Visage – Visualization of Algorithms in Discrete Mathematics[1]

Anne Geschke, Ulrich Kortenkamp,
Brigitte Lutz-Westphal, Dirk Materlik
Berlin (Germany)

Abstract: *Which route should the garbage collectors' truck take?* Just a simple question, but also the starting point of an exciting mathematics class. The only "hardware" you need is a city map, given on a sheet of paper or on the computer screen. Then lively discussions will take place in the classroom on how to find an optimal routing for the truck. The aim of this activity is to develop an algorithm that constructs Eulerian tours in graphs and to learn about graphs and their properties.

This teaching sequence, and those stemming from discrete mathematics, in particular combinatorial optimization, are ideal for training problem solving skills and modeling – general competencies that, influenced by the German National Standards, are finding their way into curricula.

In this article, we investigate how computers can help in providing individual teaching tools for students. Within the Visage project we focus on electronic activities that can enhance explorations with graphs and guide students even if the teacher is not available – without taking away freedom and creativity.

The software package is embedded into a standard DGS, and it offers many pre-built and teacher-customizable tools in the area of graph algorithms.

Until now, there are no complete didactical concepts for teaching graph algorithms, in particular using new media. We see a huge potential in our methods, and the topic is highly requested on part of the teachers, as it introduces a modern and highly relevant part of mathematics into the curriculum.

Kurzreferat: *Wie fährt die Müllabfuhr?* Eine einfache Frage, die der Ausgangspunkt für einen anregenden Mathematikunterricht sein kann. Man benötigt nichts weiter als einen Stadtplan, auf Papier oder auf dem Computerbildschirm, und schon kann die Arbeit beginnen. Lebhafte Diskussionen entfachen sich darüber, wie eine optimale Fahrtroute für das Müllauto gefunden werden kann. Um dieses Problem zu lösen, erarbeiten Schülerinnen und Schüler Algorithmen für Eulertouren und finden einiges über Graphen und ihre Eigenschaften heraus.

Diese Lerneinheit und andere aus der diskreten Mathematik – und insbesondere der kombinatorischen Optimierung – sind ideal um Problemlöse- und Modellierungskompetenzen zu entwickeln; allgemeine Kompetenzen, die, beeinflusst durch die neuen Bildungsstandards, Eingang in die Lehrpläne der Länder finden.

In diesem Artikel untersuchen wir, wie Computer sinnvoll als Werkzeuge individuellen Lehren und Lernens in diesem Gebiet eingesetzt werden können. Im Visage-Projekt entwickeln wir elektronische Arbeitsmaterialien, mit denen Untersuchungen von Graphen erleichtert werden können, sowie auch ohne Unterstützung durch Lehrerinnen und Lehrer durchgeführt werden können. Dabei legen wir Wert darauf, die Freiheit und Kreativität der Schülerinnen und Schüler nicht zu beschränken.

Das vorliegende Software-Paket ist in ein bestehendes DGS

integriert und viele Module und Algorithmen sind bereits eingebaut und einfach durch die Lehrkräfte anzupassen.

Bis heute gibt es keine vollständigen didaktischen Konzepte für das Unterrichten von Graphenalgorithmen, insbesondere nicht mit neuen Medien. Wir sehen hier ein großes Entwicklungspotenzial durch unsere Methoden. Zudem gibt es einen großen Bedarf seitens der Lehrkräfte, die das moderne und relevante Thema Graphenalgorithmen gerne im Unterricht umsetzen.

ZDM-Classification: D30, K30, N60, N70, U70

1   New Directions Through Standards

Discrete Mathematics is coming to schools. In other countries there already are initiatives that try to introduce more of discrete mathematics into mathematics teaching, e.g., in the USA (Kenny 1991), Austria (Reichel 2002) and the Netherlands (Schrijver 2001?). Germany is moving, too: the new curriculum of Hamburg includes an optional module of combinatorial optimization (Renz et. al. 2004). Several other German federal states are also thinking about integrating discrete mathematics in their curricula, or did this already.

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. We also refer to Freudenthal (1973), who remarks that it is accepted by many that students have to be able to mathematize non-mathematical content, i.e. order it in a mathematical way that makes it possible to refine its structure by mathematical means[2] – he is not the first to lay ground for today's standards, though.

It is not required that these competencies be gained from classic school subject matter. This affords new freedom in choosing topics for school use.

