Enhancing Understanding and Constructing Knowledge in Graph Theory and Combinatorial Optimization within the 'Graphs' multimedia application

Eva Milková
E-mail: eva.milkova@uhk.cz
University of Hradec Králové, Faculty of Informatics and Management
Rokitanského 62, 500 03 Hradec Králové, Czech Republic

Abstract: *Multimedia applications have substantially influenced education. They give teachers an excellent chance to demonstrate and visualize the subject matter more clearly and comprehensibly, as well as also enabling them to prepare study material for students which optimizes their study habits. The Theory of Graphs together with Combinatorial Optimization is a wonderful, practical discipline. On the one hand, there are many methods which can be used to solve the same problem, while on the other hand, using effective modifications of one algorithm, we can devise methods for solving various other tasks. To educate students in this area it is important to familiarize them with certain algorithms in context, in order to get deeper into each problem and understand it entirely. In this paper we present just a few ideas that have proved successful in teaching and learning this part of mathematics using the program 'Graphs', whose main purpose is to visually represent basic graph-concepts and graph-algorithms using a coloring process on graphs created within the program.*

1 Introduction

Information technology has substantially influenced education. Students find modern technology very handy when looking up things of their own interest. Teachers should take advantage of this fact and try to prepare multimedia study material which optimizes students' study habits. This involves using applications which make students' study more effective, time-efficient and subject matter more comprehensible.

Along with large software products dealing with a wide spectrum of objects developed by a team of professionals, there are also various programs dealing with objects appropriate to course subject matter created by the teacher with regard to students needs.

This paper illustrates advantages of the program 'Graphs' [1] used as an important support for teaching and learning the subject Graph Theory and Combinatorial Optimization taught at the Faculty of Informatics and Management, Hradec Králové, Czech Republic.

2 Program 'Graphs'

The program 'Graphs' was created in the Delphi 5 environment by one of our students for his thesis on a script given by the author of this paper. The main purpose of this application is visual representation of basic graph-concepts and graph-algorithms using a coloring process on graphs created within the program.

The program is divided into three parts (see Figure 1): Graph creation and modification (*Edit graph*), Presentation of graph characteristics (*Graph characteristics*) and Application of several fundamental graph algorithms (*Algorithms*).

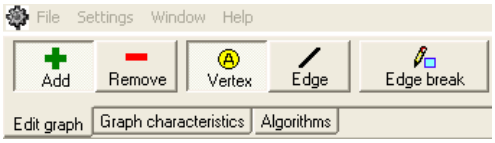

Figure 1 Three parts of the program 'Graphs' and *Edit graph* menu

The program enables the user to create, edit, and work on a graph, (moving, coloring vertices, edges, etc.), save the graph in the program, or save the graph in bmp format. It also makes it possible to display some graph properties of the given graph, to add color to vertices and edges, and to change positions of vertices and edges by "drop and draw a vertex (an edge respectively)". It allows the user to open more than one window so that more objects or algorithms can be compared at once.

2.1 Short manual to the program 'Graphs'

The link to the demo version of the program is http://edu.uhk.cz/~milkoev1/Graphs_demo, where the graphs used in paper are also available.

At the beginning, the main window of the program is empty and there is the following main menu (see Figure 2) available; *File*, *Settings*, *Window*, *Help*.

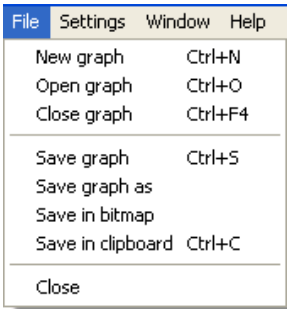

Figure 2 *File* menu of the program 'Graphs'

The *File* menu (see Figure 2) enables the creation of a new graph, opening a saved graph and saving a graph in two different formats. The program automatically saves graph into a native file format with the filename extension grf. This format enables future graph modification. However the graph can be also saved as a picture in bitmap format for use in other applications.

The *Settings* menu (see Figure 3) enables to set program, graph and algorithm settings.

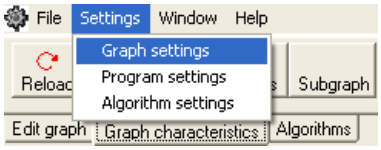

Figure 3 *Settings* menu of the program 'Graphs'

The *Window* menu (see Figure 23 thereinafter) enables opening of more windows at once.

2.1.1 Creation and modification of a graph

By clicking on *File* menu (see Figure 2) and the option "New graph" and filling the required information in the window (see Figure 4) a new graph with the name determined in the window will start.

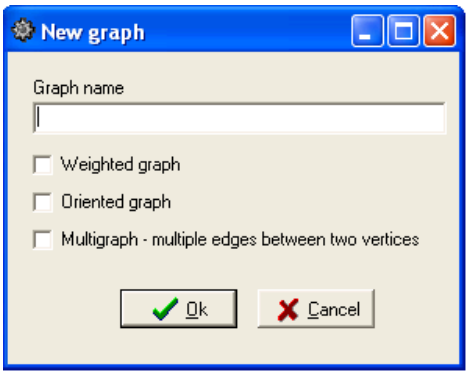

Figure 4 "New graph" option

The *Edit graph* menu (see Figure 5) serves for creation and modification of a graph. Vertices and edges can be added and/or removed using buttons shown in Figure 3.

