The difficulty of programming contests increases

Michal Forišek⋆

Comenius University, Bratislava, Slovakia
`forisek@dcs.fmph.uniba.sk`

Abstract. In this paper we give a detailed quantitative and qualitative analysis of the difficulty of programming contests in past years. We analyze task topics in past competition tasks, and also analyze an entire problem set in terms of required algorithm efficiency. We provide both subjective and objective data on how contestants are getting better over the years and how the tasks are getting harder. We use an exact, formal method based on Item Response Theory to analyze past contest results.

1 Introduction

It is a well-known consensus in the community around programming contests that the difficulty of these contests progressively increases. For example, Verhoeff et al. mention this observation in [23] as a part of the motivation to have a Syllabus for the International Olympiad in Informatics (IOI).

In this article, we try to give exact, objective arguments that this is indeed happening. We analyze the "internals" of competition tasks (such as covered topics and required algorithm efficiency). Also, we analyze the raw competition data (such as detailed results, submission logs, etc.) that can be used to show whether tasks are getting harder.

We will now give an overview of prior research in this area. The age at which children start practicing for programming contests is decreasing. Kelevedjiev and Dzhenkova [10] mention that in many countries the age at which children start with programming and programming contests is already as low as ages 11 to 12. Thanks to this early start the contestants are able to cover more areas of computer science during their preparations. In accord, the set of topics used in contest tasks is growing. For example, Manev [14] notes that tasks on graphs became common in all levels of contests after such tasks were used at the IOI. Each year there are proposals of new task types that push the boundary of the scope of programming contests still further – for recent examples see [3, 21, 16].

But even if we restrict ourselves to a fixed set of topics, there will still be both easier and harder tasks around, and the harder ones are more and more common. Kiryukhin in [12] describes the development of the Russian Olympiad in Informatics. One important point mentioned by Kiryukhin is that only after the introduction of modern, efficient computers around the year 1995 the organizers were able to use test inputs large enough to evaluate algorithm efficiency

⋆ This work was supported by the grant VEGA 1/0726/09

with sufficient precision. Hence the best contestants became motivated to learn, design and implement better, more efficient algorithms.

Note that from Kiryukhin's observation it follows that only for the last approximately 15 years algorithm efficiency plays a significant role in programming contests. In other words, since 1995 the focus in programming contests started to shift from "implement a correct algorithm" towards the much harder "implement a correct algorithm that is as efficient as possible".

In [18, Section 7] Revilla et al. discuss the issue of task difficulty. They were not able to find a satisfying way to determine the difficulty of tasks in the University of Valladolid (UVa) Online Judge [17]. They state that difficulty is subjective, and while most people agree on the ends of the spectrum, the difficulty levels of intermediate tasks are almost impossible to establish. For the book [19] the difficulty of the selected tasks was estimated manually.

For past IOI tasks there are several publications related to the scope of this article: Kiryukhin and Okulov [13] (in Russian) manually analyze and classify the past tasks, Verhoeff [22] provides a different clarification and also ranks the difficulty of tasks based on the percentage of contestants who managed to "fully solve" the task (i.e., score at least 90% of possible points). As claimed by Verhoeff, this metric only describes how hard the task was for the set of contestants who were solving it – but it is not sufficient to compare the difficulty of tasks from different years.

Kemkes et al. [11] introduce Item Response Theory (IRT) as a tool that can be used to evaluate the difficulty of competition tasks, and they use it to analyze scoring of tasks from past IOIs. Their methods were further developed by the author of this article in [5, 6].

1.1 Goals of the article

In this article we present detailed evidence (both quantitative and qualitative) for the following claims:

1. The set of topics covered by task statements and solutions is growing.
2. The topics previously considered difficult now start to appear in contests designed to be "easier" (such as categories for younger students).
3. **The difficulty of tasks in programming contests is increasing.**
4. The skill level of both top and average contestants is increasing.

We will mostly focus on Claim 3. However, it is important to realize that claims 3 and 4 are closely related. In Section 2 we illustrate this on an example.

1.2 Overview of the article

In Section 2 we show that raw scores, more precisely IOI medal boundaries, do not carry sufficient information to argue about task difficulty.

In Section 3 we address the first two goals of our article, and additionally we show that in tasks based on the same computational problem there is a trend towards requiring more efficient algorithms.

