European Journal of Engineering Education

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/ceee20

OJPOT: online judge & practice oriented teaching idea in programming courses

Gui Ping Wang[a], Shu Yu Chen[b], Xin Yang[c] & Rui Feng[d]

[a] College of Computer Science, Chongqing University, No. 174 Shazhengjie, Chongqing, Shapingba, 400044 PR China

[b] College of Software Engineering, Chongqing University, Chongqing, People's Republic of China

[c] College of Automation, Chongqing University, Chongqing, People's Republic of China

[d] Information School, Zhe Jiang University of Finance and Economy, Hangzhou, Zhejiang, People's Republic of China

Published online: 16 Jul 2015.

To cite this article: Gui Ping Wang, Shu Yu Chen, Xin Yang & Rui Feng (2015): OJPOT: online judge & practice oriented teaching idea in programming courses, European Journal of Engineering Education, DOI: 10.1080/03043797.2015.1056105

To link to this article: http://dx.doi.org/10.1080/03043797.2015.1056105

European Journal of Engineering Education, 2015
http://dx.doi.org/10.1080/03043797.2015.1056105

Taylor & Francis
Taylor & Francis Group

OJPOT: online judge & practice oriented teaching idea in programming courses

Gui Ping Wang[a*†], Shu Yu Chen[b], Xin Yang[c] and Rui Feng[d]

[a]*College of Computer Science, Chongqing University, No. 174 Shazhengjie, Chongqing, Shapingba, 400044 PR China;* [b]*College of Software Engineering, Chongqing University, Chongqing, People's Republic of China;* [c]*College of Automation, Chongqing University, Chongqing, People's Republic of China;* [d]*Information School, Zhe Jiang University of Finance and Economy, Hangzhou, Zhejiang, People's Republic of China*

(*Received 23 August 2013; accepted 26 April 2015*)

Practical abilities are important for students from majors including Computer Science and Engineering, and Electrical Engineering. Along with the popularity of ACM International Collegiate Programming Contest (ACM/ICPC) and other programming contests, online judge (OJ) websites achieve rapid development, thus providing a new kind of programming practice, i.e. online practice. Due to fair and timely feedback results from OJ websites, online practice outperforms traditional programming practice. In order to promote students' practical abilities in programming and algorithm designing, this article presents a novel teaching idea, online judge & practice oriented teaching (OJPOT). OJPOT is applied to *Programming Foundation* course. OJPOT cultivates students' practical abilities through various kinds of programming practice, such as programming contests, online practice and course project. To verify the effectiveness of this novel teaching idea, this study conducts empirical research. The experimental results show that OJPOT works effectively in enhancing students' practical abilities compared with the traditional teaching idea.

Keywords: programming practical ability; programming Contests; ACM/ICPC; online judge & practice; course project; empirical research

1. Introduction

Practical abilities, including programming and algorithm designing, are important for students from such majors as Computer Science and Engineering (CSE), Electrical Engineering (EE), Electrical Engineering and Computer Sciences (EECS) and Software Engineering (SE). Therefore, these majors offer some programming courses, including *Programming Foundation*, *Data Structures* and *Algorithm Design & Analysis*. These courses help students to develop practical abilities.

Programming Foundation course is offered to junior grades in universities using C, C++, Java or other programming languages. The purpose of this course is to instruct students in basic programming thoughts and methods, and to cultivate their elementary practical abilities in programming and algorithm designing. Yet these students usually do not have the basic knowledge

*Corresponding author. Email: w_guiping@163.com
†Present affiliation: College of Information Science and Engineering, Chongqing Jiaotong University, No. 66 Xuefu Road, Nan'an, Chongqing, 400074 PR China

of a programming language. Therefore, in the past teaching process, most emphases were usually put on grammatical knowledge of a programming language.

However, teachers find some issues during their experience in teaching. First, the grammatical system of a programming language is usually enormous since it contains many strict grammar rules. It is impossible to teach every aspect of grammatical knowledge in dozens of class hours. Second, the curriculum may be very boring if it contains much grammatical knowledge. Third, it is undoubtedly difficult for beginners to understand and master the grammatical knowledge during such a short period.

Furthermore, nowadays there are many available visualised development tools, which are usually referred to as integrated development environment (IDE). Many students put much enthusiasm in developing some simple application software once they come into contact with these IDEs. They neglect the training of thoughts and methods in programming, as well as abilities and awareness in algorithm designing and analysing.

These issues and phenomena lead teachers towards reflecting on some questions. For *Programming Foundation* course, which kind of main thread should be chosen to avoid boring grammatical knowledge but arouse students' interest in learning? Which kind of teaching idea should be adopted to guide the teaching process? Is there any kind of novel programming practice that can be introduced to stimulate students' enthusiasm in programming?

This article presents a novel contest-driven, online judge & practice oriented and course project enhanced teaching idea, which is termed OJPOT (online judge & practice oriented teaching). OJPOT is applied to *Programming Foundation* course. In order to verify the effectiveness of OJPOT, this article conducts empirical research, and then presents experimental results and discussion.

This article mainly aims at presenting a novel and systematic teaching idea, as well as verifying the effectiveness through empirical research. The research questions focus on whether (and in which aspects) OJPOT works effectively compared with the traditional teaching idea. As no similar systematic teaching idea is found in the literature, this article makes valuable contributions for the teaching and learning of programming courses.

The remainder of this article is organised as follows. Section 2 summarises related work. Section 3 presents OJPOT in detail. Section 4 conducts empirical research on OJPOT to verify its effectiveness. Lastly, Section 5 gives conclusions and looks into future work.