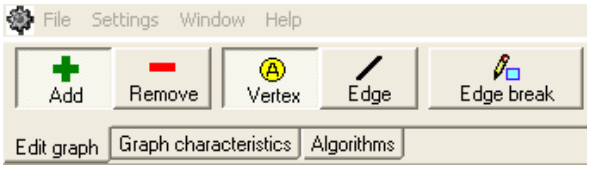

Figure 5 *Edit graph* menu

Vertices and edges of a graph can be named, renamed, resized etc. by right-mouse-button clicking on a vertex or edge and using the option "Vertex settings" or "Edge settings" (see the figures 6 – 8 as an example of editing the vertex "a" by right-mouse-button clicking on the vertex "a").

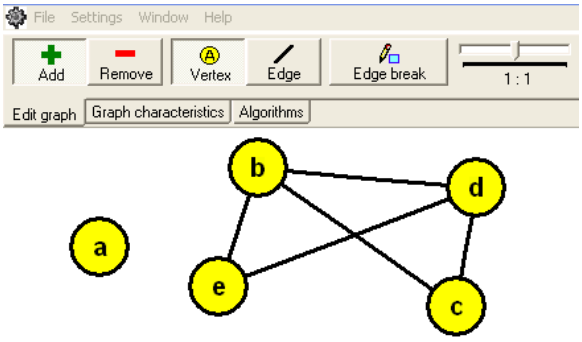

Figure 6 Editing vertex "a" in the given graph

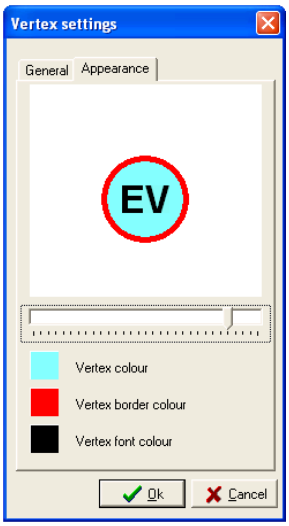

Figure 7 "Vertex settings" option

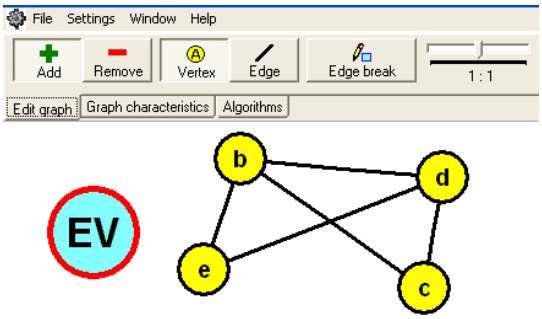

Figure 8 Result of editing the vertex "a"

The size of all vertices and edges can be changed using the *Graph Settings* menu (see Figure 3 and Figure 9).

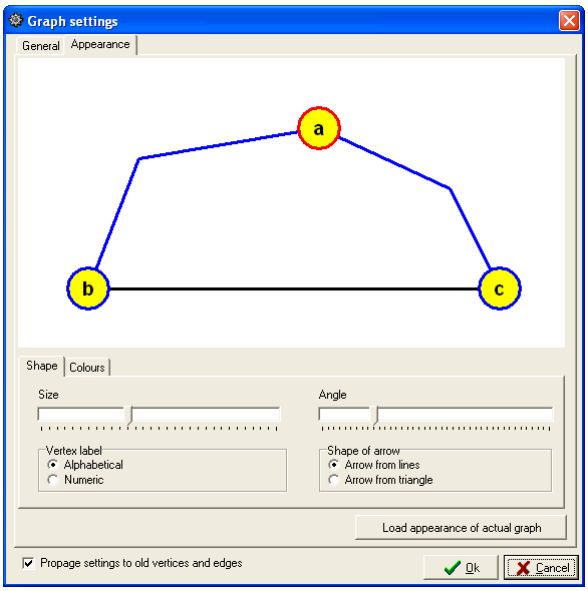

Figure 9 *Graph settings* menu

The option "Edge break" in the *Edit graph* menu serves for creation of a multigraph. It enables more edges incident with the same pair of vertices to be shown.

2.1.2 Presentation of graph characteristics

The *Graph characteristics* menu (see Figure 10) enables the display of a subgraph and the complement of the given graph.

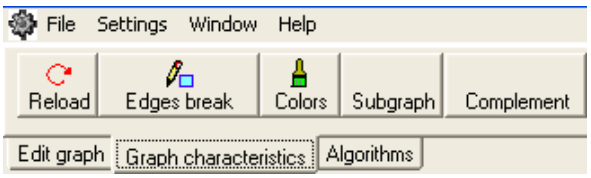

Figure 10 *Graph characteristics* menu

The main advantages of this option include the ability to color vertices and edges (the "Colors" option must be on) and to change positions of vertices and edges (the "Colors" option must be off) by "drop and draw a vertex (moreover, in the case moving an edge the option "Edge break" must be on)".

The possibility to write a text next vertices (see Figure 23) is also very useful. To do this the mouse must point at the vertex and the option "Colors" must be on.

