Olympiads in Informatics, 2017, Vol. 11, 19–28

DOI: 10.15388/ioi.2017.02

Learning and Teaching Algorithm Design and Optimisation Using Contests Tasks

Sébastien COMBÉFIS, Saïkou Ahmadou BARRY, Martin CRAPPE, Mathieu DAVID, Guillaume de MOFFARTS, Hadrien HACHEZ, Julien KESSELS

Electronics and IT Unit, École Centrale des Arts et Métiers (ECAM)
Promenade de l'Alma 50, 1200 Woluwé-Saint-Lambert, Belgium
e-mail: s.combefis@ecam.be, saikouah@gmail.com, martin.crappe@gmail.com, mathieudavid@mathieudavid.org, guillaumedemoff@gmail.com, hadrienhachez@hotmail.com, julien.kessels@gmail.com

Abstract. It is important for a future computer science engineer or scientist to master algorithm design and to know how to optimise algorithms to solve real-world problems. Most programming and IT contests require their contestants to design algorithms to solve problems and to optimise their code to get the best temporal and spatial performances. This paper presents training materials built from contest tasks to teach and learn how to design algorithms that solve concrete and contextualised problems. The first learning modules will be built thanks to a pedagogical device that will be deployed during the 2017–2018 academic year at ECAM in the frame of the LADO project. All the produced materials will be open sourced and available in English.

Keywords: algorithm and optimisation; pedagogical device; learning modules.

1. Introduction

Computer science engineers and scientists must have skills in algorithm design and optimisation. While scientists may be more interested in a theoretical understanding of algorithms, their complexities and how to optimise them, engineers need to know how and which algorithm to apply for a given real-world problem.

One way to learn how to write optimised algorithms is to make experiments. Given an implementation of the algorithm, it can be executed on several instances of a problem to measure temporal and spatial performances. Different implementations can therefore be compared, to identify the best ones. Basic algorithms and related data structures can be easily learned with textbooks or other similar non-interactive resources. What is more difficult to learn is how to chose an algorithm and how to implement it to efficiently solve a given problem. It is even more difficult when dealing with real-world problems for which new algorithms have to be devised and optimised to compute a solution within a reasonable amount of time.

Programming contests and some other more general IT contests typically ask their contestants to solve one or several challenging tasks. In addition to finding a relevant algorithm, contestants must optimise their implementation to obtain the best running time and memory consumption to get a good position in the contest ranking. Contest tasks therefore provide good case studies to learn algorithm design and optimisation. But such tasks are not designed and suited for learning, as is. They are generally devised to be difficult to solve and so that a solution to solve them is not immediate.

This paper presents how learning materials can be built from contest tasks to teach algorithm design and optimisation. Various tasks from programming and more general IT contests are considered. The paper also presents a pedagogical device that will make students building the first learning modules. It will be settled the next academic year and all the produced materials will be open source.

The remainder of this paper is structured as follows. Section 2 briefly browses related works. Section 3 presents how tasks from contests are used to build materials suited for teaching and learning algorithm design and optimisation. Section 4 discusses the LADO project that will implement the proposed pedagogical device at ECAM during the 2017–2018 academic year. Finally, the last section concludes the paper with perspectives and future work.

2. Related Work

Online programming and IT contests can be used to build trainings to teach and learn programming skills as detailed in previous works. (Combéfis *et al.*, 2014) highlights the necessity to provide additional information to the contests tasks, to make them suited for teaching purposes. In particular, offering a feedback to the learners is very important to support their learning, as detailed in (Combéfis *et al.*, 2012). Finally, as highlighted in (Combéfis *et al.*, 2013), learning algorithm design is important for young people but also for students in higher education institutions, as future professionals. This paper builds on these previous works to propose concrete learning materials.

Some other works uses contests to teach and learn programming and other IT related skills. In (Garcia-Mateos, *et al.*, 2009), the authors explain how to make courses more entertaining to increase the motivation of students by using programming contests. The main difference with the work of this paper is that they are developing their own contest-style tasks fitted for education purposes. Evaluation of students' submissions is made by Mooshak, an on-line judging system used to manage programming competitions (Leal, *et al.*, 2003). As indicated by the authors in the conclusion, having a better feedback in case of a "wrong answer" has still to be done. Competitive programming should be introduced early as highlighted by several authors (Leal, *et al.*, 2008; Ribeiro, *et al.*, 2008) as it helps to get better skills in algorithm design and optimisation.

