Olympiads in Informatics, 2008, Vol. 2, 131–148

Competitive Learning in Informatics: The UVa Online Judge Experience

Miguel A. REVILLA
Applied Mathematics Department, University of Valladolid
Prado de la Magdalena s/n, 47011-Valladolid, Spain
e-mail: revilla@mac.cie.uva.es

Shahriar MANZOOR
Computer Science and Engineering Department, Southeast University
24, Kemal Ataturk Avenue, Dhaka, Bangladesh
e-mail: shahriar_manzoor@yahoo.com

Rujia LIU
Department of Computer Science and Technology, Tsinghua University
Qinghua Yuan, Haidian District, 100084 Beijing, China
e-mail: rujia.liu@gmail.com

Abstract. The UVa Online Judge is probably the oldest and one of the most recognized programming contest training sites for ICPC format contests. It is an automatic judging system where anyone from around the world (regardless of being a contestant or not) can submit his solution to the archived problems to check its correctness and improve his programming skill in the process. Although the judge was initially developed to be used as a trainer site for potential competitors in the international programming contests (mainly ACM ICPC), we have observed that it is a very good tool for self-study. In the present paper some facts from the history of the site are given. Then the paper focus to the nature of self-competitive learning by analyzing the more frequent response sequences to the users from the judge along these 10 years. And by doing so we identify the main differences between the behaviors of the users when they are just training and when they are competing.

Key words: competitive learning, programming contests, online judge, informatics.

1. Introduction

Programming contests are probably the fastest expanding co-curricular activity related to computer science. The main reason could be that the new Technologies of Information and Communication (TIC) allow us to arrange all kind of interactive activities without too much infrastructure. Of course, the educational processes are an ideal target to use these tools as they offer multiple options to the teachers as well as to the students. It seems evident that one of the favourite topics to focus the modern e-learning systems must be informatics, as it is the base of most of the involved and developing tasks. The fact is that the programming lovers, whether they are secondary, high-school, or university students have a lot of choices to attend to programming contests. For example a university

student can participate in ACM ICPC, the national contests of his own country, local programming contest of his university or programming contests arranged by TopCoder and different online judges like UVa, SPOJ, etc. Moreover, it is an activity that can provide a method for attracting interest in computer science, as it is accessible to beginning students.

It's clear that a programming contest is, by its own definition, a competitive activity, where there are winners and others (not really losers, in general). Usually it's an additional and also co-curricular activity and in that sense they can be seen as a good model of competitive learning. Moreover, many of the programming contests are team competitions and they involve a lot of collaborative work to prepare them. In fact, the training process involves several interesting learning strategies that have nothing to do with the real competition, but with systematic pedagogical methods, which can be very positive for the student's formation and maybe neutralize the negative effects that many people impute to any kind of competitive learning activity.

There exist many online judges on the internet that can play a very important role here. An online judge is in general a server, which contains descriptions of problems from different contests, as well as data sets to judge whether a particular solution solves any of these problems. A user from anywhere in the world can register himself (or herself) with an online judge for free and solve as many problems as he likes. He can send as many solutions as he want till receiving satisfactory information, not only about the verdict, but also about the time that the code takes to run after improving the program and/or the algorithm used to solve the selected challenge. One of the main distinctive trait of the online judges is that they allow the users this self-competitive behaviour to learn informatics, not only algorithms but also programming.

2. A Brief Story of the History of the UVa Online Judge

First of all, let's remember here the name of the person mainly responsible for the existence of the University of Valladolid (UVa) Online Judge: Ciriaco García de Celis. He was a student of informatics when in November of 1995 the first version of the judge started working a few hours before the first local qualifying contest to select a team of the UVa for going to compete in the ACM-ICPC South Western European Regional Contest (SWERC). For more than eight years he was the wizard inside the judge. He, worked almost alone, designed and implemented the kernel of the judge and he also maintained the system as well as the successive migrations from one computer to another, from one version to the other. But maybe the harder work was to fight and win against the many hackers we have had as normal and habitual users. That initial version (written using Unix standard **sh** scripts) was partially rewritten in order to add some improvements to support a 24-hours judging system, capable of working without the presence of a system operator. For example, an automatic system needs to be able to detect and skip e-mail loops.

However, Unix scripts were not powerful enough to support a true reliable judging system. For example, it was not possible (at least under Linux) to limit the memory used

by a submitted program when being executed. And the judge architecture was not designed to generate events reporting its status (external utilities showed the internal judge state by polling it periodically).

For this reason, new judge software was developed. This new judge was able to work as a 24-hour Online Judge and a programming contest judge. However, it still missed many components required by a general conception for Contest Judge, as then it matched almost completely with the ICPC (the granddaddy of the programming competitions, as far as we know) model both for the problems style and for the contests dynamics and rules. However, Fig. 1 shows us that a lot of services were already included in the planning in order to provide to the users a tool to use for learning, while they train for the contests.

After checking the system for a short period of time, when some students in Algorithmics at the University of Valladolid were the only allowed users, the UVa Online Judge started its open period in 1997 with a hundred problems and a little promotion, on a day of April 1997-04-15 14:31:48 (UTC) is the date of the first submission (it was made by Ciriaco and was successful, of course). But, any person in the world can get access to it via the internet for free. The site then began to be more and more known, as programming contests were becoming more and more popular. For about two years the judge didn't really change, except minor bug fixing, or adding three or four new volumes (a volume is a set of a hundred problems) taken from different web sites, mainly corresponding to ACM-ICPC regional and final contests.