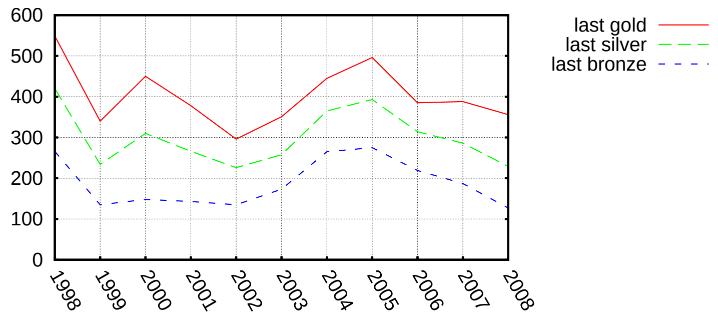


Fig. 1: IOI medal boundaries in years 1998 to 2008.

In Section 4 we present the results of our international survey that shows a strong correlation between the year in which a task was set and its perceived difficulty.

In Section 5 we address the fourth goal of this article, showing data that the skill level of both the very best and the average contestants is gradually increasing. The main claim of this article logically follows from the data presented in the previous three sections.

Finally, in Section 6 we give a short note on how Item Response Theory can be used to obtain data that will give us even more information on task difficulty.

2 IOI medal boundaries

A naïve attempt to prove that the difficulty of programming contests increases would simply lead us to examine the scores – with the hypothesis that we should see a steady decrease.

In Figure 1 we show the medal boundaries at the IOI in the years where automated grading was used.[1] For clarification: the three lines denote the score achieved by the last contestant that was awarded a gold, a silver, and a bronze medal respectively. (This corresponds to the top 8.3%, top 25%, and top 50%.)

Clearly, there is no visible decrease in the scores, more precisely, no significant negative correlation between the year and the medal boundaries. Quite on the contrary, the scores tend to be pretty balanced throughout the years, with seemingly easier and harder years alternating. As we already mentioned (and as we show in later Sections), the actual reason is that two trends occur at the same time – not only are the tasks getting harder and harder, but also the contestants are getting better and better.

The conclusion we can draw from this simple example is that if we want to argue about task difficulty, we can not do it without addressing the contestants'

[1] In all years except for 1998 and 2000 the theoretical maximum was 600 points. In 1998 the maximum was 700, the scores are scaled. In 2000 each contestant was given 100 free points, hence those 100 points were subtracted from the scores.

skills at the same time. Or, from a statistical point of view, we will not be able to draw any conclusions from contest results only, unless we assign contestant names to the scores or add some other additional data source.

3 Task topics and algorithm complexity

We now show several related trends: In Section 3.1 we show that difficult concepts such as dynamic programming become more and more common in solutions, and such concepts now occur even in tasks supposed to be "easy". In Section 3.2 we show that tasks built upon the same computational problem now usually require more efficient algorithms than in the past. Finally, in Section 3.3 we show that each year there are new tasks requiring new, more complex algorithms.

3.1 Task topics in TopCoder contests

TopCoder Inc. organizes programming contests since 2001. Since 2003, there are regular single-round competitions, called Single Round Matches (SRMs) that all use the same format: The contestants are separated into two (approximately equally large) divisions according to their current *rating.*[2] The stronger division is called Division 1 (Div1), the weaker one is Division 2 (Div2). In each Division, the contestants are given three tasks to solve. The tasks are labeled "easy", "medium" and "hard" according to their expected difficulty. Tasks in Div1 are different (and harder) than tasks in Div2.

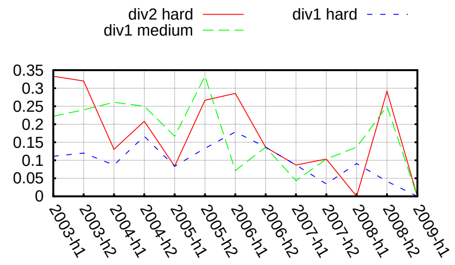
In the task archive each task has a set of labels assigned by its author. The labels list the general methods used to solve the task, such as "dynamic programming", "geometry", or "graph theory". These labels and also detailed results of all past matches can be obtained from the official data feeds [20].

We focused on four of the more interesting labels. For each of these, we plotted a graph that shows, for each difficulty level in each division, the percentage of tasks that had this label in each half-year since 2003. These plots are shown in Figure 2. The main points that can be observed by reading these plots:
– The number of "easy" tasks based on brute force and simulation decreases.
– The number of tasks that require difficult concepts increases, and such tasks are becoming more and more frequent in easier difficulty levels.

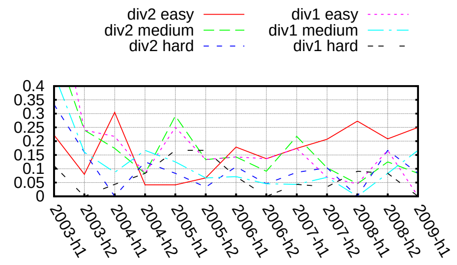
In Subfigure 2a we see the steady decline of hard tasks that can be solved by brute force. This is most clearly visible in Div1 hard tasks (the hardest task in each round), where the percentage of tasks solvable by brute force decreased steadily since 2006, and actually reached zero in 2009.

