Helena Durnová

A history of discrete optimalization

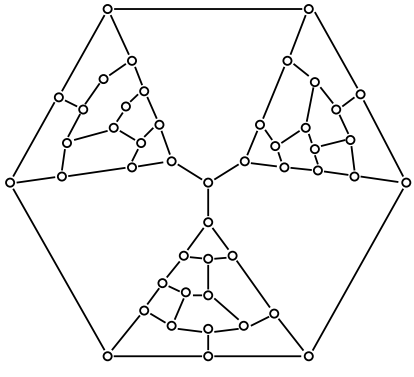
In: Eduard Fuchs (editor): Mathematics throughout the ages. II. (English). Praha: Výzkumné centrum pro dějiny vědy, 2004. pp. 51–184.

Persistent URL: http://dml.cz/dmlcz/401175

MATHEMATICS THROUGHOUT THE AGES II

A History of Discrete Optimization

Helena Durnová



PRAGUE 2004

This is a Ph.D. thesis written under the supervision of Prof. Eduard Fuchs, defended at the Department of Mathematics of the Faculty of Science of Masaryk University in Brno on 24st May 2001.

Contents

Chapter 1

Introduction

> *So also the games in themselves merit to be studied and if some penetrating mathematician meditated upon them he would find many important results, for man has never shown more ingenuity than in his plays.*[1]

In the quotation above, games are seen as the starting point of theories or even new branches of science. In discrete optimization, many a problem has its roots in games or puzzles. These roots, almost as often, intermingle with the economic aspects of a particular problem. Therefore it is legitimate to speak about two contexts in connection with the discrete optimization problems discussed in this thesis: economizing and graph theory. The latter is a result of the choice of the problems, or their solutions: all of them — shortest path problems, network flow problems, minimum spanning tree problem, and the travelling salesman problem – nowadays belong to graph–theoretical problems. Although these problems appeared independently of graph theory, graphs later provided a suitable representation for their description. This chapter shows the mutual influences of the two in a broader historical context.

[1] G. W. Leibniz, taken from Ore, O: Pascal and the Invention of Probability Theory, *American Mathematical Monthly* **67** (1960), 409–416. Quoted in [BCL70].

1.1 Optimality and Mathematics

> *The origin of the present methods provides an interesting illustration of the value of basic research on puzzles and games. Although such research is often frowned upon as being frivolous, it seems plausible that these algorithms might eventually lead to savings of very large sums of money.*[2]
>
> – E. F. Moore

Finding the optimum solution has been one of the crucial problems for a whole range of pure and especially applied mathematical disciplines. The optimum here stands, of course, either for the minimum or the maximum subject to certain constraints. Seeking the best — optimum — solution is an integral part of mathematics. According to MORRIS KLINE [Kli72], the following misunderstanding is often quoted: the Egyptians knew that a square has the largest area of all the rectangle whith the same perimeter. As KLINE says, this is inaccurate, for we the only thing we know is that they *used* this fact. They, however, might have arrived to this conclusion by simply trying out the possibilities and watching the changes: they were aware of the facts — i.e. the relations between the length of the sides of a rectanle and its area — but they need not have *formulated* or *proved* these statements. Thus, problems solved by mathematicians, and not only those that have applications in economy, are often connected with finding minimum or maximum of something. Maximum area with minimum perimeter, maximum volume with minimum surface, or winning strategies for games are only a few examples. To put it simply, mathematicians are often interested in the "extremes". In the following sections, the roots of the optimality concept in mathematics, and in particular the environment in which graph–theoretical algorithms originated, is described.

A more conscious notions of mathematical optimality can be found in the works of scientists from the half of the 18th century on.[3] As Brentjes says [Bre94], optimality was studied in the 18th and 19th centuries by EULER, ROGER BOSCOVICH, and PIERRE SIMON LAPLACE and LAPLACE in geodesy and cartography, by EULER in astronomy, and by FOURIER, CLAUDE NAVIER, ANTOINE AUGUSTIN COURNOT,

[2] E. F. Moore, 'The shortest path through a maze.' *Proceedings of an International Symposium on the Theory of Switching*, 1957, p. 292.

[3] For more detail, see Sonja Brentjes, *Linear Optimization*, p. 829. In: *Companion Encyclopedia of the History and Philosophy of the Mathematical Science*, edited by Ivor Grattan-Guiness, Routledge 1994.

and MIKHAIL OSTROGRADSKY in analytical mechanics. CAUCHY and FOURIER studied the generalized problem of n equations or inequalities with m variables already in the first half of the 19th century. In the second half of the 19th century, optimality problems appear also in mathematics itself.

The connection between economic theories and mathematics can only be traced since the 1830s [Bre94]. Among the pioneers of these methods, WALRAS and PARETO with their *marginal-utility theory*[4] can be found. A major onset of the use of mathematical methods in economy, however, took place only in the 20th century.

In the 1930s, the scientists in the Soviet Union started working on optimization methods, with the aim to maximize production in socialist economy. L. V. KANTOROVICH was the one who, at the age of 27, started working closely with economists and is probably the most famous Russian in this branch. He developed a special algorithm for the solution of the so-called transportation problem. However, his work was not known much outside the block of socialist countries and also, the Russian economists were "afraid of mathematics", which prevented more widespread use of mathematical methods in economy.

During and immediately after the World War II, scientists in the U.S.A. started working on optimization related to economy. Their work was carried out independently of the work done in the Soviet Union. It was only in the 1960s, when the Soviet works were translated into English, that Americans learnt about the results achieved by Soviet scientist. Consequently, the priority debate started.[5]

Operational Research

According to the *Soviet Mathematical Encyklopaedia* [SME], *Operational (Operations) research* is

> the construction, elaboration and application of mathematical models for making optimal decisions. The theoretical side of operations research is concerned with the analysis and solution of mathematical problems of choosing from a given set X of *feasible decisions* an element satisfying some criterion of optimality, called an *optimal decision* of the problem.

[4] Each additional unit costs more/less/the same as the previous one. The cost of the additional unit is called "marginal cost", the utility provided by this additional unit "marginal utility'.

[5] For more details on the priority debate, see [Bre94].

The history of operational research goes back only to the time just before the Second World War. The connection with war is not purely coincidental here: as DANTZIG says, operational research starts where one general cannot overlook the whole battlefield [Dan63]. Later, this "application of the scientific method to finding the most economical and timely means for getting the maximum military effect from the available or potentially available resources in materiél or personnel" [6] spread also to economical applications.

When solving problems in operations research, the set of feasible decisions and the criteria of optimality have to be determined. It is also necessary to know what information exactly the person taking decision can have at disposal. The problem to be solved can thus be either *static* (The information necessary for decision-taking is given in advance and does not change), or *dynamic* (the information stages change one another). Further, the decision problem may be *indefinite* (when information state contains the set of physical states of the subject — such problems are considered by the theory of games), *stochastic* (when the probability of each state is a priori known), *deterministic* (when there is only one possible state, and *parametric*, which comprises the solving of individual cases of indefinite problems. It should be noted that solving stochastic problem means finding a decision that is "optimal on average" over the entire set of problems.

The methods used in operations research include *mathematical (optimal) programming* for parametric and deterministic problems, *stochastic programming* for stochastic problems, and *game theory* for indefinite problems. Mathematical programming can further be divided acoording to the *objective function* on the set of feasible solutions into *linear, convex*, and *quadratic*. The set of feasible solutions may be *continuous* or *finite*. If the set of feasible solutions is finite, we speak about *discrete programming* with special parts: *integer programming* and *Boolean programming*.

To close the section on operational research, let us quote the Soviet Encyclopaedia of Matematics [SME, entry 'Operations research, emphasis mine] again:

> One of the contemporary (since 1970's) trends in operations research is the transition from the consideration of individual problems to the study of systems, spaces, calculi of such

[6] Canadian definition from 1948, quoted by Robin E. Rider in "Operational research" [GG94, p. 837].

> problems, and to the study of relations between various problems or the reduction of some problems to others that have a simpler structure. The mathematical apparatus intended for and developed with the aim of solving problems of operations research is *conveniently* called the mathematics of operations research.

Programming

The term "programming" has two basic meanings — the first one is closely connected to computers, the second one is connected with economy and the decision–taking process. A common name of the latter category is *mathematical programming*, which comprises *linear, non–linear, quadratic, convex, integer, stochastic*, and *dynamic* programming. In the previous section, *linear programming* was mentioned, where the conditions and the purpose function are expressed by linear inequalities. In *non-linear programmming*, the conditions and the purpose function are determined by a set of equations, generally non-linear. Similarly, the other kinds of programming can be described in a similar manner.

From the point of view of certainty contained in the problems, programming problems can be divided into the *deterministic* ones and those in which some uncertainty is included — i.e. *stochastic*. The deterministic group is further divided into linear (general structure and special structure) and non-linear (convex and non–convex) ones. The stochastic group can then be subdivided into problems without opponents (with known or unknown distribution of probabilities) and with opponents (two–person or n–person games). Variables used in stochastic problems can be both discrete and continuous.

Discrete programming problems, which are dealt with in this thesis, form a part of non-linear programming. [Dan63, pp. 24-25]

Theory of games

An area of mathematics dealing with stochastic problems arising in economy is the theory of games, commonly believed to have been founded by Oskar Morgenstern and John von Neumann, who published their *Theory of Games and Economic Behavior* in 1944. It is, however, not so easy to say who and when arrived at major results, such as the *minimax theorem* in the theory of games [Kje98], as the results now recognised as

belonging to game theory arose in different areas of mathematics.[7]

The RAND Corporation

The research in applied mathematics in the United States is associated with the *RAND Corporation*. The RAND Corporation first formed part of the *Douglas Aircraft* company (from March 1946 until May 1948). From May 1948 until 1962, it was funded mostly by the *U.S. Air Force Department*. It is sometimes described as a (or even as "the") "think–tank" [Hou97, p. 240]:

> RAND became the prototype for a method of organizing and financing research, development, and technical evaluation that would be done at the behest of government agencies, but carried out by privately run nonprofit research centers. [...] The RAND model flourished in flourished in the 1950s [...] [8]

As the RAND Corporation originated during wartime, many of its activities are connected with war. Mathematicians, for example, were working in the project that started as "science of warfare". The department of economics, on the other hand, was concerned with the application of game theory to the United States — Soviet Union relationship. In 1949, the RAND Corporation consisted of seven departments: Missiles, Electronics, Aircraft, Mathematics, Social Science, Nuclear Physics,and Economics.

One of the most famous result of the work of this department is the *simplex algorithm* for solving optimization problems, outlined first by GEORGE B. DANTZIG. DANTZIG's paper remained classified until 1951. The research that eventually led to the estblishment of linear programming began during the World War II and the military influence here is apparent. Mathematicians working in the U.S. Air Force Department — apart from G. B. DANTZIG also H. W. KUHN and A. W. TUCKER — then started with building a mathematical model. However, the discovery of the *duality theorem* in linear programming rendered this area interesting also from the purely mathematical point of view [Kje99].

Another class of methods, *dynamic programming*, was developed by another mathematician working for the RAND Corporation, RICHARD

[7] Similarly, discrete-optimization problems were solved by mathematicians irrespective of whether graph–theoretical (or other) methods existed.

[8] James Allen Smith: *The idea brokers: Think tanks and the rise of the new policy elite.* New York, 1991, p. xiv. Quoted by David Hounshell in [Hou97, p. 240]

Bellman. The methods advocated by Bellman are designed for solving problems under the conditions of uncertainty.

Third major area of mathematics developed within the RAND Corporation was *game theory*. The mathematicians who worked in this branch in the RAND Corporation were, among others, Lloyd Shapley, Merrill Flood, or Kenneth Arrow.[9]

[9]Julia Robinson, Delbert R. Fulkerson, or Lester R. Ford also belong among the researchers of the RAND COrporations.

1.2 Graph Theory

> *The origins of graph theory are humble, even frivolous. Whereas many branches of mathematics were motivated by* fundamental problems *of calculation, motion, and measurement, the problems which led to the development of graph theory were often little more than* puzzles*, designed to test the ingenuity rather than to stimulate the imagination. But despite the apparent triviality of such puzzles, they captured the interest of mathematicians, with the result that graph theory has become a subject rich in theoretical results of a surprising variety and depth.* [BLW76, p. 1, emphasis mine]

Graph theory is a discipline that is by some described as a "collection of applications". Although this characterisation might sound a bit crude, several features support this statement. To begin with, graph theory is used almost anywhere where the use of nodes and lines seems appropriate. In other words, the roots of graph theory are multiple. As a result of this, graph–theoretical terminology is not unified. The confluence of these small rivers into a great river of graph theory is done exactly through the use of point—line (or, alternatively, vertex/node — arc/branch/edge) terminology.

As far as the origins of graph theory are concerned, we may trace different beginnings in dependency on different notions as to what an origin is. It is of course possible to see a graph in any configuration of points and lines. However, most people speak of LEONHARD EULER (and the years 1736 or 1735 or 1742) as the founder of graph theory, although his usage of the notions "line" (edge) and "point" (vertex) is not central to his paper. Probably the first graph–theorist in the modern sense of the word was the Hungarian DÉNES KŐNIG, the author of the first graph–theoretical textbook.[10]

Some important dates in the history of graph theory

In this section, history of problems that are in some way related to one of the four problems dealt with in this thesis is outlined. Therefore, the list by no means complete.[11]

[10] Anoine Saint-Lagüe, a Frenchman who published his works resembling graph theory already in 1920s, is sometimes also quoted.

[11] Detailed history of graph theory can be found in the monograph *Graph theory 1736–1936* (1976) by Norman L. Biggs, Keith E. Lloyd, and Robin J. Wilson [BLW76]

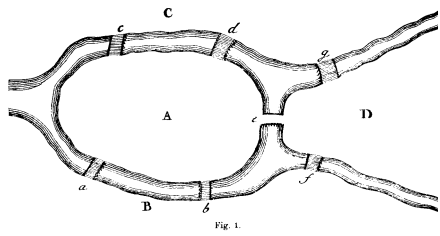


Figure 1.1: The Seven Bridges of Königsberg

1735/1736/1742 — Leonhard Euler: The seven bridges of Königsberg

Graph theory is commonly believed to have originated with EULER's solution of the *Königsberg bridges problem*, which EULER was presented with in 1735. He communicated his solution to the St. Petersburg Academy of Sciences in 1736. The third date that is sometimes used in connection with this problem is 1742, which is the date when EULER's solution was actually printed.

EULER looks at the problem for which there appears to be no solution and formulates a general criterion for solving similar problems. He, however, *did not* represent the situation with a graph in the modern sense of the word. He even does not speak about points and lines, but denotes the four islands by capital letters A, B, C, and D and the seven bridge by a, b, c, d, e, f, and g (see Fig. 1.1). He is then looking for a sequence of the letters A, B, C, and D such that each letter appears precisely as many times as there are bridges leading to the island denoted by the respective letters. Before generalizing his ideas, EULER says [BLW76, p. 5, English translation of Euler's paper]:

> So in a series of eight letters, representing the crossing of seven bridges, the letter A must occur three times, and the letters B, C and D twice each—but this cannot happen in a sequence of eight letters. It follows that such a journey cannot be undertaken accross the seven bridges of Königsberg.

This problem, in its general form, was solved also by CARL HIERHOLZER and published in 1873. It was probably inspired by the book *Vorstudien zur Topologie* by J. B. LISTING.

1845–1847 — G. R. Kirchhoff: Kirchhoff's laws

In 1845, GUSTAV ROBERT KIRCHHOFF published two rules for the flow of electricity in a network. These rules nowadays bear his name. The first of them – the one that is used in network flow problems — says that the electrical flow into and out of a certain node of the network must be equal for any node. In 1847, he published the result which, in graph theory, can be interpreted as finding the number of fundamental circuits in a graph. [12]

1856–1859 — W. R. Hamilton: "A Voyage Round the World"

WILLIAM ROWAN HAMILTON described his idea of *icosian calculus* in a "MEMORANDUM" respecting a New System of Roots of Unity" (*Philosophical Magazine*, December 1856). He presented his ideas also at the meeting of the British Association in Dublin in 1857. He sold the *Icosian Game* for only £25 to the company JAQUES AND SON. It was patented and marketed in 1859. The accompanying leaflet, written mainly by HAMILTON himself, contains five problems and suggestions for playing the game.

As BIGGS, LLOYD, and WILSON point out in [BLW76], similar problems were considered — even in a more general form — by T. P. KIRKMAN in 1856, but HAMILTON is the one who gave his name to the problem.

1857–1889 — A. Cayley: The Number of Trees

In his articles on the "analytical forms called trees" (published between 1857 and 1889),[13] CAYLEY presented his result concerning the number of trees on n knots. He is first concerned with so–called "rooted trees" and some other special classes of trees. In 1881, He published his result concerning the number of non–isomorphic trees on n nodes.

In 1889 he determined the number of *labelled trees* of a complete graph. When counting labelled trees, we consider the edges included in the tree, although the resulting graphs are isomorphic. The following

[12] The two papers published by Kirchhoff are:
Kirchhoff, G.: Über den Durchgang eines elektrischen Stromes durch eine Ebene unbesondere durch eine Kreisformige. *Annalen Phys. Chem.* **64** (1845), 492–514.
Kirchhoff, G.: Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Poggendorff Ann. Phys.*, 72 (1847), 497–508.

[13] See [BLW76, pp. 37–54] for details.

formula is appropriate for determining the number of possible solutions of the minimum spanning tree of a complete graph:

$$t_n = n^{n-2}$$

According to [BLW76, p. 52], CAYLEY's proof of the formula was not complete. A complete proof was given by PRÜFER in 1918.

1895 — G. Tarry: Labyrinths and mazes

In 1895, G. TARRY published his solution to the problem of labyrinth or maze, one of the well-known mathematical puzzles. His algorithm is nowadays used for searching graphs and is called *depth–first search*. He formulates the rule for traversing a maze in a single journey, passing twice along each passage [BLW76, p. 18, English translation of Tarry's paper]:

> *Do not return along the passage which has led you to the junction for the first time unless you cannot do otherwise.*

The rest of the paper consists of describing the method of marking the passages in such a manner that the traverser arriving to a junction knows which passages have been traversed and which led to the junction for the first time.

1936 — D. Kőnig: Theorie der endlichen und unendlichen Graphen

This book by DÉNES KŐNIG (published by Teubner, Leipzig in 1936) caused a major breakthrough in the history of graph theory as a branch of mathematics. Here is what PÁL ERDŐS said about this book in 1986 [Kőn86, English translation of the text on the cover of the 1986 edition of Kőnig's monograph]:

> The original of the work reprinted here by Dénes Kőnig was issued in 1936. [...] The stormy and fruitful development of graph theory that started shortly after this book was issued — and interrupted by the World War II — is to a great extent due to this work and its author, who unfortunately also fell a victim to fascism himself.[14]

[14] The original quotation in German can be found in the Appendix (page 172).

The list of chapters shows clearly that KŐNIG managed to describe all the basic parts of graph theory, especially when compared with later monographs on graph theory by BERGE [Ber58], ORE [Ore62], and HARARY [Har69].

Nowadays, graph theory can be divided into more branches. Some of them are connected more to topology, some more to computers. Another branch of graph theory, random graph theory, uses probability notions for graphs. Graph theory in its many forms has applications in electricity, linguistics, sociology, biology, and other areas of human life.

1.3 Objectives of the Thesis

Articles and books dealing with the subject of discrete optimization are not very numerous. Yet there are a few worth mentioning. Probably the best known historical account of a discrete optimization problem can be found in the paper [GH85], "On the history of the minimum spanning tree problem". Another paper related to the same subject was published by Korte and Nešetřil in the monograph on Vojtěch Jarník [Nov99, KN99]. Historical notes — although not always reliable — can be found also in the original mathematical papers. Several review papers on discrete optimization problems also exist: shortest path problems are reviewed in [PW60] or [Dre69], travelling salesman problem in [BN68].

Form the historical works on disciplines closely related to the subject, the history of graph theory is also interesting. Monograph [BLW76] gives a full account of the history of graph theory up to the year 1936 (with some remarks on further development). Work of Czech and Slovak mathematicians is carefully reviewed in a book by Pavel Šišma [Šiš97]. The aspects of multiple discovery in non–linear programming is dealt with in Tinne Hoff Kjeldsen's Ph.D. thesis [Kje99].

The objective of this thesis is to show the early development of graph theoretical algorithms for the shortest path, minimum spanning tree, network flows and the travelling salesman problem. It particularly focuses on the following areas of the development of the problems:

Formulations and context: The aim in this area is the description of the development of formulations and/or solutions of some discrete optimization problems, more specifically, shortest paths, network flows, the minimum spanning tree, and the travelling salesman problem.

The context and the above–mentioned change in formulations of the problems and/or solution are sometimes two sides of the same coin. The objective of this thesis is the analysis of the influence of the context on the formulations of the problems as well as of the areas in which discrete optimization problems were encountered.

"Historical precision" in mathematical papers: As was already mentioned, mathematical papers and textbooks often contain "historical notes". These notes are repeated over and over, regardless of whether they are true or not. In this thesis, the aim is to disrupt or challenge these common beliefs.

Language: The main objective of this thesis is the analysis of the language used in the algorithms and its development over time. The analysis of the onset of complexity theory is taken as an example of the development of new dialect within mathematics, a dialect which developed also as a result of the need to talk about algorithms in a standardised way.

Chapter 2

Mathematical Background

In this chapter, basic graph-theoretical definitions will be presented in order to facilitate further reading. Some remarks about algorithms and complexity measures will also be made.

2.1 Basic Definitions in Graph Theory

As far as graph theory is concerned, we will limit ourselves to finite graphs, both in this section and in the thesis as a whole. As graph–theoretical terminology is not unified, names of alternative concepts are given in brackets where appropriate.

2.1.1 Directed and undirected graphs

Definition 2.1 *A finite* undirected graph $G(V, E)$ *consists of a finite non–empty set* V *of elements (called* vertices*) and a subset* E *of the set of* unordered *pairs* $v_i v_j$*, where* $v_i, v_j \in V$ *(called* edges*) of the elements of* V*.*

A finite directed graph $G(V, E)$ *consists of a finite non–empty set* V *of vertices and a subset* E *of the set of* ordered *pairs of the elements of* V*.*[1]

The above definitions allow also *loops*, i.e. the (directed or undirected) edges of the type $v_i v_i$ — in other words, edges whose endpoints are identical. Graphs not containing any loops are called *simple*.

In the following, only the word *graph* will be used instead of the phrase *simple finite graph*, as infinite graphs or graph with loops will not

[1] Apart from finite graphs, locally finite graphs and infinite graphs can also be defined.

be dealt with. Geometrically, vertices are often represented by points in a plane and edges by lines connecting (some of) the vertices.

Elements of the set V are called *vertices, nodes, junction points*, or simply *points*; elements of E are called *edges, branches, arcs*, or simply *links*. Sometimes, the term "edge" is used for an undirected pair of vertices, while the term "arc" for a directed pair of vertices, but this is not a universal distinction. In the following chapters, the vertex–edge terminology will be used for both directed and undirected graphs; that is, apart from the quotations.

Definition 2.2 *If $v_iv_j \in E$ is an edge of $G(V, E)$, then the vertices v_i, v_j are called the* endpoints *of the edge.*

Definition 2.3 *The* degree *of vertex v, denoted as $d_G(v)$, is the number of edges with v as an endpoint.*[2]

The following theorem is evident:

Theorem 2.4 *The sum of the degrees of all the vertices $\Sigma_{i=1}^{n} d_G(v_i), v_i \in V$, of a finite undirected graph $G(V, E)$ with n vertices is even.*[3]

As a direct consequence of this theorem, this corollary holds:

Corollary 2.5 *The number of the vertices of odd degree in a finite graph is even.*

Before defining connectedness of a graph, we need to define the notions of *path* and *simple path* in a graph.

Definition 2.6 *A sequence (alternatively "chain") $v_0, v_0v_1, v_1, \ldots, v_{n-1}v_n, v_n$ of vertices $v_i \in V, i = 0, 1, \ldots, n$ and edges $v_iv_{i+1} \in E, i = 0, 1, \ldots, (n-1)$ of a graph is called a* path. *A chain in which the vertices $v_0, v_1, \ldots, v_n$ are mutually different is called a* simple path.[4]

[2]Each loop is counted twice. In a directed graph, we can further distinguish between the *in-degree* and the *out-degree* of a vertex; the former denotes the number of edges going into the vertex and the latter the number of edges going out of the vertex.

[3]Analogously, it can be proved that in a directed graph, the sum of the *in–degrees* and the sum of *out–degrees* of the individual vertices are equal.

[4]In a directed graph, the edges in the path must of course follow the orientation of the edges in the given graph.

The terminology for the concepts of *path* and *simple path* is quite diverse, with different authors using different names for the same concept.[5] The notion of *connected graph* shall now be introduced:

Definition 2.7 *An undirected graph $G(V, E)$ is called* connected *if a path exists between any two of its vertices, $v_i \neq v_j$.*[6]

In graph theory, special terms are used for various kinds of graphs. Some of these special classes of graphs are described in the definitions below.

Definition 2.8 *An undirected graph $G(V, E)$ is called* regular *if all its vertices are of the same degree. A regular graph whose vertices are of the degree n is called a* regular graph of the nth order.

Definition 2.9 *A chain $v_0, v_0v_1, v_1, \ldots, v_{n-1}v_n, v_n$ in $G(V, E)$ in which no vertex $v_i, i = 1, \ldots, (n-1)$ appears twice and the two endpoints of which are identical, i.e. $v_0 = v_n$, is called a* cycle.

Definition 2.10 *A graph $G(V, E)$ that does not contain any cycles is called a* forest.

A connected forest is called a tree.

Definition 2.11 *A* complete graph *is such an (undirected) graph in which every two vertices $v_i \neq v_j$ are connected by an edge.*

Sometimes we need to work with smaller parts of the graph. The two most often used notion are that of a *subgraph* and a *factor*:

Definition 2.12 *Graph $G_1(V_1, E_1)$ is called a* subgraph *of the graph $G(V, E)$, if $V_1 \subseteq V$ and $E_1 \subseteq E$.*

If $V_1 = V$, then $G_1(V, E_1)$ is called a factor *of $G(V, E)$.*

[5] As was already stated, diversity of the terminology is typical for graph theory as a whole.

[6] For directed graphs, two kinds of connectedness are defined: A directed graph is called *(weakly) connected* if a path in either of the two directions exists between every two vertices of the graph. The graph is called *strongly connected* if paths in both directions exist between every two vertices of a graph.

