`oii-web`: an interactive online programming contest training system*

William Di Luigi[1], Gabriele Farina[1], Luigi Laura[1,2,3], Umberto Nanni[2,3], Marco Temperini[2,3], and Luca Versari[1]

[1] Italian Association for Informatics and Automatic Calculus (AICA)
E-mail: {gabrfarina,williamdiluigi,veluca93}@gmail.com
[2] Department of Computer, Control, and Management Engineering
"Sapienza" University of Roma, Italy.
E-mail: {laura,nanni,marte}@dis.uniroma1.it.
[3] Research Center for Distance Education and
Technology Enhanced Learning (DETEL) - Unitelma University.

Abstract. In this paper we report our experience, related to the online training for the Italian and International Olympiads in Informatics. We developed an interactive online system, based on CMS, the grading system used in several major programming contests including the International Olympiads in Informatics (IOI), and used it in three distinct context: training students for the Italian Olympiads in Informatics (OII), training teachers in order to be able to assist students for the OII, and training the Italian team for the IOI. The system, that is freely available, proved to be a game changer for the whole italian olympiads in informatics ecosystem: in one year, we almost doubled the participation to OII, from 13k to 21k secondary school students.
The system is developed basing on the *Contest Management System* (CMS, http://cms-dev.github.io/), so it is highly available to extensions supporting, for instance, the production of feedback on problems solutions submitted by trainees. The system is also freely available, with the idea of allowing for support to alternative necessities and developments.

1 Introduction

The International Olympiads in Informatics (IOI) are an annual programming competition for secondary school students patronized by UNESCO. First IOI has been in Bulgaria in 1989. The 2015 IOI, held in Almaty, Kazakhstan, saw participation by 83 countries and 322 contestants (each country can have up to four contestants). Participants are usually the winners of national competitions.

Here we first introduce `oii-web`, an interactive online training platform, based on the *Contest Management System* (CMS, `http://cms-dev.github.io/` [9, 10]), that is the grading system used in several programming competitions, including IOI. We built, around `oii-web`, three distinct, in both target audience and functionalities, web based platforms: one dedicated to students preparing for the Italian Olympiads in Informatics (OII), one for the teachers, with a complete course on programming and several resources available, and the third to support the selection and the training of the Italian team for the IOI. We believe that our online training system fills a

* A preliminary version of this paper appeared as W. Di Luigi, G. Farina, L. Laura, U. Nanni, M. Temperini, and L. Versari. Three Uses of the Online Social Programming Training System: On Nature and Purpose of Spreading Algorithmic Problem Solving. *Proceedings of the 8th International Workshop on Social and Personal Computing for Web-Supported Learning Communities, (SPEL 2015)*, State of the art and Future Directions in Smart Learning, 369-379, 2016.

gap, since there are several open source grading systems and several online training platform, but to the best of our knowledge there is no open source solution if one wants to host his own training platform. We report on our experience with the three platforms, designed around the common core, `oii-web`, that allows to navigate through problems, propose solutions, and get feedback about it. The overall system is already apt to be fruitfully used, with educational aims, as a tool for competitive programing. Yet we are pursuing its enrichment with aspects of personalization to trainees characteristics and needs, aiming to better help them enhance their abilities to deal with contest problems: this would be novel, to our knowledge. So, in the last part of the paper we discuss the requirements of such extension showing an initial modeling schema for problems, solutions, and ultimately trainees.

2 Related Work

Here we deal with various topics connected to programming competitions and, more generally, computer programming learning for secondary school students: a web training platform, the organization of national olympiads in informatics, and our experience in broadening the participation to it.

On these topics a crucial information source is the *Olympiads in Informatics* journal, founded in 2007, providing "*an international forum for presenting research and developments in the specific scope of teaching and learning informatics through olympiads and other competitions*". Books such as [12] and [5] provide also essential material about algorithms, data structures, and heuristics needed in programming contests.

The importance and the effectiveness of programming contests in learning programming and, more generally, computer science has been observed and emphasized greatly in the literature: we mention the works of Dagienė [3] and Garcia-Mateos and Fernandez-Aleman [4].

Various kinds of automated support to programming education are met in research since decades. The widest area of investigation seems to be related to introductory programming courses, where students learn to write programs, according to a programming language syntax and semantics, and to solve problems. In this way students are trained on both basic algorithms and their coding. Programming errors are spotted basically in two phases: syntactic and static semantics errors are pointed out by the compiler, while logic/dynamic semantics erros are spotted by testing. So, program assessment is usually based on

- *Static Analysis*, that gathers information about the program and produce feedback without execution. In this family fall approaches based on compiler error detection and explanation [6, 14], structured similarity between marked and unmarked programs [11], and also non-structural analysis, keyword search and plagiarism detection [7].
- *Dynamic Analysis*, that tests the program on accurately chosen input datasets and compares actual and expected output. One important application of this program analyses is in competitive learning tools, used to manage programming contests, such as [8].

In [13] Wang et al. combine the two approaches: first the program undergoes static analysis, for compilation errors and to check similarity with "model programs". Then a dynamic testing is performed, and possibly the program adds to model programs.