In November 1999 the University of Valladolid hosted the official SWERC and then we realized that we must work on the environment of the judge. It's the first time we were aware about a bunch of services that we still needed to develop before having a site able to support the quickly increasing number of users and submissions and to host online contests. A group of voluntary students collaborated to plan out and then to implement a lot of new services: an electronic board, a friendly interface, a detailed set of statistics,

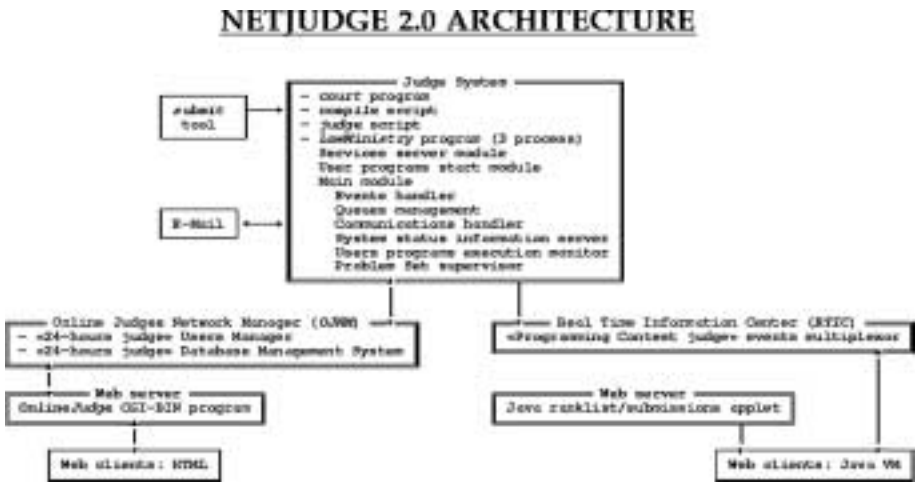

Fig. 1. Diagram of the different Online Judge modules. Each module can be located in a different computer, and are communicated by TCP connections.

rankings, etc. That team of people was the base of the users' community of the UVa judge; they started the big task to analyze every possibility we had to transform our practice and test site in a real project with plenty of objectives. Their enthusiasm and also many of their ideas are still present today.

Of course, the most important and urgent work to do was to implement a robust contest system to be used by the real time contest. After the contest some of the members of that team continued working to consolidate the tasks already done and to develop the main goal we had talked about: to have our own regular online contests. As UVa had to arrange the SWERC in November 2000 also, we decided to do it as fast as possible and by the month of July, exactly on 2000-07-14 at 14:00:00 (UTC) the first test contest was open to all the world and 5 hours later finished successfully.

There are many other important dates and facts in the life of the UVa Online Judge. The main evolutions were due to Fernando Nájera that included in 2002 the use of a SQL real database to keep all the information ready for the users in real time and the PHP tools to manage the interface easily and, of course, to Carlos Casas who is quite well known for all the users as he is still a very active member of the UVa site. A special mention is deserved for our online contests. Under the management of Shahriar Manzoor have increased the level of the problems on our site in quantity but especially in quality. In fact, as of today the set of problems specifically written for our judge are probably the main asset we have and surely our main pride. Many other people arranged some high quality online contests from the early stages such as Professor Gordon Cormack of University of Waterloo, Rujia Liu of Tsinghua University and Md. Kamruzzaman of BUET (now he is at UCSD).

Fig. 2 shows the classification by languages of the 5899124 programs submitted by 63351 users from about 180 different countries till the date 2007-09-06, 17:19:45 (UTC),

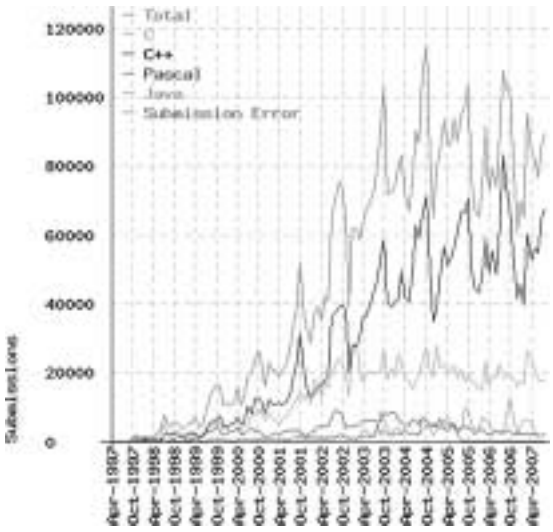

Fig. 2. Robot Judge submissions by Programming Language throught September 2007.

when after 170 online contests (more or less a half of them arranged by our own team) the old judge was definitely stopped (UVa Online Judge). Two hours later a completely new robot restarted working at a new server at the Baylor University, the headquarters of the ACM-ICPC contest. It has been developed and implemented by Miguel Revilla Jr. and it incorporates the whole history of these ten years, all the amazing information about this extraordinary experience. So the UVa Online Judge continues its new journey at the CLI website (CLI).