Any vertex or edge can be colored simply with four predefined colors by mouse clicking (left, right, Ctrl+left, Ctrl+right) on vertex or edge. Predefined colors can be set in the *Settings* menu (see Figure 3). Clicking on the "Colors" option again turns this mode off and returns a graph to its previous color-state.

2.1.3 Application of several fundamental graph algorithms

The *Algorithms* menu (see Figure 11) enables a presentation of predefined graph algorithms.

The option "Algorithm choice" opens a dialog window where the appropriate algorithm can be chosen. The toolbar with the buttons Start, Pause and Stop manages the running of algorithms in speed that can be selected as well (see Figure 10).

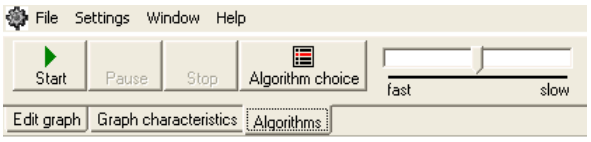

Figure 11 *Algorithms* menu

Several algorithms in different windows can be started at a time. But only the algorithm in the active window runs. Others are automatically paused.

2.2 Main benefit of the program 'Graphs'

The main benefits of the 'Graphs' program can be characterized as follows:

It enables the teacher to *complete his/her explanation within lectures* in such a way that the topic is more comprehensible:

- The option of opening more than one window enables the teacher to **explain the problem from more points of view and show mutual relations among used graph-concepts and algorithms** on graphs.
- The possibility of using colors allows the teacher to **emphasize needed objects and relations**.
- The possibility of saving graphs in bmp format allows the teacher **easy insertion of needed graphs** (e.g. graphs used during the lecture) **into the study material** (Remark: All figures used in this paper are created within the program 'Graphs'.).

It enables students to *revise subject-matter* and more deeply understand it.

- Students can use not only graphs prepared by the teacher but also graphs created by themselves and **explore the properties of** these **graphs**.
- The ability to open more than one window enables students to **follow mutual relations among concepts and algorithms**.
- The ability to save graphs in bmp format allows students **easy creation of needed graphs** for their tasks (texts and/or presentations) where they describe various practical situations with the aid of graphs and solve the given problem.

3 Teaching Principles

The aim of the subject Graph Theory and Combinatorial Optimization is to develop and deepen students' capacity for logical thinking. Students gain a basic level of competence in graph theory and graph algorithms. Well-prepared students should be able to describe various practical situations with the aid of graphs, solve the given problem expressed by the graph, and translate the solution back into the initial situation.

Thus when teaching Graph Theory and Combinatorial Optimization we always lay stress on applying the following principles so that our students, at least most of them, have entirely understood the subject matter.

Amusing motivation to the topic
As motivation in Graph Theory, we often use **puzzles**. We let students solve them and after explanation of the appropriate subject matter we discuss **an efficient solution using graph theory concepts**.

Teaching in contexts
When we deal with **a particular problem** we try **to examine** it **from more than one point of view** if possible and **discuss various approaches to its solution**. Usually, there are more methods which can be used for solving the same graph-problem. By comparing the various attitudes, students are able to explore the problem more thoroughly and understand it. Moreover, using effective modifications of one algorithm, we can devise methods of solving various other tasks. In this way students are encouraged to think about each problem more than usual.

In order to enrich students' understanding of the subject matter and deepen their awareness of it, we try to **describe a particular problem with real life examples**. We also ask students to give their own examples describing the topic to be sure that they can understand it.

Demonstration and visualization of the particular issue as completely as possible
"Students need images and visualization in addition to words. Science learning is about creating images in mind, and teaching should support such image formation."[2]

In the subject Graph Theory and Combinatorial Optimization there is no problem in illustrating the needed concepts using graphs. However, it is very important **to prepare suitable illustrative graphs** and **use colors to emphasize characteristics of the explained concepts**. Colors also play a very important role when explaining graph-algorithms. We **describe them as an edge-coloring or vertex-coloring process**.

Intensification of students' self-preparation
We are very well aware that interesting **resources prepared for self-study enable students more consistent engagement with the subject**. Students who are familiar with the materials are good partners and lessons can be run more efficiently, like a discussion or consultation.

The ability to visualize graph concepts and algorithms, and support preparation of other useful study materials were the main reasons why the 'Graphs' application was created.

The following two case studies demonstrate our teaching methods used in class, keeping the above mentioned principles with the support of the 'Graphs' program. The first case study illustrates a simple puzzle used as a motivation to the concept graph and vertex degree. The second study is focused on the importance of teaching in contexts.

3.1 Case study 1

***Competition*.** We use this puzzle as a motivation to the concepts graph and vertex degree.
An international company opened five positions for translators from the following languages: Russian, German, English, French and Spanish. Five candidates apply for the job.

- Mr. Smith can speak all 5 languages;
- Mr. Parker can speak English, French and Russian;
- Mr. Thomas can speak German and Russian;
- Mr. Brian can speak English and German;
- Mr. White can speak Russian and German.

Is the company able to provide all opened positions so that each of candidates would translate just from one language? If it is possible, propose the solution.

Solution: The situation can be easily represented by the following graph.

