Olympiads in Informatics, 2016, Vol. 10, 73–85

DOI: 10.15388/ioi.2016.05

Watch them Fight! Creativity Task Tournaments of the Swiss Olympiad in Informatics

Samuel GRÜTTER, Daniel GRAF, Benjamin SCHMID
Swiss Olympiad in Informatics
e-mail: {samuel,daniel,benjamin}@soi.ch

Abstract. As part of the qualification process for the Swiss Olympiad in Informatics, the contestants are each year confronted with one "Creativity Task". Unlike typical problems in programming competitions, creativity tasks usually do not have an optimal solution, and are often adaptations of popular board or computer games. After receiving all submissions, a tournament is organized, where the students can watch how their programs play interactively against each other, and points are awarded to the authors according to the tournament ranking.

We present and discuss this task format in general, as well as the specific creativity tasks of the past 10 years, accompanied by an online archive of the task descriptions, sample solutions and game servers.

Moreover, we describe our task selection process and criteria, the task creation process, and the experience gained and best practices established in the past years.

Finally, we present the many advantages of this task format, showing why we think it is a refreshing alternative to the common IOI-style tasks given in most national selection rounds.

Keywords: creativity tasks, interactive tasks, heuristics, programming competition, board games, artificial intelligence contest, tournaments, task visualization.

1. Introduction

During the first round of the Swiss Olympiad in Informatics (SOI), the students have two months to solve a set of tasks at home and submit them on the SOI website. There are practical and theoretical tasks, and one creativity task. This paper is about the creativity tasks.

Contrary to typical tasks in programming competitions, the creativity tasks are usually designed in such a way that there is no optimal solution, or that the optimal solutions are not efficient enough. The creativity tasks are often adaptations of popular board or computer games, and the game can be played with two or more players. The participants' task is to write a program able to play the game. We will refer to such programs as *bots*.

An interesting aspect of this task format is that the participants' bots can play and compete against each other.

At the start of each year's first round, the following material for the creativity task is published on the SOI website:

- The **task description**, describing the rules of the game and its parameters. Depending on the game, these could include, for instance, width and height of the playing field, or a map of the playing field, the number of players, the initial amount of money/food of each player, etc. The task description also specifies the *communication protocol* between the bots and the game server (see below), which defines how the bots have to communicate their actions and how they are informed about the other bots' actions. All communication is done via standard input/output.
- A **game server**, a program written by the task authors, which launches the bots, communicates with them according to the protocol and keeps track of the game state. It informs the bots about all the relevant changes and prints a log of each action and state change. The server also acts as a judge that can terminate bots that drop out of the game or take too long to answer. If the interaction protocol is easy to read, the game server can also allow for human players that type in their commands interactively. This way, the participants can play against their own bots by hand.
- Some **sample bots** in various programming languages. These bots follow all the rules of the protocol, but they just pick a random move in every step. The participants can base their solutions on these sample bots to quickly learn how to implement the protocol and input/output. Moreover, they can evaluate their bots by letting them play against the sample bots.
- A **visualization**, which reads the log produced by the game server, and displays a graphical animation of the game. For some of our tasks, the game server already provides a rudimentary text-based visualization of the game.

After the end of the first round of the SOI, the organizers let the submitted bots compete against each other in a large number of games covering many different configurations, to make sure that the obtained results are representative and random decisions average out well enough. The results of all these games are then aggregated into a final score for each participant.

Between the first and the second round of the SOI, all participants are invited to an event called "SOI-Day" consisting of task discussions, talks and presentations. At this event, a shortened version of the the creativity task tournament is presented as if it was a live tournament, and the participants can thrill while watching their bots compete against the others.

2. Ten Years of Creativity Tasks

In this section, we give a brief presentation of each creativity task of the past ten years. We omit details such as the communication protocols, and refer the reader to the online task archive (`http://creativity.soi.ch`) for the full task descriptions.

