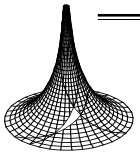

**7/1** (2009), 35–50
tmcs@math.klte.hu
http://tmcs.math.klte.hu

**Teaching**
**Mathematics and**
**Computer Science**

# Teaching graph algorithms with Visage

Andreas Fest and Ulrich Kortenkamp

*Abstract.* Combinatorial optimization is a substantial pool for teaching authentic mathematics. Studying topics in combinatorial optimization practice different mathematical skills, and because of this have been integrated into the new Berlin curriculum for secondary schools. In addition, teachers are encouraged to use adequate teaching software.

The presented software package "Visage" is a visualization tool for graph algorithms. Using the intuitive user interface of an interactive geometry system (Cinderella), graphs and networks can be drawn very easily and different textbook algorithms can be visualized on the graphs. An authoring tool for interactive worksheets and the usage of the build-in programming interface offer new ways for teaching graphs and algorithms in a classroom.

*Key words and phrases:* graph algorithms, discrete mathematics, mathematical programming, minimum spanning tree, educational software, geometry software, combinatorial optimization.

*ZDM Subject Classification:* D30, D40, D80, K30, N60, N80, U70.

## 1. Introduction

Discrete Mathematics is – finally – coming to schools. This modern and interesting part of mathematics is integrated into more and more curricula worldwide ([11], [23], [25], [24]). This article focuses on the new modules that were included in the curriculum of the state of Berlin, Germany [2]. These are based on topics from combinatorial optimization, and the curriculum states explicitely

Supported by the DFG Research Center Matheon.

that *computer software* should be used for teaching these – as it is customary at universities or in research.

In this introduction we briefly explain why the authors support the introduction of Discrete Mathematics in teaching mathematics at secondary (or even primary) level. A more detailed discussion can be found in [9].

The new standards of education emphasize the advancement of general mathematical competencies: problem solving, modeling, the ability to communicate and discuss mathematics, use of mathematical representations and handling of symbolic and formal elements of mathematics. Freudenthal [8] remarks that it is accepted by many that students have to be able to mathematize non-mathematical content.

Because modern curricula are usually based on the competencies that students should acquire, and not on the specific content to teach, we are free to choose new topics for teaching in school.

There have been several (failed) attempts to include discrete mathematics, and in particular graph theory, in school ([22], [27], [3]). The new approach is much more problem-oriented and application-based. The topic is linked to different knowledge domains from mathematics and other fields, either by the applications or through the required problem solving and modeling competencies (compare [10], [12], [1], [19], [20], [26], [4]).

For our work, we chose applications (mostly from graph theory, where students are able to create a wealth of new examples by themselves) that offer a particularly quick road to the mathematical problem without requiring much prior knowledge [21], [14]. Therefor, the mathematical theory is developed by the students themselves from questions they ask naturally, as required by the national standards [18].

## 2. The educational software Visage

Visage is an educational software package for studying graphs and algorithms. It has been developed at the DFG research center MATHEON as an extension of the interactive geometry software Cinderella [16].

Using software for learning about graphs and algorithms can be done in very different ways. To support as many ways as possible, we offer three different usage scenarios: Visage as an *interactive graph laboratory*, Visage as an *authoring tool* for interactive worksheets or presentations, and Visage as a *programming interface for implementing graph algorithms.* All levels are highly connected and

can be used by students for their learning process as well as by the teacher for preparing and supporting his lessons.

## 2.1. The Visage graph laboratory

Learning about graphs usually starts with exploring the basic attributes of them. The best activity to examine those attributes is to play around with graphs, i.e. construct some graphs and compare their characteristics. In particular, it is useful to construct examples for difficult circumstances for a better understanding of some properties. An environment for such an activity of students is called a *graph laboratory* (see Lutz-Westphal [10]).

One problem when working with a graph lab using paper and pencil is that humans tend to make mistakes and to overlook some hidden effects, just because they think that the construction was meaningful. And so, some properties are left undiscovered due to subconscious cheating.

It is much harder to make such mistakes when using a *virtual* graph laboratory, because the computer works in the background as an trustworthy referee. It can detect the correctness of a students construction and gives immediate or delayed feedback.