The idea of using topics from discrete mathematics in school is not new. Starting in the 1970's several authors, (Ore 1974, Wippermann 1976, Bigalke 1985) recommended graph theory as a suitable topic to be established in school. However, this did not catch on, maybe because the approach was geared too much towards theory. Our approach, today, is different: problem-oriented, application-based and more geared towards algorithms. Thus, the topic does not stand isolated, but can be linked to different knowledge domains from mathematics and other fields. This can be achieved either by the applications or the required problem solving and modeling competencies (compare Hußmann and Lutz-Westphal 2005, Kletzl 2002, Aigner et al. 2003, Lutz-Westphal 2004a, 2004b, Schuster 2004, Bruder and Weigand 2005).

Some applications stemming from discrete mathematics offer a particularly quick road to the mathematical problem without requiring much prior knowledge (cf. our

---

[2] *"Es wird heute von vielen anerkannt, daß der Schüler auch lernen muß, einen nicht-mathematischen Stoff (oder nicht genügend mathematischen Stoff) zu mathematisieren, d.h. so zu ordnen, daß er eine mathematischer Verfeinerung zugängliche Struktur erhält."*

example in Sect. 3). The mathematical theory can be developed by the students themselves from questions they ask naturally. Problem solving and modeling processes work together. The requirement of the *Bildungsstandards* (Kultusministerkonferenz 2003) that "students experience mathematics as a stimulating, useful and creative field of activity, in which tools, especially electronic media, are used in a reasonable way,"[3] can be fulfilled with topics such as this. Graphs allow for a particularly approachable mathematics exercise, as students themselves can quickly generate examples for experiments.

Recently, the German Research Foundation (DFG) has established the MATHEON[4], an institutaion that is devoted to doing applied mathematical research for key technologies, e.g. life sciences or telecommunication networks. It is organized both by application areas and three mathematical fields. One field is "Optimization and Discrete Mathematics", while one application area is "Education". A major goal of MATHEON is the advancement of mathematics in the public view, and the application area Education is also responsible for the implementation of this goal.

This is evidence for the universal suitability of discrete mathematics in educational contexts. The topic is easily accessible for non-mathematicians, the general public, as well as for beginners, i.e. in K-12 education. Parts of our work have been used in talks for the general public or in the "Uni for Kids", an outreach event for 10-year old children. It is also appropriate for school activities that should prepare for more theoretical work in higher grades. Some of the algorithms we are teaching, respectively we use for teaching are currently standard material in the beginners' courses on computer-oriented mathematics taught to bachelor students in mathematics, and they are the basis for advanced research in Ph.D. programs. Last, but not least, the projects within MATHEON show their immediate necessity for industry, and prove their economic relevance.

2 Individual Learning with Interactive Media

This article concentrates on the use of computer software to teach and learn about graph algorithms. Before going into the details, we shall to motivate our computer-centric research.

Why use a computer at all? First, there is no way to escape the mechanization of teaching, as this is not a didactical question, but rather a political one. Parents, and thus the government, are pushing technology into teaching, and rely on teachers to use this for "better" teaching. A lack of concepts will not stop this development, so we should try to come up with good concepts now.

It is not enough to claim the motivational aspects of using computers in the classroom. This may have been

possible in times where students did not have access to computers outside school. As this has radically changed in the last years, and students may have betters computers at home than we can offer them in class, we should not rely on this effect.

If we look at the possibilities offered by the computers, we note that they allow for individual learning speeds and degrees of difficulty. This makes them – by design – a good tool for internal differentiation, if used correctly. We can exploit this in our context if the software is open for individual progress.

While the above is a general property of computers in teaching, we also have a two-fold intrinsic reason for using these machines in teaching. First, all serious real-world problems in combinatorial optimization are solved using computers. Thus, teaching is more realistic if it takes advantage of the computer as well.

Second, by understanding the modeling done *for* the computer, we can also understand why we have to model at all, and exercise it. In the example in the next section we will come back to this, when we discuss the forced modeling. We also have the chance of exhibiting a very important detail of modeling: using the right model, a computer might be able to solve the problem quickly. But if we are using an inferior model, we might end up with a problem that is too big even for the fastest computer.[5]

3 Exemplary Use of a Computer in Teaching Discrete Mathematics

In this section, we present an example activity in graph algorithms, using computer-based support material.[6] This material has been presented from a different perspective in (Kortenkamp 2005).

3.1 Students' Situation

The material that is introduced in this section has been used for students of grades 4 to 8 (10 to 14 years old) at all levels; nevertheless, it could be adapted to university teaching as well, as it is concerned with an important basic topic from Discrete Mathematics. This activity is intended to be used for the introduction of the concept of graphs, and to explore the notion of Eulerian tours.