Grading systems such as CMS are mainly based on dynamic testing, and are many: amongst them are those used in ACM International Collegiate Programming Contest (ICPC), i.e. the

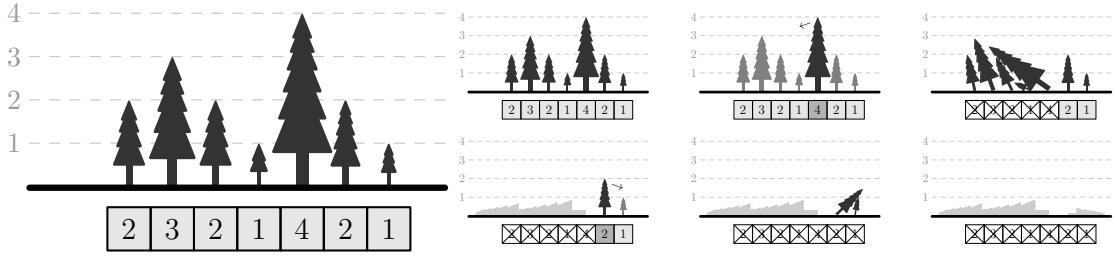


Fig. 1. The graphical representation of the task `taglialegna` (lumberjack), from OII 2014 final: an input instance (left) and a possible solution that uses two cuts (right).

proprietary Kattis[4], and the open source PC^2, available at `http://pc2.ecs.csus.edu/`. Other open source grading systems are Open Judge System[5] and DOMjudge[6].

If we focus on online training platforms, amongst several high quality ones are UVa Online Judge[7] and the more recent Sphere Online Judge[8] (SPOJ). Besides these training platform, there are several well known programming contests platforms, including Codeforces, USACO, COCI, TopCoder, Codechef, and Hackerearth, that run contests with different periodicity. There are also events based on programming contests, like the Google Code Jam and the Facebook Hacker Cup. A detailed survey of programming contests is in [2].

3 Italian Olympiads in Informatics

The International Olympiads in Informatics started in Bulgaria in 1989, patronized by Unesco. They are considered one of the most important programming competition in the world. Each country can have four contestants, and the competition is divided in two competition days. On each day contestants will be given three tasks to complete in five hours. Each task is worth 100 points and, since IOI 2010, is divided into subtasks, each worth a portion of the total points. There are time and memory limits for each subtask, and points are awarded only when all the tests in subtask yield correct results within the limits. There are also interactive tasks, like games, in which the contestant code alternates moves against an adversary.

In Fig. 1 we can see a graphical representation of a task, taken from OII 2014 final. The task, `taglialegna` (lumberjack), can be summarized in the following way: *there is a line of trees, with one meter of space between each of them. Each tree has a known height, in meters, and you can cut it aiming it toward its right or left. When an m meter tree falls, like in a domino game it forces the falling of its $m-1$ close trees, and this in turn can force other tree to fall. You can decide which tree to cut, and for each of them you can choose in which direction it will fall. What is the minimum number of trees to cut in order to remove all the trees in the line?* For this task, the subtasks were designed to distinguish algorithms of different computational costs: if we denote with n the number of trees in the line, all the points were awarded to a (definitely not trivial) $O(n)$ solution, achieved by only one contestant, and decreasing points were assigned, respectively, to $O(n \log n)$, $O(n^2)$, and $O(n^3)$ solutions.

[4] `https://kth.kattis.com/`
[5] `https://github.com/NikolayIT/OpenJudgeSystem`
[6] `http://www.domjudge.org/`
[7] `https://uva.onlinejudge.org/`
[8] `http://www.spoj.com/`

Italy participated in IOI for the first time in 2000, and since 2001 it started a national competition, promoted by a joint effort of the Italian Ministry of Education, University and Research (MIUR) and the Italian Association for Informatics and Automatic Calculus (AICA, a non-profit organization). The Italian Olympiads in Informatics (OII) are divided into three phases:

1. **First Selection (Schools, November)**: in this phase, in their own schools, approximately 20k students competes to solve, on paper, a test that involves math, logic, and programming abilities; in particular, there are some fragments of code (C/C++ or Pascal), and the students are asked to understand the behavior of the fragments.
2. **Second Selection (Regions, April)**: in this phase there are approximately 40 venues, where approximately 1200 students, selected from the previous phase, compete by solving three programming tasks on the computer. In this phase points are awarded for solving the tasks, independently from the complexity of the solution.
3. **Third Selection (National Final, September)**: approximately 100 students are asked to solve *efficiently* three programming tasks on the computer. They compete for 5 gold, 10 silver and 20 bronze medals.

From the above description it should be clear how the required programming abilities are varying through the different steps: we first ask students to be able to *read* code, then to *write* code, and finally *efficiently write* code. A more detailed picture of the OII organization is described in [1].

The selection process does not end with the national final: the gold and silver medal winners, together with at most five bronze medal winners, selected by (young) age, form the group of IOI-candidates, and four of them will represent Italy in the next IOI (usually held in July or August). Thus, there is almost one year to train and select them, and this process is mainly done in four stages held nearby Volterra[9]. In each of the stages there are theoretical lessons, ranging from traditional algorithms and data structures to competitive programming tips and tricks, as well as programming contests. Besides the stages, there is a continuos on-line support for self-improvement: the IOI-candidates are assisted by tutors (former IOI contestants) for assistance and guidance, and several training contest are organized, some of which focused on specific topics.

