A Technology-based Approach to Discrete Mathematics in the Classroom

Ulrich Kortenkamp, University of Education Schwäbisch Gmünd, Germany

In this article we describe a new approach to teaching discrete mathematics, in particular graph theory and graph algorithms, using interactive geometry software. Students are encouraged to use the computer to explore and verify their theories using a computer, and use it as a tool for modeling real-life situations.

Introduction

Discrete Mathematics, though being an important part of modern mathematics, is not part of the standard curriculum in Germany. There have been several attempts to integrate these topics into regular teaching, most of them dating back to the 1970s, but their success was limited (see Lutz-Westphal 2006 for a summary). Today, with a stronger focus on competencies like modeling and mathematical reasoning, there is a new movement of curriculum reform. Learning from the failures of earlier attempts and basing the teaching content on new and more praxis oriented paradigms, several groups show ways to teach applied discrete mathematics.
The DFG research center MATHEON in Berlin (http://www.matheon.de) hosts a project named Visage – *Visualization of Algorithms using Geometry Software*. The project is the origin of several web-based learning activities that have also been tested in the classroom. In this article we describe and discuss one of these units exemplarily. It has been used in the classroom, where it proved that also low-achieving 7th-grade students are able to work mathematically, even if they have not much experience in classical mathematical topics. It gave them confidence, and they also had the experience that finding an optimal solution is not based on being lucky, but on mathematics.
Although we restrict our report on explorations with Eulerian graphs, the Visage package is not restricted to these. The authoring tool for Visage based applets includes several standard graph algorithms, like BFS (breadth first search) and DFS (depth first search), bipartite matchings, algorithms for shortest paths and minimum spanning trees, and others. New algorithms can be added using either Java or the CindyScript scripting language.

On The Road With The Garbage Truck

Starting point of the exploration is the question for the best route a garbage truck should take. Usually (at least in Germany), the garbage collectors are able to collect the bins from both sides of the street. In our lesson, we made sure that all students agree on this fact by first showing a short movie showing the collection process – this can be replaced by a field trip if there is enough time. Considering this experience it became clear that the truck has to take each road at least once, and that going through a road once is sufficient to collect all bins. An optimal solution would be to use each road exactly once, in which case no excess tours are made.
The goal of the teaching unit was that the students can model the essential aspects of such a situation, and in particular that they can develop the simple abstraction of a *graph*[1] from it. Students should be able to turn a vague description ("the truck should go once through every road") into a model that enables them to argue on an abstract level and come up with solutions within that model. We also decided that appropriate software should assist the students.

[1] In the activity and this paper all graphs are loopless, which was also enforced by the software.

Turning a Problem Into Something Calculable

The first step in a modeling process is to remove as many details as possible. The students already know city maps (as a first abstraction from reality) and should be able to work with them. Still, these do contain too much unstructured information. Therefore, the first (electronic) activity was used to get from a pictorial representation to a mathematical one. On the computer, students could trace streets on a city map of their school´s neighborhood. Doing that, and without using the terminology at first, they drew vertices and edges of a graph describing the structure of their city. While this could have been done on a printed map as well, there are reasons to use a computer here. These originate both in restrictions *and* new possibilities of the computers!

Let us describe the restrictions first, as the fact that the computer is used in such a manner might be more interesting: It is not possible to draw anything with the software, but it is only possible to draw straight edges connecting a start- and endpoint. By this, only a structural representation is possible that is substantially different from city maps, as these can be designed freely. The computer, in this case the special software,[2] enforces the modeling simplification.

It should not be forgotten to discuss this step with the students to make them aware of their actions: They removed information from the city map – is their model still valid? What if a street was curved before? May streets be bend, shortened or prolonged? Which actions are not allowed?