With Visage we created such a virtual laboratory. Visage provides a virtual sheet of paper on which students can draw an arbitrary graph, i.e. they can draw *vertices* and *edges* just like *points* and *segments* in a geometry software. Depending on the aspects to research they can assign weights or directions on the edges of the graph. At any time students can change the graph, add new vertices and edges or delete previously drawn ones, or they can move the vertices of their graph by dragging them with the mouse.

This setup enables students to explore certain basics aspects easily: They can explore the *structural* aspects of graphs, as opposed to just arbitrary line drawings, because they are forced to create vertices and edges, and they can experience that these structural (or topological) *aspects are invariant* under certain changes of the graph embedding, like moving vertices.

In a next step, students can assign built-in algorithms to their graphs. The software features two classes of algorithms: One focusses on additional properties, like showing the adjacency matrix or the degrees of the vertices. The other contains classical text book algorithms for calculating shortest paths, minimum spanning trees or Eulerian tours, and more. Figure 1 shows the visualization of the algorithm of Dijkstra on a small graph in the Visage graph laboratory. The chosen algorithms can be executed stepwise and come with pseudo code that

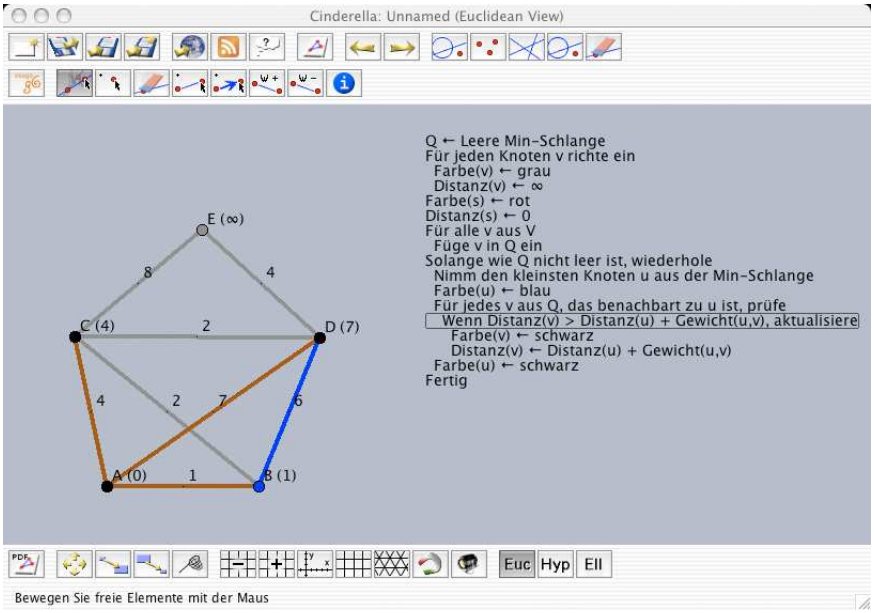


*Figure 1. The Visage Graph Laboratory* Students can draw graphs and assign algorithms to them.

can be followed during the execution. Here again it is possible to get a deeper understanding and to explore what is happening. Students are able to answer research questions such as "What happens to a shortest path tree when I delete a certain edge?" or "How must a graph look like in order to be Eulerian?"

Each construction in the graph laboratory can be enhanced by using the built-in programming interface which gives additional possibilities to study graph properties, which we will describe below.

## 2.2. Authoring interactive material

Cinderella was one of the first interactive geometry software packages that offered the possibility to export geometric constructions as a Java applet which can be integrated into any HTML document [15]. This feature can also be used in conjunction with Visage to export graphs and algorithms. Any teacher can publish interactive worksheets for their class on the web or in a learning repository such as *Moodle*.

Such worksheets can be used to enrich a lesson by single interactive exercises. Here again, the teacher can prepare a graph laboratory that is available to the students, but now the available tools may be restricted for the students, as chosen by the teacher. The advantage of this restriction is that the students are guided and will not be confused by the endless possibilities of the full geometry application.