4 The online training system: `oii-web`

Our online training platform, `oii-web`, is based on the *Contest Management System* (CMS) [9, 10], the grading system used in several programming competitions, including IOI. CMS was designed and coded almost exclusively by three developers involved in the Italian Olympiads in Informatics: Italy hosted IOI 2012 and therefore, since 2010, it started the development of CMS, that was used/tested in the OII finals 2011 and, few month later, was the grading system of IOI 2012. CMS version 1.0 was released in March 2013, and since then has been used in both IOI 2013 and 2014, together with several other programming competitions in the world [10].

We began the development of `oii-web` during the preparation of the IOI-candidates for IOI 2012: why did we need an online training platform? The short answer is: in a programming competition there are very few (usually from 3 to 7) problems, to be solved in a short frame of time; in order to train the IOI-candidates we needed a system that allowed us to give them more problems they can solve whenever they want, so the first version of `oii-web` was simply an

[9] The small city of Volterra in Tuscany is nowadays world-wide popular due to the fact that in the novels and movies of the Twilight vampire saga it is the origin place of Volturi, "the largest and most powerful coven of vampires".

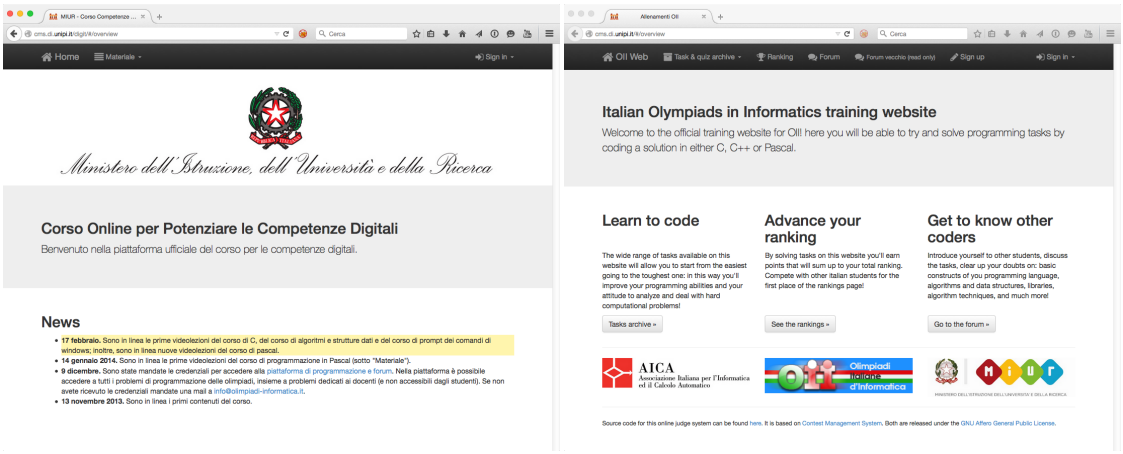


Fig. 2. The home pages of two of the platforms based on `oii-web`: the one for the teachers (left) and the one for the students (right)

instance of CMS with one competition running, with several problems and unlimited time. For training the IOI-candidates and, later, the two[10] Italian teams competing in IOI 2012.

Later, we started using it consistently, and our feature list was growing almost daily, both for the front-end and for the back-end of the system:

1. It would be nice to provide some information about each problem, so the student can choose it without reading the whole description.
2. It would be nice to have a way to exchange messages, so students and tutors can chat about the problems.
3. It would be nice to be have a way to show/hide problems, so we can use some of them in contests to rank the students.
4. It would be nice to have stats about each problem, and who was able to solve it (in a grading system there are these stats but not visible to contestants).

Thus, we decide to include all these above mentioned features, together with others, and build an online grading system, `oii-web`. We integrated the open source *Discourse*[11] to provide forum functionalities. The source code of the system is freely available[12] in github and it is released under the GNU Affero General Public License[13]. Furthermore, it is also available[14] as a "dockerized" app for docker[15], an open platform to build, ship, and run distributed applications.

5 The three platforms

In this section we briefly describes the three platforms, based on `oii-web`, we developed, and their differences.

[10] The nation that hosts IOI can have two teams of four elements, but only one team is eligible for medals.

[11] `www.discourse.com`

[12] `https://github.com/veluca93/oii-web/`

[13] `http://www.gnu.org/licenses/agpl`

[14] `https://registry.hub.docker.com/u/veluca93/oii-web/`

[15] `www.docker.com`

OII-training is the platform devoted to the students that are interested in OII. We can see a screenshot of the home page in Fig. 2 (left). In this platform there are approximately 180 problems spanning several techniques and difficulties, ranging from regional contests to IOI level. Furthermore, there are also the tests, from the first selection of OII (schools selection), available as interactive online forms. So far we did not advertise this platform in the schools, since we consider it in a beta testing phase. We allowed students to register freely, and so far we have approximately 1.500 users despite the lack of promotion.

DIGIT is the platform dedicated to teachers: we realized this platform in a project sponsored by the MIUR, where the aim was to build a self-paced online course of computer programming, focused on the olympiads in informatics. The idea was to train the teachers so they would have been able to train their students. Thus, this platform is currently the richest of the three, in terms of contents and functionalities. We can see a screenshot of the home page in Fig. 2 (right). There are video courses of C/C++ and Pascal programming, Algorithms and Data structures, and some basic video tutorial as well including how to use the platform to submit a solution. There are also some lecture notes, and all the material can be distributed to students as well; the video lectures are also available on the OII channel on youtube. The MIUR used this platform, since October 2013, in five distinct courses, with a sixth one scheduled to start in September 2016. So far approximately 3.000 teachers followed this course, and the effects on the OII were impressive: the participation of students in OII preliminary stages raised from 13k to 21k.