The general topic is the optimization of tours for garbage collection. Clearly, every student has some basic knowledge about garbage collection. Watching garbage collection trucks is an experience from early childhood. The importance of optimizing the tours of the garbage collection is obvious.

---

[3] Original text: *"Schülerinnen und Schüler [sollen] Mathematik als anregendes, nutzbringendes und kreatives Betätigungsfeld erleben, in dem auch Hilfsmittel, insbesondere elektronische Medien entsprechend sinnvoll eingesetzt werden"*

[4] http://www.matheon.de

[5] Renz et al. (2004) mention this in the Hamburg curriculum, although a little bit intricately: *"Einige der Probleme der Graphentheorie können auch mit Computereinsatz nicht gelöst werden. Hier ist eine geeignete Modellierung unerlässlich."*. English translation: *"Some graph theoretic problems cannot be solved, not even with a computer. Here, a suitable modelling is necessary."*

[6] See http://cinderella.de/visage/ for the electronic exercise sheets and downloadable material.
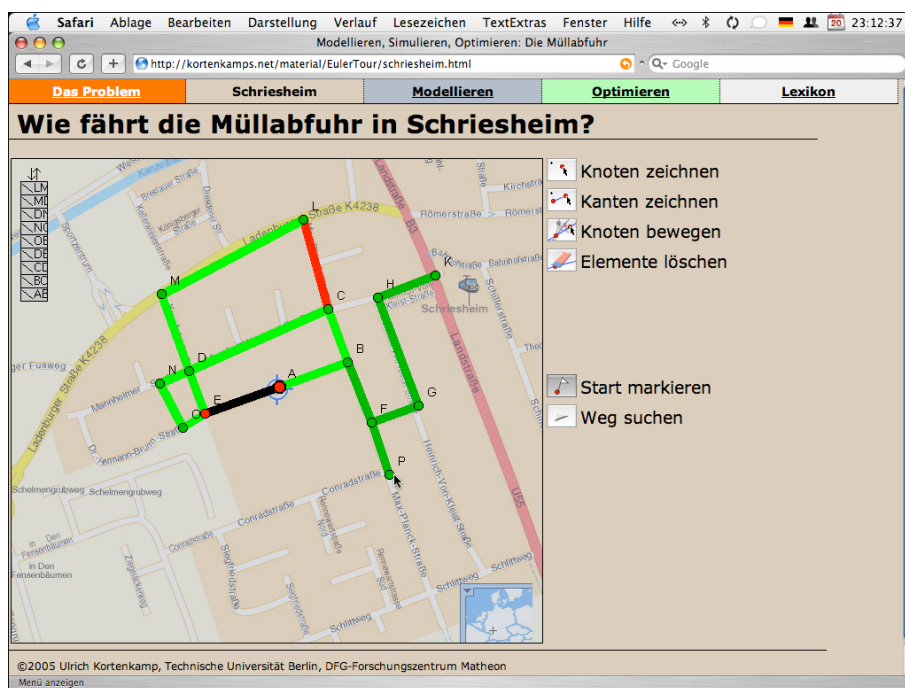
Fig. 1: A screenshot of an applet used to teach Eulerian graphs and tours to 13-year old students. The general topic of the sequence is "What is the tour of a garbage collection vehicle?" Students can draw a graph onto a map of their hometown Schriesheim and ask for a Eulerian tour through it.

Using a computer for finding the shortest, fastest, nicest, or simply the best route from one place is common practice. Route finding on the Internet is incredibly fast, and seems to be trivial.[7] Most students do not care about how these routes are found, though. They do not care about whether these routes are pre-computed and stored in a database, or computed using clever algorithms. Actually, although we have no formal evidence, it seems likely that students have *no idea* that it is actually a mathematical problem that is solved there.

In summary, we have a real-world problem that is closely connected to the personal experiences of the students, and students who know that computers are used to solve similar problems, but do not know how this is done.

3.2 Activity: Garbage Collectors' Route

Based on the observation that the garbage collection vehicle has to go through every street at least once – which is also sufficient, as two garbage collectors serve both sides of the street – students explore tours that do not use any street twice. First, they work on an electronic exercise sheet that shows a city map of their hometown. They can add vertices and edges of a graph, and they can run an algorithm that tries to find a round-trip tour that visits all edges exactly once.

The electronic exercise sheet is an HTML page with an embedded Java applet. The Java applet was created using the Visage-Extension (Kortenkamp and Materlik 2005) of

Cinderella (Richter-Gebert 1999). This exercise can run standalone from a local disc or CD-ROM or it can be put on the web. Besides a Java runtime, which is included by default on major operating systems, no software installation is necessary.

