A Survey on Online Judge Systems and Their Applications

SZYMON WASIK, * Poznan University of Technology and Polish Academy of Sciences
MACIEJ ANTCZAK,* JAN BADURA, ARTUR LASKOWSKI, and TOMASZ STERNAL,
Poznan University of Technology

Online judges are systems designed for the reliable evaluation of algorithm source code submitted by users, which is next compiled and tested in a homogeneous environment. Online judges are becoming popular in various applications. Thus, we would like to review the state of the art for these systems. We classify them according to their principal objectives into systems supporting organization of competitive programming contests, enhancing education and recruitment processes, facilitating the solving of data mining challenges, online compilers and development platforms integrated as components of other custom systems. Moreover, we introduce a formal definition of an online judge system and summarize the common evaluation methodology supported by such systems. Finally, we briefly discuss an Optil.io platform as an example of an online judge system, which has been proposed for the solving of complex optimization problems. We also analyze the competition results conducted using this platform. The competition proved that online judge systems, strengthened by crowdsourcing concepts, can be successfully applied to accurately and efficiently solve complex industrial- and science-driven challenges.

CCS Concepts: • **General and reference** → **Evaluation**; • **Mathematics of computing** → **Combinatorics**; *Discrete optimization*; • **Theory of computation** → **Design and analysis of algorithms**; • **Networks** → *Cloud computing*;

Additional Key Words and Phrases: Online judge, crowdsourcing, evaluation as a service, challenge, contest

ACM Reference format:
Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2018. A Survey on Online Judge Systems and Their Applications. *ACM Comput. Surv.* 51, 1, Article 3 (January 2018), 34 pages.
https://doi.org/10.1145/3143560

[*] S. Wasik and M. Antczak contributed equally to this work.

All authors were supported by the National Center for Research and Development, Poland [grant no. LIDER/004/103/L-5/13/NCBR/2014]. Moreover, development tools JIRA and Bitbucket were shared by PLGrid infrastrucutre. The authors would like to thank Szymon Acedanski from Warsaw University for his advice on the implementation of execution time measurements methods.

Authors' addresses: S. Wasik (corresponding author), Poznan University of Technology, Institute of Computing Science, Piotrowo 2, Poznan, 60-965, Poland, Polish Academy of Sciences, Institute of Bioorganic Chemistry, Z. Noskowskiego 12/14, Poznan, 61-704, Poland; email: szymon.wasik@cs.put.poznan.pl; M. Antczak, Poznan University of Technology, Institute of Computing Science, Piotrowo 2, Poznan, 60-965, Poland; email: maciej.antczak@cs.put.poznan.pl; J. Badura, Poznan University of Technology, Institute of Computing Science, Piotrowo 2, Poznan, 60-965, Poland; email: jan.badura@cs.put.poznan.pl; A. Laskowski, Poznan University of Technology, Institute of Computing Science, Piotrowo 2, Poznan, 60-965, Poland; email: artur.laskowski@cs.put.poznan.pl; T. Sternal, Poznan University of Technology, Institute of Computing Science, Piotrowo 2, Poznan, 60-965, Poland; email: tomasz.sternal@cs.put.poznan.pl.

ACM 0360-0300/2018/01-ART3 $15.00
https://doi.org/10.1145/3143560

1 INTRODUCTION

In 1970, when Texas A&M University organized the first edition of the ACM International Collegiate Programming Contest (ICPC), no one could have guessed that, in a few dozen years, it would be the largest and most prestigious programming contest in the world. In 2015, more than 40,000 students from almost 3,000 universities and 102 countries participated in a regional phase of this contest (40th Annual World Finals of the ACM ICPC 2016). During the contest, lasting 5 hours, participants solve from 8 to 13 algorithmic problems. The winner is the team that first solves the highest number of problems. The key component of this contest environment is a system that automatically verifies the correctness of solutions submitted by participants. It assesses the correctness of the submitted solutions based on the results obtained from their execution on predefined test sets. It also verifies that the solution does not exceed resource utilization limits (such as time and memory). Based on the conducted evaluation, the online ranking of all participants is computed and presented in real-time during the ongoing contest.

Since the first finals of the ICPC, many other algorithmic competitions requiring similar automatic evaluation systems have been started. Most likely, the most important of them is the International Olympiad in Informatics (IOI), first organized in 1989. It is comparable to the ICPC but dedicated to secondary school pupils. The biggest differences between the competitions are that, in the ICPC, the participants solve problems individually instead of collaboratively in teams; the number of problems considered by the IOI is lower than that by the ICPC; and, in the IOI, the participants receive partial scores for each submitted solution instead of the binary scoring applied in the ICPC (every solution is treated as only correct or incorrect). Detailed information regarding the basic rules and scoring functions used by the most popular programming competitions was published by Cormack et al. (2006).

The important supplemental role of the IOI is as a place for building an international community of people committed to the organization of programming contests. One of its important activities was foundation of the *Olympiads in Informatics* journal, which publishes papers submitted annually for presentation in the conference organized in parallel with the IOI (Dagiene et al. 2007). It provides an international forum for discussion regarding the experiences gained during national and international Olympiads in Informatics, including preparation of interesting problems and software supporting their organization. As mentioned before, the most important entries among such software are systems that automatically evaluate solutions submitted by participants; such systems are called *online judges*. The term *online judge* was introduced for the first time by Kurnia, Lim, and Cheang in 2001 as an online platform that supports fully automated, real-time evaluation of source code, binaries, or even textual output submitted by participants competing in a particular challenge (Kurnia et al. 2001). However, the development of online judge systems boasts a much longer history, dating back to 1961, when they emerged at Stanford University (Forsythe and Wirth 1965; Leal and Moreira 1998).

During the design and implementation of online judge systems, many important factors should be taken into consideration. The key issue is the security of such a system. The concept of an online judge assumes that the user submits the solution as the source code, or sometimes even the executable file, which will be evaluated in the next step, often in a cloud-based infrastructure. The designers of the online judge should ensure that the system is resistant to a broad range of attacks, such as forcing a high compilation time, modifying the testing environment, or accessing restricted resources during the solution evaluation process. A detailed description of possible attack types and the methods of protection used against them in 2006 during Slovakian programming competitions can be found in Forišek (2006b). Unfortunately, the solutions of security threats presented in the aforementioned paper are mostly outdated, although their causes are still unchanged.

Currently, the most popular methods of avoiding such issues rely on the execution of submitted solutions in dedicated sandboxes managed by the online judge system (Yi et al. 2014), such as virtualization, LXC containers (Felter et al. 2015), and the Docker framework (Merkel 2014). Such approaches could significantly increase the safety and reliability of the system.

Another important aspect that should be taken into consideration during the development of such systems is the measurement precision of execution time. The time limit for a single test case execution is often measured in milliseconds, and thus the performance analysis method used during evaluation should be sufficiently sensitive and deterministic to precisely distinguish such small fractions of time and ensure reproducible measurements of consecutive executions of the same code for the particular test case. There are various methods that can be used to measure the processing time, such as the utilization of simple command line utilities, analysis of hardware performance counters, code instrumentation, or even code sampling. Applications of each entail various advantages, as well as disadvantages, that one should be aware of related to measurement precision, time overhead, and available integration methods (Ilsche et al. 2015).

In all online judge systems, one of the requirements is that the solution code submitted by a user should be evaluated in a coherent and reliable server infrastructure. Hence, these systems are developed as Platform-as-a-Service (PaaS) cloud computing services because the scalability of such a highly interactive system is crucial, especially as the competition deadline approaches and the number of submissions increases rapidly. For this reason, the efficiency of such systems is often guaranteed by an architecture utilizing concurrency and parallel processing. A detailed analysis of the utilization of Symmetric Multiprocessing (SMP) environments by online judges is presented by Drung et al. (2011).

Finally, not only is the software itself important, but so are the description of the problem and the quality of the prepared test cases stored in the system. Forišek noted that authors of problems, which are evaluated automatically during competitions such as the ACM ICPC and the IOI, should give special attention to both the type of the problem and the preparation of test cases. He demonstrated that some types of problems, such as substring search, can be easily solved using heuristic algorithms. Additionally, he presented several problems from the ICPC and IOI competitions that can be solved using generally incorrect algorithms and still receive very good evaluation score (Forišek 2006a). Moreover, Mani et al. (2014) noticed that the manner of presentation of the online judge output is also very important. In particular, the evaluation summary presented in online judge systems that educational institutions use during courses should be easy to read and understand.

The problems that are usually published in online judge systems are generally classified as combinatorial problems. A combinatorial problem relies on discovering values of discrete variables that satisfy specific constraints. They are divided into decision problems, for which it must be verified that a provided solution exists, and search problems, for which the solution has to be found. A special case of the search problem is the optimization problem, for which we have to find the optimal solution subject to some objective function. Combinatorial problems are very interesting because they are intuitively understandable, but solving them is often challenging. The aforementioned observation was confirmed for various combinatorial problems and became a cause of the rise of computational complexity theory (Edmonds 1965; Garey and Johnson 1979). This research field focuses on the classification of such problems, taking into account the complexity of the process of searching for a solution. Problems published in online judge systems are usually solvable in polynomial time, and the maximal limit for processing time needed to find a solution is adjusted in such a way as to ensure that the optimal solution can be found for each considered test case. Although these systems can also be used to evaluate more complex problems for which only local optimums can be found during the processing time limit, none of the currently available online judges can simply manage such problems. The most well-known competition with the longest tradition is the ROADEF Challenge (Artigues et al. 2012), organized by the French Operational Research Society. The challenge is conducted every two years. Its primary objective relies on solving some industrial-driven problem proposed in collaboration with a business partner and supplemented with realistic test cases. There are also other specialized contests, usually focused on more specific topics; for example, the International Planning Competition (Coles et al. 2012) organized during the International Conference on Automated Planning and Scheduling (ICAPS) focuses on the implementation of planners that solve instances of planning problems defined in the Planning Domain Definition Language (PDDL) format (Mcdermott et al. 1998). Another example is the JILP Workshop on Computer Architecture Competitions, where participants solve problems related to optimizing parameters of algorithms used by computer hardware or infrastructure.

These contests apply a very efficient technique called crowdsourcing to solve practical optimization problems. Crowdsourcing outsources work to a large network of people in the form of an open call (Wasik et al. 2015). In the case of optimization challenges, the topic of the call is related to the optimization problem, and the network of potential participants includes interested programmers and scientists grouped together through the Internet. Although the concept of crowdsourcing has been intuitively applied since at least the eighteenth century (Dawson and Bynghall 2012), it has been formally defined relatively recently in 2006 by Jeff Howe (Howe 2006). The first 15 years of the twenty-first century have been a time of rapid development in crowdsourcing thanks to the popularization of Internet access. The best example of success in solving industrial challenges using crowdsourcing is the Kaggle platform. It is a web portal where data mining problems are published and subsequently solved by external experts participating in the competition. In 2016, Kaggle completed 34 competitions with total awards valued at US$1,160,000. The number of teams participating in the considered competitions varied between 50 and 3,500. Assuming that the average number of teams participating in a single contest is equal to 500, the average size of a team is 3, and average time spent solving the problem is up to 3 days per person thus yields 135,000 days or 370 person-years spent solving industrial problems.

The Kaggle platform focuses mainly on data-mining problems related to the concept of data science, which has recently gained popularity (Dhar 2013). However, solutions of various optimization problems can also be practically applied (e.g., in problems originating from the operational research field). Thus, there is a great need to develop efficient algorithms to solve them in safe and reliable environments by a resilient community. In this article, we would like to review the state of the art in the field of online judge systems and briefly discuss the Optil.io platform as an example of such systems, designed with the application of crowdsourcing in mind to solve complex optimization problems. This implements the online judge concept that utilizes an objective function instead of a binary evaluation to rank submitted solutions.

The scope of the article is as follows: As there is no formal definition of *online judge* systems in literature, we therefore decided to fill this gap to clarify further analysis. Next, we present a survey of existing online judge systems, prepared with a focus on potential applications, and we provide a short presentation of each of them. The diversity of these types of systems is so tremendous that the application-based classification is of great importance for interested practitioners. We focus mainly on systems that support an evaluation of algorithms for solving combinatorial problems. This is because, first, they are an elementary type of problems from the theoretical point of view (Garey and Johnson 1979); second, they include many data mining problems (Saedi and Kundakcioglu 2013); and, third, they are most commonly addressed by this kind of systems. Additionally, we summarize the common evaluation methodology supported by these systems. Finally, we briefly demonstrate potential applications of the online judge systems based on the discussion of the contest conducted using the Optil.io platform. A brief discussion of the results of the application of this system allows for a broader understanding not only of the concepts described in this article, but also of their practical implications to the design and development of online judge systems. We believe that such a review, supplemented with an example application of an online judge system, can be interesting for many scientists who want to use online judge systems in their own research or for teachers who would like to use them for educational purposes.

2 ONLINE JUDGE SYSTEMS

In general, the aim of online judge systems is a safe, reliable, and continuous cloud-based evaluation of algorithms that are submitted by users distributed around the world. Toward a better understanding of the scope of this article, we will first define an evaluation procedure, which is crucial and should be implemented, at least partially, by any online judge.

Definition 2.1 (Evaluation Procedure). An evaluation procedure consists of three steps: (1) submission, (2) assessment, (3) scoring.

During the submission phase, the submitted code is compiled, if needed, and verified if it can be successfully executed in the homogeneous evaluation environment. After a successful verification, each submission is reliably assessed on coherent infrastructure based on the problem-specific set of test cases. For each test case execution of the particular submission, it is verified whether:

(1) the execution process proceeded without errors,
(2) any problem-specific resource limitations have not been exceeded,
(3) the obtained output complies with the rules described in the problem definition.

Finally, the aggregated score for the submission is computed based on the results of all considered test cases. The detailed definition of the commonly used evaluation procedure is presented in Section 3. We treat various online platforms as an online judge in a broad sense, including all systems that support any subset of evaluation procedure phases often found in a cloud-based environment. A formal definition of online judge is provided in Section 3 in Definition 3.6.