3. The Analysis of Statistics

The immense amount of data from users in different parts of the world provides the opportunity of making lots of analysis and it opens up the possibility of getting some important results. These results may enable us to find out what aspects of online judging or the programming contests need to be changed to make it more meaningful, how practice and experience improves the performance of a contestant and many other interesting issues. It will also create new openings on which we can continue our study in future. For example, it will help us to identify the geographic locations where it's almost impossible to compete in programming contests and then we can take initiatives for those regions, and try to extend our internet community to those parts of the world.

Although there is no academic or social bondage between the members of this online judge community, they are still training in our judge, testing (individually or by groups) their programming skills, self-competing or arranging contest in our server, discussing on our board and participating in our Online Contests just to be better programmers and to learn more complex and new topics. Many of them may not even see each other but still they are good friends, helping each other out in many occasions. An extreme example may be that the three authors of this paper have been collaborating since year 2001, but Miguel and Shahriar first met in 2005 and Shahriar and Rujia also first met in 2005. And the occasion was the 2005 ACM ICPC World Finals that took place in Shanghai. Miguel and Shahriar still meet only annually, while they are still waiting for their second opportunity to meet Rujia Liu.

The main conclusion we have made is that we are managing a tool with a very large potential not only for training but also for teaching informatics, a merge of competitive and cooperative learning, let's say a kind of collaborative competition. Moreover, it can be successful where the conventional systems often fail: to make students curious, to make them do non-academic works that often seems more interesting or to enhance their creativity. As everybody can access to these online judges and start practicing with easy and funny problems, it could be a good way to attract the newcomers (for example, secondary school students) to the world of programming and to computer sciences in general. It is very important in a world where many of us including some software giants have a perception that the young generation is losing interest in computer science.

In subsequent steps, from the perspective of algorithm design, the programming contest is a treasure trove. There appear to be numerous ways to solve the same problem. But

also for software reliability engineers this is the case: there are even more ways to not solve the problem. Most authors first submission is incorrect. They take some trials to (in most cases) finally arrive at the correct solution.

Suppose one submits a program in a contest and gets accepted, another contestant submits a program he gets wrong answer and then he submits again he gets accepted, another contestant submits a program six times and every time he gets wrong answer. Which one of these three events is more likely to happen in a programming contest? To find it out we analyzed all the submissions of UVa site and found out which are the most common response sequence for a contest. We actually took a method like digraph, trigraph analysis of a text. First we tried to analyze which submission response is most common for a problem. And the most frequent response sequences are given in the Tables 1 and 2. These tables are based on the analysis of the first 4 millions submissions to the UVa till October 2005, but the amount is enough to be statistically significant.

In fact, the partial analysis we have done later show that the order of popularity is more or less the same although the number of submissions now is near of seven millions. Only the number of PE (Presentation Error) verdicts has decreased proportionally as we have purged the data sets to fix some trivial mistakes.

Table 1
A table for most popular response sequence

Monograph	**AC**	**WA**	**CE**	**TL**	**PE**
Frequency	465516	214187	104952	76806	73526
Digraph	**WA|WA**	**WA|AC**	**AC|AC**	**CE|CE**	**TL|TL**
Frequency	164521	71018	49743	39732	30830
Trigraph	**WA|WA|WA**	**WA|WA|AC**	**CE|CE|CE**	**TL|TL|TL**	**AC|AC|AC**
Frequency	92545	32765	20049	14436	14203
Tetragraph	**WA|WA|WA|WA**	**WA|WA|WA|AC**	**CE|CE|CE|CE**	**RE|RE|RE|RE**	**TL|TL|TL|TL**
Frequency	55504	16518	11566	7947	7474

Table 2
A table for most popular responses ending with an AC

Monograph	**AC**				
Frequency	465516				
Digraph	**WA|AC**	**CE|AC**	**TL|AC**	**PE|AC**	**RE|AC**
Frequency	71018	18099	10612	9213	8205
Trigraph	**WA|WA|AC**	**CE|CE|AC**	**TL|TL|AC**	**CE|WA|AC**	**RE|RE|AC**
Frequency	32765	4685	3540	3511	2620
Tetragraph	**WA|WA|WA|AC**	**CE|CE|CE|AC**	**CE|WA|WA|AC**	**TL|TL|TL|AC**	**RE|RE|RE|AC**
Frequency	16518	1750	1636	1340	1158

Many comments can be made based on these tables. But some things are obvious:

a) When individual contestants make a particular type of mistakes for a problem they tend to make the same mistake again, which encourage the group to try working together for the contests. Let's say one more time, competitive and cooperative learning in informatics are not opposite but complementary.

b) We can say that if someone gets five consecutive wrong answers then in the next submission he is four times more likely to get a wrong answer than an accepted verdict. That means that after four or five errors, the best is to analyse carefully what happen as, probably, the mistake is not trivial.

c) It is very important the influence of the kind of mistakes in these sequences. That is, mainly, because some responses of the judge give us information about the error and others tell us nothing at all. This is very important in order to improve our system judge to become a real learning tool, by adding new features.