Finally, (Booth, 2001) discusses the need and importance of context while learning computer science and engineering. The author draws up theoretical foundations and considerations about the move to highly constructivist pedagogies in higher education, which is in line with the method proposed in this paper.

3. Teaching and Learning with Contests Tasks

This paper proposes learning materials based on tasks from programming and IT contests. It also presents a pedagogical device that will result in the production of these learning materials by a selected group of students. The developed learning modules will be about how to design and optimise algorithms. Contests tasks are used as illustrating and motivating real-world problems and also as case studies to assess the algorithms choice to solve the given problem and to test and experiment with their implementations.

As previously mentioned, this paper goes one step further than (Combéfis *et al.*, 2014). It proposes a concrete way to build learning materials from contest tasks, to teach one aspect of programming, namely algorithm design and optimisation.

3.1. *Algorithm Design and Optimisation*

As highlighted by (Skiena, 2008), most professional programmers are not prepared enough to tackle algorithm design problems. It is therefore important that future professionals, namely current students, are taught how to correctly and efficiently implement algorithms for real-world problems as testified by a survey presented in (Lethbridge, 2000).

Algorithm designers have to deal with several processes. According to (Kant, 1985), there are six main processes when designing an algorithm. The designer starts with the understanding of the problem and then proposes a kernel idea to solve the problem. The next steps consist in executing and analysing the algorithm in order to formulate any difficulties or opportunities to improve the current solution. Finally, the last two steps is firstly a verification of the correctness, that is, assessing whether the obtained result is as good solution for the problem and secondly an evaluation of efficiency and other quality criteria.

A simplified algorithm design and optimisation process is depicted on Fig. 1. The two first steps are similar to these of (Kant, 1985), namely the understanding of the problem from which a first kernel idea emerges, typically based on a brute-force algorithm. The kernel idea should also explore the basic data structures that can be used to model the problem. All these ideas are then implemented into a program before jumping to the next two steps. The third step consists in the execution and correctness analysis of the algorithm, which can be made on small instances of the problem with a manual check, with the help of an online grader if available or with a solution check program. Finally, in the last step, the algorithm is evaluated according to its efficiency and improvement points are identified. The process then loops between the two last steps until the obtained algorithm is efficient enough.

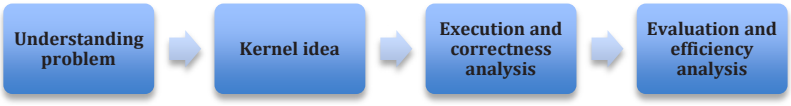


Fig. 1. The four main steps of algorithm design and optimisation process.

Students must be able to implement the algorithms they are taught thanks to the learning modules by themselves, for the situation provided by the contest task, but also for other new similar situations. It means that they should have a deep understanding of these algorithms. In particular, for a given algorithm, students must be able to:

1. Understand the algorithm, its precise specifications and all the used data structures the algorithm relies on.
2. Know the class of problems for which it is relevant to apply the algorithm.
3. Implement the algorithm.
4. Compare and test different implementations, in particular in terms of time and space complexities.

These needs are related to the four mains steps of Fig. 1 and are clearly delineated in the proposed training materials. During the last two steps, students must be able to justify all their choices and argue about why they are relevant. For that, they can use complexities analysis, research papers, reference books, etc.

For example, (Levitin, 1999) describes four general design techniques (brute force, divide-and-conquer, decrease-and-conquer and transform-and-conquer) that are qualified as more general than usual classification (greedy, dynamic, backtracking, branch-and-bound, etc.) which suffers from different levels of generality and fails to classify many classical algorithms. This classification can, for example, be used to compare different algorithms and ensure that several tracks have been explored.

Concerning the optimisation that can be done at each iteration of the last two steps, they can be done at different levels. Students must be aware of these possibilities along with their advantages and disadvantages. An algorithm can be mainly optimised by:

1. Selecting appropriate data structures.
2. Choosing the appropriate parameters and configuration for the algorithm.
3. Exploiting programming language related techniques.

It is important for the students to be able to test different optimisations. The contest task can be used as a direct case study for that. Some tasks are already prepared for such experiments, while some work has to be done for others.

3.2. *Tasks from Contests*

As detailed in (Combéfis, *et al.*, 2014), there is a great variety of IT contests, at least for online ones. Most of them involve choosing algorithms and writing programs. Whereas these two activities are obvious for programming contests, it is not necessarily obvious for some computer security contests, innovation contests, etc. but yet possibly existing. Of course, not all the IT related contest tasks can be used to build learning materials as proposed in this paper, since they should include a possible programming activity.