2.1 Methods

In the literature, one can find various attempts at classification of contests organized using online judge systems and problems solved in such environments. In 2006, Pohl proposed a first simple classification (Pohl 2006) taking into account criteria related to a contest's style, duration, grading, submission procedure, and entrance criteria. In 2014, Combéfis and Wautelet proposed another classification (Combéfis and Wautelet 2014) according to criteria focusing on the programming contests and the problems solved during these matches. In 2015, Németh et al. refined this classification with several new criteria, characterizing contests more deeply and describing types of programming exercises and features of online judge systems crucial to the educational perspective (Németh and László 2015). However, all these reviews had a scope limited to a single application, either in education or in the organization of programming contests. Until now, no one has classified online judge systems according to their potential applications, taking into account a much broader range of interests of potential users. Such classification should present the primary objective of the system and can be extremely useful for users looking for an online judge system that meets their needs. That is why we decided to treat this criterion as crucial to the differentiation of online judge systems.

According to this criterion, we distinguished six classes of systems that integrate online judge systems. The largest class represents online platforms dedicated to the sharing of challenges solved during programming contests (ACM ICPC-like) and Olympiads in Informatics. The other categories of systems are helpful for educational purposes, recruitment of employees, evaluation of specialized algorithms that solve data mining problems, and integration as a crucial component of online compilers that allow users to compile arbitrary code online. Finally, we present the class of development platforms that provide an online judge component that can be simply used in custom applications. In some cases, it was difficult to apply such a classification because there are several systems that can be considered members of more than one group. In such cases, we identified the principal objective of the service to classify it.

The objective of the article is to provide a tutorial demonstrating a broad range of online judge systems; it was prepared based on an extensive review of literature conducted using the Web of Science, Google Scholar, and Scopus Indexes. Moreover, we reviewed in detail the vast majority of articles published in *Olympiads in Informatics* journal and queried the Internet to find such systems that are available even if they have not yet been published. For all queries, we used the following set of keywords: *online judge*, *online judge systems*, *automated programs grading*, and *automated grading programming*. The area of students' assignments assessment has been quite extensively explored in many well-known review papers where such systems are described in the light of potential applications, such as teaching use cases (Caiza and del Álamo Ramiro 2013; Ihantola et al. 2010, 2015; Romli et al. 2010; Staubitz et al. 2015; Wilcox 2016). To ensure completeness of the possible applications of online judges in the proposed review, we selected systems that are the most comprehensive, from the applications-oriented perspective, as representatives in this category.

Following the guidelines for preparing systematic literature reviews (Keele University 2007; Kitchenham et al. 2009), we defined the following acceptance criteria, all of which should be satisfied by the system considered in the review; that is, the system:

(1) supports any subset of evaluation procedure phases,
(2) is able to evaluate algorithms that are used to solve combinatorial problems,
(3) is available in English, or is described in an article published in English in some journal or conference proceedings,
(4) is publicly available (for free or on commercial basis),
(5) operates properly (i.e., it should have provided a possibility to register and submit solutions for at least one problem provided through it in June 2017).

A classification of online judge systems presented in this article has been prepared with the main focus on the summary of their possible applications from the user's perspective. For each online judge system that we found in the literature, we selected two to five of the most intuitive usage scenarios. Afterward, for each usage scenario identified earlier, we computed a simple coverage coefficient as the number of online judge systems that implement such a scenario. Thus, we obtained a list of usage scenarios ordered in a decreasing manner according to the coverage coefficient. During the team brainstorming session, we discussed the significance and usability of the considered usage scenarios. Eventually, we defined the following categories of online judge systems that differ in supported functionalities:

(1) *Online compilers*: Systems that support only a compilation of source codes performed during a submission phase of the evaluation procedure.
(2) *Data mining, education, and competitive programming*: Systems that, by inferring from our functionality-oriented analysis, focus on (a) evaluation of solutions submitted for data mining problems are classified in a data mining category; (b) educational processes are considered as educational software; (c) implementation of ICPC or IOI scoring rules and archiving problems from past challenges are classified as competitive programming software.

(3) *Recruitment platforms*: Online platforms targeted to employers to facilitate the recruitment of software developers for various IT projects and which allow the evaluation and ranking of submitted solutions according to specific objective functions.
(4) *Development platforms*: Systems that are often provided as open source projects or binary archives, allowing users to customize deployments and configurations of these systems in their infrastructure.

For each online judge system, we provide the most crucial information; namely, if it is open source, what natural language it uses in its user interface, number of supported compilers, number of published challenges, number of registered users, year of establishment, and if it is actively maintained. We included only systems that are available online and working correctly. We classified as inactive systems that had no new problems or news published on their websites over 2016. Otherwise, a system is classified as active. When we could not locate information on the web page, we tried to contact administrators. The number of compilers supported by many platforms is changing very quickly because it is rather easy to extend the range of supported compilers. Thus, we assigned the considered systems into four classes: Systems that do not support any compilers because they only assess the results obtained by algorithms executed by the user using his infrastructure (Class 0); systems that support a single specific programming language (Class 1); systems that support several of the most popular programming languages, such as C++, Java, Python, and C# (Class 2); and systems that support a broad range of compilers (Class 3). For development platforms and online compilers, the assessment of the number of challenges published is not applicable because their business goal is to be integrated as part of more complex systems and to compile code that can be executed on arbitrary data. A summary of all this information is presented in Tables 1 to 6.

2.2 Competitive Programming

Online platforms that collect and share challenges similar to those used during competitive programming contests (Khera et al. 1993) constitute the largest group of services that integrate online judge systems because many universities provide this type of system to help their students prepare for competitive programming championships. Moreover, organizations that conduct such competitions are interested in the popularization of challenges solved during the past editions of these contests. The list of online judge systems presented in this subsection can be found in Table 1.

The first online judge system that gained high popularity worldwide is the **UVa Online Judge** (Revilla et al. 2008). It was founded in 1995 by Miguel Ángel Revilla, a mathematician who lectures on algorithms at the University of Valladolid in Spain. It provides an enormous archive of programming challenges originating from the ACM contests. Inspired by the massive dataset collected by UVa, Skiena and Revilla wrote their first book, the objective of which was primarily to help students prepare for team programming competitions (Skiena and Revilla 2008). In that volume, they presented a large subset of challenges, followed by their most compelling solutions and hints that can be used by readers to solve them themselves.

Currently, the most extensive online judge, where more than 10,000 challenges have been published, is the **National Tsing Hua University Online Judge**. It is an ACM ICPC-like online judge designed for training purposes, as well as being the online platform often used for organization of programming contests. It supports only the C and C++ programming languages. In turn, one of the most popular online judges is **Codeforces**. It organizes contests on a regular basis. Moreover, it provides independent instances prepared for Russians only. It divides participants into two divisions according to their adaptation of the Elo rating (Elo 1978), which is updated for every competition conducted. During the contest, they randomly split contestants into rooms of approximately

Table 1. Online Judge Systems Applied for Competitive Programming Contests

Name and URL Address	OSS	GUI Language	Compilers Class id	#Problems	#Users	Established	Active
A2 Online Judge	No	Eng	3	300	55000	2011	Yes
AC 2333	No	Chi	2	670	3300	2011	Yes
AcDream	No	Eng,Chi	2	300	5200	2013	Yes
ACM-ICPC live archive	No	Eng	3	1000	52000	2003	Yes
ACM-Kyrgyzstan Subregion	No	Eng	3	422	3600	2005	Yes
Adjule Online Judge	No	Pol	2	120	3000	2011	Yes
Aizu Online Judge	No	Eng,Jpn	3	1000	36000	2004	Yes
Al Zimmermann's Programming Contests	No	Eng	0	26	2000	2009	Yes
BNU OJ	Yes	Eng,Chi	3	51000	31000	2013	Yes
Carribean Online Judge	No	Eng,Spa	3	2700	28000	2010	Yes
CDOJ	Yes	Eng,Chi	2	1300	9600	2014	Yes
Codeforces	No	Eng,Rus	3	3000	32500	2010	Yes
Don Mills Online Judge	Yes	Eng	3	700	7700	2014	Yes
e-olymp	No	Eng	3	7500	47000	2006	Yes
EI Judge	No	Eng,Rus	3	400	20000	2003	Yes
Facebook Hacker Cup	No	Eng	0	N/A	80000	2011	Yes
Fuzhou University Online Judge	No	Eng,Chi	2	1300	34000	2008	Yes
Google Code Jam	No	Eng	0	450	200000	2008	Yes
Herbert Online Judge	No	Eng,Chi	1	1761	1200	2010	2011
HIT ACM/ICPC	No	Eng	2	1300	54000	1998	Yes
HUSTOJ	Yes	Eng,Chi	2	650	26000	2014	Yes
Indian Computing Olympiad Problems Archive	No	Eng	2	45	1250	2003	Yes
Internet Problem Solving Contest	No	Eng	0	240	5000	1999	Yes
Light OJ	No	Eng	2	430	14000	2012	Yes
LYDSY	No	Chi	2	1000	30000	2008	Yes
Main	No	Eng,Pol	1	1000	33000	2005	Yes
National Taiwan University Online Judge	No	Chi	2	2600	600	2016	Yes
National Tsing Hua University Online Judge	No	Eng	1	10000	-	2015	Yes
North University of China Online Judge	No	Eng	1	2000	4000	2006	Yes
P3G	No	Eng	3	1100	500	2008	Yes
Peking University Judge Online	No	Eng	2	3000	250000	2003	Yes
Petrozavodsk State University	No	Eng,Rus	2	450	140	2010	Yes
Project Euler	No	Eng	0	550	650000	2001	Yes
SPOJ	No	Eng	3	6000	60000	2004	Yes
SPOJ PL	No	Pol	3	800	36000	2004	Yes
Szkopuł	Yes	Pol	1	1000	-	2012	Yes
Teddy Online Judge	Yes	Spa	3	250	1900	2009	Yes
Timus Online Judge	No	Eng	3	1000	110000	2000	Yes
TJU ACM-ICPC Online Judge	No	Eng,Chi	2	3000	52000	2005	Yes
TopCoder Competitive Programming	No	Eng	2	5200	4000	2001	Yes

(Continued)

Table 1. Continued

Name and URL Address	OSS	GUI Language	Compilers Class id	#Problems	#Users	Established	Active
USA Computing Olympiad	No	Eng	2	150	12000	2013	Yes
UVa Online Judge	No	Eng	2	5000	250000	1995	Yes

Successive columns include: (1) the name and URL address available in electronic version when hovering the computer mouse over its name (to print URL addresses please download the electronic supplement), (2) if this system is Open Source Software (OSS), (3) the ISO 639-2/B Code (LLC Books 2010) of languages supported by the user interface of the system, (4) the class regarding the number of supported compilers, (5) the number of published problems, (6) the number of registered users, (7) the year of establishment, and (8) the date of the last update of the system. All data were collected in June 2017.

40 people. A single round takes 2 hours and considers five programming challenges. After final submission of the solution for a given challenge, the judge allows each contestant to inspect other participants' solutions to *hack* them. By *hack*, we mean that Codeforces allows the participants to verify the code submitted by rivals on an instance of their choice. When the verification fails, the system automatically rewards the contestant who demonstrated the failure of the submitted solution with additional points, and the author of the *hacked* solution receives a penalty.

Three other large online platforms with more than 5,000 published problems are **Sphere Online Judge** (**SPOJ**), **E-Olymp,** and **TopCoder Competitive Programming**. The first (Kosowski et al. 2007) is a very popular system that offers various types of challenges, ranging from classical to optimization, code-golf, and riddles. It also provides two independent instances of the system dedicated to Polish and Brazilian users. Each of these instances includes separate challenges. In turn, E-Olymp is a Ukrainian portal supporting national institutions responsible for teaching gifted young people who participate in programming competitions worldwide. Finally, TopCoder provides a very innovative way of conducting algorithmic contests. TopCoder organizes short rounds, with three challenges and a 75-minute time limit per round. For each submitted solution that proves correct, the authors receive a reward inversely proportional to the time that they needed to solve it and an additional bonus when they identify an issue in an opponent's code, similarly to Codeforces. TopCoder is a service with a very long tradition and has already organized more than a thousand algorithmic matches.

There are also plenty of less-popular online judges maintained by various universities. For example, **Tianjin University's Online Judge** (**TJU**) provides an extensive database of challenges that can be used to conduct contests that are custom, open, or addressed to a limited set of participants free of charge. These are so-called virtual contests. The organizer of such a contest is responsible for choosing its start and end date and selecting from the provided database the set of challenges that should be solved during this contest. Next, the system hosts the contest, following the rules defined for ACM ICPC contests. In this context, it is also worth mentioning **Peking University** (**PKU**) **Judge Online** (**POJ**) and **Timus Online Judge**, which are the largest online judge systems located in China (Wen-xin and Wei 2005) and Russia, respectively. **EI Judge** is a relatively small system of that type in comparison with its predecessors and is hosted by the Moscow Institute of Physics and Technology. In turn, **AC 2333** is the Ningbo University of Technology's online judge system, which only operates in China. The first Polish online judge in this grid is the **Adjule** system hosted by Adam Mickiewicz University, which mainly shares challenges proposed for the annual contest called the Poznan Open Championships. The following systems, namely those of Petrozavodsk State University (which hosts the **PetrSU Programming Club**) and **National Taiwan University**, are used to organize programming contests for their students. In contrast to the systems just mentioned, **P3G** is a resource for competitive programmers worldwide and was created by members of Woburn Collegiate Institute's Programming Enrichment Group. Creators of this platform also run **PEGWiki**, which is a site prepared for algorithm enthusiasts. Other universities that boast their own online judges include **Fuzhou University** (China), **Harbin University of Technology** (China), **University of Electronic Science and Technology of China**, **North University of China**, University of Information Science in Cuba (hosting **Caribbean Online Judge**), and Huazhong University of Science and Technology in China (hosting **HUSTOJ**).