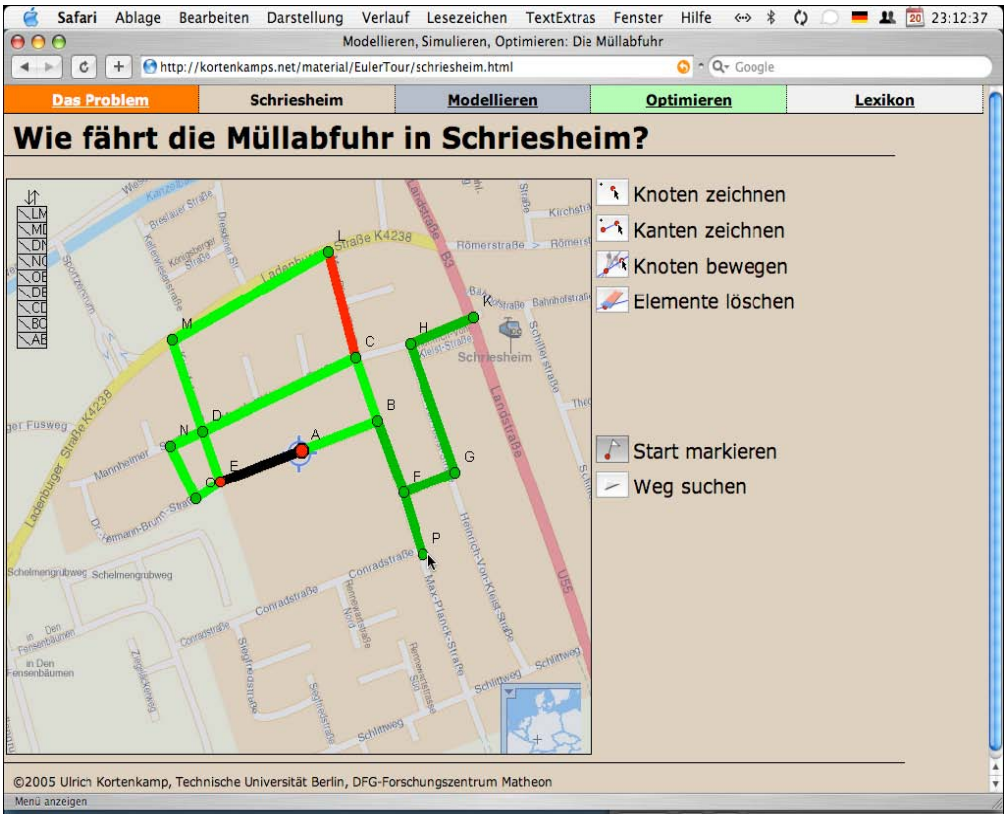

Fig. 1: Modeling the streets using a graph.

The additional possibilities, the added value of the computer, is the immediate usability of the model. With the press of a button the software can test immediately – using a built-in algorithm – whether the graph can be drawn in a single stroke or not. As an additional requirement, the start and end of such a circuit should be the same vertex – otherwise the truck will not return to the depot.

---

[2] The Visage extension of the geometry software Cinderella.2 was used. More information about Visage is available at http://cinderella.de/visage. The activities mentioned in this article are available from http://kortenkamps.net/material/EulerTour.

At this point we change the initially very complex question into an easier one. It is not necessary to do this explicitly, though. Instead of asking for the best route for a certain city, we are now looking for imaginary cities for which we know the best solution. More exactly: We are looking for graphs that are Eulerian, or, as characterized above, can be drawn in a single stroke.
Here the students are able to experience that the graphs they draw offer a better model than the maps they are used to work with for several years. As discussed above, a lot of information that is available in a map is no longer necessary. In particular, the lengths and shapes of the streets as well as the exact location of vertices are completely irrelevant for finding a solution. When we tested the activity, the students came to the same conclusion *by themselves* and started to accept that a proper model does indeed reduce their work. This made it possible to continue work on a purely mathematical question that no longer needed an application.

Back to Pen and Paper!

The time has come to leave the computer for a moment. On the one hand this introduces a new phase in the lesson, on the other hand we offer another way to tighten the just found abstraction. Also, we want to emphasize that the abstraction is still useful.
One way to do this is to use prepared graphs (hand-drawn on paper slips). In the classroom we hand out several graphs, with equal (i.e. isomorphic) ones on paper of the same color. The students were asked to find out which graphs belong to “good” cities, i.e. cities that allow a single-stroke-tour. Checking of the solutions was done with the computer, again, but the computer was not used while the students were looking for their solutions.
For “good” graphs, checking the answer with the computer is not really necessary, as the students are able to prove their answer by showing others the tour they found, which they did. As the ad-hoc algorithm for finding such a tour usually succeeds, it is only necessary to use the computer to verify that a graph is indeed “bad.” For these, the software serves as a kind of referee. If a tour is possible, but wasn‘t found by the students, it is demonstrated. In the other case the algorithm stops and shows an offending edge in red.
The example graphs should be chosen carefully to contain enough bad examples, because the most important part of the learning sequence starts here. Until now it seemed as if finding an Eulerian tour is a matter of “just seeing it,” and the lower achieving students are easily discouraged if they are not as fast as the others are. The decision whether there is a tour or not seems to be dependent on the fact whether a student is “somehow good at mathematics” or must be left to the computer. Here is a *chance* to replace nebulous helplessness by competence and to strengthen the confidence of the students in their own abilities. We would like to see this also as a contribution to teaching the sensible use of computers: If we understand how a computer comes to a decision, then we can question these decisions better.