2. Related work

2.1. *Programming contests and ACM/ICPC*

During the last two decades, various programming contests have been carrying out in full swing all over the world. These contests include ACM/ICPC (ACM International Collegiate Programming Contest), TopCoder contests (TopCoder Collegiate Challenge, TCCC; TopCoder Open, TCO), Google Code Jam BaiDu Astar. The most popular one is ACM/ICPC. Its history can be dated back to a contest held by Texas A&M University in 1970. The first world final was held in 1977. The scale and impact of ACM/ICPC expand broader and broader year by year in many universities all over the world. Over 8000 teams from 2219 universities in 85 countries were attracted in the regional contests of the 36th ACM/ICPC. The top 112 teams selected from these regional contests competed for champion prize of the 36thAnnual ACM/ICPC World Finals on 17 May 2012 (data from: http://icpc.baylor.edu/).

This article mainly concerns ACM/ICPC due to its popularity. It presents a short description below about the types and features of problems, as well as evaluation methods in ACM/ICPC.

Table 1. The types and proportions of problems in ACM/ICPC.

Type of problems	Proportion
Enumeration	3%
Simulation	5%
Construction	5%
High-precision calculation	5%
Search	10%
Greedy	5%
Dynamic programming	15%
Graph theory	10%
Computational geometry	5%
Number theory	10%
Mathematical problem	10%
Data structure	5%
Others	12%

2.1.1. *Problem types in ACM/ICPC*

The problems in ACM/ICPC cover (a) basic algorithms, such as enumeration, simulation, construction, high-precision calculation and search; (b) greedy, dynamic programming and other optimisation algorithms; (c) basic algorithms in graph theory, computational geometry, number theory and other application fields. According to the authors' statistics, the types and proportions of problems in ACM/ICPC are shown in Table 1.

Some of these algorithms are simple enough for novices. The basic thoughts and methods adopted by such algorithms are fit to be taught in *Programming Foundation* course. After students master these thoughts and methods, as well as possess basic abilities in algorithm designing and analysing, it is relatively easier for them to learn more complex algorithms in subsequent courses.

2.1.2. *Features of problems in ACM/ICPC*

An ACM/ICPC problem typically contains the following five parts.

- *Problem description*: The problem description usually begins with a story or a game as background. Therefore, it is usually fussy, but interesting as well.
- *Input description*: The input description gives the specification of an input file, from which users' solution programmes read input data.
- *Output description*: Users' solution programmes should write results to an output file. The output description gives the contents and the formats of output data.
- *Sample input*.
- *Sample output*: Several sets of correct input and output data are provided to help users understand the problems, and test their solution programmes as well.

For each problem in ACM/ICPC, there are multiple sets of test data to be processed, which is an important feature for ACM/ICPC. This requirement has two purposes. First, these multiple sets of test data are used to test all possible situations and prevent cheating. Second, they are used to calculate running time of users' solution programmes and then test the strengths and weaknesses of the designed algorithms.

Compared with problems in ACM/ICPC, traditional exercises usually need to process only one set of test data. The insufficiency of one set of test data is that it is of great chanciness to determine whether the programme is right or not. However, when a student submits a solution

programme in ACM/ICPC, it is the duty of the online judge (OJ for short) system to determine whether the submitted programme is correct or not. Even the submitted solution programme has passed given sample input and output data, it is not always correct. In order to ensure the correctness of the designed programme, the student needs to introduce more test data and verify his programme offline. Therefore, these contests and problems can cultivate students' awareness of testing programme.

2.1.3. *Evaluation methods of ACM/ICPC*

At the server of ACM/ICPC, for each problem there are an input file with huge test datasets, and a standard output file gained from a standard solution programme based on the input file. These files are used to test the submitted programmes and usually capable of testing various special circumstances to be considered. The OJ system compares a user's output file with the standard output file character after character, and feeds back results to the user as soon as the comparison completes.

The problems in ACM/ICPC are very strict with the output requirements and formats. As long as the submitted solution programme considers insufficiently or produces output of incorrect format, it will not be accepted.

In addition, each problem in ACM/ICPC is subject to strict time and memory space constraints, which requires students considering time and space complexities when designing algorithms. Although students from junior grades do not have the ability of analysing algorithm complexities, this training process undoubtedly can develop their awareness of algorithm analysis.

2.1.4. *Popularity of ACM/ICPC and related research*

Along with the popularity of ACM/ICPC in China, provincial level (or even campus level) contests are popular among universities in China. For example, in the ninth Zhejiang Provincial Collegiate Programming Contests (7 April 2012), 298 teams selected from 79 universities competed for the final champion prize. These contests usually adopt the same rules as those in ACM/ICPC.

In recent years, some research efforts in the literature address strategies in programming contests. For example, Trotman and Handley (2008) show how to choose a good strategy during ACM/ICPC. ACM/ICPC and most of other programming contests are team based (usually no more than three students per team); Amraii (2007) summarises teamwork strategies in ACM/ICPC.

Test datasets of problems in contests are usually generated either by hand for small-sized tests or by programmes written by jury members that generate datasets according to some predefined patterns or at random (Buzdalov 2012). Generation of such datasets requires deep understanding of the programming task and its possible solutions. Therefore, the quality of the datasets relies very much on the human factor. Buzdalov (2012) adopts evolution strategy to automate this process of test datasets generation.