Teachers can bundle a sequence of such activities as a complete learning unit for a special topic. According to the new Berlin curriculum for Mathematics in secondary schools [2], we exemplarily developed two interactive teaching units for the topic "Optimal Paths". Figure 2 shows screenshots from both units.


*Figure 2. Interactive Teaching Units* We developed two units to support the new Berlin curriculum.

The first unit called "Wie fährt die Müllabfuhr?" ("How to route a garbage collector?") is about the Eulerian tour problem. The second learning unit is called "The Shortest Path Problem". While the aim of the first unit is the discovery of some basic properties of graphs, the second unit explains the working of two algorithms for constructing a shortest path in a graph.

Both modules are based on the same didactical concepts. The main idea is to present problems to the students they can answer using the software. The software itself does not answer the given questions but offers an environment to explore the underlying theory by the students. This helps them to find the answers on their own. For example, each learning unit contains a dictionary that can be used as a tool to discover unknown ideas and terms. The dictionary contains all theory

specific terms that are used in the unit. But also there are some unused terms acting as hidden hints. A detailed description of one of the units and its didactical principals can be found in [14]. Some first experience in using the software are reported in [9].

Using the authoring tool of our software package is an opportunity for the students as well. For example, they can use this tool to present the results of their work. In a project oriented learning environment students acquire knowledge in a self-organized way. At the end of a sequence of lessons the students should present their results in order to secure their progress and to train their skills in communicating mathematics. Such a presentation might be done in various ways: by creating a poster, writing a short report, giving a presentation or – using the possibilities of the new media – by creating a web site on the school's internet server. If they decide to do the last one, they can enrich their presentation with interactive graphics created with the Visage graph lab as a Java applet.

### 2.3. Visage and the CindyScript programming interface

We identified two main reasons why it is desirable to provide a programming interface for interactive learning software for graphs and algorithms.

The first reason is the intrinsic motivation mentioned in Section 1. Studying combinatorial optimization presupposes that students think about strategies for finding a problem's solution. Students should develop own ideas for such algorithms and they are forced to understand and execute the standard textbook algorithms. In most cases this will end up in a situation where the students might want to implement algorithms on their own, either in order to verify their findings or to better understand a textbook algorithm by modifying it and observing the consequences of the changes.

Beside this, there is also a very technical aspect that requires the existence of such a programming interface. The developers of the software cannot think of all possible applications of the Visage software. Many teachers – and students as well – are extremly creative in finding new ways to deal with graph-algorithmic topics. They develop new ideas which attributes of graphs could be explored and which hidden hints the computer should offer. The pool of build-in algorithms of visage can only support a fraction of all possible applications. This requires that users and authors of activities can extend the algorithms and visualizations accordingly, which is only possible with a programming interface.

In both scenarios it is important to offer the possibility to program own algorithms that can be applied to graphs in the Visage graph lab or that can be

exported to interactive Java applets. In doing so, the focus should not lie on the teaching and learning of a programming language, in particular if implementations are done by the students. In fact, the programming should be done in a problem oriented way and only on demand. Then the students are able to learn mathematical skills like modeling, abstraction, formalization and structuring. Furthermore it should be possible to *experience* the result of an implemented algorithm visually with a lot of custom examples without much effort.

The integrated programming language of Cinderella, CindyScript, fulfills these needs. CindyScript is a functional programming language. Each statement does return a value, just like a mathematical function. This result again can be used as an input parameter for a further function. This also holds for control structures like loops or conditional branches, which are implemented as CindyScript functions. This matches the mathematical way of thinking better than any imperative programming language.

Additionally, each of the geometric objects in a Cinderella construction is mapped to a corresponding CindyScript object. This allows for direct access and manipulation of all geometric objects. Using this technique, it is easy to implement visualizations of the results of programmed algorithms, for example by changing colors or moving elements.

The programming code can be assigned to the Cinderella `draw`-event, i.e. each time the construction is redrawn, the program will be executed. Whenever the graph in the Visage graph lab was changed, the algorithmic results are immediately updated. New results of the algorithms are visualized in real-time (or near-real-time, if the algorithm is more complex) whenever the vertices are moved, vertices and edges are added or deleted or the weights of edges are varied.