The three main kinds of tasks that are considered in this work are:

- Programming tasks where the contestant has to choose an algorithm and implement it to solve a problem;

- Security tasks for which the contestant has to analyse data, a protocol, a file, etc. with scripts and hack a system, decrypt a file, etc. with selected algorithms.
- Innovation tasks in the frame of which the contestant may be asked to implement a prototype of an application.

For programming tasks, contestants are often given instances of the problem they have to solve. These instances can exhibit two kinds of differences:

- They can be of gradual complexities: small ones can be solved by hand while large ones require an optimised algorithm so that to be solved within the imposed time and memory limits.
- They can have different structures: a single algorithm may not run efficiently for all the instances and instances peculiarities must be identified to correctly chose a relevant algorithm for each instance.

These kinds of tasks are the best ones for the proposed learning modules since they naturally drive their contestants towards algorithm optimisation.

Using security and innovation tasks is less obvious but not impossible since they often involve writing code based on an algorithm that has first to be identified. Also, such tasks are often better more directly related to a real-world issue to solve and could therefore be more motivating for students.

3.3. *Learning Modules*

Combining algorithms to be taught and their possible optimisations with contests tasks can result in the creation of learning modules, which is the purpose of this paper. The idea is to build learning materials, which is then used in classroom with students following the "learning by doing" of (Dewey, 1938). Several kinds of learning modules, with different learning objectives, can be designed:

- They can focus on one specific algorithm and possibly on how it can be tuned and optimised.
- They can present different types of algorithms that are used to solve one given class of problems, and make it possible to compare them.
- They can present one class of algorithms suited for the given problem and compare their performance.

One given contest task can be used in several learning modules with different objectives, but it can sometimes be more suited for one specific kind. On the contrary, one learning module should be limited to only one contest task. It makes it possible to ensure that the focus is put on a single context.

The structure of a learning module is always the same, regardless of its type. The description and material of a module contain the following elements:

1. An introduction to the learning module that draws up what the learning objectives are and what will be concretely learned.

2. A brief presentation of the contest task and any other additional information that can help the learners to better understand the problem, and in particular a precise specification and description of the given inputs and expected outputs.
3. A presentation of a brute force solution, and possibly some expected solutions for small instances of the problem.
4. A detailed presentation of the algorithm/types of algorithms/class of algorithms that is the subject of the learning module.
5. A discussion about the implementation of the presented algorithms to solve the given contest task and more generally the associated problem.
6. A detailed presentation of the optimisation techniques that can be applied to improve the performances to solve the various instances of the problem.

A learning module contains both theoretical and practical activities. One module can be used to organise an activity spread on one or two weeks, for a programming course, for example. Fig. 2 shows a possible sequence to be followed, which is similar to the four main steps shown on Fig. 1. This sequence is supported by the description of the learning module just presented here above.

In the first step, the learners receive the problem statement and the associated contest task. They read it by themselves at home, and they try to understand precisely the problem and the given specifications. The second step takes place in the classroom with the professor. He or she starts by answering questions from learners about their understanding of the problem and then presents the basic algorithm(s) that can be used to solve it, and gives some tips about the possible optimisations that could be done. Then, the learners work by themselves, implementing their ideas and testing them on the contest task. They will go through several iterations between the last two steps, optimising their solution. At the end, a final exchange with the professor is organised, after he or she looked at the learners' solutions, to make a collective debriefing and feedback session.

Implementation examples contained in the learning module can be available in several programming languages. A note is associated to them, describing briefly the implementation and any language-specific features that have been used to solve the contest task. It is indeed also important to take into consideration the actual implementation since it may influence performances. Moreover, learning modules are not intended to focus on one single programming language. The material associated to a learning module should be language-agnostic, except for elements 5 and 6 mentioned above that may contain discussions related to the actual implementation with given programming languages.

To be able to use the proposed learning modules to teach algorithms design and optimisation, they have to be related in some way. Two elements can be used to classify and relate the learning modules:

- Keywords can be used to search for learning modules that are related to the same content. For example, all the modules concerning dynamic programming or all the modules with algorithms dealing with trees can be identified.
- References between learning modules can be used to build dependency trees grouping modules into learning blocks with a common theme and following given learning paths. For example, a sequence of modules about dynamic programming from simple to advanced techniques can be identified.