A similar trend is visible in Subfigure 2b. In 2005 simulation tasks were still pretty common even as Div1 hard tasks, but since 2006 the number of such tasks never exceeded 10%, and it was zero in 2009. Almost the same is true for

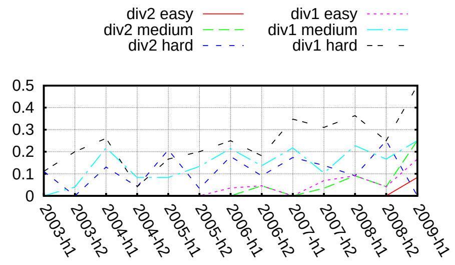
[2] The rating is a TopCoder-calculated numeric estimate of the contestant's skill. Ratings currently lie in the range $[0, 4000)$, higher is better.



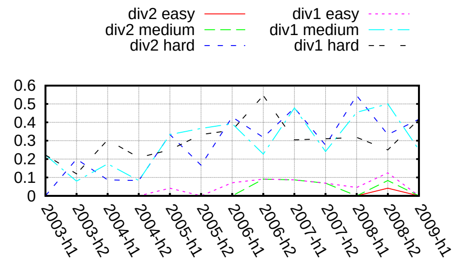
(a) Brute force.



(b) Simulation.



(c) Graph Theory.



(d) Dynamic programming.

Fig. 2: Percentages of topic-related tasks in TopCoder's SRMs.

Div1 medium tasks. On the other hand, more and more of the easiest tasks are simulation-based.

Subfigure 2c shows the percentage of graph theoretical tasks. Even in the second half of 2004 almost no Div1 hard tasks contained graph theory. Ever since, this number is steadily growing, and in 2009 it reaches 50%. Also observe that several years ago graph theoretical tasks were used as the more difficult problems only. Until 2005 such tasks were only used as the harder two tasks in Div1, and as the hardest task in Div2. In 2006 the first such tasks appeared as Div1 easy and Div2 medium tasks, and the percentage of such tasks is growing steadily ever since. In 2009 the first task on graphs was used as Div2 easy.

Finally, in Subfigure 2d we can observe similar trends for tasks that require dynamic programming. The popularity of such tasks was growing between 2003 and 2006, and is more or less constant since. Again, we can observe that since 2005 dynamic programming is finding its way into the easier task levels. In 2008 a dynamic programming task was used as Div2 easy for the first time.

3.2 Efficient algorithms in UVa contests

For technical reasons, the number of elements in data structures used in TopCoder tasks is limited to 50. E.g., if the input is an adjacency matrix of a graph, the graph can only have at most 50 vertices. This does, to some extent, make it impossible for problem setters to enforce the use of the most efficient algorithms for a given task. E.g., the most efficient algorithms for single-source shortest paths in a graph have time complexity roughly proportional to $N \log N$, but for $N = 50$ even an $O(N^3)$ algorithm will do fine. However, other contests lack this limitation, and in such contests we can also observe that as years progress, it is more and more common to use test data sizes that require the contestants to implement more efficient algorithms.

We analyzed of tasks in the "Contest" section of the UVa Online Judge [17] archive. We focused on several well-known computational problems, for each of them we found tasks based on it, and examined their input sizes. When analyzing the tasks, two helpful resources were the manual classifications of UVa Online Judge tasks done independently by Greve [7] and by Navernioulk [15].

The results are shown in Table 1. In all four cases, the evolution of maximum instance sizes clearly shows the progress towards requiring asymptotically more efficient algorithms – with the most recent versions requiring (almost) optimal ones known.

id	**name**	**date**	**N**	**M**	**comment**
10269	Adventure of Super Mario	2002-04-20	1 000		requires preprocessing
10342	Always Late	2002-07-27	200		2nd shortest walk
10603	Fill	2004-01-10	80 000	240 000	solvable via BFS as well
10740	Not the Best	2004-10-16	1 000	10 000	k shortest walks
10917	Walk Through the Forest	2005-09-24	1 000		number of shortest paths
10986	Sending email	2006-01-21	20 000	50 000	
11367	Full Tank?	2007-12-01	100 000	1 000 000	state: city and fuel amount
11635	Hotel booking	2009-07-18	10 000	100 000	additional complications

(a) Single-source shortest paths.

