A Web Platform for Problem Solving Competitions

Marco Chiarandini
Department of Mathematics and Computer Science
University of Southern Denmark
marco@imada.sdu.dk

Abstract

Computational contests on challenging problems are often organized by the research community in Computer Science. They help to gather researchers, to align, verify and standardize programs, and to objectively assess the solutions proposed. In this short note, we illustrate how to use a problem solving contest as a feedback and assessment activity in education. The context is a course for a Bachelor degree in Computer Science. The advantages envisioned are motivation, automatic assessment and immediate feedback. Some pitfalls and evaluation by students are also discussed.

Keywords: Autoassessment, e-learning tools, competition-based learning, writing assignments, student motivation

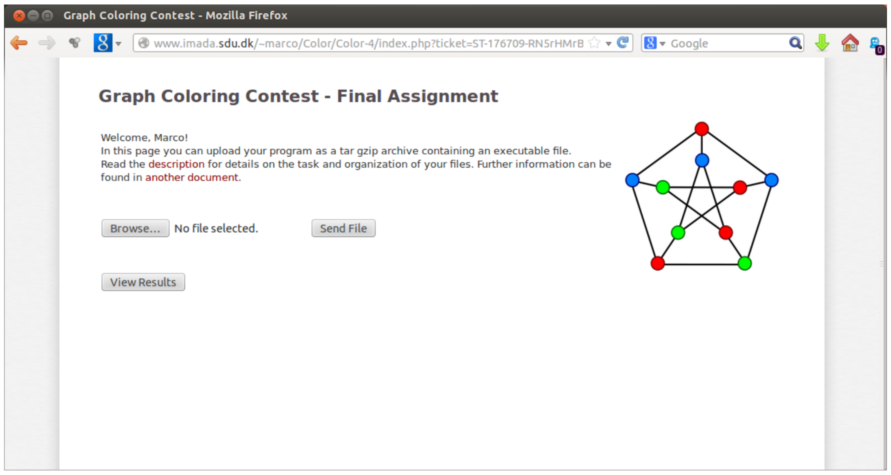
1 The Course

The course "Heuristics for Optimization" is an elective course in Computer Science offered to students in the last year of their Bachelor or in their Master education. The course typically enrolls 20-25 students. The course runs for 8 weeks including the final assessment. The working load is equivalent to 100 working hours or 5 ECTS.

The content of the course is heuristic algorithms to solve combinatorial optimization problems. Combinatorial optimization is a branch of mathematics that studies decision problems that can be modeled mathematically by means of variables, constraints on the variables and objective functions. Variables can take only discrete values and the task is to find the assignment that maximizes an objective function. Examples are finding the shortest tour around a number of points to visit or the scheduling of classes in such a way that no two classes that share attendants are placed at the same time. These problems are often reformulated using simplified mathematical structures, such as graphs. For example, the scheduling of classes can be formulated as a graph coloring problem. These problems are typically very hard to solve exactly, hence in computer science, one uses heuristic algorithms that make reasonable and efficient decisions even though they might be non-optimal. The practical deployment of heuristic algorithms is only in part guided by theoretical thinking while a good part is due to experience and revision. Often the design of these algorithms is regarded as engineering, or even art, rather than science [HS04]. The algorithms are iteratively refined in a so-called engineering cycle where new ideas are iteratively introduced on the basis of analytical considerations and hypothesis testing on empirical data. The experimental assessment requires that the algorithms are implemented in a programming language and statistical methods are used in the evaluation process.

2 Learning Goals

At the end of a course on these topics we would like that students are able to design efficient heuristic algorithms for a variety of combinatorial optimization problems. As explained, mas-


Graph Coloring Contest - Mozilla Firefox
www.imada.sdu.dk/~marco/Color/Color-4/index.php?ticket=ST-176709-RN5rHMrB
Google
Browse...
Send File
View Results

Graph Coloring Contest - Final Assignment

Welcome, Marco!
In this page you can upload your program as a tar gzip archive containing an executable file.
Read the description for details on the task and organization of your files. Further information can be found in another document.

No file selected.