Moreover, many organizations that host programming competitions are also interested in managing online judge systems to popularize challenges solved in past events and help users prepare for upcoming ones. One such system is the USA Computing Olympiad (**USACO**) platform, where a vast number of programming challenges and well-written tutorials covering various examples are published (Kolstad and Piele 2007). It introduces the world of competitive programming step by step. Moreover, it organizes five or six online contests each year. The **ACM-ICPC Live Archive** is an ACM programming challenges database originating from regional contests and finals held worldwide since 1988. Any interested participant can solve those problems in numerous programming languages. In turn, the users of the **Aizu Online Judge** solve challenges originating from both the Japanese Olympiad in Informatics and Japanese high school contests. The **MAIN** website provides an archive of various challenges originating from Polish and international contests co-organized by the Polish Olympiad in Informatics. Moreover, it also offers interactive courses on programming and algorithms. A platform supplementing MAIN is **Szkopuł**, which allows users to create virtual contests using challenge sets hosted on MAIN. There are also other, smaller systems, such as the **ACM Kyrgyzstan Subregion Challenges Archive**, **Indian Programming Olympiad Archive**, **A2 Online Judge**, **AcDream**, and **Light OJ**.

Unfortunately, most of the systems just mentioned are closed source, except HUSTOJ. There are also three other platforms that are open source: **CDOJ**, **Teddy Online Judge**, and **Don Mills Online Judge (DMOJ)**. The last one is especially worth mentioning in this category because it is a fully open-source and well-documented system available on GitHub. It publishes challenges originating from the Canadian Computing Competition, Canadian Computing Olympiad, Croatian Open Competition in Informatics, International Olympiad in Informatics, and many others. It also runs a contest once every month. Thanks to the extensive documentation, it is also very easy for potential newcomers to submit novel challenges to DMOJ.

There are also several websites that present less classical applications of online judge systems, such as **Project Euler**, which hosts more specific mathematical challenges in comparison with regular ACM counterparts. Every week, a new challenge is posted. Users are ranked according to their quality scores obtained for submitted solutions. The quality score depends on the number of users who have submitted the correct solution earlier. Global ranking is constructed by taking into account the quality scores obtained for the 10 most recent challenges. Users usually develop a dedicated program. However, in general, they can even try to solve the current challenge analytically because only the textual representation of the solution obtained for the particular test instance can be submitted and automatically judged. In turn, **Al Zimmermann's Programming Contests** is another online platform where contestants are supposed to send only the textual representation of the solution for computationally intensive problems. It runs new contests once or twice a year. Another event worth mentioning is the **Internet Problem Solving Contest** (**IPSC**), which is an annual programming competition considering fancy challenges of various types (e.g., the ACM, Capture the Flag, optimization problems, etc.). Contestants are grouped together in teams of three. Those who sent a postcard to coordinators are granted an additional time bonus. At the end of a particular contest, all challenges are moved to the archive where they are available for practicing as part of a virtual contest system. Another interesting system is the **Herbert Online Judge**, which provides more than a thousand challenges that are solved by users in the H language. The H language is a very simple programming language proposed for controlling a robot called Herbert that was developed for the purpose of the challenge, called the Algorithm Competition, conducted during the Imagine Cup tournament in 2008. It allows the participant to strengthen his or her algorithmic skills, especially regarding the finding of patterns and implementing of recursion. Finally, two of the most prominent IT companies, Facebook and Google, also organize their own programming competitions. **Facebook Hacker Cup** is an annual competition that uses an online judge system to evaluate the quality of contestants' submissions. Unfortunately, the challenge archive is not available publicly once the particular contest ends. **Google Code Jam** is an event that is also organized annually. Sometimes there are also regional or even special editions. During the competition, participants send the textual representation of the solution computed for the particular test instance and the program sources that were used to obtain it. However, the latter are used only for verification if participants do not share solutions among each other, and only the former is evaluated.

Because there are thousands of challenges collected in numerous online judge systems provided on the Internet, there have also been some attempts to index and classify them. Zhu and Fu (2012) introduced a system for automatic classification of challenges based on hierarchical knowledge representation (Yoon et al. 2006). Unfortunately, their system has not been put into practice. There are also other online resources maintained manually that could be helpful in choosing interesting challenges from among the many provided examples, such as **uHunt** dedicated to the UVA online judge. The very useful initiative emerged in the **Beijing Normal University Online Judge** (**BNUOJ**). It provides a single local judge equipped with slightly more than 1,000 challenges and a virtual one that allows the user to apply 24 other online judges. In this manner, the system integrates more than 50,000 challenges published on various websites. When the user submits a solution to one of these remotely available challenges, the BNUOJ automatically forwards it to the appropriate online judge and acquires from the remote system the evaluation result. It also supports the inclusion of the challenge phase, similarly to the Codeforces and TopCoder Competitive Programming contests. Moreover, it allows users to organize virtual competitions using challenges originating from local as well as remote archives. This system can replay contests and merge the final standings. Last, but not least, the BNUOJ is an open-source project available on GitHub.

2.3 Education

The application of online judge systems in education has a very long history; they have been used at least since 1961, when they were introduced at Stanford University (Forsythe and Wirth 1965; Leal and Moreira 1998) to support the evaluation of students' programs coded in ALGOL. This concept was followed by many more systems, none existing currently, such as Ceilidh (Benford et al. 1993), Kassandra (von Matt 1994), CoBalT (Joy and Luck 1995), or RoboProf (Daly 1999) and finally resulted in the implementation of online judges in the form of a Massive Open Online Course (MOOC). In general, a MOOC is addressed to an unlimited number of participants who can learn independently worldwide without the requirement of respecting a fixed course schedule (Pieterse 2013). We can distinguish two main types of MOOCs; namely, traditional online courses and those requiring collaboration between participants and teaching staff (i.e., xMOOC (Prpić et al. 2015) and cMOOC (Kop 2011), respectively). The former usually provides a fixed set of recorded lectures and self-test exercises. In the case of online judge systems, these exercises are presented in the form of automatically judged programming tasks. The latter is much more successful in knowledge building as a consequence of a collaborative dialogue (Bell 2011). In the case of online judge systems, such dialogue is usually implemented by the possibility of adding annotations to the students' source code. In 2012, several of the most well-known organizations in this field emerged, such as edX (a nonprofit organization formed by MIT and Harvard), Coursera, and Udacity (originating from Stanford). In the middle of 2013, edX and Google started a partnership to establish a commonly used resource (i.e., mooc.org) dedicated to building and hosting various e-learning courses. Currently, the numbers of courses provided by the organizations just mentioned are on the order of hundreds or even thousands. Development of a MOOC is not cheap for organizations but, in the long term, yields savings. Moreover, it allows them to differentiate from other educational organizations, even without an appropriate level of expertise. However, it is worth noticing that MOOCs require a high degree of motivation and self-discipline from the participants. Thus, it is usually a way to strengthen professional skills, and it is rarely used to learn a new field from scratch (Kaplan and Haenlein 2016).

Utilization of online judges allows an educational organization's staff to assess students' assignments automatically. Application of these systems results in several advantages. First, the teacher can verify the correctness of solutions submitted by students with higher accuracy. When the teacher prepares the complete set of test instances covering all corner cases resulting from the problem definition, the possibility of acceptance of an incorrect solution is almost negligible. Second, the time needed for evaluation is much shorter; therefore, the teacher can prepare and assign students many more exercises. Finally, students receive an almost instant answer as to whether their solution is correct. An inspiring description of the successful application of an online judge in the teaching of Algorithms and Data Structures and Competitive Programming courses at the National University of Singapore is presented in Cheang et al. (2003). Ala-Mutka presented an in-depth review of applications of online judges in education based on the analysis of several systems (Ala-Mutka 2005). In 2010, Ihantola et al. prepared a more recent review of available software dedicated to automatic assignment of programming exercises, focusing on detailed descriptions of their features and various usage scenarios, which is interesting from a pedagogical or educational point of view (Ihantola et al. 2010). In turn, Cruz et al. presented how to extend the typical online judge architecture to develop a system that goes a step further by providing valuable feedback to users at a semantic level in the form of meaningful advice to understand where the problem is and how to improve the code (Fonte et al. 2013). Such an approach can significantly impact the student learning process (Wang et al. 2016). Finally, there is an interesting idea of building computer games that integrate or wrap online judge systems to be more attractive to users (Ivanova 2016). The list of existing online judge systems presented in this subsection can be found in Table 2.

Because online judges that publish challenges collected from competitive programming contests have become very popular these days, there also exist sophisticated adaptations of such systems for educational purposes. For example, the **URI Online Judge** is a system where provided challenges are distinguished into eight categories, allowing users to easily find exercises from a given topic. It also provides a panel addressed to teaching staff that supports tracing of the student learning process. Another adaptation worth mentioning is **CodeChef**. It is a system that provides an online code editor and compilers that support more than 40 programming languages. It classifies all provided challenges into categories according to their difficulty levels. However, even the easiest ones are relatively difficult. Much easier tasks can be found at **Jutge.org** (Petit et al. 2012), which is an online judge (free for educational purposes) where students can solve thousands of challenges using approximately 20 different programming languages, including even less popular ones, such as Verilog. It provides sophisticated panels for students and instructors. Finally, **Programming Grid** is an interesting online platform that allows users to organize educational courses based on a challenges archive provided by the Peking University Judge Online. It follows a course-based concept instead of a challenge-based concept (Luo et al. 2008). Unfortunately, this service is only available in Chinese.

Some of the online judges introduce additional gamification elements (Deterding et al. 2011) to strengthen the commitment and motivation of the students. For example, **CheckiO** is a platform

Table 2. Online Judge Systems Applied for Educational Purposes

Name and URL Address	OSS	GUI Language	Compilers Class id	#Problems	#Users	Established	Active
CheckiO	No	Eng	1	100	110000	2013	Yes
Code Fights	No	Eng	3	1250	500000	2015	Yes
Codeboard	Yes	Eng	3	24000	60000	2015	Yes
Codecademy	No	Eng	3	N/A	25000000	2011	Yes
CodeChef	No	Eng	3	1500	300000	2009	Yes
CodeHunt	No	Eng	1	134	350000	2014	Yes
Codewars	No	Eng	3	1200	400000	2012	Yes
CodinGame	No	Eng	3	55	500000	2012	Yes
CodingBat	No	Eng	2	300	-	2009	Yes
Embedded Security CTF	No	Eng	1	19	35000	2014	Yes
Exercism	Yes	Eng	3	1450	30000	2013	Yes
Jutge.org	No	Eng,Spa,Cat,Ger,Fre	3	2000	14000	2006	Yes
Leek Wars	No	Eng,Fre	1	1	54000	2013	Yes
Programming Grid	No	Chi	2	640	-	2008	Yes
Python Challenge	No	Eng	0	33	N/A	2005	Yes
RACSO	No	Eng,Spa,Cat	0	330	-	2012	2015
The AI Games	No	Eng	3	8	2700	2013	Yes
URI Online Judge	No	Eng,Spa,Por	2	1170	4100	2011	Yes

For column descriptions, see Table 1.

that provides a large number of relatively easy tasks prepared to support the learning process of the Python and JavaScript programming languages. It provides a few elements of gamification and social networking. Moreover, it allows the users to define “classrooms” and assign particular students to them to support the process of monitoring their learning progress. The platform was designed in a very attractive way and is still actively extended with new features. CheckiO is also supplemented with a massively multiplayer online strategy game called **Empire of Code** where players can solve various types of problems collected by the CheckiO platform to gain various in-game bonuses. To be successful, they also have to develop artificial bots that are responsible for the protection of their units as well as for attacking their potential enemies. Another platform in this category that is worth mentioning is **Codewars**. It collects a large number of exercises ordered by both difficulty level and topic-oriented categories. Codewars always provides several basic test cases, which are used for preliminary verification of the solution. A unique feature of this platform is that its user interface is inspired by Japanese culture (e.g., *kata*, *kyu*, and Kumite). Finally, **CodeHunt** is a game developed by Microsoft to strengthen the algorithmic skills of players. It provides a snippet of code together with corresponding test cases. The goal of the player is to implement in C# or Java an algorithm that will generate the expected output for all provided test cases.

There are also two additional educational games worthy of consideration in this category. In **Leek Wars**, the player is responsible for implementation of an artificial intelligence script in LeekScript, a specialized programming language designed especially for Leek Wars, to win in a turn-based game. By defeating the opponents in the game, the user can upgrade his personal features, which allow him to be significantly better armed. It is also possible to improve the AI script based on the experience gained from skirmishes against provided bots as well as AI scripts implemented by other users. In turn, **PythonChallenge** is another in-browser game where the user develops the solution to simple riddles following hints provided in real-time by the system.

There are also other online platforms supporting less classical attitudes for the evaluation of submitted solutions to programming exercises. **Codecademy** is one of them, focusing mainly on providing programming courses. Instead of exercises, the platform provides many professional tutorials. To progress in the given course, the user has to finish all the steps considered by the particular tutorial. The platform provides a code editor, a terminal, and other practical tools required to practice newly learned skills. In turn, **CodinGame** is an online platform providing a variety of programming challenges. It supports programming puzzles, optimization and code-golf problems, and multiplayer games where the user implements an artificial intelligence script to control the bot. The significant advantage of this website over its counterparts is the vast number of supported programming languages and the visually attractive and exciting challenges, guaranteeing much fun and satisfaction during the solving of them. It also continuously gives the opportunity to participate in so-called *code clashes*, which represent very short competitions organized for several users that are started automatically by the system many times a day. Another platform, **CodingBat,** is a very simple page for people who have just started programming. This site supports only the Python and Java programming languages. It is not even necessary to be registered to submit solutions. Another tool, called **Exercism.io,** provides a very original approach for the solving of programming assignments. It provides a command line tool and an API that is used to fetch task descriptions and submit solutions. The problem definitions are prepared in such a way as to follow the Test Driven Development (TDD) methodology. Authors of this platform provide only a general description of the problem and the automated test suite. The task of the user is to deduce all relevant details from the inspected test suite. Finally, **Codeboard** is a user-friendly system that supports not only the teaching of various programming languages by creating and sharing exercises with students, but also a test case-based automated evaluation that can be easily customized by a teacher. Moreover, the teacher can easily follow the students' progress (Antonucci et al. 2015).