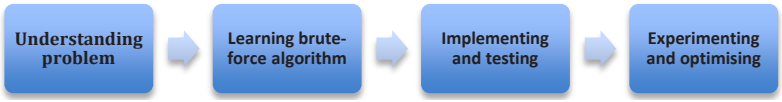


Fig. 2. A possible sequence of activities to follow in order to use proposed learning modules to teach a course.

3.4. *Evaluation*

Since this paper proposes materials to teach and learn, it should also take the evaluation part into account. In particular, learning modules should contain materials that can be used to assess whether the learners successfully achieved the four steps shown in Fig. 2. For that purpose, the learning materials contain several simple exercises that can be proposed to the learners to check whether they got a good understanding of the newly learned concepts. These exercises can go from simple multiple choices questions to open questions requiring writing small codes. For the latter, future work includes the possibility to provide automated evaluation and feedback through the use of the Pythia platform (Combéfis, *et al*, 2012; Combéfis*, et al.*, 2015). Also, several contest tasks could be used for the same learning modules since they are dealing with the same algorithmic techniques. They can be used to assess whether the learner is able to transfer its new knowledge to a different context.

To ease the use of the learning modules to support a course, their associated materials are split in two parts, that is, presented in two different ways. The complete description as presented in the previous section is dedicated to the professor. Learners get a partial description not showing all the example solutions. The proposed materials can of course be used in other contexts. It is, for example, possible for autodidacts to use the description intended to professors to learn the concepts by themselves.

4. The LADO Project

The *LADO project* (Learn and Teach Algorithm Design and Optimisation) consists in the design and construction, with a group of students, of several learning modules as proposed in this paper. It will start during the 2017–2018 academic year at ECAM with a small team composed of six students. The produced material will be in English and made open source on GitHub at the following address: https://github.com/ECAM-Brussels/lado.

This project will follow a special pedagogical device and is therefore a learning experiment for the involved students. They will indeed have to learn algorithm design and optimisation techniques to be able to create learning materials. This work will be done under the supervision of a coach-professor. Involved students will work collaboratively, which is eased thanks to the use of a Git repository.

The first steps that will be taken for the creation of a learning module, in a classroom supervised by the coach-professor, are as follows:

1. Reading and understanding the problem to solve, identifying the given inputs and the expected outputs to express the problem specifications.
2. Finding a brute-force algorithm that solves the problem, so that to have a correct reference implementation to use for comparisons.
3. Analysing the given instances to see whether some of them does exhibit any peculiarities that can be exploited so that to solve them more efficiently.
4. Optimising the problem solving by trying several algorithms or with any other optimisation techniques and by comparing with the reference implementation.

After these experiments the students should have a better understanding of the task and should have discovered new algorithms and optimisation techniques. In particular, they are also able to implement the learned algorithms and try them on the case study, namely the contest task.

The next step is related to the construction of the material for the learning module, following the template presented in the previous section. This step is done remotely and collaboratively thanks to GitHub. Involved students share the work among them, make edits on their fork of the repository and submit them with pull requests. The coach-professor then uses the review mechanism provided by GitHub to provide students with feedbacks. When everything is correct, changes are merged into the master branch by the coach-professor, making a new learning module available for the community.

Tasks that will be used come from several contests, which make their tasks openly available such as the IOI, ACM-ICPC, Google HashCode, Cyber Security Challenge Belgium, etc.

5. Conclusion

To conclude, this paper proposes a way to build learning materials to teach and learn algorithm design and optimisation. The learning modules are built with tasks coming from programming and IT related contests. They are used as a context for the learning module, as a motivation and also as a concrete case study with which the learner can make experiments.

The LADO project, that will start the next academic year, will result in the creation of the first learning modules implementing the propositions from this paper. In this pedagogical device, a group of six students, supervised by a coach-professor, will produce several modules that will be open source. This project is itself a learning experience since the involved students will learn as a side effect of the modules creation.

The next step of this work, after the creation of the first modules, will be their evaluation in real conditions, that is, using them to teach algorithm design and optimisation related stuff to interested learners.

References

Booth, S. (2001). Learning computer science and engineering in context. *Computer Science Education*, 11(3), 169–188.

Combéfis, S., le Clément de Saint-Marcq, V. (2012). Teaching programming and algorithm design with Pythia, a web-based learning platform. *Olympiads in Informatics*, 6, 31–43.

Combéfis, S., Van den Schrieck, V., Nootens, A. (2013). Growing algorithmic thinking through interactive problems to encourage learning programming. *Olympiads in Informatics*, 7, 3–13.