As a next step, we let the students work with pencil and paper. They are supposed to check by hand whether a given graph admits a Eulerian tour, i.e. a closed tour that visits every edge exactly once.

The solutions will not be checked by the teacher, but by the students themselves. They use the next electronic work sheet to draw the graph and ask for a Eulerian tour. If there is one, the computer will show one; otherwise the algorithm will fail and highlight a problematic part of the graph. In Fig. 1 we can see a graph drawn by a student.

During this phase, students naturally will ask themselves[8] whether there is an easy way to distinguish between Eulerian graphs and non-Eulerian ones. The answer to this question is positive – any graph that contains only vertices of even degree, that is, has an even number of edges incident to it, is Eulerian, and only these. The proof of that fact is easy from a mathematician's point of view, and can be done by an induction argument, but discovering this conjecture is not as easy.

In this activity, we have added a subliminal clue for arriving at that concept. The vertices in Fig. 2 are colored automatically in either white or black, depending on their degree. White vertices have an even degree; black vertices have an odd degree. Our experience is that students will not notice the coloring at all at first. However, when they investigate possible reasons for some graphs to be

---

5   The service Google Earth is a premier example: Route finding seems to be incredibly simple, within seconds we can see the best route and see a preview of it in a stunning visualization.

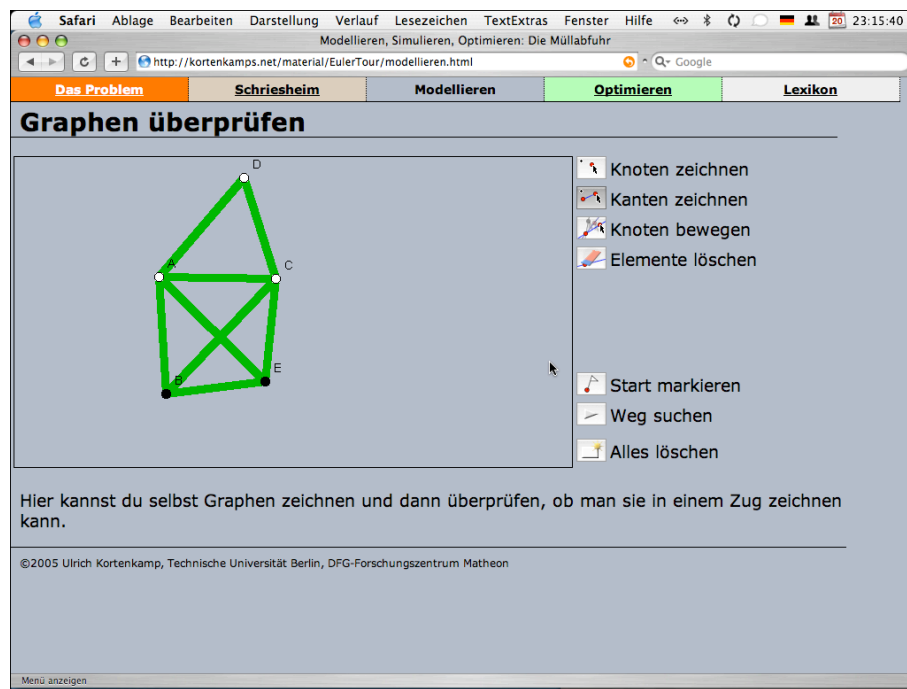8   If they do not ask themselves, the teacher should activate this discussion

Fig. 2: The open learning environment for drawing and testing graphs. On the left hand side we see a 5-vertex graph drawn by the student. The coloring of vertices is done automatically (see text). On the lower right there are buttons for selecting a start vertex and starting the algorithm that finds Eulerian tours.

"good" and others being "bad," they can use this coloring as a first hint: Graphs that were proved being Eulerian (using the built-in algorithm) have only white vertices.

The last part of the sequence is another electronic work sheet (Fig. 3) that opens the problem further and can be used a starting point for other, related activities. Students are given a graph that contains both black and white vertices, and their task is to change the color of all vertices to white. This is always possible – a theorem about graphs states that every graph contains an even number of odd-degree vertices, so we can connect pairs of them with paths. As the students do not know the theorem yet, they are invited to play a game where they should try to give a graph that their classmates cannot complete to a Eulerian graph.

Building on this, we could either pursue different directions, or bring this sequence to an end. Possible continuations are, for example, proofs for various graph properties, shortest-path algorithms, or (weighted) matchings in graphs.

3.3 Conceptual Design

We want to highlight some considerations of ours that led to this example.

