Inhaltsverzeichnis

1 Introduction

Today more and more real-world problems in the areas of simulation and optimization are solved by mathematical and computational methods. A growing number of these problems can be solved without problems, i.e. even huge instances give an optimal or near optimal solution within seconds. Still, there remain problems that even on modern computers are hard to solve. For these problems it is important to find ways to increase the efficiency of the algorithms.
The topic of this thesis arises from the computation of flow in natural gas networks, which is currently developed in the FORNE Project in a cooperation of OGE with universities and research insitutes including ZIB. The flow of natural gas in a network is described by nonlinear equations and depends on many parameters, which makes the problem hard to solve. If we can find good upper and lower bounds for the flow on an arc during the preprocessing, we can hope to improve the behavior of the nonlinear solver by giving these tigther bounds.
The flow is induced by pressure differences, so in reality there can't be cyclic flow (if we exclude compressor stations). Without the condition of acyclic flow, it is sufficient to run a standard min-cost-flow algorithm where the maximized arc e gets weight $w_e = -1$ and all others are 0. However, the arising bounds are far from optimal. If arc e is contained in any cycle we could decrease the cost by pushing more and more flow around this cycle until the arcs capacity is at its limits.
This master thesis will deal with the problem of finding a network flow with no directed cycles (acyclic flow), which at the same time maximizes the amount of flow on a specified arc e of the network. We will discuss the complexity, an exact algorithm based on a mixed integer program with separation of inequalities that forbid cycles and also a heuristic approach that yields results much faster (but not optimal).

1.1 The gas flow problem

Although it is mainly the motivation, not really the topic of this thesis, we want to briefly introduce the gas transport problem. For more a detailed description we refer to LINK

2 Basic Notation and Definitions

Since there are many definitions, which may differ slightly, we want to introduce now the basic notation and definitions that we use throughout this thesis. The definitions in this chapter are mainly taken from the textbook about combinatorial optimization from Korte and Vygen [2].
An undirected graph is a triple (V, E, Ψ), where V and E are finite sets and $\Psi : E \to \{X \subseteq V : |X| = 2\}$. A directed graph or digraph is a triple (V, E, Ψ), where V and E are finite sets and $\Psi : E \to \{(v, w) \in V \times V : v \neq w\}$. In this thesis by a graph we mean normally the directed graph. If we talk about undirected graphs it will be stated explicitly. The elements of V are called vertices, the elements of E are the edges. Edges of undirected graphs can also be called arcs to make clear that they are directed.
Two edges e, e' with $\Psi(e) = \Psi(e')$ are called parallel. Graphs without parallel edges are called simple. For simple graphs we usually identify an edge e with its image $\psi(e)$ and write $G = (V(G), E(G))$, where $E(G) \subseteq \{X \subseteq V(G) : |X| = 2\}$ or $E(G) \subseteq V(G) \times V(G)$. We often use this simpler notation even in the presence of parallel edges, then the "set" $E(G)$ may contain several "identical" elements. In this thesis all graphs are considered simple if nothing different is said. $|E(G)|$ denotes the number of edges, and for two edge sets E and F we always have $|E \cup F| = |E| + |F|$ even if parallel edges arise.
We say that an edge $e = \{v, w\}$ or $e = (v, w)$ joins v and w. In this case, v and w are adjacent. v is a neighbour of w (and vice versa). v and w are the endpoints of e. If v is an endpoint of an edge e, we say that v is incident with e. In the directed case we say that (v, w) leaves v and enters w, v is the tail and w is the head of the arc e. Two edges which share at least one endpoint are called adjacent.
For a digraph G we sometimes consider the underlying undirected graph, i.e. the undirected graph G' on the same vertex set which contains an edge $\{v, w\}$ for each edge (v, w) of G. We also say that G is an orientation of G'. A subgraph of a graph $G = (V(G), E(G))$ is a graph $H = (V(H), E(H))$ with $V(H) \subset V(G)$ and $E(H) \subset E(G)$. We also say that G contains H. H is an induced subgraph of G if it is a subgraph of G and $E(H) = \{\{x, y\}$ resp. $(x, y) \in E(G) : x, y \in V(H)\}$. Here H is the subgraph of G induced by $V(H)$. We also write $H = G[V(H)]$. A subgraph H of G is called spanning if $V(H) = V(G)$. If $v \in V(G)$, we write $G-v$ for the subgraph of G induced by $V(G) \setminus v$. If $e \in E(G)$, we define $G-e := (V(G), E(G) \setminus \{e\})$. Furthermore, the addition of a new edge e is abbreviated by $G + e := (V(G), E(G) \cup e)$. If G and H are two graphs, we denote by $G+H$ the graph with $V(G+H) = V(G) \cup V(H)$ and $E(G + H)$ being the disjoint union of $E(G)$ and $E(H)$ (parallel edges may arise).
For a graph G and $X, Y \subseteq V(G)$ we define $E(X, Y) := \{\{x, y\} \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$ resp. $E^+(X, Y) := \{(x, y) \in E(G) : x \in X \setminus Y, y \in Y \setminus X\}$. For undirected graphs G and $X \subseteq V(G)$ we define $\delta(X) := E(X, V(G) \setminus X)$. The set of neighbours of X is defined by $\Gamma(X) := \{v \in V(G) \setminus X : E(X, \{v\}) \neq \emptyset\}$. For digraphs G and $X \subseteq V(G)$ we define $\delta^+(X) := E^+(x, V(G) \setminus X)$, $\delta^-(x) := \delta^+(V(G) \setminus X)$ and $\delta(X) := \delta^+(x) \cup \delta^-(x)$. We use subscripts (e.g. $\delta_G(X)$) to specify the graph G if necessary.
For singletons, i.e. one-element vertex sets $\{v\}(v \in V(G))$ we write $\delta(v) := \delta(\{v\})$, $\Gamma(v) := \Gamma(\{v\}), \delta^+(v) := \delta^+(\{v\})$ and $\delta^-(v) := \delta^-(\{v\})$. The degree of a vertex v is $|\delta(v)|$, the number of edges incident to v. In the directed case, the in-degree is $|\delta^-(v)|$, the out-degree is $|\delta^+(v)|$, and the degree is $|\delta^+(v)| + |\delta^-(v)|$. A vertex v with zero degree is called isolated. A graph where all vertices have degree k is called k-regular.