As CindyScript does not support special attributes of graphs (weigths or directions) in its core, we are developing an extension of CindyScript consisting of functions for accessing the elements and properties of a graph. Basic graph algorithms have been implemented already as well. The Visage graph library is written in CindyScript itself. Users have full access to the functions and the complete programming code. Students or teachers can re-use the implemented algorithms as a programming tutorial or as templates for their own algorithms. Also, students can use them as a basis to study the effects on the algorithms of minor changes of the code.

In the next section we show how to use the programming language to let students implement a minimum spanning tree algorithm.

## 3. Programming graph algorithms within Visage

Using prepared learning units as interactive activities in a classroom offers additional possibilities for individual and self-directed learning. In [9] it was shown that students hold these options in high regard.

Nevertheless, in a survey conducted after the classroom tests the students critizised one disadvantage of the presented units: students don't have direct access to the embedded algorithms. All algorithms used in the modules are implemented using a black box principle, even if some of them are shown as pseudo code or are executed step by step. The students neither have a possibility to influence the behavior of the algorithms nor they can change them at all.

Questions like "What will happen if I change the order of nodes to be treated in the algorithm" that can be starting points of inspiring and englightening discussions cannot be answered in that framework. The learning environment cannot accomodate structural changes in the algorithms or own inventions of the students. We resolve this deficit of the pure visualization of algorithms using the programming interface of Visage.

We provide a collection of additional functions for the built-in programming language CindyScript. These functions allow for full access to all attributes of a graph drawn in the graph lab. A set of standard graph algorithms is already implemented. All available functions and algorithms are written in CindyScript themselves and their source code can be examined and changed. This enables students – and the teachers as well – to import and change the programming code in the script editor of the Visage graph lab.

We want to present the programming interface in this article exemplarily with an activity that was used in a teacher students' course. A detailed description of the interface is available on the website of the Visage project [7]. A complete documentation of the CindyScript programming language is given in [17]. For a short CindyScript tutorial see [6].

We want to stress the fact that our aim is not to teach students programming as an end in itself. The students should learn how programming can be used as a tool to answer questions and to solve problems. The question whether a programming language is indeed need is a valid one. There are many problems that can be solved by widely accepted programming environments like computer algebra systems (CAS) or spreadsheet applications like Excel. However, problems in graph theory are geometric in nature and it is only natural to start computer-based explorations in a geometry software. The necessary interactions and structuring

can be stated informally, or in a formal language. The drawback of informal languages is that the correctness of answers and the necessary exactness can only be checked by experts (teachers), not by the students themselves. By introducing a formalized way of notation we enable students to self-assess their solutions with the help of a computer.

### 3.1. Programming minimum spanning tree algorithms

A well known optimization problem on graphs is the minimum spanning tree problem, cf. [5] and [13]. Given a graph with edge weights, you want to find a spanning tree of minimum total weight, i.e. a cycle-free subgraph that connects all vertices of the given graph and the sum of the weights of all chosen edges should be as small as possible.

```
// Algorithm of Prim
vertices=allvertices();               // List of all vertices
numb=length(vertices);                // Number of vertices
tree=[vertices_1];                    // The first vertex is the starting tree vertex

edges=sort(alledges(),getweight(#));  // List of all edges sorted by weight
forall(edges,#.color=[1,0,0]);        // coloring all edges red

forall(1..(numb-1),i,                 // We have to chose n-1 edges

   newtreeedge=false;                 // no new tree edge found yet

   // find cheapest edge leaving the tree
   forall(edges,edge,

      vw=incidentvertices(edge);      // the two incident vertices of the edge
      vwtree = common(tree,vw);       // which vertices belong to the tree?

      // Two tree vertices? Closed cycle, ignore edge!
      // No tree vertex? Edge not connected to tree, ignore!
      // One tree vertex? Found tree edge!
      if(length(vwtree)==1,

           if(newtreeedge==false,          // Only use the cheapest (first) edge!
              newtreeedge = true;          // Found tree edge!
              edge.color=[0,1,0];          // green!
              tree = tree ++ (vw--vwtree); // Add vertex to tree
          );
      );
   );
);
```

*Figure 3.* CindyScript implementation of the algorithm of Prim

A typical[1] application for this problem is the planing of optimal telecommunication networks. Consider a company that has several offices all over the country. Now, this company connects all offices by a telecommunication network. Therefor they must rent the desired cable from a telecommunication company. For each line they hire, the company has to pay a certain amount. The goal is to connect all offices and paying as less as possible. Other applications that fit to an authentic mathematics instruction are presented in [10].