As of now, we can divide the judge responses into two types: Informed Response and Uninformed Response. These divisions will help us to propose a combined system to bring IOI and ICPC closer later on. Informed responses are the responses that allow the contestants to know whether their program logic is correct or not correct: AC (for ACcepted), PE (for Presentation Error) and WA (for Wrong Answer) are such types of responses. The other three TL (for Time Limit exceeded), RE (for Run-time Error) and CE (for Compilation Error) are uninformed responses, because it is not known what would have happened if the program, in case it starts, was allowed to run longer or not crashed. And, of course, unless we give one test case per file as input it would be impossible to judge the 'degree of correctness' of the submissions that get TL or RE in the present ICPC system. Including new models of judging, the grading system used at IOI is the main project we are working on.

Similar statistics could be done about the submissions to the online contests arranged on our server, but after 135 contests over five years, we saw that the great figures were very similar. But there are other very interesting details to analyze in the contests. In fact, whether it is an Online Contest or in the 24 Hour Online Judge the acceptance rate is around 30%. But this acceptance rate is not so bad when we consider the statistics of accepted problems only. For example suppose there are eight problems in a contest A, B, C, D, E, F, G and H. One team solves problem A, B and G and attempts problem C and D. In this section we will not consider the judge responses for problem C and D for that team.

Table 3 shows the judge response statistics, but considering only the initial submissions from a team for which they finally got an accepted verdict. It is found that its probability of getting Accepted in the first submission is 44.16%. The percentage of informed responses is 80.89% and uninformed responses is 18.14%. But more important is the fact that percentage of informed errors is 36.73% and of uninformed errors is 18.14%. So their ratio is roughly 2:1.

Table 3

Judge response statistics for accepted problems/team only

Verdict	Percentage	Informed vs uninformed response	Informed vs uninformed errors
AC	44.16	80.89%	Not considered
PE	3.08		36.73%
WA	33.65		
TL	8.03	18.14%	18.14%
RE	3.72		
CE	6.39		
Others	0.97	Not considered	Not considered

4. How Does Practice Change Things?

The most important part of the analysis of the millions of programs we have received at the UVa Online Judge is to know our users and try to learn from them as many details as we need to improve our services. And not only about the demographic distribution, as we told above, but also about their evolution along the time. We certainly hope that almost all of them have improved their skills in programming and algorithms, but it's interesting to quantify this fact. And, as far as it's possible, try to get an idea about the different patterns of the verdicts in function of a user is a newcomer or an expert in the use of the judge. Of course, there are many details we can't be sure about most of these users, we don't know if they are individuals, a regular team or a variable group. It would be a tuning process, almost impossible to do, of selection of submissions in order to get more categorical conclusions.

Table 4 shows the error rate of people with different experience. The first column on the left actually describes the experience of the user that is being considered. Experience means the number of problem he has solved in the judge, it does not consider anything

Table 4

Based on all problems

Solve Range	**AC**	**PE**	**WA**	**TL**	**RE**	**CE**
0–49	23.76	4.93	36.13	8.36	8.01	12.24
50–99	33.81	5.57	34.18	7.33	7.54	6.35
100–149	35.08	6.41	33.59	6.70	7.50	5.62
150–199	37.02	4.95	33.01	7.07	6.90	5.70
200–249	37.74	5.01	32.85	7.11	6.83	5.31
250–299	39.90	4.60	32.41	6.89	6.16	5.17
300–349	40.86	4.08	32.56	7.34	5.87	4.63
350–399	42.03	4.30	32.21	6.51	5.97	4.49
400–449	41.96	4.03	32.16	6.86	6.37	4.05
450–499	41.82	3.65	31.50	7.10	5.98	4.68
500+	42.36	3.53	31.83	8.06	5.42	4.06

about the time he is associated with the judge. Each of the next six columns actually shows the rates of six major judge responses in a programming contest. For example the third row of the table says that the contestants who have solved more than 50 and less than 100 different problems has 33.81% acceptance rate, the rate for wrong answer is 34.18 and so on.

Table 4 can have some interpretation troubles, because their can be some confusions: as people solve more problems they have less easy problems to solve (assuming that people tend to solve easy problems first). When someone has already solved 400 problems he has no more easy problems to solve, so his acceptance rate can go down a little. But as he is more experienced the acceptance rate does not go down but remains similar. In Table 5 and Table 6 we have put the same results but this time separated by the 'experimental' difficulty of the problems based on their low or high acceptance rate.

Table 5

Based on problems with low (less than 25%) acceptance rate

Solve Range	AC	PE	WA	TL	RE	CE
0–49	11.09	1.71	41.73	14.62	12.43	11.52
50–99	17.45	2.15	42.48	13.25	12.00	6.88
100–149	18.98	2.69	42.13	11.85	12.44	6.16
150–199	20.29	2.37	41.61	12.47	10.79	6.23
200–249	20.86	2.46	42.17	12.78	10.15	5.77
250–299	23.09	2.37	41.91	12.43	9.19	5.16
300–349	24.24	1.94	42.17	12.46	8.92	5.01
350–399	24.15	2.54	42.99	11.30	9.44	4.92
400–449	25.61	2.33	41.32	11.42	9.51	4.47
450–499	27.21	2.09	38.57	12.36	8.89	5.38
500+	27.20	1.65	41.04	13.53	7.20	4.24