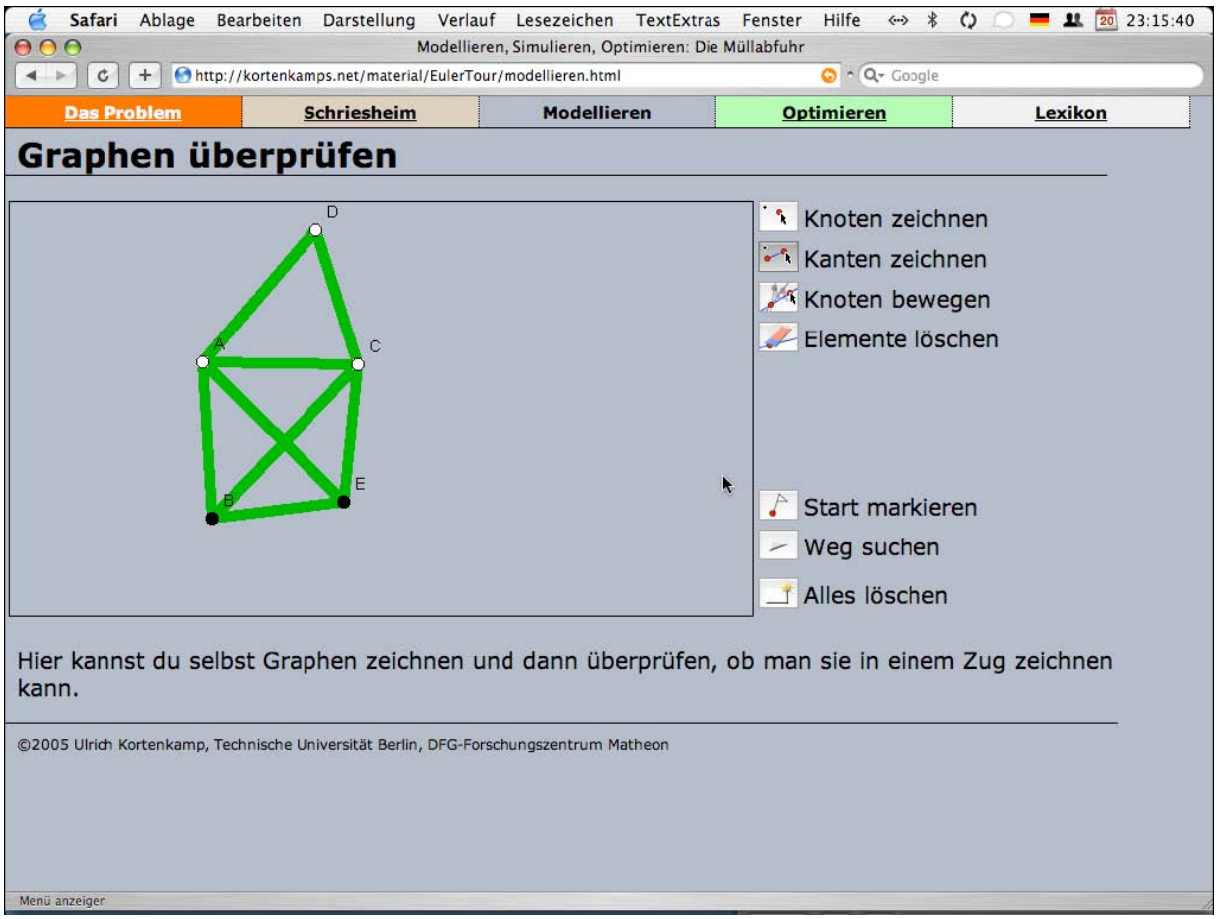

Fig. 2: Checking a solution

The Good and the Bad

Within the activity, we are using a “trick” that might not be necessary with higher achieving students, but lowers the difficulties of abstraction significantly in our learning group.
Using the built-in scripting language CindyScript of the software we color vertices with an even degree (i.e. having an even number of edges connected to them) white, and those with odd degree black. Instead of talking about the degree of a vertex the students can talk about black and white vertices. It is immediately visible whether a graph contains only vertices of even degree. We can hope that students