Connectedness is one of the central notions in graph theory. For example, some theorems are stated for a connected graph: namely, it only makes sense to look for the shortest paths, minimum spanning trees, or maximum flow in connected graphs. Graphs that are not connected can be divided into so–called *components*, parts of graph that are connected.

Factoring of graphs forms a whole branch of graph theory. For the purposes of this thesis, three factors are of particularly important: a *matching*, a *Hamiltonian cycle*, and a *spanning tree*.[7]

Definition 2.13 *Regular factor of a graph $G(V, E)$ of the first order is called a* matching *in G.*

Regular connected factor of a graph $G(V, E)$ of the second order is called a Hamiltonian cycle *in G.*

Before moving onto the weighted graphs, another well–known notion will be introduced: that of *Eulerian path*.

Definition 2.14 *A path $v_0, v_0v_1, v_1, \ldots, v_{n-1}v_n, v_n$ of vertices $v_i \in V, i = 0, \ldots, n$ and edges $v_iv_{i+1} \in E, i = 0, \ldots, n-1$ of the graph $G(V, E)$, containing all the edges of the graph exactly once is called* Eulerian path.

Eulerian path can be either *closed* or *open*. In a *closed Eulerian path*, the first and the last vertex of the sequence are identical (i.e. $v_0 = v_n$), while in an *open Eulerian path*, they are not.

The notion of *Hamiltonian cycle* is in a way similar to that of Eulerian path; only in a Hamiltonian cycle, we require each *vertex* (not edge) to be passed once and only once. Not all the edges of the graph must be included in the Hamiltonian cycle.

Both the preceding definitions gave names to special classes of graphs: a graph containing an Eulerian path is called an *Eulerian graph* and a graph containing Hamiltonian cycle is called a *Hamiltonian graph*. Despite the aforementioned "similarities", however, the tasks of determining whether the graph is Hamiltonian and finding Hamiltonian cycle are much more difficult than the tasks of determining whether the graph is Eulerian and finding Eulerian path.[8]

2.1.2 Weighted graphs

In discrete optimization problems, the graphs dealt with are mostly *weighted*. In practice, we proceed in the opposite direction; i.e. we know what needs to be described, and a graph serves as a suitable representation. Working with this kind of model can yield more problems: e.g. normally, one would not immediately think of negative weights, but when we generalize the concept of weight as in the following definition, we have to take negative weights into account as well.

[7] The notion of "spanning tree" will be defined in Chapter 5.

[8] More details on Eulerian and Hamiltonian graphs can be found in Chapter 6.

Definition 2.15 *A graph $G(V, E)$ is called* weighted *if there exists a mapping from the set of edges E into the set of real numbers $w : E \mapsto \mathbf{R}$. $w(e_i)$ is then called* weight *of the edge $e_i \in E$.*

The *weight* of edges can be interpreted in various ways, e.g. as distance or time in the shortest–path, minimum spanning tree, and travelling–salesman problems, or as capacity in network flow problems, or as cost in any of the above–mentioned problems. It is also acceptable to assign negative weights to the edges: for example, in network flow problems, *negative* flow through an edge means *positive* flow in the opposite direction.

A difficulty arises in the shortest–path problems, where the "length" of the path defined as the number of edges on the path can be confused with the total length of the path defined as the sum of the lengths (weights) of the individual edges. It should always be clear from the context which of the two lengths is being discussed.

2.1.3 Description of graphs

For the purposes of algorithms, it is necessary to store the data about the graphs in some reasonable data structures. There are various possibilities of describing graphs. In the following, the description of graphs through adjacency lists and matrices will be presented.[9]

Adjacency and distance matrices

The *adjacency matrix* of a graph is a square matrix with the number of rows (and also the number of columns) equal to the number of vertices of the given graph. The elements of adjacency matrix $A = \{a_{ij}\}$ of the graph $G(V, E)$ are defined in the following way:

$$a_{ij} = \begin{cases} 1 & x_i x_j \in E \\ 0 & x_i x_j \notin E. \end{cases}$$

In this matrix, the existence of *loops* (edges $v_i v_i$) in the graph is recorded on the main diagonal. On the other hand, *parallel edges* (that are allowed in some kinds of graphs, but not used in this thesis)[10] cannot

[9] *Incidence matrices* can also be used. However, as those do not appear in the algorithms described, this kind of matrix is not presented here.

[10] By "parallel edges" we mean edges with identical endpoints (or, in directed graphs, with identical starting point and end point). If there are parallel edges, integer numbers can be used to denote the number of edges incident with the two vertices.

be recorded in this version of adjacency matrix. This version is therefore ideal for *unweighted* graphs without parallel edges. For undirected graphs, the adjacency matrix is, moreover, *symmetric*. For directed graphs, positive and negative numbers are sometimes used to determine whether the edge is going out of the vertex or into it.

If the graph is *weighted*, a modification of the adjacency matrix is used — the *Distance Matrix* $D = \{d_{ij}\}$, whose elements are

$$d_{ij} = \begin{cases} d_{ij} & x_i x_j \in E, w(x_i x_j) = d_{ij} \\ \infty & x_i x_j \notin E. \end{cases},$$

where $w(x_i x_j)$ denotes the *weight* of the edges $x_i x_j$. There can be another interpretation of the term *"distance matrix"*: sometimes, the shortest distance between two vertices is used instead of the length of the relevant edge in the *distance matrix*.[11] For graphs whose edges satisfy the triangular inequality, the two concepts of distance matrix are identical. Distance matrices are used for example in some minimum spanning tree algorithms and other discrete optimization problems.

Adjacency lists

This type of structure for describing a graph consists basically of a list of vertices, with additional information. Usually, vertices and their neighbours (i.e. vertices adjacent to them) are listed. Before describing adjacency lists for directed graphs, we need to define the notion of a *predecessor* and a *successor* of a given vertex.

Definition 2.16 *Let $G(V, E)$ be a finite directed graph, $v_i v_j$ an edge in G. Then vertex v_i is called the* predecessor *of vertex v_j and vertex v_j is called the* successor *of vertex v_i.*

When we use adjacency lists, we can see immediately to which other vertices we can proceed from the given vertex. In *undirected graphs*, we can proceed to all the vertices adjacent to the relevant vertex, whereas in *directed graphs*, we can proceed from each vertex to its successors. This distinction implies at least two different kinds of adjacency lists: one for undirected and another for directed graphs. Such lists are used in shortest–path algorithms, for example.

[11] This is the case e.g. for the travelling salesman problem, where determining shortest paths might diminish the size of the problem significantly when the graph does not conform with triangular inequality.

In the following, we will usually talk about *connected* graphs, that is graphs where one can "go" from one vertex to any other using the edges of the graph, and about *loopless* graphs, i.e. graphs not containing edges that begin and end in the same vertex (so–called *loops*).

2.2 Algorithms, Heuristics, Complexity

As the quotation below demonstrates, the notion of algorithm was not always an explicit one [Moo59, p. 285]:

> *The methods given in this paper require no foresight or ingenuity, and hence deserve to be called algorithms. They would be especially suited for use in machine, either a special-purpose or a general-purpose digital computer.*

In the 1950s, however, the notion of an algorithm as a mechanical procedure started to be the most used one. The aim of this section is to unify the terminology in this area for the purposes of this thesis.

2.2.1 Algorithms and heuristics

According to the Soviet Mathematical Encyclopedia [SME, Volume 1, p. 131], the term *algorithm* stands for

> detailed instructions defining a computational process [...], which begins with an arbitrary *input* [...], and with instructions aimed at obtaining a *result* (or *output*) which is fully determined by the input.

The phrase "the output is fully determined by the input" means that if two people (or even machines) perform the same algorithm, they arrive at exactly the same result. It is sometimes wrongly assumed that an algorithm must be finite: this is certainly not true, as some algorithms may take too long and some may even not terminate at all. Such algorithms are obviously not suitable for solving real–life problems. That is why other procedures have to be used, such as *heuristics*.

Heuristics is also a method for solving problems. The method consists of trying several procedures (or the same procedure for several times) and, after each given course of operations, questioning whether or not we got any nearer to the optimum solution. Heuristics may also end after a given number of steps, divulging several possible results (the

best one need not necessarily be included) and leaving the possibility of choosing one of them up to the user.[12]

The crucial difference between algorithm and heuristics — not universally accepted, though — is the stopping criterion. Unlike algorithm, heuristics can stop after giving a couple of feasible answers to the problem, without arriving at a unique solution, or when the improvement from the last result is not significant. In the case of an algorithm — but not heuristics — we also prove *correctness* of the procedure, i.e. that the algorithm really gives the desired solution. In the following chapters, the term "heuristics" will denote the method of solving problems that gives a "good", but not necessarily the best, solution in a "reasonable" time. The term "algorithm", on the other hand, will denote the procedure that determines the optimum solution, without considering the time criterion.

2.2.2 Complexity

When judging efficiency of algorithms or heuristics, we are interested in the *time* it consumes and, in the case of heuristics, in the "degree of optimality" of the solution. Various measures can be used for the description of the time consumption. However, as nowadays these methods are intended mostly for digital computers, some universal method has to be employed.

At present, *asymptotic complexity* of the algorithm (heuristics) is a commonly given measure of its efficiency. It is based on the number of basic operations (steps) that need to be performed — i.e. the number of multiplications and/or additions. It is generally assumed that algorithms which require a *polynomial* number of steps are quite good — the complexity of such algorithms is something like $O(n^k)$, where n determines the size of the input and k is a finite number.[13] The trouble with some problems, such as the travelling-salesman problem, is that no polynomial algorithm for them is known yet (and most probably does not exist). That is why we have to retreat to inexact heuristics, which nevertheless work in "reasonable" time.

[12] The notion of heuristics seems to be even less definite than that of algorithm.

[13] A lot has been written on the subject of usefulness of this division of algorithms into the classes of polynomial and non–polynomial algorithms, especially as k can be very large and the algorithm, therefore, rather slow. The subject is treated e.g. in [GJ79]. More on the history of the notions of algorithm and complexity can be found in Chapter 1.

Solving problems by algorithms

In discrete mathematics, one way of solving the problem is always ready at hand: the enumeration of all the possibilities. However, this is usually ineffective when dealing with problems of large size. Computers could help, but even they are not almighty. In other words, the use of computers is, in itself, not the solution. In order to help computers arrive to the solutions in "reasonable" time, algorithms are employed.

In colloquial language, one could describe an algorithm as "transparent", when one knows why the steps are taken in this or that particular order. That is not always the case, for sometimes the algorithm gives the correct result by using a complicated procedure. This "obscure" algorithm might, however, be faster than the "transparent" one. To make it even more down–to–earth, sometimes the decisions that seem to be good in the short term are not good in the long term.

However, there is a special class of algorithms, the so-called *greedy algorithms* which are notable for their special feature: although they use "greedy strategy", i.e. the algorithm always chooses the *locally optimal* solution, they solve the problem successfully. "Greedy" algorithms make use of the special property of the problem: good short– and long–term decisions are the same. These algorithms behave like a person in a maze: they can see the corridors, but have no access to the plan of the maze. This procedure is used in searching a graph (*breadth–first search, depth–first search*) or for the *minimum spanning tree*.

Comparing the number of possibilities in a discrete–optimization problem and the time needed for solving these problems with the help of algorithm can serve as an example of how helpful an algorithm can be. Table 2.1 shows the numbers of possible solutions of some discrete optimization problems and the complexity of the best known algorithms that can be used to solve the particular problem. The numbers of vertices $|V|$ and edges $|E|$ of the graph $G(V, E)$ are denoted by n and m, respectively.

Algorithms can be divided into two basic classes: *polynomial* and *non–polynomial*. As was said before, the time consumed by polynomial algorithms grows polynomially with the size of the input, while the time consumed by non-polynomial algorithms grows exponentially. The polynomial algorithms allow us to obtain better results by their implementation, using various data structures (*heaps, Fibonacci heaps*, etc.). Finally, algorithms may also differ in the speed of "updating", i.e. how fast they find the new solution if new vertices or edges are added.

Table 2.1: Efficiency of algorithmic solutions

Discrete Optimization Problem	Number of Possible Solutions	Complexity of the Best Algorithm
ShP	$(n-2)!$	$O(n^2)$ (non–negative lengths) $O(nm)$ (negative lengths allowed)
MST	n^{n-2}	$O(nlogn)$
TSP	$\frac{(n-1)!}{2}$	NP

Legend:

ShP ... Shortest Paths
(complete graph, fixed source and end)
MST ... Minimum Spanning Tree
(in a complete graph)
TSP ... Travelling Salesman Problem
(complete undirected graph)
NP ... Polynomial solution not known

Chapter 3

The Shortest Path Problems: Crystalization of Ideas and Algorithms

Finding a shortest path in a graph — either directed or undirected — is a necessary prerequisite for some other discrete optimization problems, such as network flows or the travelling–salesman problem.[1]

In the early articles, shortest paths algorithms are usually described only on specific examples of routes from one vertex to another. The algorithmic notation comes only later. In Section 3.3, examples of the early description of algorithms are given — those of Dantzig, Moore (including the improvement due to D'Esopo), and Minty (all 1957). These are followed by a sample of two methods (Section 3.4.2) solving the same problem while using techniques which are far apart: that of Bellman, 1958, and another one by Ford, 1956. In a paper by Dijkstra, 1959, described in Section 3.4.3, the shortest path problem is already recognized as a graph–theoretical problem.

The level of rigour in the description of shortest path algorithms varies not only in time; notable differences can also be found between the individual authors. The algorithms of Moore and Dijkstra, published within only two years, can serve as an example. Differences can be seen in the context in which the authors published their solution: for example, the articles by Moore [Moo59] and Minty [Min57] are written in a

[1] "Shortest path" problem is not the only term used for this problem: some authors call it also the "shortest route" or "shortest chain" problem.

much more colloquial language than the articles by DIJKSTRA [Dij59a] or DANTZIG [Dan57].

The way algorithms are reported upon and re–told by other mathematicians is also examined in this chapter. During this process, the algorithms are often adapted. The transition from the "old" to the "new" descriptions of algorithms is shown on the examples of MOORE's and DIJKSTRA's algorithms.

Having found more or less suitable solutions to shortest path problems, mathematicians focused on the aspect of computation in a more sophisticated way. This includes the use of the decomposition principle as well as of data structures, implementing the algorithms as well as upgrading the solution after some vertices and edges have been added.

3.1 Specific Background

Before describing specific approaches to the problem by different mathematicians, basic concepts used in the algorithms are explained. First of all, we need to distinguish between two kinds of length of a path: the length of a path in an *unweighted* graph and the length of a path in a *weighted* graph.[2] As the problem is referred to as the "shortest path" problem, it seems inappropriate to use something else than "length" for the sum of the weights of the edges constituting the path. It should always be clear from the context whether the length is related to the *number of edges* of the path, or whether it is related to the *sum of the lengths (i.e. weights) of the constituent edges.*

Definition 3.1 *The* length *of the path* $p = v_0, e_1, v_1, \ldots, e_n, v_n$ *is the sum of the weights of its constituent edges. A* shortest path *from vertex* u *to vertex* v *is defined as any path* p *with the minimum length of all paths between* u *and* v*.*

In some algorithms, *shortest route trees*, defined below, are used:

Definition 3.2 *A* shortest–route tree *is a tree which specifies a unique path from the origin vertex to each of the other vertices in the graph.*

The definition above is given by POLLACK and WIEBENSON in [PW60].[3] The vertices are the same as in the given graph, but not all of the edges of the original graph are included in the *shortest–route tree*. The term

[2] The notions of "path" and "simple path" are defined in Chapter 2.

[3] In the original, the authors use the term "node" instead of "vertex".

"shortest–route tree" was probably first used by DANTZIG in 1957, for in MINTY's *A comment on the shortest route problem* [Min57], we read: "Dantzig's 'shortest–route tree' can be found [in the following way]." It is apparent that shortest–route trees differ according to their starting vertices (roots).

Shortest path problems can be divided into the following four basic subproblems:

1. *Single–source shortest path problem:* Find the shortest path(s) from one source to all other vertices.
2. *Single–destination shortest path problem:* Find the shortest path(s) to a single destination from all other vertices.
3. *Single–pair shortest path problem:* Find the shortest path(s) between a given pair of vertices (source–destination).
4. *All–pairs shortest path problem:* Find the shortest path(s) from vertex u to vertex v for all pairs of vertices u, v.

It is interesting to note that no algorithm for the *single–pair* shortest path problem running asymptotically faster than algorithms for *single–source* shortest paths is known. For example, the algorithm of DIJKSTRA (1959) for the single–pair shortest path finds, as a by–product, the shortest paths from the source to all the other vertices.

Let us end this introductory section by the words of D. B. JOHNSON [Joh77, p. 1]:

> Finding shortest paths in networks is a fundamental problem in combinatorial optimization. The best routings of vehicles or messages, for instance, are shortest paths with respect to costs associated with the arcs of the transportation or communications network involved. Finding shortest paths is a subproblem in many optimization problems such as finding optimal flows in networks.

3.2 Origins and Development of Shortest Path Problems

From the historical point of view, finding shortest paths in graphs is connected to finding a way through a labyrinth — which is by no means a new problem.[4] However, since the standard procedures for going

[4] It was solved e.g. by Tarry in 1895 [Tar95].

through a maze were formally described rather recently, it is rather difficult to say when exactly shortest path problems originated. According to E. F. MOORE, shortest path problems were first solved by himself and C. E. SHANNON[Moo59]:

> The problem was first solved in connection with CLAUDE SHANNON's maze-solving machine. When this machine was used with a maze which had more than one solution, a visitor asked why it had not been built to always find the shortest path. Shannon and I each attempted to find economical methods of doing this by machine. He found several methods suitable for analogue computation, and I obtained these algorithms. Months later the applicability of these ideas to practical problems in communication and transportation systems was suggested.

MOORE might be right in his claim that looking for standardized procedures in shortest path problems is a twentieth–century problem. He seems to be one of those who care about the history of the problem and the consequences of its solution, yet his seemingly perfect account of the history of the problem leaves out some of the early solutions: for example, BELLMAN, COOKE, and LOCKETT in [BCL70] refer to a 1954 solution of the problem by SHIMBEL [Shi55], and POLLACK and WIEBENSON in their review article [PW60] claim that the first shortest path algorithm using the shortest–route tree (i.e. belonging to the same category as MOORE's algorithm) is due to DANTZIG. The claims made in [PW60] are not as contradictory to the following statement in [BCL70, p. 99] as might at first appear:

> A labelling algorithm essentially equivalent to the one we have presented [...] was given by E. F. Moore, and is perhaps the earliest algorithm of this type [Moo59].

BELLMAN et al. are talking about *labelling process*, but not directly about the methods based on the use of shortest–route tree. However, even this shows that in mathematical papers, the credit given tfor solving a problem often seems to be assigned at random.

In BERGE's monograph [Ber58] we find allusions to much earlier papers by TARRY and TRÉMAUX, both concerned with finding some special kind of path in a graph.

Most mathematicians do not pretend to care much about the history of the problems they are solving. For example, E. W. DIJKSTRA gives

very few clues about the history of the problems in his 1959 article on the *minimum spanning tree* and the *shortest path* problems.[5] However, they mostly believe either that they are the first one to solve the problem, or that their solution is better than any of the previous ones. On the other hand, it is worth noticing that the first solutions of the problem were found independently by several mathematicians in the 1950s. These were the matrix solution by SHIMBEL, the shortest–route tree solution by G. B. DANTZIG, and the labelling algorithm by E. F. MOORE.

3.3 Early Shortest Path Algorithms

3.3.1 Shimbel, 1954

SHIMBEL uses operations with a matrix describing a graph in his solution. His contribution was presented at the *Symposium on Information Networks* in April 1954 and published in the *Proceedings* in 1955.[6] On his solution of the shortest route problem, we can read in [BCL70, p. 99]:

> One of the earliest papers to give a clear statement of the shortest route problem was [Shi55]. Shimbel gave a solution based on operations with what we have called the matrix of a graph.

The same paper is also quoted in [PW60] and the authors claim that SHIMBEL's and BELLMAN's solutions are essentially equivalent [PW60, pp. 227–228]. In the quotation below, the distance matrix of the graph need not be symmetric.

> Let the distance from node j to k be represented by an element of the matrix $\mathbf{D}^1$, where $j, k = 1, 2, \ldots, N$, and $d^1_{jj} = 0$ for all values of j. An element of the matrix $\mathbf{D}^2$ is defined as
>
> $$d^2_{jk} = min_l(d^1_{jl} + d^1_{lk}), (l = 1, 2, \ldots, N) \qquad (3.1)$$
>
> so that d^2_{jk} is the length of the minimum route between j and k by means of a one - or two-link path. In general,

[5] Dijkstra is not to be criticised for this approach. It happens quite often that mathematicians re–invent solutions and this fact seems to be an inherent part of mathematics and the communication between mathematicians. The answer to the question *why* mathematicians do not look so much for earlier solutions could perhaps be found in philosophy of mathematics.

[6] The *Symposium on Information Networks* took place from 12th to 14th of April 1954 at the Polytechnic Institute in Brooklyn.

the matrix $\mathbf{D}^i = ||d^i_{jk}||$ consists of elements representing the shortest–route path between each j and k from among all possible paths consisting of from one through i links, where

$$d^i_{jk} = min_{a,b,\ldots,i-1}(d^1_{ja} + d^1_{ab} + \cdots + d^1_{i-1,k}), \qquad (3.2)$$
$$(a, b, \ldots, i-1 = 1, 2, \ldots, N).$$

The process terminates either when $\mathbf{D}^{m-1} = \mathbf{D}^m$ or when $m + 1 = N - 1$ since no minimal path can have more than $N - 1$ links.

The Bellman–Ford algorithm, to which SHIMBEL's solution is equivalent, can be used even for graphs with negative edge–weights. It is surprising that one of the earliest shortest path algorithms published is not restricted to positive weights. However, (and this seems to be typical of early graph algorithms), the case of negative edge–weights is not discussed; moreover, it is not even mentioned that negative weights could present a problem.

3.3.2 G. B. Dantzig, 1957

As far as mathematical and technical papers on the subject of shortest paths are concerned, the results presented in the article *Discrete-variable extremum problems* [Dan57] by G. B. DANTZIG present the first solution of the shortest path problem using the shortest–route tree. Dantzig's article was published in April 1957 in the *Operations Research*, MOORE described his algorithm at a symposium held on 2nd to 5th April 1957, which means that both algorithms were probably discovered independently. On the other hand, in a footnote of his paper, DANTZIG says "in summer, 1955, meeting of ORSA at Los Angeles where this paper was first presented, ...", which suggests Dantzig's priority. It is, however, not clear whether the whole or only a part of the paper in *Operations Research* was presented at the ORSA meeting, as DANTZIG gives two or three references which might have been published later than in "summer, 1955". It is probable that changes made between "summer 1955" and the publication in 1957 were only slight. POLLACK and WIEBENSON attribute the priority of solution to DANTZIG; however, they are not satisfied with the computational results of the procedures given [PW60]:

> The algorithm given here is based on the simplex method. An arbitrary tree is chosen, edges are added and then deleted. By this process we get the shortest–route tree. This algorithm is rather inefficient, but was the first one.

Dantzig himself says in the abstract [Dan57, p. 266]:

> This paper reviews some recent successes in the use of linear programming methods for the solution of discrete–variable problems. One example of the use of the multistage approach of dynamic programming for this purpose is also described.

In his article, George B. Dantzig describes two other discrete–optimization problems: the marriage problem and the knapsack problem. On his methods, he says [Dan57, pp. 266–267, emphasis mine]:

> The purpose of this paper is to outline an approach that, we believe, has a *high probability* of verifying whether or not, in any particular numerical case, an optimum combination has been selected. The human mind seems to have a remarkable facility for scanning many combinations and arriving at what *appears* to be either the best one or a very good one. [...] The technique presented here is not foolproof and cannot be guaranteed to work in all cases.

For the shortest route problem, Dantzig chooses a description of a concrete example: the shortest route for a package from Los Angeles to Boston. He uses a set of linear equations for the solution. The starting point for any solution is a "shortest–route tree", which is then improved by testing triangle inequality for the individual vertices.

3.3.3 Moore's algorithm, 1957

In 1957, Edward F. Moore, a "Member of the Technical Staff, Bell Telephone Laboratories, Incorporated" presented four different solutions of the shortest path problem at the *International Conference on the Theory of Switching*. His motivation for solving this problem was the so-called "routing of telephone traffic," especially applicable if traffic is congested and finding an alternative route is necessary.[7] Moore also suggested another area where the problems involving finding shortest paths arise — transportation: routing one or more airplanes / trains so as to arrive at the destination as early as possible. His methods are in a certain way general — with his own words [Moo59, p. 292], "all

[7] Routing of telephone traffic is dealt with also by other authors using various methods: for example R. I. Wilkinson [Wil56] uses statistical methods. The aim is to provide the best connections possible in a given network, at a given time and situation (e.g., when some lines are blocked and the usual path cannot be used).

four algorithms given in this paper are applicable to mazes having paths which can be traversed in only one direction, such as one–way streets."

In the introduction to the paper, MOORE says [Moo59, emphasis mine]:

> Suppose that it is desired to find the shortest path through a maze. For a very simple maze this can be done just by observing it and trying a few paths that look reasonable. *But this trial-and-error procedure does not apply to a complicated case.*

He touches here upon the fundamental characteristics of algorithmic solutions: it *may* be faster to solve problems by intuition, but it may not work for large and/or complicated cases. Furthermore, "the methods [given in this paper] would be especially suited for use in machine, either a special-purpose or a general-purpose digital computer". [Moo59]

However, MOORE describes a 'pen-and-pencil' solution. What is more, the first three out of the four algorithms in the paper [Moo59], denoted as 'Algorithm A', 'B', 'C', and 'D', are designed for graphs with unweighted edges or for edges with their weights equal to one; only the last algorithm is designed for weighted graphs. In the first three examples, finding the shortest path can thus be understood as finding a path through as few vertices as possible.