id	**name**	**date**	**N**	**comment**
10065	Useless Tile Packers	2001-01-19	100	
10173	Smallest Bounding Rectangle	2001-09-01	1 000	$O(N^2)$ postprocessing needed
10652	Board Wrapping	2004-05-22	2 400	
11072	Points	2006-08-12	100 000	followed by point-in-polygon queries
11096	Nails	2006-09-21	100	
11168	Airport	2007-02-17	10 000	non-trivial postprocessing
11626	Convex Hull	2009-06-13	100 000	

(b) Convex hull.

id	**name**	**date**	**N**	**comment**
10131	Is Bigger Smarter?	2001-06-29	1 000	
10534	Wavio Sequence	2003-07-26	10 000	two computations needed
10635	Prince and Princess	2004-03-13	62 500	non-trivial preprocessing needed

(c) Longest increasing subsequence.

id	**name**	**date**	**N**	**M**	**comment**
10319	Manhattan	2002-06-29	120	400	solving 2-SAT
10731	Test	2004-09-25	26		
11324	The Largest Clique	2007-10-27	1 000	50 000	
11504	Dominos	2008-09-27	100 000	100 000	

(d) Strongly connected components.

Table 1: Maximum instance sizes in UVa contest tasks.

In all tables, N is the main instance size: the number of vertices/sequence elements/points. Wherever the input is a graph, M is the limit on its number of edges. If omitted, $M = O(N^2)$.

3.3 New task types by year

In this Section we attempt to show how the set of topics used in contest tasks is growing in time. In order to do so, we created a list of topics that, according to our opinion, form a representative sample of different algorithms and areas of Computer Science. For each of the topics we attempted to find the earliest contest and task that involved this topic. We then arranged the topics into a list ordered by the year of the first appearance.

Obviously, it is not humanly possible to take all international contests into account. The list we present is based on the following contests:

- International Olympiad in Informatics [8] 1989–2009
- The set of ACM ICPC contests at UVa Online Judge [17] 2000–2009
- Internet Problem Solving Contest [9] 1999–2009
- TopCoder competitions [20] 2000–2009

One important set of contests that is missing in the list is the set of World Finals of the ACM ICPC [1] – so far we did not have sufficient resources to analyse these tasks. (Also, this analysis is only possible to some extent, due to the fact that the official test data used for the contest is destroyed after the contest.)

1991: grammars and rewriting systems (IOI: S-terms)
1995: theoretical task; process communication (IOI: Printing),
interactive task (IOI: Wires and Switches)
1996: a two-player interactive game (IOI: A game),
optimal job scheduling (IOI: Job processing)
1997: optimization tasks and approximation algorithms (entire IOI)
1999: open-data task format (entire IPSC 1999),
code analysis and white-box testing (IPSC: Coins)
2000: game played in multiple submissions (IPSC: Trolls),
querying an unknown sequence (IOI: Median)
2001: Fenwick trees (IOI: Mobiles),
meet-in-the-middle search (IOI: Double crypt),
max. weight bipartite matching (BUET/UVA Oriental C.1: Bob Laptop),
a-star heuristic search (2001 Regionals Warmup Contest: 15-Puzzle problem)
2002: reducing a 2d task to independent 1d tasks (IOI: Utopia),
dynamic programming exponential in some dimension (TC: PinballLanes),
general matching, Chinese postman problem (U. of Waterloo June Contest)
2003: programming non-traditional computation models (IPSC: begin 4 7 add)
2004: reducing runtime from $O(N!)$ to $O(2^N)$ via DP (TopCoder: KiloManX)
2006: pen and paper cryptoanalysis (IPSC: Encoded messages),
modern cryptoanalysis (IPSC: h4x0r t3h c0d3),
page faults in caching (IPSC: Librarian)
2007: minimum cost maximum flow (TC: RadarGuns)
2008: low-level data representation (IPSC: Comparison mysteries)
2009: regular expressions processing (IPSC: Muzidabutur)

4 Subjective task difficulty rating

For the purpose of obtaining useful data on how contestants perceive task difficulty, we carefully selected a set of 12 tasks using the following methodology: All the tasks come from different years of the ACM ICPC [1] Northwestern European regional contest (NWERC). More precisely, for each year between 1997 and 2008 inclusive we examined the results of the contest and picked the median task[3] according to the number of teams that solved it.[4] The rationale behind this choice was that the median task should reasonably well represent the difficulty of the problem set.

The statements of those 12 tasks (without any indication of their origin) were presented as a survey to active contestants. The goal in the survey was to estimate the relative difficulty of those 12 tasks. Details on the survey formulation are given in Appendix A.