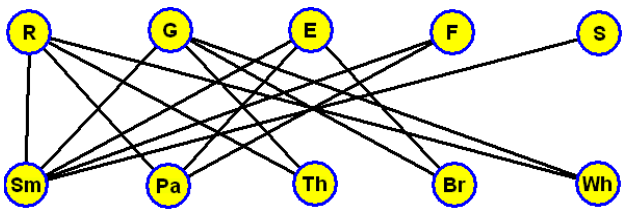

Figure 12 Graph-representation of the puzzle *Competition*

Looking at the graph on Figure 12, it is obvious that the vertex *S* with the degree 1 must be connected with the vertex *Sm*. Thus, at first, we color the edge {*S*, *Sm*} blue and the other edges incident with the vertex *Sm* red to demonstrate that we have rejected them. Then, considering the graph with uncolored edges, it is obvious that the edge {*F*, *Pa*} must be denoted by the color blue and hence the other edges incident with the vertex *Pa* by the color red, etc. Finally, the blue edges represent two possible solutions (see the following figures).

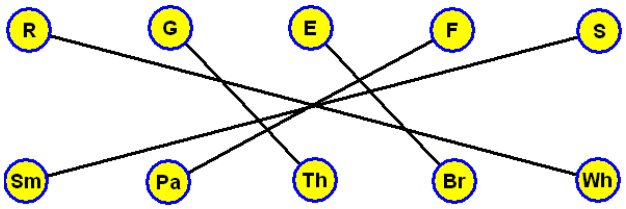

Figure 13 The first solution of the puzzle *Competition*

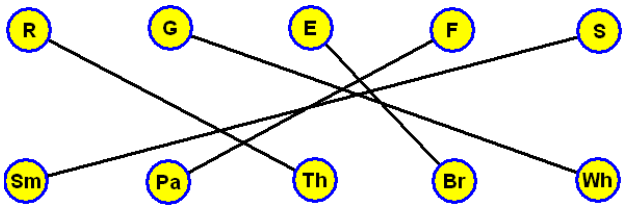

Figure 14 The second solution of the puzzle *Competition*

We introduce students to the simple puzzle *Competition* during the first lecture when explaining basic graph concepts. We let students work on the puzzle for 5 minutes. Most of them describe the situation as follows

- Smith ... R, G, E, F, S
- Parker ... E, F, R
- Thomas ... G, R
- Brian ... E, G
- White ... R, G

and "guess" the puzzle starting with one of the men speaking two languages. After 5 minutes they are mostly able to find out only part of the solution and are surprised **how easy it is to solve the given puzzle using graph and colors to visualize needed relations. We illustrate this process using the program 'Graphs'**.

A much more interesting and difficult problem is the following puzzle *Towns* that we discuss with students when explaining the concept isomorphism. Let us introduce it here.

***Towns*.** We use this puzzle to introduce and explain the concepts subgraph and isomorphism. Try to place the names of towns mentioned below into the frames of the given map (see Figure 15) so that no town shares any letter in its name with any towns in adjacent frames (neither horizontal nor vertical).

Atlanta	London	Paris	Lima
Berlin	Metz	Quito	Oslo
Caracas	Nairobi	Riga	Tokyo
Dallas	New York	Rome	

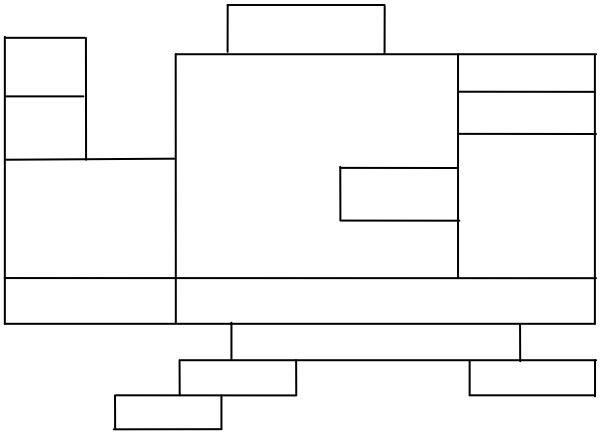

Figure 15 Map of the puzzle *Towns*

Solution. We advise students to describe both the map and also the relation between two towns that do not contain a same letter in their names, using graphs and find out **the isomorphism between the graph representing the map and the subgraph of the graph representing the relation**.

The given map can be represented by the following graph G_1 (see Figure 16).

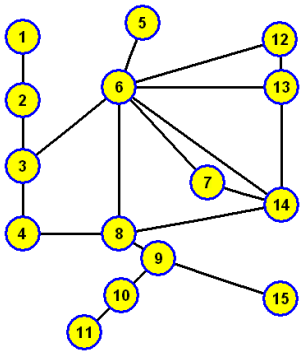

Figure 16 Graph G_1, graph-representation of the map of the puzzle *Towns*

The relation "be adjacent" defined as "town *x* is adjacent with town *y* if there is no same letter in their names" can be described by the graph G_2, whose vertices represent the towns and edges corresponding with the relation "be adjacent" (see Figure 17).