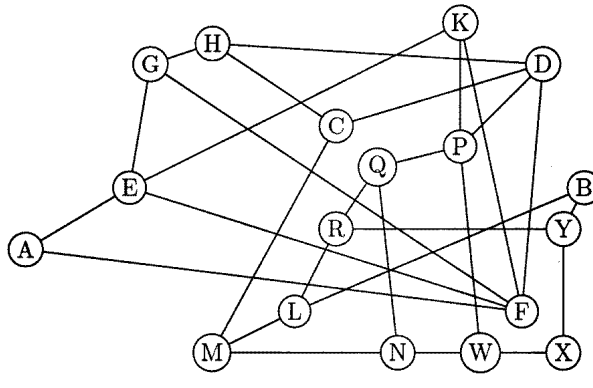


Figure 3.1: Moore's "Maze"

MOORE demonstrates his algorithms on an example: a graph with 18 vertices labeled by letters of the alphabet (See Figure 3.1). Letter A denotes the source, letter B the destination, and "[...] it is desired to

place a call from city A to city B." A very complicated picture of the situation is also given, maybe to advocate the methods presented in the paper.

The basic algorithm, *Algorithm A*, the first one described and probably the most comprehensible one, is described in the following way [Moo59, p. 286]:

> On the *first step*, we shall write the number 0 on city A. [...] On the *second step*, we shall write the number 1 on the cities E and F; on the *third step*, write the number 2 on cities D, G, and K. On each step the next larger number will be written on all cities which have not yet been written on, but which are adjacent to cities which have.

Reading the instructions above, one must refer to the picture quite often. However, although MOORE demonstrates his method on a concrete example, it would be wrong to suspect him of not knowing exactly what he is doing, for he continues [Moo59, p. 286]:

> It can easily be proved that the number being written will always be the length of the shortest path from city A to the city on which the number is written.[8]

When the 'destination' (vertex B) is labeled, we know how long the path between A and B is; what remains is the task of determining the vertices through which this shortest path goes. This is done by drawing arrows from the 'destination' until the 'source' is reached. MOORE presents a general rule for the drawing of arrows [Moo59, p. 287]:

> At each step, we now draw an arrow (in a telephone mechanization of this we might at this point actually be setting up the connection over which talking will take place) from the city last reached to any city numbered one lower, that is, any city one step nearer to the beginning point. On the *tenth* step [...]

This, however, is only done after he starts describing the procedure on the specific example, without giving any reason prior to carrying out the following two steps [Moo59, p.286]:

[8] Here, the term "length of the shortest path" refers to the number of edges lying on the path.

> "On the *eighth* step draw an arrow from B to L; on the *ninth* step draw an arrow from L to M."

MOORE further notes the following on the length of the designed procedure [Moo59, p. 287]:

> "In general, if the length of the shortest path is n, algorithm A will require $2n + 2$ steps."

And MOORE does this in spite of the fact that the algorithm is *described* on a *specific example*! Again, it is apparent that MOORE was perfectly aware of what he was doing, only he did not have the necessary language for the description of the algorithm. Nowadays, *Algorithm A*, in its "computerized" form, is usually referred to as the *breadth–first search*[9] and is commonly described by so-called "pseudo–code", a commonly used code resembling a programming language [CLR91]:

```
BFS (G, s)
1   for each vertex u ∈ V[G] − {s}
2       do color[u] ← WHITE
3          d[u] ← ∞
4          π[u] ← NIL
5   color[s] ← GRAY
6   d[s] ← 0
7   π[s] ← NIL
8   Q ← {s}
9   while Q ≠ ∅
10      do u ← head[Q]
11         for each v ∈ Adj[u]
12            do if color[v] = WHITE
13               then color[v] ← GRAY
14                    d[v] ← d[u] + 1
15                    π[v] ← u
16                    ENQUEUE (Q, v)
17         DEQUEUE (Q)
18         color[u] ← BLACK
```

Algorithm B differs from the *Algorithm A* only slightly: instead of all integers, only the numbers 0, 1, and 2 are used, the motivation here

[9] A procedure in a certain sense opposite to the *breadth–first search* (BFS) is the *depth–first search* (DFS) procedure. The latter was published already in 1895 in a paper by G. Tarry [Tar95]. *Depth–first search* is used e.g. for finding a way through a labyrinth.

being especially the reduction of the storage requirements of a computer. Description of at least three different states of the vertex is necessary, as we must be able to recognise uniquely which vertex is the predecessor and which is the successor. To start with, the vertex mark is 'blank'; the source vertex A is labeled with 0, the adjacent vertices with 1, the vertices adjacent to these with 2, the next level of vertices with 0, and so on, until the destination is reached. MOORE explains that other modulos can also be used, but that 3 is the smallest of all the suitable ones, since [Moo59, p. 287]

> "each city must have an integral label which is equal to, one less than, or one more than the label of any adjacent city. Integers mod(3) are sufficient to distinguish among these three cases, but the integers mod(2) are not".

As a result of this improvement, *Algorithm B* uses less memory than *Algorithm A*: only two bits for each city. The rest of the procedure, and hence also the number of steps required, are the same.

Algorithm C uses the *breadth–first search* repetitively. Instead of performing the breadth–first search until we reach the end vertex (vertex B) and marking the vertices with an increasing sequence of numbers, we perform the algorithm several times while using only zeroes and ones to mark the nodes. When we reach a vertex adjacent to the end vertex, we draw an arrow from the end vertex to the previous one. For the first time, the breadth-first search is performed n times, for the second time $(n-1)$ times, and so on until we reach the starting vertex.

Algorithm C requires only one bit of memory for each city. First, all nodes are assigned the value '0'. In the next step, node A is assigned the value 1. On each subsequent step we write 1's in all those cities adjacent to cities in which 1's were written the previous time. After n steps city L is reached, and an arrow is written from B to L [the same arrow is written in the eighth step of *Algorithm A*]. Next, all the 1's are changed to 0's and the whole procedure of writing 1's is repeated, until the vertex L (the penultimate vertex of the path from A to B, in this specific example) is reached. The procedure continues in the same manner until the arrows reach vertex A. Algorithm C, however, requires more time than the previous two: $\frac{1}{2}(n+1)(n+2)$.[10]

Only the *Algorithm D* in [Moo59] describes a solution for weighted graphs. This is again done on an example. This algorithm is a modi-

[10]It should be noted that n stands for the length of the shortest path here, not for the number of vertices.

fication of *Algorithm A* for weighted graphs. In unweighted graphs, we always add one to the mark of the previous vertex; in weighted graphs, we add the weight of the edge incident with both vertices.

To conclude with, MOORE says [Moo59, pp. 290–291]:

> These four algorithms do not exhaust all of the variations on this method of solving mazes. As a further example, which will only be described roughly (and hence does not deserve the name algorithm), suppose that you are in a certain city at a certain time and wish to arrive at your destination by train as soon as possible. Then by consulting your timetable you can write down the earliest time at which you can arrive at each city which it is possible for you to reach without changing trains. Then, on the next step, you can write down the earliest time at which you can reach each city which can be reached by changing trains only once. It will be necessary to take assumptions about how long it takes to change trains in each city. You can continue this process, always keeping track of the earliest time that you have found for reaching each city, until you reach a point where none of these times can be improved. Then you pick up the route you could follow.

MOORE's solutions are described in a very straightforward way, while maintaining a crucial aspect of an algorithmic solution, namely that of not leaving out any possibility. His solution carries the basic idea that has been improved by other mathematicians and now bears the author's name.

Improvement by D'Esopo

"Another efficient algorithm given by MOORE is particularly applicable to road networks. The use of an index number, resulting in a more efficient algorithm, is suggested by D'ESOPO." say POLLACK and WIEBENSON in [PW60, p. 225]. This improvement is described in the following section of their paper [PW60, pp. 225–226]:[11]

> As before, a network is given and all links are labeled with their lengths. An unsymmetric distance matrix is allowed

[11] Pollack and Wiebenson must have learnt about this improvement some time before 1960, when their article was published, as they refer to "Private communication with D'Esopo" in their references.

and the same convention of one-way streets is used.[12] The starting and terminating cities are labeled A and B. In the algorithm, two numbers are associated with each node. One is a variable 'label' number, the other is a fixed and unique 'index' number. At any given stage in the application of the algorithm, the label number represents the shortest determined distance to a node. The index numbers are the positive integers that are assigned in ascending order to each new node as it is considered. Finally, a control register is maintained in which can be found the index number of a node, the CR-node.

The improved algorithm is described below and resembles very much the algorithm due to DIJKSTRA published in [Dij59a] and described in Section 3.4.3. The procedure is based on the use of the so-called *CR-node*. The number in the control register (abbreviated as"CR") determines which node is being examined at each step.

1. Assign to city A the index 1 and the label 'zero'. Enter the number 1 in the control register. Go to **2.**

2. Locate all nodes connected to the CR-node. To each of these nodes that do not already have index numbers, assign the next available integer as its index. Go to **3.**

3. For each of these nodes, form the sum of the CR-node label and the street length to the connected node. For each connected node there will be three possibilities. *(a)* If the connected node does not have a label, assign this sum as its label and place a check-mark next to the street. Proceed to the next connected node or, if there are no more connected nodes to be examined, go to **4.** *(b)* If the connected node has a previously assigned label that is lower in value than the formed sum do nothing to this node. Proceed to the next connected node or, if there are no more connected nodes to be examined, go to **4.** *(c)* If the connected node has a previously assigned label that is larger than the new sum, assign the new sum as the label. Also place a check-mark beside the new street and erase the check-mark that is next to the street

[12]Streets are represented by a couple of one-way streets if they can be traversed in both directions. The distance in one direction may be different from the distance in the other direction. (Footnote mine.)

associated with the replaced label. Now examine the index of this relabeled node. If the index is higher than the index of the CR-node, proceed to the next connected node, or if none remains, go to **4.** If the index is lower than the index of the CR-node, note the value of this lower index number. Proceed to the next connected node or, if none remains, go to **4.**

4. If there are any lower index numbers noted in step **3***(c)*, select the lowest one and place it in the control register. If not, add 1 to the number in the control register. In either case go to **2.**

The index number is also used to determine when the algorithm should stop [PW60, p. 226]:

> The process terminates when the index number in the control register is one higher than the number of nodes in the network. The procedure for finding the shortest path to any node is the same as that given in the preceding algorithm.

The above quotation also states an important result, namely that for determining the shortest paths from one node to all nodes, we use exactly the same procedure as for determining the shortest path between two specific nodes.

Moreover, it is worth noticing that

1. the description of an algorithm in [PW60] is much more visually structured than the one given in [Moo59]

2. POLLACK and WIEBENSON give D'ESOPO's improvement of MOORE's algorithm for *weighted graphs*

Moore's algorithm for weighted graphs

MOORE's algorithm D for weighted graphs is probably the most known nowadays. It is quoted in contemporary textbooks on graph–theoretical algorithms, but it is actually adapted for contemporary users. An example can be found in a book by the Slovak mathematician JÁN PLESNÍK [Ple83, p. 116]:

> **STEP 0:** Vertex s is marked $(0, 0)$, other vertices are without marks, $k := 0, V_0 := \{s\}$.

Table 3.1: Moore: assessment of his own algorithms

	Number of steps	Storage requirements
Algorithm A	$(2n+2)$	arbitrary integer
Algorithm B	$(2n+2)$	two bits
Algorithm C	$\frac{1}{2}(n+1)(n+2)$	one bit

STEP 1: For every $i \in V_k$ we perform:
$V_{k+1} := \emptyset$. We mark every unmarked successor j of vertex i: $(p_j, d_j) := (i; k+1), V_{k+1} := V_{k+1} \cup \{j\}$

STEP 2: If $V_{k+1} = \emptyset$, then STOP. Else $k := k+1$ and return to **STEP 1.**

The idea of the algorithm given above is really due to MOORE; it has, however, become necessary to describe algorithms in a more precise way. PLESNÍK therefore adapted MOORE's algorithm and provides such a description of the process, that it is ready for immediate transformation into a computer programme.[13]

An interesting part of MOORE's paper is the discussion on the complexity and storage requirements. Although he presents his method on a concrete example with given weights, he gives the following estimates about the storage requirements and number of steps needed to arrive at the final solution.

Note: The number of steps is given for a path of length n; storage requirements determine how much memory per city is needed: the phrase 'arbitrary integer' means that the computer must be capable of remembering any integer, up to the length of the shortest path, for every city.

The figures shown in Table 3.1 may be compared with the data PLESNÍK gives about the complexity of Algorithm A: for a graph determined by successors of the vertices, the time necessary for finding a path between the starting vertex s and the end vertex t is $O(m)$, for finding the shortest path between s and t it is $O(m+n)$, and finally for finding the shortest paths from s to all other vertices is $O(mn+n^2)$, where n is the number of vertices and m is the number of edges. When comparing the figures given by MOORE and PLESNÍK, however, one has to bear in

[13] An analogy can be seen even with the methods used in Egyptian papyri: the Egyptians probably had no idea of equations, but was their thinking really that much different from ours?

mind that n is once the length of the shortest path and once the number of vertices.

3.4 Different Approaches

Nowadays, one tends to associate the shortest path problems with graph theory. In the 1950s, however, it was a problem that needed to be solved and was solved by a variety of means, which becomes apparent in this section.

3.4.1 Some picturesque solutions

The following two methods use neither matrices, nor graph terminology, nor functional equations. Yet, they provide a nice insight into the process of the formation of algorithms. The first method seems to be more of a joke; the second one, however, can be transformed into a "serious" method.

Minty's "string model," 1957

In the same year when DANTZIG's article was published (1957), a short and illustrative reply came from MINTY. It seems unnecessary to comment on this article [Min57]:

> The shortest route problem (discussed by DANTZIG, *Operations Research*, vol. 5, p. 270, April, 1957) can be solved very simply in the case where the distance matrix is symmetrical, as follows: Build a string model of the travel network, where knots represent cities and string lengths represent distances (or costs). Seize the knot 'Los Angeles' in your left hand and the knot 'Boston' in your right hand and pull them apart. If the model becomes entangled, have an assistant untie and re-tie knots until the entanglement is resolved. Eventually one or more paths will stretch tight — they then are the alternative shortest routes.
>
> Dantzig's "shortest–route tree" can be found in this model by weighting the knots and picking up the model by the knot 'Los Angeles'.
>
> It is well to label the knots since after one or two uses of the model their identities are easily confused.

This is perhaps the article on shortest paths written in the most colloquial language. Yet it includes all the necessary information and the author suggests a procedure for determining the 'shortest–route tree', a kind of 'by–product' of finding shortest paths.

Rapaport and Abramson, 1959

A quite different method is used by RAPAPORT and ABRAMSON. They do not use a string model like MINTY, they make use of electrical energy, timers, and lights. Here is the description of their method as given by POLLACK and WIEBENSON in [PW60]:

> RAPAPORT and ABRAMSON describe an analog device with variable electric timers used to stimulate link distance. The initial and final nodes are chosen and a master clock is started. When the initial node is energized, all timers outgoing from it are started. This process continues until either the final node is reached or when all nodes are reached. All timers that energize a node are distinguished with a light. If a timer reaches a node that is already energized the timer does not go on. The links composing the shortest–route tree thus consist of all the lighted links. The particular analog described in reference [RA59] will only work for the case where the distance matrix is symmetric. However, this type of analog can easily be converted to accommodate the nonsymmetric case if two 'one-way' timers are used between each pair of nodes. Note that the digital version of this 'timer' process is essentially the same as the algorithm given by Minty.[14] Rapaport and Abramson propose to use electronic timers in a later model, with the possibility of using this device for real time control.

From the above said, it is evident that in the 1950s, mathematicians used virtually all kinds of methods for finding shortest paths in a graph. The following section informs about some other, more 'mathematical' approaches: the *Bellman–Ford algorithm*, working with a minor restriction on the weight function, and the *Dijkstra algorithm*, working only for graphs with positive weights of edges.

[14]The algorithm Pollack and Wiebenson refer to is Minty's algorithm obtained in "private communication". It is described in Section 3.4.3 of this thesis.

3.4.2 The Bellman–Ford algorithm

The so–called *Bellman–Ford algorithm* was named after two mathematicians, RICHARD BELLMAN and LESTER R. FORD, who formulated the algorithms in quite different contexts. The basic idea of their algorithm(s) is that of examining first the paths of length one, then the paths of length two, etc., that is paths consisting of one edge, then two edges, etc. (By "length" of the path, the number of edges is meant here.) Although the algorithm is commonly attributed to BELLMAN and FORD, it was probably first formulated by A. SHIMBEL in 1954 (see Section 3.3.1 for details).[15] FORD's [16] algorithm was discovered as a part of the network flow theory, in 1956, BELLMAN's algorithm employs the dynamic programming technique and was first published in 1958 [Bel58].

Ford, 1956 and Ford & Fulkerson, 1962

FORD's algorithm was first published in the RAND Corporation Paper P–923 and is reproduced also in the book on network flows [FF62]. The *shortest chain algorithm* formulated by FORD and FULKERSON in [FF62, p. 131] runs as follows:

> (1) Start by assigning all nodes labels of the form $[-, \pi(x)]$, where $\pi(s) = 0, \pi(x) = \infty$ for $x \neq s$.
>
> (2) Search for an arc (x, y) such that $\pi(x) + a(x, y) < \pi(y)$. (Here $\infty + a = \infty$.) If such an arc is found, change the label on node y to $[x, \pi(x)+a(x, y)]$, and repeat. (That is, the new $\pi(y)$ is $\pi(x) + a(x, y)$). If no such arc is found, terminate.

The interesting feature of this method is that FORD and FULKERSON make use of network flow algorithm here. The above–mentioned algorithm is an adaptation of the minimal–cost network flow problem [FF62, p. 130, emphasis mine]:

> A special minimal cost flow problem having independent interest is that of finding a minimal cost (or shortest) chain from one node to another in a network in which each arc

[15] This confirms the rule (with many exceptions) that mathematical theorems and the like do not bear the name of their inventor. Shimbel's solution is reported by Bellman et al. in [BCL70], by Pollack and Wiebenson [PW60], and by Dreyfus [Dre69].

[16] For one of the versions of his shortest path or chain algorithms, see below.

> (x, y) has associated cost (or length) $a(x, y) \geq 0$. While this is a purely combinatorial problem, it may also be considered as a minimal cost flow problem by placing unit supply at the first node (the source) and unit demand at the second node (the sink), taking arc capacities infinite, and asking for a feasible flow that minimizes cost. Since the algorithm of §3 constructs an integral flow, it solves the shortest chain problem. *In other words, the first unit of flow constructed by the algorithm travels by a least cost chain.*

Thus FORD and FULKERSON suggest also another aspect of discrete optimization problems: it is often the *model* we work with, and the model can be interpreted in various ways. Here, the interpretation of the weight of the edge is either cost (minimal–cost flow problem), or length (shortest path problem).

Bellman, 1958

In [Bel58], BELLMAN describes the shortest route algorithm in something we might in graph terminology call a *complete directed graph* with n vertices. He introduces a *search technique* on the basis of the functional equation technique of dynamic programming. The distances between the individual pairs of vertices are given in a matrix $T = (t_{ij})$. Further, f_i denotes the time required to travel from i to N, $i = 1, 2, \ldots, N-1$ using an optimal policy; $f_N = 0$.

> Employing the principle of optimality, we see that f satisfy the nonlinear system of equations
>
> $f_i = Min_{j \neq i}[t_{ij} + f_j], i = 1, 2, \ldots, N-1,$
>
> $f_N = 0.$

BELLMAN continues by giving the proof that there is at most one solution of the above-mentioned system and the approximation policy in space:

> Perhaps the simplest policy we can employ[17] is to proceed directly from i to N. Define
>
> $f_i^{(0)} = t_{iN}, i = 1, 2, \ldots, N-1.$

[17] Bellman's footnote: "I owe this choice of an initial policy to F. Haight." [Bel58, p. 89]

It follows that $f_i^{(1)}$ as defined by

$f_i^{(1)} = Min_{j \neq i}[t_{ij} + f_j^{(0)}], i = 1, 2, \ldots, N-1,$

$f_N^{(1)} = 0,$

satisfies the inequality

$f_i^{(1)} \leq f_i^{(0)}.$

This inequality is immediate when we realize that $f_i^{(1)}$ represents the minimum time for a path with at most one stop.

...

It is clear from the physical interpretation of this iterative scheme that at most (N-1) iterations are required for the sequence to converge to the solution.

BELLMAN gives also a different sequence of approximations, but he notes that "it is to be expected that the first method will converge more rapidly" [Bel58].

3.4.3 Dijkstra, 1959

A year after BELLMAN, E. W. DIJKSTRA [Dij59a] publishes an algorithm in graph-theoretical language. This algorithm is faster than the *Bellman–Ford algorithm*, but works only for graphs with nonnegative edge weights. However, DIJKSTRA does not warn his readers of this fact.

The following extract can help to illustrate DIJKSTRA's style. After describing the subject he is going to deal with in his paper, he poses two problems [Dij59a, pp. 269–270]:

> We consider n points (nodes), some or all pairs of which are connected by a branch; the length of each branch is given. We restrict ourselves to the case where at least one path exists between any two nodes. We consider two problems.
>
> **Problem 1.** Construct the tree of minimum total length between the n nodes. (A tree is a graph with one and only one path between every two nodes.)
>
> [...]
>
> **Problem 2.** Find the path of minimum total length between two given nodes P and Q.

After describing the shortest path algorithm, he puts his algorithm as preferable to those of Berge and Ford, his reasons being "we need not store the data for all branches simultaneously but only for those for the branches in sets I and II, and this number is always less than n. Furthermore, the amount of work to be done *seems to be* considerably less." [Dij59a, emphasis mine] In other words, we can find some attempts at judging computational complexity here, although the judgement is not formalised yet.

Berge's monograph

Dijkstra refers to Berge's monograph [Ber58] where the tasks and solutions (algorithms) connected with the shortest paths [Ber58, Chapter 7: "Shortest path problems", pp. 75–81] are also described. Berge first formulates two tasks and gives examples:

Problem 1. Find such a path in the graph that leads from a to b.

Problem 2. Find the path between a and b that has the shortest length.

Example 1. One–person games (labyrinth as a special example of a one–person game, the "game of 15").

Example 2. The problem of wolf, goat, and cabbage.

The first of these problems is related to the *connectedness* of the graph. In fact, both of the above-mentioned examples can be understood as special cases of **Problem 1**. However, once we find out that a path between the two vertices exists, it is logical to ask for the shortest of those paths. The algorithms Berge presents in the monograph focus mainly on finding a way through a labyrinth. He describes the following algorithms:

1. *Algorithm for the first task in the planar case – labyrinth:* When at a crossing, choose always the far left (alternatively, the far right) corridor.

2. *General algorithm for the first problem:* To start with, we examine a graph representing an arborescence with the root a and we find the way going from a to some vertex b; there is the following easy way: we start in a, we follow any branch as long as it is possible,

then we return to the nearest crossing and follow another branch, etc., until we reach vertex b. The desired path from a to b will be made up of those edges that have been passed exactly once in the process of seeking.

If graph G is not an arborescence, we first determine an arborescence.

3. *Second algorithm for the first problem:* TARRY, 1895, [Tar95] suggests a method for walking through a labyrinth using the following principles: never use any edge twice in the same direction; when at a crossing x, do not choose the edge that brought you to this crossing for the first time – unless there is no other possibility.

4. *The algorithm for the second problem* is probably due to BERGE himself (as no references are given) and runs as follows: with the help of iteration method, we subsequently assign an index, equal to the length of the shortest path from a to x, to each vertex x. The description of the algorithm follows:

   1. Vertex a is assigned the index 0.

   2. If all vertices that have received the index m form a known set $A(m)$, then by assigning the index $m+1$, we receive vertices of the set

   $$A(m+1) = \{x | x \in \Gamma A(m), x \notin A(k)\}$$

   for all $k \leq m$.[18]

   3. When vertex b has been assigned a number, we stop; if $b \in A(m)$ then we look through such vertices $b_1, b_2, \ldots$ that

   $$b_1 \in A(m-1), b_1 \in \Gamma^{-1} b,$$

   $$b_2 \in A(m-2), b_2 \in \Gamma^{-1} b_1,$$

   $$\ldots$$

   $$b_m \in A(0), b_m \in \Gamma^{-1} b_{m-1}.$$

   The path $\mu = [a = b_m, b_{m-1}, \ldots, b_1, b]$ is a solution to the problem.

[18] The set $\Gamma A(m)$ is the set of all the vertices connected by exactly one edge to the vertices in $A(m)$. The second condition — $x \notin A(k)$ — excludes the vertices for which the length of the shortest path from the source is already known from the set $A(m+1)$.

Like MOORE in [Moo59], BERGE gives more algorithms for unweighted graphs (and especially for labyrinths) than for weighted graphs. Unlike MOORE, however, BERGE says little on the history of the problem.[19] After describing these basic problems and their solutions, BERGE suggests the following generalizations — and it is only now that he mentions weighted graphs [Ber58, comment in brackets mine]:

> *Problem 3.* Every edge of a given graph G is assigned a number $l(u) \geq 0$, the *length* of the edge. The path μ going from a to b with the shortest length is to be found.
>
> [Classical graph–theoretical formulation of the single–pair shortest path problem.]
>
> *Problem 4.* To each path μ, a number $f(\mu)$ is assigned. Find a path μ for which the number $f(\mu)$ is the smallest.
>
> *Problem 5.* To each partial subgraph G_1 of a given graph G, number $h(G_1)$ is assigned; find such a subgraph G_1 for which the number $h(G_1)$ is the smallest.
>
> *Note:* Each of the above-stated problems is a generalization of the previous one.

For the third problem, BERGE quotes FORD's method, which is described in greater detail in Section 3.4.2.[20] The algorithms for weighted graphs are dealt with in greater detail in the forthcoming sections.

Dijkstra's algorithm

In his formulation of the shortest path problem, DIJKSTRA says [Dij59a]:

> **Problem 2.** Find the path of minimum total length between two given nodes P and Q.

The algorithm he describes is very similar to the algorithm D of MOORE [Moo59]. In the following, DIJKSTRA's algorithm is described.

[19] On the other hand, it should be noted that he quotes a paper more than fifty years old — Tarry's paper [Tar95], which is something Moore does not do. This might be caused by the fact that Tarry's work is mentioned in D. Kőnig's monograph.

[20] Claude Berge quotes the RAND Corporation Paper P-923 by L. R. Ford, Jr. This paper most probably formed the basis of a classic book written by L. R. Ford and D. R. Fulkerson [FF62].