We got 33 answers to the survey. The respondents that filled in the survey were from over 20 different countries on 5 different continents, and all of them are successful participants in international programming contests.

The results of the survey are presented in Table 2. Tasks are labeled 0 to 11 according to the year in which they were used (0 is oldest). The left half of Table 2 contains the raw answers: the number in row X and column Y is the number of respondents who think task X is harder than task Y. Shaded cells are those for which the opposite opinion received strictly less votes.

The right half contains the same data, but we only count the 26 non-anonymous respondents for which we know their TopCoder ratings (see Section 3.1 for explanation of ratings). We used the ratings as weights of their votes. In this way we gave a higher significance to votes by better contestants. The totals are in thousands, rounded to the closest one.

	0	1	2	3	4	5	6	7	8	9	10	11
internet/0	■	0	0	1	2	0	0	1	0	2	0	0
space/1	27	■	20	11	18	8	8	5	3	12	5	5
papergirl/2	23	5	■	8	11	6	6	5	3	8	4	6
railroads/3	21	14	16	■	19	12	12	9	6	8	6	8
dates/4	24	6	15	6	■	8	7	6	4	9	5	5
floors/5	24	15	15	12	17	■	10	7	7	13	6	8
boss/6	23	14	15	10	13	12	■	4	5	9	6	8
taxicab/7	22	17	15	15	18	14	15	■	15	13	14	13
tantrix/8	24	20	21	18	22	15	16	8	■	19	13	11
setstack/9	21	10	13	13	18	10	11	7	5	■	10	9
escape/10	26	20	17	16	20	15	15	7	11	13	■	14
mobile/11	22	15	16	13	18	12	12	7	10	12	7	■

	0	1	2	3	4	5	6	7	8	9	10	11
internet/0	■	0	0	1	3	0	0	1	0	3	0	0
space/1	46	■	40	16	32	12	12	8	2	18	9	9
papergirl/2	41	5	■	10	17	7	8	6	4	8	3	9
railroads/3	40	31	37	■	35	28	27	15	10	17	10	16
dates/4	41	12	28	10	■	14	14	10	6	11	8	9
floors/5	42	28	32	17	30	■	14	10	10	17	9	14
boss/6	42	27	30	16	23	25	■	5	9	14	8	16
taxicab/7	36	34	33	28	32	29	30	■	24	21	26	24
tantrix/8	40	39	41	32	36	30	31	14	■	31	22	22
setstack/9	39	21	29	25	33	23	21	11	7	■	20	18
escape/10	43	38	35	29	35	31	30	12	18	19	■	25
mobile/11	39	27	32	23	32	24	22	12	16	20	14	■

Table 2: Survey of NWERC task difficulties. Left part: raw, right part: weighted.

[3] Ties were broken by the total number of submissions (or equivalently, number of incorrect submissions, less means easier), and if the problem set contained $2N$ tasks then the N-th easiest task was selected.

[4] Even more precisely, for the years 1997 to 1999 only summary ranklists were available. For these contests we selected the median task by hand.

There clearly is a significant correlation between the year in which the task was used and its perceived difficulty. (A huge majority of shaded cells is below the diagonal, which means that the tasks from later years were labeled as more difficult.) The only significant difference between the two tables is in the row/column 9 corresponding to the "SetStack" task from NWERC 2006. This task is deceiving: it seems solvable by a plain simulation, but this is false. The replies in our survey match the actual results of the contest, where 31 out of 42 teams attempted to solve this task, but only 9 of them solved it. In the right half of Table 2 we see that the higher rated responders found this task harder.

Finally, we note that the trend of increasing median task difficulty seems to stall in last five years, hence it would be interesting to repeat the survey after several years to find out more about this new trend.

5 Comparison of performance in different years

We already argued that when discussing task difficulty we have to take into account the skill levels of the contestants. However, the converse is not necessarily true: we may be able to compare the skill levels of contestants in different years. Where does the symmetry break? For the contestant, a few years can mean all the difference in the world – a few years of practice (or inactivity) can strongly influence the contestant's skill in solving the tasks. However, a task after several years is precisely the same task. Hence the tool that can help us compare skill levels of the contestants in different years is a task that was given in different years to (ideally) two disjoint sets of contestants. We were able to discover multiple such situations and we analyzed them.

5.1 TopCoder tasks

In general, TopCoder does a pretty good job in avoiding repeating the same task. Still, we were able to find several pairs of almost identical tasks in past TopCoder matches. Statistic data on these tasks is given in Table 3. The column "opened" is the number of contestants who attempted to solve the task, "solved" is the number (and percentage) of those who managed to submit a correct solution. The next two columns give the fastest and the average solving time, respectively; the average is only computed over all contestants who solved the task. The last column is the number of contestants who competed in both rounds.