IOI-candidates is the last platform, and the only one not publicly available, since it is devoted to the IOI-candidates. This platform, as we mentioned before, has been the original motivation to develop the whole `oii-web`system. This platform has all the problems available to the other two platform, together with a *reserved* set of problems that we use in the contests to rank the students. The students are asked not to discuss these problem in public forum or social network, since we usually reuse them after few years.

6 Our experience

The advantages of a training system are clear: without it, we need to give students, besides the text of the problem: the input cases, the rules for counting the points, and, in some cases[16], a code to check the correctness of the produced output. And the student has to: run its code against every input, run the checker against each input and matching output, check the time and memory limits, that can be a cumbersome operation for a beginner. Furthermore, even if we automatize this task, for example by a script given to the student, there is still the problem of measuring the running times in different machines: students can have very different hardware and it is meaningless to state time limits without knowing their hardware.

Our path began, as we mentioned before, with the needs of a training system able to assist us with the preparation of italian IOI-candidates. We soon realized the advantages of such a system, as opposed to the use of an online platform like UVa Online Judge: we simply had more control, and this leads to a more effective teaching experience. We almost immediately decided to develop an online platform for the OII students as well, and we enriched our basic system with more features, in order to be able to deal with a much larger number of (averagely) less motivated students. In the beginning of 2013, the OII-training platform went online, in the form of a publicly available beta, as we were planning to add more features to make it more appealing

[16] To check the correctness of some problems is enough to check that the output produced by the student is the same as the output produced by the correct solution; in other cases, usually when there is more than a unique solution for a problem, like finding a path in a graph under some constraint, it is necessary to write a checker code that verifies the solution proposed for the given input.

for a larger audience. Almost concurrently, the MIUR asked us to design an online course for teachers, and we immediately decided to built it around our platform. So, in the next months, we adapted the system for the DIGIT platform, and realized the video lectures; in October 2013 we launched the first course: the MIUR opened a call for 250 teachers to be freely allowed to follow the course. The call was supposed to stay open for ten days, but we reached 250 teachers in the first day, and we decided to admit more. In subsequent courses, since we observed that the server was working fine, we raised the number of teachers per course to 700. At the end of each course there is a programming contest, and the ones that perform above a threshold (solving three problems out of seven) are awarded with a certification. Note that once a teacher has access to the platform, (s)he is allowed to use it also after the end of the course. Many teachers reported us that they had fun using the platform, and that they plan to keep on using it.

Our experience shows that the engagement in having or not a training system is completely different: we witnessed this at all the learners' levels; beginners were more involved, and advanced learners often joined the developers community (mostly made of tutors and former IOI contestants) to either contribute the system development or to propose new problems. The teachers were incredibly active in the forum, exchanging tips and solution strategies as well as mutual support. The IOI-candidates are literally eager to contribute to the system or in the design of new problems, maybe because they see the tutors as a model, or simply because they enjoy it so much that they want to be part of it.

We also asked the IOI-candidates to "adopt a past IOI problem": our goal is to have, in the system, all the problems from past IOIs, and therefore there is a (shrinking) list of the problems that need to be produced: in all the cases the text of the problem is available on the web, usually together with some solution, but we need to write input generators and fine-tune the time and space limits. Currently we have almost all the problems of the last ten IOIs.

7 Validation of the system: a descriptive analysis

In this section we discuss the results of a validation of the system; we performed our study by means of a survey technique, with a questionnaire as a tool. We focused on the users of the **OII-training** platform, and we report some stats in Table 1. With active user we denote a user that submitted at least one solution of a problem; with problem solved we denote the number of submission that completely solved a problem.

Number of registered users	1413
Number of active users in the period Jan. 2015 - May 2016	812
Number of active users Jan. 2016 - May 2016	399
Problems in the system	253
Problems solved by users in the period Jan. 2015 - May 2016	9754
Problems solved by users in the period Jan. 2016 - May 2016	6192
Average number of problems solved per user in the period Jan. 2015 - May 2016	≈12
Average number of problems solved per user in the period Jan. 2016 - May 2016	≈15,5

Table 1. Some statistics about the **OII-training** platform

.

We sent the users of the platform the link of the questionnaire in May 2016; we had 171 users that answered, and this means almost half of the current active users. In Appendix A we report the statistics of all the answers provided.

The experimental setup is based on the collection of general information about the responders, and on scales aiming at *Satisfaction*, *Usability*, *Effectiveness*, *Active learning*, *Fun*.

Where possible we used a Liket scale, with five grades: two highest, two lowest, and an intermediate one. This allowed to separate clearly *mainly positive* judgements from *mainly negative* ones. Exceptions (questions Q6 and Q12) are motivated by their, less progressive nature.

About the general satisfaction of the learners, we considered important the learner's feeling about the actual "learning results". In this respect it is quite satisfactory for us that 65% of the respondents selected mainly positive (the highest two) grades, while the mainly negative (lowest two grades) were chosen by a 6.5%.

Usability of the system was marked mainly positively by a 70%, with mainly negative scores below 4%.

Of course we were mainly interested in the effectiveness shown by the system, as witnessed by the higher number of questions dedicated to that topic. One main issue , in that respect, is the number of problems (meaning exercises) that the learner undertook/solved. A second issue regards the perception of the learner about having fruitful and not tiresome sessions of use of the system.