recognize that all graphs they already identified as being good have white vertices only. From there it is only a small step to formulate a criterion for Eulerian graphs.
In the classroom the students were able not only to discuss this question on a mathematical level, but they also arrived easily at the crucial observation that it is necessary to leave a vertex exactly as many times as to enter it. So they found a mathematical description of the necessary criterion of even degrees for all vertices, and they could connect the abstract notion with an actual conception.

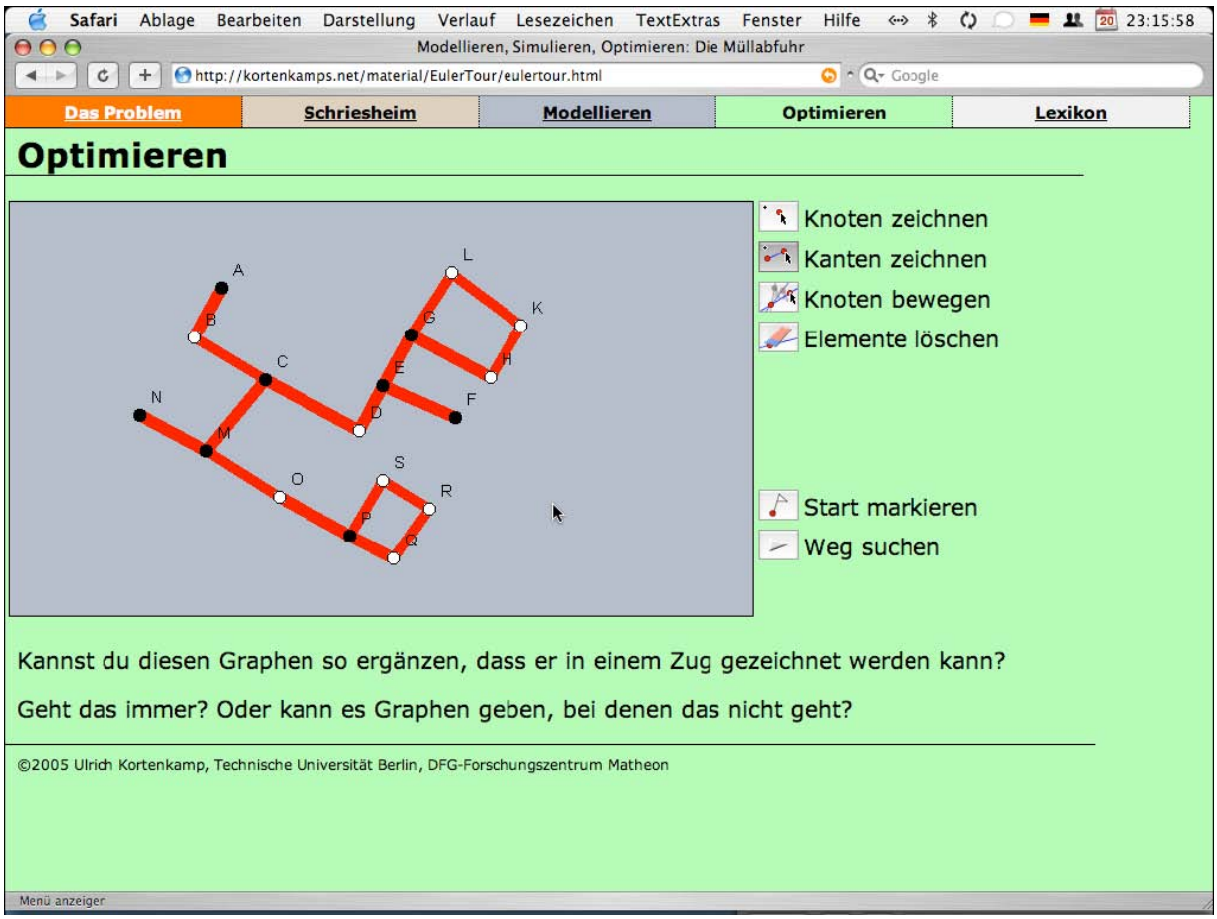

Fig. 3: Adding edges to a graph such that it contains only white vertices.