One of the unique features of the CodinGame platform is the opportunity to compete in multiplayer games. In such games, the users develop AI scripts that control bots competing with other bots in the virtual arena generated by the system. There are also two additional portals that follow such a concept: **Code Fights** and **AI Games**. The former is a platform where users develop their AI scripts in one of nine modern programming languages. Users receive experience points for every victory gained by their bots in duels. Based on gained points, users rise in level and earn badges. The idea behind the latter platform is to create AI scripts for playing most popular games, such as Tic-Tac-Toe or Go. This platform provides eight popular game engines and judges. Users can submit an AI script representing the particular bot that will compete in the given game with AI scripts provided by other users. Users are highly motivated to continuously improve their AI scripts to climb in the rankings. In fact, the system is closed source. However, the source code of engines is shared on GitHub.

Finally, two systems support the strengthening of professional skills in highly sophisticated programming concepts. The objective of the **Embedded Security CTF** system represents support for learning assembly language during a password cracking game. The user has to discover the password by analyzing the assembly program using the provided debugger to inspect the details of the password verification process. Based on the performed analysis, the user has to guess the password that will be accepted by the system. The tools provided by the platform are advanced; for example, the user can analyze the entire code during its execution, set breakpoints, and inspect values stored in processor registries and memory dumps. In turn, **RACSO** is a platform where exercises related to automata and formal languages are provided (Creus and Godoy 2014). In this system, the judge accepts the description of the automaton as well as grammar.

Table 3. Online Compilers

Name and URL Address	OSS	GUI Language	Compilers Class id	Established	Active
C++ Shell	No	Eng	1	2014	Yes
Codeanywhere	No	Eng	3	2013	Yes
Codepad	No	Eng	3	2008	Yes
CodeSkulptor	Yes	Eng	1	2012	Yes
Coding Ground	No	Eng	3	2006	Yes
Codio	No	Eng	3	2013	Yes
Ideone	No	Eng	3	2009	Yes
Online Compiler	No	Eng	2	2009	2013
Web Compiler	No	Eng	1	2014	Yes

For column descriptions, see Table 1. Online compilers do not provide a problem database and usually do not require registration; thus, the columns related to the number of problems and users have been removed.

2.4 Online Compilers

Another category of online judges, defined in a broad sense according to Definition 3.6, represents online platforms where user source code, developed in various programming languages, can be remotely compiled and executed via a browser. In fact, they do not allow the publishing of any challenges because they usually support only the first step of the evaluation procedure provided in Definition 2.1. Sometimes, partial support is also included for the assessment phase by allowing the user to submit his own test instances, which are used during evaluation. The list of online compilers discussed in this subsection is presented in Table 3.

One of the most feature-rich online compilers is **Codeanywhere**, which is a cloud-based IDE that allows users to share a dedicated virtual development environment and collaborate in real-time. It provides the ability to connect automatically to GitHub, Bitbucket, FTP server, and Amazon cloud. A user can also set up for free a single, specialized, virtual container where a custom development environment will be created out of the box. This environment is very stable, even when many developers collaborate within the same project in real time. Another online platform offering a fully featured IDE is **Coding Ground**. It allows users to edit, compile, execute, and share their projects in a cloud-based environment. It provides free terminals and IDEs that support development in plenty of different programming languages. Every program is executed in a dedicated Docker-based container created on demand. Finally, **Codio** is a cloud-based online IDE supporting a large number of programming languages. It provides many useful features, such as a remote Ubuntu-based development machine, integration with e-learning platforms, and plagiarism detection.

There are also several simpler online compilers, whose objective is to provide the opportunity to compile and verify user code quickly without the need to construct separate virtual containers for specialized purposes. For example, **Ideone** is a freely accessible online compiler that supports plenty of programming languages and is maintained by the authors of the SPOJ online judge. In turn, **CodeSkulptor** is an online interpreter of the Python programming language that has a very aesthetic user interface and supports the learning process, especially for beginners. Moreover, it visualizes the execution of the particular program and provides an unofficial open-source offline server that can be used to run CodeSkulptor locally. Another system in this category, called **C++ Shell,** provides an online interface for the GCC compiler. This system allows the user to compile the submitted source code and run it in a virtual sandbox environment created on demand. **Codepad**, in comparison to the other counterparts, allows users to share code among

Table 4. Online Judge Systems Applied for Recruitment Purposes

Name and URL Address	OSS	GUI Language	Compilers Class id	#Problems	#Users	Established	Active
CodeEval	No	Eng	3	240	85000	2009	Yes
Codility	No	Eng	3	3000	-	2009	Yes
HackerEarth	No	Eng	3	3700	1000	2012	Yes
HackerRank	No	Eng	3	1000	84000	2009	Yes
LeetCode Online Judge	No	Eng	3	190	-	2010	Yes
Qualified	No	Eng	3	4500	-	2015	Yes

For column descriptions, see Table 1.

collaborators using custom URLs. Another system worth mentioning is called **Online Compiler**, which is a platform that allows users to remotely compile submitted source code developed in C/C++, Fortran, Java, Pascal, or Basic programming languages and download the executable file built for Windows or Linux. Finally, **Web Compiler** is an online compiler for Visual C++ that provides a minimalistic user interface.

2.5 Recruitment

There are also several, mainly commercial, platforms that use online judge systems primarily to support the recruitment process. The list of systems discussed in this subsection is presented in Table 4.

First, **CodeEval** is a platform used by developers to showcase their skills in application-building competitions and programming challenges. CodeEval represents an exclusive community of developers who can compete and, as a result, build out their profiles to showcase their coding skills in the software development community. It uses Docker-based containers constructed on demand. Another platform worth mentioning is **Codility**, which supports recruiters in reaching out to a large number of promising candidates in a relatively short amount of time. Moreover, developers can strengthen their coding skills by competing in programming challenges to build their professional reputation within the community. In turn, **HackerEarth** is an online platform whose the main objective is hiring talented developers, organizing hackathons, and hosting crowdsourcing-based ideas. Developers can win various types of rewards. From companies' point of view, it is an excellent way to gather innovative ideas from a diverse community, including developers as well as computer scientists. **HackerRank** is an online tool similar to HackerEarth and was designed with supporting the hiring of developers in mind. It focuses mainly on supporting recruiters with the delivery of fully customized coding challenges addressed directly to the potential candidates and the seamless integration of real-time assessments into the recruiting process. From the developer point of view, it is a place where people from all over the world can solve rather tricky programming challenges classified into several categories, namely algorithms, machine learning, and artificial intelligence. Finally, **Qualified** is a platform provided by the authors of Codewars. Here, the developer implements the solution in real time, and then the system automatically executes and qualifies it based on a predefined unit test cases set. It provides an interactive IDE, which supports incremental development. During real-time interviews, the recruiter can benefit from the application of whiteboard tests. On the other side, there is another very useful platform called **LeetCode**, which is a web service that supports preparation for technical job interviews. Users can solve exercises that may occur during such interviews divided into the following categories: algorithms, databases, and Linux shell.

Table 5. Online Judge Systems Applied for Data-Mining Purposes

Name and URL Address	OSS	GUI Language	Compilers Class id	#Problems	#Users	Established	Active
CrowdANALYTIX	No	Eng	0	105	16000	2012	Yes
DREAM Challenges	No	Eng	0	45	5000	2006	Yes
Kaggle	No	Eng	0	220	550000	2010	Yes
MLcomp	No	Eng	1	12400	7000	2010	2017
OpenML	Yes	Eng	3	19600	2500	2016	Yes
Optil.io	No	Eng	3	11	300	2016	Yes
TopCoder Data Science	No	Eng	2	400	-	2001	Yes
TunedIT	No	Eng,Pol	3	36	10000	2008	2015

For column descriptions, see Table 1.

2.6 Data-Mining Services

There are also various websites that use online judge systems to evaluate data-mining algorithms. Their list is presented in Table 5.

Most likely, the best known is the **Kaggle** platform (Goldbloom 2010), for which the primary objective is the organization of data-mining challenges with monetary prizes. However, Kaggle does not incorporate a regular online judge. Users of this service have to execute their code locally using test data provided by Kaggle and submit only the results generated by the algorithm. Providing the source code is not required. **CrowdANALYTIX** implements an idea similar to Kaggle. The system hosts contests, and users submit outputs for a given problem before the deadline. The winner is chosen based on these solutions. **DREAM Challenges** is another platform similar to Kaggle, but it focuses on problems related to systems biology and translational medicine (Costello and Stolovitzky 2013; Saez-Rodriguez et al. 2016). As opposed to Kaggle, its primary objective is to solve scientific challenges; therefore, it is targeted specifically at researchers. It provides expertise and institutional support with the help of Sage Bionetworks, along with the infrastructure to host challenges via its Synapse platform (Derry et al. 2012), thus creating a system with outstanding scientific value.

MLcomp is another online platform that was designed with a slightly different idea in mind. It provides a cloud-based platform dedicated to data-mining research. Instead of challenges, it stores datasets. Any user can upload his or her own datasets and algorithms. All algorithms submitted by users are stored in the system and can be executed later by any other user processing any uploaded dataset using the computing infrastructure provided by MLcomp. Such an approach creates a very versatile data analysis platform. Unfortunately, this platform is no longer maintained because its authors are currently developing a new online platform, called **CodaLab**, dedicated to planning and conducting research experiments. However, there is another platform called **OpenML** that offers a similar approach (van Rijn et al. 2013), providing at the same time a much more modern user interface design, as well as interfaces for the most popular scientific languages, namely, R, Python, Java, and supplemental REST API. Currently, it stores more than 19,000 datasets.

It is also worth mentioning two other online platforms dedicated to publishing data-mining problems in the form of programming contests: **TopCoder Data Science** (TopCoder DS) and **TunedIT**. The former, previously called TopCoder Marathon Matches, has already organized several hundred competitions, including many sessions devoted to solving industrially inspired challenges. Unfortunately, the set of supported programming languages is limited to only C++, C#, and Java. The latter (Wojnarski et al. 2010), as opposed to the TopCoder DS, obligates users to submit reports describing their solutions. After the end of a particular challenge, all reports submitted by

Table 6. Online Judge Systems That Can Be Used as Development Platforms

Name and URL Address	OSS	GUI Language	Compilers Class id	Established	Active
A+	Yes	Eng	1	2017	Yes
BOSS	Yes	Eng	3	2012	2009
CloudCoder	Yes	Eng	2	2012	Yes
Code Runner for Moodle	Yes	Eng	3	2016	Yes
DOMjudge	Yes	Eng	3	2004	Yes
Mooshak	Yes	Eng	2	2005	2015
Online Judge Plugin for Moodle	Yes	Eng,Chi,Por,Pol	3	2012	2015
SIO2	Yes	Eng,Pol	2	2012	Yes
TestMyCode	Yes	Eng	1	2013	Yes
Tsinghua Online Judge	No	Eng,Chi	2	2012	Yes
Virtual programming lab	Yes	Eng	3	2012	2015
Web-CAT	Yes	Eng	3	2003	Yes
xLx	No	Eng	1	2001	2008

For column descriptions, see Table 1. Such systems usually do not provide problems databases and do not require registration, so the columns related to the number of problems and users have been removed.

participants are verified by the organizers. Unfortunately, the website is not maintained as of late. Thus, new contests are not being added.

Finally, **Optil.io** is a platform used to publish optimization problems that require the design of algorithms for optimizing objective functions on a provided dataset (Wasik et al. 2016). Users can submit solutions in several supported programming languages, as well as statically compiled Linux binaries. It is also possible to spread the code of the solution across several source files if a CMake file is also submitted by the user. Moreover, submissions can also use other libraries and specialized solvers that are supported by the proposed platform. The range of external software can be expanded on user demand.

2.7 Development Platforms

Anyone who would like to host a programming competition or a course using his or her own infrastructure can apply one of the several available online judge development platforms. These systems can be downloaded and installed locally, providing full administrative privileges to the user. Moreover, most can be adapted to user needs and integrated with external services. The list of services discussed in this subsection is presented in Table 6.

Currently, the platform that is worthiest of recommendation is **DOMjudge**. It is a fully automated judge system that allows users to prepare and perform programming contests following the rules defined for the ACM ICPC. It is an actively developed and feature-rich system. Its quality has been proved through its application in the ACM ICPC finals since 2012. **Mooshak** (Leal and Silva 2003) is another online platform designed for managing programming contests following rules similar to those designed for the IOI, as well as for the ICPC. It offers a classic GUI, which has attained a very mature stage and is extensively used at many universities (e.g., in Portugal). Finally, **SIO2** is a stable online judge platform used and developed by the Polish Olympiad in Informatics.

There are also other development platforms that are dedicated to the support of the educational process. For example, **CloudCoder** (Spacco et al. 2015) is an open-source web-based system inspired by CodingBat that was designed to simplify the lives of instructors of programming courses by allowing them to assign exercises to students to assess their skills. **Tsinghua University Online Judger** is a course-oriented online judge designed for universities. The system allows for the organization of programming classes using automatically evaluated challenges (Zheng et al. 2015). It is hosted on the Tsinghua University server and allows users to add new problems. However, the source code of the system is not open source. **BOSS** is a system that was designed to support the performance of programming courses using automatically judged assignments (Joy et al. 2005). Over several years, it has evolved into a platform that supports the teaching of any topic, including the online verification of programming exercises. Unfortunately, the system was updated for the last time in 2009. Finally, **Web-CAT** is an automated grading system implemented in Java to provide an approach to the evaluation of students' assignments. This is done by collecting test cases submitted by students for assessment using test coverage measures (Edwards and Perez-Quinones 2008).