The first above issue is met by questions Q3 and Q4. As it was expectable, there are more exercises "tried" than "solved": the system is not a *panacea*. On the other hand, while only 1% of the respondent tried some exercises (probably between 1 and 5) with no success, data show that 56% of the students was able to give a try between 11 and 20 problems (one third of this share) or more than 20 (two thirds of them), suceeding in quite a respectable 47% of the whole sample. In this respect we notice that 3 learners out of 4 that tried more than 20 exercises, succeded in more than 20 exercises.

The second issue above (regarding fruitful and not tiresome sessions of work in the system), is cared by questions Q5 through Q8: to some extent Q7 helps focusing the result of Q5, while Q8 does the same for Q6. From Q5 and Q7 we clearly see that (at least the learner's perception of) fruitfulness is high, with mainly negative results below 6% and 10%, respectively for Q5 and Q7. Q7 was indeed useful in pointing out one crucial aspect of fruitfulness (comprehension): it's result sports a quite rewarding 61% of mainly positive marks. Finally, Q6 and Q8 tell us that there is scarce perception of a session of work in the system being tiresome or slow.

The approach to learning sought by the system is coherent with the concept of *active learning*, so we thought it would be interesting to probe the perception of learners in that respect. Question Q9 is quite direct in that respect: the results show mainly positive response (73%, equidistributed between the two highest marks). Questions Q10 and Q11 took a less direct route to the learner's attitude toward the system: the former question wanted to reveal the induced engagement, and scores almost 64% of mainly positive answers, while the mainly negative feedback is limited to 8%.

Question Q11 tried to connect the work in the system with the perceived gain on terms of problem solving skills. In this case we have still more than half the sample showing mainly positive response (56%), with 11% mainly negative and a third of the sample set on the intermediate grade.

Since the use of systems like ours is not usual in the Italian School, we liked the idea to fetch some reactions in relation to the "fun" factor. Such an investigation would be more proper in game based, or gamified, systems; however, since we actually plan to add gamified aspects (namely a *badge* feature), it seemed good to add a question whose response would be more useful in future comparisons.

With respect to the present state of the system, the results are surely quite good, with feelings of motivation (53%) and curiosity (24%) encompassing more than three quarters of the sample's feedback.

Satisfaction
Q1. Did you find the system useful to fulfill your learning goals?
Usability
Q2. Is the system simple to use?
Effectiveness
Q3. How many problems did you tackle in the system? Q4. How many problems were you able to solve satisfactorily in the system? Q5. According to your perception, your sessions using the system were fruitful? Q6. According to your perception, your sessions using the system were long enough (to be fruitful), but not too long (to be tiring)? Q7. By solving a problem, did you improve your comprehension of the algorithm or the technique involved? Q8. Is the system quick enough in providing response?
Active Learning
Q9. I felt active in approaching the problems with the aid of the system Q10. The study of algorithms and related techniques is more interesting with this approach Q11. Due to the interaction with the system I have identified and trained upon central issues in the problem solving activity, and important concepts in the solution of problems
Fun
Q12. Using the system I felt mainly: i) Motivated, ii) Happy iii) Curious, iv) Relaxed, v) Other
General questions
Q13. Which institution are you currently enrolled?
Suggestions
Q14. What would you improve in the platform? Q15. Other ideas or suggestions

Table 2. List of the questions we used in the evaluation of the system.

We conclude this section by discussing the suggestions we received in questions Q14 and Q15. Question Q14 was *What would you improve in the platform?*, and users were allowed to choose one or more of the proposed answer, that are the directions we are working on. We report below the results (that were not mutually exclusive, as the other questions), in order of the expressed preferences:

77.2% I would add a wiki with documentation about algorithms and related techniques.
57.3% I would add a system to help the user to choose the next problem to solve.
36.3% I would add a badge system, to show achievements using distinct badges.
12.3% I would add more problems involving Mojito, the JackRussell mascotte of OII.

From the first of the above results we gather the obvious: of course a repository of centralized information, about algorithms and techniques needed in the solution of the exercises, is highly attractive. Such a development is actually in our plans, needing basically quite a lot of wear and tear in order to structure and feed the wiki, and not much more in terms or research. Being it a wiki, however, we are planning to make it available to contributions coming from all the members, so to make of it another opportunity for social collaboration, and social-collaborative learning.

On the contrary, the second preferred choice (recommending system for the next_problem_to_solve, that would be based on the student model) is a topic for further research, met preliminary in the next section.

8 Further developments

A work of Wang et al. [13] states the following requirements for a comprehensive programma assessment system: 1) *sufficiently extended testing*, so to cover the various cases of computation; 2) *checking on the program structure*, to see that the problem specification is met, and no cunning shortcuts bring to the correct output; 3) *accepting and reasonably assessing programs with static errors* s; 4) *providing immediate and correcting feedback*.

We think that the developments in the `oii-web` system should ultimately fulfill these requirements, while the present directions should deal closely with the present purposes of the system, that is to allow non novice students to train for the contests. So here we try and define a model to support

- a static analysis stage where solution strategies (algorithm, data structure and their mutual feasibility) rather than syntactic/semantics errors, are considered,
- an interactive communication between system and student, to help
  - developing one's capability to select solution strategies, by giving feedback on the actual choice, and
  - planning a path of growth of one's skills, by suggestions about next suitable problems to undertake.
- (and dynamic testing in the usual form, as it is already done).