2.1. *Connect Five (2007)*

Task description On a 20×20 grid, two players (black and white) alternate in placing a stone of their color on an empty square. As soon as a player succeeds in placing 5 stones in a row (horizontally, vertically or diagonally), he wins.

Note that unlike the game sold under the name "Connect Four", the stones do not "fall down" in their column, but stay exactly where they were placed.

Discussion This variant of the game is also known under the name "Gomoku", and appeared at the International Computer Games Association's *Computer Olympiad* in 1989, 1990, 1991 and 1992.[1]

2.2. *Fight of the Ant Populations (2008)*

Task description On a grid with obstacles, multiple ant colonies are fighting against each other. Each player controls all ants of a colony. In every turn, the player sees the 7×7 neighborhood of each of his ants and can move them individually to an adjacent tile. Some tiles contain food, others contain ants or the hill of the enemy. Collecting food and carrying it back to the own hill allows the colony to grow. If an ant attacks another ant both die. If an ant reaches an adversarial hill, three random ants of that colony die. Which colony survives the longest?

Discussion This game allows for a huge variety in strategies. Some submissions assigned different jobs to the ants, for instance: explorers that go to unknown territory, workers that collect the closest food, guards that stay near the hill and warriors that try to reach the hill of the enemy.

2.3. *Grand Theft Cake (Portal Maze) (2009)*

Task description The players are searching for a big cake in an unknown, polygonal maze with walls all around them. They have a handheld portal device that allows them to teleport from A to B instantly. They can walk around with a speed of 1 meter per second. In 0.1 seconds, they can look in a direction to learn how far away the wall is. It takes 5 seconds to shoot a portal in a direction (either of type A or B). The cake is a circle of one meter diameter. The goal is to find the cake and return to the starting position as quickly as possible.

Discussion The submitted solutions made creative use of the portal. Some just used it to return to the start quickly, others repeatedly used it as a shortcut throughout the entire search for the cake.

[1] `http://www.game-ai-forum.org/icga-tournaments/game.php?id=30`

2.4. *Tanks (2010)*

Task description Each player controls a tank, which can move on a line and fire one missile per round. The tanks can use different weapons, which are specified by their range, their impact radius, and their damage points. The tanks have an unlimited number of missiles of each weapon. When a missile hits a position, the damage points of all tanks within the impact radius increases by the number of damage points of the weapon, and tanks reaching a predefined number of damage points die. Moving around costs fuel, and the fuel is limited. The last surviving tank wins.

Discussion Most submissions implemented some simple ad hoc heuristics. The winning solution was 285 lines long and in each round, it chose the weapon and target position which would cause the highest total damage to all opponents, supposing that they would not move. If there was a tie between several possibilities, the one with the biggest impact radius was chosen.

2.5. *Multisnake (2011)*

Task description *Multisnake* is a multi-player version of the popular computer game called *snake* (Fig. 1). It is played on a rectangular grid which contains some obstacles. Each player controls a snake, and in each turn, all snakes move simultaneously by one step. When a snake moves onto a field occupied by an obstacle or by a snake, it dies. The winner is the last surviving snake. Some tiles contain a black or white ball. When a snake eats a white ball, it grows by one, and when it eats a black ball, it shrinks by one. To enforce termination of the game, all snakes grow by one each T turns, where T is a small positive integer.

Discussion The winning solution was 1005 lines long, implemented a complex scoring function for possible states, and explored them recursively.