In brackets and in italics, allusions to MOORE's algorithm are given — in DIJKSTRA's terminology:

> In the course of solution the nodes are subdivided into three sets:
>
> **A.** the nodes for which the path of minimum length from P is known; nodes will be added to this set in order of increasing minimum path length from node P;
>
> **B.** the nodes from which the next node to be added to set A will be selected; this set comprises all those nodes that are connected to at least one node of set A but do not yet belong to A themselves;
>
> **C.** the remaining nodes.
>
> The branches are also subdivided into three sets:
>
> **I.** the branches occurring in the minimal paths from node P to the nodes in set A;
>
> **II.** the branches from which the next branch to be placed in set I will be selected; one and only one branch of this set will lead to each node in set B;
>
> **III.** the remaining branches (rejected or not yet considered).
>
> To start with, all nodes are in set C and all branches are in set III. We now transfer node P to set A and from then onwards repeatedly perform the following steps.
>
> *[We write* 1 *on node* P *and* 0 *on all the other nodes.]*
>
> *Step 1.* Consider all branches r connecting the node just transferred to set A with nodes R in sets B or C. If node R belongs to set B, we investigate whether the use of branch r gives rise to a shorter path from P to R than the known path that uses the corresponding branch in set II. If this is not so, branch r is rejected; if, however, use of branch r results in a shorter connexion between P and R than hitherto obtained, it replaces the corresponding branch in set II and the latter is rejected. If node R belongs to set C, it is added to set B and branch r is added to set II.
>
> *Step 2.* Every node in set B can be connected to node P in only one way if we restrict ourselves to branches from set I

and one from set II. In this sense each node in set B has a distance from node P: the node with minimum distance from P is transferred from set B to set A, and the corresponding branch is transferred from set II to set I. We then return to step 1 and repeat the process until node Q is transferred to set A. Then the solution has been found.

[On each node X, a number denoting the length of the shortest path from node P to X that has been founded until the moment is written. At each step, numbers written next to nodes being considered are compared with new path lengths: if the original number is greater, it is erased and substituted by the new path length. When the lengths of shortest paths do not change any more, the algorithm terminates.]

DIJKSTRA further remarks about his algorithm that it can be used also for directed graphs, that a better implementation of it is possible, and finally, that it is "to be preferred to the solution by L. R. FORD as described by C. BERGE in [Ber58]." In BERGE's monograph, we read [Ber58, pp. 79-81]:

1. Every vertex x_i gets an index λ_i; initially we take $\lambda_0 = 0$ and $\lambda_i = +\infty$ for $i \neq 0$.

2. We are looking for such edge (x_i, x_j) for which $\lambda_j - \lambda_i > l(x_i, x_j)$; then we replace index λ_j by index $\lambda_j' = \lambda_i + l(x_i, x_j) < \lambda_j$; we note that $\lambda_j' > 0$ for $j \neq 0$. We continue in this mode if there is still an edge which might enable us to diminish some λ_i.

3. After the indices have been established, we find such vertex x_{p_i} that $\lambda_n - \lambda_{p_i} = l(x_{p_i}, x_n)$; in fact, index λ_n diminishes monotonously during the process, and x_{p_1} is the last vertex which has contributed to the diminishing of λ_n. The vertex x_{p_2}, for which $\lambda_{p_1} - \lambda_{p_2} = l(x_{p_2}, x_{p_1})$, is found exactly in the same manner, etc.

The sequence $\lambda_n, \lambda_{p_1}, \lambda_{p_2}, \ldots$ is sharply decreasing, and therefore at some point, $x_{p_{k+1}} = x_0$. I claim that λ_n is *the length of the shortest path from x_0 to x_n and that $\mu_0 = [x_0, x_{p_k}, x_{p_{k-1}}, \ldots, x_{p_1}, x_n]$ is this shortest path.*

Analysis of this algorithm and of the algorithm due to FORD and FULKERSON, described in Section 3.4.2, reveals that they are equivalent,

even though the former formally consist of three and the latter of only two steps.[21]

Minty, before 1960

Another algorithm by MINTY[22] is reported by POLLACK and WIEBENSON in [PW60]. This algorithm is again very similar to that of DIJKSTRA published in 1959. It is also worth noticing that this algorithm is designed also for unsymmetric distance matrices, i.e. MINTY's "string model" would not work for this case.

> **Assumptions:** A network and distances between the nodes are given. Unsymmetric distance matrix is given. A ... starting city, B ... terminating city.
>
> **1.** Label city A with the distance 'zero'. Go to **2.**
>
> **2.** Look at all one way-streets with their 'tails' labeled and their 'heads' unlabeled. For each such street, form the sum of the label and the length. Select street making this sum a minimum and place a check-mark beside it. Label the 'head' city of this street with the sum. Return to the beginning of **2.**
>
> **Termination conditions:** (a) city B is labeled, (b) all cities have been labeled, (c) minimum of the sums is 'infinity'.
>
> **Advantage:** The labels do not change, once assigned. Thus, distances to all labeled nodes are known irrespective of whether the remaining nodes are labeled or not.

3.5 Conclusion

The early algorithms shown here do not use the complexity measure to describe the efficiency of their algorithm. However, they are concerned with the effectiveness of calculations. Later, special implementations of algorithms and algorithms for special classes of graphs were devised (see e.g. [Joh77]). Worth mentioning is also the paper by IRA POHL [Poh70], in which the author shows the connection between heuristic search and artificial intelligence.

[21] It is of course irrelevant how many steps are declared in the description, as it does not say much about the number of cycles the algorithm has to go through.

[22] Quoted as "Private Communication from G. J. Minty, University of Washington, Seattle, Washington" in [PW60].

Table 3.2: Complexity of Shortest Path Algorithms

Author, year	Complexity	Note
Moore, 1957	$O(m+n)$	unweighted graph; one $s-t$ path
Moore, 1957	$O(mn+n^2)$	unweighted graph; all $s-v$ paths
Dijkstra, 1959	$O(n^2)$	weighted graph; all $s-v$ paths

As far as complexity is concerned, we can find the following data (Table 3.2) on the complexity of MOORE's and DIJKSTRA's shortest path algorithms in PLESNÍK's monograph [Ple83, pp. 117-119]:

A particularly interesting article with respect to complexity is D. B. JOHNSON's article [Joh77] on shortest path algorithms in *sparse networks*. It reflects the move from looking for general solutions to looking for solutions that are better for some cases and worse for other ones: one algorithm is to be used for a "dense" graph and a different one for a "sparse" graph; and still a different one for "upgrading" the solution.

The multiple language of algorithms described in this chapter as well as the variety models suggested confirm the thesis that although there is one solution (the optimum one) to the problem, there might be many ways of arriving to it. The fact that the algorithms due to MOORE, DIJKSTRA, or FORD and BELLMAN are still used confirms the validity of their solutions. Nevertheless, the algorithms are being refined, rewritten, and, last but not least, implemented by more sophisticated methods.

Chapter 4

Network Flows: Multiple Roots

Network flow problems form another class of basic problems in discrete optimization. They are related to economy, like the shortest–path problems, and also to physics. From the mathematical point of view, the connections between shortest–path and network flow problems, as well as between matching theory and network flow problems are interesting: namely, shortest–path problems are dual to certain network flow problems and vice versa. Nowadays, flows in networks constitute a separate branch in discrete optimization. This chapter deals almost exclusively with single–commodity network flows, with some minor remarks on other variants of network flow problems. The history up to the publication of the monograph [FF62] is described.

Like the shortest–path problems, network flow problems were also formulated in different contexts. The history of the problem is often traced back to the physicist Gustav Kirchhoff and to the so-called *Kirchhoff's laws* for electrical current, formulated by him. The first people to formulate the problem in mathematical context were L. V. Kantorovich and F. L. Hitchcock. The latter also gave name to the *Hitchcock problem*, which is a term equivalent to the *transportation problem* or the *network flow problem*.[1] Scattered results were united by L. R. Ford and D. R. Fulkerson in 1962.

After the publication of the monograph [FF62], most authors writing on network flows quote this book. The most famous result stated in the monograph, bearing even the authors' name, is probably the *Ford-*

[1]The last term was advocated by Ford and Fulkerson in their monograph; see quotation on page 117.

Fulkerson Theorem, called also the *Max-Flow Min-Cut Theorem*. Finiteness of the *labelling method* is also examined in the book,[2] as well as the duality of network flow and shortest path problems.[3] As in the case of shortest–path problems, the authors of later papers on network flows refine and adapt the previous solutions. However, flows in networks seem to be dominated by the tandem FORD–FULKERSON, as these two coined the term and because the subject is so well dealt with in their monograph.[4]

4.1 Specific Background

In this section, the words *network* and *graph* are interchangeable. The term "network" is favoured by some authors because of its more direct visual interpretation. The most important results and specific definitions related to network flow problems are stated in this section, especially the connection between minimum cut and maximum flow in a network (graph). It can be shown that network flow algorithms are not finite if capacities of the edges are allowed to be real numbers. However, it holds that if all edge capacities are integers, then the maximum flow is also integer. This statement can be extended also to rational numbers, which thus provides optimistic computation results.

Apart from liquid transportation, the same methods can be used for transportation of any product, not excluding even the traffic in a town, where the capacities determine how many cars can go through a certain street in a certain time (one hour). From this, it is only a step further to the problems connected with the costs of transportation.

Definition 4.1 *Given a graph $G(V, E)$, suppose that each edge $v_iv_j \in E$ $(v_i, v_j \in V)$ is associated with a non-negative real number $c(v_iv_j)$, called the* capacity *of the edge v_iv_j. The function $c : E \mapsto \mathbf{R}_0^+$ is called the* capacity function.

Thus, for the purposes of network flow algorithms, the weight function bears the denotation "capacity". The *capacity* of an arc determines the quantity of a product that can flow through the edge (in a given period of time). According to the capacities given, some nonnegative *flow* can be constructed.

[2] For these theorems, see Section 4.1.

[3] Obviously, discussions on complexity start only later: the first hints on the need to compare algorithms appear in the mid-1960s.

[4] A similar situation arises with Dénes Kőnig's 1936 monograph: we can find many traces of Kőnig's exposition of graph theory in the works of later authors.

However, even though the original graph (in which we are to find maximum flow) is undirected, the graph describing the flow must always be directed. As was already mentioned in Chapter 2, we distinguish between the *in–degree* and the *out–degree* of a vertex in a directed graph:

Definition 4.2 *In a directed graph $G(V, E)$, $d_G^-(v)$ denotes the number of directed edges with their endpoint in v (the* in–degree*) and $d_G^+(v)$ denotes the number of directed edges with their starting point in v (the* out–degree*).*

The *in–degree* and the *out–degree* of the vertex tell us how many *predecessors* and *successors* the vertex v_i has. The set of predecessors of v_i (i.e. the set of vertices v_j for which the (directed) edge $v_jv_i \in E$) is denoted by $\Gamma_G^-(v_i)$ and the set of the successors of v_i (i.e. the set of vertices v_j for which the (directed) edge $v_iv_j \in E$) by $\Gamma_G^+(v_i)$. The cardinality of the set $\Gamma_G^-(v_i)$ is $d_G^-(v)$, and the cardinality of the set $\Gamma_G^+(v_i)$ is $d_G^+(v)$.

The following definition tells us what conditions must be satisfied by any flow in a network:

Definition 4.3 *Let s and t be two distinct vertices of V. A* (static) flow of value v from s to t *in G is a function $f : E \mapsto \mathbf{R}_0^+$ that for each $v_i \in V$ satisfies the linear equations and inequalities*

$$\sum_{v_j \in \Gamma_G^+(v_i)} f(v_i, v_j) - \sum_{v_j \in \Gamma_G^-(v_i)} f(v_iv_j) = \begin{cases} v & v_i = s \\ 0 & v_i \neq s, t \\ -v & v_i = t \end{cases}$$

$$f(v_iv_j) \leq c(v_iv_j), (v_iv_j) \in E.$$

The vertex s is called the source*, and the vertex t the* sink*.*

The middle condition $\sum_{v_j \in \Gamma_G^+(v_i)} f(v_iv_j) - \sum_{v_j \in \Gamma_G^-(v_i)} f(v_iv_j) = 0$ when $v_i \neq s, t$, namely that the flow into the vertex must be equal to the flow out of the vertex, is the so–called *Kirchhoff's law*. It is obvious that the same equation need not be valid for the capacity function.

Basic network flow algorithms operate with *single-source single-sink networks*, i.e. networks where the product flows from only one source to only one sink. Such algorithms can easily be adapted for problems with more sources and more sinks in the cases where the set of sources and the set of sinks have no intersection, namely by adding a source and a sink and edges with appropriate edge capacities. For these algorithms, the notion of a *cut* separating source and sink is central:

Definition 4.4 *Let $\overline{X} = V - X, X \subseteq V$ and let $(X, \overline{X})$ denote the set of all edges going from X to $\overline{X}$. A* cut *in $G(V, E)$ separating s and t is a set of edges $(X, \overline{X})$, where $s \in X$ and $t \in \overline{X}$. The* capacity *of the cut $(X, \overline{X})$ is denoted by $c(X, \overline{X})$, where*

$$c(X, \overline{X}) = \sum_{x\overline{x} \in (X, \overline{X})} c(x\overline{x}).$$

In the above–stated definition of a cut, the "edges between X and $\overline{X}$" are the edges going from X to $\overline{X}$; in directed graphs, the direction of the edges must be taken into account.

The Lemma 4.5 further specifies the relation between the cut and flow in the network and Theorem 4.6 states the equality between maximum flow and minimum cut and is one of the central theorems of network flow theory:

Lemma 4.5 *Let f be a flow from s to t in a graph $G(V, E)$ and let f have value v. If $(X, \overline{X})$ is a cut separating s and t, then*

$$v = f(X, \overline{X}) - f(\overline{X}, X) \leq c(X, \overline{X}).$$

Theorem 4.6 *[Max-flow min-cut theorem.] For any network the maximal flow value from s to t is equal to the minimal cut capacity of all cuts separating s and t.*

The phrase "flows in networks" evokes some product, or, more precisely, liquid flowing through piping. And indeed, it is sometimes suggested (e.g. in [BFN58]) that the product should be divisible into as small quantities as possible. However, if the capacities are allowed to be any real numbers, network flow algorithms need not be finite. On the other hand, already FORD and FULKERSON state [FF62, p. 19] the *Integrity theorem:*[5]

Theorem 4.7 (Integrity theorem) *If the capacity function c is integral valued, there exists a maximal flow f that is also integral valued.*

[5]The theorem follows from the construction of maximal flow — the labeling method: When all the capacities are integers, the algorithm proceeds by enlarging the flow by one (or another integer). As the upper bound on the maximum flow is the sum of the capacities of the edges going out of the source (coming into the sink), it is evident that in a finite graph with finite integer capacities, the algorithm takes no more steps than the "size" of the maximum flow.

The values of capacity function can also be rational, as all rational numbers — fractions — can be multiplied by the smallest common denominator, by which they "become" integers.

The following quotation describes why the use of graph theory is justified for electrical networks — i.e. for another "commodity", electricity:

> A means of describing the connections of electrical networks is provided by graph theory. Its application yields a method for solving network analysis problems, by means of a systematic derivation of an appropriate number of linearly independent equations. Digital computers are readily utilized for writing the necessary relationships and solving them. It is for this reason that the application of graph theory to the calculation of electrical networks has gained widespread use in recent decades.
>
> After the earliest work on graph theory (that of L. EULER published in 1736) it was G. KIRCHHOFF, as far as we know, who was the first to deal with this subject issued in 1847,examining primarily the laws of electrical networks. The first comprehensive book to discuss graph theory was that by D. KŐNIG and it was published in 1936. A detailed study of the aplication of graph theory to electrical networks is presented in the book of S. SESHU published in 1961. [Vág85, p. 5—Preface.]

4.2 The Springs and Streams

The origins of network flow theory are in various branches not only of mathematics, but also other sciences. KIRCHHOFF's paper is usually quoted as the first one on this topic. The connection of *Kirchhoff's laws* with graph theory was recognized already by DÉNES KŐNIG in his monograph [Kőn86, pp. 139–141]:[6]

> The previous investigations partly owe their origins in the question in electricity theory, which was posed and solved in 1845 by K i r c h h o f f. In a finite connected directed graph G, the edges $k_1, k_2, \ldots, k_{\alpha_1}$ should be understood as wires through which electricity circulates. For every (directed)

[6] The original quotation can be found in the Appendix on page 173.

> edge k_i, its electrical resistance is denoted by $\omega_i (> 0)$ and the electromotoric power E_i that resides in k_i (measured in the direction from k_i) are given. [...]

FORD and FULKERSON, on the other hand, start from the linear programming formulations of *transportation problems.* They say on the history of network flows [FF62, Preface]:

> Just where the study of network flow problems may be said to have originated is a debatable question. Certain static minimal cost transportation models were independently studied by Hitchcock, Kantorovich, and Koopmans in the 1940's. A few years later, when linear programming began to make itself known as an organized discipline, Dantzig showed how his general algorithm for solving linear programs, the simplex method, could be simplified and made more effective for the special case of transportation models. It would not be inaccurate to say that the subject matter of this book began with the work of these men on the very practical problem of transporting the commodity from certain points of supply to other point of demand in a way to minimize shipping cost. [...] However, dismissing the formulational and applied aspects of the subject completely, and with the advantages of hindsight, one can go back a few years earlier to research of König, Egerváry, and Menger on linear graphs, or Hall on systems of distinct representatives for sets, and also relate this work in pure mathematics to the practically oriented subject of flows in networks.

They also trace history of network flow problems back to KIRCHHOFF, but their main concern seems to be the mutual influence of mathematical results and practical transportation problems. The results of MENGER and EGERVÁRY bring the problem more to the mathematical side: their theorems and methods form the basis of the *Hungarian Method* for maximum matching.[7]

4.2.1 Hitchcock and Kantorovich

One of the classic articles dealing with network flows is HITCHCOCK's paper *The distribution of a product from several sources to numerous*

[7]The term "Hungarian Method" was coined by Harold W. Kuhn.

localities[Hit41] published in 1941. In his paper, HITCHCOCK defines the transportation problem in the following way [Hit41, p. 224]

> **1. Statement of the problem.** When several factories supply a product to a number of cities we desire the least costly manner of distribution. Due to freight rates and other matters the cost of a ton of a product to a particular city will vary according to which factory supplies it, and will also vary from city to city.

HITCHCOCK first shows the way of finding a feasible solution, then he takes the costs of transportation between the individual cities into account, and finally he gradually improves the solution. Apart from the statement of the problem, the method of solving the transportation problem is demonstrated on a concrete example.[8] HITCHCOCK further divides the paper into three sections. In the second section (Geometrical interpretation), he gives a geometrical representation of the problem,[9] which resembles the simplex used in linear programming. In the third section (Finding a vertex), he formulates the thesis that a feasible solution can be found in one of the vertices of the simplex. The term "vertex" is used here in a sense quite different from the term "vertex" used in graph theory. Finally, in the fourth section (Finding a better vertex) he gradually improves the solution by "travelling" to other vertices of the simplex.

The problem dealt with in the paper [Hit41] evidently belongs to the network flow problems. The problem was even named "Hitchcock problem", after the author. However, the solution presented is not a graph-theoretical one, but rather one using linear programming methods.

The paper by KANTOROVICH and the joint paper by KANTOROVICH and GAVURIN are often quoted as the first attempts at formulating linear programming. The methods they use also belong rather to the domain of linear programming than to the domain of graph theoretical algorithms.

Even G. B. DANTZIG admits that linear programming methods were formulated in the Soviet Union prior to their development in the U.S.A. However, he claims that it is legitimate to consider linear programming to be a U.S. patent, for which the sole fact that the methods of KANTOROVICH were unknown in the U.S.A. is a sufficient reason.[10] KAN-

[8] The style reminds one of E. F. Moore, who also described his algorithms on a specific example (1957).

[9] There is no picture in Hitchcock's paper, only a description of the situation.

[10] It is not the aim of this chapter to resolve the linear programming priority debate.

TOROVICH's method of solving the problem was not a graph–theoretical one, and is thus not of interest here.[11]

4.2.2 Graph theory versus simplex method

The paper *Die Graphentheorie in Anwendung auf das Transportproblem* presented by the Czech mathematicians J. BÍLÝ, M. FIEDLER and F. NOŽIČKA in 1958 consists of three parts: a theoretical one (Sections 1 to 3), an example (Section 4), and a historical note (Section 5). In Sections 1 to 3, the authors define graph–theoretical concepts, while in Section 4, they solve the Hitchcock transportation problem. Again, the method is presented on an example. As HAROLD KUHN says in his review, "it is the simplex method in disguise." They proceed in a way similar to HITCHCOCK and the paper makes the impression that they deliberately chose to "translate" the simplex method into graph–theoretical language.

In the second and third parts of the article, they introduce even such basic notions of graph theory as *(finite) graph, edge, cycle, even graph*, and the like.[12]

On the history of the transportation problem, they say [BFN58, pp. 119–120]:[13]

> The problem of distributing a product manufactured at several production sites to certain consumption sites of given consumtion volume (equal to the production volume) in such a way that transport costs are minimum was first formulated and solved by mathematical means by HITCHCOCK [14] [...] The lack of ships that already during World War I led to a certain regulation of the ship transport (??) and that became much more manifest during World War II led, during World War II, to the mathematical formulation and solving of this problem. In this respect it is necessary to note that sea transport poses certain special conditions, that are different from those of railway transport. [...] The simplex method for the solution of the transportation problem was given by DANTZIG [4]. [15]

[11] Kantorovich and Gavurin use metric spaces and the theory of potential.

[12] The authors are using German equivalents, obviously: *(endlicher) Graph, Kante, Kreis, paar Graph.*

[13] The original quotation is given in the Appendix on page 171.

[14] Reference mine.

[15] Reference [4] cites Dantzig, G. B.: Applications of the Simplex Method to a

The authors also shed some light on the research connected to network flows in Czechoslovakia and on the importance of network flow problems for economics [BFN58, p. 119]: [16]

> In the Czechoslovak Republic, the transportation problem, as defined above, was encountered in connection with the efforts to economize railway transport. The problem was solved in 1952 by Nožička independently of the papers stated above; the method was explained in detail in [12].
>
> The growing interest of mathematicians and people from the practice in Czechoslovakia and abroad — see e.g. [1] and [14] — in the transportation problem as well as in other linear programming problems led the authors of this article to write a shortened and revised version of the paper [12], taking graph theory as the basis. The possibility to solve the transportation problem with the help of this theory was mentioned in [4] and [13] and used by FLOOD in [6]; on this, see also BELLMAN in [2]. We find this method very illustrative and particularly suitable when there are no modern computing machines available; on their utilization see Eisenmann [5]. On the existence questions see [3].

They state the following limitations of the transportation (network flow) problems [BFN58, pp. 119–120]: [17]

> The transportation problem is solved under the following c o n d i t i o n s:
>
> 1. The product can be divided into arbitrarily small amounts.
> 2. There is no limit on the smallest amount to be transported; this condition does definitely not apply, e.g. for coal the smallest amount is a railway wagon.
> 3. The means of transportation used for transport remain empty on the return journey.
> 4. The transport continues with a constant intensity.
> 5. The transport routes have an unlimited capacity. [...]

Transportation Problem, *Activity Analysis of Production and Allocation*, 359–373. (The year of publication not stated.)

[16] The original quotation is given in the Appendix on page 171.

[17] The original quotation is given in the Appendix on page 172.

4.3 1962: The Confluence

A major breakthrough in the network flow theory can be seen in the publication of the classic monograph *Flows in Networks* by L. R. FORD and D. R. FULKERSON from the *RAND Corporation*. Their rather thin book was published in 1962.[18] In this book, the authors managed to encompass network flow theory up to their time.[19] In the preface of their book, the authors say [FF62, p. vii.]:

> This book presents one approach to that part of linear programming theory that has come to be encompassed by the phrase "transportation problems" or "network flow problems".

KANTOROVICH and GAVURIN, as well as DANTZIG used linear programming methods for solving transportation (network flow) problems. In the book by FORD and FULKERSON, the methods are not explicitly stated to be graph theoretical; yet it is evident that their meaning of *nodes* and *arcs* corresponds with the notions in graph theory. In the preface of [FF62], the authors say:

> While this is primarily a book on applied mathematics, we have also included topics that are purely mathematically motivated, together with those that are strictly utilitarian in concept. For this, no apology is intended. We have simply written about mathematics which has interested us, pure or applied.
>
> To carry the historical sketch another (and our last) step back in time might lead one to the Maxwell-Kirchhoff theory of current distribution in an electrical network.

In this book, we find the treatment of *static maximal flow*, *minimal cost flow problems*, as well as *multi-terminal network flows*. The second chapter of the book — *Feasibility Theorems and Combinatorial Applications* — brings the readers' attention to more general results and puts network flow theory into a wider mathematical context. Namely, "various combinatorial problems [...] can be posed and solved in terms of network flows. The remainder of this chapter illustrates this method of attack on a number of such problems". [FF62, p. 36]

[18] For the text of the review in *Mathematical Reviews*, see the end of this chapter.

[19] Major part of this was most probably published by L. R. Ford, Jr. in the RAND Corporation Paper P-923 in 1956.

The usage of network flows in economy

The "commercially most successful by–product" of network flow theory, the critical path method, is ommitted in the monograph [FF62]. However, its authors published other papers in which they advocated the use of network flows. The article by D. R. FULKERSON "A network flow computation for project cost curves" is one of them. Linear programming problem is transformed into a network flow problem here: namely, the differences between the normal and crash completion time for a job are transformed into capacity of the arcs.

4.4 After "Flows in Networks"

The subject of multiterminal network flows is alloted a comparatively short space in the monograph [FF62]. FORD and FULKERSON actually put this problem aside, as they say that the basic procedures can easily be adapted from the single–source single–sink network flow problems. These adaptations are often the themes of more recent papers on network flows.

4.4.1 Multiterminal network flows

In his paper *Sensitivity analysis of multiterminal flow networks* [Elm64], SALAH E. ELMAGHRABY deals with multiterminal network flows: networks in which any node can serve as a source and any other as a terminal. In such networks, using the results of network flow theory for single–source single–sink network flows, it would be necessary to solve $\frac{1}{2}N(n-1)$ maximum flow problems. However, ELMAGHRABY quotes a source for a better procedure [Elm64]:

> Fortunately, GOMORY and HU have demonstrated that such is not the case and devised an ingenious technique by which the desired maximal flow can be determined in at most $N-1$ steps. Each step involves the solution of a 'maximal flow problem', oftentimes in a much reduced network.

ELMAGHRABY states also the equivalence between a certain class of network flow problems, namely network flow problems in planar graphs, with the shortest paths problems. [Elm64, p. 688]

4.5 Conclusion

Network flow problems are a complex class of discrete optimization problems. These "transportation problems" can be solved very well by both linear programming methods and graph–theoretical algorithms, which distinguishes them from the shortest–path problems mentioned in the previous chapters. It is also worth mentioning that the first concise treatment of the network flow problems was published almost forty years ago, including the transformation of multi–terminal network flow problems into single–source single–sink ones.