Table 6

Based on problems with high (more than 50%) acceptance rate

Solve Range	AC	PE	WA	TL	RE	CE
0–49	40.81	6.67	26.37	4.03	4.00	11.79
50–99	53.86	7.47	21.77	2.99	3.37	5.94
100–149	53.97	9.18	21.18	2.51	3.31	5.38
150–199	58.33	7.21	18.66	2.57	3.10	5.33
200–249	59.67	6.66	19.25	2.39	3.02	4.78
250–299	62.30	6.24	18.25	2.29	2.39	4.51
300–349	64.56	6.40	16.42	2.12	2.65	3.92
350–399	64.44	5.01	17.48	2.12	2.44	3.91
400–449	65.17	6.26	17.74	2.23	2.13	2.84
450–499	63.15	5.19	17.50	2.10	2.72	4.68
500+	67.73	4.31	15.46	2.22	2.33	3.97

Tables 5 and 6 indicate that with practice the acceptance rate increases a lot, mainly for the problems with high acceptance rate, and also compilation errors decreases a lot and quickly for all the three categories. But, surprisingly, wrong answer and TL percentage does not change that much, even for the group of very expert users. So does this indicate no matter how experienced you are you can always get wrong answer? Of course, every person will always have a harder problem to solve, a new programming challenge to face in order to continuously increase his skills in informatics.

Usually, by 'programming ability' people means coding, debugging and testing. Though, these individual abilities greatly affect cooperative works too (it's easy to suppose that many of our users work in group, being a team or not). It's better for the team members to use the same language and similar coding conventions. In such way, if one cannot find his bug then one can ask another person to read her/his code. Though programming is the very first skill, it needs improving all the time. For example, coding complex algorithm can only be trained after studying these algorithms.

Most people got started by solving easy problems. Here, by easy problems, we mean the problems in which you only need to do what you're asked to do, i.e. a direct implementation of the problem description. For example, do some statistics, string processing or simulation. These problems mainly require some coding ability but not any sophisticated algorithm, deeper mathematics or logical insights. When getting started, practice is much more important than theory. (Practice, practice and practice).

Everyone is encouraged to program as much as he can, as long as enthusiasm is perfectly kept. But there is one thing you need to know first: ICPC, IOI and most of the existing contests concentrate on problem solving and apparently the enjoy of programming comes from solving easy problems in which you only need to do what you are asked to do, i.e. a direct implementation of the problem description. But keep the limits. Trying to solve more problems is good, but the quantity is not the most important thing. When you've managed to solve 50 easier problems somewhere, it's better to seek for more challenges. In other word it is better to solve many problems of various kinds and difficulty. In real contests and online judges, there are a large number of problems that require a few lines of code but more maths and algorithmic thought. So when you are challenged with problems that are more interesting and difficult, you will find it necessary to think about something serious: becoming a great contestant.

5. The EduJudge European Project

The users of Online-Judge are demanding a greater pedagogic character for this tool (at least one request per week is sent to the UVA On-line Judge creator via email, also some requests are available in the forum `http://online-judge.uva.es/board/`). For example, teachers would like to use it as one more activity for their official courses. This requires the possibility of managing courses and students and an extension of the current functionalities of the Judge so that it can provide gradual evaluation or different difficulty levels of problems. On the other hand, the set of problems is continuously being

incremented but it is necessary to give the problems an adequate and common structure, adding metadata and creating a search engine so that the problems are more accessible for the community of teachers.

It is easy to understand that these sets of achievements are only possible within the frame of a collaborative project involving experts from several countries and different areas of knowledge. This is the origin of the EduJudge project. EduJudge is an innovative system based on ICT that can be incorporated into the learning processes in the mathematical and programming field and is addressed to higher education students and secondary education students. It has been managed and coordinated by CEDETEL (Centre for the Development of Telecommunications in Castilla y León), a non-profit Technology Centre located in Spain. The project has been funded with support from the European Commission into the frame of the Lifelong Learning Programme of the European Union. Other than the University of Valladolid, there are three more partners from different European countries: the University of Porto (Portugal), the KTH Royal Institute of Technology (Stockholm, Sweden) and the Institute of Mathematics and Informatics (Vilnius, Lithuania).

The main goal of the project is to give a greater pedagogic character to the UVA Online Judge, and adapt it to an effective educational environment for higher and secondary education. We want to give the Online Judge a pedagogical character by means of a redesign, improvement of contents and its integration into an e-learning platform. All these will contribute to the development of quality lifelong learning, and also to promote innovation providing new methods of teaching through contests, instead it being focused exclusively on competition. The work packages UVa leads will make it more suitable for its use on a learning environment. The different tasks are:

- Solution quality evaluation: the user will receive a more complete feedback from the system, not only indicating that the problem is solved (or not) but grading the quality of the solutions. This can range from a no significant solution (less than 50% of correctness) to a completely correct solution (100%).
- Generic Judge Engine: the system will support several problem formats, allowing different kinds of learning approaches. By allowing different formats of problems the system is not limited to a right/wrong evaluation method. There can be problems in which the challenge is not only solving a problem, but solving it in the most efficient way. Also there can be cases where a student's solution must 'compete' against another student solution in an interactive way. Having a generic judge engine that can easily be extended to support more problem formats will make this possible.
- Automatic Test case Generation: the automatic generation of test cases will allow different levels of difficulty in the solving of the problems. Having good quality and heavily checked test cases is essential for a good evaluation of a solution. The creation of such cases by hand is a difficult task. The automatic system should be able to generate good test cases based on a given set of rules describing the format of the test case. There can be another interesting situation with automatic generation of test cases. We just give a trivial example here. Suppose a user is

trying to solve "The closest pair Problem" – given n points find the distance of the closest two points. Now the user finds that he is struggling to solve the problem with $n <= 10000$. So using the generator he can generate test cases with smaller values of n and check whether his program works for that. Or we can even propose a WIKI system where users can submit their own generator, and our input tester will check whether the submitted generator generates according to the specified rule before passing the input to other users' solution. In other words if the problem's input statement specification is editable by the user, he can even generate his own test cases with different values of the parameters and actually solve a very different problem that the original problem setter did not intend to solve. Of course all these changes will be within certain limits so that the original author's solution can solve it. All these will help the teachers to create easier problems for their weaker and/or younger students. Also the teacher can specify in which format he wants the test cases to be IOI, ICPC or any other new format. But to implement all these a good number of dedicated people are needed to be involved with it.

6. IOI vs. ICPC. Is the Convergence Possible?

Looking at the tasks mentioned above, it's clear that one of the more important ideas behind the work package to be developed by the University of Valladolid in the frame of EduJudge, is trying to find a meeting point between IOI and ICPC, as far as it's possible. The reason is they are the two most "academic" of the programming contests existing now and in fact there is a continuity from the first to the second, as many of the contestants of the second had their first contact with this activity in the IOI. Probably a more interesting statistics would be how many have actually won a medal in ICPC, without participating in IOI.

From the point of view we are implied, the automated judging, the first problem we have to face is to try and overcome is the issue of grading. It's evident that this is an additional trouble for the problem setters, because the test cases need to be more carefully selected in most of the problems in order to produce a gradual punctuation correlated with the correctness of the code. Even the description of the problems need to be analyzed in detail to allow different sets of inputs that make it reasonable to claim that a program is 50% correct and to prevent the criticisms about from the people that defend the strict binary system of the ICPC: a program that fails in solving an only case is not correct. Certainly any kind of conventional grading system is closer to our competitive learning objective than the 0/1 approach of ICPC.

The IOI is more positive than ICPC because (i) It allows partial marking unlike the 0/1 approach of ICPC, and (ii) It requires the contestants to solve only three problems in five hours which is a lot of time (even though the contest is by individuals). So anyone with a bad start can make up, because there is no penalty on submission time. So the speed of a contestant is not a strong factor. But the ICPC, in spite of its very strict 'either correct or incorrect', still has some very good sides: it gives real time feedback to contestants about the correctness of their solution and also it is not bad to give some credit

Table 7

Judge response statistics based on accepted problems/team only

Subm. Serial	Cumulative Acceptance Percentage	Acceptance Percentage	Cumulative Number of Acceptance	**Subm. Serial**	Cumulative Acceptance Percentage	Acceptance Percentage	Cumulative Number of Acceptance
1	53.622455	53.622455	24358	**11**	98.908090	0.305999	44929
2	72.686846	19.064392	33018	**12**	99.119428	0.211337	45025
3	82.875069	10.188222	37646	**13**	99.317556	0.198129	45115
4	88.920198	6.045129	40392	**14**	99.493671	0.176114	45195
5	92.631811	3.711613	42078	**15**	99.583930	0.090259	45236
6	94.996147	2.364337	43152	**16**	99.667584	0.083654	45274
7	96.398459	1.402312	43789	**17**	99.749037	0.081453	45311
8	97.367089	0.968630	44229	**18**	99.806274	0.057237	45337
9	98.093561	0.726472	44559	**19**	99.856907	0.050633	45360
10	98.602091	0.508531	44790	**20**	99.894331	0.037424	45377

to the contestants for their speed. Moreover, the three member team structure promotes the cooperative learning added to the competitive situation, because it requires an active interaction between them, which results in a positive interdependence.

So to eliminate the short comings of these two major types of contests we need a contest that (a) Gives partial marks to contestants. (b) Gives real time responses to contestants. (c) Possibly informs the contestant which test cases match (only the serial of test case) and which don't. (d) If we don't use separate files for each set of input no information regarding correctness will be available if the submitted program does not run within the time limit (TL) or crashes (RE) for any one of the inputs. In continuation to this discussion a new probable approach will be proposed after we see some interesting statistics related to the UVa Online Judge programming contest.

Every year in the prize giving ceremony the Chief Judge (aka Head Jury) often loves to say how a team failed to solve a problem after submitting it 30 (thirty) times, or another team got a problem accepted in their 20th attempt. These types of things are mentioned because they are rare events in a programming contest. Before proposing a new model of contest, we tested these kinds of events in our Hosting Contest Service. We were afraid they would be significant more frequent as the users play for nothing really important, but to check their competitive level. Our interests were to extrapolate the new ideas about possible new models of contest by simulating what would be the result with our contest. Then we needed to check our online contest with real ones.

The Table 7 shows the statistics on how many submissions are required to get a problem accepted based on the first 135 online contests of Valladolid Site. We can see that in 10 or less submissions almost 98.6% accepted verdicts are found. It means on average in a programming contest only 1.4% of total accepted problems require more than 10 submissions. But, even more important, almost three from each four contestants get an AC verdict on their first or second submission.