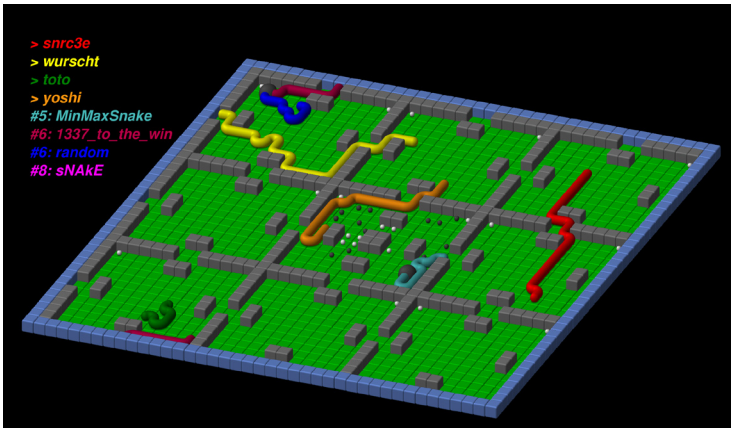


Fig. 1. Multisnake.

2.6. *Find the Anthill (2012)*

Task description In this second ant-themed task, the story was more peaceful: each ant just has to find the way back to its own ant hill (Fig. 2). The big difference is that this time each ant is controlled by a separate instance of the participant's submission. So the program controlling a single ant does not know where the other ants of its colony are, unless they are in its 7×7 field of view.
Like in nature, there is a distributed way of communication though: scents. The ants can place one out of 256 different scents on their current tile and they can sense the scents that other ants (of their own or of other colonies) put there. So if one ant sees the hill, it can leave hints for its colleagues.

Discussion Successful submissions made creative use of the scent hints to share knowledge between the ants. Some even tried to learn and imitate the marking patterns of the opponents to mess with it and cause confusion.

2.7. *Who wants to be a billionaire (2013)*

Task description The task is to write a bot which can interactively answer multiple choice questions with four possible answers, using Wikipedia articles as a knowledge base. For each question, the bot is allowed to consult up to 25 articles on the English Wikipedia. To do so, the bot has to provide a query string to the game server, and the game server will look up the article on Wikipedia (or in its cache), and feed a plain text version to the bot.

Discussion Most solutions were based on keyword search combined with some strategy to avoid frequent words which do not carry any meaning. The submitted bots would not have become billionaires, but the best was still twice as good as a random bot (which would have scored $\frac{1}{4}$ of the points).

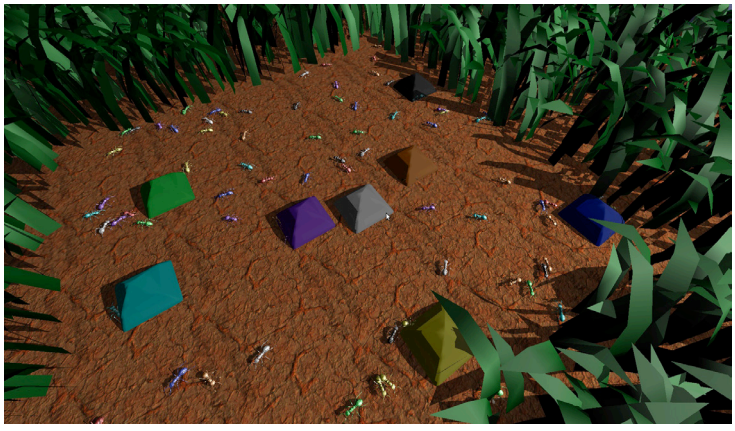


Fig. 2. Find the Anthill.

There were 11 submissions, and for the tournament, they were fed a total of 547 questions. Table 1 shows the ranking and their percentage of correct answers. Note that the primary ranking criterion was whether they were compilable and never crashed, and the number of correct answers was only the secondary criterion.

2.8. *Cops and Robbers (Scotland Yard) (2014)*

Task description Some cops are hunting a robber in a city whose streets and junctions are given as an undirected connected graph (Fig. 3). Initially, the cops and robbers are positioned on distinct junctions (nodes in the graph), and in each round, the robber and all cops move along an edge of the graph. As soon as a cop moves onto the same node as the robber, the cops win, and if this never happens during a predefined number of rounds, the robber wins.