The first modeling phase could be done using a copy of a map and a pen, but by using the computer, we coerce the students to create a *model* of streets instead of just another drawing (observe that the map is already a first model of reality, and the graph is a more abstract one).

We took care to integrate phases where students work on paper. This transition slows down the lesson deliberately and enables the students to re-think in a different

mode what they are doing.

By enabling the students to do their own checking, we invite them to be more confident in their own judgments, and strengthen their argumentation skills. The subliminal hints are used for proper internal differentiation, where students get as much help as they need to advance, but not too much to stop thinking.

The whole material is modularized and customizable. It is easy to split up into several parts and to adapt and adopt it for the individual curriculum.

The graph algorithm used here is *not* explicitly given. In this activity we do not want to teach the algorithm per se, but rather general competencies: modeling, problem solving, mathematical thinking, etc. Still, we make the algorithm available, and exhibit its step-by-step behavior, so that students can use the algorithmic structure for their own conjectures and structural observations.

4 Design of Interactive Teaching Material for Discrete Mathematics and Graph Algorithms

The example of Sec. 3 suggests that a computer is particularly suited for work with graph algorithms. Our classroom experience confirms this. However, this single example can only be a starting point for a comprehensive suite of electronic activities. As there is no fixed syllabus for graph algorithms or, more generally, discrete mathematics, any offering has to be flexible enough to support different styles of teaching and learning, and must adapt to the specific content chosen.

For our explorations, we decided to make algorithms and algorithmic thinking the central theme. All algorithms that are adequate for school use should be included
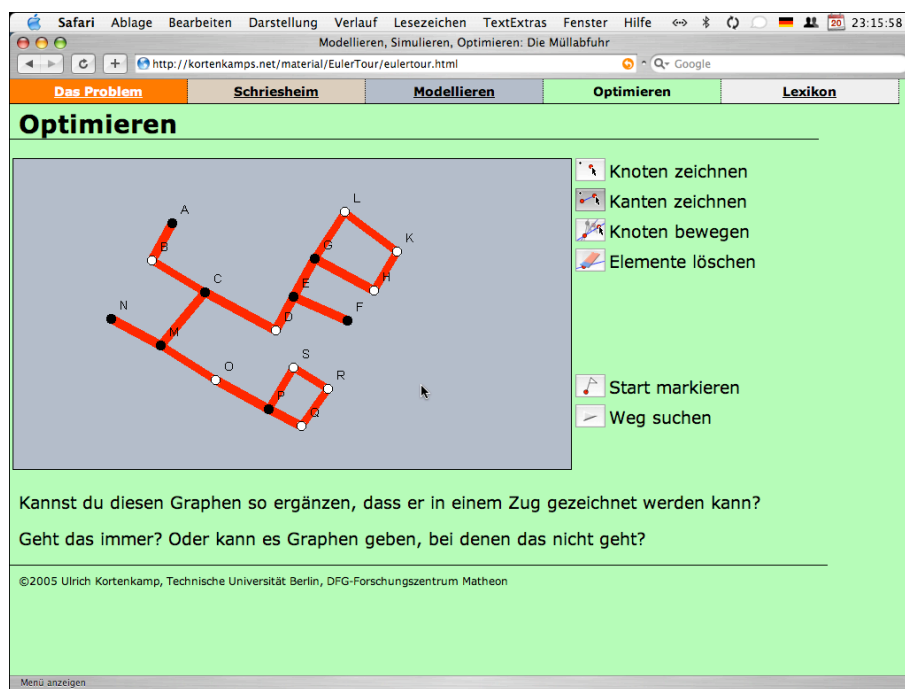
Fig. 3: The electronic work sheet for "optimization". The German instructions are: *Is it possible to extend this graph to make it possible to draw it without lifting the pen? Is this always possible (for any graph)? Or are there graphs that do not permit it?*

– at least depth-first and breadth-first search, shortest path algorithms like Dijkstra's algorithm or Bellman-Ford, minimum-spanning-tree algorithms, network flow algorithms, and (bipartite) matchings.

A quick inquiry on the Internet shows a wealth of implementations of demonstrations of these algorithms. But our hands-on approach requires that the algorithms can be *used by the students*; even more, students should be able to interact with the algorithm, influence how they run, change the input, get additional visual hints about the state of the algorithm, etc. For example, they should see the backtracking that occurs in a depth first search, move vertices to see that a graph removes geometric information, or add and remove edges to understand the different behavior of the algorithm.

