Network Models

Ufuk Bahçeci

v0.24.10.01

Network Models

MIT License

Network Models

Author

- Ufuk Bahçeci, Ph. D. (Industrial Engineering, University of Galatasaray)

Table of Contents

Graph

Definition

Graph

Given a list of locations, a graph is a structured representation of the locations and the relationships between them.

Network Flow

Definition

Network flow

Network flow is the sending of a certain amount of assets from one location to another on the graph.

Mathematical Programming

Definition

Mathematical programming

Mathematical programming is the optimization of problems formulated as minimization (or maximization) of an objective function subject to a set of constraints.

Combinatorial Optimization

Definition

Combinatorial optimization

Combinatorial optimization is a class of mathematical programming, where optimization is performed over a discrete set of feasible solutions.

Network Flow Problem

Definition

Network flow problem

Network flow problems are mathematical programming problems that can be converted into combinatorial optimization problems dealing with network flows.

Mathematical Optimization

- Linear programming
  - Simplex algorithm
  - Duality
- Decomposition methods
  - Dantzig-Wolfe (complicating constraints, column(extreme point) generation, duality gap between upper and lower bounds)
  - Benders (complicating variables, cut generation, duality gap between upper and lower bounds)
- Mixed-integer programming
  - Branch-and-bound (BaB)
  - BaB + Cutting planes = Branch-and-cut
  - BaB + Column(variable for pricing, extreme point for decomposition) generation = Branch-and-price
  - BaB + Cutting planes + Column generation = Branch-price-and-cut

- Constraint programming
  - Constraint propagation
  - Domain reduction
- Combinatorial optimization
  - Some problems are easy to solve
    - Special fast algorithms
  - Some problems are hard to solve
    - Mixed-integer programming
    - Heuristics

Motivations

Network Flow Problems

- Network flow problems
  - Combinatorial optimization
  - Wide application area in Operations Research
  - Special fast algorithms suitable for large problem instances
  - Network flow problem as an embedded subproblem

Graph
Definition

Graph [1]

A graph $G(V, E)$ consists of a set of vertices V and edges E. Edges are used to model the relationship between vertices.

Graph

Definition

Graph [2]

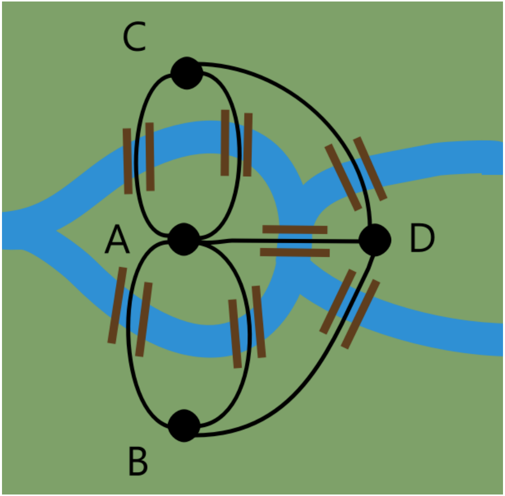
A graph $G(N, A)$ consists of a set of nodes N and arcs A. Arcs are used to model the relationship between nodes.

Subgraph

A graph $G'(V', E')$ is a subgraph of $G(V, E)$ if $V' \subset V$ and $E' \subset E$.

Graph

Example



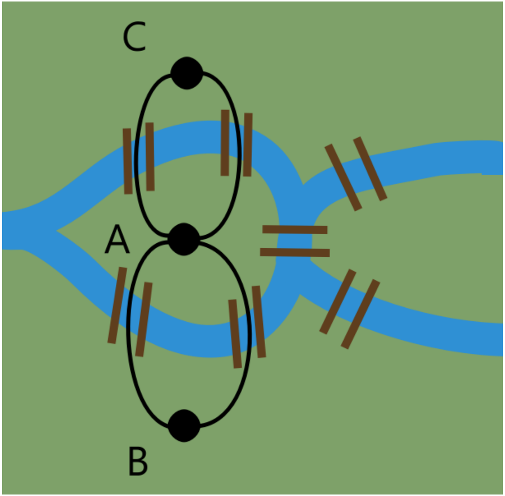
Graph

The Euler's problem

- Is it possible to start from a vertex, move along all edges, traversing every edge only once, and finally return to the starting vertex?

Graph

Example



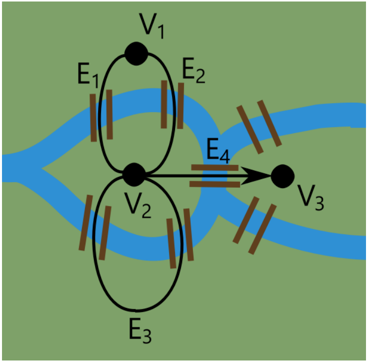
Graph

The Hamilton's problem

- Is it possible to start from a vertex, visit each of all vertices exactly once, and finally return to the starting vertex?

Graph

Directed edges, multiple edges and loops



- E_1 and E_2 are multiple edges
- E_3 is a loop
- E_4 is a directed edge
- V_2(tail) and V_3(head) are the endpoints of the edge(arc) E_4.

