Efficient Implementations of Minimum Cost Flow Algorithms
Péter Kovács, PhD student
ELTE CNL, kpeter@inf.elte.hu, supervisor: Zoltán Király

The Minimum Cost Flow Problem
- The MCF problem is one of the most fundamental models in network flow theory.
- It is to find a feasible flow of a given value \( k \) with minimum total cost from a source node \( s \) to a target node \( t \) in a network with capacity constraints and arc costs.
- In most cases all data are integral and we search for an integral flow.
- Applications:
  - network design,
  - VPN allocation,
  - traffic engineering,
  - resource planning, etc.

Cost Scaling
- The cost scaling (COS) algorithm is a generalization of the preflow-push algorithm by Goldberg and Tarjan.
- At each iteration it performs local push and relabel operations.
- It successively achieves an ε-optimal flow from an ε-optimal flow.
- We implemented this algorithm with various efficient heuristics that highly affect its real-time performance.

Network Simplex
- The network simplex (NS) algorithm is a specialized variant of the simplex method.
- In the base solutions the arcs with non-trivial flow value form a spanning tree.
- The crucial points of the implementation are the data structure used for representing spanning trees with associated data and the pivot rules.
- The following charts compare the four pivot rules we implemented.

Implementation and Testing
- We implemented six different algorithms and many variants for the MCF problem.
- We used the LEMON C++ library, http://lemon.cs.elte.hu.
- Our codes are part of the library now.
- Validity and benchmark tests were performed on various large scale random networks (up to 1 million nodes and 30 million arcs).
- The test graphs were generated with NETGEN.

Cycle Canceling
- Cycle Canceling (CC) is the simplest solution method for MCF.
- It is a primal method:
  - find a feasible solution;
  - at each step find a directed cycle with negative cost in the residual network and augment flow on it to saturate an arc.
- We implemented two CC algorithms:
  - a simple cycle-canceling (SCC) algorithm that uses Bellman-Ford algorithm for finding negative cycles and runs in \( O(knmCU) \) time;
  - the minimum mean cycle-canceling (MMCC) algorithm, which runs in strongly polynomial \( O(n^3m^2 \log n) \) time.

Dual Algorithms
- We implemented two efficient dual algorithms:
  - the successive shortest path (SSP) algorithm,
  - the capacity scaling (CAS) algorithm.
- The SSP algorithm maintains an optimal flow of value \( k \leq k \) and node potentials.
- At each step it sends flow from \( s \) to \( t \) along a shortest path in the residual network with respect to reduced costs.
- It increases the flow value until the solution becomes feasible.
- It performs \( O(kU) \) iterations, so it runs in \( O(nU SP(n, m, nC)) \) time.
- The CAS algorithm is a more efficient variant of SSP that uses capacity scaling and runs in \( O(m \log U SP(n, m, nC)) \) time.

Comparison
- We combined the two fastest implementations: for rather sparse graphs Candidate List rule is used, otherwise Block Search rule is used.

Conclusions
- There is no absolute winner.
- In most cases, especially on dense graphs NS proved to be the fastest among our implementations.
- On rather large and sparse graphs COS is the most efficient.
- CAS is usually fast, especially if the the arc capacities are relatively small.
- NS proved to be even more efficient than LEDA when
  - the network is small (it has at most 2000-5000 nodes) or
  - the network is not too sparse.
- However on sparse graphs LEDA is usually faster than all of our implementations.