In the literature, one can also find online judge development platforms integrated with e-learning systems. One of the oldest such approaches is **xLx**. It has existed since 2001, when the first prototype was developed to support the educational process conducted at WWU Münster (Husemann et al.). It first published courses related to SQL and XML languages. In 2008, the second version of this platform was released; however, since that time, it has not been updated. Moreover, the source code of the system is not currently available. In 2011, Xavier and Coelho presented the review of several online judge platforms, as well as a description of the platform developed at the **University of Porto** inspired by the Moodle and DOMjudge systems (Xavier and Coelho 2011). Kaya and Özel (2012, 2014) presented a similar approach, but enriched it with a plagiarism-detection package called **Moss** (Schleimer et al. 2003). **Code Runner** is a question type plug-in for Moodle that allows one to execute code submitted by a particular student as an answer for a broad range of programming puzzles, and it supports various programming languages simultaneously. It is intended primarily for use in computer programming courses, although it can be used to grade any question for which the answer is represented in textual form. Currently, the most comprehensive and mature plug-in for Moodle is **Virtual Programming Lab** (Rodríguez-del Pino et al. 2012), which allows users to edit their source code interactively in a browser, develop custom grading scripts, and detect plagiarism automatically. Unfortunately, it has not been updated for over a year and thus is not compatible with the newest version of Moodle. Finally, the **Online Judge plug-in for Moodle** (Zhigang et al. 2012) is a component that allows users to submit solutions to programming exercises and supports approximately 40 programming languages. However, it uses only the deprecated activities API supported by Moodle up to version 2.2 and can currently only be used in legacy mode. Because Moodle is a very popular online system supporting the educational process, there have been additional attempts to integrate it with online judge systems. However, all of them are either currently inactive, not compatible with the current stable version of Moodle, or exist as proof-of-concept projects only, such as those proposed by Danutama and Liem (2013), Carral (2013), and Davuluri and Madadi (2016). A similar plug-in-based concept is provided by **TestMyCode**, which allows for an automated transfer of programming assignments directly from within the most widely known IDEs (Pärtel et al. 2013). It is worth noting that the submission phase is much simpler in this case; nevertheless, the evaluation is provided based on predefined unit tests. Moreover, a new e-learning platform called **A+** has been recently published by Aalto University based on previous work by Karavirta et al. (2013). The main advantage lies in its flexible utilization of external services. Additionally, it provides a REST API, allowing for the automatic acquisition of various information stored in the system.

3 EVALUATION METHODOLOGY

The most important component of each online judge system is an evaluation engine that assesses submissions uploaded by users. The evaluation procedure of such systems usually consists of the three steps described in Definition 2.1. The first step is relatively simple and technical. First, it usually relies on the execution of an appropriate compiler. Second, it verifies that the compiler does not return any compilation errors and that compilation time and output binary size do not exceed the specified threshold. Using recursive macro-definitions, it is relatively easy to abuse the system in this way.

Usually, when the solution compiles successfully, there are no obstacles preventing it from execution. However, sometimes online judge systems allow for submitting static binaries compiled by the author on the client side. In such a case, the system must verify that all required dynamically linked libraries are available and that the binary is compatible with the evaluation environment's architecture.

The remaining steps of the evaluation procedure are more complex and diversified. We review them in detail in Sections 3.2 and 3.3. However, before presenting our explanation, we introduce basic concepts related to the theory of combinatorial problems (see Section 3.1). Because most problems published on online judge systems are some type of combinatorial problem, such explanation is crucial to better understand the evaluation process.

3.1 Combinatorial Problems

Any combinatorial problem Π is defined in the form of a set of parameters associated with it and a set of required constraints that must be satisfied by its solution. An instance I of the problem Π is obtained by setting particular values for all parameters considered in the problem definition. One of the most well-known combinatorial problems is the 0-1 knapsack problem, which is defined by a given set of n items, each described by a weight w_i and a value v_i, along with a maximum weight capacity W. The solution of this problem is a combination of items (assuming that every item can be introduced into the knapsack at most once) that fit in the knapsack and for which the total value is maximal. The set of all instances of the problem Π is called the *domain* of the problem D_Π.

In fact, two main classes of combinatorial problems can be distinguished: decision problems and search problems.

Definition 3.1 (Decision Problem). A decision problem Π is a problem for which a solution is an answer either "yes" or "no" for a question associated with the set of its parameters.

An example of this problem can be a particular instance of the knapsack problem and the following question associated with it: Is it possible to pack the knapsack according to a given set of items reaching a total value not lower than a particular constant V_{max} and, at the same time, not exceeding the maximum weight capacity of knapsack W?

Definition 3.2 (Search Problem). A search problem is a problem for which a solution is either a particular object (or value) satisfying given constraints or the answer "no" when such a solution does not exist.

In this case, the problem relies on finding any feasible solution of the knapsack problem based on a given set of items; that is, such a combination of items that fit in the knapsack for which total value will not be lower than the particular constant V_{max}.

An *optimization problem* is a special case of the search problem where the solution is represented by the optimal object (or value) according to a given objective function. An example is another version of the knapsack problem formulated in the following manner: Find such a combination of items that fit in the knapsack for which the total value is maximal. In contrast to the search problem, here the solution is the best one among a set of all feasible solutions for this problem.

For more information regarding optimization problems and their complexity, see the excellent book by Garey and Johnson (1979).

3.2 Assessment

The assessment phase is usually the most computationally expensive. It requires execution of the compiled solution in each considered test instance. Each of these executions can take several dozen minutes for complex industry-inspired problems. Here, we provide the formal definition of a test instance:

Definition 3.3 (Test Instance). Let Σ denote an alphabet used to encode both input and output data. Test instance $t_i \in T$, where T is a set of all considered for the particular problem test instances, is defined as a triple $t_i = (d_i, o_i, p_i)$, where $d_i = \Sigma^*$ is an input data, $o_i = \Sigma^*$ is a reference output data, and p_i is a set of parameters passed to the evaluation engine.

In the vast majority of problems, the alphabet Σ that is used to encode both input and output data consists of digits, spaces, and new line characters $\Sigma = \{0, 1, \ldots, \textvisiblespace, \backslash n\}$. Sometimes, it is extended with lower- and uppercase letters and up to several special characters. Nevertheless, the format used to encode both types of data is usually as simple as possible for parsing purposes. Usually, it is represented by a list of integer numbers the structure of which is defined in the problem description or a simple Comma-Separated Values (CSV) file often used in data-mining applications. Commonly, each input and output data is represented by a single file redirected to the standard input and from the standard output of the solution program, respectively. However, in case of specific problems, it could happen that several files stored in the single archive are provided.

When, according to the problem definition, for each input instance d_i there is exactly one correct solution, this solution can be precomputed and stored in the reference output data file o_i. In this way, the evaluation engine computes the reference solution immediately and next uses it during verification of each user submission. Whenever there are many feasible solutions for a single input instance, the reference output data file can store certain precomputed values that allow the computational complexity of the evaluation process to be reduced. When no additional data can simplify the evaluation process, the reference output data o_i can be empty (i.e., $o_i = \varnothing$).

The set of parameters, p_i, represents the specific resource limitations (e.g., CPU time, maximum utilization of RAM) that cannot be exceeded during the evaluation based on this particular instance. However, additional parameters can also be passed if necessary, such as a random number generator seed when the engine allows randomized solutions or a maximum size limit for output data generated as a result of the solution execution for this particular instance. When the evaluation engine is configured to use the default resource limits and no other parameters are needed, the set p_i can be empty (i.e., $p_i = \varnothing$).

During the assessment phase, the solution compiled as the output of the submission phase is used. Here, we provide the formal definition of the solution:

Definition 3.4 (Solution). A solution is a function, $b(d_i, p'_i) \rightarrow o'_i$, representing the binary form of the submission that, based on the input data d_i, computes output data o'_i, taking into consideration execution parameters p'_i provided by the evaluation engine.

The set of execution parameters provided by the evaluation engine p'_i can be equal to the set of parameters defined as part of the particular test instance (i.e., $p'_i = p_i$). However, in particular, p'_i can be the subset of p_i (i.e., $p'_i \subset p_i$), p'_i can be a completely different set of parameters, or p'_i can be empty (i.e., $p'_i = \varnothing$). As opposed to p_i, the set p'_i is usually empty because parameters influencing the evaluation process are hidden from the solution. The most commonly used execution parameters often passed to the user's solution are the seed for the random number generator and the maximal time limit that cannot be exceeded by the solution during its execution for the particular test instance.

Definition 3.5 (Evaluation Engine). An evaluation engine is a function, $E(b, t_i) \rightarrow (s_i, v_i, e_i)$, that executes the binary file b, giving it as an input the test instance t_i, and returning the execution status s_i, an evaluation score calculated for the solution output, $v_i \in \mathbb{R}$, and the list of statistics collected for the execution process e_i.

The status s_i of the submission execution can take one of the following values:

- Accepted (ACC) when the submission execution terminates successfully without any runtime error, without exceeding resources limits, and returns feasible output data coherent with the format described in the problem description;
- Time Limit Exceeded (TLE) means an incorrect execution of the submission due to exceeding the maximal processing time limit;
- Memory Limit Exceeded (MLE) means an incorrect execution of the submission due to exceeding the maximal RAM utilization limit (covering both the stack and heap);
- Wrong Answer (WA) means that the program generated an output of unknown format (i.e., the existing format is not coherent with the format requested in the problem description) or some additional constraints formulated in the problem description were not satisfied;
- Runtime Error (RE) means that a runtime error occurred during the particular submission's execution;
- Output Limit Exceeded (OLE) means that the submission exceeded the maximal limit for the size of output data.

In the case of a WA status code, the user may also receive additional information regarding the reason the answer is incorrect. However, this is not a common practice. Various online judges can introduce other statuses, but those listed are the most commonly used. Most commonly, users can instantly see the status of the submission, at least for some example instances, as soon as it is assessed.

Many online judge systems follow the ICPC rules and evaluate the submissions on each test instance in a binary way—as a correct or incorrect solution only. In such a case, the evaluation score is always equal to 0 (i.e., $v_i = 0$). There are two major cases when this value is used. First, in optimization problems, when it stores the value of an objective function computed for the output data obtained as a result of a particular submission execution on a particular test instance. It can be both, a classical optimization problem (Garey and Johnson 1979), or a code-golf problem when the objective is to optimize the size of the source code solving some task. Second, in competitions following IOI rules, v_i characterizes the score that the user gains for receiving the "Accepted" status for this particular instance (i.e., $s_i = ACC$). In general, different test instances can be characterized by various score values, or even more complex scoring procedures can be applied. For example, Polish Olympiad in Informatics penalizes solutions that use more than half the time limit by decreasing the score proportionally, according to the following formula:

$$v_i = V_i \cdot \min\left(1.0, 2.0 \cdot \frac{\mathcal{T}_i - \tau_i}{\mathcal{T}_i}\right). \tag{1}$$

Here, V_i denotes the maximal number of points that can be awarded for some test instance, $\mathcal{T}_i$ denotes the maximal time limit for the solution, and τ_i denotes the processing time used by the solution to generate the output for the particular instance.

Finally, statistics e_i, collected for the execution process of the solution, usually include information about maximum values of resource utilization observed during the particular submission execution (e.g., time and memory consumption). In particular, when the online judge does not share such information with the user, the list of execution statistics can be empty (i.e., $e_i = \varnothing$).

3.3 Scoring

The objective of the scoring phase is to compute an aggregated status s and aggregated evaluation score v of each user submission. These values are then used to display results of the evaluation procedure to the user and rank solutions to the problem.

If and only if, for all test instances, the solution received ACC status, then the aggregated status is also ACC. Otherwise, the most common procedure is to select the first status different from ACC:

$$s = ACC \Leftrightarrow \forall_i s_i = ACC \tag{2}$$

$$s = s_j \Leftrightarrow (\forall_{i<j} s_i = ACC) \wedge (s_j \neq ACC). \tag{3}$$

However, some variations are sometimes utilized, for example, returning an RE status when any "Runtime Error" occurs before any other statuses.

Computing the aggregated evaluation score v is relatively simple when the system does not deal with optimization problems. In such a case, for systems evaluating submissions in a binary way and following ICPC rules, this score is always equal to 0 (i.e., $v = 0$). Otherwise, it is usually the total evaluation score computed for all correctly solved instances:

$$v = \sum_{i=1}^{|T|} \begin{cases} v_i, & \text{if } s_i = ACC \\ 0, & \text{otherwise} \end{cases}. \tag{4}$$

In the case of optimization problems, the system has to consider the score computed based on the best solution among all submissions or a reference solution computed by the problem author. In most cases, the following formula is used to rank the submitted solutions for problems where the objective function is maximized:

$$v = \frac{100}{|T|} \sum_{i=1}^{|T|} \begin{cases} \frac{v_i}{b_i}, & \text{if } s_i = ACC \\ 0, & \text{otherwise} \end{cases}, \tag{5}$$

where b_i denotes the best solution score for the ith instance (i.e., the solution among all submissions for which the objective function score computed for this particular instance has the optimal value). Similar types of formulas can be easily proposed for minimization problems as well, taking into account an extended scoring procedure and execution times or even the utilization of other computing environment resources.

3.4 Online Judge

Based on the definitions presented in this section, we can define an online judge system as follows:

Definition 3.6 (Online Judge System). An online judge system is an online service that performs any of the steps of the evaluation procedure in a cloud; that is:

(1) collects, compiles sources if needed, and verifies executability of resultant binaries;
(2) assesses solutions $b(d_i, p_i')$ based on a set of test instances, T, defined for a particular combinatorial problem Π in a reliable, homogeneous evaluation environment;
(3) computes the aggregated status s and evaluation score v based on the statuses and scores for particular instances (i.e., s_i and v_i, where $1 \leq i \leq |T|$).