There are some very elementary solution algorithms known for the minimum spanning tree problem. Some of them follow the principals of so-called *greedy* algorithms. Those algorithms can easily be discovered by students on their own in school.

Two well-known algorithm for minimum spanning trees are the algorithms of Prim and Kruskal. We recommend these two for classroom investigations. Let us shortly recall these algorithms below.

The algorithm of Prim starts with an arbitrary vertex, which now is marked as "connected." All other vertices are unconnected. Now, in each step the algorithm chooses the cheapest edge (i.e. the edge with the least weight) connecting a connected vertex with an unconnected one. The new vertex is then marked as being connected. This is repeated until all vertices are connected. Figure 3 shows the implementation of this algorithm using CindyScript. The result as it is shown in the Visage graph lab for an example graph are displayed in Figure 4.

Note that the presented implementation of the algorithm does not have the best known complexity as found in literature, cf. [5] and [13]. But for visualization and didactical purposes it is more important to present a simple implementation than a sophisticated coding using advanced data structures.

The second algorithm, the algorithm of Kruskal seems to be quite similar at first sight. But now, there is no distinguished vertex to start with. In each step the cheapest edge connecting two vertices that are unconnected until now is chosen.

Whereas Prim always works on one tree that is continuously growing, Kruskal constructs a bundle of smaller trees and merges two of them in each step.

For teaching the algorithmic solution of the minimum spanning tree problem we propose to let the students develop the algorithms on their own. We expect them to discover both of the mentioned algorithms or very similar ones. For

[1]Though typical, this problem is a little bit artificial. Usually, problems in combinatorial optimization contain many subtleties, which makes them more difficult than expected. Still most solutions are based on algorithms that are derived from the basic algorithms as presented here.

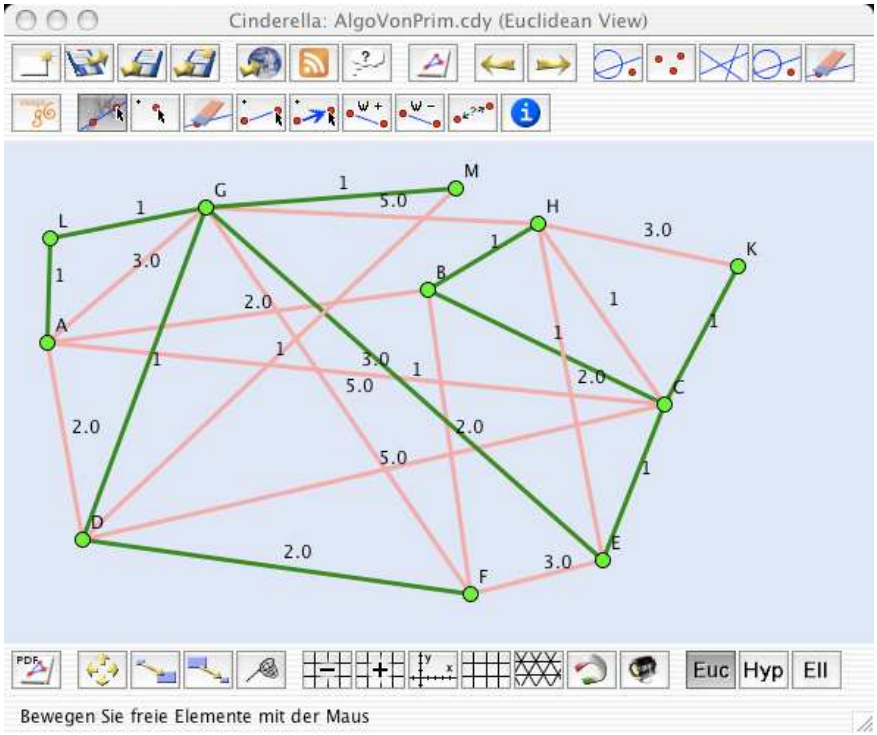


*Figure 4. A minimum spanning tree.* The result of the Prim's algorithm (cf. Figure 3) is visualized in the Visage graph laboratory.