2.2. *Online judge websites and online practice*

In programming contests, the most important system is an automatic grading system, which can automatically and timely judge correctness or rate grade about submitted programmes. Kurnia, Lim, and Cheang (2011) first referred to these grading systems as *online judge*.

Wu, Chen, and Yang (2012) present the development and application of an OJ system. In order to enhance the performance of OJ systems in SMP (Symmetric Multiple Processor) environment,

Drung, Wang, and Guo (2011) introduce the affinity algorithm to improve the precision of user programmes' processing time. They further improve the OJ system's performance using queuing theory.

Along with the promotion of ACM/ICPC and other programming contests, a number of OJ websites come into being. The most famous OJ websites all over the world include ZOJ, POJ, UVA and Timus OJ.

Many researches have been done in utilising OJs to aid teaching. Aleman (2011) presents experience using automatic assessment in a programming tools course, which aims at extending the traditional use of an OJ system with a series of assignments related to programming tools. Luo, Wang, and Zhang (2008) develop a system called Programming Grid system, which is based on the OJ system of POJ and aims at computer-aided education for programming courses. Zhu and Fu (2012) propose a method to organise the programming resources on OJs automatically on a basis of a predefined hierarchical body of programming knowledge. Except course teaching, OJs are also used in academic tuition (Kosowski, Malafiejski, and Noinski 2007).

ACM/ICPC and other programming contests provide an opportunity for many enthusiastic programmers to demonstrate their abilities in analysing and solving problems. More importantly, the OJ websites, which emerge along with these contests, provide a practical platform for beginners to implement basic programming thoughts and methods. Therefore, OJ websites provide a new and novel practice, i.e. online practice, for programming courses.

Online practice is quite different from traditional programming practice. The mechanism of online practice is that: OJ websites provide problems, and students submit their solution programmes online; the judge systems in OJ websites evaluate students' solution programmes and return judgement results in real time. Most of the problems on OJ websites are collected from ACM/ICPC at various levels. Therefore, these problems are usually interesting and challenging. The judge process is fair and the results are fed back timely. Therefore, online practice can stir up students' interest in programming. Johnson (2008) systematically studies online practice, as well as designs and conducts questionnaires to verify the effectiveness of online practice.

In sum, there are a few research efforts in the literature addressing the utilisation and extension of OJs and online practice. Compared with existing researches, this article combines OJs and online practice with programming contests and traditional programming practice (e.g. course project) to form a novel and systematic teaching idea, i.e. OJPOT. To the best of our knowledge, no similar systematic teaching idea is proposed in the literature.

3. OJPOT: online judge & practice oriented teaching

3.1. *The main thread of the course*

The main thread designed for *Programming Foundation* course is that *the course should focus on programming thoughts and methods, and be supplemented by grammatical knowledge*. The underlying reasons can be summarised as two aspects.

First, programming thoughts and methods are embodied in specific application problems. In the real curriculum of OJPOT, interesting but rather challenging application problems are introduced to illustrate these thoughts and methods. Therefore, this main thread arouses the interest of students in understanding and mastering programming thoughts and methods. At the same time, this main thread avoids boring grammatical knowledge at a great extent.

Second, this course should systematically instruct basic programming thoughts and methods to students. The past teaching process occasionally taught few programming thoughts and methods along with much grammatical knowledge. But for beginners, occasional teaching does not truly attract their attention.

Based on this main thread, this article presents a novel OJPOT idea for *Programming Foundation* course.

OJPOT can be concreted as follows:

- *Contest driven*: During the teaching process, ACM/ICPC and other programming contests are adopted to stimulate students' interest in learning and consciousness in competition, as well as develop their initiative thinking skills.
- *Online judge & practice oriented*: Most of the examples taught in curriculum and exercises arranged after class are from several famous OJ websites, which benefits to develop students' abilities in problems analysing and solving, group discussion, teamwork, etc.
- *Course project enhanced*: At the final two weeks of the course, course project is disposed in grouping to strengthen teaching effectiveness.

3.2. *Detailed description of OJPOT*

3.2.1. *Contest driven*

Programming contests are useful for promoting students' interest in programming and enhancing their programming abilities (Almeida et al. 2012). During teachers' past teaching process, they found that most students still regarded course exam as the goal of a course and lost motivation in learning. On the contrary, during several years' contest-training work, the authors are deeply impressed by students' sense of achievement when they solve a problem independently and successfully. The sense of achievement is even higher than obtaining a high score in exams. However, it is difficult for students to gain this sense of success in past teaching process. In addition, during several years' contest-training work, the authors are shocked by junior grade students' enthusiasm in participating contests. The authors realise that programming contests bring great incentive for the learning of programming courses.

Therefore, it is necessary to introduce contest-training methods and evaluation rules into this course. Driven by ACM/ICPC and other programming contests, students' interest in learning and consciousness in competition are deeply stimulated.

3.2.2. *Online judge & practice oriented*

The traditional practice in *Programming Foundation* courses is usually arranged by teacher, as shown in the exercises and appendixes of a very popular traditional textbook edited by Tan (2011). Students are asked to write programmes to solve some problems. It is teacher's work to determine whether the submitted programmes are correct or not. This programming practice has two defects. The first one is that this practice is lack of incentive mechanism, thus it is difficult to stimulate students' interest in practice. The second one is that the evaluation results are subjective and not timely due to its requirement for manual judge, thus this practice cannot inspire students' enthusiasm in programming.