Figure 1: The submission page of the Problem Solving Contest

tering this process entails becoming acquainted with all the aspects of the engineering cycle and with the guidelines and tools for its stages. Hence, a secondary learning goal is learning to carry out a sound experimental evaluation using proper methods from statistics. Finally, since the way these algorithms are communicated is by description rather than by source code, we wish the students to learn to describe their work in a precise language that includes mathematical formalism and algorithmic sketches.

3 Feedback and Assessment Activities

In the last two editions of the course we have used the following feedback and assessment activities. During the course the students have to hand in three assignments. The assignments can be seen as a single assignment divided into parts consisting in designing and implementing a solution algorithm for a combinatorial optimization problem that reminds those seen in class, e.g. the graph coloring problem. A pass in all three assignments is a necessary condition to be admitted to the final fourth assignment that alone determines the final grade.

Obligatory assignments pass/fail

The three obligatory assignments ask to:

- design a solution algorithm on the basis of the theory delivered in class
- implement the algorithm in a programming language (Java, C++)
- describe the algorithm in a two page text document.

Students submit their work through a *web platform* built specifically for the course. The submission consists of a program and a two page document describing the algorithm that the program implements. The simple interface of the platform is shown in Figure 1.[1]

In this phase, students are allowed to work in pairs and to discuss their solutions. Pair programming is believed to be helpful in programming tasks [LBC10], moreover discussions and exchange of ideas may be inspiring. The goal in a collaborative approach is also to favour situated and social learning [LW91]. However, submissions must be individual and students are recommended to work independently on the descriptions.

[1]Marco Chiarandini. Problem Solving Contest. `http://www.imada.sdu.dk/~marco/Teaching/AY2013-2014/DM811/psc/`. Created October 2013, last visited: December 2013

The program The assignment is organized in form of a competition. The program submitted must comply with a few requirements on the output of the solution to the problem and on the halting time. Immediately after the submission the student receives information whether the program satisfies these requirements. If it is so, the program is run on a set of numerical instantiations of the generic problem. The results are collected and analyzed. Finally, the outcome of the analysis is shown on a web page. Here, the student can compare how good he/she did in comparison to peers. As an example, the final analysis for the edition 2013 on the graph coloring problem is shown in Figure 2. The plots visualize the experimental results collected on three sets of problem instances indicated in the strips by the text 0.1, 0.5, 0.9. Results are transformed in ranks according to the quality on each specific problem instance. The first three panels are boxplots and are used to represent graphically the raw data, that is, the empirical distributions of results of each student, whose username is reported on the vertical axis. The second three panels are more involved representations to assess the statistical significance of the observed differences. A Friedman one-way analysis of variance is performed to produce confidence intervals around the median ranks. The students are instructed on the statistical theory behind these plots and on how to reproduce them independently. In short, distributions of results shifted to the left represent better solutions to the problem. Thus, the plots are used to provide to the students an immediate feedback on the validity of the algorithmic ideas tried to solve the problem.

There are a few variations on this procedure. On the one side, the methods for the analysis change throughout the three parts that compose the assignment, as different relevant scenarios are presented. This is used to teach the students the different analysis situations that may arise. On the other side, resubmission can be controlled. Allowing resubmission focuses the development on the algorithms since the student has the possibility to test immediately every new idea. Allowing to submit only once focuses on the assessment methods since, in the ideal case, students have to compare offline their ideas using, for example, the methods for the empirical analysis shown in Figure 2 and explained earlier in the course.

A further element of feedback are the discussions in class stimulated by the results on the contest. Here students learn which idea works, which do not, and why. These activities are designed to let the student learn about the algorithms and the engineering cycle.

The written description The two-page written descriptions in the obligatory assignments are reviewed by peers and by the teacher. Each document receives a student as peer reviewer. A list of common error to which reviewers can refer to is maintained and published at the course web page. This activity aims at letting reviewers experience the reading process and become aware of the elements that are present in a good report and those that determine bad quality. Learning to assess elicits forms of elaboration, meta-cognition and self-insight that are at the higher-levels of the Structure of Observed Learning Outcome (SOLO) taxonomy [BT11].