a further examination of the algorithms an implementation of the algorithms is necessary. We suppose to present one of the algorithms to the students and let them implement the other on their own. Since Kruskal is less restrictive in the conditions for the next edge to be chosen we recommend to set the algorithm of Prim in advance and let the students implement Kruskal in a comparable manner. Further studies can be done by trying to implement both algorithms as similar as possible.

### 3.2. Experiences with the programming approach

We completed the first tests of the presented concept in a course "Discrete Mathematics and its Applications" for teacher students at the Technical University of Berlin held by B. Lutz-Westphal in the summerterm of 2007. This

course consisted of a four-hour lecture on combinatorial optimization problems and its school didactics. It was complemented by a two-hour exercise on typical techniques of discrete mathematics and algorithmic analysis. Thirty students attended the course. They had no or only little experience in programming. None of them ever used CindyScript before.

The course was designed based on the requests of many students to learn how to program for teaching. To fulfill this demand, the students had to work on two optional programming exercises besides a series of theoretical exercises. The first programming exercise was a short tutorial in programming CindyScript and was meant to train them in implementing an interaction between the programming interface and the graphical worksheet of Cinderella. The students had to implement an interactive visualization of a three term recursion. The results were very promising and they were assigned a second exercise on the minimum spanning tree problem.

The main programming exercise was to implement the algorithm of Kruskal. In an introducing lecture we presented the programming code for the algorithm of Prim shown in Figure 3 and explained all the necessary programming commands from the Visage programming library. Additionally, the students could make use of a pseudo code representation of the algorithm of Kruskal, so that they just had to fill in the blanks with corresponding CindyScript commands.

The students did not have access to teaching assistens during their work with the exercises. They could ask questions in during the regular office hours, though.

We could observe the typical beginners' problems in programming. A particular problem poses the insufficient debugging features of the Cinderella script editor. It was very difficult for the students to overcome syntactical errors they made, but a face-to-face programming supervision would have avoided most of the problems.

Nevertheless, 22 students completed the exercise. Most of the solutions were absolutely satisfying and some of the students solutions did exceed our requests by far. For example, some students used the graphical elements of Cinderella to implement a sophisticated user interface for their visualization. Other students implemented plausibility checks (*unit tests*) for their results, a thing that was not requested but is very reasonable for programming complex algorithms.

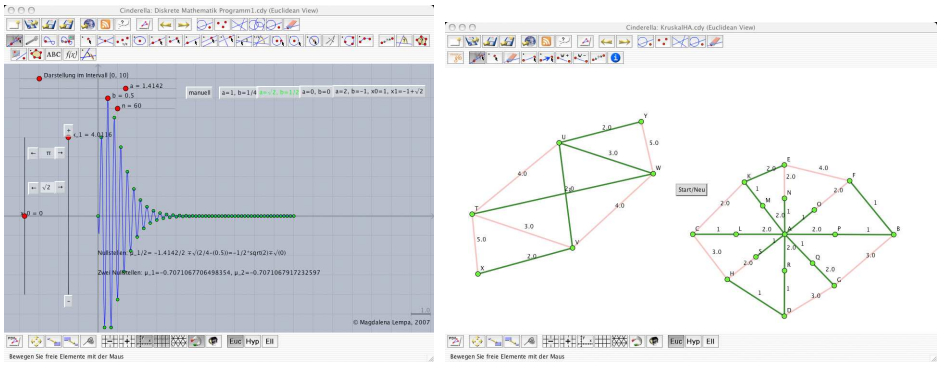

*Figure 5. Students' results.* Two students' solutions are shown. The left picture shows a sophisticated user interface for an interactive visualization of a three term recursion. The right picture is a visualization of Kruskal's algorithm for minimum spanning trees.