Online judge has a great potential for enhancing the teaching and learning process (Neilor et al. 2012). Due to fair and timely feedback results from OJ websites, online practice outperforms traditional programming practice. When submitting solution programmes through OJ websites, students can redesign their solutions based on the feedback results. Finally, after students repeatedly modify and resubmit their programmes, they gain Accept (solution programmes are correct and accepted by OJ). This process can train students' abilities in analysing and solving problems independently. Students gain tremendous sense of achievement when they solve a problem successfully. The effectiveness of online practice is also verified by Johnson (Johnson 2008).

3.2.3. *Course project enhanced*

In programming courses, the purpose of course project is to provide students with software development experience and engineering consciousness from junior grade.

The usual viewpoint is that what taught in *Programming Foundation* course are only grammatical knowledge of a programming language and basic programming methods. It is impossible for junior grade students to complete a course project of practical sense. However, the authors believe that for junior grade students they can complete a programme with 300–500 line codes through three persons/group teamwork in course project. In addition, those poor in programming are guided by excellent students through teamwork. Their learning motivation is stimulated greatly.

When designing and arranging a course project, two modes are adopted: students may choose to find a project task independently or accept teacher's arrangement. In fact, many ACM/ICPC problems are extracted from some classical games or specific applications. These problems are constructed by simplifying rules of games or requirements of applications. When students complete these problems, they will usually have their own ideas. They also initiatively think about the possibility to improve the rules or perfect the requirements. Through expansion of these problems and their solution programmes, they can usually find good course project tasks.

In addition, the contents of evaluating course project include completing a project programme, submitting a programme handbook and replying to course project. These evaluation contents provide students not only with training in software development and software engineering, but also with an earlier understanding for their future thesis reply. Furthermore, these evaluation contents can train students' abilities in written and verbal expression.

3.3. *Course contents*

In order to adapt to OJPOT, this article redesigns the contents of *Programming Foundation* course, including theoretical teaching and practical teaching.

3.3.1. *Theoretical contents*

Theoretical contents of this course based on OJPOT can be divided into three parts. The teaching contents arrangement and class hour allocation are shown in Table 2. For comparison, the traditional teaching contents of this course are listed in Table 3.

Part 1: Basic knowledge of a programming language. The basic knowledge of a programming language taught in this part is only the minimum set of grammatical knowledge required for writing a complete programme. When selecting the knowledge, the teachers conform to two principles. The first principle is that the knowledge used first should be taught in priority. The second one is that, for such grammatical knowledge only used in later chapters, the teachers put it into corresponding chapters as the relevant basic knowledge.

Part 2: Basic thoughts and methods in programming. This part is the emphasis of *Programming Foundation* course. The basic programming thoughts and methods taught in classes (as shown in Table 2) usually do not involve complex theoretical knowledge, thus there are fit to be taught in this course. After students master these thoughts and methods, they will soon be able to solve some exercises with different difficulties, and then submit their solutions through OJ websites. By using results fed back from OJ websites, they repeatedly modify the solution programmes until they gain Accept. During this process, students rapidly improve their abilities in analysing and solving problems independently. More importantly, when they solve each problem successfully, they gain strong sense of achievement.

Table 2. Teaching contents arrangement and class hour allocation in OJPOT.

Chapters	Contents	T	E
Part 1: Basic knowledge of a programming language	Basic framework of a programme, data types, operators and expressions, mathematical functions, algorithms and control structures, function design, arrays application, pointers	21	14
Part 2: Basic thoughts and methods in programming	Introduction of programming contests and online practice	3	2
	Enumeration	3	2
	Simulation	3	2
	Processing for character and string	6	4
	High-precision calculation	3	2
	Recursion	6	4
	Simple sorting and retrieval	3	2
Part 3: Course project	Course project case: developing a character-based mine-clearing game	6	4
Total		54	36

T: theoretical class hours. E: experimental class hours.

Table 3. Traditional teaching contents arrangements and class hour allocation.

Chapters	Contents	T	E
1	Elementary knowledge of a programming language	3	2
2	Data types, operators and expressions	6	4
3	Algorithms and control structures	15	10
4	Functions design	9	6
5	Arrays	9	6
6	Pointers	9	6
7	User defined data types	3	2
Total		54	36

Part 3: Course project. Course project is introduced after students learn basic grammatical knowledge, as well as master basic programming thoughts and methods. It aims at providing an opportunity for them to implement a small application software (even its interface is only character based). When they master IDE tools in subsequent courses, they can transplant their thoughts and methods in their course projects into visualised interface and develop some excellent application software.

3.3.2. *Practical contents*

Traditional practical teaching usually proceeds as follows: the teacher arranges programming tasks; the students finish these tasks individually and submit programmes; the teacher grades submitted programmes. In contrast, the practical contents of OJPOT are divided into three parts shown in Table 4, which are synchronised with theoretical contents.

Based on OJPOT, the authors compile a novel textbook (Wang et al. 2010), which is adopted in *Programming Foundation* course by several universities in China.

3.4. *Teaching methods*

Some effective teaching methods are summarised as follows.

(1) *Case-based teaching*. The case-based teaching is used to introduce programming methods or algorithm ideas through specific application problems. For boring grammatical knowledge and profound theoretical algorithm knowledge, case-based teaching is more effective.

Table 4. Practical teaching contents arrangements.