We want to mention some prior packages that are meant to fill this gap. LEDA[9], written in C++, originally developed by Mehlhorn and Näher, provides extremely efficient and exact implementations of graph algorithms and many more. Since it has graduated from a university project to a commercial product, it is no longer affordable for school use, unfortunately. The software as such is very good, but as it is non-trivial to create LEDA-based software, we cannot expect anyone to create educational content with it.

Gato[10] is an open-source solution written in Python that is also used as a basis for CATbox by Schliep and Hochstättler (2002). Unfortunately, it is rather difficult to extend the package, and the visualizations suffer from the windowing toolkit that is used. Both LEDA and Gato have in common that it is not easily possible to create small web-based examples that can be combined into a self-contained activity.

A package that tried to solve this problem is EVEGA[11] by Holzapfel and Khuri (2001). This package is written in Java, so it could be used on the web, but this is not implemented. Also, it only offers three algorithms (BFS, DFS and MaxFlow), and thus it is not much better than a single-purpose visualization.

A sophisticated, and continuously developed, software is GraphBench[12] by Markus Brändle. It is written in Java and visualizes graph algorithms. However, it has some shortcomings. One thing that impedes understanding is that supplemental data structures such as stacks are not consistently visualized. For another thing, input and manipulation of graphs is unintuitive and error-prone; it is clear that this is not the main focus of the tool. The pre-installed algorithms are mostly too advanced for school use (e.g., Clique, Vertex Cover).

The tool comes with an embedded programming interface. However, the programming interface is too narrow, there are no data structures to use. Neither is it possible to have algorithm-specific annotations such as weight on edges or vertices. Also, the choice of Java as a language implies that the user must have a complete Java development kit installed.

By analyzing the existing tools we came to the conclusion that from a user's point of view it is important to support an intuitive, powerful, flexible and well-known user interface. We decided that it is a good start to extend existing geometry software that is used in mathematics education. This will also reduce the initial skepticism of

---

[9] http://www.algorithmic-solutions.com/enleda.htm
[10] http://gato.sf.net

[11] http://www14.in.tum.de/EVEGA
[12] http://www.inf.ethz.ch/personal/braendle/graphbench/

teachers and make it easier to introduce the software to the students. Even if we offer several modules, then each of them will be similar to the others and the other software used in the class, so the use of technology will not add too much overhead. As graphs are often embedded into the plane, they can be understood as geometric objects. It is a natural decision to take existing geometry software[13] and add algorithmic capabilities to it.

The resulting package consists of an authoring tool which can be used for creating web pages that contain electronic activities as seen in Sec. 3, and a collection of ready-made exercises, examples and demonstrations. Teachers can either use the material as-is, recombine it in order to adapt it to their teaching, or change it or create missing parts using the authoring tool. While we are aware of the fact that most teachers neither have time nor capabilities to create much material on their own, we consider it an important feature for the material to be *customizable*.

The algorithms are built-in (including all of the algorithms mentioned above), and currently it is not possible to write new ones (except for the authors of this article). We will discuss this in the next section. We offer visualizations of the graphs and their states, as well as visualizations of elementary data structures such as stacks and queues that are used by the algorithms.

The timing of the algorithm visualization is flexible. It is often useful to stop only at certain lines, e.g., the beginning of a loop. Students, who tend to be bored after seeing the same loop execution over and over again, appreciate that the speed adapts to their knowledge.

All of the examples are designed to connect to the real world. In (Faltin 2002) a student testing the algorithm visualization tool SALA asks for a realistic example of Heaps. Faltin acknowledges this as a valid request, which can be fulfilled with a Dijkstra algorithm demonstrating the use of a priority queue, but also turns it down as being too advanced for the purpose of SALA. With Visage, we try to address this. For the prototype package, for example, we included shortest-path algorithms, and we also took care to apply them to real-life problems like subway travel routes in Berlin.

5 Lessons learned so far

We already had a chance to evaluate a first prototype of our software and material in the classroom. Of course, this is by no means a serious empirical evaluation, but only a first test for being on the right track.

The tests using material about the shortest path problem were conducted in a computer science course (first year of secondary level) in a Gymnasium, the Berlin Kolleg.[14] This school in Berlin is an adult's education institute; the students already have some professional experience and are older than the usual students in that grade. All students were given a CD containing the Visage software

and HTML pages containing many examples.

Unfortunately, the course was optional for most of them, and many of the students appeared irregularly. Still, we did personal interviews with each of them, in order to gain insight about technical, didactical and other problems and benefits.

Here, we want to highlight some of the responses that will guide the further development of Visage.

Overall, those students that used the material gave positive feedback. At least subjectively, the additional possibilities that arise from using a computer facilitate the understanding of the concepts. Being able to quickly create and change a test example is one such motivational and useful ability.