Problems (the exercises proposed in the various contests, yearly), *Solution*s (the programs proposed by the students), and ultimately the *Student*s are modeled basing on a tagging mechanism. Tags are the names of problems (P), the algorithms (A) and data structures (DS) usable and/or used in the solutions (S) of problems, the contests (C), and levels of confidence (L) in the use of combinations of As and DSs.

Teachers in charge of the organization of a contest are named *guru*s; students that came out to be "exemplary" in a contest, and so "whose choices can count" when a solution to a problem is to be assessed, are called *EP*s. EPs can be promoted as guru.

A problem is modeled as a family of strategy choices (A and DS), suitable for its solution. Differences in that suitability can be pointed out by a weight. The weight is computed basing on the frequency of that A/DS were chosen, and on the reputation of who performed that choice in the related contest. Notice that the reputation of t gurus and EPs is contextualized to the contest.

$$P = \{< A, DS, weight, contest >\}$$

A solution submitted by a student is modeled by the strategy chosen for it:

$$S = \{< person, P, A, DS, conteXt >\}$$

where conteXt is either a contest name or "training" (off contest).

This metadata is provided by the student, in order to allow for a timely feedback from the dynamic analysis. On the other hand that metadata might be inaccurate, so it is subject to scrutiny: when a check points out that the data was wrong, it is changed accordingly, or (in the extreme case) the solution is removed altogether.

This check is done by gurus. A more social kind of scrutiny has been devised, yet it can't be applied, as the students solutions submitted to the system are not to be shown in public, at least for the time being.

A submitted solution is statically checked by comparing its specification S with that of the problem P. A feedback can be then given, about the appropriateness of the choice, its present weight, and possible better weighted alternatives.

The lightweight student model we can define in this framework define the skills shown by the student while solving problems in the system; it is a collection of "acquirements" each one expressing the fact that a given problem has been solved, how, how well and in what conteXt

$$SM(person) = \{< P, A, DS, level, conteXt >\}$$

, where *level* is a discrete variable in $[1 \dots 5]$ associated to the outcome of the dynamic analysis of the solution (or solutions) submitted by the student on the problem.

The above modeling framework can be used during a contest, in order to collect problems models and data on students (for instance to compute students reputation and define the set of EPs for that contest).

However here we are also interested in the possible use of this framework in a social web-based settings, to foster training in view of a next contest. The CMS would be the place for such training, organized by the following protocol.

1. A *Target Skills* is given. This is a set of triples designating the aim of the the training (at this stage for all the system): $TS = \{< A, DS, level >\}$
2. The trainee can access the set of problems available from previous contest, her/his student model, and the TS.
3. While the trainee is entitled to select any problem and submit the related solution, the system can provide a list of suggestions, for "best next problems to undertake" in order to enhance SM(trainee) towards coverage of TS. This list is done by
   (a) defining the set of elements in TS that are close to be covered by tuples in SM (*Proximal Coverage* – PC);
   (b) defining a set of problems, whose undertaking can bring to add elements in PC to the student model.
4. Upon submission of a solution, the trainee provides its initial modeling ($< A, DS >$).
5. The dynamic analysis of the solution establishes the *level* value for the t-uple

   $$< P, A, DS, level, conteXt >$$

   going to join the student model.
   Notice that the trainee specification of the solution can be subject only to late evaluation (by guru), in order to allow for a timely feedback coming from the dynamic analysis. So the new element in the SM is *sub-judice* and it could be modified or, in the extreme cases, deleted.

9 Conclusions

In this paper we introduced `oii-web`, an online training system for programming contests. The system is based on CMS, the grading system used currently in IOI competitions and other programming contests as well. We developed three distinct platforms, based on `oii-web`, aimed at three distinct user sets: students enrolled in OII, their teachers, and IOI-candidates, i.e. the small set of students amongst which will be selected the four to represent Italy at IOI. We discussed briefly our experience, together with some current developments. We believe that, as

happened in our case, the use of such a system can contribute to spread the algorithmic problem solving skills needed in programming contests.

We also believe that this tool can scale up toward being an educational support to refining students skills in "algorithm mastery", and we have presented lines of development in that direction.