Practical teaching		Contents	Grouping	Evaluation contents
OJPOT	*Part 1: Basic knowledge of a programming language*	Imitating → rewriting → independent programming; analysing programme results → analysing programme implementation process → debugging programme	Individual	Submission of experimental reports
	Part 2: Basic thoughts and methods in programming	Discussing thoughts of algorithm in groups; completing the assigned exercises on OJ; preparing problem-solving reports.	3 persons/group	Submission of problem-solving reports and solution programmes
	Part 3: Course project	Completing a course project	3 persons/group	Submission of programme handbook and complete programme
	Traditional practical teaching	The teacher arranges programming tasks.	Individual	The teacher grades submitted programmes

(2) *Group discussion*. In experimental classes, the teachers adopt group discussion which is similar to ACM/ICPC. The teachers arrange students into groups to discuss algorithm thoughts in examples taught in classes or exercises they need to finish. They are asked to compile the discussion results into problem-solving reports.

(3) *Exercise report*. During the first half hour of each experimental class, the teacher arranges several students to report exercises they finished. Exercise report is evaluated and graded by the teacher and all of other students. This grading form and evaluation contents, as shown in Figure 1, can fully examine students' understanding of programming methods and algorithm thoughts in reported problems. The teacher's score and the mean score of other students are averaged, since the teacher and the other students have equal voice in the evaluation system.

Exercise report grading form

Name:_____ Problem no:_____ Problem title:_____

Evaluation aspects	Evaluation standard	Score
Courseware	Animation, graphics, contents. (20 points). Perfect, 19-20; Very good, 17-18; good, 15-16; general, 12-14; bad, <12.	
Problem analysis	Comprehensive analysis, easy to understand. (20 points). Perfect, 19-20; very good, 17-18; good, 15-16; general, 12-14; bad, <12.	
Code specifications	Code specifications, code submission. (20 points). Perfect, 19-20; very good, 17-18; good, 15-16; general, 12-14; bad, <12.	
Linguistic expression	(20 points). Very fluently, 19-20; relatively fluently, 17-18; good, 15-16; general, 12-14; poor, <12.	
Question and answer	(20 points). Quite right, 19-20; relatively correct, 17-18; good, 15-16; general, 12-14; poor, <12.	
Total	100 points	

Figure 1. The grading form of exercise report.

Course project grading form

Project title: _____

Team member: _____

Evaluation aspects	Evaluation standard	Score
Program handbook	Document contents, document layout. (30 points). Perfect, 27-30; very good, 24-26; good, 21-23; general, 18-20; does not meet the requirements <18.	
Novelty	(10 points). Very novel, 9-10; relatively novel, 7-8; general, 5-6; nothing new, <5.	
Independent creation	(10 points). Complete independent, 9-10; mostly independent, 7-8; partly independent, 5-6; non-independently, <5.	
Technology depth	Compact logical structure of the program, sophisticated program, fault-tolerant processing ability. (10 points). Very good, 9-10; good, 7-8; general, 5-6; poor, <5.	
Technology breadth	Some humanistic tips when running; rich in output contents, etc. (10 points). Very good, 9-10; good, 7-8; general, 5-6; poor, <5.	
Code specification	Code formatting; correct indentation, spaces and line breaks; more comments. (10 points). Very good, 9-10; good, 7-8; general, 5-6; poor, <5.	
Linguistic expression	(10 points). Very fluently, 9-0; relatively fluently, 7-8; general, 5-6; poor, <5.	
Question and answer	(10 points). Quite right, 9-10; relatively correct, 7-8; general, 5-6; poor, <5.	
Total	100 points	

Figure 2. The grading form of course project.

(4) *Course project reply*. After students finish their course project, the teacher randomly selects a student to reply the course project for each group. At the same time, the teacher and other groups of students evaluate and grade the replied course project. The grading form and evaluation contents, as shown in Figure 2, can fully examine students' comprehensive abilities to apply programming methods and algorithm thoughts during their course projects. Similarly, the teacher's score and the mean score of other groups of students are averaged.

4. Empirical Research

4.1. *Research questions*

In order to evaluate the effectiveness and validity of OJPOT compared with the traditional teaching idea, this article conducts empirical research. The formulated research questions are listed as follows:

(1) Does OJPOT work effectively to enhance students' practical abilities compared with the traditional teaching idea?
(2) In which aspects can OJPOT improve the students' practical abilities?

4.2. *Participants*

This article conducts two sets of experiments. The first set of experiments is conducted at Information School, Zhe Jiang University of Finance and Economy (ZUFE). The experiments last for a semester, i.e. the first semester of academic year 2010–2011. Two natural classes are chosen, and to rearrange them is inconvenience. One class (which contains 41 students) is defined as

the Control Class (CC). The other class (which contains 43 students) is defined as the Experimental Class (EC). In the CC, the traditional teaching idea is applied; while in the EC, OJPOT is applied. The students in both classes are freshmen. Before experiments, both classes are arranged to a pre-test to evaluate students' elementary knowledge in programming and C language. Then these two classes are taught by the same teacher with the same timetable (three theoretical class hours plus two experimental class hours per week) to avoid the variables which may affect the validity of the research. One of the most popular textbooks edited by Tan (2011) is adopted in the CC class; while a newly edited textbook (Wang et al. 2010) is adopted in the EC class.

The second set of experiments is conducted in the first semester of academic year 2011–2012 at College of Computer Science, Chong Qing University (CQU). The CC class and the EC class contain 34 students and 33 students, respectively. The other settings are the same as that in the first set of experiments.