Even more: We lay foundation for a proof using induction that a graph always contains an even number of vertices of odd degree (black vertices). This is because the students experience that the color of a vertex only changes when a new edge is added to it, and then the color alternates between white and black. New vertices without edges are always white (degree 0, which is even). Thus a new edge can only increase the number of black vertices by two (connect two white vertices), decrease it by two (connect two black vertices), or leave it unchanged (connect a white and a black vertex). This stepwise approach, which leads to the inductive proof if the number of edges is not fixed in advance, is again enforced by the computer.

This means that we can "risk" a very open question, like the one in the electronic activity in Fig. 3: *Gibt es Graphen, die man nicht so ergänzen kann, dass alle Knoten weiß werden? – Are there graphs where we cannot change all vertices to white by adding edges?*

This last, very ambitious step can be prepared for very easily. In work in pairs students challenge each other: The goal is to give the partner a graph that he or she cannot complete to an Eulerian graph (all vertices have even degree). Using the black/white hints of the computer there is no place for futile discussions about the winner. As the completing party will always win we raise a natural desire for finding the reason – or proof! – of this apparent asymmetry.

The use of the computer initiates the necessity of proof and exact mathematical thinking, instead of making it unnecessary as mere augmentation of empirical evidence as is often feared.

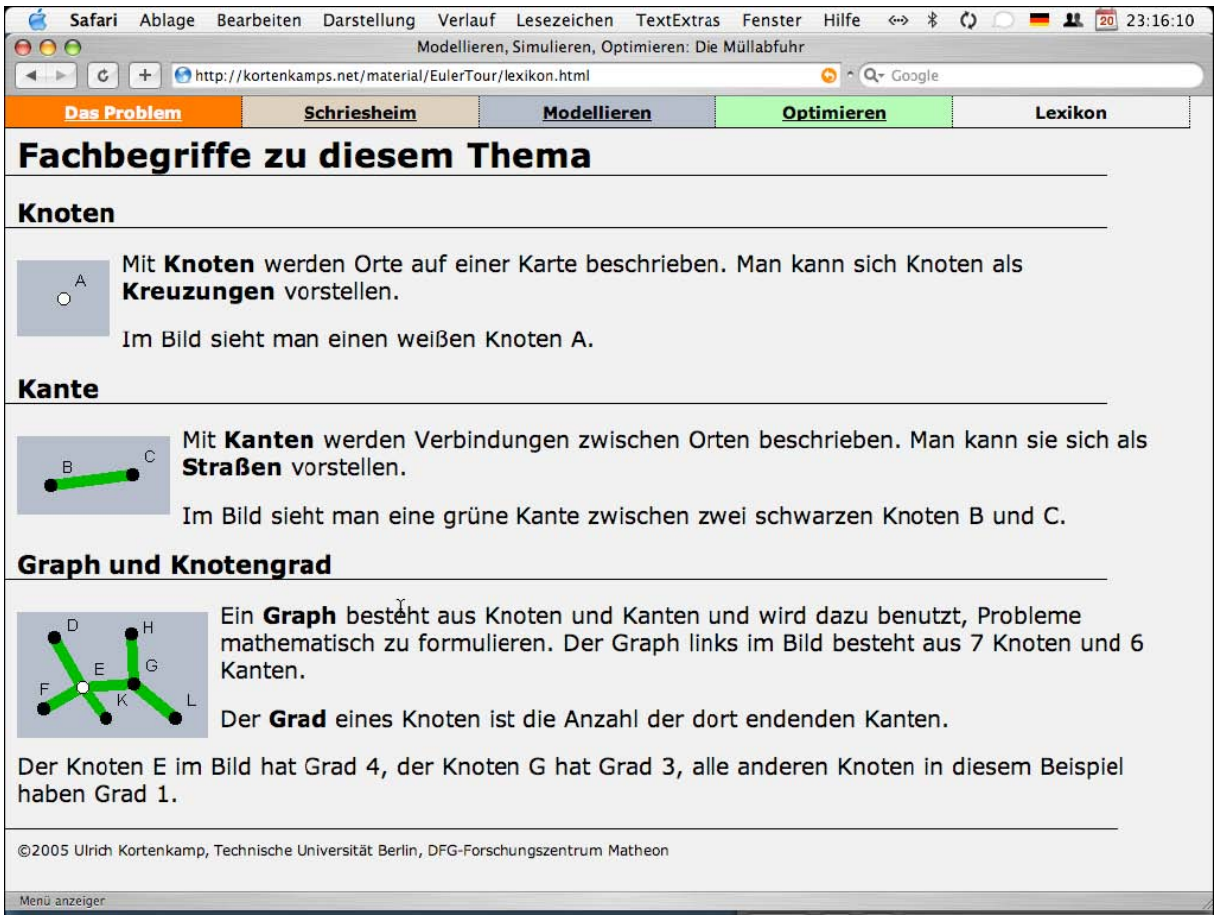

Fig. 4: The included dictionary of graph terms and definitions

Conclusion and Further Questions

It is not necessary to rely on the computer for teaching Eulerian tours or other graph algorithms in the classroom. Nevertheless, we could demonstrate a few immediate pedagogical opportunities which we review again below.

I. Forced modeling with graphs
   Using a special software for graph modeling we constrain deliberately the freedom that students have when using pen and paper or general drawing software. Students are forced to think about the difference between “two points connected by a line” and the combinatorial structure “two vertices connected by an edge.”

II. Immediate trial and error with own models
   The abstract model is transferred into the direct and hands-on experience of the students. Modeling is not an end in itself, but connected to real-world application. The self-directed work with graphs and the option to start the algorithms on their own enables the students to express their conjectures and verify or falsify them.

III. Automated checking of solutions
   The freedom a computer gives can be difficult to handle in the classroom. The teacher has to be able to judge and value many different solutions in short time. Using the computer, part of this task can delegated to it. However, it should be made clear that the computer is not an almighty tool of control. Instead, students should learn the inner mechanisms and acquire the necessary competence to use computers and software in a sensible way.