4.6 Appendix: Flows in Networks, 1962

In this section, the reviews of the book L. R. Ford, Jr. and D. R. Fulkerson: *Flows in Networks* (1962) are reprinted, as they appeared in the *Mathematical Reviews* and the *Zentralblatt für Mathematik und ihre Grenzgebiete.*

Mathematical Reviews 28 #2917

This book is an attractive, well-written account of a fairly new topic in pure and applied combinatorial analysis. The prospective reader should not misconstrue the title so as to expect current to flow in electrical networks anywhere in this book. The networks are simply linear graphs, and the applications are ultimately descendants of Euler's "bridges of Konigsberg" problem, where modern highway engineering has restricted most of the bridges to one-way traffic, and imposed load limitations on the vehicles which traverse them. So if, in modern Kaliningrad (even the name of the city has changed), it were required to transport a specified load to specified destinations over these one-way, load-limited bridges at minimum cost (whether in time or gasoline), this would be a problem in network flow theory. In general, the applications are to problems in "operations research" (routing, scheduling, inventory control, program management, etc.), problems which seemed purely recreational a few decades ago, but which have suddenly become technologically indispensible, partly because modern computers can make practical use of algorithms which would formerly have been considered unmanageable and partly because they give to every eager bureaucrat the opportunity to become a "management scientist". Thus the subject has the dual virtues of mathematical interest and commercial marketability.

An accurate reflection of the topics covered is best conveyed by the Table of Contents:

Chapter I, Static maximal flow:

(1) Networks,

(2) Flows in networks,

(3) Notation,

(4) Cuts,

(5) Maximal flow,

(6) Disconnecting sets and cuts,

(7) Multiple sources and sinks,

(8) The labeling method for solving maximal flow problems,

(9) Lower bounds on arc flows,

(10) Flows in undirected and mixed networks,

(11) Node capacities and other extensions,

(12) Linear programming and duality principles,

(13) Maximal flow value as a function of two arc capacities;

Chapter II, Feasibility theorems and combinatorial applications:

(1) A supply-demand theorem,

(2) A symmetric supply-demand theorem,

(3) Circulation theorem,

(4) The Konig-Egervary and Menger graph theorems,

(5) Construction of a maximal independent set of admissible cells,

(6) A bottleneck assignment problem,

(7) Unicursal graphs,

(8) Dilworth's chain decomposition theorem for partially ordered sets,

(9) Minimal number of individuals to meet a fixed schedule of tasks,

(10) Set representatives,

(11) The subgraph problem for directed graphs,

(12) Matrices composed of 0's and 1's;

Chapter III, Minimal cost flow problems:

(1) The Hitchcock problem,

(2) The optimal assignment problem,

(3) The general minimal cost flow problem,

(4) Equivalence of Hitchcock and minimal cost flow problems,

(5) A shortest chain algorithm,

(6) The minimal cost supply-demand problem: non-negative directed cycle costs,

(7) The warehousing problem,

(8) The caterer problem,

(9) Maximal dynamic flow,

(10) Project cost curves,

(11) Constructing minimal cost circulations;

Chapter IV, Multi-terminal maximal flows:

(1) Forests, trees, and spanning subtrees,

(2) Realization conditions,

(3) Equivalent networks,

(4) Network synthesis.

It should be noted that Chapter II contains many topics of considerable combinatorial significance, quite independent of possible applications.

In a book of only moderate length, it is not possible to discuss everything. The following omissions are noted only to inform the prospective reader of their absence, and are in no way intended as criticisms of the authors' choice of material. The simplex method for solving network flow problems is not developed. There is no discussion of the famous "travelling salesman" problem. As previously noted, applications to electrical networks are absent, along with continuous flow problems generally. (It is a book, after all, on combinatorial analysis.) Finally, the most commercially successful byproduct of network flow theory, the "PERT chart", has been mercifully omitted. [Reviewer: Golomb, S. W.]

Zentralblatt 106.34802

The present work, based primarily on the author's research, is the first systematic and complete treatment of the class of problems dealing with flows in capacity constrained networks. For this reason, it is difficult to present the numerous and interesting results. Chapter I studies the static maximal flow problem. If we denote by $N = \{x, y, \ldots\}$ a finite set of nodes, by s, t two distinguished elements, by A a subset of arcs (x, y), by $c(x, y)$ the non-negative real number associated to (x, y) and called capacity and by $G = (N; A)$ the resulting connected network, then the problem is to maximize v subject to the flow constraints $\sum_{y \in A(x)} f(x, y) - \sum_{y \in B(x)} f(y, x) = v, 0, -v$ (according as $x = s, x \neq s, t$ or $x = t$) and $f(x, y) \leq c(x, y)$, where $f : A \mapsto \Re^+$ (non-negative reals), $A(x) = \{y \in N | (x, y) \in A\}$ and $B(x) = \{y \in N | (y, x) \in A\}$. Given a network G, a cut C separating s and t is a set of arcs $(X, \overline{X})$, where $s \in X$ and $t \in \overline{X}$; the capacity of the cut is $c(X, \overline{X}) = \sum_{(x,y) \in (X, \overline{X})} C(x, y)$. One obtains the fundamental result, namely the max-flow min-cut theorem: for any network the maximal flow value from s to t is equal to the minimal cut capacity of all cuts separating s and t. A simple and efficient method (labelling process) is given in order to obtain a maximal flow. Also, an important result is proved (integrity theorem), asserting that if the capacity function is integral valued then the resulting flow is also integral valued. These three topics form the base of the whole book. In Chapter II, several necessary and sufficient conditions (feasibility theorems) for the existence of a network flow satisfying various linear inequalities are obtained. Using the integrity theorem, the authors give a treatment in terms of the network flows of some combi-

natorial problems such as Kőnig-Egerváry and Menger graph theorems, Dilworth's chain decomposition theorem for partially ordered stes, set representatives, matrices composed of 0's and 1's. The minimal cost flow networks, considered in Chapter III, are of the type of transport problems. A combinatorial technique, based on the Hungarian method, is applied to numerous problems: the Hitchcock problem, the optimal assignment problem, the shortest chain problem, the warehousing problem, the caterer problem. One presents also a general algorithm for solving the minimal cost flow problem in networks with constraints and an equivalence is proved between this problem and the Hitchcock problem. The last chapter of the book is devoted to multiterminal maximal flows. The problems are similar with those of the first chapter; now one considers no more a single pair of nodes but the whole set. The book has no banal parts; the attention is captivated from the first to the last page. [Reviewer: D. Vaida]

Keywords: applications of probability theory, mathematical programming.

Chapter 5

Minimum Spanning Tree: Neglected Papers

The minimum spanning tree problem, like the shortest–path problems, is one of the "easy" discrete–optimization problems, in terms of complexity of the existing algorithms for solving the problem. The minimum spanning tree problem was formulated in many different ways and contexts as early as the first half of the twentieth century. Later, this problem became one of the basic graph–theoretical problems. Already in the 1950s, formulations of this problem in a "purely theoretical" mathematical language appeared. With the emergence of complexity theory in the 1970s, attempts at reducing the computing time of the minimum spanning tree algorithms, particularly by employing special implementation of algorithms and special data structures, become more important.

5.1 Specific Background

Before going into a more detailed analysis of the minimum spanning tree algorithms, a brief survey of the notions and results on minimum spanning trees is presented.

Definition 5.1 *A* subtree *of a finite graph $G(V, E)$ is a subgraph that does not contain cycles. A* spanning tree *of a finite graph $G(V, E)$ is a connected factor that does not contain cycles.*

Definition 5.2 Minimum spanning tree *of a finite weighted graph $G(V, E)$ is a* spanning tree *of the graph $G(V, E)$, for which the sum of the weights of the edges is minimal of all the spanning trees of the graph $G(V, E)$.*

In contemporary graph–theoretical textbooks, the task of finding a minimum spanning tree can be formulated in the following way: "Given an undirected weighted graph, find its minimum spanning tree." The graph for the minimum spanning tree problem is *weighted* and *loopless*.[1] Some mathematicians also require that the weights of the edges are mutually different. When this condition is satisfied, the minimum spanning tree is uniquely determined. On the other hand, if the contrary holds, the minimum spanning tree still exists — only it may not be uniquely determined.

Theorem 5.3 *If the weights of the edges of a graph are mutually different, the minimum spanning tree of this graph is uniquely determined.*

Proof: The theorem will be proven by contradiction.
Assume that there exist two mutually different minimum spanning trees T_1, T_2 of a finite graph $G(V, E)$. Then there is at least one edge in each minimum spanning tree that is not contained in the other tree: $e_1 \in T_1$ and $e_2 \in T_2$. Let us assume (without loss of generality) that the weight of e_2 is less the weight of e_1. (According to the assumption, the weights cannot be equal.) When e_1 is removed from T_1, the resulting graph $T_1^{'}$ consists of two subtrees. The two following cases can occur:

1. e_2 connects the two subtrees. Then we connect the two subtrees by e_2. This way, we construct a minimum spanning tree that is *smaller* than the minimum spanning tree T_1, which is a contradiction.

2. e_2 does not connect the two subsets. Then e_2 must form a cycle with some edges of T_1. We return the removed edge e_1 to $T_1^{'}$, add e_2 to $T_1^{'}$ and delete the edge with the greatest weight from the cycle formed by the edges of T_1 and e_2. It is obvious that the weight of the resulting tree is again smaller that the weight of the minimum spanning tree T_1, which is a contradiction again. □

When edge weights are not mutually different, there can be more then one minimum spanning tree. However, it can easily be seen that this condition (for mutually different edge weights) can be secured in practice by adapting the edge weights slightly. This claim is justified e.g. by ANTON KOTZIG in his paper [Kot61b], in which he also states and proves the necessary and sufficient condition for the existence of a unique minimum spanning tree of a graph. The requirement of mutually different edge weights has hardly any consequences in the sense of

[1] For precise definitions of these notions, the reader is referred to Chapter 2.

limitations for the practical usage of the minimum spanning tree algorithms.[2]

The classical minimum spanning tree problem, as described above, is defined for undirected weighted graphs (usually, but not necessarily, without multiple edges). The corresponding term for directed graphs is "arborescence". This problem is dealt with, for example, by the Chinese mathematicians CHU and LIU in their paper [CL65].[3]

Generalisations of the problem can also be connnected with the network's reliability: namely, discarding any edge from an existing (minimum spanning) tree renders the graph disconnected.[4] The task of designing the cheapest network which would at the same time "survive" removal of any one of the edges was posed by KAREL ČULÍK [Čul60a, Čul60b] and ANTON KOTZIG [Kot61a].

5.2 The Development of the Minimum Spanning Tree Problem

Up to now, the minimum spanning tree problem has been solved by many mathematicians — the best known algorithm is probably that of KRUSKAL (1956) and the oldest one that of BORŮVKA (1926). Some procedures for cluster analyses are sometimes also considered to belong to minimum spanning tree algorithms (see [GH85]). [5]

There are many practical applications of the minimum spanning tree problem, or rather, practical usage was often the motivation for solving this problem. Such was the case with at least some mathematicians, for example BORŮVKA, PRIM, and LOBERMAN and WEINBERGER. Electricity, telephone, or computer networks may be used for illustrating the problem. For electricity networks, the task may be formulated as follows:

There are n towns which need to be connected in an elec-

[2]Similarly, the fact that integral–valued flow exists in graphs with integral–valued capacity function does not limit the use of Ford-Fulkerson algorithm in practice. (See Chapter 4 for details.)

[3]For definition, see Section 5.4.

[4]Any tree – weighted, unweighted, or minimum – with one edge left out is, of course, not a connected graph.

[5]Graham and Hell trace the prehistory of the minimum spanning tree problem as far as the cluster analyses used by the Polish anthropologist Jan Czekanowski as early as 1905. This seems a bit far–fetched, but as the work done by Łukaszewicz et al. is also connected to anthropology, the connection of cluster analyses and the minimum spanning tree problem could be considered.

> tricity network. The distances between every two towns are given. The task is to find the network which would connect all the towns using the least wire possible (and thus minimizing the cost of building the network).

The above is, in fact, the formulation OTAKAR BORŮVKA, a Czech mathematician, was presented with in 1925 by an employee of the West Moravian Power Plant Company, JINDŘICH SAXEL.[6] However, BORŮVKA did not use the graph–theoretical language of *vertices* or *nodes* connected by *edges, arcs,* or *branches* in his solution. Instead, his language is a language of algebra: *matrices*, *rows*, and *columns*.[7]

Soon after BORŮVKA published his solution, another Czech mathematician, VOJTĚCH JARNÍK, reacted by publishing his own solution to the problem. His solution was formulated clearly and succinctly. It is not very formalized (which is not to say that it is mathematically imprecise), but, unlike BORŮVKA, JARNÍK is not concerned with the practical application of the problem too much. Similarly, we find little information on the use of spanning trees in practice in another reaction to BORŮVKA's paper [Kru56], written in 1956 by the American mathematician J. B. KRUSKAL. Other mathematicians do not react to BORŮVKA's paper, although at least one of them quotes him (R. C. PRIM, 1957: [Pri57]). Some mathematicians (e.g. LOBERMAN and WEINBERGER, 1957: [LW57]; R. C. PRIM, 1957: [Pri57]) are concerned with other applications of the minimum spanning tree problem, or look at the problem from the mathematical point of view only (e.g. E. W. DIJKSTRA, 1959: [Dij59b]).

The last mathematician mentioned, DIJKSTRA, actually presents JARNÍK's method for finding the minimum spanning tree, only in a different mathematical language.[8] When VOJTĚCH JARNÍK published his response in 1930 [Jar30], he used the same words as BORŮVKA to name

[6] Borůvka himself says he wants to maintain the memory of Jindřich Saxel, including his name, as Saxel, being a Jew, fell victim to fascism during the Second World War:
"At that time, that is, some time between the the end of 1925 and the beginning of 1926, I met an employee of West Moravian Power Plants, a man of the name Jindřich Saxel. He was a graduate of a secondary technical school, but a very talented and hard–working person. I want to mention his name, particularly to pay homage to his remembrance — as a person of Jewish descent, he was executed by the Nazis during the martial law in Brno in the Kaunitz college." [TŠP96, p. 52]

[7] Borůvka's solution will be analysed in greater detail in Section 5.3.1.

[8] At this point, the distinction is being made between Jarník's and Dijkstra's terminology ("numbers" versus "nodes"). It is irrelevant that one of the papers was written in Czech and the other in English. See Sections 5.3.2 and 5.4.4 for details.

the problem: it is also a *certain minimum problem* for him. In contrast to that, DIJKSTRA speaks about a standard graph–theoretical problem: *construction of the tree of minimum total length.*

Before the most famous paper on the minimum spanning tree was published [Kru56], at least two other papers were published in French: a note by GUSTAVE CHOQUET (1938) and a joint paper by FLOREK, ŁUKASZEWICZ, PERKAL, STEINHAUS, and ZUBRZYCKI (1949), which both present solutions similar to those of BORŮVKA and JARNÍK.

J. B. KRUSKAL, in his reaction [Kru56] to the abstract of BORŮVKA's 1926 paper, introduces the term "shortest spanning subtree". He cites BORŮVKA's paper [Bor26a] as an inspiration for his work, but his assessment of BORŮVKA's solution as "unnecessarily elaborate" [Kru56, p. 48] probably caused the luke–warm interest in this paper in the mathematical community in the years to follow. This is how PRIM interpreted KRUSKAL's words [Pri57, p. 1401]:

> Kruskal refers to an *obscure* Czech paper [Bor26a] as giving a construction and uniqueness proof *inferior* to his.[9]

LOBERMAN and WEINBERGER, the authors of a minimum spanning tree algorithm equivalent to KRUSKAL's, seem to have encountered the minimum spanning tree problem quite independently of other mathematical papers. They pose the following problem [LW57, p. 428]:

> Given a number of terminals, fixed in space, which must be electrically connected together, what procedures will provide the minimum wire length? A *proper pattern* of connections is one in which there exists one and only one path, either direct or through other terminals, from each terminal to every other terminal and in which there are no loops created by redundant connections.

As can be demonstrated by the quotation above, LOBERMAN and WEINBERGER use a completely different term for minimum spanning tree: for them, it is a "proper pattern". They, however, show some awareness of the fact that this problem belongs to a particular area of mathematics. In the following paragraph, they say [LW57, p. 428]:

> Problems of this type have been considered in topological areas of mathematics, more particularly in the theory of graphs [Kőn86].[10]

[9] Emphases and reference mine.
[10] Reference mine.

In the same year, R. C. PRIM published his solution of the *shortest connection networks*. Again, his paper is practically motivated. His two principles for building the "shortest connection network" suggest a procedure similar to BORŮVKA's algorithm, but his demonstration suggests the use of the procedure put forth by JARNÍK. The fact that PRIM does not state clearly the order of the steps puts his paper on a similar level with that of E. F. MOORE on the shortest path problem.[11]

A quite different approach can be found in the paper of E. W. DIJKSTRA, published only two years later [Dij59b]. DIJKSTRA simply speaks about "two problems in connexion with graphs". The algorithm presented in his paper is actually the same as the one presented by PRIM two years before, or even earlier (1930) by JARNÍK. DIJKSTRA, however, quotes neither. He only quotes the papers [Kru56] and [LW57] and demonstrates that his solution is better than these two.

In the 1960s and 1970s, the debate about the quality of algorithms shifts towards the speed of the algorithms. This is for example manifest in the article by YAO [Yao75], a major part of which is devoted to the analysis of the algorithm presented. Of course, assessment of algorithms according to their effectiveness and storage requirements can be found also in earlier papers, but these remarks are often not comparable (especially because of the terminology, which was far from being united then).

The classical minimum spanning tree problem can also be generalised. In his papers [Kot61b, Kot61a], ANTON KOTZIG proposes two modifications of the problem: one is concerned with the uniqueness of the solution, the other with the network's stability. The former, as KOTZIG himself claims, is not of great importance. The latter, however, changes the minimum spanning tree problem into a problem of building a reliable network. The same problem was posed also by the Czech mathematician KAREL ČULÍK in 1960 [Čul60b].

The following sections consist of the description of the algorithms presented by BORŮVKA (1926), JARNÍK (1930), KRUSKAL (1956), LOBERMAN and WEINBERGER (1957), PRIM (1957), and DIJKSTRA (1959), respectively. The algorithms are presented also in adapted versions in order to make the comparison of the methods used by the individual mathematicians possible. Section 5.5 then points out some directions of the further development of the minimum spanning tree problem.

[11] See Chapter 3 for details of Moore's paper.

5.3 The "Neglected Papers"

The three papers presented in this section all deserve the attribute "neglected." They all present good algorithms for solving the minimum spanning tree problem, but they remain unnoticed by contemporary mathematicians, who usually cite the algorithms of KRUSKAL and PRIM. Two of them are written by Czech mathematicians BORŮVKA [Bor26a] and JARNÍK [Jar30] (in Czech), the third one is written by a group of Polish mathematicians led by ŁUKASZEWICZ (in French).

5.3.1 Borůvka, 1926

From the two articles BORŮVKA published on this topic, one is the mathematically precise formulation [Bor26a], the other one is an article in *Elektrotechnický obzor*, a journal for electrotechnics [Bor26b]. The former is written in a mathematically precise way using terms like "matrix", "row", and "column", the latter in plain language, but the idea behind both of them is the same. It is also possible that the use of the "language of algebra" in the paper [Bor26a] was deliberate, namely, that the article [Bor26b], written in plain language, would not have been accepted as a "mathematical paper".

Borůvka's mathematical formulation of the problem

In the mathematical paper, BORŮVKA starts by posing the following problem [Bor26a, p. 37]:

> Let a matrix of M numbers $r_{\alpha\beta}(\alpha, \beta = 1, 2, \ldots, n; n \geq 2)$, up to the condition $r_{\alpha\alpha} = 0, r_{\alpha\beta} = r_{\beta\alpha}$ positive and mutually different, be given.
>
> We are to choose from it a group of mutually different non-zero numbers such that
>
> 1° it were possible, if p_1, p_2 are any two different natural numbers $\leq n$, to choose from it a partial group of the form
>
> $$r_{p_1 c_2}, r_{c_2 c_3}, r_{c_3 c_4}, \ldots, r_{c_{q-2} c_{q-1}}, r_{c_{q-1} p_2}$$
>
> 2° the sum of its members is lower than the sum of the members of any other group of mutually different non-zero numbers complying with the condition 1° .

BORŮVKA then presents his algorithm, described below. In the article, operations are performed on rows and columns of a matrix. Another interesting feature is the presentation of the algorithm: as no code for describing such procedures existed then, the algorithm is rather difficult to follow. As a final remark, BORŮVKA connects "points in a plane (or generally in a r-dimensional space" [Bor26a, p. 16] and refers the reader to the more down–to–earth description of his procedure in his article [Bor26b].

Algorithm: Borůvka, 1926

A symmetric matrix $A(n \times n)$ of mutually different positive numbers and with zeros on the main diagonal is given. The aim is to find a group of n numbers satisfying the following two conditions (as quoted above):

1° the condition of connectedness, and

2° the condition of minimality.

The algorithm presented in the paper [Bor26a] is difficult to follow for the contemporary reader, and therefore it is presented in an adapted form here.[12] The version presented below is of course not yet the nowadays desired computer programme; however, changing BORŮVKA's expression in a pseudo-code would completely distort the picture of *how* the algorithm was formulated.

Step 1. Choose a row r_α in the matrix.[13]

Step 2. Choose the smallest positive number in this row, $r_{\alpha\beta}$. [This number, marked T_1, becomes the first member of the minimum spanning tree.] We now look at the row r_β, determined by the column index of the number $r_{\alpha\beta}$ that has just been chosen.

Step 3. (The algorithm now divides into two branches according to the answer to the following question:) Is there a number in the new row r_β that is smaller than the number $r_{\alpha\beta}$ included in the minimum spanning tree T_k in the previous step?

Yes: The smallest of such numbers becomes a new member of the minimum spanning tree, and is marked $T_(k+1)$. Perform **Steps 2** and **3** with the newly added number.

[12] In the original, BORŮVKA does not distinguish between the individual steps, for example.

[13] Alternatively, we may start by choosing a column.

No: Choose a different row (arbitrary choice) and perform **Steps 2** and **3**. If there are no isolated rows, **Stop**.

The procedure given above is performed until there are no isolated rows, i. e. there are no rows in which there is no number chosen. The maximum number of repetitions is $(n-1)$, according to BORŮVKA. As the result of these $(n-1)$ repetitions, we get smaller sets which need to be connected. BORŮVKA computes a smaller matrix: the distances are now computed as the shortest distances between the subsets of numbers. Finally, after we get a set that contains all the subsets, we construct the minimum spanning tree. Since the distances are mutually different for BORŮVKA, there is no problem with constructing the tree using the numbers (lengths or weights of the edges) chosen in the course of the algorithm, for each edge is uniquely determined by its length.[14]

From the historical point of view, it is important to mention that BORŮVKA gave a lecture on this problem and his solution of it while he was in Paris in 1926. The topic was chosen by PROF. COOLIDGE,the chair of the seminar, out of three topics offered by BORŮVKA. According to R. L. GRAHAM and P. HELL [GH85, p. 47], the following note recommended by ELIE CARTAN,with whom BORŮVKA was in close contact during his 1926 stay in Paris, was published in *Comptes Rendus* by GUSTAVE CHOQUET ([Cho38], translation taken from [GH85, p. 47]) in 1938:

> *Construction of the Minimum Network:* One joins by a segment each city with the city nearest to it. If the set of all these segments forms a continuum line [is connected], it is the desired network. If not, one joins each of the continua [connected components] with the continuum nearest to it by means of a segment joining the closest two cities of the continua. If the set formed in this fashion is a continuum, it is the desired network. If not, one continues in the same way. The desired network will be found after at most $2n$ elementary operations, where an elementary operation is the search for the continuum nearest to a given continuum.

The above description is actually very similar to BORŮVKA's article in *Elektrotechnický obzor* [Bor26b, pp. 153–154]:[15]

[14] It was proved in Section 5.1 that when all the edge weights are different, the minimum spanning tree is uniquely determined.

[15] For full version of the article, see the Appendix.

> I will solve the task [of joining given points into a connected network with shortest length] for the forty points given in [Fig. 1].
>
> I connect each of these points with the closest one. [...] I get several polygonal lines. [...]
>
> I connect each of these [polygonal lines] in the shortest way with the nearest [polygonal line]. [...]
>
> Finally, I get one polygonal line that solves the given problem. [16]

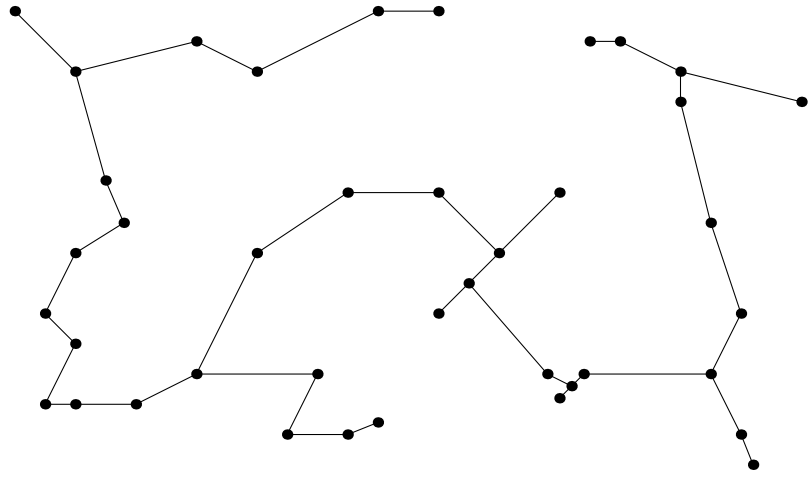


Figure 5.1: Borůvka: The Solution

It was not until 1956 that BORŮVKA's paper became more widely known outside Czechoslovakia. In Czechoslovakia, there were some reactions to BORŮVKA's solution. The first response came from another Czech mathematician, VOJTĚCH JARNÍK.BORŮVKA's paper is also quoted by ANTON KOTZIG in his papers [Kot61b, Kot61a], in the textbook [ČDF67], and (obviously) in many later Czech and Slovak textbooks.