4.3. *Instruments*

The experiments adopt the following instruments:

(1) *Pre-test* and *post-test*. In each set of experiments, both classes are arranged to a pre-test before experiments, which evaluate students' programming proficiency. At the end of the semester, both classes are arranged to a post-test, which helps to compare the effectiveness and validity of two teaching methods.
(2) *An OJ website*. The OJ website designed by ZUFE can record the following data of each user: the number of accepted solutions, the number of submitted solutions, a ratio between aforementioned two values (i.e. the AC ratio) and the number of code lines in each accepted solution, etc. These data are valuable to support the empirical research of this article.
(3) *Course project*. Course project is a further and centralised examination of teaching effectiveness. The score achieved by a student reflects the student's comprehensive practical abilities.
(4) *Statistical analysis software*: Statistical Package for the Social Sciences (SPSS).

4.4. *Experiment procedures and data collection*

Each set of experiments is conducted under the following procedures. Data are collected accordingly.

(1) *Pre-test and scores*. The participants are all freshmen. Most of participants have not been educated and trained in programming in senior high school, not ruling out very few students who have learned such knowledge before university entrance. In order to examine students' programming proficiency before experiments, a pre-test is arranged, which includes a written test and a machine test. The written test covers elementary knowledge in programming and C language (mainly including grammatical knowledge). In the machine test, the students are asked to write C programmes to solve some simple problems, e.g. outputting the sum of two given integers, solving the area of a triangle given three sides, determining a leap year. For pre-test and post-test, the problems in both the written test and the machine test are designed by two experienced teachers. Both the written test and the machine test are scored by these two experienced teachers. The final score of each participant is the mean value of the scores given by the two teachers to increase the scoring reliability. The scores are collected and analysed by SPSS.
(2) *Online practice and statistical data*. During the semester, in the CC and the EC, the traditional teaching idea and OJPOT are applied, respectively. However, both in the CC and

the EC, the students are encouraged but on their own to solve problems on the OJ website designed and maintained by ZUFE. The following statistical data of each participant are collected and analysed: the number of accepted solutions, the number of submitted solutions, the AC ratio, the number of code lines in each accepted solution, etc.

(3) *Post-test and scores*. At the end of the semester, both the CC and the EC are arranged to a post-test, which also includes a written test and a machine test. The written test mainly focuses on grammatical knowledge of C language; while the machine test mainly examines the programming abilities of the participants. Both the written test and the machine test are still scored by these two experienced teachers. The scores are collected and analysed. The scores in the post-test are compared with those in the pre-test to evaluate the effectiveness and validity of OJPOT and the traditional teaching idea.

(4) *Course project and scores*. At the final two weeks of the semester, course project is disposed in grouping both in the CC and the EC. The submitted project of each group is scored by these two experienced teachers. The score of each student in the group is given based on the student's contribution to the project, which is the combination of the contribution clearly stated in the programme handbook and the contribution observed by the teachers during two weeks' teamwork.

4.5. *Results and discussion*

4.5.1. *The set of experiments in ZUFE*

In Table 5, for the pre-test of the CC and the EC, the measures of analysing the mean scores and the Independent-Samples *T*-test from the SPSS are applied. Although the minimum, maximum and mean scores of the CC and the EC are a little different, there is no significant difference between the mean scores in pre-test before experiments, with the *P* value 0.931 and 0.946, respectively, in the written test and the machine test (If the *P* value is higher than 0.05, it implies that the difference is not significant; while if the *P* value is lower than 0.05, then the significant difference is suggested.). Therefore, the results in Table 5 suggest that these two classes have the similar programming foundation before experiments. Note that, in both classes, there are very few students who have learned such knowledge in senior high school and therefore gain high scores in the pre-test.

From the OJ website, the following four values of each student are recorded: the number of accepted solutions, the number of submitted solutions, the AC ratio and the number of code lines in each accepted solution (and therefore the total number of code lines in all of his accepted solutions). Table 6 shows the statistical data of online practice in these two classes at the end of the semester. From the results, it is clear that there is significant difference between the CC class and the EC class in all of these four values since the corresponding *P* values are all far lower than 0.05. For all the statistical data, the mean values in the EC class significantly transcend those in the CC class. Since the students participate in online practice on their own, the more problems the students solve on the OJ website, the higher enthusiasm they exhibit in online

Table 5. The independent-samples *T*-test results of the CC and the EC classes in pre-test (in ZUFE).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (written test)	41	2	86	14.12	19.75	0.931
The EC class (written test)	43	2	83	14.49	18.93	
The CC class (Machine test)	41	0	81	10.15	20.84	0.946
The EC class (Machine test)	43	0	79	10.44	19.33	

Table 6. The independent-samples *T*-test results of the CC and the EC classes in online practice (in ZUFE).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (# of accepted solutions)	41	0	49	16.10	11.71	0.0000024
The EC class (# of accepted solutions)	43	2	75	36.51	23.07	
The CC class (# of submitted solutions)	41	8	360	101	82.32	0.0016
The EC class (# of submitted solutions)	43	6	600	185.72	144.94	
The CC class (the AC ratio)	41	0	0.293	0.171	0.055	0.00075
The EC class (the AC ratio)	43	0.11	0.375	0.221	0.075	
The CC class (the total number of code lines)	41	0	5160	1362.07	1325.37	0.0001
The EC class (the total number of code lines)	43	54	7722	3044.20	2290.50	

Table 7. The independent-samples *T*-test results of the CC and the EC classes in post-test (in ZUFE).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (written test)	41	39	97	73.07	14.65	0.896
The EC class (written test)	43	37	96	72.70	11.48	
The CC class (machine test)	41	40	98	73.29	11.48	0.019
The EC class (machine test)	43	53	100	80.51	9.95	

Table 8. The independent-samples *T*-test results of the CC and the EC classes in course project (in ZUFE).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class	41	55	96	76.68	9.53	0.011
The EC class	43	60	98	81.79	8.53	

practice. Therefore, the above experimental results show that the students in the EC class exhibit higher enthusiasm in online practice than those in the CC class.

