NETWORK FLOWS IN OPTIMISATION PROBLEMS AND THEIR EXTENSIONS

Miloš Šeda
Institute of Automation and Computer Science
Faculty of Mechanical Engineering, Brno University of Technology
Technická 2, Brno 616 69, Czech Republic
E-mail: seda@fme.vutbr.cz
EPI Kunovice, Osvobození 699, Kunovice 686 04, Czech Republic

KEYWORDS
Maximum flow, Steiner tree, flows with priorities, multicommodity flow, NP-hard problem, heuristic method

ABSTRACT

Network flow problems are among the most important ones in graph theory. Since there are many well-known polynomial algorithms for solving the classical Maximum Flow Problem, we, instead of summarising them, focus on special formulations and their transformation into the basic one, and because other graph theory problems may be formulated with the help of network flow tools, we show how to formulate the Minimum Steiner Tree Problem using the maximum network flow terminology and derive its mathematical model.
Finally, we discuss the Integer Maximal Multi-commodity Flow Problem. Since this network flow version belongs to the class of NP-hard combinatorial problems, for large scale instances, it must be solved by approximation or heuristic techniques. We present a stochastic heuristic approach based on a simulated annealing algorithm.

INTRODUCTION

The flow problems have many specific formulations depending on the constraints imposed resulting in various applications in transportation, distribution, telecommunications, etc. Examples of flow networks can be found, e.g., in electrical circuits, pipe networks, and city plans. In the literature, many other important applications are described, such as machine scheduling, assignment of computer modules to computer processors, tanker scheduling, project management, facility layout and location, production and inventory planning, selection problem solving, open-pit mining problem, forest clearing problem, producing memory chips (VLSI layout), maximum closure problem, vertex cover on bipartite graphs, and mesh generation.

For some of the flow problems such as maximum flow problem, minimum cost flow problem, polynomial time algorithms are known (Ahuja et al 1993), others such as integer maximum multicommodity flow problem (Ouorou et al, 2000), (Hochbaum 1996), (Plesník 1983) are NP-hard combinatorial problems and, for large-scale instances, must be solved by approximation or using heuristic techniques.

Now, we will summarise some network-flow-related graph-theoretic concepts and, without giving proofs, mention the key theorems that result in the classical algorithms to compute a maximal flow.

Definition 1.
A (*single source – single sink*) *network* is a quadruple $N=(G,s,t,c)$ where $G=(V,E)$ is a simple directed graph without loops, $s \in V$ is the *source* of network N, $t \in V$, $t \neq s$ is the *sink* of network N and c: $E(G) \rightarrow \mathbf{R}^+$ is a nonnegative edge-weight function called the *capacity* of network N. For each edge $e \in E$, we call the corresponding value $c(e)$ the capacity of edge e. □

Definition 2.
A *flow* in a network $N=(G,s,t,c)$ is an edge weight function f: $E(G) \rightarrow \mathbf{R}$ satisfying the following conditions (the second one is also known as Kirchhoff's law):

(a) $(u,v) \in E(G)$: $0 \leq f(u,v) \leq c(u,v)$ [*capacity constraints*]

(b) $\forall v \in V - \{s,t\}$: $\sum_{(u,v)\in E(G)} f(u,v) - \sum_{(v,w)\in E(G)} f(v,w) = 0$

[*conservation-of-flow constraints*]

□

Definition 3

- The *value* $|f|$ *of flow* f is the total network flow leaving the source (= network flow entering the sink).

$$|f| = \sum_{(s,v)\in E(G)} f(s,v) = \sum_{(v,t)\in E(G)} f(v,t)$$

- If $f(u,v) = c(u,v)$, then we say that (u,v) is a *saturation arc*.
- A flow f is called *saturated* if, for each directed path $s = v_0, \ldots, v_k = t$ from s to t, there is i such that $f(v_{i-1}, v_i) = c(v_{i-1}, v_i)$.
- A *maximum flow* f^* in a capacitated network N is a flow in N having the maximum value, i.e. $|f| \leq |f^*|$ for every flow f in G. □