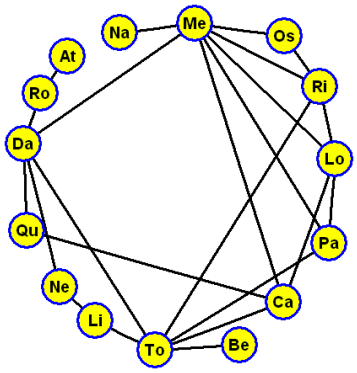

Figure 17 Graph G_2, graph-representation of the above-described relation "be adjacent"

From both figures it is obvious that the vertex 6 with the degree 7 must correspondent with the vertex *Me* (the only vertex with the degree bigger or equal 7). Let us draw the induced subgraph of the graph G_1 (of the graph G_2 respectively) containing the vertex 6 (the vertex *Me* respectively) and its neighbors (see Figure 18).

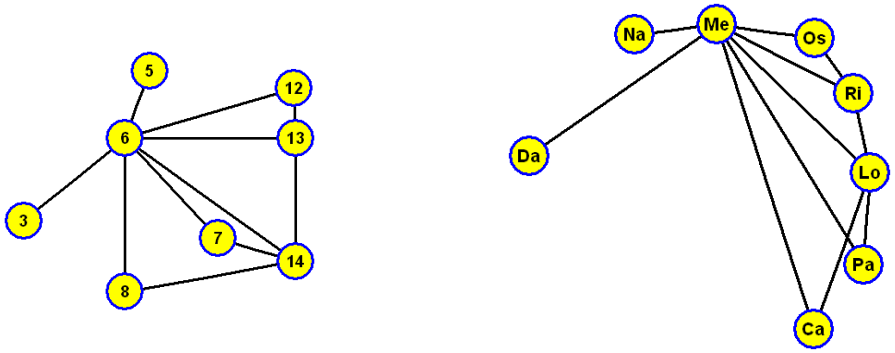

Figure 18 Subgraph of the graph G_1 and the subgraph of the graph G_2

The given subgraphs are isomorphic. There exist the following two bijections characterizing isomorphism of the given subgraphs:
I)
3 → *Da*, 5 → *Na*, 6 → *Me*, **7 → *Ca*, 8 → *Pa***, 12 → *Os*, 13 → *Ri*, 14 → *Lo*
II)
3 → *Da*, 5 → *Na*, 6 → *Me*, **7 → *Pa*, 8 → *Ca***, 12 → *Os*, 13 → *Ri*, 14 → *Lo*

We choose the second bijection (compare degrees of vertices in the given graphs G_1 and G_2) for isomorphism of the graph G_1 and the subgraph of G_2 and, **with help of the program 'Graphs'**, we draw the graph G_2 using the picture similar to the figure of the graph G_1 (see Figure 19).

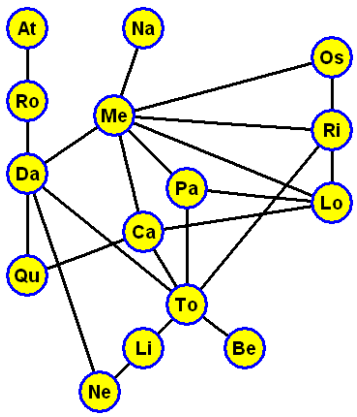

Figure 19 Graph G_2 – another picture

Hence, the rest of the demanded isomorphism is obviously determined as follows:
1 → *At*, 2 → *Ro*, 4 → *Qu*, 9 → *To*, 10 → *Li*, 11 → *Ne*, 15 → *Be*

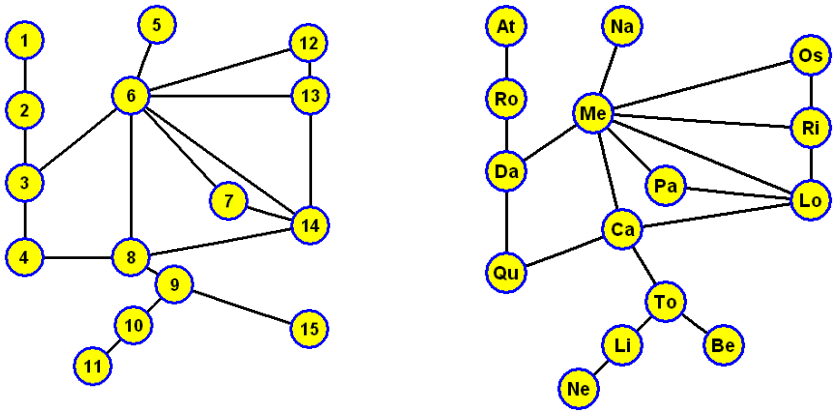

Figure 20 Result of the puzzle *Towns* – the found isomorphism

There are an endless number of enjoyable tasks, puzzles and logic problems in books like "Mathematics is Fun", in riddles magazines and on the Internet. We have been looking for problems from these sources that can be efficiently solved with the help of graphs and introduce them in lectures devoted to the appropriate topic.

3.2 Case study 2

From "Jarník to Dijkstra". Here we introduce a short example of our approach to teaching in context. We illustrate a relationship between Jarník's solution of the Minimum Spanning Tree problem and Dijkstra's algorithm finding the shortest path between two vertices.

One of the most fundamental problems is the Minimum Spanning Tree (MST in short) problem, which is generally regarded as a cornerstone of Combinatorial Optimization. Moreover, this problem should be interesting for our students for another reason, namely that two excellent **Czech mathematicians** Otakar Borůvka and Vojtěch Jarník are involved in the genesis of its solutions. The first formulation and solution of the problem was given in 1926 by Otakar Borůvka[1] and another solution added Vojtěch Jarník[2].