For most of the students computers are recognized as a tool for gaming, a device for investigations using the Internet, and word processing only. In school they only expect the two latter activities. Asked for possible usages of the computer in school, they find these two applications, and they cannot think of anything beyond them. In other words: Computers are input/output devices for data. The computational (sic!) capabilities of computers do not seem to be of any importance in the view of the students. This is a clear evidence for the need of algorithmic training in mathematics.

Many students are not motivated sufficiently by the use of computers in a teaching unit alone. One reason may be the unexpected kind of use. Therefore, the teacher has to present the subject matter. In this case, the lessons were interesting and enjoyable for the students in our evaluation. Only after being asked to work with the computer in front of the class they concluded "Hey, this is fun!" In the interviews, many students indicated that the usage of computer and projector in the classroom made sense.

For this reason, the handout material was not developed to be completely self-explanatory. We did include short summarizing reminders to allow students to use the material at home. It should be noted, however, that the technical problems that made use on the school's computers problematic, are common on home computers as well.

Currently, students develop their algorithms on paper. The actual simulation running on the computer has nothing to do with their own handiwork. Therefore, they cannot see the success of their own algorithms on the computer. That is not satisfactory and was also marked as a deficit by one student. We intend to close this gap by embedding an interpreter into our tool that allows users to program their own graph algorithms easily. The existing interfaces of the underlying DGS to Jython can be used.

6 Future Directions

With the current state, we are close to a viable solution for using a computer for teaching graph algorithms and combinatorial optimization. All standard graph algorithms are available immediately. They can be tried out and their properties can be discovered interactively. Strengths of the computer are both its ability to speed up processes that otherwise require too much time, and its superior visualization in comparison to hand-drawn ex-

---

[13] We used Cinderella, see http://cinderella.de, as it is being developed in our group as well

[14] http://www.berlin-kolleg.de

amples.

Our next step is to disseminate the material and improve it based on the feedback we get. The major goal is to make the whole package easily usable by other teachers. It should become flexible and stable enough for widespread use.

It will be of special importance to encourage teachers to recombine the existing parts. This requires that they have both the technical and *mathematical* knowledge about the subject. At least in Germany, many teachers never got a proper education in graph theory, or they did not use it in teaching during the last years and forgot what they learned. We hope to change this situation by providing the means to use the computer sensibly. To ensure and sustain our approach we will offer teacher training on the mathematics, the software, and the educational concepts behind both.

References

Aigner, M.; Bänsch, C.; Grötschel, M.; Lutz-Westphal, B.; Unterreiter, A.; Ziegler, G. M. (2003): Lebendige Mathematik! Berliner Thesen zum Mathematikunterricht, *Mitteilungen der Deutschen Mathematiker-Vereinigung* (pp. 29-31)*.* Vol. 4-2003. Berlin: DMV

Bodendiek, R.; Bigalke, H.-G. (1985): Graphen in Forschung und Unterricht, In: Festschrift K. Wagner. - Hildesheim: Franzbecker

Braendle, M.; Nievergelt, J. (2004). Tackling Complexity: A Case Study on Educational Software. *World Conference on E-Learning in Corporations, Government, Healthcare, & Higher Education 2004* (pp. 1794-1799). Online: http://www.inf.ethz.ch/personal/braendle/graphbench.pdf

Bruder, R.; Weigand, H.-G. (Eds.) (2005): Mathematik lehren Vol. 129 "Diskrete Mathematik"

Freudenthal, H. (1973): Mathematik als pädagogische Aufgabe. - Stuttgart: Klett

Green, N. (1997): Unterrichtsvorschläge zur diskreten Mathematik. - In: Mathematik lehren Vol. 84

Gritzmann, P.; Brandenberg, R. (2002): Das Geheimnis des kürzesten Weges. Ein mathematisches Abenteuer. - Berlin/Heidelberg: Springer

Hußmann, S.; Lutz-Westphal, B. (2005): Kombinatorische Optimierung. Für Studium und Unterricht. - Wiesbaden: Vieweg

Kenny, M. J.; Hirsch, C. R. (1991): Discrete Mathematics across the Curriculum, K-12. - Virginia: National Council of Teachers of Mathematics (1991 Yearbook)

Kletzl, H. (2002): Daten- und Beziehungsstrukturen. Eine didaktische Analyse im Spannungsfeld von angewandter Informatik und angewandter Mathematik. - Universität Salzburg: Doctoral Dissertation

Khuri, S.; Holzapfel, K. (2001): EVEGA: An Educational Visualization Environment for Graph Algorithms. *Proceedings of the 6th annual conference on Innovation and technology in computer science education.* Canterbury: ACM

Kortenkamp, U.; Richter-Gebert, J. (1998), Geometry and Education in the Internet Age, *Proceedings of the ED-MEDIA & ED-TELECOM 1998 World Conference on Educational Multimedia, Hypermedia and Telecommunications,* Freiburg: AACE. Online: http://www.cinderella.de/papers/geo-i.pdf.gz

Kortenkamp, U. (2001). The Future of Mathematical Software, *Proceedings of MTCM 2000.* Heidelberg: Springer-Verlag.