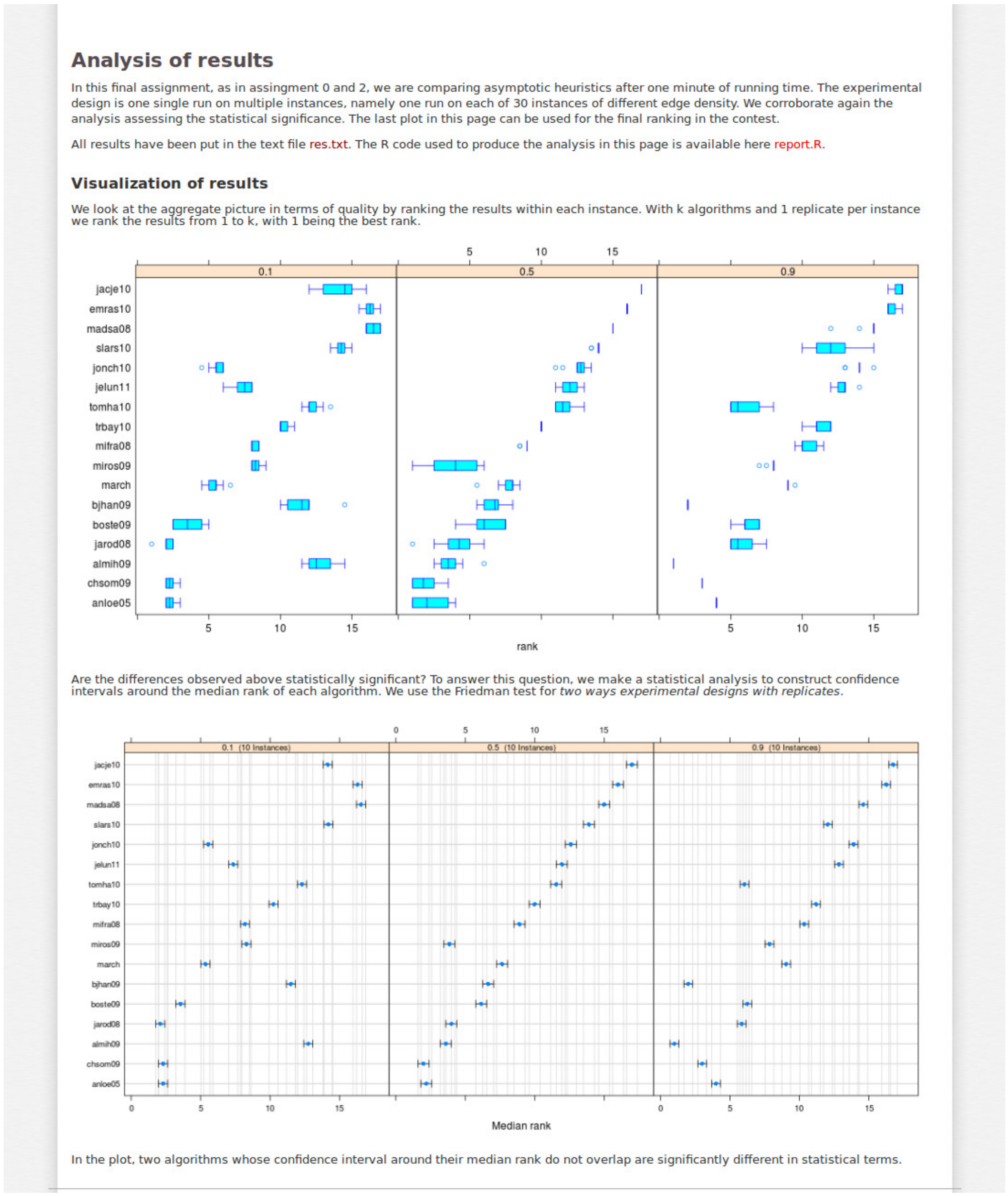
The teacher's review is instead used for feedback to the student who wrote the description.

The final assignment

The final assignment is carried out individually and graded, with external censorship. It consists of a revision and collection of the tasks from the previous assignments. It is expected that the feedback is used to improve the work.

As in the other assignments students have to submit a program and a description. This time the description is longer, up to 12 pages, as it has to encompass all tasks from the three obligatory assignments. The programs are again compared in the web page and resubmissions are allowed. Past examples of reports with best, middle and worst grades are published on the course web page.