Notable facts: By observing the number of solvers and the average solving times for Div1 tasks, the rate of their improvement is obvious – both for the top and the average Div1 contestants. On the other hand, the performances on the easy Div2 tasks are almost identical in early 2007 and late 2008. Essentially the same task was used as a hard task in 2007 and as a medium in 2009.

5.2 Slovak selection camp

Tasks are sometimes reused in the Slovak selection camp – a week-long camp for approximately the best 10 contestants in the Slovak Olympiad in Informatics,

task	**date**	**level**	**opened**	**solved**	**fastest**	**average**	**overlap**
Layoff	Mar 2003	div1 hard	138	11 (7.97%)	0:21:45	0:35:51	10
Terrorists[a]	Jan 2007	div1 hard	385	102 (26.49%)	0:02:44	0:18:13	
InstantRunoff	Dec 2003	div1 easy	160	85 (53.13%)	0:07:07	0:23:01	10[b]
InstantRunoffVoting	Mar 2008	div1 easy	583	469 (80.45%)	0:02:48	0:16:59	
InstantRunoff	Dec 2003	div2 med.	187	37 (19.79%)	0:16:08	0:44:42	0[b]
InstantRunoffVoting	Mar 2008	div2 med.	759	220 (28.99%)	0:07:52	0:34:03	
Graduation	Jun 2004	div1 hard	120	2 (1.66%)	0:20:13	0:30:07	12[c]
SharksDinner	Jul 2007	div1 hard	379	51 (13.46%)	0:10:27	0:24:18	
CountPalindromes	Feb 2007	div1 hard	379	11 (2.90%)	0:25:47	0:42:47	63[d]
PalindromePhrases	Apr 2009	div1 med.	447	61 (13.65%)	0:07:21	0:41:19	
Palindromize	Jan 2007	div2 easy	569	371 (65.20%)	0:01:42	0:20:54	39
ThePalindrome	Dec 2008	div2 easy	751	454 (60.45%)	0:02:47	0:19:59	

[a] Layoff is a plain maximum flow task, Terrorists requires finding multiple minimum cuts.
[b] Two more people competed in Division 2 in 2003, and in Division 1 in 2008.
[c] None of the two solvers in 2004 took part in 2007.
[d] Only 24 of these managed to solve the task. The other 37 solvers did not participate in 2007.

Table 3: Pairwise similar tasks used in TopCoder contests over the years.

where the IOI team is selected. The data for those tasks is presented in Table 4. For each task, we normalized the score to 100 points. The column "top 4" contains the sum of the normalized scores of the four best solvers for that task, the column "OK" is the number of participants that scored at least 90% of points for the given task. In the column "medals" you can find the medals acquired by the Slovak IOI team that year – G, S, B standing for gold, silver, and bronze, respectively.

Out of the 11 tasks, only in 5 of them was the best result obtained in the most recent year the task was used. From this observation, we have to conclude that in the data from the selection camp we do not observe the increase of skill levels that was clearly visible on the international scale. We see two factors that could contribute to this contradiction significantly.

First, the Slovak OI is so small that it is not statistically significant. With only about 100 students competing in this contest each year, the skill levels of the top ones can vary wildly from year to year. And judging by the Slovak results at the IOI, they indeed do vary. As for the data from the selection camp, kindly note that for 9 of the 11 tasks[5] the best result in the selection camp occurred in the year in which Slovakia obtained the best medals.

Second, in Section 5.1 we were not able to consider one important factor – the ages of the contestants. Note that the TopCoder data set contained contestants of all ages, whereas the selection camp is always attended by secondary school students only. At the moment, we do not have sufficient data to look into this, and we consider this to be an interesting question for further research.

[5] The task "sally" is one of the other two. For this task the conditions in 2003 and 2009 were not identical. This is a task solvable by brute force. In 2003 the task was used with a strict time limit, and implementing good pruning was necessary to get a full score. In 2009 all correct solutions got a full score.

task	**year**	**OK**	**top 4**	**medals**
wine	2009	7	400.0	SSB
	2007	7	400.0	SSB
syr	2003	4	400.0	GSB
	2001	4	400.0	GGSS
sally	2009	8	400.0	SSB
	2003	1	347.6	GSB
dazdovky	2007	1	222.0	SSB
	2004	1	253.3	SSBB
	2001	2	269.2	GGSS
jazdci	2007	1	188.6	SSB
	2003	0	105.7	GSB
	2001	2	337.5	GGSS
klinec	2007	0	292.0	SSB
	2003	3	362.7	GSB
guards	2007	0	166.7	SSB
	2003	0	130.7	GSB
junior	2006	0	147.7	SSS
	2001	1	360.0	GGSS
domceky	2006	2	308.0	SSS
	2002	4	390.0	GSSB
prufer	2005	6	390.0	GGGG
	2001	4	384.0	GGSS
gule	2004	1	215.0	SSBB
	2002	0	119.3	GSSB