(Remark: The survey of the works devoted to the MST problem until 1985 is given in the article [6] and this historical paper is followed up in articles [7] and [8]. Moreover, in the paper [8] it is also shown that out of many available algorithms which solve the MST problem, Borůvka's algorithm is the basis of the fastest known algorithms)

[1] Otakar Borůvka was introduced to the problem by his friend, Jindřich Saxel, an employee of the West_Moravian Powerplants. It was at that time that electrification of the south and west parts of Moravia was beginning, and Borůvka was asked for help in solving the problem. The challenge was how and through which places to design the connection of several tens of municipalities in the Moravia region so that the solution was as short and consequently as low-cost as possible. Borůvka not only correctly stated this problem but also solved it in the papers [3] and [4]).

[2] Vojtěch Jarník quickly realized the novelty and importance of the problem after reading Boruvka's paper. However the solution seemed very complicated to him. He started to think about another solution and soon afterwards wrote a letter to Otakar Borůvka in which he suggested a much easier method and consequently he published it in the article [5].

Let us recap the formulation of the problem. In the contemporary terminology MST problem can be formulated as follows (see e.g. [9]):

Given a connected undirected graph $G = (V, E)$ with n vertices, m edges and real weights assigned to its edges (i.e. $w: E \rightarrow R$). Find among all spanning trees of G a spanning tree $T = (V, E')$ having minimum value $w(T) = \Sigma(w(e); e \in E')$, a so-called minimum spanning tree.

Here let us describe Jarník's solution to the MST problem. We describe it as an edge-coloring process (see [9]). The process we demonstrate to students on a connected undirected edge-weighted graph within the 'Graphs' program.

Jarník's algorithm

1. Initially all vertices and edges of the graph G are uncolored. Let us choose any single vertex and suppose it to be a trivial blue tree.
2. At each of $(n - 1)$ steps, color the minimum-weight uncolored edge, having one vertex in the blue tree and the other not, blue. (In the event that there are more such edges, choose any of them.)
3. The blue colored edges form a minimum spanning tree.

Much later, during the time of the newly developing field of computer science, R.C. Prim created the same solution as Jarník. He was unaware of Jarník's solution. He used a more detailed implementation suitable for computer processing in the paper [10].

To use the following description (see [11]) let us consider weights $w(e)$ assigned to edges of a given graph as distances.

Jarník's-Prim's algorithm

1. Let us choose any single vertex a and suppose it to be a blue tree. Put the value $(0, a)$ by the vertex a. By each vertex v which doesn't belong to the blue tree, the actual information $(f(v), u)$ is saved, describing the nearest distance $f(v)$ between the vertex v and the blue tree (from the vertex u). Thus initially by each vertex $v \neq a$ put the value $(w(\{a, v\}), a)$ if v is a neighbor of the vertex a and the value (∞, a) if v is not a neighbor of the vertex a.
2. At each of $(n - 1)$ steps take the following commands:
   - choose a vertex z with the actual information $(f(z), t)$ such that $f(z) = \min\{f(v);$ v doesn't belong to the blue tree$\}$,
   - color the corresponding edge $\{z, t\}$ blue,
   - by each neighbor v of the vertex z, change the value $(f(v), u)$ to the value $(w(\{z, v\}), z)$ in the case that $w(\{z, v\}) < f(v)$.
3. The blue colored edges form a minimum spanning tree.

During the lecture we illustrate Jarník's-Prim's solution of MST problem not only **step by step on a graph within the program 'Graphs'** but also in the way described below using the adjacency matrix of the given graph.

Suppose we have the following graph G_1 (see Figure 21) and its adjacency matrix (see Figure 22).

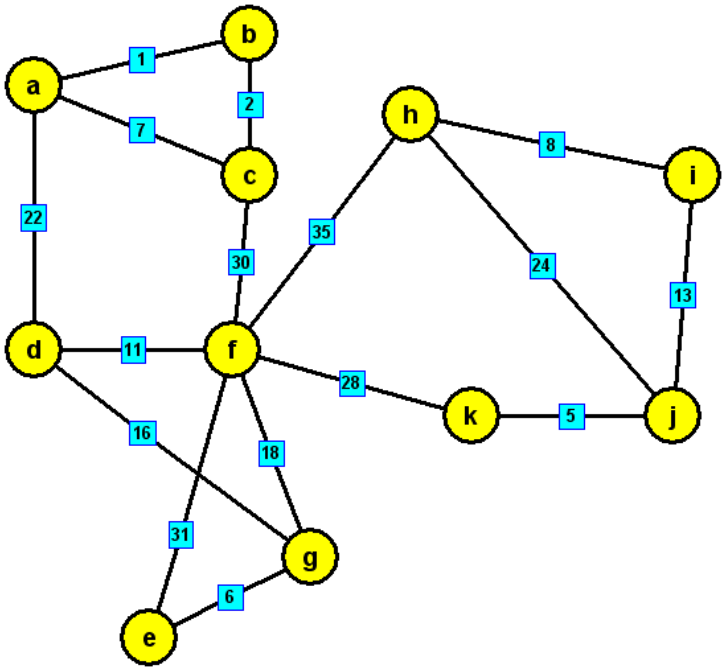

Figure 21 Graph G_1

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*
a		1	7	22							
b	1		2								
c	7	2				30					
d	22					11	16				
e						31	6				
f			30	11	31		18	35			28
g				16	6	18					
h						35			8	24	
i								8		13	
j								24	13		5
k						28				5	