4 EXAMPLE APPLICATION BASED ON OPTIL.IO PLATFORM

To demonstrate the process of an online judge system, in this section, we present the results of an example contest that we conducted using the Optil.io online judge system (Wasik et al. 2016) described in Section 2.6. We selected this platform because, as its authors, we had the greatest

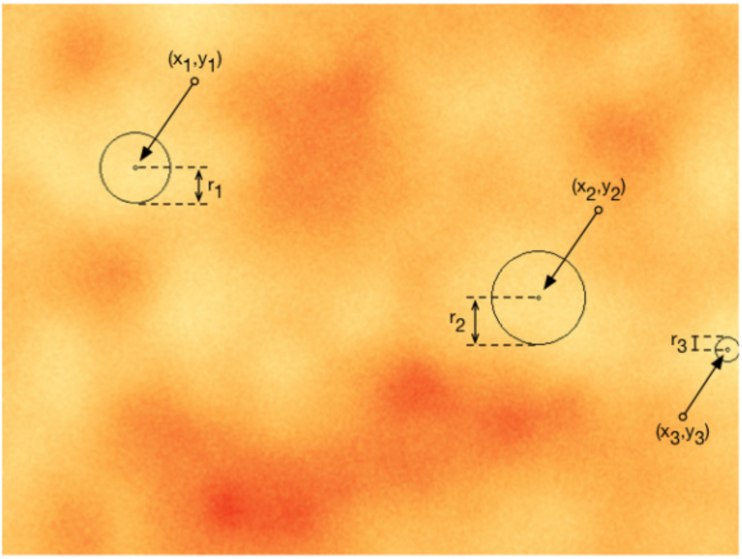

Fig. 1. Visualization prepared for an example input instance of the first challenge conducted on the Optil.io platform. The objective was to find locations (x_i, y_i) for three factories with various radii r_i. Example positions of factories and the locations that they affect are marked with black circles. The red color displayed on the map represents those areas where the values of the discontent coefficient are higher. The better locations for factories are represented by circles surrounding lighter areas.

control over the system. Based on this platform, we organized a contest where the example—an optimization-based challenge—was solved. It was a variation of the multiple facilities location problem, for which the computational complexity was proven to be NP-hard (Megiddo and Tamir 1982). Contestants were responsible for placing several factories on the Euclidian plane, ensuring minimization of the total discontent of people living next to them. Each factory was described by a single parameter (i.e., influence range). In turn, each position on the plane was characterized by the dissatisfaction coefficient computed for this position when it was affected by any factory. The visualization prepared for an example input instance of the conducted challenge and its solution is presented in Figure 1. Participants had to solve the problem for 10 randomly generated instances by placing from 3 to 50 factories on a grid of up to 1,000-by-1,000 points. For each considered instance, the processing time limit could not exceed 10 seconds, and memory used could not exceed 1 GB. For each submission executed on a particular test instance, the system generates one of the execution status codes described in Section 3.2 and computes a score using Equation (5).

We recruited the participants of this challenge primarily among a large group of second-year students of a computer science course at the Poznan University of Technology. Before participating in the competition, all accepted the terms of use of the tested service and the cookies policy according to European law. Before conducting statistical analysis of this study's results, we anonymized all data in accordance with the previously accepted terms of use.

The contest lasted 18 days, from January 23 to February 9, 2016. However, after that time, the problem was still available on the website. During the contest period, 85 users submitted 1,191 solutions to the problem, and, since its end, another 400 answers have been submitted. It is notable to underline that the main aim of the contest was to identify the best solution. It is typical for industry-inspired challenges that the solution is extremely important; however, the winning

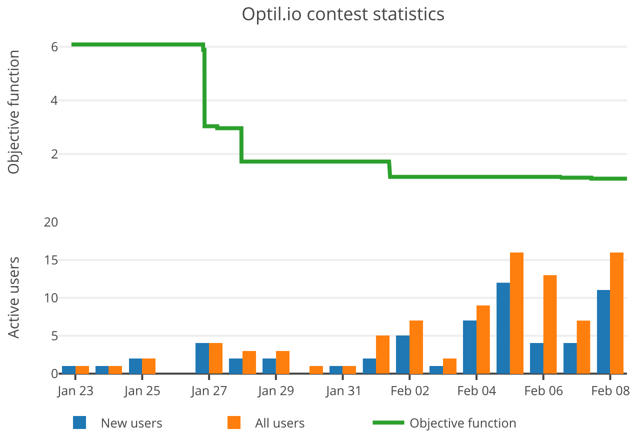

Fig. 2. Upper plot: Objective function ratio computed between the best solution currently and the winner of the entire competition. The objective function score is averaged over all successfully processed test instances. Lower plot: The number of users who submitted their solutions during that particular day. The orange bars denote the total numbers of users; the blue bars denote only new users who submitted their solutions on that particular day for the first time.

person is not that important. Therefore, the participants could refine their solutions during the ongoing competition motivated by the online rank list presenting the best solution of each contestant evaluated on each instance. Such an approach allows crowdsourcing to generate much better solutions than during challenges in which participants submit a single solution at the end of the contest. The objective function ratio of the best solution in the current time to the best counterpart provided in the entire competition and the number of users participating in the contest over the days is presented in Figure 2. At the beginning of the competition, the solutions submitted by users were more than six times worse than the one that won it. However, in a relatively short amount of time (i.e., four days), the best solution was significantly improved (i.e., the best score in that time was two times better than the previous one). This is an expected behavior because, at the beginning of the optimization, the best of submitted solutions is usually relatively far from the global optimum, such that it can be easily improved by contestants. In the middle of this contest (i.e., a week after the beginning of the contest), the best solution score was close to the contest winner. It is worth mentioning that the breakthrough in the solution space was obtained by a small number of users, and then the final tuning of this solution occurred when the number of active users remarkably increased. The winning solution was submitted 52 minutes before the end of this contest. Despite that, the winning solution was significantly better than the initially submitted solution from January 23, and the improvement during the second half of the contest, between February 2 and 13, was not so significant. Nevertheless, it was still equal to almost 13%. It is worth underlining that, while optimizing industrial processes (e.g., in logistics), such a decrease of costs can potentially generate savings on the order of millions of dollars.

In turn, Figure 3 presents the number of correct and incorrect solutions submitted by users. The solution was classified as incorrect when it generated the wrong answer or raised execution errors (i.e., TLE, MLE, RE, or WA; see Section 3.2). At the beginning of the competition, the number of submitted solutions classified as potentially incorrect significantly overwhelmed the number of correct ones. However, the contestants quickly learned how to successfully submit their solutions, and thus, at the end of the competition, the number of submitted solutions classified as potentially

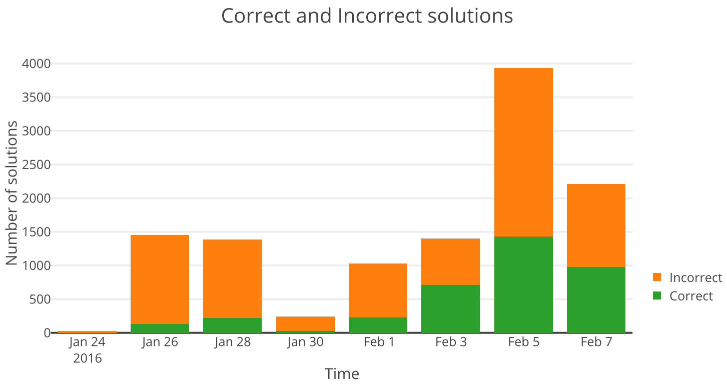

Fig. 3. The numbers of correct and incorrect solutions submitted during the consecutive days of the contest.

correct oscillated approximately 50%. Such an observation related to the ratio of correct and incorrect solutions confirmed the results characterizing other online judge systems reported by Manzoor (2006). This proves that the proposed GUI is ergonomic as well as user-friendly. The approximately 60% of invalid submissions presented in Figure 3 are primarily the consequence of two reasons. First, usually, users do not test their code sufficiently before submission. They assume that it is the role of the online judge system to verify their code and submit an algorithm that generates incorrect answers for some instances or that it does not compile. Second, the users do not comply with the execution limitations described in the challenge definition. Such solutions are terminated by the online judge system, receiving TLE status at the same time.

After five days of competition, the observed improvement slowed significantly unless a significantly larger number of participants joined the competition. This observation demonstrates perfectly the advantage of crowdsourcing. A crowd of highly skilled participants can achieve much better quality results than a single developer or even a small team. The results of the contest organized using Optil.io platform proved that it could be successfully applied to solve optimization problems interactively by a large number of programming enthusiasts.

5 CONCLUSION

Online judge systems have a very long history, lasting more than half a century. However, their popularization and widespread use have been empowered over the first decade of the twenty-first century. These systems have emerged as simple web applications provided by universities to support lecturers in teaching and students in preparation and participation in competitive programming championships. Meanwhile, the organizations hosting such programming competitions became interested in the application of online judge systems to share challenges formulated in the past to help users prepare for upcoming events. Thus, many online platforms emerged to publish challenges that originated from competitive programming contests. The primary reason was the rapid spread of Internet access, which gave many contestants worldwide an opportunity to easily submit and verify their solutions.

A majority of these systems are open for users worldwide who want to improve their algorithmic and programming skills. Moreover, there are a few that support the organization of customized contests addressed to the public or a limited group of participants, solving challenges originating from local as well as remote archives. To strengthen their popularity, several systems provide independent, localized instances for particular nationalities. The contest organization procedure often varies among the systems in time duration and number of challenges considered in a single round, number of provided programming languages, and even participant groups. However, in many, the inspiration from ICPC or IOI contests is noticeable. The contests are usually organized annually, a few times a year, or on demand of the challenge organizer. For example, Facebook and Google organize their own competitions annually. Sometimes participants are scored not only for the quality of their solutions, but also for successful identification of issues in other participant submissions. Usually, to distinguish submissions that are characterized by the same quality, the final score considers also the submission time or even the number of participants or groups that have submitted that solution previously. There are also other websites that present less classical applications of online judges designed to host domain-specific challenges or support only a textual representation of the output for computationally intensive problems. It is worth noting that there exist systems that support the selection of appropriate problems within a tremendous set of published challenges or even meta systems that are integrated with many online judges, thus allowing the user to submit his or her solution from a single place. A distinguishable feature of competitive programming systems is usually a simple GUI that only supports the most popular, low-level programming languages such as C/C++ and Pascal.

Online judges are also used in rather more sophisticated applications, such as education, employee recruitment, and even data mining. Online judge systems are of interest for educational organizations because they support fully automatic and accurate evaluation of student assignments, allowing the teacher to focus on exercise quality and the teaching process. We previously mentioned certain successful applications of these systems in the teaching of IT courses at various universities. Currently, the systems that allow users to strengthen their commitment and motivation by incorporating additional elements of gamification and social networking are significantly gaining popularity. For example, students are strengthening their coding skills not by solving typical exercises, but by developing specialized codes to protect their own units or attack enemies in a multiplayer strategy game. On the other side, there are also tools providing programming courses and professional tutorials for computer science enthusiasts (e.g., cracking passwords using the assembly language). It is worth remarking that educational online judge systems usually provide a rich, ergonomic GUI and gamification elements to engage students in the problem-solving process. The number of programming languages supported by such systems is also much higher in comparison to competitive programming systems, so that the focus is mainly on the educational process, independently of the programming language that is taught.

To ensure completeness of the presented classifications, we also considered online compilers because they share common mechanisms used to compile and execute source code submitted by the user to online judge systems. Moreover, they are often used as a crucial component in the development of online judge systems. In this group, we can find a wide range of tools, from simple online services supporting compilation and execution of user-submitted codes to systems providing cloud-based IDEs that support collaboration in real time by sharing a dedicated virtual development environment; however, a common feature is the ability to support users in solving programming assignments using only a web browser, without access to the standalone software. Such systems are especially useful for users who usually work with various workstations without administrative privileges.

The platforms proposed to support the recruitment process usually integrate and extend basic mechanisms provided by online judge systems. These systems allow developers to strengthen their skills by competing in various programming competitions, thus building their professional reputation within the community. They are also very useful for recruiters who can easily find many reliable employment candidates. A few of them allow even for delivery of fully customized challenges for specific candidates and their assessment in real time. All these systems are characterized by a professional GUI and enhanced functionality that allow for an analysis of programming skills based on traced interactions of the person being recruited using the system. Supported programming languages are usually limited to the most popular in the industry, such as C/C++, C#, Java, JavaScript, Python, and PHP.

There are also successful applications of online judge systems in platforms supporting the organization of data mining challenges. These are dedicated to solving complex industry- as well as science-inspired problems. The common usage scenario for these systems is that users submit the output of their programs obtained for specific test instances that are then assessed by the particular challenge organizers, and a final score is computed. However, it is worth noting that the user solutions can be rarely submitted in other forms, for example, by providing the algorithm's source code, or even in specialized reports summarizing the result. The number of challenges provided by these systems is growing extremely quickly.

The approach offered by online judge platforms has such a large impact that the concept they utilize has already been named *cloud-based evaluation*, or, strictly following the cloud computing naming scheme, *Evaluation-as-a-Service (EaaS).* At least two international workshops devoted to this topic were organized in 2015: Workshop on cloud-based evaluation approaches in the United States (Müller et al. 2016) and Evaluation-as-a-Service Expert Workshop in Switzerland (Hopfgartner et al. 2015). There is also the Metaheuristics in the Large initiative that integrates around this topic many prominent researchers from the field of operational research (Swan et al. 2015). The objective is to provide computing infrastructure that allows enthusiasts or researchers to incorporate state-of-the-art metaheuristics and solvers into an optimization workflow.

Unfortunately, most of the aforementioned systems are closed source. Thus, it is hard to apply them successfully, considering various user expectations. That is why, last but not least, we mentioned those systems that can be downloaded and installed on the user's own infrastructure by practitioners to provide an instance of a highly configurable online judge system. This can be easily applied to the organization of their own programming competition, as well as to support the educational process through the automated evaluation of student programs. Thus, such systems are of great interest to universities and organizations providing e-learning courses.

Currently, the systems providing problem archives for participants in competitive programming contests constitute the majority of all online judge systems because the development of these systems started approximately 10 years earlier than the remaining ones. However, in the near future, this proportion will certainly invert as a result of the intensive development of cloud-based computing (Puthal et al. 2015), which forms the basis for systems designed with EaaS architecture in mind.

The classification proposed in this article is addressed to all interested practitioners to support them in finding the appropriate online judge platform regardless of their needs. A classification is the most important contribution distinguishing this survey from earlier reviews. For example, Ihantola et al. described principles that are crucial in the design of online judge systems just for assessment of programming assignment (Ihantola et al. 2010). Another example is the article by Németh and László, which focused on a classification of online judge systems that only supports the organization of programming contests (Németh and László 2015).

Moreover, we formally defined what an online judge system is, and we described an evaluation procedure commonly used by such systems. While the intuitive perception of an online judge system was first introduced by Kurnia et al. (2001), a formal definition of such a system remained undefined until now. Similarly, the evaluation procedure, mentioned several times especially in relation to ACM ICPC and IOI contests (Cormack et al. 2006), was only described informally, with an emphasis on the context of the particular programming contest.