One bot controls the robber, and another bot controls all cops. Every bot must be able to play both roles. At any time, the bots know the entire graph and the positions of all agents.

Discussion This is a classic problem in graph theory and we refer to Aigner and Fromme for a nice introduction (Aigner and Fromme, 1984).

The participants mostly implemented sophisticated scoring functions that would for instance weigh the distance between the cops and the robbers against the number of possible escape routes.

Table 1

The ranking and their percentage of correct answers

rank	1	2	3	4	5	6	7	8	9	10	10
% correct	50%	37%	35%	45%	36%	28%	27%	2%	0%	N/A	N/A
behavior	never crashed			sometimes crashed						did not compile	

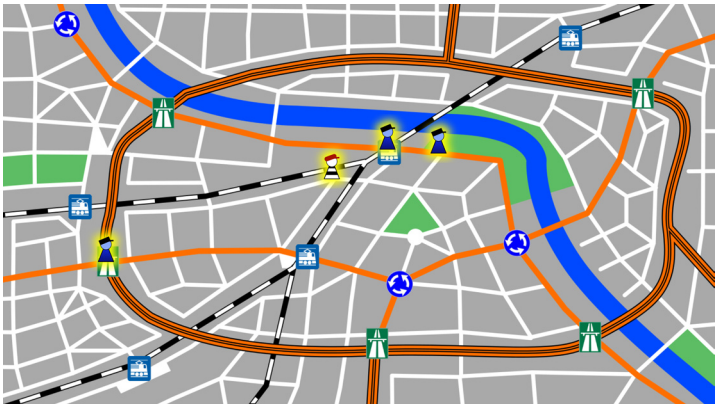


Fig. 3. Cops and Robbers.

2.9. *Diamond Auction (2015)*

Task description The players are at an auction bidding for diamonds. Each player starts with the same amount of gold and for each diamond everyone can bid an amount of his choosing. The highest bid gets the diamond, but all offered gold has to be paid. The goal is to maximize the number of bought diamonds using only the given amount of gold. To allow for more interesting strategies, there are several games played with the same players without restarting the participant's programs. This way, they can learn and exploit each other's strategies.

Discussion Being a very simple game with a simple protocol, we had many submissions for this task. Many of them implemented a random strategy sometimes even worse than our sample bots. Most of the more successful solutions tried to imitate the opponent or to predict the next move. The following graph (Fig. 4) shows the last game of the final, the lines showing the bids of the two players for the different rounds. Although the bids seem random the players are able to predict each others bid and bid just a little bit more. This results in a very clever use of the available gold.

2.10. *Vector Car Racing (2016)*

Task description Each player controls a car and in each round he can change his velocity vector by at most one in both directions. The game is played on a grid (i.e. a graph paper) with a start, a goal, some checkpoints and some walls enclosing a racetrack. The goal is to visit each checkpoint at least once and be the first to reach the goal. Should a player drive into the wall (e.g. if he didn't slow down early enough in a turn) he will be reset to the last visited checkpoint. To allow for some interaction, we added two items that can manipulate the velocity vector of other players.

Discussion This game is a well known pen and paper game and goes back to Jürg Nievergelt. It became widely known by an article in Martin Gardner's column in the Scientific American (Gardner, 1973).

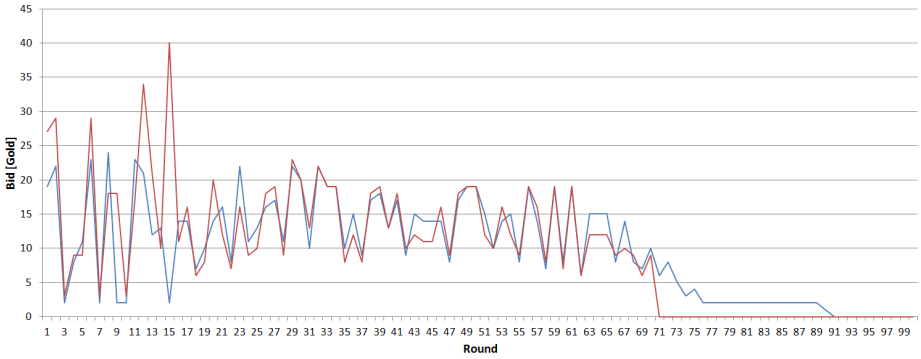


Fig. 4. Diamond Auction (the graph shows the last game of the final).