Proceedings 24th European Conference on Modelling and Simulation ©ECMS Andrzej Bargiela, Sayed Azam Ali David Crowley, Eugène J.H. Kerckhoffs (Editors)
ISBN: 978-0-9564944-0-5 / ISBN: 978-0-9564944-1-2 (CD)

Maximum flow problem is formulated as the following linear programming problem:

$$\text{Maximise } |f| = \sum_{(s,v)\in E(G)} f(s,v) \quad (1)$$

subject to

$$(u,v)\in E(G): 0 \le f(u,v) \le c(u,v) \quad (2)$$

$$\forall v \in V - \{s,t\}: \sum_{(u,v)\in E(G)} f(u,v) - \sum_{(v,w)\in E(G)} f(v,w) = 0 \quad (3)$$

SPECIAL FORMULATIONS AND THEIR TRANSFORMATION

In real situations, many other constraints may be added: the capacities of vertices may be constrained in addition to edge constraints (the number of cars, for example, passing a crossroads during a certain time interval).

Networks with constrained vertex capacities

If the maximum flow that can pass through a vertex is constrained, such as by $\sum_{u\in V} f(u,v) \le w_v$, then

1. we split vertex v into 2 new vertices v' and v" connected by an edge having the capacity wv,
2. all incoming edges of v will enter into v', and all outgoing edges of v will be moved to start in v".

Networks with more sources and sinks

Let us assume that we need to determine the maximal flow in a network with m sources and n sinks and each source can supply each sink.

This problem can be transformed into the basic problem with a single source and sink by adding a supersource and a supersink. The supersource will be connected by an outgoing edge with each original source and the supersink by incoming edges with each original sink. The weights of these new edges are initialised with unrestricted capacities.

Maximal flow with priorities of sources and sinks

In practice, sources and sinks are frequently ordered by their priorities (Plesník 1983), i.e. first the demands are met from a source with the highest priority, then from the source with the second highest priority, etc. Similarly, priorities can be defined for sinks, i.e. first the demands of a sink with the highest priority are met, etc.

Definition 4
Let f be a flow from sources $s_1, s_2, \dots, s_m$ to sinks $t_1, t_2, \dots, t_n$ and $|f^+(s)| = \Sigma_{(s,v)\in E(G)} f(s,v)$ denote the outflow from source s and $|f^-(t)| = \Sigma_{(v,t)\in E(G)} f(v,s)$ inflow into sink t.
Then the flow f is called *lexicographically maximal with respect to sources* if the vector $(|f^+(s_1)|,|f^+(s_2)|,\dots,|f^+(s_m)|)$ is lexicographically maximal. Similarly, the flow f is called *lexicographically maximal with respect to sinks* if the vector $(|f^-(t_1)|,|f^-(t_2)|,\dots,|f^-(t_n)|)$ is lexicographically maximal.
□

Note that the *lexicographical order* $\ge$ is defined as follows: Let $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n)$, then $\mathbf{a} \ge \mathbf{b} \Leftrightarrow_{\text{df}}$ $\mathbf{a} = \mathbf{b}$ or $\exists\, k\in\{1, \dots, n\}: a_k > b_k$ and $a_i = b_i\ \forall i\ (1 \le i < k)$.

The lexicographically maximal flow with respect to sources can be found as shown in Figure 2. All sinks are connected to a supersink t with edges having infinite capacities, a supersource s is connected to source s_1 by an edge of infinite capacity and a maximal flow f_1' from s to t is computed. Then the edge $s - s_2$ is added and, starting from f_1', the flow f_{12}' is determined and, from f_{12}' after addition of the edge $s - s_3$, the flow f_{123}' is determined, etc. The lexicographically maximal flow with respect to sinks can be obtained by a symmetrical approach.

Theorem 3. *The lexicographically maximal flow with respect to sources* (*or sinks*) *is the maximal flow.*
Proof. This is straightforward since, otherwise, an augmenting path from s_i to t_j should exist and, by increasing the flow along the augmenting path, we would find a flow that was lexicographically bigger. ■

A lexicographically maximal flow with respect to sources $f_{12\dots m}$' and a lexicographically maximal flow with respect to sinks $f_{12\dots n}$'' can be combined. Since both these flows are maximal, there is a cut $(A, A\sim)$ separating the source and the sink where edges $(u,v)\in(A\times A\sim)\cap E$ are saturated by both of them and, along each edge $(v,u)\in(A\sim\times A)\cap E$, the flow is zero. The resulting flow can be created by using the flow $f_{12\dots m}'(u,v)$ for edges $(u,v)\in(A\times A)\cap E$ and the flow $f_{12\dots n}''(u,v)$ for the other edges .