Students are informed that performance in the competition counts: a perfect report with a poorly performing program will not get the highest grade. The viceversa holds as well thus the role of the contest in the evaluation is moderated. Other factors are communicated that can



Analysis of results

In this final assignment, as in assingment 0 and 2, we are comparing asymptotic heuristics after one minute of running time. The experimental design is one single run on multiple instances, namely one run on each of 30 instances of different edge density. We corroborate again the analysis assessing the statistical significance. The last plot in this page can be used for the final ranking in the contest.

All results have been put in the text file res.txt. The R code used to produce the analysis in this page is available here report.R.

Visualization of results

We look at the aggregate picture in terms of quality by ranking the results within each instance. With k algorithms and 1 replicate per instance we rank the results from 1 to k, with 1 being the best rank.

Are the differences observed above statistically significant? To answer this question, we make a statistical analysis to construct confidence intervals around the median rank of each algorithm. We use the Friedman test for *two ways experimental designs with replicates*.

In the plot, two algorithms whose confidence interval around their median rank do not overlap are significantly different in statistical terms.

Figure 2: Best algorithms have distributions shifted on the left

influence the grade: the level of detail of the study; the complexity and originality of the approaches chosen; the organization of experiments that guarantees reproducibility of conclusions; the clarity of the report; and the effective use of graphics in the presentation of experimental results. Finally, it is warned that a few, well thought algorithms are better that than many naive ones.

4 Reflections

The students' evaluations of the course in the last two editions in which it has been running with the illustrated modalities can be summarized as follows. The assignments with the peer comparison of results are perceived as fun. The controlled programming context is deemed to have given the possibility to acquire competencies that it would have not been possible to gain by traditional teaching. The peer reviewing activity was perceived positively in the phase of giving feedback, while indifferently in the phase of receiving it from other students. Only one third of the students perceived to have received useful and sufficient feedback in the activity. Finally, students felt a lack of time and that it was hard to focus on handing in a competitive implementation of the algorithms and at the same time a well written report.

In our perception, the presence of the contest brought more enthusiasm in the participation of both students and teacher. It provided more objective assessment and it even made possible assessment at all in a context in which it would otherwise be extremely cumbersome. Indeed, manually checking the source code of 20 and more students would require considerable time for instructors and teachers and would be prone to human error or subjective evaluation. The feedback on the report yielded noticeable improvements on the quality of the final reports if compared to previous years when no feedback and no description was asked in the preliminary assignments.

A few details can be improved. The peer reviewing activity can be fine tuned to provide trustful and sufficient feedback to everybody. One adjustment that we are considering is the postposition of the peer reviewing process to a later obligatory assignment, thus letting students focus first on the corrections from the teacher only. A more radical shift that we are considering is the change of some of the lecture hours to class discussion and practice hours with the teacher. The content of the lectures could be left to the students for self study or recorded in a video lecture.

The course is perceived as too heavy by the students in terms of working load. Furthermore, the contemporaneous submission deadline for both program and report may make it difficult to focus effectively on both of them. Finally, the comparison of results in the contest exposes to the risk of making relative evaluation within the single year, in contrast to the good practice of aiming at absolute evaluations. To address these issues we plan in the next editions to separate the submission of the final program and report, using two distinct deadlines. Moreover, in order to favor an absolute assessment we plan to introduce in the contest the program of some students from the previous years in form of an "hall of fame" of results. If the course was to be extended in terms of working load, we would include different problems in the contest to boost even more the ability to apply the techniques from the course in a new context.

Acknowledgments I am grateful to Stefano Gualandi, researcher at the University of Pavia, for discussions, feedback and support on the design and development of the activities presented.

References

[BT11] John Biggs and Catherine Tang. *Teaching for Quality Learning at University.* Open University Press, McGraw-Hill, fourth edition, 2011.

[HS04] Holger Hoos and Thomas Stützle. *Stochastic Local Search: Foundations and Applications.* Morgan Kaufmann Publishers, San Francisco, CA, USA, 2004.

[LBC10] Kim Man Lui, Kyle Atikus Barnes, and Keith C.C. Chan. Pair programming: Issues and challenges. In Torgeir Dingsøyr, Tore Dybå, and Nils Brede Moe, editors, *Agile Software Development*, pages 143–163. Springer Berlin Heidelberg, 2010.

[LW91] Jean Lave and Etienne Wenger. *Situated Learning. Legitimate peripheral participation.* University of Cambridge Press, 1991.