The submissions fell into one of two categories: sophisticated algorithms beating human players easily (3 submissions) or being barely able to play the game (4 submissions). One example from the second category used DFS to determine the "shortest" path resulting in some of the longest possible paths.

The winning submission consists of 1741 lines of code using a wide range of algorithms. On small maps it uses a four dimensional BFS (location and velocity) to find the optimal solution. On bigger maps, a heuristic is used to improve the runtime of this algorithm and only paths between two checkpoints are calculated. To find the best order of checkpoints it uses some more heuristics to find a good approximations for the traveling salesman problem.

2.11. *Statistics*

Table 2 shows our observed participation rates in the last ten years. Given the *bonus*-like character of the creativity tasks, participation rates in the past years fluctuated heavily. Usually, only students with enough time (and interest) left after solving the five *classical* problems tackled the creativity task. Tasks with a very simple interface (like the diamond auction in 2015) seem to have drawn additional interest, probably because writing a first solution was easily possible within an hour, also for beginners.

3. Implementation Details

3.1. *Our Task Selection and Creation Process*

The SOI is organized by a team of approximately twenty people, most of whom are former participants and now university students. Each year when the preparations of the first round of the SOI start, they discuss a few proposals for the creativity task on their internal mailing list. Once the task is chosen, two to three organizers are selected to write the detailed task description and implement the game server and the visualization. A few more organizers proofread the task description and write sample bots in as many languages as possible.

Table 2
Participation rates in the last ten years

year	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016
task	c5	ants	maze	tanks	snake	ants	quiz	c&r	◇	race
creativity	17	N/A	4	6	2	10	11	9	25	7
total	**21**	**31**	**35**	**40**	**48**	**29**	**29**	**28**	**52**	**72**

The organizers of the creativity task are usually people who have successfully taken part in the creativity task a few years before when they were participants, and they are supported by older organizers who prepared the creativity task a few times before. This ensures a good knowledge transfer, which leads to good creativity tasks, even though this task format is more challenging for task creators than ordinary tasks and even though every year, many organizers join and leave the team.

3.2. *Evaluating the Submissions*

After the end of the first round of the SOI, the organizers review and compile all submissions for the creativity task, and then run a large number of games, covering a representative selection of different game parameter and player combinations. The results of all these games are then aggregated into a final score for each participant, and points for the creativity task are awarded as follows:

- A bot that follows the protocol and is capable of playing the game without violating any rules is awarded a predefined percentage p of the points, usually between 30% and 50%, depending on how difficult the task is. We also look at the source code to see if it witnesses more effort than just copying one of the provided sample bots.
- The rest of the points are awarded according to the aggregated score described above, in such a way that the participant ranked last gets p points, and the participant ranked first gets full score.
- If the bot contains minor bugs, such that it can still play in most games, but sometimes crashes, runs into an infinite loop, or violates the rules, some points are deducted.

This grading scheme ensures that every effort to write a program for this task is rewarded, even if it ranks last, which hopefully encourages beginners to tackle this task.

3.3. *Technical Considerations*

Ease of running the game server and the visualization It is crucial that all participants are able to run the game server on their own computers.[2] However, the game server needs to be able to invoke other programs (the bots) and read from their standard output and write to their standard input, a feature which is hard to get operating system independent. And even if the game server works on all major operating systems, the instructions on how to install the dependencies, compile and run it might look so long and complicated that beginners are quickly scared off. For the visualization, it can even get worse, if specific graphics libraries have to be installed and linked.