NETWORK STEINER TREE PROBLEM AND ITS NETWOK FLOW FORMULATION

The *Network Steiner tree problem* (NSTP) (or *Steiner tree problem in graphs*) (Du et al. 2000), (Hwang et al. 1992) is concerned with connecting a subset of vertices at a minimal cost. More precisely, given an undirected connected graph $G=(V,E)$ with vertex set V, edge set E, nonnegative weights associated with the edges, and a subset B of V (called *terminals* or *customer vertices*), the problem is to find a subgraph T that connects the vertices in B so that the sum of the weights of the edges in T is minimised. It is obvious that the solution is always a tree and it is called a *Steiner minimum tree* for B in G.

Applications of the NSTP are frequently found in the layout of connection structures in networks and circuit. Their common feature is that of connecting together a set of terminals (communications sites or circuits components) by a network of the minimal total length.

If $|B|=2$ then the problem reduces to the *shortest path problem* and can be solved by Dijkstra's algorithm. In

the case of $B=V$ the NSTP reduces to the *minimum spanning tree* (MST) *problem* and can be solved by Jarník's (Prim's), Borůvka's or Kruskal's algorithm. All these algorithms are polynomial. However, in the general case the NSTP is NP-complete (Hwang et al. 1992), (Plesník 1983) and therefore it cannot be solved exactly for larger instances, i.e. heuristic or approximation methods must be used. Normally a Steiner minimum tree is not a minimum spanning tree only, it can also span some nonterminals, called *Steiner vertices*.

Let $V=\{1,2,\ldots,n\}$ and S be a set of Steiner vertices. For every edge (i,j), c_{ij}, $c_{ij} \geq 0$ is a weight of the edge. The aim is to find a connected graph $G'=(B \cup S, E')$ (Steiner tree), $E' \subset E$, for the sum of weights to be minimal.

In other words, the Steiner minimum tree problem can be described as a problem of finding a set of edges that connects terminals.Therefore we can define a bivalent variable x_{ij} for each edge $(i,j) \in E$ indicating whether the edge (i,j) is included into the Steiner tree ($x_{ij} = 1$) or not ($x_{ij} = 0$) and similarly a bivalent variable f_i indicating whether vertex i is included in the Steiner tree ($f_i = 1$) or not ($f_i = 0$). For terminals, $i \in B$, it is satisfied $f_i = 1$, and $f_i = 0$ for the other vertices, $i \in (V-B)$.

Using these denotations, we can derive the model based on a network flow formulation of the NSTP as follows. The variable y_{ij} represents a flow through the edge $(i,j) \in E'$.

$$\text{Minimise} \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \tag{4}$$

subject to

$$r := \min \{k \mid k \in B\} \tag{5}$$

$$\forall j \in (V - \{r\}): x_{rj} = 0 \tag{6}$$

$$\forall i \in (V - \{r\}): \sum_{j=1}^{n} x_{ij} = f_i \tag{7}$$

$$\forall i \in (V - \{r\}): \sum_{j=1}^{n} y_{ij} - \sum_{j=1, j\neq r}^{n} y_{ij} = f_i \tag{8}$$

$$\forall i,j \in V: \; y_{ij} \leq (n-1) x_{ij} \tag{9}$$

$$\forall i,j \in V: \; y_{ij} \in \mathbf{Z}_+ \tag{10}$$

$$\forall i,j \in V, c_{ij} = 0: x_{ij} := 0 \tag{11}$$

$$\forall i,j \in V: \; x_{ij} \in \{0,1\} \tag{12}$$

$$\forall i \in B: \; f_i = 1 \tag{13}$$

$$\forall i \in (V-B): f_i \in \{0,1\} \tag{14}$$

where $\mathbf{Z}_+$ denotes the set of nonnegative integers.