An edge progression W in G is a sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ such that $k \geq 0$, and $e_i = (v_i, v_{i+1}) \in E(G)$ resp. $e_i = \{v_i, v_{i+1}\} \in E(G)$ for $i = 1, \ldots, k$. If in addition $e_i \neq e_j \, \forall \, 1 \leq i < j \leq k$, W is called a walk in G. W is closed if $v_1 = v_{k+1}$. A path is a graph $P = (\{v_1, ..., v_{k+1}\}, \{e_1, ..., e_k\})$ such that $v_i \neq v_j$ for $1 \leq i < j \leq k+1$ and the sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_{k+1}$ is a walk. P is also called a path from v_1 to v_{k+1} or a $v_1 - v_{k+1}$-path. v_1 and v_{k+1} are the endpoints of P. By $P_{[x,y]}$ with $x, y \in V(P)$ we mean the (unique) subgraph of P which is an $x - y$-path. Evidently, there is an edge progression from a vertex v to another vertex w if and only if there is a $v - w$-path.

A cycle is a graph $(\{v_1, \ldots, v_k\}, \{e_1, \ldots, e_k\})$ such that the sequence $v_1, e_1, v_2, \ldots, v_k, e_k, v_1$ is a (closed) walk and $v_i \neq v_j$ for $1 \leq i < j \leq k$. An easy induction argument shows that the edge set of a closed walk can be partitioned into edge sets of cycles.

The length of a path or cycle is the number of its edges. If it is a subgraph of G, we speak of a path or cycle in G. A spanning path in G is called a Hamiltonian path while a spanning cycle in G is called a Hamiltonian cycle or a tour. A graph containing a Hamiltonian cycle is a Hamiltonian graph. For two vertices v and w we write $dist(v, w)$ or $dist_G(v, w)$ for the length of a shortest $v - w$-path (the distance from v to w) in G. If there is no $v - w$-path at all, i.e. w is not reachable from v, we set $dist(v, w) := \inf$. In the undirected case, $dist(v, w) = dist(w, v)$ for all $v, w \in V(G)$.

We shall often have a cost function $c : E(G) \to \mathbb{R}$. Then for $F \subseteq E(G)$ we write $c(F) := \sum_{e \in F} c(e)$ (and $c(\emptyset) = 0$). This extends c to a modular function $c : 2^{E(G)} \to \mathbb{R}$. Moreover, $dist_{(G,c)}(v, w)$ denotes the minimum $c(E(P))$ over all $v - w$-paths P in G.

3 The Acyclic Flowbound Problem

We already described the gas flow problem, where the motivation for this thesis came from. Here we want to define the problem as a general combinatorial flow problem with specific contraints. We will also give the formulation as a Mixed Integer Program (MIP) and as well some natural relaxations, which might be easier to solve.
We will represent our originally undirected graph by a directed graph where flow is allowed to go over edges backward and forward as well. This allows us to specify directions forward and backward on every edge in a consistant way.

Definition 3.1. Let $G = (V, A)$ be a directed Graph and $e \in A$ a specific arc of G. For every vertex $v \in V$ let there be a prescribed amount of flow $b(v) \in \mathbb{R}$ entering or leaving the network, where $\sum_{v \in V} b(v) = 0$.
Let there be capacities $c_l(a) \leq 0 \leq c_u(a) \, \forall a \in A$, a flow $f : A \to \mathbb{R}$ with $c_l(a) \leq f(a) \leq c_u(a) \, \forall a \in A$ and $\sum_{a \in \delta^+(v)} f(a) - \sum_{a \in \delta^-(v)} f(a) + b(v) = 0 \, \forall v \in V$. We call this flow on such a graph a *feasible network flow on G*.

Definition 3.2. A *cyclic flow* within such a feasible network flow f is a flow $f' : C \to \mathbb{R}$, where $C \subseteq A$ is a cycle, $\sum_{a \in \delta^+(v) \cap C} f(a) - \sum_{a \in \delta^-(v) \cap C} f(a) = 0 \, \forall v \in C$, for all arcs the flow direction in f' and f are the same, i.e. $f(a) \geq 0 \Rightarrow f'(a) \geq 0$, $f(a) \leq 0 \Rightarrow f'(a) \leq 0$ and $f'(a) \neq 0 \, \forall a \in C$.

Definition 3.3. The problem of finding a flow $f : A \to \mathbb{R}$ with $f(e) \geq f'(e) \, \forall f : A \to \mathbb{R}$s.t.

4 Complexity

In order to talk about algorithms it is always good to kno the complexity of the underlying problem. This chapter will deal with this issue. We show, that Cycle-free Min-Cost-Flow with negative weights is NP-complete, and that our problem is just a special case.

5 Implementation

6 Practical Results

7 Conclusion

Literatur

[1] Martin Grötschel, Michael Jünger, and Gerhard Reinelt, *On the acyclic subgraph polytope*, Mathematical Programming **33** (1985), no. 1, 28–42 (English).

[2] Bernhard Korte and Jens Vygen, *Combinatorial optimization: Theory and algorithms*, 4th ed., Springer Publishing Company, Incorporated, 2007.