33%	7.50%	12.00%
6	4.28%	1.26%	0.00%	0.00%	0.00%	2.00%
	$X^2 = 0$	$X^2 = 12$	$X^2 = 43$	$X^2 = 29$	$X^2 = 22$	$X^2 = 13$

Exp.2. Width (n=7)

w	%ALL	%RND	%R20	% R30	%R40	%R50
1	4.28%	1.26%	0.00%	0.00%	0.00%	2.00%
2	57.83%	44.86%	45.00%	46.67%	57.50%	66.00%
3	33.20%	43.04%	50.00%	43.33%	30.00%	24.00%
4	4.46%	9.89%	5.00%	10.00%	12.50%	8.00%
5	0.21%	0.89%	0.00%	0.00%	0.00%	0.00%
6	0.01%	0.08%	0.00%	0.00%	0.00%	0.00%
	$X^2 = 0$	$X^2 = 18$	$X^2 = 16$	$X^2 = 17$	$X^2 = 19$	$X^2 = 8$

Conclusion

- Random test data generation is useful for evaluation of the quality of programs but has to be applied carefully with corresponding theoretical research of the involved objects.
- *The same experiments are impossible for large trees*. For the purpose some combinatorial and statistical results for the distribution of the observed characteristics have to be attracted.
- Perspectives ?

Any questions?

Thanks!