IV. Additional visual hints
   Many solutions are immediate and easy to understand if you can “see” them. This “seeing” can be advanced by good visualizations of complex or abstract criteria. In the example above the

even/odd degree of a vertex is visualized as black/white vertex. Similar examples would be the coloring of prime numbers or equivalent terms in computer algebra software, or the automatic highlighting of equally sized angles in a geometry software.

All these effects can be achieved in different organizational forms of teaching, depending on the available equipment and other premises.

In our example above we could finish the sequence, but on the other hand there are many loose ends that could be used for continuing questions and more explorations in discrete mathematics: What shall we do if the original city map does not admit a Eulerian tour? How can we find an optimal tour? What does "optimal" mean? How can we (or the computer) find this tour – can we just start or do we have to do some planning before?

The answers to these questions are important, but we deem it even more important that questions like these are raised. A central issue of mathematical modeling in school is the recurring circle of modeling and analysis. In reality, this rarely stops, every mathematical model has to show that is remains valuable all the time.

Of course, this means that starting from easy problems we will always end up with more complex ones, until we cannot solve them any more. This is not to be seen as a predicament, but as a chance. Topics like Eulerian graphs that can serve as cross-section themes from early education up to college are the key to seize these chances.

References

Bruder, Regina & Hans-Georg Weigand (2005): Problemlösen, Verstehen, Anwenden ... aber bitte diskret. mathematik lehren, 129, 4–8

Geschke, Anne, Ulrich Kortenkamp, Brigitte Lutz-Westphal & Dirk Materlik (2005): Visage – Visualization of Algorithms in Discrete Mathematics. Zentralblatt für Didaktik der Mathematik, 37(5), 395–401

Kortenkamp, Ulrich (2005): Visage – Visualisierung von Graphenalgorithmen. In: Beiträge zum Mathematikunterricht. Vorträge auf der 39. Tagung für Didaktik der Mathematik, Gesellschaft für Didaktik der Mathematik, Bielefeld: Franzbecker

Lambert, Anselm & Pia Selzer (2007): Schillernde Diskretisierung – eine Schnittstelle von Mathematik und Informatik. In: Kortenkamp, Ulrich, Hans-Georg Weigand & Thomas Weth (Hg.): Bericht über die 23. Arbeitstagung des Arbeitskreises „Mathematikunterricht und Informatik", Franzbecker

Lutz-Westphal, Brigitte (2006): Kombinatorische Optimierung – Inhalte und Methoden für einen authentischen Mathematikunterricht. Dissertation, Technische Universität Berlin, Berlin

Richter-Gebert, Jürgen & Ulrich H. Kortenkamp (1999): The Interactive Geometry Software Cinderella. Heidelberg: Springer-Verlag, http://cinderella.de