Table 4: Tasks reused in the Slovak selection camp

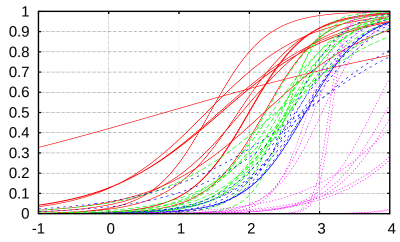
6 Evaluating task difficulty using IRT

Item Response Theory [2] is a modern testing theory that gives us a set of new tools to analyze contest results. Among other things, this theory allows us to model tasks with different difficulties. More precisely, the two-parameter logistic model is a suitable model to describe programming contest tasks. In this model, each contestant c is assumed to have a scalar ability level θ_c, and each task t is assumed to have two scalar parameters a_t and b_t that describe its difficulty. In this model, the probability that a contestant with ability θ_c will solve a task with parameters a_t, b_t is $Pr(\theta, a, b) = 1/\left(1 + e^{-a(\theta-b)}\right)$. (Note that b is equal to the value θ for which $Pr(\theta, a, b) = 0.5$, and a is the derivative in this point.)

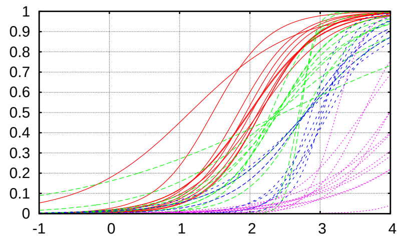
In [5] the author of this paper developed a framework how Item Response Theory can be applied to the results of programming contests. Using this framework, it should be possible to make a thorough analysis of the past contests, compute maximum likelihood estimates of their task parameters, and use this data to draw conclusions about the relative difficulty of those contests.

Due to space restrictions we will not attempt such analysis here. Instead, we will just limit ourselves to a single example of such results. Using the method described in [5] we computed the maximum likelihood estimates for contestant and task parameters describing all tasks from TopCoder's SRMs in 2007 and 2008, and all contestants that took part in those matches.

In Figure 3 we show the plots of item characteristic curves for all Div1 hard tasks used in matches in 2007 (left) and 2008 (right). The x axis on both plots represents ability levels (the entire range being $[-5, 5]$), the y axis is probability of solving a given task, and each task corresponds to one of the plotted logistic functions. In layman's terms, the further to the right a curve is, the harder the task – because this means that the contestant's ability has to be large in order to have a chance to solve the task.



(a) 2007.



(b) 2008.

Fig. 3: Item characteristic curves for hard tasks in TopCoder matches.

For easier readability of the plot, we ordered the tasks according to their b_t parameter, grouped them into four quarters, and plotted each quarter using a different color and pattern. Note that the parameter b_t is equal to the ability level necessary to have a 50% chance of solving the task. In other words, b_t is the x coordinate of the point where the logistic curve crosses the line $y = 0.5$.

Observe the line $y = 0.5$ to see the values b_t in Figure 3. The first quartiles of b_t are almost the same (2.23 in 2007, 2.3 in 2008). However, note that in 2008 almost all b_t in the first quarter exceed 2. The median task in 2007 is already significantly easier than in 2008 (2.6 vs. 2.8), and the trend is even more visible at the third quartile (2.82 vs. 3.23).

7 Conclusion

Programming contests are still growing in popularity. The community of contestants is growing, and this puts an added pressure on their preparation (and performances). The contestants are getting better and better each year. As a consequence, the tasks in the contests they solve must become harder and harder, to be able to distinguish between the top contestants.

In this article we give sufficient evidence that this process is still happening. When will it stop? In our opinion, we should see parallels between the area of programming contests, and all of Computer Science – which is still one of the most rapidly developing sciences. As long as the boundaries of Computer Science grow at the current pace, the contests will mimic this growth, and incorporate the new discoveries into new, even more challenging tasks – thereby preparing a future generation of scientists who can push the boundaries even further.

We see many open questions and directions for further research in this area. To list some: Is this process a good thing, or should we attempt to stop it? Aren't the contests already too difficult? How does the gradually increasing difficulty impact newcomers? To what extent does the increase in difficulty also require an increase in mathematical skills of the contestants? What is the relation between the age of contestants and their skill levels? How much more insight can the Item Response Theory based models give us about the results of past contests, and how can they be used in order to prepare better contests in the future?