After completition of the exercise we asked the students whether they can imagine a similar exercise for high school students. Due to their own problems with the implementation only six of the participants believed that pupils are capable of solving the task. Those six were solely those students who succeeded themselves successfully, and among them mostly the women.

Here again, a more intensive personal support during the programming work of the students could have decreased their fear for problems. In school it is required that the teacher himself has enough experience in programming to make him compentent enough to help his pupils. Hence, in our view learning how to implement mathematical problems with the help of a computer must be an integral component of teacher education.

Asked for their personal opinion, most of the participants of our course answered that they hat much fun in solving the exercises or at least that it was an interesting experience. Many of our students liked to learn more about programming if they had any time besides their mathematical studies.

We want to conclude with a quotation from one of our students:

> "Besides *[all the problems we had – remark of the authors]* it increased my joy with programming and showed me how to do pretty nice things using the computer with just a little effort."

## Conclusions and future work

Using the combination of Visage and the built-in programming language CindyScript opens a great pool of new possibilities for our software-based teaching. On one hand, it is a flexible way to implement useful visualizations of algorithms. Further learning units will be created using these tools. On the other hand it has become easier to integrate programming of (graph) algorithms into school education. Now, teachers and student have a substantial tool to program the algorithms they developed as well as textbook algorithms. It must be determined wether the implementation of algorithms leads to a better mathematical understanding, though.

In December 2007 a project week with the topic "*Spannungen*" at a cooperation school in Berlin will take place, and we will use the software there. The german word "Spannung" means "friction" and "strain" as well as "tension", "pressure" or "voltage". In the figurative sense the meaning of "spanning" also fits to the subject of the project week. We will offer a student project on minimum spanning trees. The experience we expect to gather there with regard to the programming framework will be used to bring forward similar exercises and additional combinatorial optimization topics.

## Acknowledgments

Most of the implementation of the Visage graph lab was done by Dirk Materlik. A prototype of the interactive learning unit on "shortest paths" was written by Anne Geschke. We would like to thank Brigitte Lutz-Westphal who gave some helpful advise.

## References

[1] M. Aigner, C. Bänsch, M. Grötschel, B. Lutz-Westphal, A. Unterreiter and G. M. Ziegler, Lebendige mathematik! Berliner thesen zum mathematikunterricht, in: *Mitteilungen der Deutschen Mathematiker-Vereinigung*, Vol. 4, 2003, 29–31.

[2] Berliner Landesinstitut für Schule und Medien, Rahmenplan für die Sekundarstufe I, Mathematik, Senatsverwaltung für Bildung, Jugend und Sport, Berlin, 2006.

[3] R. Bodendiek and H.-G. Bigalke, *Graphen in Forschung und Unterricht*, in: *Festschrift K. Wagner*, Franzbecker, 1985.

[4] R. Bruder and H.-G. Weigand, eds., Diskrete mathematik, *Mathematik Lehren* **129** (2005).

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, *Introduction to Algorithms*, Second edition, MIT Press, Cambridge, 2001.

[6] A. Fest, *CindyScript – Ein kleiner Programmierkurs*, Berlin, 2007, `http://www.math.tu-berlin.de/~fest/cindyscript`.

[7] A. Fest, U. Kortenkamp, A. Geschke and D. Materlik, *Visage – A software package for the Visualization of Algorithms with Geometry Software*, Berlin, 2007, `http://cinderella.de/visage/`.

[8] H. Freudenthal, *Mathematik als pädagigische Aufgabe*, Klett, Stuttgart, 1973.

[9] A. Geschke, U. Kortenkamp, B. Lutz-Westphal and D. Materlik, Visage – visualization of algorithms in discrete mathematics, *Zentralblatt für Didaktik der Mathematik* **37**, no. 5 (2005), 395–401.

[10] S. Hußmann and B. Lutz-Westphal, eds., *Kombinatorische Optimierung erleben*, Studium und Unterricht, Vieweg, Wiesbaden, 2007.