Table 7 shows the results in post-test. In the written test which mainly focuses on grammatical knowledge, although the mean value in the CC class is slightly higher than that in the EC class, but there is no significant difference between the mean values since the *P* value is 0.896. However, in the machine test which mainly examines the programming abilities, there is significant difference between the CC class (its mean value is 73.29) and the EC class (its mean value is 80.51) since the *P* value is only 0.019 (lower than 0.05). The experimental results in the machine test show that the practical abilities of students in the EC class are promoted more significantly than that of students in the CC class after a semester's teaching and learning.

Table 8 shows the results in course project which reflects comprehensive abilities. From these results, it is clear that there is significant difference between the CC class (its mean value is 76.68) and the EC class (its mean value is 81.79) since the *P* value is only 0.011 (lower than 0.05). These results show that the comprehensive abilities of students in the EC class are significantly higher than that of students in the CC class after a semester's teaching under different teaching ideas.

4.5.2. *The set of experiments in CQU*

The following tables illustrate the results in the set of experiments in CQU. Table 9 shows that there is no significant difference between the CC class and the EC class in pre-test, with the *P* value 0.851 and 0.941, respectively, in the written test and the machine test, which suggests that these two classes have the similar programming foundation before experiments.

Table 9. The independent-samples T-test results of the CC and the EC classes in pre-test (in CQU).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (written test)	34	6	90	23.71	20.96	0.851
The EC class (written test)	33	5	91	22.73	21.41	
The CC class (machine test)	34	0	89	17.62	24.91	0.941
The EC class (machine test)	33	0	87	17.18	22.94	

Table 10. The independent-samples T-test results of the CC and the EC classes in online practice (in CQU).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (# of accepted solutions)	34	0	48	18.94	14.68	0.00084
The EC class (# of accepted solutions)	33	1	79	36.21	24.61	
The CC class (# of submitted solutions)	34	0	307	113.32	85.38	0.034
The EC class (# of submitted solutions)	33	9	760	183.42	167.17	
The CC class (the AC ratio)	34	0	0.250	0.154	0.066	0.000035
The EC class (the AC ratio)	33	0.1	0.341	0.229	0.071	
The CC class (the total number of code lines)	34	0	6364	1386.2	1493.5	0.0027
The EC class (the total number of code lines)	33	41	10640	2961.6	2528.1	

Table 11. The independent-samples T-test results of the CC and the EC classes in post-test (in CQU).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class (written test)	34	44	97	75.03	14.00	0.906
The EC class (written test)	33	48	96	75.36	8.25	
The CC class (machine test)	34	45	98	76.21	9.12	0.016
The EC class (machine test)	33	53	98	82.67	11.16	

Table 12. The independent-samples T-test results of the CC and the EC classes in course project (in CQU).

Class	Number	Min.	Max.	Mean	Std. deviation	P (2-tailed)
The CC class	34	55	98	77.15	8.12	0.017
The EC class	33	58	100	82.61	9.97	

The online practice experimental results in Table 10 are similar with that in Table 6. From the results, it is clear that there is significant difference between the CC class and the EC class in all of these four values since the corresponding P values are all far lower than 0.05. The results indicate that the students in the EC class exhibit high enthusiasm in online practice compared with that in the CC class.

The post-test experimental results in Table 11 are similar with that in Table 7. In the written test, the mean value in the CC class is slightly lower than that in the EC class, but there is no significant difference between the mean values since the P value is 0.906. However, in the machine test, there is significant difference between the CC class (its mean value is 76.21) and the EC class (its mean value is 82.67) since the P value is only 0.016.

Table 12 shows the results in course project. Similarly, there is significant difference between the CC class (its mean value is 77.15) and the EC class (its mean value is 82.61) with the P value 0.017.

4.5.3. *Discussion*

The answers to the aforementioned two research questions are drawn from the above two sets of experiments.

(1) Compared with the traditional teaching idea, OJPOT does work effectively to enhance students' practical abilities. In both sets of experiments, the mean scores of the EC class in practical abilities tests (e.g. the machine tests in post-test in Tables 7 and 11, the course project in Tables 8 and 12) significantly transcend that of the CC class (the *P* values in these experiments are lower than 0.05).
(2) The aspects that OJPOT improves the students' practical abilities mainly lie in abundant and novel forms of programming practice (contests, online practice and course project). In our view, OJPOT stimulates the students' enthusiasm in programming, and spur the students participating in programming practice. As for online practice, compared with the students in the CC class, those in the EC class solve more problems in the OJ website (as shown in Tables 6 and 10), thus obtaining more programming training which is supported by their better mean scores.

4.5.4. *Limitations of the empirical research*

Although the empirical research validates the effectiveness of OJPOT, several limitations should be acknowledged.

First, since the CC class and the EC class are taught with different textbooks, whether this difference affects the experiments or not is not clear.

Second, both classes are taught by one teacher, so the reliability and validity of the empirical research may be affected by his (or her) individual understanding and executive force of OJPOT.

5. Conclusion and future work

The proposed OJPOT, which is a contest driven, online judge & practice oriented and course project enhanced teaching idea, is proposed during the authors' continuously exploring teaching practice in programming courses. In the actual teaching process, OJPOT shows its strength gradually.

In recent years, the authors also have extended OJPOT to some related courses, including *Data Structures* and *Graph Theory & Algorithms*. Empirical research on the extension of OJPOT to these related courses remains as future work.