MULTICOMMODITY FLOWS

Multicommodity flow problems are a special class of operations research problems with many applications such as transportation, distribution and telecommunications. If the commodities do not share common facilities, we would solve each single-commodity problem separately using traditional polynomial algorithms. Unfortunately, some resources (e.g. capital, labour, equipment, space) can be shared by several commodities.

This problem has a variety of formulations depending on the constraints defined. The paper (Ouorou *et al*., 2000) provides a survey of algorithms for convex multicommodity flow problems. A special case of the problem when the capacity constraints are modelled by random variables because some of the network nodes may fail, is studied in (Lin 2000). The paper (Ouorou and Mahey 2000) presents a nonlinear version of the problem where a nonlinear cost function associated with each edge is considered. Special cases of multicommodity flow problems are studied in (Lozovanu and Fonoberova 2006), (Wang and Kleinberg 2009) and (Srivastav and Stangier 2000).

Consider a network represented by a directed graph $G=(V,E)$ with $n=|V|$ vertices and $m=|E|$ edges. Let K be a set of commodities, let, for $k \in K$, s^k and t^k represent the source and sink vertices for the commodity k, x_{ij}^k and c_{ij}^k be the flow of commodity k along the edge (i,j) and its capacity constraint, let further c_{ij} represent the amount of some common resource available for all commodities combined, w_{ij}^k be the amount of this resource needed for processing one unit of commodity k along the edge (i,j), and let, finally, f^k be the total flow of commodity k leaving the source vertex s^k and reaching the sink vertex t^k. Using this notation and requiring the flows to be restricted to integer values, the integer maximal multicommodity flow problem can be formulated as the following integer linear programming problem:

$$\text{Maximise} \quad \sum_{k \in K} f^k \tag{15}$$

subject to

$$\sum_{j:(i,j)\in E} x_{ij}^k - \sum_{j:(j,i)\in E} x_{ji}^k = \begin{cases} f^k & , \; i = s^k, k \in K \\ 0 & , \; i = V - \{s^k, t^k\}, k \in K \\ -f^k & , \; i = t^k, k \in K \end{cases} \tag{16}$$

$$x_{ij}^k \leq c_{ij}^k , \quad (i,j) \in E, k \in K \tag{17}$$

$$\sum_{k \in K} w_{ij}^k x_{ij}^k \leq c_{ij} , \quad (i,j) \in E \tag{18}$$

$$x_{ij}^k \in Z_+ , \quad (i,j) \in E, k \in K \tag{19}$$

$$f^k \in Z_+ , \quad k \in K \tag{20}$$

Equation (20) can be omitted because this integrality constraint results from (19). However, this implication does not hold in the opposite direction. In spite of an optimal integer-valued solution f^k, the flows along the

edges may be multiples of ½. Furthermore the optimal flows are not guaranteed to be integer when the capacity of each of the edges is integer as it is in the case of a single commodity flow problem (Ahuja et al 1993).

The integer maximal multicommodity flow problem will be solved using the following allocation strategy. Initially, assign the maximum possible integer capacity to each edge $(i,j)\in E$ for commodity 1 and find the maximum flow. Reduce the capacity of each edge (i,j) by the flow for commodity 1, assign the maximum possible integer capacity for each edge (i,j) for commodity 2, and find the maximum flow. Continue the procedure until the maximum flows are found for all commodities.

Consider r_{ij}^{k} as a remaining capacity available to be assigned to commodity k on edge (i,j). Denote $\lceil x \rceil$ the largest integer $\leq x$ and IIF(a,b,c) the "if-then-else" function which returns b or c depending on whether a condition a is satisfied or not. Then we can formulate the following algorithm

ALLOCATION-PROCEDURE

$k := 1$;

$\forall (i,j)\in E$ **do** $r_{ij}^{k} := c_{ij}$;

while $k \leq |K|$ **do**

begin SINGLE-COMMODITY-PROBLEM(

k,

$$x_{ij}^{k} \in \left[0, IIF\left(w_{ij}^{k} = 0, c_{ij}^{k}, \min\left\{ c_{ij}^{k}, \left\lceil \frac{r_{ij}^{k}}{w_{ij}^{k}} \right\rceil \right\} \right) \right], \quad (i,j) \in E,$$