Figure 22 Adjacency matrix of the graph G_1

Jarník's-Prim's solution of MST problem can be demonstrated as follows:

a	(0, *a*)									
b	(**1**, *a*)									
c	(7, *a*)	(**2**, *b*)								
d	(22, *a*)	(22, *a*)	(**22**, *a*)							
e	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(31, *f*)	(**6**, *g*)				
f	(∞, *a*)	(∞, *a*)	(30, *c*)	(**11**, *d*)						
g	(∞, *a*)	(∞, *a*)	(∞, *a*)	(16, *d*)	(**16**, *d*)					
h	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(35, *f*)	(35, *f*)	(35, *f*)	(35, *f*)	(24, *j*)	(**8**, *i*)
i	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(**13**, *j*)	
j	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(**5**, *k*)		
k	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(28, *f*)	(28, *f*)	(**28**, *f*)			

Blue colored edges of found minimum spanning tree are $\{a, b\}$, $\{b, c\}$, $\{a, d\}$, $\{g, e\}$, $\{d, f\}$, $\{d, g\}$, $\{i, h\}$, $\{j, i\}$, $\{k, j\}$, $\{f, k\}$.

Let's ask a question. Could the solution of the MST problem also serve as a solution to the problem of how to find the shortest path from one vertex to another in a connected undirected graph with *n* vertices and non-negative weights assigned to its edges?

The answer is no. It is easy to see from the figure 3.2.1 that the shortest path from the vertex *a* to the vertex *h* in the graph G_1 goes through vertices *b*, *c* and *f*. Its length is 68 while the length of the path (*a*, *d*, *f*, *k*, *j*, *i*, *h*) between the same vertices in the blue minimum spanning tree found above is 87! However, there is a relationship between the two problems.

Let's consider the following small **modification of Jarník's-Prim's algorithm:** At each step of the Jarník's-Prim's algorithm, by each vertex *v* which doesn't belong to the blue tree, save the actual information describing the nearest distance between the vertex *v* and the initial vertex *a* (instead the nearest distance between the vertex *v* and the blue tree).

In this way we get the correct solution, namely the solution found in 1950's by E.W.Dijkstra (see [12]).

At first **we illustrate both algorithms on the same graph opened in two windows in the program 'Graphs'** and emphasize the difference between them (see Figure 23).