[2] We cannot provide an online game server instead, because in the first round of the SOI, no participants should be disadvantaged because they cannot use their favorite programming language. This would mean that we would have to support a very large number of programming languages, and moreover, it would require a big effort for the sandboxing.

In our experience, implementing the game server in Java, which is available on all platforms, seems to be the best solution. But even getting a Java game server to run was a bit challenging for some participants, because some could only install Java 6 (instead of newer versions) on their system, while others did not know how to specify the paths to the bots as command line arguments, etc. But we could always help them by email support or through our forum. What we do not know, however, is how many people did not succeed in getting the server to run, but did not ask for help.

For the visualization, the two best solutions seem to be either to integrate the visualization in the Java game server, or to implement it as an HTML5/JavaScript page, where one can paste the log of the game server, and then watch the visualization. An advantage of the latter is that it only requires a web browser, which is available on every computer, so we can provide some sample game logs, so that the participants can watch some games before even starting to implement their own solution and to bother getting the game server to run.

Fraud Detection The grading of the creativity task is not fully automated on purpose. The submissions are compiled manually to check for compatibility issues and the source code is read to get an idea of the participant's solution and to check that all the rules are followed. Given the small number of participants, this is feasible and also eliminates the need for a separate fraud detection or sandboxing of the submitted programs.

4. Discussion

4.1. *Task Selection Criteria*

We now present our criteria for the selection of the creativity task. This is more a collection of useful arguments that came up in past discussions rather than an ultimate list.

- **Interactivity:** It should be interactive, and interaction should not only happen between the game server and the bots, but also between the bots, i.e. the possible moves of a player should depend on the actions that the other players took before. This is mostly to make the task more fun and different from the standard tasks, but it is not strictly necessary. For instance, the *Grand Theft Cake* task (section 2.3), or the *Who wants to be a billionare?* task (section 2.7) do not have any interaction between the players, and still were successful creativity tasks. In the case of the *Vector Car Racing* task (section 2.10), however, some special effects were added to the traditional game to make it interactive, and to increase the size of the search space that optimal solutions would have to explore.
- **Hardness:** There should be no optimal solution, or optimal solutions should be too expensive to calculate.
- **Flat learning curve:** The task should be simple, with few, easy to understand rules, so that it is suitable for beginners, but it should also offer many interesting options for advanced participants.

- **Novelty:** The task should not be "too standard" to make sure one cannot just copy and adapt a standard solution found on the internet. For instance, chess or Nine Men's Morris were proposed in the past years, but not chosen for this reason.
- **Continuous Complexity Curve:** There should be a continuous spectrum of imagineable solutions between straightforward random bots and very sophisticated solutions.
- **Plethora of solutions:** There should be many different solutions that could work reasonably well and are feasible to implement. For instance, for some proposed tasks, it was feared that applying a standard minimax algorithm would be almost the only reasonable solution, and these tasks were thus not chosen.
- **Manual playing:** It is a plus if the game can be played by hand, such as *Multisnake* (section 2.5) or *Vector Car Racing* (section 2.10), so that the participants can compare their bots to their own intuitive playing style.
- **Visualizability:** A good task should allow for an appealing way of watching the interaction between the bots and giving a dramatic view on the progress of the game.

4.2. *Similar Task and Contest Types*

The IOI and other programming contest are always experimenting with new, less algorithmic task types. We refer to Verhoeff (Verhoeff, 2009) for an overview of *classical*, algorithmic IOI tasks. Forišek (Forišek, 2013) gives an overview of the programming contest landscape. He also presents some tasks from the IPSC and Slovak national competitions whose goal is different from just finding the fastest algorithm. Some contests entirely focus on hard optimization problems like Topcoder's Marathon matches and Google's Hash Code and other contests feature tournaments of competing bots for interactive problems like the AI Challenge or the Computer Olympiad.