Kortenkamp, U.; Materlik, D. (2005). Visage - A software package for the Visualization of Algorithms with Geometry Software. Online: http://www.cinderella.de/visage

Kortenkamp, U (2005). Guidelines for Using Computers Creatively in Mahtematics Education. *In: Proceedings of the first KAIST workshop on enhancing university mathematics teaching.* Daejon, Korea: KAIST.

Kultusministerkonferenz (Ed.) (2003): Bildungsstandards im Fach Mathematik für den Mittleren Schulabschluss. Darmstadt: Luchterhand.

Lutz-Westphal, B. (2004): Erlebnis Mathematik - Kombinatorische Optimierung im Unterricht, *Mitteilungen der Deutschen Mathematiker-Vereinigung* (pp. 78-81)*,* Vol. 2-2004. Berlin: DMV.

Lutz-Westphal, B. (2004): Lebendiger Mathematikunterricht mit kombinatorischer Optimierung. *Beiträge zum Mathematikunterricht 2004* (pp. 353-356). Hildesheim: Franzbecker

Nitzsche, M. (2004): Graphen für Einsteiger. Wiesbaden: Vieweg

Ore, O. (1974): Graphen und ihre Anwendungen. - Stuttgart: Klett

Reichel, H.-C.; Kubelik, T. (2002): Mathematik – unsichtbar, doch allgegenwärtig. - In: Arbeitskreis "Mathematik und Bildung" der GDM: Köhler, H.; Röttel, K. (Eds.): Bildungsraum Schule , Vol. 7. - Eichstätt: Polygon-Verlag, p. 99-128

Renz, W.; Euba, W.; Kaiser, G.; Löding, W.; Weitendorf, J. (2004): Rahmenplan Mathematik, gymnasiale Oberstufe. - Hamburg: Behörde für Bildung und Sport. Online: http://www.bildungsplaene.bbs.hamburg.de

Richter-Gebert, J.; Kortenkamp, U (1999). The Interactive Geometry Software Cinderella, Heidelberg: Springer-Verlag.

Rosenstein, J. G.; Franzblau, D. S.; Roberts, F. S. (1997): Discrete Mathematics in the schools. - American Mathematical Society, Series in Discrete Mathematics and Theoretical Computer Science, Vol. 36

Schliep, A.; Hochstättler, W. (2002): Developing Gato and CATBox with Python: Teaching graph algorithms through visualization and experimentation. In: Proceedings of MTCM 2000, pp. 291-310. Heidelberg: Springer Verlag

Schrijver, A. (2001?): Kleuren en routeren in grafen. Notes for high school teachers masterclass. - http://www.cwi.nl/~lex/

Schuster, A. (2004): Kombinatorische Optimierung als Gegenstand der Gymnasialdidaktik im Umfeld von Mathematik- und Informatikunterricht. - Universität Würzburg, Habilitation Thesis

Wippermann, H. (1976): Graphen im Unterricht.

───────────

Authors

Geschke, Anne, Mathematisches Institut, Fachgebiet Mathematikdidaktik, Technische Universität Berlin, Sekr. MA 7-3, Straße des 17. Juni 136, 10623 Berlin
Email: geschke@math.tu-berlin.de

Kortenkamp, Ulrich, Prof. Dr., Mathematisches Institut, Fachgebiet Mathematikdidaktik, Technische Universität Berlin, Sekr. MA 7-3, Straße des 17. Juni 136, 10623 Berlin
Email: kortenkamp@math.tu-berlin.de

Lutz-Westphal, Brigitte, Mathematisches Institut, Fachgebiet Mathematikdidaktik, Technische Universität Berlin, Sekr. MA 7-3, Straße des 17. Juni 136, 10623 Berlin
Email: westphal@math.tu-berlin.de

Materlik, Dirk, Mathematisches Institut, Fachgebiet Mathematikdidaktik, Technische Universität Berlin, Sekr. MA 7-3, Straße des 17. Juni 136, 10623 Berlin
Email: materlik@math.tu-berlin.de