Combéfis, S., Wautelet, J. (2014). Programming trainings and informatics teaching through online contest. *Olympiads in Informatics*, 8, 21–34.

Combéfis, S., Paques, A. (2015). Pythia reloaded: an intelligent unit testing-based code grader for education. In: *Proceedings of the 1st Int'l Code Hunt Workshop on Educational Software Engineering (CHESE 2015)*. 5–8.

Combéfis, S., Beresnevičius, G., Dagienė, V. (2016). Learning programming through games and contests: overview, characterisation and discussion. *Olympiads in Informatics*, 10, 39–60.

Dewey, J. (1938). *Experience and Education*. New York, The Macmillan Publishing Company.

García-Mateos, G., Fernández-Alemán, J.L., (2009). Make learning fun with programming contests. *Transactions on Edutainment II*, LNCS Volume 5660. Springer-Verlag, 246–257.

García-Mateos, G., Fernández-Alemán, J.L. (2009). A course on algorithms and data structures using on-line judge. In: *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2009)*, 45–49.

Kant, E. (1985). Understanding and automating algorithm design. *IEEE Transactions on Software Engineering*, SE11(11), 1361–1374.

Leal, J.P., Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6), 567–581.

Leal, J.P., Silva, F. (2008). Using Mooshak as a competitive learning tool. In: *Proceedings of the ACM-ICPC Competitive Learning Institute Symposium (CLIS 2008)*.

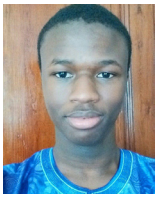
Lethbridge, T.C. (2000). What knowledge is important to a software professional? *Computer*, 33(5), 44–50.

Levitin, A. (1999). Do we teach the right algorithm design techniques? In: *Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education (SIGCSE 1999)*, 179–183.

Ribeiro, P., Guerreiro, P. (2008). Early introduction of competitive programming. *Olympiads in Informatics,* 2, 149–162.

Skiena, S. (2008). *The Algorithm Design Manual*. 2nd edition, Springer.

S. Combéfis obtained his PhD in engineering in November 2013 from the Université catholique de Louvain in Belgium. He is currently working as a lecturer at the École Centrale des Arts et Métiers (ECAM Brussels), where his courses mainly focus on computer science. He also obtained an advanced master in pedagogy in higher education in June 2014. Co-founder of the Belgian Olympiad in Informatics (be-OI) with Damien Leroy in 2010, he later introduced the Bebras contest in Belgium in 2012 and at the same time founded CSITEd. This non-profit organisation aims at promoting computer science in secondary schools.



S.A. Barry is a bachelor student in computer science engineering at the École Centrale des Arts et Métiers (ECAM, Brussels). He has been interested in electronics and programming for many years. In particular, he likes computer security and wants to improve his hacking skills to detect potential flaws in computer systems to protect them. The LADO project could help him to improve his computing skills.

M. Crappe is a bachelor student in electronics engineering at the École Centrale des Arts et Métiers (ECAM Brussels). Born with the entrepreneurial spirit, he is actively involved in innovative extra-curricular projects in electronics and software developments. He hopes that the LADO project will allow him to grasp new skills in algorithm design and optimisation in order to further push his projects.

M. David is a bachelor student in electronics engineering at the École Centrale des Arts et Métiers (ECAM Brussels). He shares a great passion for both electronics and programming, as he is involved in the Rust programming language community and enjoys embedded systems development and the internet of things. Joining the LADO project is for him an opportunity to continue to improve his programming skills while pursuing his studies in electronics.

G. de Moffarts is a bachelor student in computer science engineering at the École Centrale des Arts et Métiers (ECAM Brussels). He is interested in computer science and electronics, and very curious about engineering and new technologies, such as 3D printing, artificial intelligence and the internet of things. He joined the LADO project to get an early hands-on approach in algorithm design, and hopes to put his training to use in future personal projects.

H. Hachez is a bachelor student in computer science engineering at the École Centrale des Arts et Métiers (ECAM, Brussels). Naturally curious, he wants to increase his knowledge in computer science. In particular, he is interested in artificial intelligence and data sciences. He also likes to shine in all the projects he is involved in, and he hopes that the LADO project will help him to improve his skills to drive him among the best.

J. Kessels is a bachelor student in computer science engineering at the École Centrale des Arts et Métiers (ECAM Brussels). He is passionate about software development, web technologies and electronics. Having already created several mobile applications and websites, he seeks to improve his coding skills, optimise his applications and learn new algorithms, as a contributor to the LADO project.