References

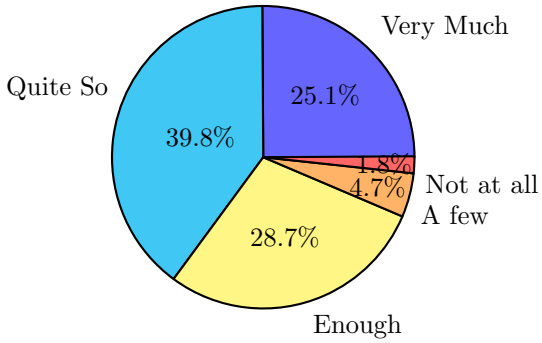
1. G. Casadei, B. Fadini, and M. Vita. Italian olympiads in informatics. *Olympiads in Informatics*, 1:24–30, 2007.
2. S. Combéfis and J. Wautelet. Programming trainings and informatics teaching through online contests. *Olympiads in Informatics*, page 21, 2014.
3. V. Dagienė. Sustaining informatics education by contests. In *Teaching Fundamentals Concepts of Informatics*, pages 1–12. Springer, 2010.
4. G. Garcia-Mateos and J. L. Fernandez-Aleman. Make learning fun with programming contests. In *Transactions on Edutainment II*, pages 246–257. Springer, 2009.
5. S. Halim and F. Halim. *Competitive Programming, Third Edition*. Lulu. com, 2013.
6. M. Hristova, A. Misra, M. Rutter, and R. Mercuri. Identifying and correcting java programming errors for introductory computer science students. In *Proc. SIGCSE 03, Feb. 19-23, 2003, Reno, Nevada, USA*, 2003.
7. A. Khirulnizam and J. Md. A review on the static analysis approach in the automated programming assessment systems. In *Proc. Nat. Conf. on Software Engineering and Computer Systems, Pahang, Malaysia*, 2007.
8. J. Leal and F. Silva. Mooshak: a web-based multi-site programming contest system. *Software: Practice and Experience*, 33:567–581, 2003.
9. S. Maggiolo and G. Mascellani. Introducing cms: a contest management system. *Olympiads in Informatics*, 6:86–99, 2012.
10. S. Maggiolo, G. Mascellani, and L. Wehrstedt. Cms: a growing grading system. *Olympiads in Informatics*, page 123, 2014.
11. K. Naudé, J. Greyling, and D. Vogts. Marking student programs using graph similarity. *Computers and Education*, 54:545–561, 2010.
12. S. S. Skiena and M. A. Revilla. *Programming challenges: The programming contest training manual*. Springer Science & Business Media, 2003.
13. T. Wang, P. Su, X.and Ma, Y. Wang, and K. Wang. Ability-training-oriented automated assessment in introductory programming course. *Computers and Education*, 56:220–226, 20011.
14. C. Watson, F. Li, and J. Godwin. Bluefix: Using crowd-sourced feedback to support programming students in error diagnosis and repair. In *Proc. Int. Conf. on Web-based Learning, ICWL 2012*, volume 7558 of *LNCS*, pages 228–239. Springer Verlag, 2012.

A Responses

In this section we present the results of the online evaluation form we asked the student to answer.

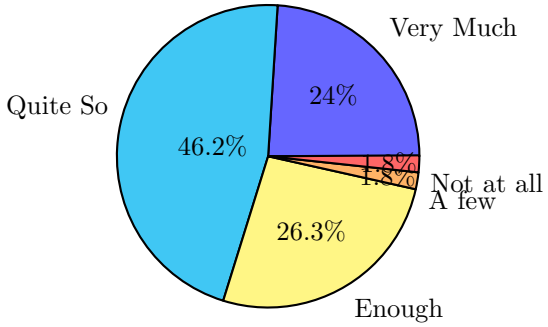
Satisfaction

Q1. Did you find the system useful to fulfill your learning goals?



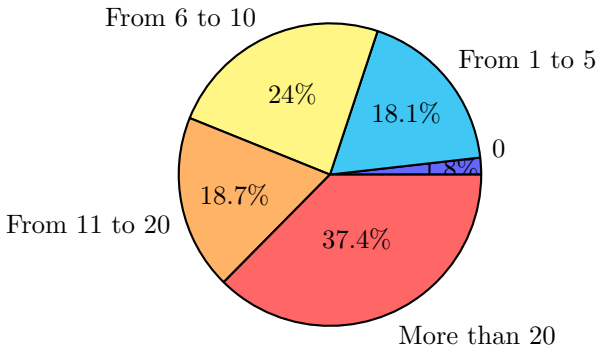
Usability

Q2. Is the system simple to use?

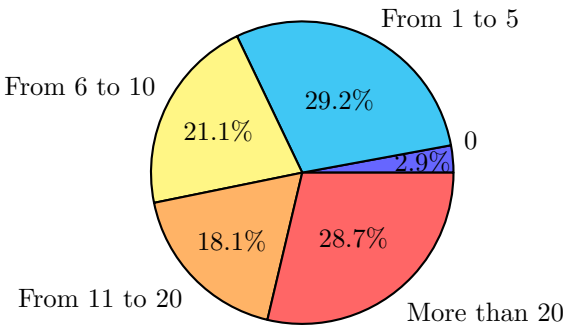


Effectiveness

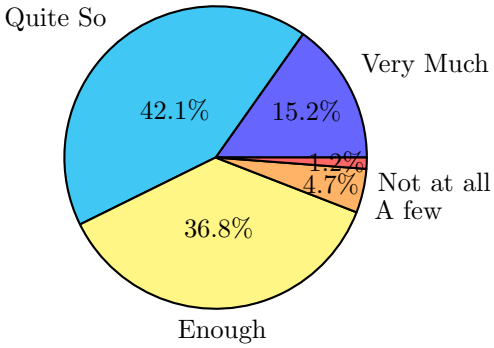
Q3. How many problems did you tackle in the system?



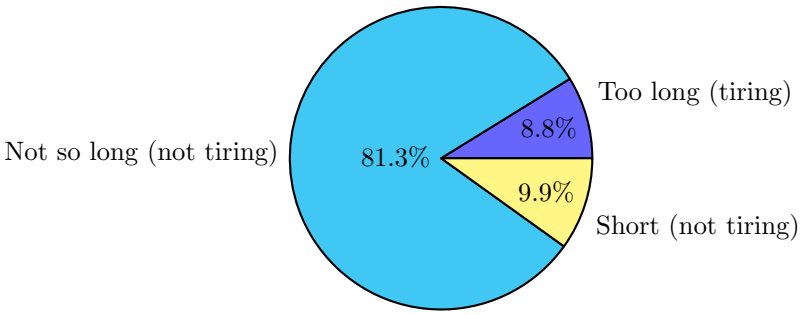
Q4. How many problems were you able to solve satisfactorily in the system?