References

1. ACM International Collegiate Programming Contest. http://cm2prod.baylor.edu/ (accessed 2008)
2. Baker, F. B., and Kim, S.: Item Response Theory: Parameter Estimation Techniques. CRC http://edres.org/irt/baker/ (2004)
3. Burton, B.: Breaking the routine: events to complement informatics olympiad training. Olympiads in Informatics 2, 5–15 (2008)
4. Fenwick, P.: A New Data Structure for Cumulative Frequency Tables. Software – Practice And Experience 24, 327–336 (1994)
5. Forišek, M.: Theoretical and Practical Aspects of Programming Contests. PhD thesis, Comenius University (2009)
6. Forišek, M.: Using Item Response Theory to Rate (Not Only) Programmers. Olympiads in Informatics 3, 3–16 (2009)
7. Greve, M.: UVA toolkit. http://uvatoolkit.com/ (2009)
8. International Olympiad in Informatics. http://ioinformatics.org (accessed 2009)
9. Internet Problem Solving Contest. http://ipsc.ksp.sk/ (accessed 2009)
10. Kelevedjiev, E., and Dzhenkova, Z.: Tasks and training the youngest beginners for informatics competitions. Olympiads in Informatics 2, 75–89 (2008)
11. Kemkes, G., Vasiga, T., and Cormack, G.: Objective Scoring for Computing Competition Tasks. LNCS 4226 – Proceedings of ISSEP 2006, 230–241 (2006)
12. Kiryukhin, V.: The Modern Contents of the Russian National Olympiads in Informatics. Olympiads in Informatics 1, 90–104 (2007)
13. Kiryukhin, V., and Okulov, S.: Methods of Problem Solving in Informatics: International Olympiads. (In Russian.) Izdatelstvo BINOM. (2007)
14. Manev, K.: Tasks on graphs. Olympiads in Informatics 2, 90–104 (2008)
15. Navernіouk, I.: Igor's UVa Tools. http://shygypsy.com/acm/ (2009)
16. Opmanis, M.: Team Competition in Mathematics and Informatics "Ugāle" – finding new task types. Olympiads in Informatics 3, 80–100 (2009)
17. Revilla, M., et al.: University of Valladolid (UVa) Online Judge. http://uva.onlinejudge.org/ (2009)
18. Revilla, M., Manzoor, S., and Liu, R.: Competitive Learning in Informatics: The UVa Online Judge Experience. Olympiads in Informatics 2, 131–148 (2008)
19. Skiena, S., and Revilla, M.: Programming Challenges. Springer (2003)
20. TopCoder, Inc.: Algorithm Data Feeds. http://www.topcoder.com/wiki/display/tc/Algorithm+Data+Feeds (2009)
21. Truu, A., and Ivanov, H.: On Using Testing-Related Tasks in the IOI. Olympiads in Informatics 2, 171–180 (2008)
22. Verhoeff, T.: 20 Years of IOI Competition Tasks. Olympiads in Informatics 3, 149–166 (2009)
23. Verhoeff, T., Horváth, G., Diks, K., and Cormack, G.: A Proposal for an IOI Syllabus. Teaching Mathematics and Computer Science 4, 193–216 (2006)
24. Verhoeff, T., Horváth, G., Diks, K., Cormack, G., and Forišek, M.: IOI Syllabus for IOI 2009. http://www.ioi2009.org/GetResource?id=32 (2009)

A Survey on NWERC tasks

The survey contained a list of 12 tasks, ordered randomly. The order was different for different people. The purpose of this randomization was to ensure that the respondents have no preconceptions on the difficulty of the tasks.

These are the exact instructions given to the people taking the survey:

Your task is to **order** the following ACM ICPC problems based on **how difficult** you find them.

– Click the problem name to read the problem statement. *Please read the problem statements and think about the solutions!*
– Consider the overall difficulty of solving the task, from opening the problem statement to submitting a correct solution.
– Use the dropdown boxes to assign numbers 1 to 12 to the problems.
– You *may assign the same number* to problems that have approximately the *same difficulty* according to you.
– You also may *only rate a subset of the tasks* and omit those where you are not certain. Even such feedback is valuable.
– The absolute values of numbers you use to rate the tasks do not matter. The only thing that matters is: *If you give task A a **smaller** number than task B, you think that A is **easier** than B.*
– The tasks are shown in random order. This order changes when you reload the page. Hence it is a good idea not to reload the page while you are entering your choices.