Creativity-like tasks are also mentioned as *game-playing events* by Burton in (Burton, 2008), where many interesting suggestions for out-of-the-ordinary activities in Olympiad training camps are proposed.

4.3. *Advantages of this Task Format*

In our opinion, creativity tasks offer a number of unique possibilities. These nonstandard tasks can keep the participants busy for many weeks. They can try out all kinds of crazy algorithms and even the best ones are never really done. But also students with limited time can participate with some quick extension of the provided random bot.

The visualization of these tasks are ideal to be shown on a large screen during an award ceremony as parents, friends and teachers can really see what the contestants were doing. As related board and computer games are well-known, an audience that is not familiar with specifics of the algorithmic challenge can also appreciate the results.

The tournaments are fun to watch as the interaction between the bots suggests that the contestants are immediately fighting against each other.

Finally, we believe that creativity tasks are also a way to train for the newer, less-standard IOI tasks where heuristical solutions are required. Examples of such creative IOI tasks include *Languages* and *Maze* from IOI 2010, *Odometer* from IOI 2012 and *Art Class* from IOI 2013.

4.4. *Disadvantages of this Task Format*

It often proved challenging to find tasks that beginners can easily get started with but still leave a lot of options for the creativity of more advanced students. This higher initial hurdle lead to fairly small numbers of serious submissions for many of the presented tasks. From an organizer's point of view, creativity tasks take significantly more time to prepare and grade than regular tasks.

5. Conclusion

In summary, creativity tasks proved to be a beneficial addition to the types of algorithmic challenges given in the first national round of the Swiss Olympiad in Informatics.

We provide the full description and supplementary material for the presented tasks online at `http://creativity.soi.ch` and we want to encourage other delegations to experiment with such task types and look forward to learning about their experiences.

Acknowledgments

We would like to thank all the volunteers of the Swiss Olympiad in Informatics who created, prepared and graded these tasks over the years.

References

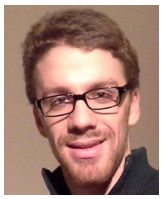
Aigner, M., Fromme, M. (1984). A game of cops and robbers. *Discrete Applied Mathematics*, 8(1), 1–12.

Burton, B. (2008). Breaking the routine: events to complement Informatics Olympiad training. *Olympiads in Informatics*, 2, 5–15.

Forišek, M. (2013). Pushing the boundary of programming contests. *Olympiads in Informatics*, 7, 23–35.

Gardner, M. (1973) Sim, Chomp and Race Track: new game for the intelligent (and not just for Lady Luck). *Scientific American*, 228(1), 108–115.

Verhoeff, T. (2009). 20 years of IOI competition tasks. *Olympiads in Informatics*, 3, 149–166.



S. Grütter was a participant at CEOI 2009 and IOI 2010, and has helped with the organization of the Swiss Olympiad in Informatics since 2011. He was deputy leader of the Swiss team at IOI 2013.

He holds a BSc in Computer Science from Ecole Polytechnique Fédérale de Lausanne (EPFL), where he is currently pursuing his MSc and working at the Scala Lab under the supervision of Prof Martin Odersky.

His research interests are logic, formal languages, type systems and compilers.

D. Graf participated at CEOI 2009 and IOI 2009 (bronze) and returned to IOI as the delegation leader of the Swiss team in 2012, 2014 and 2015. He volunteers for the Swiss Olympiad in Informatics since 2010 and is its president since 2011.

He holds a MSc in Computer Science from the Swiss Federal Institute of Technology in Zurich (ETHZ), where he is currently pursuing his PhD in the group for Algorithms, Data Structures, and Applications of Prof. Peter Widmayer.

B. Schmid participated at IOI 2013, IOI 2014 (bronze) and CEOI 2013 (bronze). Since 2015 he helps organizing the Swiss Olympiad in Informatics.

He is currently pursuing a BSc in Computer Science from ETH Zurich.