It has already been said that an ideal contest model should have partial credits like IOI

and also real time feedback like ICPC. But ICPC allows the contestant to submit problem infinite times. But a proposal of a contest model giving partial credit and infinite time submission is a bit too much because in each submission the contestant has the option to try different kinds of tests and moreover if he is allowed to know which test cases are getting wrong he might use one of his solution to produce output for some test cases and another solution to produce outputs for other cases just depending on the case number. In our study we also found that the ratio of informed and uninformed errors is roughly 2:1. So we can set a new limit that a team will be allowed to make total eight wrong submissions per problem and another four uninformed responses will be allowed. So a team can get 4 RE and 8 WA for a problem but he cannot get 9 WA because maximum 8 informed errors will be allowed. In other words we can say that total 8 errors will be allowed and first four uninformed errors will not be counted in these eight errors. With this new rule the statistics of Table 7 becomes as in Table 8.

As we are allowing 8 errors if the ninth submission is an accepted verdict, it will be granted. However if a team fails to get the problem accepted in these submissions he will be given the highest points that he obtained among these submissions. Now the question comes how can we prevent poorly written solutions to get good scores? – in this model the answer is simple. As we are allowing the contestant to fix his mistakes we don't need to be as lenient as the current IOI, so partial marks will only be given if someone gets more than 60% of the marks, otherwise he will get a zero. Now the question that may come how weak coders will get marks as there is no lenient rule like the classical 50% rule, and the answer is just to give an easy problem to the contestants to solve so that they can get some marks and let the hard ones remain hard. The total number of problems can also be increased (Say five problems in five hours) to include easy and easy medium problems.

The problem with an ideal programming contest model is that it needs to be fair but it also needs to be simple because the same model will be followed in regional (ICPC) and national contests (IOI). Also some of the models are extremely popular so it will take

Table 8

Judge response statistics ignoring first four uninformed responses and allowing maximum eight informed errors

Subm. Serial	Cumulative Acceptance Percentage	Acceptance Percentage	Cumulative Number of Acceptance	**Subm. Serial**	Cumulative Acceptance Percentage	Acceptance Percentage	Cumulative Number of Acceptance
1	63.077600	63.077600	28653	**10**	99.225096	0.323610	45073
2	80.061640	16.984040	36368	**11**	99.392405	0.167309	45149
3	88.453495	8.391855	40180	**12**	99.509081	0.116676	45202
4	93.021464	4.567969	42255	**13**	99.643368	0.134287	45263
5	95.601541	2.580077	43427	**14**	99.720418	0.077050	45298
6	97.076500	1.474959	44097	**15**	99.795267	0.074849	45332
7	97.932856	0.856357	44486	**16**	99.843698	0.048431	45354
8	98.507430	0.574573	44747	**17**	99.876720	0.033021	45369
9	98.901486	0.394056	44926	**18**	99.898734	0.022014	45379

some time to replace them. All online judges are written in the existing rules and it will take some time to change them as well. Many regions and nations are still struggling to adopt the present simple contest models so the new more complex models can be impossible for them to follow. So a new full proof system can first be followed in international level and then in course of time poured into national and regional level.

7. About the Categorization of Tasks

About classification, serious solvers are not interested in doing some classification of the UVa archive. They think that making it public would take away the fun part. Many times the more important task for solving the problem is to decide the type of the designing technique to use. So, some of our previous attempts have failed. Of course, grouping problems into specific categories is very useful, especially for beginners, teachers or for those who want practice on a particular problem type. The users can then try to solve all variety of problems of the same type to master the technique. And, in the frequent case, when a problem can be assigned to several classes it's also very useful to learn new algorithmic concepts behind the technique itself.

However, maintaining such a kind of list is a really hard task, especially when the number of problems is as big as 2500+, and to do it well is very troublesome. First of all we need a prototype controlled list where to classify the problems. Even though there is an almost standard universally accepted list, the experience shows us that the contribution of the users must be managed if we want to prevent a real chaos. The first version of our judge allowed to the users fill a field to write the algorithm they had used in the code, and many of the people didn't use or made an undesirable use of it and that made the field useless. There are too many details to decide before to go on with these helping tools, because it could be negative and confusing if we are not careful enough.

Talking about difficulty level, the problem is even worse as for most of the problems it is very subjective opinion depending of the expertise of the person making the decision. Probably most the people agree about trivial and very hard classes, but the opinions for intermediate levels can be almost impossible to fix. And this is a very important detail for the learning efficiency of the site. A user trying and trying an "easy" problem without getting a positive answer probably lives a traumatic experience in his mind. Then it must be very clear that the difficulty level is a relative concept and a good idea is that the user completes this kind of information with additional data, mainly heuristics consequences of the statistics. In fact, the only published classifications we have done are included in the book that the first author wrote with the Professor Skiena. After arguing for long and working a lot we decided that talking about popularity was better than difficulty and success rate was better than difficulty.