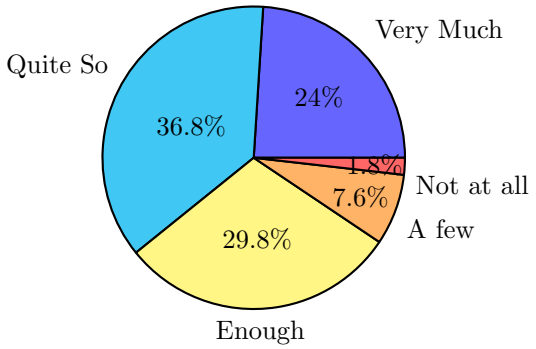
Q5. According to your perception, your sessions using the system were fruitful?



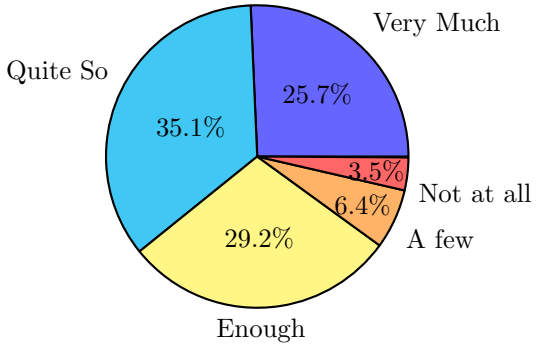
Q6. According to your perception, your sessions using the system were long enough (to be fruitful), but not too long (to be tiring)?



Q7. By solving a problem, did you improve your comprehension of the algorithm or the technique involved?

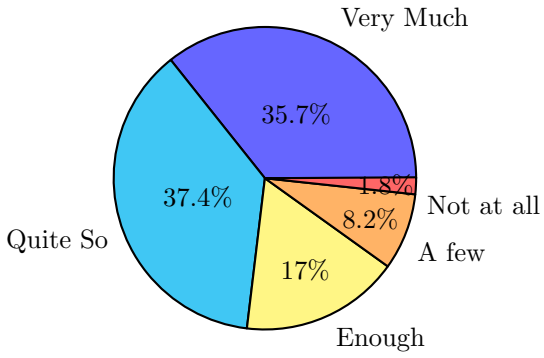


Q8. Is the system quick enough in providing response?

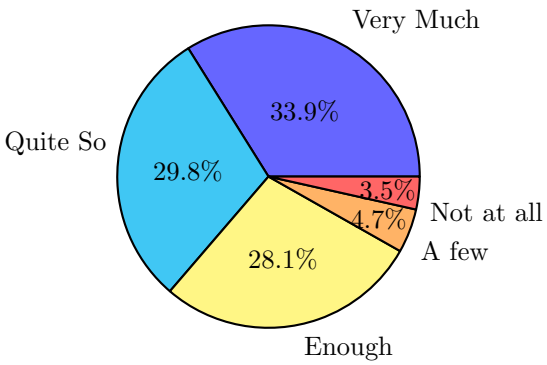


Active Learning

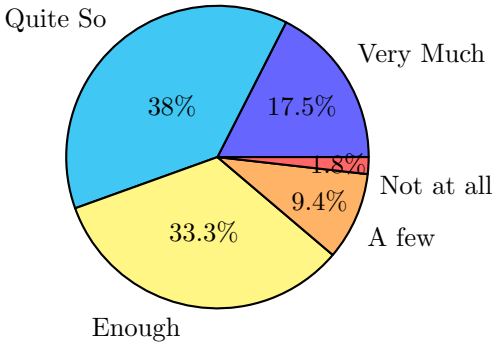
Q9. I felt active in approaching the problems with the aid of the system



Q10. The study of algorithms and related techniques is more interesting with this approach

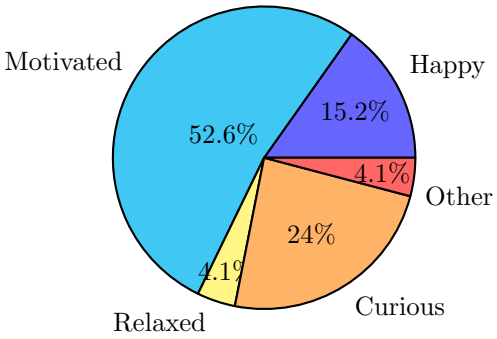


Q11. Due to the interaction with the system I have identified and trained upon central issues in the problem solving activity, and important concepts in the solution of problems



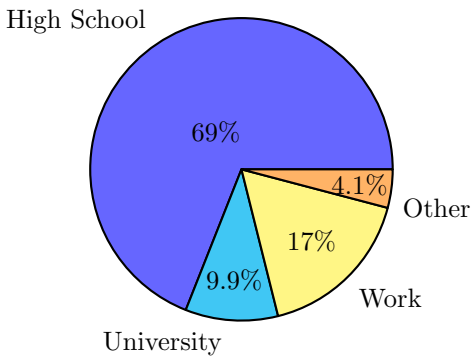
Fun

Q12. Using the system I felt mainly:



General questions

Q13. Which institution are you currently enrolled?


High School
69%
4.1%
Other
17%
9.9%
Work
University