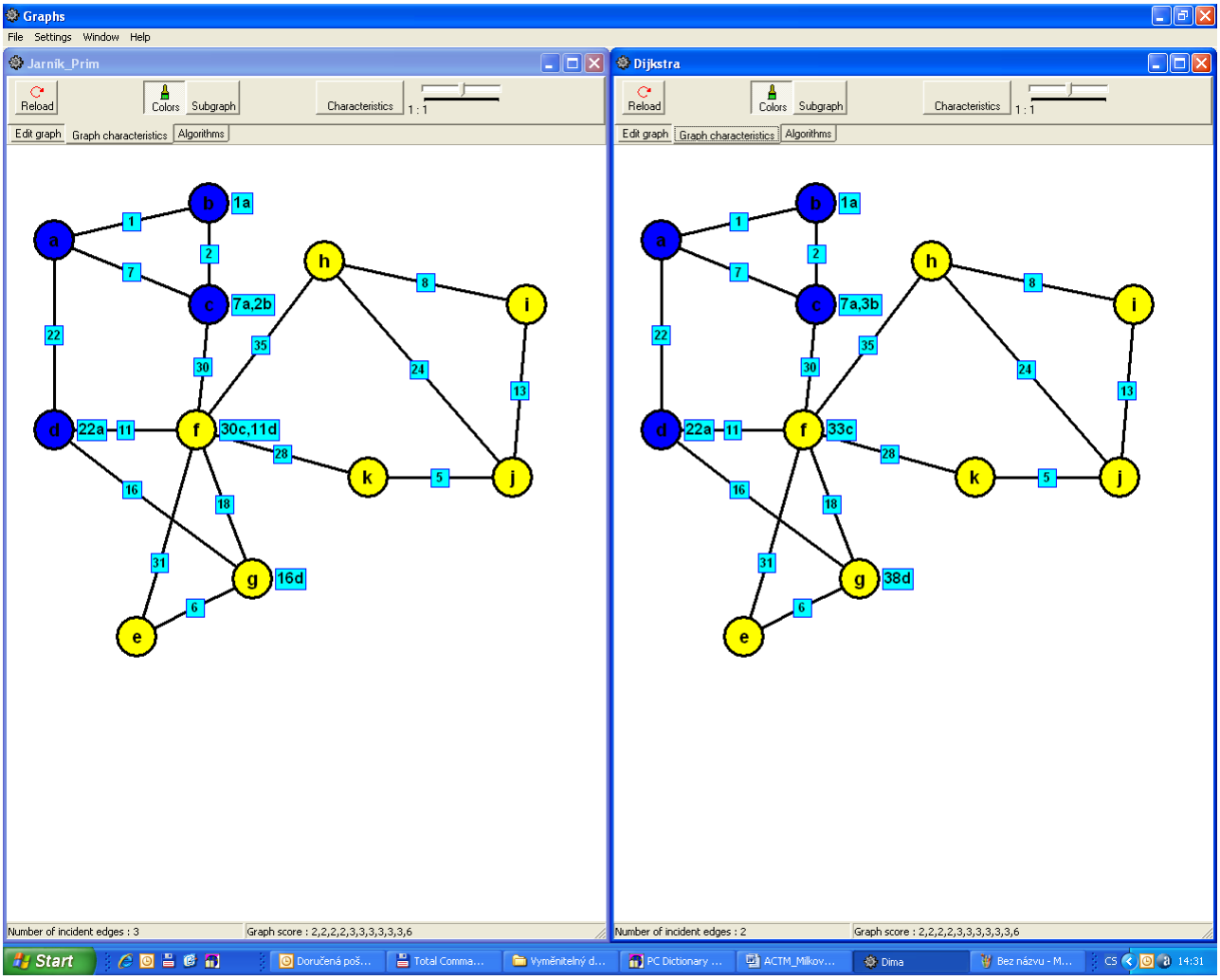

Figure 23 Third step of Jarník's-Prim's and Dijkstra's algorithm in two opened window of the program 'Graphs'

Then we let students to solve Dijkstra's algorithm finding the shortest path between two vertices of a given graph using its adjacency matrix in the way described above, i.e.:

Dijkstra's solution of finding the shortest path from the vertex *a* to the vertex *h* in the graph G_1

a	(0, *a*)									
b	(**1**, *a*)									
c	(7, *a*)	(**3**, *b*)								
d	(22, *a*)	(22, *a*)	(**22**, *a*)							
e	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(64,f)	(**44**,g)				
f	(∞, *a*)	(∞, *a*)	(33, *c*)	(**33**, *c*)						
g	(∞, *a*)	(∞, *a*)	(∞, *a*)	(38, *d*)	(**38**, *d*)					
h	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(68, *f*)	(68, *f*)	(68, *f*)	(68, *f*)	(**68**, *f*)	
i	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(79, *j*)	(**79**, *j*)
j	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(**66**, *k*)		
k	(∞, *a*)	(∞, *a*)	(∞, *a*)	(∞, *a*)	(61, *f*)	(61, *f*)	(**61**, *f*)			

The shortest path from the vertex *a* to the vertex *h* in the graph G_1 is (*a*, *b*, *c*, *f*, *h*).

4 Summary and conclusion

Students have all graphs used during the lectures of the subject Graph Theory and Combinatorial Optimization together with the program 'Graphs' and other electronic study materials assigned to the subject available in the virtual learning WebCT environment that has been in use at our university. They also have available there various self-tests with automatic checking.

In this paper we offered some ideas how to improve both the teaching and learning of the subject Graph Theory and Combinatorial Optimization through modern technology. The author would appreciate any comments and any co-operation in the mentioned fields.

Reference

[1] Pozdílek M. (2004). *Grafové algoritmy: vizualizace*. Diploma thesis, Hradec Králové, CZ: University of Hradec Králové.

[2] Williams, R. (2005). *e-Learning Strategy: What's in the Blend?*. In: Proceedings of the 4th Europ. Conference on e-Learning, Amsterdam, Nederland: ACL, pp. 245-51.

[3] Borůvka, O. (1926). *O jistém problému minimálním*. Práce Mor. Přírodověd. Spol. v Brně, 3, pp. 37-58.

[4] Borůvka, O. (1926). *Příspěvek k řešení otázky ekonomické stavby elektrovodních sítí*. Elektrotechnický obzor, 15, pp. 153-154.

[5] Jarník, V. (1930). *O jistém problému minimálním*. Práce Mor. Přírodověd. Spol. v Brně, 6, pp. 57-63.

[6] Graham, R. L., Hell, P. (1985). *On the History of the Minimum Spanning Tree Problem*. Annals of the History of Computing 7, 1, pp. 43-57.

[7] Nešetřil, J. (1997). *A few remarks on the history of MST-Problem*. Archivum Mathematicum Brno, 33, pp. 15-22

[8] Nešetřil, J., Milková, E., Nešetřilová, H. (2001). *Otakar Borůvka on minimum spanning tree problem: Translation of both the 1926 papers, comments, history*. Discrete Mathematics, 233, pp. 3-36.

[9] Milková, E. (2007). *THE MINIMUM SPANNING TREE PROBLEM: Jarník's solution in historical and present context*. Electronic Notes in Discrete Mathematics, 28, pp. 309–316

[10] Prim, R.C. (1957). *The shortest connecting network and some generalization*. Bell. Systems Tech. J. 36, pp. 1389-1401.

[11] Milková, E.: *Combinatorial Optimization: Mutual Relations among Graph Algorithms*. WSEAS TRANSACTIONS on MATHEMATICS, Issue 5, Volume 7, May 2008, pp.869-879.

[12] Dijkstra, E. W. (1959), *A note on two problems in connection with graphs*, Numer. Math.,1, pp. 269-271.