Additionally, we presented an example use case based on the Optil.io platform, which is the first system, worldwide, providing a continuous evaluation of algorithms solving complex optimization challenges in a safe and homogenous cloud-based infrastructure. Optimization problems are problems significant from the point of view of possible applications, as well as often difficult to solve problems that are formulated in a very broad range of fields from logistics (Silva et al. 2008) and decision-making (Blazewicz et al. 2014; Blazewicz and Musial 2011; Marler and Arora 2004), through software design (Marszalkowski et al. 2015), to even biology (Antczak et al. 2016; Lukasiak et al. 2015; Prejzendanc et al. 2016; Szostak et al. 2016; Wang et al. 2010; Wasik et al. 2013). Despite the great development of mathematical programming solvers and meta-heuristics that has occurred in recent years, many of these problems are still difficult to solve. These are NP-hard problems that can be solved by currently known methods for small instances or some special cases only. According to the "no free lunch" theorem applied for optimization purposes (Wolpert and Macready 1997), designing well-performing algorithms for such problems requires the application of sophisticated approaches for each of them. The experiment conducted using the Optil.io platform demonstrated that online judge systems can significantly enhance this process by providing a reliable platform that can be applied to effectively discover the solutions of science- and industry-inspired optimization problems using a crowdsourcing approach. We are sure that Optil.io and other online judge systems can assist users in utilizing the power of crowdsourcing to solve the most difficult problems known in science and industry.

REFERENCES

Kirsti M. Ala-Mutka. 2005. A survey of automated assessment approaches for programming assignments. *Computer Science Education* 15, 2 (Jun 2005), 83–102. DOI: http://dx.doi.org/10.1080/08993400500150747

Maciej Antczak, Marta Kasprzak, Piotr Lukasiak, and Jacek Blazewicz. 2016. Structural alignment of protein descriptors – a combinatorial model. *BMC Bioinformatics* 17, 1 (Sep 2016), 383. DOI: http://dx.doi.org/10.1186/s12859-016-1237-9

Paolo Antonucci, Christian Estler, Durica Nikolić, Marco Piccioni, and Bertrand Meyer. 2015. An incremental hint system for automated programming assignments. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 320–325. DOI: http://dx.doi.org/10.1145/2729094.2742607

Christian Artigues, Eric Bourreau, H. Murat Afsar, Olivier Briant, and Mourad Boudia. 2012. Disruption management for commercial airlines: Methods and results for the ROADEF 2009 Challenge. *European Journal of Industrial Engineering* 6, 6 (2012), 669. DOI: http://dx.doi.org/10.1504/ejie.2012.051072

Frances Bell. 2011. Connectivism: Its place in theory-informed research and innovation in technology-enabled learning. *International Review of Research in Open and Distributed Learning* 12, 3 (Mar 2011). DOI: http://dx.doi.org/10.19173/irrodl.v12i3.902

Steve Benford, Edmund Burke, and Eric Foxley. 1993. Learning to construct quality software with the Ceilidh system. *Software Quality Journal* 2, 3 (Sep 1993), 177–197. DOI: http://dx.doi.org/10.1007/bf00402268

Jacek Blazewicz, Nathanael Cheriere, Pierre-Francois Dutot, Jedrzej Musial, and Denis Trystram. 2014. Novel dual discounting functions for the Internet shopping optimization problem: New algorithms. *Journal of Scheduling* 19, 3 (Aug 2014), 245–255. DOI: http://dx.doi.org/10.1007/s10951-014-0390-0

Jacek Blazewicz and Jedrzej Musial. 2011. E-commerce evaluation: Multi-item internet shopping. optimization and heuristic algorithms. In *Operations Research Proceedings*. Springer, Berlin, 149–154. DOI: http://dx.doi.org/10.1007/978-3-642-20009-0_24

Julio C. Caiza and José María del Álamo Ramiro. 2013. Programming assignments automatic grading: Review of tools and implementations. In *Proceedings of the 7th International Technology, Education and Development Conference (INTED2013)*. 5691–5700. http://oa.upm.es/25765/

Manuel M. Carral. 2013. *Moodle Autograder Plugin*. Master's thesis. Universidad de Cantabria.

Brenda Cheang, Andy Kurnia, Andrew Lim, and Wee-Chong Oon. 2003. On automated grading of programming assignments in an academic institution. *Computers & Education* 41, 2 (Sep 2003), 121–131. DOI: http://dx.doi.org/10.1016/s0360-1315(03)00030-7

Amanda Coles, Andrew Coles, Angel G. Olaya, Sergio Jiménez, Carlos L. López, Scott Sanner, and Sungwook Yoon. 2012. A survey of the seventh international planning competition. *AI Magazine* 33, 1 (2012), 83–88. https://vvvvw.aaai.org/ojs/index.php/aimagazine/article/view/2392.

Sébastien Combéfis and Jérémy Wautelet. 2014. Programming trainings and informatics teaching through online contests. *Olympiads in Informatics* 8 (2014), 21–34. http://www.mclibre.org/descargar/docs/revista-oii/oii-08_201407.pdf.

Gordon V. Cormack, J. Ian Munro, Troy Vasiga, and Graeme Kemkes. 2006. Structure, scoring and purpose of computing competitions. *Informatics in Education* 5, 1 (2006), 15–36.

J. C. Costello and G. Stolovitzky. 2013. Seeking the wisdom of crowds through challenge-based competitions in biomedical research. *Clinical Pharmacology & Therapeutics* 93, 5 (Feb 2013), 396–398. DOI:http://dx.doi.org/10.1038/clpt.2013.36

Carles Creus and Guillem Godoy. 2014. Automatic evaluation of context-free grammars (system description). In *Lecture Notes in Computer Science*. Springer International, 139–148. DOI:http://dx.doi.org/10.1007/978-3-319-08918-8_10

Valentina Dagiene, Arturo Cepeda, Richard Forster, and Krassimir Manev. 2007. Editorial. *Olympiads in Informatics* 1 (2007), 3–4. https://www.mii.lt/olympiads_in_informatics/files/volume1.pdf.

Charlie Daly. 1999. RoboProf and an introductory computer programming course. In *Proceedings of the 4th Annual SIGCSE/SIGCUE ITiCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE'99)*. ACM Press. DOI:http://dx.doi.org/10.1145/305786.305904

Karol Danutama and Inggriani Liem. 2013. Scalable autograder and LMS integration. *Procedia Technology* 11 (2013), 388–395. DOI:http://dx.doi.org/10.1016/j.protcy.2013.12.207

Prathibha Davuluri and Pranavi R. Madadi. 2016. *Moodle Java Autograder*. Technical Report. Governors State University.

Ross Dawson and Steve Bynghall. 2012. *Getting Results from Crowds*. Advanced Human Technologies San Francisco, CA.

Jonathan M. J. Derry, Lara M. Mangravite, Christine Suver, Matthew D. Furia, David Henderson, Xavier Schildwachter, Brian Bot, Jonathan Izant, Solveig K. Sieberts, Michael R. Kellen, and Stephen H. Friend. 2012. Developing predictive molecular maps of human disease through community-based modeling. *Nature Genetics* 44, 2 (Jan 2012), 127–130. DOI:http://dx.doi.org/10.1038/ng.1089

Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. 2011. From game design elements to gamefulness. In *Proceedings of the 15th International Academic MindTrek Conference on Envisioning Future Media Environments (MindTrek'11)*. ACM Press. DOI:http://dx.doi.org/10.1145/2181037.2181040

Vasant Dhar. 2013. Data science and prediction. *Communications of the ACM* 56, 12 (Dec 2013), 64–73. DOI:http://dx.doi.org/10.1145/2500499

Cheedoong Drung, Jianwen Wang, and Ning 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 & Mechanical Engineering and Information Technology*. IEEE. DOI:http://dx.doi.org/10.1109/emeit.2011.6024016

Jack Edmonds. 1965. Paths, trees and flowers. *Canadian Journal of Mathematics* 17 (1965), 449–467.

Stephen H. Edwards and Manuel A. Perez-Quinones. 2008. Web-CAT: Automatically grading programming assignments. In *ACM SIGCSE Bulletin*, Vol. 40. ACM, 328. DOI:http://dx.doi.org/10.1145/1597849.1384371

Arpad E. Elo. 1978. *The Rating of Chessplayers, Past and Present.* Arco Pub.

Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. 2015. An updated performance comparison of virtual machines and Linux containers. In *Proceedings of the 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE. DOI:http://dx.doi.org/10.1109/ispass.2015.7095802

Daniela Fonte, Daniela da Cruz, Alda L. Gançarski, and Pedro R. Henriques. 2013. A flexible dynamic system for automatic grading of programming exercises. In *Proceedings of the 2nd Symposium on Languages, Applications and Technologies (OpenAccess Series in Informatics (OASIcs))*, José Paulo Leal, Ricardo Rocha, and Alberto Simões (Eds.), Vol. 29. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 129–144. DOI:http://dx.doi.org/10.4230/OASIcs.SLATE.2013.129

Michal Forišek. 2006a. On the suitability of programming tasks for automated evaluation. *Informatics in Education* 5, 1 (2006), 63–76.

Michal Forišek. 2006b. Security of programming contest systems. *Information Technologies at School* (2006), 553–563.

George E. Forsythe and Niklaus Wirth. 1965. Automatic grading programs. *Communications of the ACM* 8, 5 (may 1965), 275–278. DOI:http://dx.doi.org/10.1145/364914.364937

Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W. H. Freeman, New York.

Anthony Goldbloom. 2010. Data prediction competitions: Far more than just a bit of fun. In *Proceedings of the2010 IEEE International Conference on Data Mining Workshops*. IEEE. DOI:http://dx.doi.org/10.1109/icdmw.2010.56

Frank Hopfgartner, Jimmy Lin, Krisztian Balog, Ivan Eggel, Allan Hanbury, Henning Müller, Noriko Kando, Simon Mercer, Jayashree Kalpathy-Cramer, Martin Potthast, Tim Gollub, and Anastasia Krithara. 2015. Report on the evaluation-as-a-service (EaaS) expert workshop. *ACM SIGIR Forum* 49, 1 (Jun 2015), 57–65. DOI:http://dx.doi.org/10.1145/2795403.2795416

Jeff Howe. 2006. The rise of crowdsourcing. *Wired Magazine* 14, 6 (2006), 1–4.

B. Husemann, J. Lechtenborger, G. Vossen, and P. Westerkamp. XLX-A platform for graduate-level exercises. In *Proceedings of the 2002 International Conference on Computers in Education.* IEEE Computing Society. DOI: http://dx.doi.org/10.1109/cie.2002.1186207

ICPC Foundation. 2016. ICPC Fact Sheet. Retrieved from https://icpc.baylor.edu/worldfinals/pdf/Factsheet.pdf.

Petri Ihantola, Tuukka Ahoniemi, Ville Karavirta, and Otto Seppälä. 2010. Review of recent systems for automatic assessment of programming assignments. In *Proceedings of the 10th Koli Calling International Conference on Computing Education Research (Koli Calling'10).* ACM Press. DOI: http://dx.doi.org/10.1145/1930464.1930480

Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstle, Stephen H. Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, Miguel Ángel Rubio, Judy Sheard, Bronius Skupas, Jaime Spacco, Claudia Szabo, and Daniel Toll. 2015. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports.* ACM, 41–63. DOI: http://dx.doi.org/10.1145/2858796.2858798

Thomas Ilsche, Joseph Schuchart, Robert Schöne, and Daniel Hackenberg. 2015. Combining instrumentation and sampling for trace-based application performance analysis. In *Tools for High Performance Computing 2014.* Springer International, 123–136. DOI: http://dx.doi.org/10.1007/978-3-319-16012-2_6

Slavina Ivanova. 2016. Learning computer programming through games development. In *The International Scientific Conference eLearning and Software for Education*, Vol. 1. Carol I National Defence University, 492–497.

Mike Joy, Nathan Griffiths, and Russell Boyatt. 2005. The boss online submission and assessment system. *Journal on Educational Resources in Computing* 5, 3 (Sep 2005), 2–es. DOI: http://dx.doi.org/10.1145/1163405.1163407

Mike Joy and Michael M. Luck. 1995. On-line submission and testing of programming assignments. In *Innovations in the Teaching of Computing*, J. Hart (Ed.). SEDA Papers, Vol. Volume 1. SEDA, London, 95–103.

Andreas M. Kaplan and Michael Haenlein. 2016. Higher education and the digital revolution: About MOOCs, SPOCs, social media, and the cookie monster. *Business Horizons* 59, 4 (Jul 2016), 441–450. DOI: http://dx.doi.org/10.1016/j.bushor.2016.03.008

Ville Karavirta, Petri Ihantola, and Teemu Koskinen. 2013. Service-oriented approach to improve interoperability of e-learning systems. In *Proceedings of the 2013 IEEE 13th International Conference on Advanced Learning Technologies.* IEEE, 341–345. DOI: http://dx.doi.org/10.1109/ICALT.2013.105

Mümine Kaya and Selma A. Özel. 2012. An online compiler module for grading programming assignments on Moodle distance education system. *Global Journal of Technology* 1 (2012), 715–720.

Mümine Kaya and Selma Ayşe Özel. 2014. Integrating an online compiler and a plagiarism detection tool into the Moodle distance education system for easy assessment of programming assignments. *Computer Applications in Engineering Education* 23, 3 (Jul 2014), 363–373. DOI: http://dx.doi.org/10.1002/cae.21606

Software Engineering Group, Keele University. 2007. Guidelines for performing systematic literature reviews in software engineering. In *Technical Report, Ver. 2.3 EBSE Technical Report. EBSE.* sn. Retrieved https://pdfs.semanticscholar.org/e62d/bbbbe70cabcde3335765009e94ed2b9883d5.pdf.

Vivek Khera, Owen Astrachan, and David Kotz. 1993. The internet programming contest. *ACM SIGCSE Bulletin* 25, 1 (mar 1993), 48–52. DOI: http://dx.doi.org/10.1145/169073.169105

Barbara Kitchenham, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. Systematic literature reviews in software engineering: A systematic literature review. *Information and Software Technology* 51, 1 (2009), 7–15. DOI: http://dx.doi.org/10.1016/j.infsof.2008.09.009

Rob Kolstad and Don Piele. 2007. USA Computing Olympiad (USACO). *Olympiads in Informatics* 1 (2007), 105–111.