5.3.2 Jarník, 1930

VOJTĚCH JARNÍK's article on this topic [Jar30] is an extract from a letter to O. BORŮVKA. Therefore it is written in the first person singular,

[16] See Figure 5.1

which is quite unusual for a mathematical paper. JARNÍK introduces the problem by saying [Jar30, p. 1]:

> The interesting problem you solved in your paper [Bor26a][17] can be solved in a different and — in my opinion — better way.

The formulation of the problem by JARNÍK is particularly interesting [Jar30, p. 1]:

> Let $n(n \geq 2)$ elements, denoted by the numbers $1, 2, \ldots, n$, be given. From these elements, I construct $\frac{1}{2}n(n-1)$ couples $[i, k]$, where $i \neq k; i, k = 1, 2, \ldots, n$; the couple $[i, k]$ is identical with the couple $[k, i]$. Every couple $[i, k]$ is assigned a positive number $r_{i,k}(r_{k,i} = r_{i,k})$. These numbers $r_{i,k}1 \leq i < k \leq n)$, in the number of $\frac{1}{2}n(n-1)$, should be mutually different.

JARNÍK further speaks about a *chain*, *complete part*, and *minimum complete part*. He finally describes his solution on an example of n small balls with sticks of different weight, namely the small balls i and k are connected by a stick with weight $r_{i,k}$. The sticks do not touch each other — for this purpose, they can also be bent. The task is: remove some of the sticks in such a way that all the small balls remain connected and the weight of the remaining sticks is minimum of all such sets of sticks.

Algorithm: Jarník, 1930

JARNÍK connects the given elements starting from any one of the elements and ending when all the elements have been connected into an uninterrupted whole — the minimum spanning tree, in later terminology. The first and second steps are identical with BORŮVKA's, the third step is different: JARNÍK's algorithm never proceeds to a completely new vertex, outside the existing "partial set"; he is always trying to extend the already existing part.

Step 1. Choose a point. (This is just a different formulation of Borůvka's choice of a collumn.)

Step 2. Choose the shortest edge going out of that point. (Again, equivalent to Borůvka's choice of the smallest number in the chosen row.) **Steps 1** and **2** give a *fragment*.

[17] Reference mine.

Step 3. Find the point closest to the fragment. Perform **Step 3** until all the points are joined.

JARNÍK's article is much shorter than BORŮVKA's, yet he manages to describe the algorithm concisely. He actually presents the algorithm in a *definition* of the *minimum complete part (mcp)* and says at the end of the article that "the construction of the *mcp* is clear from its definition."

Since BORŮVKA's article was read at least in Paris and thirty years later by J. B. KRUSKAL, JARNÍK's article is perhaps the one that deserves the name "neglected" most. As JAROSLAV NEŠETŘIL and BERNARD KORTE say [KN99, p. 42]:

> [...] O. Borůvka is quoted by both the standard early references: J. Kruskal [Kru56] and R. Prim [Pri57]. Vojtěch Jarník's article only began to be quoted later, see e.g. K. Čulík, V. Doležal, and M. Fiedler [ČDF67], despite the fact that his treatment was very precise (like all his mathematical work) and modern.[18]

On the other hand, JARNÍK's paper was written in Czech and did not have a German summary like BORŮVKA's article, which could be a good enough reason for it to be forgotten.[19] There is also another point about quoting the article [Bor26a]: the fact that other mathematicians quote it is, in itself, not enough. The citation by KRUSKAL is accompanied by the remark in the text that "the construction given by Borůvka is unnecessarily elaborate", and PRIM even indirectly suggests that he has not read BORŮVKA's paper at all, because he did not consider it worth reading. The same assessment — that BORŮVKA's solution is unnecessarily elaborate — is presented also in [ČDF67, p. 32]:

> We will immediately see that this procedure is, in comparison with the others, very complicated, and therefore unsuitable for practical usage, especially for computers. Therefore we will not even prove its correctness.

How wrong ČULÍK, DOLEŽAL, and FIEDLER were only became apparent recently when the minimum spanning tree algorithms were tested

[18] References mine.

[19] O. Borůvka was very much aware of the necessity to publish the results of scientific work, which can be demonstrated also by his strive for a Brno mathematical journal, now known as *Archivum Mathematicum*.[TŠP96, pp. 134–137]

on computers, and BORŮVKA's algorithm came out as the best one. (See [Šiš97, p. 50].)

5.3.3 Łukaszewicz et al., 1949

The paper reporting on the work of a group of Polish mathematicians led by J. ŁUKASZEWICZ gives the following hints on the motivation of the problem [FŁP+51, p. 282]:

> This work is a summary of certain applications of mathematics in natural sciences.

The algorithm presented by ŁUKASZEWICZ is based on the same idea as BORŮVKA's, or more recent PRIM's, algorithm. No reference to other mathematical papers is given, which suggests independent work of ŁUKASZEWICZ and his colleagues. In a footnote, they say [FŁP+51, p. 282]:

> These methods of connection and division were proposed to antropologists for arranging excavated cranes. They were also applied, with effect, to problems in biology, agriculture, technology, and also linguistics. See *Taksonomia Wroclawska*, work of the General Group for Applications of the State Mathematics Institute (Przeglad Antropologiczny, to appear). Cf. there the collective communication *Une méthode taxonomique et ses applications aux sciences naturelles*, this volume, p. 319.

ŁUKASZEWICZ and his colleagues were thus well aware of the practical applications of their method. The method of constructing the minimum spanning tree is described generally, using the terms "polygonal line" ("ligne polygonal"), "segment" ("segment"), or "tree" ("dendrite").

Algorithm: Łukaszewicz et al., 1949

The algorithm begins by connecting each point to the nearest one, which is reminiscent of CHOQUET's note discussed in Section 5.3.1 [FŁP+51, p. 283]:

> We connect into a segment two nearest points: the segments thus obtained are called *lines of the first order*. [20]

[20] The original quotation from the paper by Łukaszewicz et al. are given in the Appendix on page 173.

In the following steps, the distance between the segments of points is defined and the nearest segments are connected. This procedure only differs from that of Borůvka in that a point is only connected to the nearest and no other lines are drawn.[21] After the construction is given, Łukaszewicz et al. present a theorem [FŁP+51, p. 284] saying that the constructed connection of the points satisfies the conditions set for a "dendrite" (i.e. the minimum spanning tree, in contemporary graph–theoretical terminology). The theorem is then proven, which might be seen as the origin for the proof of correctness of an algorithm. The algorithm of Łukaszewicz et al. is essentially equivalent to Prim's 1957 solution [Pri57].

5.4 Towards the Graph–Theoretical Formulation

In the mid–1950s, more graph–theoretical formulations of the minimum spanning tree problem appear. The authors are, however, not always conscious of the graph–theoretical aspect of the problem and the terminology is not yet established. The articles by Kruskal, Prim, Dijkstra, and Loberman and Weinberger all belong to this period.

5.4.1 Kruskal, 1956

Kruskal's algorithm is the most frequently cited one of all the algorithms mentioned here. Kruskal gives only one reference: to Borůvka's paper [Bor26a]. Forty years later, he says in his recollections about the minimum spanning tree problem [Kru97, p. 13]:

> Someone [...] handed me two pages of very flimsy paper stapled together. He told me it was 'floating around the math department'. [...] the pages were typewritten, carbon copy, and in German. [...] I never found out who did the typing or why.

The 'flimsy paper' was the German abstract of Borůvka's article [Bor26a]. Kruskal found the problem challenging, but he said in his paper that he was always looking for easily comprehensible solutions.

[21] Borůvka connects another point to the segment if the weight of the line going to that point (for Borůvka, a number in the appropriate column of the matrix) is less than the line that has just been added. It is probably this formulation that makes Borůvka's algorithm less comprehensible.

His algorithm, indeed, is very easy to understand, but its execution might take longer, especially when the graph contains a greater number of edges.

Algorithms: Kruskal, 1956

KRUSKAL presents three different constructions of the shortest spanning subtrees: A, B, and A'. In fact, construction B is only a special case of A and A' is in "some sense dual to A" [Kru56, p. 49]. Here is KRUSKAL's formulation [Kru56, p. 49]:

> CONSTRUCTION A. Perform the following step as many times as possible: Among the edges of G not yet chosen, choose the shortest edge which does not form any loops with those already chosen. Clearly the set of edges eventually chosen must form a spanning tree of G, and in fact it forms a shortest spanning tree.

The above can be divided into three steps:

Step 1. Sort edges according to their lengths ('distances') from the smallest to the largest.

Step 2. Take edges from the shortest to the longest and decide whether the particular edge is part of the minimum spanning tree or not.

The criterion: Does the edge form a cycle with the previously chosen edges?

Yes: Discard the edge.

No: Add the edge to the minimum spanning tree.

CONSTRUCTION A' consists of *removing* the *longest* edges of the graph, i.e. different sorting of edges is required and the minimum spanning tree is formed by the *not chosen* edges. Construction B is basically the same as construction A, only it is performed on a subset of vertices and it reduces to Construction A when the subset of vertices is the whole set of vertices of the given graph [Kru56, p. 49].

In his article, KRUSKAL also talks about BORŮVKA's algorithm as being in some sense inferior to his own solution: "The construction given in [Bor26a][22] is unnecessarilly elaborate." [Kru56, p. 48] It was probably

[22] Reference mine

just this sentence that caused that other mathematicians did not bother to try and find Borůvka's article, the fact that Borůvka's article was written in Czech being another nuisance. Thus, Borůvka "disappears" to be rediscovered for the mathematical world by Graham and Hell in 1985 [GH85].

5.4.2 Loberman and Weinberger, 1957

The two authors are aware of the equivalence of their solution to the solution of Kruskal, which had been published earlier. In a footnote, they say [LW57, p. 429]:

> This reference [Kru56][23] was discovered by the present authors after their procedures had been formulated. It is seen that the "procedures" presented here and Kruskal's "constructions" are identical. However, it is felt that the more detailed implementation and general proofs of the procedures justify this paper.

Algorithm: Loberman and Weinberger, 1957

The algorithms ("procedures") A and B are presented in *flow diagrams* [LW57, p. 432]: Procedure A as well as Procedure B begin by sorting the edges sequentially in order of increasing length. Loberman and Weinberger start by including the shortest edge into the minimum spanning tree. The edges are then considered in order of increasing length and are (not) included into the minimum spanning tree under the following conditions [LW57, p. 430]:

> *Condition 1.* Neither of the two nodes is present in a subtree. Therefore the branch is made. The two nodes are recorded as constituting another new subtree.
>
> *Condition 2.* Only one of the nodes is present in a subtree. Therefore a branch is made. The new node is added to the subtree containing the other node of this branch.
>
> *Condition 3.* Each of the two nodes is present in a different subtree. Therefore the branch is made. The two different subtrees, each containing one of the nodes, are combined into a single subtree.

[23] Reference mine.

Condition 4. Both nodes are present in the same subtree. Therefore the branch is *not* made.

LOBERMAN and WEINBERGER were solving a practical problem in their paper, or at least practical problems were the source of their problem [LW57, p. 428]:

> In the construction of a digital computer in which high-frequency circuitry is used, it is desirable and often necessary when making connections between terminals to minimize the total wire length in order to reduce the capacitance and delay–line effects of long wire leads.

Despite the practical motivation behind this paper, the formulation of the solution is quite detailed and the explanations clear. It includes examples as well as general statements. The same, except for clarity, is true for another practically oriented paper [Pri57].

5.4.3 Prim, 1957

ROBERT C. PRIM found the minimum spanning tree problem also challenging. In his article [Pri57], he gives a solution which is, at least according to him, better than KRUSKAL's. In fact, his approach to the minimum spanning tree problem reminds one of BORŮVKA's solution.

Algorithm: Prim, 1957

PRIM states two *principles* for finding the minimum spanning tree of a graph [Pri57, p. 1391]:

P1: Any isolated terminal can be connected to the nearest neighbour.

P2: Any isolated fragment can be connected to the nearest neighbour.

Following these two principles might result in formulating *either* BORŮVKA's, *or* JARNÍK's algorithm, depending on the choice of the third step (see Sections 5.3.1 and 5.3.2).

Like BORŮVKA, PRIM also presents a *distance matrix* for his solution. As PRIM says, $(n-1)$ applications of either of the principles give the desired result. The only complication is that we have to compute the distance to the nearest neighbour anew every time.

Looking at the history of the problem, it is interesting to note that although PRIM gives the reference to BORŮVKA, he apparently had not

read the article. He says: [Pri57] "Kruskal refers to an obscure Czech paper as giving a construction inferior to his." However, Borůvka's technical paper [Bor26b] presents the same idea as Prim's article [Pri57].

5.4.4 Dijkstra, 1959

If Prim's algorithm can be described as being very similar to that of Borůvka or Jarník, then Dijkstra's solution is equivalent to Jarník's.

Algorithm: Dijkstra, 1959

Dijkstra uses three sets of edges and two sets of vertices. The sets of edges are denoted I, II, and III, the sets of vertices as A and B ('MST' stands for "minimum spanning tree"):

I. MST edges,
II. edges from which the next MST edge will be chosen, and
III. not yet considered or rejected edges.
A. vertices connected by edges from I, and
B. remaining vertices.

To begin with, set I is empty, and set A contains one arbitrarily chosen edge. The algorithm ends when the set B is empty, i.e. when no unconnected vertices remain.

Initialization: Choose any vertex u as a starting point — the only member of set A. The set II is the set of all edges going out of u.

Step 1. Choose the shortest edge from set II; this is the first edge of set I (in which the edges of the minimum spanning tree are gathered).

Step 2. Consider edges going from the vertices of the chosen edge and adjust set II: Always choose the shortest edge going from vertices in set A to vertices in set B.

Repeat Steps 1 and **2** until set B is empty.

Like Jarník, Dijkstra starts from one point and 'spreads' the network across the set of vertices (numbers denoting elements, for Jarník). Also, there is some similarity between Prim's and Dijkstra's algorithm. However, Dijkstra cites neither Jarník's, nor Prim's solution.

After describing the algorithm for the minimum spanning tree problem, he remarks that "the solution given here is to be preferred to the

solution given by JOSEPH B. KRUSKAL [Kru56] and H. LOBERMAN and A. WEINBERGER [LW57]" and gives reasons why.

5.5 Generalisations and Extensions

Once we know several good solutions to the minimum spanning tree problem, it seems as if the problem is "dead". There are, however, several directions in which we can pose new problems: the minimum spanning tree problem can be extended also to directed graphs (and the result is then called and "arborescence"); the solution of the minimum spanning tree may not be unique when the edge weights are not mutually different; the connectedness of the minimum spanning tree can be destroyed by removing any one of the edges; and finally, there are better implementations of the existing algorithms which might significantly reduce computing time for graphs of greater size.

5.5.1 Uniqueness of solution

In 1961, the Slovak mathematician ANTON KOTZIG published two articles on the minimum spanning tree problem. In the first one, he generalizes the minimum spanning tree problem also for graphs whose edge lengths are not mutually different.

The uniqueness of the minimum spanning tree problem solution

KOTZIG notes on the practical usage of his results [Kot61b, p. 2]:

> Note 1. The presented generalization is of no practical importance, since the opposite was done by O. BORŮVKA as a simplification of the original problem.

KOTZIG gives his construction in a theorem [Kot61b, p. 3]:

> Let G be a connected graph with p vertices and q edges. Let δ be some mapping assigning weight to the edges of G. Let $m = q - p + 1$. We form a sequence of subgraphs of G: $G_0, G_1, \ldots, G_m$ and a sequence of edges $\in G$ in the following way: The edge h_i is any arbitrarily chosen edge from $\overline{H}(G_{i-1})$.

This is basically KRUSKAL's construction A', of which KOTZIG is fully aware:

> The construction of graph G_m presented in Theorem 1 is an appropriately adapted Kruskal's construction A'.

KOTZIG then formulates the necessary and sufficient condition for the uniqueness of the minimum spanning tree [Kot61b, p. 4]:[24]

Theorem 5.4 (Kotzig, 1961) *Let G be a connected weighted graph and G_0 any chosen subgraph of G with the properties (α) — it contains all the edges of G, (β) — is connected, (γ) — is minimal for the given weight function on G. Let $H_0 = \{h_1, h_2, \ldots, h_n\}$ be a set of all such edges $\in G$ that do not belong to G_0. We construct the set of cycles $\overline{K} = \{K_1, K_2, \ldots, K_n\}$ in the following way: The cycle $K_i \in \overline{K}$ is such a cycle in graph G that contains the edge $h_i \in H_0$ and all its other edges belong to the set G_0.It holds that in the graph G there exists only one subgraph with the properties (α), (β), (γ) if and only if for every $i \in \{1, 2, \ldots, n\}$ the following condition holds: The weight of the edge h_i is greater than any other edge in K_i.*

5.5.2 Reliability: spanning trees of higher order

In the article [Kot61a] on spanning trees of higher order, KOTZIG generalises the minimum spanning tree problem. The notion of the *degree of connectivity of vertices*, defined below, plays a crucial role in his paper:

Definition 5.5 *We say that the* connectedness *between two vertices u, v of a graph G is of* degree *k if and only if there exists in graph G such set of k of edges that if all the edges of this set are left out, we obtain a graph in which the vertices u, v are not connected; if we remove any $k-1$ edges of this graph, we always obtain a graph in which the vertices u, v are connected.*

Spanning tree of k–th order is then defined as a subgraph which remains connected after removal of any $k-1$ edges and is disconnected after the removal of k edges. It is evident that the *spanning tree* as defined earlier in this chapter is a *spanning tree of first order*.

In the last part of this article, KOTZIG tries to construct a minimum spanning tree of k–th order. However, as he demonstrates on an

[24] The condition that the edge weights of the graph are mutually different is sufficient, but not necessary. For example, we can construct a graph in the following way: take a minimum spanning tree of a graph and add edges with weight greater than the maximum edge weight of the minimum spanning tree.

example, an analogy of KRUSKAL's algorithm does not work even for spanning trees of second order.[25]

5.5.3 Arborescences: directed spanning trees

In the abstract of their article on *shortest arborescences* [CL65, p. 1396], CHU and LIU introduce their generalisation of the minimum spanning tree problem:

> In many practical problems, we should consider not only the "line–segment" but also the "directional line–segment". For example, we may prepare a scheme of routes to a canal under certain conditions.

An *arborescence* is defined only in a directed graph. If we omit the direction of the edges, the resulting graph is a spanning tree. A precise definition is presented below.[26]

Definition 5.6 *An* arborescence *of a directed graph $G(V, E)$ is a subgraph $H(V, A)$ $(A \subseteq E)$ of G which contains no cycle such that*

(a) *there is a particular vertex called the* root*, which is not a terminal vertex of any arc in A.*

(b) *For any vertex x_j, there is one and only one arc in A, whose terminal vertex is x_j.*

For the existence of a minimum spanning tree in a graph, it is enough to require *connectedness* of a graph. The search for an *arborescence* in a graph, however may not always be successful.

The algorithm given by CHU and LIU consists of four steps. They not only give the general rules for constructing an arborescence, they also show how the algorithm works on a specific example.

Note: The notation in the following quotation is identical with the original: the set of vertices is denoted by X and the set of edges by U, (directed) edges between the vertices v_i, V_j are denoted by (v_i, v_j); the term "loop" here stands for "(oriented) cycle". The set $U^-(x_i)$ denoted the edges going into x_i:

$$U^-(x_i) = \{(x_j, x_i) | (x_j, x_i) \in U, j = 1, 2, \ldots, n\}.$$

The solution given in [CL65, pp. 1397–1398] follows.

[25] This is rather common in discrete optimization problems: a slight change in the task causes much greater difficulty with the solution.

[26] This definition is identical with the one presented by Chu & Liu in their paper, only the notation is partly different.

Algorithm: Chu and Liu, 1965

Step 1. For any vertex x_i whose $U^-(x_i) \neq \emptyset$, take the arc u_{ij} such that

$$l(u_i) = min_{u \in U^-(x_i)} l(u)$$

(if such arcs are more than one, take one arbitrarily). The set of these arcs is designated by W_0. [...]

Step 2. If $|W_0| < n-1$, then the process stops, and there is no arborescence of G; if $|W_0| \geq n-1$, then we choose $n-1$ arcs in $|W_0|$ and the set of these $n-1$ arcs is designated by V_0 such that

$$max_{a \in V_0} l(a) \leq max_{b \in W_0 - V_0} l(b).$$

The choice may not be unique. [...]

Step 3. If there is no loop in V_0, then the process stops and $H_0 = \{X, V_0\}$ is the shortest arborescence of the graph; otherwise, there exist some loops in V. [...] We retract [the loops into single vertices] and the graph obtained from the graph G by such a retraction is designated by $G_1(X_1, U_1)$. The length of the arcs of G_1 is redefined.

Repeat steps 1, 2, 3 until the process stops. If the process stops at Step 2, the arborescence does not exist; otherwise the process stops at Step 3 and the arborescence of the relevant retracted graph G_p exists.

Step 4. The arborescence G_p obtained in Step 3 is now extended, until the shortest arborescence of G is obtained.

5.5.4 Implementations

The article [Yao75] by YAO presents an adapted version of an algorithm attributed to SOLLIN, which is basically the same as BORŮVKA's algorithm. Apart from the expected form of presentation of their algorithm in the programming language *Pascal*, we can notice several new aspects in the discussion of the algorithm:

1. Complexity measure is used throughout the article, without being defined. It should be noted that this happens only ten years after the first ideas about the necessity to distinguish between "good" and "bad" algorithms appeared in [Edm65].

2. The term "sparse graph" is used for graphs containing relatively few edges and a distinction is made between the algorithm with more than $|V| \log |V|$ edges and with less than $|V| \log |V|$ edges.

3. References are made to special procedures and data structures: e.g. "median–finding algorithm" or "circular queue".

A similar tendency can be seen also in other papers on the minimum spanning tree since the beginning of the 1970s.

5.6 Conclusion

As the first mathematical formulations and solutions of the minimum spanning tree problem were presented by Czech mathematicians OTAKAR BORŮVKA and VOJTĚCH JARNÍK, it is probably the most famous combinatorial optimization problem for Czech mathematicians. The tradition was followed also by KAREL ČULÍK, a Czech graph theorist, and ANTON KOTZIG, a Slovak graph theorist. It is often pointed out that the two early solutions – by BORŮVKA and JARNÍK — are often neglected, although they are both very good. One of the reasons might be the fact that both were written in Czech, but as another paper – by a group of Polish mathematicians lead by ŁUKASZEWICZ, but written in French – also seems a bit neglected, this reason seems unsatisfactory.

The algorithms described in Sections 5.3 and 5.4 all deal with the same problem, the minimum spanning tree, and all give the same final result. However, the computing time for the algorithm is not the same, and neither is the time for "updating solution". Similarly, the general idea of the algorithm varies. It is worth noticing, though, that similar solutions were found by BORŮVKA, ŁUKASZEWICZ and PRIM, by KRUSKAL and LOBERMAN and WEINBERGER, and by JARNÍK and DIJKSTRA (and also PRIM, in a sense). What is even more surprising, there are four terminologies used: matrix terminology by BORŮVKA, set–theoretical by JARNÍK, geometrical by ŁUKASZEWICZ, and graph–theoretical by KRUSKAL, PRIM, LOBERMAN and WEINBERGER, DIJKSTRA (and of course the more recent authors: KOTZIG, ČULÍK, CHU and LIU, and others).

Chapter 6

Travelling Salesman Problem: Accuracy versus Speed

The travelling–salesman problem differs from the problems dealt with in the previous chapters in one important aspect, in the aspect of complexity. No polynomial algorithm is known for solving the travelling salesman problem. The existing algorithms are thus not very useful for problems on graphs with a large number of nodes. In this chapter, an overview of the history of the problem, starting in the 1930s and finishing in the 1960s, will be presented. From the historical point of view, the most interesting features of this problem are the development of algorithms (or rather heuristics) and the manner in which the efficiency of the algorithms (heuristics) is measured. Computers obviously play a role in this development.

Since World War II, computers have gained in importance – and have also undergone a rapid development. This can also be demostrated on the example of the travelling–salesman problem: the measures for efficiency of the solution changed significantly. Another major shift in the approach to the travelling–salesman problem is the preference of approximate solutions to the exact ones. It no longer seems profitable to look for the best solution: in other words, fast ways of finding "good" solutions are preferred to (slow) finding of the "best" solution. The changes in the approach to the travelling–salesman problem reflect the change the use of computers and programming languages has brought to mathematics: the change from the *Theorem–Proof* method to the *Algorithm–Analysis* method.

Before poining out the differences between the problems described in the previous chapters and the travelling–salesman problems, we will focus on two similar problems in unweighted graphs. The relationship between the formulation of the problem and the difficulty of the solution in combinatorial optimization problems can be demonstrated on Eulerian and Hamiltonian graphs.

6.1 Eulerian and Hamiltonian Graphs

In 1735, LEONHARD EULER gave his solution to the *problem of seven bridges of Königsberg* [Eul36, translation taken from [BLW76], p. 3]:[1]

> The problem that should be well known was the following: in Königsberg in Prussia is an island A, named "der Kneiphof", and the river that flow around it, divides into two branches [...]. There are seven bridges over the river, *a, b, c, d, e, f* and *g*. The question now is, whether somebody can plan his walk is such a way that he crosses every bridge once and not more than once.

EULER's answer was, obviously, negative. However, as a by–product, he stated the necessary and sufficient conditions for a finite graph to be *Eulerian*;[2] however, he uses a different terminology. EULER first presents his description of a journey, stating that it is in a certain sense irrelevant whether we use the bridge a or the bridge b. He then states that each letter denoting an island must appear in the description either $\frac{n}{2}$ times for an even number of bridges incident with the island, or $\frac{n+1}{2}$ times for an odd number of bridges incident with the island. In such a way, we can count how many times each of the letters denoting an island appears in the sequence and by comparing the sum of them with the number of bridges we get the answer to the problem. EULER starts from a given problem, arrives at the *general* conclusion, and finally proves that the desired walk over the Königsberg bridges is impossible.

EULER also mentions one of the crucial features of combinatorial analysis problems. He says [Eul36, translation taken from [BLW76]]:

> As far as the problem of the seven bridges of Königsberg is concerned, it can be solved by making an exhaustive list of all possible routes, and then finding whether or not any

[1] For the picture of Königsberg bridges, see Chapter 1.
[2] For definition of Eulerian graph, see Chapter 2.

> route satisfies the conditions of the problem. Because of the number of possibilities, this method of solution would be too difficult and laborious, and in other problems with more bridges it would be impossible.