$f^{k'}$ ... maximal flow,

$x_{ij}^{k'}$ ... optimal integer flow along the edge (i,j)**)**;

$k := k+1$;

if $k \leq |K|$

then $\forall (i,j)\in E$ **do** $r_{ij}^{k} := r_{ij}^{k-1} - w_{ij}^{k} x_{ij}^{k'}$

end ;

The values $f^{k'}$, $x_{ij}^{k'}$ for all $(i,j)\in E$ and $k = 1,2,...,|K|$ give a feasible integer solution with a value $f' = \sum_{k\in K} f^{k'}$. The allocation of the capacity of edge (i,j) for commodity k is given by $w_{ij}^{k} x_{ij}^{k'}$ for $k = 1,2,...,|K|-1$ and $c_{ij} - \sum_{k=1}^{|K|-1} w_{ij}^{k} x_{ij}^{k'}$ for commodity $|K|$.

As to the single commodity flow problem, there are many algorithms for solving it, e.g., Goldberg & Tarjan's, Dinic's or Karzanov's algorithm (Ahuja et al 1993), their implementations are described in (Palubják 2003).

It is straightforward that the allocation of the edge capacities among the commodities and the corresponding combined maximal flow depend on the order in which the commodities are selected. The optimal ordering of the commodities in the allocation procedure is a difficult combinatorial problem. For $|K|$ commodities the number of their possible orderings is $|K|!$ because the number of all permutations of $1,2,\ldots,|K|$ is $|K|!$. Therefore, a search for the optimum in the space of permutations is only feasible for a not very high number of commodities.

This means that the time complexity of the problem in question is equal to that of the travelling salesman problem (Gutin and Punnen 2002). For higher numbers of commodities, we must use a heuristic or approximation method. Stochastic heuristics, mainly genetic algorithms, tabu-search and simulated annealing (Michalewicz 1996), (Michalewicz and Fogel 2000), (Reeves 1993), are among them the most popular methods.

The proposed algorithm was implemented using the simulated annealing (SA) technique. Its idea is based on simulating the cooling of a material in a heat bath - a process known as annealing. More details can be found in (Reeves 1993). For short, only note, that its performance is sensitive to the choice of a cooling schedule, i.e. it depends on the initial temperature, final temperature and temperature-reduction function.

Parameters of the simulated annealing were set as follows: Initial temperature: 1800, final temperature: 12 to 290, temperature reduction function $\alpha(t) = t/(1+a\,t)$ where $a = 0.000008$, neighbourhood – shift operation (two randomly selected positions are used in a different way, it removes a value at one position and puts it at another position. An example of this procedure is shown in Figure 1.). Each test was executed 30 times.

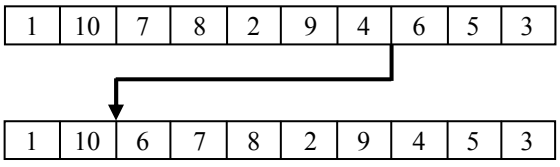

Figure 1. Shift operation for determining a permutation neighbour.

The proposed modules for solving the maximal multicommodity integer flow problem were implemented and tested for three types of networks N_1, N_2, N_3 with the following numbers of vertices and edges: (1) $|V|$=100, $|E|$=300, (2) $|V|$=100, $|E|$=600 and (3) $|V|$=100, $|E|$=900. Each network was generated for 5, 6, … , 11 commodities.

The results are summarized in Table 1.