Rita Kop. 2011. The challenges to connectivist learning on open online networks: Learning experiences during a massive open online course. *The International Review of Research in Open and Distributed Learning* 12, 3 (Mar 2011), 19. DOI: http://dx.doi.org/10.19173/irrodl.v12i3.882

Adrian Kosowski, Michał Małafiejski, and Tomasz Noiński. 2007. Application of an online judge & contester system in academic tuition. In *Lecture Notes in Computer Science.* Springer Berlin, 343–354. DOI: http://dx.doi.org/10.1007/978-3-540-78139-4_31

Andy Kurnia, Andrew Lim, and Brenda Cheang. 2001. Online judge. *Computers & Education* 36, 4 (May 2001), 299–315. DOI: http://dx.doi.org/10.1016/s0360-1315(01)00018-5

José P. Leal and Nelma Moreira. 1998. *Automatic Grading of Programming Exercises.* Technical Report. Universidade do Porto.

José P. Leal and Fernando Silva. 2003. Mooshak: A Web-based multi-site programming contest system. *Software: Practice and Experience* 33, 6 (2003), 567–581. DOI: http://dx.doi.org/10.1002/spe.522

LLC Books. 2010. *ISO 639: List of ISO 639-3 Codes, ISO 639 Macrolanguage, ISO 639: K, ISO 639: M, ISO 639: B, ISO 639: A, ISO 639: T, ISO 639-1.* General Books LLC.

Piotr Lukasiak, Maciej Antczak, Tomasz Ratajczak, Marta Szachniuk, Mariusz Popenda, Ryszard W. Adamiak, and Jacek Blazewicz. 2015. RNAssess—a web server for quality assessment of RNA 3D structures. *Nucleic Acids Research* 43, W1 (Jun 2015), W502–W506. DOI: http://dx.doi.org/10.1093/nar/gkv557

Yingwei Luo, Xiaolin Wang, and Zhengyi Zhang. 2008. Programming grid. In *Proceedings of the 1st ACM Summit on Computing Education in China (SCE'08)*. ACM Press. DOI: http://dx.doi.org/10.1145/1517632.1517643

Anaga Mani, Divya Venkataramani, Jordi Petit, and Salvador Roura. 2014. Better feedback for educational online judges. In *Proceedings of the 6th International Conference on Computer Supported Education (SCITEPRESS'14)*. Science and and Technology Publications. DOI: http://dx.doi.org/10.5220/0004842801760183

Shahriar Manzoor. 2006. Analyzing programming contest statistics. In *Proceedings of Perspectives on Computer Science Competitions for (High School) Students Workshop*. https://www.bwinf.de/competition-workshop/papers.html.

R. T. Marler and J. S. Arora. 2004. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization* 26, 6 (Apr 2004), 369–395. DOI: http://dx.doi.org/10.1007/s00158-003-0368-6

Jakub Marszalkowski, Jan Mizgajski, Dariusz Mokwa, and Maciej Drozdowski. 2015. Analysis and solution of CSS-sprite packing problem. *ACM Transactions on the Web* 10, 1 (Dec 2015), 1–34. DOI: http://dx.doi.org/10.1145/2818377

Drew Mcdermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. *PDDL: The Planning Domain Definition Language*. Technical Report. CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Nimrod Megiddo and Arie Tamir. 1982. On the complexity of locating linear facilities in the plane. *Operations Research Letters* 1, 5 (Nov 1982), 194–197. DOI: http://dx.doi.org/10.1016/0167-6377(82)90039-6

Dirk Merkel. 2014. Docker: Lightweight Linux containers for consistent development and deployment. *Linux Journal*. 2014, 239, Article 2 (Mar 2014), 76–91.

Henning Müller, Thea Norman, David Kennedy, Ganapati Srinivasa, Artem Mamonov, Nina Preuss, Jayashree Kalpathy-Cramer, Allan Hanbury, Keyvan Farahani, Rinat Sergeev, Jin H. Paik, Arno Klein, Antonio Criminisi, and Andrew Trister. 2016. Report on the cloud-based evaluation approaches workshop 2015. *ACM SIGIR Forum* 50, 1 (Jun 2016), 38–41. DOI: http://dx.doi.org/10.1145/2964797.2964804

Ágnes E. Németh and Zsakó László. 2015. Online training and contests for informatics contestants of secondary school age. *Edukacja-Technika-Informatyka* 6, 1 (2015), 273–280.

Jordi Petit, Omer Giménez, and Salvador Roura. 2012. Jutge.org. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education (SIGCSE'12)*. ACM Press. DOI: http://dx.doi.org/10.1145/2157136.2157267

Vreda Pieterse. 2013. Automated assessment of programming assignments. In *Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research (CSERC'13)*. Open Universiteit, Heerlen, The Netherlands, Article 4, 12 pages.

Wolfgang Pohl. 2006. Computer science contests for secondary school students: Approaches to classification. *Informatics in Education* 5, 1 (2006), 125–132. https://www.ceeol.com/search/article-detail?id=151845 Copyright Institute of Mathematics and Informatics 2006; Document feature; Last updated 2011-06-03.

Tomasz Prejzendanc, Szymon Wasik, and Jacek Blazewicz. 2016. Computer representations of bioinformatics models. *Current Bioinformatics* 11, 5 (2016), 551–560. DOI: http://dx.doi.org/10.2174/1574893610666150928193510

John Prpić, James Melton, Araz Taeihagh, and Terry Anderson. 2015. MOOCs and crowdsourcing: Massive courses and massive resources. *First Monday* 20, 12 (Dec 2015). DOI: http://dx.doi.org/10.5210/fm.v20i12.6143

Deepak Puthal, B. P. S. Sahoo, Sambit Mishra, and Satyabrata Swain. 2015. Cloud computing features, issues, and challenges: A big picture. In *Proceedings of the 2015 International Conference on Computational Intelligence and Networks*. IEEE. DOI: http://dx.doi.org/10.1109/cine.2015.31

Martin Pärtel, Matti Luukkainen, Arto Vihavainen, and Thomas Vikberg. 2013. Test my code. *International Journal of Technology Enhanced Learning* 5, 3–4 (2013), 271–283. DOI: http://dx.doi.org/10.1504/IJTEL.2013.059495

Miguel A. Revilla, Shahriar Manzoor, and Rujia Liu. 2008. Competitive learning in informatics: The UVa online judge experience. *Olympiads in Informatics* 2 (2008), 131–148.

Juan C. Rodríguez-del Pino, Enrique Rubio Royo, and Zenón Hernández Figueroa. 2012. A virtual programming lab for moodle with automatic assessment and anti-plagiarism features. In *Proceedings of the 2012 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government.*

Rohaida Romli, Shahida Sulaiman, and Kamal Z. Zamli. 2010. Automatic programming assessment and test data generation a review on its approaches. In *Proceedings of the 2010 International Symposium on Information Technology*, Vol. 3. IEEE, 1186–1192. DOI: http://dx.doi.org/10.1109/ITSIM.2010.5561488

Samira Saedi and O. Erhun Kundakcioglu. 2013. *Combinatorial Optimization in Data Mining*. Springer New York, 595–630. DOI: http://dx.doi.org/10.1007/978-1-4419-7997-1_7

Julio Saez-Rodriguez, James C. Costello, Stephen H. Friend, Michael R. Kellen, Lara Mangravite, Pablo Meyer, Thea Norman, and Gustavo Stolovitzky. 2016. Crowdsourcing biomedical research: Leveraging communities as innovation engines. *Nature Reviews Genetics* 17, 8 (Jul 2016), 470–486. DOI: http://dx.doi.org/10.1038/nrg.2016.69

Saul Schleimer, Daniel S. Wilkerson, and Alex Aiken. 2003. Winnowing. In *Proceedings of the 2003 ACM SIGMOD International Conference on on Management of Data (SIGMOD'03*). ACM Press. DOI:http://dx.doi.org/10.1145/872757.872770

C. A. Silva, J. M. C. Sousa, and T. A. Runkler. 2008. Rescheduling and optimization of logistic processes using GA and ACO. *Engineering Applications of Artificial Intelligence* 21, 3 (Apr 2008), 343–352. DOI:http://dx.doi.org/10.1016/j.engappai.2007.08.006

Steven S. Skiena and Miguel A. Revilla. 2008. *Programming Challenges: The Programming Contest Training Manual (Texts in Computer Science).* Springer.

Jaime Spacco, Paul Denny, Brad Richards, David Babcock, David Hovemeyer, James Moscola, and Robert Duvall. 2015. Analyzing student work patterns using programming exercise data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE'15).* ACM Press. DOI:http://dx.doi.org/10.1145/2676723.2677297

Thomas Staubitz, Hauke Klement, Jan Renz, Ralf Teusner, and Christoph Meinel. 2015. Towards practical programming exercises and automated assessment in massive open online courses. In *Proceedings of the 2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE'15).* IEEE, 23–30. DOI:http://dx.doi.org/10.1109/TALE.2015.7386010

Jerry Swan, Steven Adriaensen, Mohamed Bishr, Edmund K. Burke, John A. Clark, Patrick De Causmaecker, Juanjo Durillo, Kevin Hammond, Emma Hart, Colin G. Johnson, Zoltan A. Kocsis, Ben Kovitz, Krzysztof Krawiec, Simon Martin, J. J. Merelo, Leandro L. Minku, Ender Ozcan, Gisele L. Pappa, Erwin Pesch, Pablo Garca-Sánchez, Andrea Schaerf, Kevin Sim, Jim E. Smith, Thomas Stützle, Stefan Voß, Stefan Wagner, and Xin Yao. 2015. A research agenda for metaheuristic standardization. In *MIC 2015: The XI Metaheuristics International Conference.* http://www.cs.nott.ac.uk/~pszeo/docs/publications/research-agenda-metaheuristic.pdf.

Natalia Szostak, Szymon Wasik, and Jacek Blazewicz. 2016. Hypercycle. *PLOS Computational Biology* 12, 4 (04 2016), 1–13. DOI:http://dx.doi.org/10.1371/journal.pcbi.1004853

Jan N. van Rijn, Bernd Bischl, Luis Torgo, Bo Gao, Venkatesh Umaashankar, Simon Fischer, Patrick Winter, Bernd Wiswedel, Michael R. Berthold, and Joaquin Vanschoren. 2013. OpenML: A collaborative science platform. In *Machine Learning and Knowledge Discovery in Databases.* Springer Berlin, 645–649. DOI:http://dx.doi.org/10.1007/978-3-642-40994-3_46

Urs von Matt. 1994. Kassandra. *ACM SIGCUE Outlook* 22, 1 (Jan 1994), 26–40. DOI:http://dx.doi.org/10.1145/182107.182101

Yanqing Wang, Xiaolei Wang, Yu Jiang, Yaowen Liang, and Ying Liu. 2016. A code reviewer assignment model incorporating the competence differences and participant preferences. *Foundations of Computing and Decision Sciences* 41, 1 (Jan 2016). DOI:http://dx.doi.org/10.1515/fcds-2016-0004

Yong Wang, Xiang-Sun Zhang, and Luonan Chen. 2010. Optimization meets systems biology. *BMC Systems Biology* 4, Suppl 2 (2010), S1. DOI:http://dx.doi.org/10.1186/1752-0509-4-s2-s1

Szymon Wasik, Maciej Antczak, Jan Badura, Artur Laskowski, and Tomasz Sternal. 2016. Optil.io: Cloud based platform for solving optimization problems using crowdsourcing approach. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion.* ACM, 433–436. DOI:http://dx.doi.org/10.1145/2818052.2869098

Szymon Wasik, Filip Fratczak, Jakub Krzyskow, and Jaroslaw Wulnikowski. 2015. Inferring mathematical equations using crowdsourcing. *PLOS ONE* 10, 12 (Dec 2015), e0145557. DOI:http://dx.doi.org/10.1371/journal.pone.0145557

Szymon Wasik, Tomasz Prejzendanc, and Jacek Blazewicz. 2013. ModeLang: A new approach for experts-friendly viral infections modeling. *Computational and Mathematical Methods in Medicine* 2013 (2013), 8. DOI:http://dx.doi.org/10.1155/2013/320715

Li Wen-xin and Guo Wei. 2005. Peking university online judge and its applications. *Journal of Changchun Post and Telecommunication Institute* (2005), 170–177.

Chris Wilcox. 2016. Testing strategies for the automated grading of student programs. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education.* ACM, 437–442. DOI:http://dx.doi.org/10.1145/2839509.2844616

Marcin Wojnarski, Sebastian Stawicki, and Piotr Wojnarowski. 2010. TunedIT.org: System for automated evaluation of algorithms in repeatable experiments. In *Rough Sets and Current Trends in Computing.* Springer Berlin, 20–29. DOI:http://dx.doi.org/10.1007/978-3-642-13529-3_4

D. H. Wolpert and W. G. Macready. 1997. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* 1, 1 (Apr 1997), 67–82. DOI:http://dx.doi.org/10.1109/4235.585893

Joao Xavier and António F. Coelho. 2011. Computer-based assessment system for e-learning applied to programming education. In *ICERI'11 Proceedings.* IATED, 3738–3747.

Chao Yi, Su Feng, and Zhi Gong. 2014. A comparison of sandbox technologies used in online judge systems. *Applied Mechanics and Materials* 490-491 (Apr 2014), 1201–1204.

Yongwook Yoon, Changki Lee, and Gary Geunbae Lee. 2006. An effective procedure for constructing a hierarchical text classification system. *Journal of the American Society for Information Science and Technology* 57, 3 (2006), 431–442. DOI:http://dx.doi.org/10.1002/asi.20281

Ninghan Zheng, Shuzhen Tian, and Yongqiang Chen. 2015. Online learning management system. In *Proceedings of the 2015 International Conference on Computational Science and Computational Intelligence (CSCI'15)*. IEEE. DOI:http://dx.doi.org/10.1109/csci.2015.160

Sun Zhigang, Su Xiaohong, Zhu Ning, and Cheng Yanyu. 2012. Moodle plugins for highly efficient programming courses. In *Proceedings of the 1st Moodle Research Conference (MRC'12)*. 157–163. http://research.moodle.net/51/.

Guojin Zhu and Lichao Fu. 2012. Automatic organization of programming resources on the web. In *Advances in Intelligent and Soft Computing*. Springer Berlin, 675–681. DOI:http://dx.doi.org/10.1007/978-3-642-30126-1_106

Received January 2017; revised September 2017; accepted September 2017