This describes the very aspect of the difficulty of solving the travelling salesman problem: we are certain that there is a solution, but finding the best one might take too much time. EULER managed to answer the question as to which graph is *Eulerian*; however, nobody has managed to find a similar condition for a graph to be *Hamiltonian* so far.[3]

In 1857, SIR WILLIAM ROWAN HAMILTON sold his *Icosian game* to a wholesale dealer in games and puzzles for £25. In 1859, the game was sold with a leaflet with instructions, whose principal author was W. R. HAMILTON himself. The game consisted of a wooden board with holes, with a letter assigned to each of them, and numbered pegs. The board was a certain diagram of *icosaedr* (Figure 6.1):

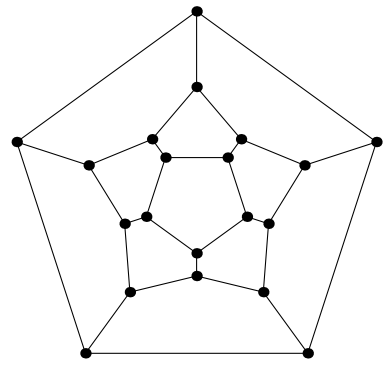


Figure 6.1: Icosaedr

The first problem suggested by HAMILTON was the following [BLW76]:

> *Five* initial points are given: cover the board and finish *cyclically*. (As hinted on the preceding page, a succession is said to be *cyclical* when the *last* piece is adjacent to the *first*).

Translated into graph–theoretical language, this means finding such a *cycle* in the graph which contains all the vertices of the graph. For the *Icosian Game*, one such solution is shown on Figure 6.2.

[3] Apparently the reason is not that there is no "Euler" to solve it.

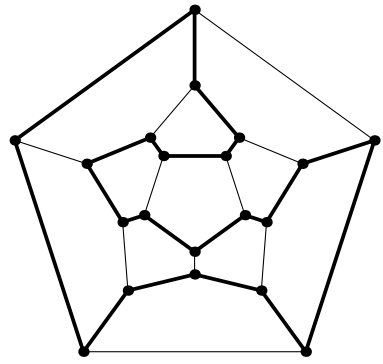


Figure 6.2: The Icosian Game: Solution

The solution to HAMILTON's first problem is called *Hamiltonian cycle* and a graph in which such a cycle exists is called *Hamiltonian.* As is obvious from Figure 6.2, HAMILTON's game has a solution; however, the necessary and sufficient condition for the graph to be Hamiltonian is not known. There are only sufficient conditions, namely the conditions of DIRAC, ORE,and PÓSA:

Theorem 6.1 (Dirac,1952) *Let $G(V, E)$ be a finite graph, $|V| \geq 3$, and for the degrees of vertices $deg(v_i)$, it holds that $deg(v_i) \geq \frac{n}{2}$. Then G contains a* Hamiltonian cycle.

Theorem 6.2 (Ore,1961) *Let $G(V, E)$ be a finite graph, $|V| \geq 3$. Let for any two not neighbouring vertices v_i, v_j $deg(v_i) + deg(v_j) \geq |V|$. Then G contains a* Hamiltonian cycle.

Theorem 6.3 (Pósa, 1962) *Let G be a finite graph, $|V| \geq 3$ such that*

(1) *for each k such that $1 \leq k \leq \frac{n-1}{2}$, the number of vertices of degree $\leq k$ is $< k$.*

(2) *(if n is odd) the number of vertices of degree $\leq \frac{n-1}{2}$ is $\geq \frac{n-1}{2}$.*

Then G contains a Hamiltonian cycle.

The minimum spanning tree problem and the travelling–salesman problem can be compared to the two above–described problems in terms of difficulty of finding a solution. It should be emphasised that the two problems — the minimum spanning tree and the travelling–salesman

problem — always have a solution.[4] Nevertheless, whereas the algorithms for the finding minimum spanning tree in a given graph can be quite simple and still efficient, this is not the case for the travelling–salesman problem, for which a polynomial algorithm does not exist.

It should also be noted that the total number of solutions in a complete (weighted) graph with n vertices is n^{n-2} for the minimum spanning tree[5] and $\frac{(n-1)!}{2}$ for the symmetric travelling–salesman problem. The amount of time consumed by trying out all the solutions grows faster with growing number of vertices for the minimum spanning tree problem than for the travelling–salesman problem; yet efficient algorithms for finding the minimum spanning tree are known.[6] The exact algorithms for the travelling–salesman problem would take too long for a larger number of vertices (cities), and so inexact heuristics are designed to solve the problem.[7]

When judging the efficiency of the algorithm or heuristics, we are interested in the *time* it consumes and, in the case of heuristics, in the "degree of optimality" of the solution.

The *asymptotic complexity* of an algorithm (heuristics) is nowadays commonly given as a measure for its efficiency. It is generally assumed that algorithms which require *polynomial* number of steps are *good* — the complexity of such algorithms is $O(n^k)$, where k is a finite number. The difficulty with the travelling salesman problem is that no polynomial algorithm has been found (and most probably does not exist). That is why we have to retreat to inexact heuristics, which nevertheless work in "good" time.

6.2 Formulations of the Problem

The formulations of the travelling–salesman problem differ for different authors. The most general one could be stated as follows:

> A salesman wants to visit certain towns in his area and he wants to do so as fast (or as cheaply) as possible. The task

[4] The reasons are obvious: there are only finitely many possible solutions to these problems. (For the travelling–salesman problem, the formulation not requiring a Hamiltonian cycle must be used.)

[5] This result is due to A. Cayley, 1889.

[6] See Chapter 5 for details.

[7] *Heuristics* is a method for solving problems. The method consists of trying several procedures and, after each step, questionning whether we got any nearer to the solution. The heuristics may also end after a given number of steps, divulging several possible (not necessarily best) answers.

is: in what order should he visit the towns in order to spend the least time (money)?

In graph–theoretical language, this could be interpreted in two ways:

1. Find such ordering of the vertices of a graph $G(V, E)$ with n vertices that every vertex $1, \ldots, n$ is visited *at least* once.

2. Find such ordering of the vertices of a graph $G(V, E)$ with n vertices that every vertex $1, \ldots, n$ is visited *exactly* once.

It can easily be seen that in the first case, the tour does not have to form a Hamiltonian cycle, whereas if the second formulation is used, the resulting tour must form a Hamiltonian cycle. The travelling–salesman problem thus has basically two interpretations: one of them allows only solutions in the form of Hamiltonian cycles, the other accepts also other solutions. However, it seems that the former interpretation quickly gained popularity, whereas the latter was only used in the beginning, as the formulations of the problem by different mathematicians, listed below, show:

1. DANTZIG, FULKERSON and JOHNSON, 1954 [DFJ54]: "Find the shortest route (tour) for a salesman starting from a given city, visiting each of specified group of cities, and then returning to the original point of departure." This formulaiton does *not* require a Hamiltonian cycle as a solution.

2. FLOOD, 1956 [Flo56]: "The traveling salesman problem is that of finding a permutation $P = (1i_2i_3 \cdots i_n$ of the integers from 1 through n that minimizes the quantity $a_{1i_2} + a_{i_2i_3} + \cdots + a_{i_n1})$, where the $a_{\alpha\beta}$ are a given set of real numbers. More accurately, since there are only $(n-1)!$ possibilities to consider, the problem is to find an efficient method for choosing a minimizing permutation." This formulation *does* require a Hamiltonian cycle.

3. LITTLE, MURTY, SWEENEY and KAREL, 1963 [LMSK63]: "The travelling–salesman problem is easy to state: A salesman, starting in one city, wishes to visit each of $n-1$ cities once and only once and return to the start." Again, the solution must form a Hamiltonian cycle in the graph.

4. KARG and THOMPSON, 1964 [KT64]: "Let $A = ||a_{ij}||$ be an $n \times n$ matrix of real numbers. The travelling–salesman problem asks

for an acyclic permutation $(i_1, i_2, \ldots, i_n)$ of the integers $1, 2, \ldots, n$ such that the sum $a_{i_1 i_2} + a_{i_2 i_3} + \cdots + a_{i_n i_1}$, is a minimum." This again results in Hamiltonian cycles as the only feasible solutions.

5. GILMORE and GOMORY, 1964 [GG64]: These two authors do not define the travelling–salesman problem. They only say: "Our problem [sequencing one state-variable machine] is closely connected to the well–known and difficult Traveling Salesman problem. [Ref. to [Ackoff 1961], [HK62], [LMSK63].] To see this, let the J_i play the role of nodes or cities, and let the c_{ij} of [equation 1] be the cost of going from node i to node j. We are looking for the cheapest path that passes once through each node. The traveling salesman looks for the cheapest path that passes once through each node and ends up at the starting point. He looks for the cheapest tour."

   Thus, it is not emphasised that we are looking for a Hamiltonian cycle. In the paper [LMSK63], the authors *are* looking for Hamiltonian cycles.

6. SHEN LIN, 1965 [Lin65]: "A salesman is required to visit each of the n given cities once and only once [...]". The solution yields a Hamiltonian cycle.

7. OBRUČA, 1968: "Let us define a *network* N, (of n points and m lines) to be a graph with costs/distances to be associated with each line. [...] A *chain* is a sequence of adjacent lines. A *circuit* is a chain such that the first line is adjacent to the last. We define a *feasible* solution to the travelling salesman problem as any circuit which covers the network. An *optimal* solution will be a feasible one with least cost." It is debatable whether the author requires a Hamiltonian cycle as a result. From the definitions, it is not clear; from the rest of OBRUČA's article, however, one can infer so.

8. BELLMORE & NEMHAUSER, 1968 [BN68] (Survey Article): "In the traveling salesman problem we are given a nonnegative integer n and an n-dimensional square matrix $C = \{c_{ij}\}$. Any sequence of $p + 1$ integers taken from $(1, 2, \ldots, n)$, in which each of the n integers appears at least once and the first and last integers are identical is called a tour. (...) By a feasible solution to the traveling salesman problem, we mean a tour. [...]" The required solution is a Hamiltonian cycle.

9. PAPADIMITROU and STEIGLITZ, 1982 [PS85]: the solution of the travelling salesman problem requires that each town is traversed exactly once; in other words, a Hamiltonian cycle is the only acceptable solution.

10. PLESNÍK [Ple83] views the travelling salesman as a generalization of the problem of finding a Hamiltonian cycle for weighted graphs, i.e. a solution that is not a Hamiltonian cycle is out of the question.

The travelling–salesman problem can be specified in the following way:

1. Symmetric travelling–salesman problem: the distances between any two cities are independent of the direction of the travel.

2. Non-symmetric travelling–salesman problem: the distance between two cities v_i, v_j may be different if we travel from v_i to v_j and from v_j to v_i.

3. Euclidean travelling–salesman problem: the distances between cities are given as distances in the plane. These distances satisfy the *triangle inequality.*

It is apparent that the number of possible solutions $\frac{(n-1)!}{2}$ holds only for the Euclidean or symmetric travelling–salesman problem. For the asymetric travelling–salesman problem, the number of solutions is even greater, namely $(n-1)!$.

6.2.1 Solutions

As was already metioned, the travelling-salesman problem is a difficult one. (Generally NP-complete). As a result of that, mathematicians usually tried to find solutions of some variations on the problem, special cases, etc. The connection with the *assignment problem* is mentioned quite often.

There are special cases of the problem: symmetric, non-symmetric, euclidean, travelling–salesman problem with the triangle inequality, and so on. The solutions vary accordingly: the approaches are very different.

The results up to 1968 were collected and commented upon by BELLMORE and NEMHAUSER. In 1982 the book by PAPADIMITROU and *Steiglitz* [PS85] was published, containing also other approximate methods.

6.2.2 Polynomial algorithm – approximation

For a long time, the attempts at designing good heuristics for the travelling salesman problem kept arriving at a seemingly unsurpassable boundary. The upper bound of most of the solutions was 200% of the optimum. In the 1970s, this was greatly decreased by NICOS CHRISTOFIDES, who designed an algorithm with the upper bound at 150% of the optimum.

The best approximation algorithm up to 1982 is, according to PAPADIMITROU and STEIGLITZ, the algorithm by CHRISTOFIDES [PS85].

Christofides: 1976

1. Find the minimum spanning tree T for the distance matrix $[d_{ij}]$.
2. In the minimum spanning tree T, find the vertices of *odd degree* and find the shortest complete matching M in the complete graph containing only those vertices. Let G be a multigraph with vertices $1, 2, \ldots, n$ in which all edges of T and M are included.
3. Find an Eulerian path in G and the respective circuit.

This algorithm is polynomial: the first step is finished in $O(n^2)$, the second step (minimum [covering] matching) takes $O(n^4)$ or $O(n^3)$ (when another matching algorithm is used), the third and last step runs in linear time. The important fact is that the algorithm gives a solution that is at most 50% larger than the optimum.

"Locally-based method"

This method is based on the oldest and probably most natural optimization method, the *trial-and-error method*. PAPADIMITROU and STEIGLITZ formally describe this basic trial-and-error method [PS85, p. 467]:

IMPROVEMENT (t) = any $s \in N(t)$, for which $c(s) < c(t)$, if such s exists; or 'no' in the opposite case

The base for the following algorithm is any feasible solution. The algorithm stops when we reach a local optimum.

```
procedure LOCAL METHOD
begin
    t := some initial point of F
    while IMPROVEMENT(t) ≠ "NO" do
        t := IMPROVEMENT(t);
    return t
end
```

Definition 6.4 k-exchange ($k \geq 2$) *for any tour f:*

$N_k(f) = \{g | g \in F$ *and* g *can be obtained from* f *by choosing* k *edges of* f *and substituting them with other* k *edges.*$\}$

In 1958 there appeared a solution of the travelling–salesman problem based on this method and some other methods similar to the branch–and–bound methods by CROES [Cro58].

6.3 Origins of the Travelling Salesman Problem

The motivation for the travelling–salesman problem is, as suggested above, primarily economical. The first formulation of the problem is — allegedly — due to the American mathematician HASSLER WHITNEY,who is said to have formulated it in a seminar in 1934. Two of his students, ALAN W. TUCKER and MERRILL M. FLOOD, agree on this; WHITNEY himself, however, does not remember it [DFJ54, see below]. In addition, it is not clear what WHITNEY's formulation was.

In 1937, MERRILL FLOOD was trying to find the cheapest route for a school bus. ALAN W. TUCKER brought his attention to the connection between the travelling–salesman problem (in FLOOD's case, the school–bus routing) and Hamiltonian cycles. The task MERRILL FLOOD was trying to solve was the following:

> Design the cheapest route for a school bus, which has to collect children from a certain area, making given stops, so that all children get picked up in the morning (and get home in the afternoon) and the costs are minimum.

This formulation implies that each place should be visited *at least* once, not *exactly* once.

DANTZIG, FULKERSON and JOHNSON say — in their "historical note" [DFJ54, p. 393]:

> The origin of this problem is somewhat obscure. It appears to have been discussed informally among mathematicians at mathematics meetings for many years. Surprisingly little in the way of results has appeared in the mathematical literature. It may be that the minimal–distance tour problem was stimulated by the so-called Hamiltonian game which is concerned with finding the number of different tours possible over a specified network. The latter problem is cited by

> some as the origin of graph theory and has some connections with the famous Four–Color Conjecture. Merrill Flood (Columbia University) should be credited with stimulating interest in the traveling salesman problem in many quarters. As early as 1937, he tried to obtain near optimal solutions in reference to routing of school buses. Both Flood and Tucker (Princeton University) recall that they heard about the problem first in a seminar talk by Hassler Whitney at Princeton in 1934 (although Whitney, recently queried, does not seem to recall the problem). The relations between the traveling–salesman problem and the transportation problem of linear programming appear to have been first explored by M. Flood, J. Robinson, T.C. Koopmans, M.Beckmann, and later by I. Heller and H. Kuhn.

According to Morton and Land, [ML55] the term "travelling salesman problem" was coined by Julia Robinson in 1949:

> In the United States this problem is known as the Travelling–salesman problem; the salesman wishes to visit one city in each of the 48 States and Washington D.C., in such a sequence as to minimize total road distance travelled. Thinking of the problem independently and on a smaller geographical scale, we used to call it the laundry van problem, where the conditions were a daily service by a one–van laundry. Since the American term was used by Robinson (1949) we propose to adopt it here.

The paper Morton and Land refer to is the RAND Corporation Paper RM–303 [Rob49]. They also speak about similar problem in statistics: the "Minimum Mean Distance" problem.

They also — consciously — do not use graph terminology, but "point" for "node" and "link" for "edge". They were aware of the existence of graph theory, but their contribution was communicated at an international conference on linear programming, which might have been the reason for choosing a different terminology.

The travelling–salesman problem became especially popular after World War II. It should be noted that the minimum spanning tree problem also became popular after the war, although the first solutions appeared in the period before World War II. [Dur98, Dur99]

6.4 The First Results

At first, mathematicians tried to find exact solutions to the travelling–salesman problem. A classic paper is the one written by DANTZIG, FULKERSON, and JOHNSON in 1954 [DFJ54]. In their paper, they give an exact solution to a certain "42–city problem". They were actually looking for the shortest route through 49 cities in the U.S.A. (one in each of the 48 states of the U.S.A. (mainland) and Washington D.C.), but they reduced the size of the problem by finding such a shortest path between two cities that other 7 cities were on the shortest path between the two. That is why we refer to the "42-city problem (of Dantzig, Fulkerson, and Johnson)" today.

In 1956, FLOOD published his paper on the travelling–salesman problem. He presents the *method of the nearest neighbour*, which resembles the *greedy algorithms* used for the minimum spanning tree problem.[8]

The "nearest-neighbour" approach

Step 1. Choose any two vertices.

Step 2. From this arbitrary tour, go to the nearest not yet visited vertex.

Repeat Step 2 until all vertices have been visited.

Unfortunately, the greedy strategy does not always work for the travelling–salesman problem. However, it can be used to determine the *starting tour* for the algorithms mentioned later.

In his 1956 paper [Flo56], MERRILL FLOOD also draws attention to another problem related to the travelling–salesman problem: the *assignment problem*. The task is to optimize the distribution of work among people (each person is assigned one job and it is known how "efficient" each person is in each of the jobs) so that the whole bulk of work is done as fast as possible. It is apparent that the solution to this problem need not be a Hamiltonian cycle. However, if the solution to the assignment problem happens to be a Hamiltonian cycle, it is also a solution to the travelling–salesman problem. The solution of the assignment problem is also useful as a starting point for solving the travelling–salesman

[8] The so-called *greedy algorithms* are notable for their special feature: although they use "greedy strategy", i.e. the algorithm always chooses the *locally optimal* solution, they solve the problem successfully. They are also described in the Chapter 2.

problem.[9]

A variation on this problem is the *marriage problem*, where the objective is to maximize the amount of happiness in a certain group of people:

> We are given n young men and n young women who want to get married. How should they be coupled so that the amount of happiness in the whole group is maximum? (For each couple, we are given a number determining the "happiness" that occurs when these two people get married.)

This problem can also be linked to *matching theory*, where it is solved for unweighted graphs. In that case, the question is whether it is possible to couple all the people, when it is known that some couples hate each other.

6.5 Heuristics: Travelling Salesman Problem

As no "good" algorithms for the travelling–salesman problem were found,[10] it became necessary to look for other methods for solving this problem. The methods providing at least some "reasonably good" solution are called *heuristics*[11]. The heuristics can be broken up into three phases:

1. Starting point
2. The method for generating the solution

[9] Fulkerson talks about something similar when he talks about the *production cost curves*. For the production cost curves, the normal completion time and the crash completion time are given and the task is to optimize the sequence of the individual tasks. [Ful61]

[10] The meaning of the adjective "good" is used in the sense advocated by [Edm65, p. 450]:

> An explanation is due on the use of the words "efficient algorithm." First, what I present is a conceptual description of an algorithm and not a particular formalized algorithm or "code."
>
> For practical purposes computational details are vital. However, my purpose here is to show as attractively as I can that there is an efficient algorithm. According to the dictionary, "efficient" means "adequate in operation or performance." This is roughly the meaning I want — in the sense that it is conceivable for a maximum matching to have no efficient algorithm. Perhaps a better word is "good."

[11] See Chapter 2 for definition of the concept

3. The condition for ending the procedure

The last phase – the condition for terminating the procedure – tells us whether the heuristics gives an exact, or an approximate solution. The two conditions can be described as *absolute* and *relative*. As for the former, the process terminates when the optimal tour is found; as for the latter, the process terminates when a tour is found that cannot be shortened by interchanging two vertices in the sequence.

It should be emphasised that each travelling salesman problem can be transformed into the travelling salesman problem on a complete graph: if we substitute the lengths of edges between two vertices by the shortest–paths distances, the problem is transformed into a problem on a complete graph with fewer vertices.

The heuristics for the travelling salesman problem can be divided into three categories: (a) tour–to–tour improvement; (b) building the tour; and (c) subtour elimination. They are described in a greater detail in the following sections.

6.5.1 Tour–to–tour improvement

As the starting point of this method, we take any tour. When generating the solution, we are trying to find a neighbouring tour which is better than the current one. The neighbouring tour is generated by interchanging two vertices in the current tour.

These methods are only approximate and are judged according to the ratio between the quality of the solution and the time consumed.

6.5.2 Tour building

The starting point in this method is any vertex. From this vertex, a sequence of vertices is constructed. The procedure ends when a cycle is found. This method is also approximate. A special modification of this method (and also one easily comprehensible) is the nearest-neighbour method.

6.5.3 Subtour elimination

These methods use the connection between the assignment problem and the travelling salesman problem. If the solution of the assignment problem is a cycle, it is also a solution to the travelling salesman problem. If not, it is necessary to eliminate the partial tours.

Some of these methods are exact: integer linear programming and the branch–and–bound methods.

6.6 Approximate Solutions

The goal of more recent programmers was to find a "reasonably good" solution in a "reasonably short" time. For a long time, it was thought that the upper bound of twice the optimum will not be surpassed. In 1975, however, NICOS CHRISTOFIDES came with a $O(n^3)$ algorithm that gives a solution which is at most 150% of the optimum.

6.7 Evaluation of the Solutions

In the 1950s, the methods were judged according to the actual time spent. In the 1960s, a new way of judging appeared: the complexity measure. The results of the evaluation, as presented in the earlier papers, are given in Table 6.1. It is necessary to note that the IBM 7090 is about five times faster than the IBM 1620.

A very different approach was used by OBRUČA in 1968 [Obr68]: in his approximate method, he makes use of the observation that arcs forming the minimum spanning tree are often used in the best solutions to the travelling salesman problem. He calls his technique *spanning tree manipulation*. He tested his results on a large number of small problems and on some large published problems. Evaluation of his results is given in Table 6.2.

6.8 Conclusion

The travelling–salesman problem is a very popular one. It gained popularity again in the 1960s, when a soap company organized a contest. The task was to find the shortest way connecting 33 cities, the prize ten thousand U.S. dollars. Some people came with a correct solution, whereas others claimed that the problem has no solution. This was a mis-interpretation of the fact that there is no polynomial algorithm for solving the travelling salesman problem; wihch, obviously, does not mean that there is no solution.Considering the computation results, it is important to realize that in "reality" it does not help one much to know that something could be solved in a million years. "In reality", this is almost the same as "never". However, there surely is a signifi-

Table 6.1: Results of the methods for solving the travelling salesman problem

Author(s),Year	Computer	No. of cities	Time
Eastman, 1958	??	10	??
Roseman, Tweery, Stone, 1958	??	13	8 man–days
Gonzales, 1962	IBM 1620	5	10 sec.
Gonzales, 1962	IBM 1620	10	8 mins
Held and Karp, 1962	IBM 7090	13	17 sec.
	IBM 7090	20	10 hours
Little, Murty, Karel, Sweeney, 1963	??	13	3.5 hours
	IBM 7090	up to 10	"by hand"
	IBM 7090	20	several seconds
	IBM 7090	40	8 mins
	IBM 7090	25*	4.7 mins
Martin, 1963	IBM 7094	42	5 mins.
Shapiro, 1966	IBM 1620	70	103.5 mins
	IBM 1620	40	8.16 mins

Legend:

***** 25–city problem of HELD & KARP [HK62].

?? Machine not specified.

Table 6.2: Results of Obruča's spanning-tree manipulation method

Number of cities / Specification of the problem	Evaluation of the solution
randomly assigned 5- to 11-cities' problems	56% of solutions optimal all solutions max. 15% greater than optimum
Dantzig, 1959	all solutions between 0.8 and 5.8 greater than the published optimum
Held, Karp, 1962	
Croess, 1958	
Dantzig et al., 1954 (42 cities)	solutions max. 4.4 greater than optimum

cant difference between "no solution" and "no solution to be obtained in polynomial time".

The travelling salesman problem demonstrates mathematics, and graph theory in particular, in its "secular" form. It is not (only) the *ethereal*, but (also) the real world we are trying to tackle here. On this example, we can also observe the change in talking about the time consumed by computation: many reports from the 1950s and 1960s state which particular computer was used and the time (in seconds or minutes) the computation required. However, in his 1968 paper, Obručа uses the complexity measure to evaluate his algorithm, without even explaining the notion. On the other hand, Bellmore and Nemhauser, in their paper from the same year, still use the "computer type — time" description. This brings up a question:

Is mathematics going to change style from the Theorem–Proof *to the* Algorithm–Analysis*, or will there be "two kinds of mathematics"?*

Chapter 7

Conclusion

> *The problem we wish to treat is a combinatorial one involving the determination of an optimal route from one point to another. These problems are usually difficult when we allow a continuum, and when we admit only a discrete set of paths, as we shall do below, they are notoriously so.* [Bel58, p. 87]

Formulations and Context

The problems discussed in this thesis are very often too difficult, or rather, solving them would mean testing too many possibilities, to be handled just by pure common sense. They often appear in practical applications — and as "the variety of real–world problems is enormous" [HHK74, p. 96], there is no possibility to tell in advance that this or that problem would be trivial or extremely difficult. Moreover, the same problem may be encountered independently in several branches, which provides good conditions for multiple discoveries.

Historical Precision

One of the aims of the present thesis was challenging the common beliefs as far as the history of discrete optimization problems is concerned. On the examples of algorithms due to Moore, Borůvka, Jarník, or Dijkstra, it was shown how the mathematical re–formulation of the problems and their solutions blurred the historical picture.

Another feature which, of course, does not help the historical precision of mathematical papers and textbooks, is the immediate assessment of previous solutions as "worse", or just the present one as "better". In this respect, mathematicians seem to be rather lazy — and we proba-

bly should not be surprised by this, as it is more the problem than its history that is interesting to mathematicians. This approach, however, might sometimes lead to "re–inventing the wheel", as was the case with PRIM's or DIJKSTRA's algorithm for the minimum spanning tree.