Table 1: Computational results for 3 types of networks and 5 to11 commodities.

number of products (number of iterations)	network	opt.	the best solution	average solution
5 (120)	N_1	89868	89868	89852
	N_2	15767	15767	15767
	N_3	35143	35143	35143
6 (300)	N_1	95510	95510	95495
	N_2	17192	17192	17181
	N_3	44319	44319	44301
7 (2000)	N_1	97644	97644	97644
	N_2	17578	17578	17578
	N_3	47368	47368	47368
8 (4000)	N_1	96998	96998	96965
	N_2	19769	19769	19757
	N_3	53098	53098	53083
9 (5800)	N_1	87700	87700	87700
	N_2	18909	18909	18818
	N_3	54555	54555	54555
10 (8000)	N_1	90636	90636	90625
	N_2	15897	15897	15875
	N_3	40861	40861	40847
11 (10200)	N_1	105267	105267	104923
	N_2	17333	17333	17324
	N_3	55519	55519	55430

CONCLUSIONS

In this paper we studied network flows and their role for the modelling of other problems and presented for the Minimum Steiner Tree Problem. Besides it, we presented possible extensions of the basic maximum flow problem, e.g., for finding maximum flow with priorities of sources and sinks.

Finally, we studied the integer maximal multi-commodity flow problem, which belongs to the class of NP-hard combinatorial problems. In contrast to obvious approaches, mainly based on deterministic decomposition algorithms, we propose a stochastic heuristic approach using the simulated annealing algorithm.

Computational results show that, for suitable parameter settings presented in the paper, this approach is able to find the optimal solution almost in all cases or at least a solution very close to optimum when the test is executed several times. Furthermore, the proposed algorithm is stable because even the average results gained from all the executions are close to optimum.

In the future, we foresee further tests with other stochastic heuristic methods – genetic algorithms and tabu-search and their parameter settings because their temporary results are worse than the simulated annealing results.

Acknowledgments This research has been supported by the Czech Science Foundation GA ČR in the frame of GA ČR 102/09/1668 project Control Algorithm Design by Means of Evolutionary Approach and the Czech Ministry of Education in the frame of research plan MSM 0021630518 Simulation Modelling of Mechatronic Systems.

REFERENCES

Ahuja, R.K.; T.L. Magnanti; and J.B. Orlin. 1993. *Network Flows. Theory, Algorithms and Applications*. Prentice Hall, Englewood Cliffs, New Jersey.

Du, D.-Z.; J.M. Smith; and J.H. Rubinstein. 2000. *Advances in Steiner Trees*. Kluwer Academic Publishers, Dordrecht.

Gutin, G. and A.P. Punnen (eds.). 2002. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, Dordrecht.

Hochbaum, D. (ed.). 1996. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, Boston.

Hwang, F.K.; D.S. Richards, and P. Winter. 1992. *The Steiner Tree Problem*. North-Holland, Amsterdam.

Lin, Y.K. 2002. "Study on the System Capacity for a Multicommodity Stochastic-Flow Network with Node Failure". *Reliability Engineering and System Safety*, Vol. 78, 2002, 57-62.

Lozovanu, D. and M. Fonoberova. 2006. "Optimal Dynamic Multicommodity Flows in Networks". *Electronic Notes in Discrete Mathematics,* Vol. 25, 93-100.

Michalewicz, Z. and D.B. Fogel. 2000. *How to Solve It: Modern Heuristics*. Springer-Verlag, Berlin.

Michalewicz, Z. 1996. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, Berlin, Germany.

Mrad, M. and M. Haouari. 2008. "Optimal solution of the discrete cost multicommodity network design problem". *Applied Mathematics and Computation*, Vol. 204, No. 2, 745-753.

Ouorou, A. and P. Mahey. 2000. "A Minimum Mean Cycle Cancelling Method for Nonlinear Multicommodity Flow Problems". *European Journal of Operational Research*, Vol. 121, 532-548.

Ouorou, A.; P. Mahey; and J.-P. Vial. 2000. "A Survey of Algorithms for Convex Multicommodity Flow Problems". *Management Science*, Vol. 46, No. 1, 126-147.

Palubják, P. 2003. Multicommodity Network Flows (in Czech). PhD thesis, Brno University of Technology, Faculty of Mechanical Engineering, Brno, Czech Republic, 142 pp.

Plesník, J.. 1983. *Graph Algorithms* (in Slovak). Veda, Bratislava, Slovakia.

Reeves, C.R. 1993. *Modern Heuristic Techniques for Combinatorial Problems*. Blackwell Scientific Publications, Oxford.