We are convinced that, as long as teachers continuously arouse students' interest in learning and stimulate their enthusiasm in programming, students will better master programming courses. Students' innovation senses and abilities will be promoted greatly during their experience in enjoying successful learning.

Acknowledgments

The authors would like to thank the editors and the anonymous reviewers for their invaluable feedback.

Funding

The work of this article is supported by Zhejiang provincial Education Science annual planning project (Grant No. SCG156), and partially supported by National Natural Science Foundation of China (Grant No. 61272399).

References

ACM/ICPC. http://icpc.baylor.edu/
Aleman, J. L. F. 2011. "Automated Assessment in a Programming Tools Course." *IEEE Transaction on Education* 54(4): 576–581.
Almeida, F., J. Cuenca, R. F. Pascual, et al. 2012. "The Spanish Parallel Programming Contests and its use as an Educational Resource." In: *Proceedings of the 26th IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 1302–1305.
Amraii, S. A. 2007. "Observations on Teamwork Strategies in the ACM International Collegiate Programming Contest." *Crossroads* (the ACM student magazine) 14 (1): article no.: 9.
BaiDu Astar. http://star.baidu.com/
Buzdalov, M. 2012. "Generation of Tests for Programming Challenge Tasks on Graph Theory Using Evolution Strategy." In: *Proceedings of the 11th International Conference on Machine Learning and Applications (ICMLA)*, 62–65.
Drung, C., J. W. Wang, and N. Guo. 2011. "Enhance Performance of Program Automatic Online Judging Systems Using Affinity Algorithm and Queuing Theory in SMP Environment." In: *Proceedings of the 2011 International Conference on Electronic and Mechanical Engineering and Information Technology (EMEIT)*, 4425–4428.
Google Code Jam. http://code.google.com/codejam/
Johnson, G. M. 2008. "Online Study Tools: College Student Preference Versus Impact on Achievement." *Computers in Human Behavior* 24(3): 930–939.
Kosowski, A., M. Malafiejski, and T. Noinski. 2007. "Application of an Online Judge & Contester System in Academic Tuition." In: *Proceedings of theSixth International Conference on Web Based Learning (ICWL)*, 343–354.
Kurnia, A., A. Lim, and B. Cheang. 2011. "Online Judge." *Computers & Education* 36(4): 299–315.
Luo, Y. W., X. L. Wang, and Z. Y. Zhang. 2008. "Programming Grid: A Computer-aided Education System for Programming Courses Based on Online Judge." In: *Proceedings of the First ACM Summit on Computing Education in China (SCE)*.
Neilor, T. A., Z. A. Fabio, and J. B. Luca. 2012. "Enhancing Traditional Algorithms Classes Using URI Online Judge." In: *Proceedings of the IEEE International Conference on e-Learning and e-Technologies in Education (ICEEE)*, 110–113.
POJ. http://poj.org/
Tan, H. Q. 2011. *C++ Programming* (2nd ed., in Chinese). Beijing: Tsinghua University Press.
Timus Online Judge. http://acm.timus.ru/
TopCoder contests. http://www.topcoder.com/
Trotman, A., and C. Handley. 2008. "Programming Contest Strategy." *Computers & Education* 50(3): 821–837.
UVA. http://uva.onlinejudge.org/
Wang, Y., G. P. Wang, R. Feng, and X. Y. Ma. 2010. *Programming Methods and Guides to Online Program Practice (in Chinese)*. Hangzhou: Zhejiang University Press.
Wu, J. H., S. P. Chen, and R. R. Yang. 2012. "Development and Application of Online Judge System." In: *Proceedings of the International Symposium on Information Technologies in Medicine and Education (ITME)*, 83–86.
Zhu, G. J., and L. C. Fu. 2012. "Automatic Organization of Programming Resources on the Web." *Advances in CSIE* 1 (AISC 168): 675–681.
ZOJ. http://acm.zju.edu.cn/onlinejudge/

About the authors

Gui Ping Wang received his B.S. degree and M.S. degree in Chongqing University, PR China, at 2000 and 2003, respectively. Since July 2003, he acts as an assistant professor and an ACM/ICPC coach in Information School, Zhejiang University of Finance and Economy. He has nearly 10 years' teaching experience in such courses as Programming Foundation, Data Structures, Algorithm Design and Analysis, Graph theory and Algorithms, etc. Currently he is a Ph.D. candidate in College of Computer Science, Chongqing University. His research interests include fault diagnosis, dependability analysis and design of distributed systems, cloud computing, etc. As the first author, he has published nearly 20 papers in related research areas during recent years at journals.

Shu Yu Chen received his Ph.D. degree in Chongqing University, PR China, at 2001. Currently, he is a professor and the dean of college of Software Engineering at Chongqing University. He has over 25 years' teaching experience in programming courses. His research interests include embedded Linux system, distributed systems, cloud computing, etc. He has published over 120 journal and conference papers in related research areas during recent years.

Xin Yang received her B.S. degree, M.S. degree and Ph.D. degree in Chongqing University, PR China, at 2000, 2004 and 2008, respectively. Currently she is an associate professor in College of Automation, Chongqing University. She participates in the teaching practice of OJPOT since September, 2011.

Rui Feng is a professor and the associate dean of Information School, Zhejiang University of Finance and Economy. She has over 25 years' teaching experience in programming courses. She directs ACM/ICPC contest and participates the teaching practice of OJPOT in Zhejiang University of Finance and Economy since 2003.