In fact, there are several pages with information about our UVa Online site (and maybe much more we don't know about, as we can't control every thing, of course). For example, one of the most popular pages dedicated to algorithms and programming contest (Pi algorithmist) contains a section specifically dedicated to the UVa Online Judge with several links to some of those pages, as well as an open subsection about categories. The

sites of Felix and Steven Halim are excellent, with a lot of interesting information, but maybe the users like more the site managed by Igor Naverniouk (Igor's UVa tools) where the problems are classified by difficulty level (trivial, easy, medium, hard, very hard, IMPOSSIBLE) and it allows to compare with each other user as well as to get a tip about the following problems to try in function of the past history.

Of course, we have this information and we check it from time to time, but they are not managed by us. Although we contact the responsible staff of these sites, and try to help them to develop their initiatives, they are independent. But all of them have some common characteristics. For example, there is almost general agreement about labeling only a few problems into each of the categories. At this moment there is not an 'official' categorization of the site even though the data base is ready to do it, and we hope that all these features will be included as a built-in feature in the UVa Online Judge with the collaboration of all these persons.

8. Conclusions

Competitive learning in informatics, as we understand it in the present paper (training to participate in programming contests by using online judges and taking part in internet contests) can be an adequate method to learn algorithms and programming, as it is free of the most frequent criticisms that many other methods have. It's true that the final objective is the competition, and probably a hard competition, but there are a lot of constructive outcomes on the way. It is something like climbing mount Everest. One may not be able to reach the top, but the courage, physical ability required to reach even half the height is praise worthy and requires a lot of skill. It doesn't the matter whether the contest is individual or by teams, most of the work to do is self-competitive as well as cooperative.

Depending on the contest the students are preparing for and the actual stage of training they are in, the teachers in charge may promote different kinds of activities, by group or by individual, to prevent as far as possible negative consequences of competitive learning. From this point of view, probably the ICPC and the team competitions in general, are a level under the individual ones, as IOI, as the 'learning team' criteria require that the common work and the individual effort must go together.

Of course, we can't forget that at the end there is only one (person or team) winner of the real contests. It's clear that winning must be the main goal for all the contestants, but the statistical analysis of the millions of submissions to our UVa online judge shows that many times the users, whatever there is behind, try and try the same problem till they get a successful verdict and/or CPU time, by using the informed responses of the judge as well as the electronic board of the site for checking with the other users results. And in the end the self improvement of individual users is the most important outcome of the practice, not the one champion that we get. Many students qualify for the big events of ICPC, IOI and TopCoder but many more students never qualify for a bigger event, but behind this tangible failure, they become better programmers and thinkers, which may in future help them to become something special.

In any case, remember that we are talking about learning informatics for free (we mean here algorithms and programming) as the main step of the process is the training period and it can be scheduled as a really funny work. Let's cite the starting words of the Programming Challenges book (Skiena and Revilla, 2003): "There are many distinct pleasures associated with computer programming (...). The games, puzzles, and challenges of problems from international programming competitions are a great way to experience these pleasures while improving your algorithms and coding skills."

Acknowledgements

The activities described in this article are part of the project "Integrating On-line Judge into effective e-learning". This project has been funded with support from the European Commission. This publication reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

References

CLI. Competitive Learning Institute at the Baylor University of Texas (USA).
`http://icpcres.ecs.baylor.edu/onlinejudge`
Igor's UVa tools.
`http://shygypsy.com/acm/`
Liu, R. (2008). *Training ICPC Teams: A Technical Guide*. CLIS, Banff.
Manzoor, S. (2006). *Analyzing Programming Contest Statistics*. CLIS, San Antonio.
Pi Algorithmist. The Algorithmist is a resource dedicated to anything algorithms.
`http://www.algorithmist.com/index.php/Main_Page`
Skiena, S.S. and Revilla, M.A. (2003). *Programming Challenges. The Programming Contest Training Manual*. Springer-Verlag, New York.
UVa Online Judge. Online Judge and Contest system developed by the University of Valladolid (Spain).
`http://online-judge.uva.es/problemset`

M.A. Revilla is a professor of applied mathematics and algorithms at the University of Valladolid, Spain. He is the official website archivist of the ACM ICPC and creator/maintainer of the primary robot judge and contest-hosting website. He is involved with the ICPC contest for more than ten years, and now is member of the International Steering Committee of the ACM. He received the 2005 Joseph S. DeBlasi Outstanding Contribution Award.

S. Manzoor was born in Chittagong, Bangladesh on 12th August, 1976. He is probably the first person with the concept of arranging monthly ACM ICPC format online contests. He is also first person to arrange ACM ICPC World Finals Warmups with the help of many other persons and these contest have been arranged for consecutive eight years (2001–2008) via UVa Online Judge. He is also a ACM ICPC World Finals Judge for six consecutive years (2003–2008). He is the chairman of Computer Science and Engineering Department of Southeast University, Bangladesh.

R. Liu is a coach of IOI China national training team – a team consisting of 20 students from which the final national team is selected) since 2002. Being a contestant, he participated in the 2001–2002 ACM/ICPC, winning the champion of Shanghai regional contest in 2001, and then a silver medal (the 4th place) in the world finals, Hawaii in 2002. Being a problem setter, he authored over 10 problems for the national Olympiad, winter camp and IOI team selection contests in the past (2002–2006). Currently he's still active in creating problems for online contests in UVa Online Judge and other programming contests.