Srivastav, A. and P. Stangier. 2000. "On Complexity, Representation and Approximation of Integral Multicommodity Flows". *Discrete Applied Mathematics*, Vol. 99, 183-208.

Wang, D and R. Kleinberg. 2009. "Analyzing Quadratic Unconstrained Binary Optimization Problems via Multicommodity Flows". *Discrete Applied Mathematics*, Vol. 157, No. 18, 3746-3753.

AUTHOR BIOGRAPHY

MILOŠ ŠEDA was born in Uherské Hradiště, Czech Republic. He graduated from Technical University of Brno in 1976 (majoring in Technical Cybernetics) and Masaryk University of Brno in 1985 (majoring in Computer Science). He received his Ph.D. degree in Technical Cybernetics in 1998, Assoc. Prof. degree in Applied Mathematics in 2001 and in 2009 he was appointed professor of Design and Process Engineering at Brno University of Technology. His work is concerned with graph theory, computational geometry, robotics, combinatorial optimisation, scheduling manufacturing processes, project management, fuzzy sets applications and database systems.

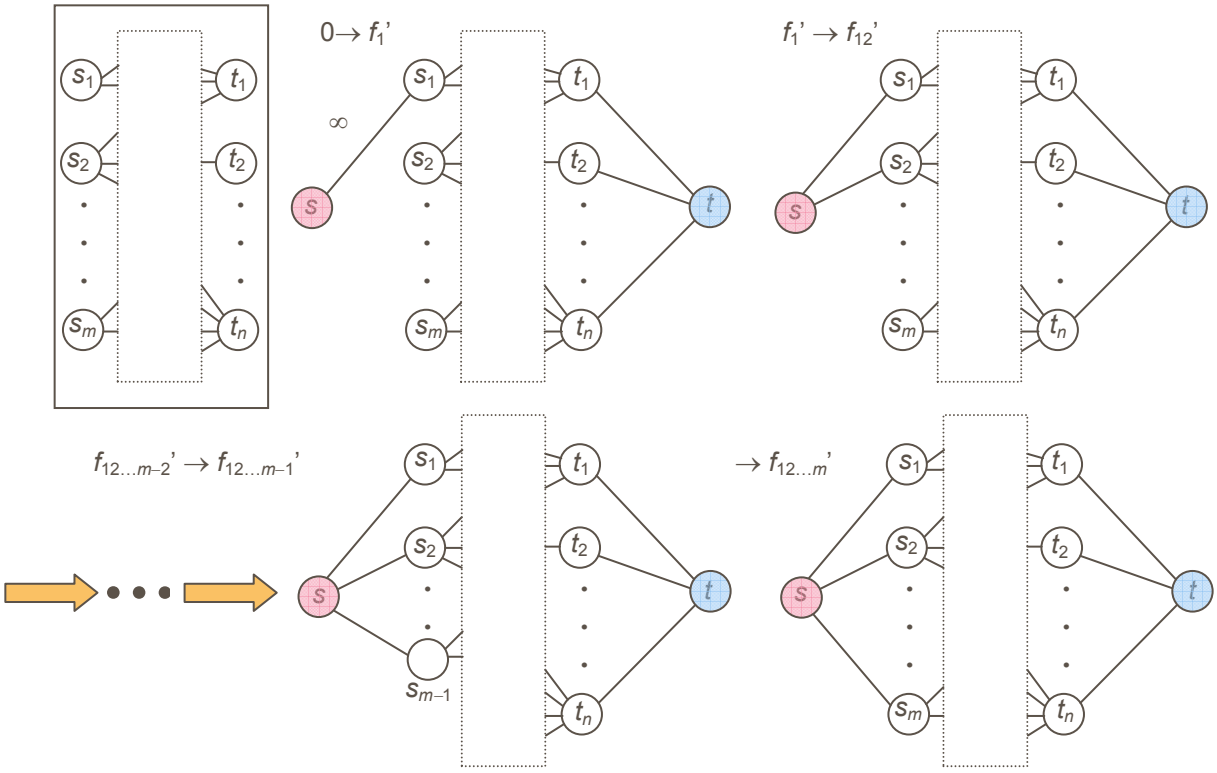

Figure 2. Maximal flow with priorities of sources and sinks