Language

The language of the mathematical papers discussed in this thesis reflects both the condition of multiple discovery and the necessity to modernize the language we are using to communicate.

The situation of multiple discovery can be disclosed even by such an everyday thing as *notation*. The need to discuss things and the need to name these things led, in each of the contexts, to the emergence of new words — words describing the same things, yet different. Sometimes, this might even extend to procedures. An example of this is the *Bellman–Ford algorithm* for shortest paths, which was formulated in at least two different contexts.

Not only the notions, but also the procedures and the way of talking about them needs to be standardized. The procedures from the earlier periods, e.g. showing the way of solving on specific examples, can be recognized as algorithms. However, only during the post-war years the concept of algorithm developed and the word itself became much more widely used. This tendency can well be seen on the development of discrete optimization problems after World War II, when mathematicians slowly started abandoning the description on examples and began using standardized descriptions — nowadays mostly in the form of a pseudo–programming language, the *pseudo–code*.

Chapter 8

Appendices

8.1 Biographical Data

The biographical data given in this section are rather brief and serve only for the most rough orientation. The exceptions are only the biographies of O. Borůvka, V. Jarník, and A. Kotzig, Czech and Slovak mathematicians who contributed most to discrete optimization problems discussed in this thesis.

Kenneth Joseph Arrow, * 1921, American economist; Nobel prize in economy (1972).

Richard E. Bellman, 1920–1984, American mathematician.

Roger Joseph Boscovich (Bošković, Rudjer Josip), 1711–1787, Croatian scientist (mathematical physics).

Otakar Borůvka, 1899 – 1995, Czech mathematician.

Otakar Borůvka was born in Uherský Ostroh, a small Moravian town. His father was a teacher. Borůvka himself was an excellent student. When he was at secondary school, he was the best student in his class. His interests were multiple. Apart from mathematics, he was also interested in the classical languages, Latin and Greek. Thus, the choice of a university subject was not an easy one for him.

In 1918, he chose to study at the university which was closest to his hometown. It was the University of Technology in Brno. However, after he had spent one year at this university studying to become a civil engineer, he went to study mathematics, which was since

then his major field of interest. His future career was influenced also by Matyáš Lerch, Borůvka's professor of mathematics at the University of Technology in Brno. In 1919, Masaryk University was founded in Brno, and professor Lerch then asked Borůvka whether he would like to study "pure" mathematics.

Borůvka accepted. He started studying mathematics, but at the same time, he continued in his studies at the University of Technology. After graduation, he continued with his doctoral studies. As it is not the aim here to describe his life in detail (the interested reader is referred to the monograph [TŠP96]), only the parts of mathematics he was interested in are mentioned here. These were above all differential equations, differential geometry, and algebra. His graph–theoretical work consists of just two articles on the same topic, the minimum spanning tree ([Bor26a, Bor26b]).

Èlie Joseph Cartan, 1869–1951, French mathematician.

Augustin Louis Cauchy, 1789–1857, French mathematician.

Arthur Cayley, 1821–1895, English lawyer and mathematician.

Gustav Adolf Astor Choquet, *1915, French mathematician.

Julian Lowell Coolidge, 1873–1954, American mathematician.

Antoine Augustin Cournot, 1801–1877, French mathematician, economist and philosopher. Founder (with L. M. E. Walras) of the mathematical school of political economy.

Karel Čulík, * 1926, Czech mathematician

George Bernhard Dantzig, *1914, American mathematician and scientist. Author of the book *Linear programming and extensions* (1963)

Gabriel Andrew Dirac, 1925–1984, adoptive son of Paul Dirac (1902–1984, Nobel prize for Physics, 1933), Professor of Mathematics at the University of Aarhus, Denmark.

Pál (Paul) Erdős, 1913–1996, an outstanding Hungarian mathematician, known not only for his countless mathematical papers (many of them were written with other mathematicians from all over the world), but also for his extraordinary life-style.

LEONHARD EULER, 1707–1783, Swiss mathematician, physicist and astronomer. He worked in St Petersburg and Berlin.

JEAN BAPTIST JOSEPH FOURIER, 1768–1830, French mathematician.

DELBERT RAY FULKERSON,1924–1976 , American mathematician.

SIR WILLIAM ROWAN HAMILTON, 1805–1865, Irish mathematician and physicist.

FRANK HARARY, * 1921, American mathematician

CARL FRIDOLIN BERNHARD HIERHOLZER, 1840–1871, German mathematician.

VOJTĚCH JARNÍK, 1897–1971, Czech mathematician.

Vojtěch Jarník was born in a family of Jan Urban Jarník, Professor of Romanic philology at Charles University. He studied mathematics and physics at the Faculty of Arts of Charles Univeristy in the years 1915–1919. He was awarded his doctor's degree in 1921 for his thesis "On the roots of Bessel functions". He was influenced by the Czech mathematician Karel Petr.

In the years 1919–1921, Jarník taught at the University of Technology in Brno. Since 1921, he worked at Charles University in Prague. Between World War I and II, Jarník visited Göttingen, which was then the centre of European mathematical life, with personalities such as David Hilbert, Emmy Noether, Edmund Landau, or Otto Neugebauer.

Professor Jarník actively participated in Czech mathematical life. He was also an excellent teacher. As far as his scientific career is concerned, his work stretches over the field of graph theory, analytical number theory, geometry of numbers, and theory of real functions to diophantine approximations.

LEONID VITALJEVICH KANTOROVICH, 1912–1986, Soviet mathematician and economist, Nobel prize in economy (1975) for his contribution to the theory of the optimal allocation of sources.

GUSTAV ROBERT KIRCHHOFF, 1824–1887, German physicist (theory of circuits, using topology and elasticity).

THOMAS PENYNGTON KIRKMAN, 1806–1895, rector at a parish of the Church of England. He published many mathematical papers.

Dénes Kőnig, 1884–1944, Hungarian mathematician, son of the Hungarian mathematician Gyula Kőnig.

Anton Kotzig, 1919–1991, Slovak mathematician.

Anton Kotzig was born in Kočovce in Slovakia. He started studying at Charles University in Prague, but after the Czech universities had been closed, he continued in Bratislava. He received his doctorate for his work in mathematical statistics. He worked for an insurance company (1940–1948) and in agricultural research (1948–1951). In 1951, he became professor at the University of Economics.

Kotzig is considered to be the founder of the Slovak graph–theoretical school. His work in graph theory is extensive and includes results on convex polyedra, Hamiltonian graphs, Latin squares, and factorisation of graphs.

Until 1969, Kotzig was an active member of Czech and Slovak mathematical life. However, in 1970 he decided to stay in Canada and never returned to his native country. In Canada, he also had many followers.

Harold W. Kuhn, * 1925, American mathematician.

Pierre Simon de Laplace, 1749–1827, French mathematician, physicist, astronomer, and politician.

Johann Benedict Listing, 1808–1882, German mathematician.

Oskar Morgenstern, 1902–1977, American economist of German descent. Founder (with J. von Neumann) of game theory.

Claude-Louis-Marie Navier, 1785–1836, French technician.

John von Neumannn, 1903–1957, American mathematician of Hungarian descent. Founder (with O. Morgenstern) of game theory.

Øystein Ore, 1899–1968, American mathematician of Norwegian descent.

Mikhail Ostrogradskij, 1801–1862, Russian mathematician and mechanic.

Vilfredo Pareto, 1848–1923, Italian sociologist and economist, founder of the Laussane school of economy, one of the founders of the mathematical school in economy.

ERNST PAUL HEINZ PRÜFER, 1896–1934, German mathematician.

JULIA BOWMAN ROBINSON, 1919–1985, American mathematician.

CLAUDE ELWOOD SHANNON, 1916–2001, American mathematician.

ALAN W. TUCKER, 1905–1995, American mathematician.

MARIE ESPRIT LEÓN WALRAS, 1834–1919, French economist, founder of the mathematical school of national economy.

HASSLER WHITNEY, 1902–1989, American mathematician.

8.2 Original Quotations

Bílý, Nožička, Fiedler

on the history of network flow problems [BFN58, pp. 119–120]:

> Die Aufgabe, ein in mehreren Produktionsstellen erzeugtes Produkt unter bestimmte Verbrauchsstellen mit gegebenem Verbrauchsumfang (gleich dem Produktionsumfang) so zu verteilen, dass die Transportkosten minimal werden, wurde zuerst von Hitchcock [Hit41] mathematisch formuliert und mit mathematischen Mitteln gelöst. [...] Der Schiffmangel, der schon während des ersten Weltkrieges zu gewissen Regulierungen des Umlaufes von Schiffen zwang und während des zweiten Weltkrieges in viel grösserem Ausmasse in Erscheinung trat, führte im zweiten Weltkriege zur mathematischen Formulierung und Lösung der Aufgabe, wobei zu bemerken ist, dass der Transport zur See gewisse besondere Bedingungen stellt, die von denen des Eisenbahntransports unterschiedlich sind. [...] Die Simplexmethode der Lösung des Transportproblems wurde von Dantzig [4] angegeben.[1]

Bílý, Nožička, Fiedler

on Czech mathematicians dealing with network flows [BFN58, p. 119]:

> In der Tschechoslowakischen Republik ist man zum Transportproblem im obigen Sinne im Zusammenhang mit dem Bestreben nach einer ökonomischen Gestaltung des Einbahntransportes gekommen. Das Problem wurde im Jahre 1952 von Nožička unabhängig von den oben angeführten Arbeiten gelöst; die Methode wurde ausführlich in [12] erläutert.
>
> Ein wachsendes Interesse der Mathematiker und Praktiker in der Tschechoslowakei und im Ausland — siehe z. B. [1] und [14] — an dem Transportproblem und an anderen Problemen der Linearplanung bewegte die Autoren dieses Artikels zu einer verkürzten Umarbeitung der Arbeit [12], wobei die Theorie der Graphen als Grundlage gestellt wurde. Die Möglichkeit, das Transportproblem unter Mithilfe dieser Theorie zu lösen,

[1] [4] refers to the article Dantzig, G. B.: Applications of the Simplex Method to a Trasportaion Problem, *Activity Analysis of Production adn Allocation*, 359–373.(The year of publication not stated.)

> wurde in [4] und [13] erwähnt und in [6] von FLOOD ausgenützt; siehe darüber auch BELLMAN in [2]. Wir finden diese Methode sehr anschaulich und besonders dann anwendbar, wenn keine modernen Rechenautomaten zur Verfügung sind; über deren Ausnützung siehe Eisenmann [5]. Die Existenzfragen siehe in [3].

Bílý, Nožička, Fiedler

on the formulation of the problem [BFN58, pp. 119–120]:

> Das Transportproblem wird unter folgenden V o r a u s s e t z u n g e n gelöst:
>
> 1. Das Produkt ist in beliebig kleine Mengen teilbar.
> 2. Es besteht keine Einschränkung über die kleinste zu transportierende Menge; diese Voraussetzung trifft gewiss nicht zu, z. B. für Kohle ist die kleinste Menge ein Eisenbahnwagen.
> 3. Die zur Beförderung benützten Transportmittel bleiben auf der Rückreise unbenutzt.
> 4. Der Transport wickelt sich mit einer gleichbleibenden Intensität ab.
> 5. Die Transportwege haben eine unbeschrankte Kapazität.

Pál Erdős

on DÉNES KŐNIG's monograph (Cover of the 1986 edition of Kőnig's monograph [Kőn86]):

> Das Original dieses hier als Nachdruck vorliegenden Werkes von DÉNES KŐNIG erschien im Jahre 1936 ... Die stürmische und fruchtbringende Entwicklung der Graphentheorie, die bald nach Erschienen des Buches einsetzte – durch den 2. Weltkrieg unterbrochen und verzögert –, ist in nicht geringem Maße eben diesem Werk und seinem Autor zu verdanken, der leider selbst ein Opfer des Faschismus wurde.

D. Kőnig

on Kirchhoff's laws and graph theory:

> Die vorangehenden Untersuchungen verdanken teilweise ihren Ursprung einer Fragestellung der Elektrizitätslehre, welche 1845 von K i r c h h o f f gestellt und gelöst wurde. In einem endlichen zusammenhängenden gerichteten Graphen G sollen die Kanten $k_1, k_2, \ldots,$
> k_{α_1} als Drähte aufgefaßt werden, in denen ein elektrischer Strom zirkuliert. Für jede (gerichtete) Kante k_i sei ihr elektrischer Widerstand $\omega_i (> 0)$ und die elektromotorische Kraft E_i die in k_i ihren Sitz hat (in der Richtung von k_i gemessen), gegeben. [...]

Dénes Kőnig, *Theorie der endlichen und unendlichen Graphen*, Teubner—Leipzig 1936; pp. 139–141.

J. Łukaszewicz et al.

on their work related to anthropology [FŁP+51, pp. 282–283]:

> Ce travail est un sommaire géometrique de certaines applications de mathématiques aux sciences naturelles. [...]
>
> Ces méthodes de liason et de division ont été proposées aux anthropologues pour ranger les crânes des fouilles. Elles s'appliquent aussi, avec effet, à des problèmes de biologie, d'agriculture, de technologie, même de linguistique. Voir *Taksonomia Wroclawska*, travail du Groupe Géneral des Applications de l'Institut Mathématique de l'Etat (Przeglad Antropologiczny, à paraître). Cf. aussi la communication collective *Une méthode taxonomique et ses applications aux sciences naturelles*, ee fascicule, p. 319. [...]
>
> Unissons, par un segment, chacun d'eux au point le plus proche; les segments ainsi obtenus seront appelés *liens du premier ordre*.

Note: The thesis, as presented before the committe in 2001, also contained the section *Annotated Bibliographies*, consisting of annotated lists of articles — primary sources — on the shortest–path problems, network flow problems, the minimum spanning tree problem, and the

travelling–salesman problem. Reviews from referative journals *Zentralblatt für Mathematik und ihre Grenzgebiete* and *Mathematical Reviews* were used when available. If the article was not reviewed in either of the two journals, the abstract or some other information about the article was given. For the reasons of the space, this bibliography was left out for this edition.

Bibliography

[BCL70] Richard Bellman, Kenneth L. Cooke, and Jo Anne Lockett. *Algorithms, Graphs, and Computers*. Academic Press, New York and London, 1970.

[Bel58] Richard E. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, 16(1):87–90, 1958.

[Ber58] Claude Berge. *Théorie des graphes at ses applications*. Dunod, Paris, 1958. Russian translation by A.A. Zykov, Moskva 1962.

[BFN58] Josef Bílý, Miroslav Fiedler, and František Nožička. Die Graphentheorie in Anwendung auf das Transportproblem. *Czechoslovak Mathematical Journal*, 8 (83):94–121, 1958.

[BLW76] Norman L. Biggs, Keith E. Lloyd, and Robin J. Wilson. *Graph Theory 1736-1936*. Clarendon Press, Oxford, 1976.

[BN68] M. Bellmore and G. L. Nemhauser. The travelling salesman problem: a survey. *Operations Research*, 16:538–558, 1968.

[Bor26a] Otakar Borůvka. O jistém problému minimálním [On a certain minimal problem]. *Práce Moravské přírodovědecké společnosti*, III(3):37–58, 1926.

[Bor26b] Otakar Borůvka. Příspěvek k řešení otázky ekonomické stavby elektrovodných sítí [contribution to the solution of the problem of economical constructions of electrical networks]. *Elektrotechnický obzor*, 15(10):153–154, 1926.

[Bre94] Sonja Brentjes. *Linear optimization*, pages 828–836. In Grattan-Guinness [GG94], 1994.

[ČDF67] Karel Čulík, Václav Doležal, and Miroslav Fiedler. *Kombinatorická analýza v praxi.* SNTL – Nakladatelství technické literatury, Praha, 1967.

[Cho38] Gustave Choquet. Étude de certains réseaux de routes. *Comptes Rendus Academie des Sciences, Paris*, 206:310–313, 1938.

[CL65] Yoeng-Jin Chu and Tseng-Hong Liu. On the shortest arborescence of a directed graph. *Scientia Sinica*, XIV(10):1396–1400, 1965.

[CLR91] Thomas H. Carmen, Charles E. Leiserson, and Ronald R. Rivest. *Introduction to Algorithms.* MIT Press, 5th edition, 1991.

[Cro58] G. A. Croes. A method for solving traveling salesman problems. *Operations Research*, 6:791–812, 1958.

[Čul60a] Karel Čulík. K jednomu minimálnímu problému O. Borůvky. *Časopis pro pěstování matematiky a fysiky*, 85:93–94, 1960.

[Čul60b] Karel Čulík. Úlohy a problémy; č. 1. *Časopis pro pěstování matematiky a fysiky*, 85:92, 1960.

[Dan57] George B. Dantzig. Discrete–variable extremum problems. *Operations Research*, 5:266–270, 1957.

[Dan63] George B. Dantzig. *Linear Programming and Extensions.* Princeton University Press, Princeton, 1963. In Slovak: 1966, SNTL Bratislava. Translated by Dr. Rudolf Briška.

[DFJ54] George B. Dantzig, Delbert R. Fulkerson, and Selmer M. Johnson. Solution of a large–scale travelling salesman problem. *Operations Research*, 2:393–410, 1954.

[Dij59a] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[Dij59b] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[Dre69] Stuart E. Dreyfus. An appraisal of some shortest-path algorithms. *J. Ops. Res. S. America*, 3, Vol. 17:395–412, 1969.

[Dur98] Helena Durnová. Minimum Spanning Tree: A Neglected Paper, 1998. 9. Novembertagung zur Geschichte der Mathematik, Nijmegen.

[Dur99] Helena Durnová. Otakar Borůvka and the Minimum Spanning Tree, 1999. Presented at the 5. Tagung der Fachsection für Geschichte der Mathematik der DMV, 2–6 June 1999.

[Edm65] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[Elm64] Salah E. Elmaghraby. Sensitivity analysis of multiterminal flow networks. *Operations Research*, 12:680–688, 1964.

[Eul36] Leonhard Euler. Solutio problematis ad geometriam situs pertinensis. *Comm. Acad. Sci. Imp. Petropol*, 8:128–140, 1736. (Quoted in *Graph Theory 1736-1936* by Biggs, Lloyd and Wilson).

[FF62] Lester R. Jr. Ford and Delbert R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, New Jersey, 1st edition, 1962.

[Flo56] Merrill M. Flood. The traveling-salesman problem. *Operations Research*, 4(1):61–75, 1956.

[FŁP+51] K. Florek, J. Łukaszewicz, J. Perkal, H. Steinhaus, and S. Zubrzycki. Sur la liason et la division des points d'un ensemble fini. *Colloquium mathematicum*, 2:282–285, 1949–1951.

[Ful61] Delbert R. Fulkerson. A network-flow computation for project cost curves. *Management Science*, 7:167–178, 1960/1961.

[GG64] P. C. Gilmore and R. E. Gomory. Sequencing a one state variable machine: a solvable case of the travelling saleman problem. *Operations Research*, 12:655–679, 1964.

[GG94] Ivor Grattan-Guinness, editor. *Companion Encyclopedia of the History and Philosohpy of the Mathematical Science*. Routledge, 1994.

[GH85] R. L. Graham and P. Hell. On the history of the minimum spanning tree problem. *Annals of the History of Computing*, 7:43–57, 1985.

[GJ79] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, 1979. (Russian translation: Moscow 1982).

[Har69] Frank Harary. *Graph Theory.* Addison-Wesley Publishing Company, Reading, Massachesetts, 1969. Russian translation by V. N. Kozyrev and G. P. Gavrilov, Mir, Moscow 1973.

[HHK74] Keld Helbig Hansen and Jakob Krarup. Improvements of the Held-Karp algorithm for the symmetric traveling–salesman problem. *Mathematical Programming*, 7:87–96, 1974.

[Hit41] Frank L. Hitchcock. The distribution of a product from several sources to to numerous localities. *Journal of Mathematics and Physics*, 20:224–230, 1941.

[HK62] Michael Held and Richard M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 10(1, March):196–210, 1962.

[Hou97] David Hounshell. The Cold War, RAND, and the generation of knowledge, 1946–1962. *Historical Studies in the Physical and Biological Sciences*, 27(2):237–267, 1997.

[Jar30] Vojtěch Jarník. O jistém problému minimálním [On a certain minimal problem]. *Práce Moravské přírodovědecké společnosti*, VI(4):57–61, 1930. Jahrbuch: 0.

[Joh77] D. B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J of the Association for Computing Machinery*, 24:1–13, 1977.

[Kje98] Tinne Hoff Kjeldsen. A History of the Minimax Theorem: a journey through different mathematical contexts, 1998. 9. Novembertagung zur Geschichte der Mathematik, Nijmegen.

[Kje99] Tinne Hoff Kjeldsen. *En kontekstualisieret matematikhistorisk analyse af ikke-lineær programmering: udviklingshistorie og multipel opdagelse.* PhD thesis, Roskilde Universitet, 1999.

[Kli72] Morris Kline. *Mathematical Thought from Ancient to Modern Times*. Oxford University Press, New York, Oxford, 1972.

[KN99] Bernhard Korte and Jaroslav Nešetřil. Vojtěch jarník's work in combinatorial optimization. In *Life and Work of Vojtěch Jarník, 1897–1970*, pages 37–53. Prometheus, Praha, 1999. In English. Czech translation by EVA MILKOVÁ in *Pokroky matematiky, fyziky a astronomie* **44** (1999), No. 3, pp. 187–200.

[Kőn86] Dénes Kőnig. *Theorie der endlichen und unendlichen Graphen (Kombinatorische Topologie der Streckenkomplexe. Mit einer Abhandlung von L. Euler)*. Number 6 in Teubner-Archiv zur Mathematik. BSB B. G. Teubner Verlagsgesellschaft, Leipzig, 1986. Photographic reproduction of a book originally published in 1936 by the Akademische Verlagsgesellschaft M. B. H., Leipzig. Edited and with comments and an introduction by H. Sachs, an introduction by Paul Erdős, a biography of Kőnig by T. Gallai.

[Kot61a] Anton Kotzig. O kostrách vyššieho než prvého rádu [über Gerüste höherer Ordnung]. *Časopis pro pěstování matematiky a fysiky*, 86:288–307, 1961. In Russian. Slovak and German summaries.

[Kot61b] Anton Kotzig. Súvislé podgrafy s minimálnou hodnotou v konečnom súvislom grafe [Connected subgraphs with the minimum value in a finite connected graph]. *Časopis pro pěstování matematiky a fysiky*, 86:1–6, 1961.

[Kru56] Joseph B. Kruskal, Jr. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proceedings of the American Mathematical Society*, 7:48–50, 1956.

[Kru97] Joseph B. Kruskal. A reminiscence about shortest spanning subtrees. *Archivum Mathematicum*, Tomus 33:13–14, 1997.

[KT64] L. L. Karg and G. L. Thompson. A heuristic approach to solving traveling salesman problems. *Management Science*, 10:225–248, 1964.

[Lin65] Shen Lin. Computer solution of the traveling salesman problem. *Bell System Technical Journal*, 44(10):2245–2269, 1965.

[LMSK63] J. D. C. Little, K. G. Murty, D. W. Sweeney, and C. Karel. An algorithm for the travelling salesman problem. *Operations Research*, 11:972–989, 1963.

[LW57] H. Loberman and A. Weinberger. Formal procedures for connecting terminals with a minimum total wire length. *Journal of the Association for Computing Machinery*, 4:428–437, 1957.

[Min57] George J. Minty. A comment on the shortest route problem. *Operations Research*, 5:724, 1957.

[ML55] G. Morton and A. H. Land. A contribution to the "travelling-salesman" problem. *Journal of the Royal Statistical Society, Series B*, 17(2):185–194, 1955.

[Moo59] Edward F. Moore The shortest path through a maze. In *Proceedings of an International Conference on the theory of Switching 1957, Part II.*, pages 285–292. Harvard Univ. Press, Cambridge, Mass., 1959.

[Nov99] Břetislav Novák, editor. *Life and Work of Vojtěch Jarník, 1897–1970.* Prometheus, Praha, 1999. In cooperation with the Union of Czech Mathematicians and Physicists.

[Obr68] Alex K. Obruča. Spanning tree manipulation and the travelling salesman problem. *Computer Journal*, 10:374–377, 1968.

[Ore62] Oystein Ore. *Theory of graphs*, volume XXXVIII of *American Mathematical Society Colloquium Publications.* American Mathematical Society, Providence, R.I., 1962. Russian translation published in 1968 by Nauka, Moscow.

[Ple83] Ján Plesník. *Grafové algoritmy.* Veda - Vydavatelstvo Slovenskej akademie ved, Bratislava, 1 edition, 1983.

[Poh70] Ira Pohl. Heuristic search viewed as path-finding in a graph. *Artificial Intelligence*, 1:193–204, 1970.

[Pri57] Robert C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1339–1401, 1957.

[PS85] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity.* Mir, Moscow, First edition, 1985.

[PW60] Maurice Pollack and Walter Wiebenson. Solutions of the shortest route problem – a review. *Operations Research*, 8:224–230, 1960.

[RA59] H. Rapaport and P. Abramson. An analog computer for finding an optimum route through a communications network. *IRE Transactions on Communications Systems*, CS-7:37–42, 1959.

[Rob49] Julia Robinson. On the Hamiltonian game, a traveling salesman problem. Technical Report RM-303, RAND Corporation, 1949.

[Shi55] Alfonso Shimbel. Structure in communication nets. In *Proceedings of the Symposium on Information Networks*, pages 199–203, April 12–14, 1954 1955.

[Šiš97] Pavel Šišma. *Teorie grafů 1736–1963*. Prometheus, Praha, 1997.

[SME] *Soviet Mathematical Encyklopedia*. (English translation with editorial comments.).

[Tar95] G. Tarry. Le problème des labyrinthes. *Nouvelles Annales Math.*, 14:187, 1895.

[TŠP96] Z. Třešňák, P. Šarmanová, and B. Půža. *Otakar Borůvka*. Universitas Masarykiana, Brno, first edition, 1996.

[Vág85] István Vágó. *Graph Theory: Application to the Calculation of Electrical Networks*. Akadémiai Kiadó, Budapest, 1985.

[Wil56] R. I. Wilkinson. Abridged bibliography of articles on toll alternate routing. *Bell System Technical Journal*, 35:421–507, 1956.

[Yao75] Andrew Chi-chih Yao. An $O(|E| \log \log |V|$ algorithm for finding minimum spanning trees. *Information Processing Letters*, 4(1):21–23, 1975. Zbl 307.68028 (no review).

Name Index