[11] M. J. Kenny and C. R. Hirsch, editors, *Discrete Mathematics across the Curriculum, K-12*, Yearbook, National Council of Teachers of Mathematics, Virginia, USA, 1991.

[12] H. Kletzl, *Daten- und Beziehungsstrukturen. Eine didaktische Analyse im Spannungsfeld von angewandter Informatik und angewandter Mathematik*, Doctoral dissertation, Universität Salzburg, 2002.

[13] B. Korte and J. Vygen, *Combinatorial Optimization. Theory and Algorithms*, Second edition, Springer, 2002.

[14] U. Kortenkamp, *Guidelines for using computers creatively in mathematics education*, in: *Proceedings of the first KAIST workshop on enhancing university mathematics teaching*, Daejon, Korea, 2005.

[15] U. Kortenkamp and J. Richter-Gebert, *Geometry and education in the Internet age*, Thomas Ottmann and Ivan Tomek, editors, *Ed-Media & Ed-Telecom 98. Proceedings of the Tenth World Conference on Educational Multimedia and Hypermedia & World Conference on Educational Telecommunications, Freiburg, Germany, June 20-25, 1998*, Charlottesville, 1998, AACE.

[16] U. Kortenkamp and J. Richter-Gebert, *The interactive Geometry Software Cinderella*, Springer, Heidelberg, 1999, `http://cinderella.de/`.

[17] U. Kortenkamp and J. Richter-Gebert, *Cinderella.2 Documentation*, 2005, `http://doc.cinderella.de/`.

[18] Kultusministerkonferenz, eds., Bildungsstandards im Fach Mathematik für den Mittleren Schulabschluß, Luchterhand, Darmstadt, 2003.

[19] B. Lutz-Westphal, Erlebnis Mathematik – Kombinatorische Optimierung im Unterricht, in: *Mitteilungen der Deutschen Mathematiker-Vereinigung*, Vol. 2, 2004, 78–81.

[20] B. Lutz-Westphal, *Lebendiger Mathematikunterricht mit kombinatorischer Optimierung*, in: *Beiträge zum Mathematikunterricht*, Franzbecker, Hildesheim, 2004, 353–356.

[21] B. Lutz-Westphal, *Kombinatorische Optimierung – Inhalte und Methoden für einen authentischen Mathematikunterricht*, Dissertation, TU Berlin, 2006.

[22] O. Ore, *Graphen und ihre Anwendungen*, Klett, Stuttgart, 1974.

[23] H.-C. Reichel and T. Kubelik, *Mathematik – verborgen und dennoch allgegenwärtig. Außermathematische Anwendungen der Mathematik - eine neu konzipierte Lehrveranstaltung zur Mathematikdidaktik*, in: *Mathematik unsichtbar und doch allgegenwärtig*, Polygon Verlag, Buxheim, 2002.

[24] W. Renz, W. Euber, G. Kaiser, W. Löding and J. Weitendorf, *Rahmenplan Mathematik, gymnasiale Oberstufe*, Behörde für Bildung und Sport, Hamburg, 2004.

[25] A. Schrijver, *Kleuren en routeren in grafen*, Notes for high school teachers masterclass, `http://www.cwi.nl/~lex/`.

[26] A. Schuster, *Kombinatorische Optimierung als Gegenstand der Gymnasialdidaktik im Umfeld von Mathematik- und Informatikunterricht*, Habilitationsschrift, Würzburg, 2004.

[27] H. Wippermann, *Graphen im Unterricht*, 1976.

ANDREAS FEST
PÄDAGOGISCHE HOCHSCHULE SCHWÄBISCH GMÜND
OBERBETTRINGER STR. 200
D-73525 SCHWÄBISCH GMÜND
GERMANY

*E-mail:* `andreas.fest@ph-gmuend.de`

ULRICH KORTENKAMP
PÄDAGOGISCHE HOCHSCHULE KARLSRUHE
BISMARCKSTR. 10
D-76133 KARLSRUHE
GERMANY

*E-mail:* `ulrich.kortenkamp@ph-karlsruhe.de`

*(Received October, 2007)*