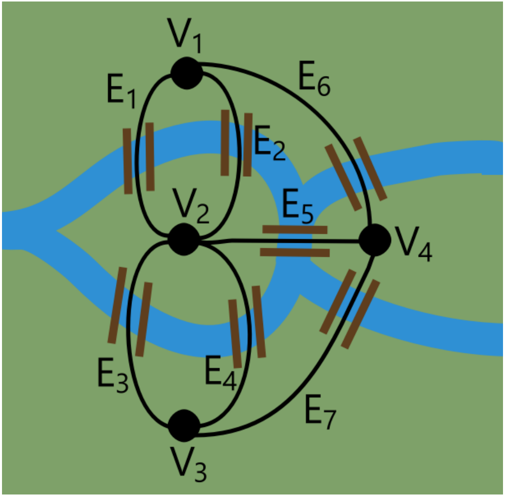
Graph

Graph types [1]

Type	Edges	Multiple edges	Loops
Simple graph	Undirected	✘	✘
Multigraph	Undirected	✔	✘
Pseudograph	Undirected	✔	✔
Simple directed graph	Directed	✘	✘
Directed multigraph	Directed	✔	✔
Mixed graph	Directed and undirected	✔	✔

Graph

A multigraph

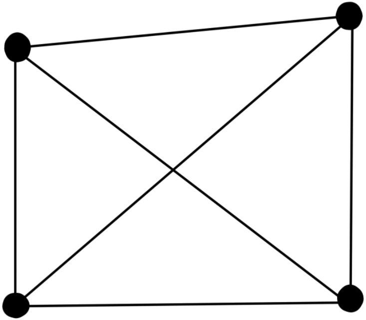


Complete graph [1]

Complete graph is a simple graph where each pairs of distinct vertices are connected.

Graph

A complete graph

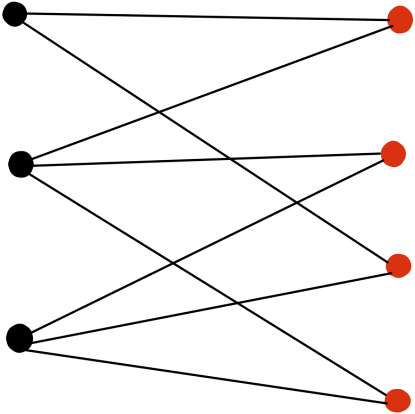


Bipartite simple graph [1]

A simple graph $G(V, E)$ is bipartite if $\exists V_1, V_2 : V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$ such that every edge in E connects a vertex in V_1 to a vertex in V_2.

Graph

A bipartite simple graph



Matching [1]

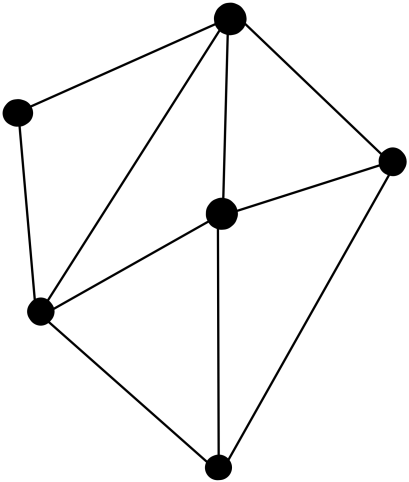
A matching M in a simple graph $G(V, E)$ is a subset of E, i.e. $M \subseteq E$ such that $\forall m, m' \in M$, all the endpoints of m and m' are distinct vertices.

Maximal matching

The maximal matching of G is the matching with the largest $|M|$.

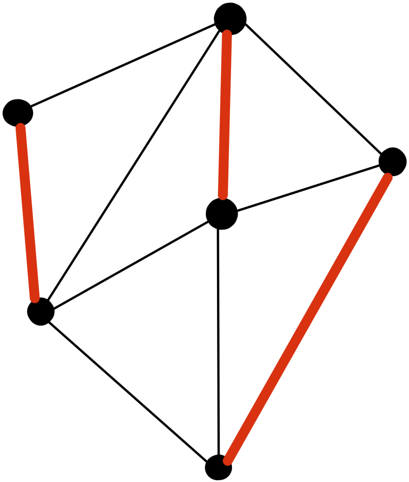
Graph

A simple graph



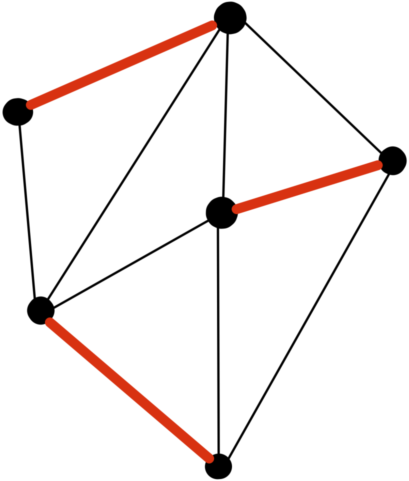
Graph

A maximal matching



Graph

Another maximal matching



Graph
Definitions

Adjacent vertices in an undirected graph

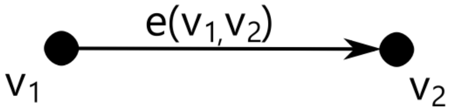
Two vertices are adjacent in an undirected graph G if they are endpoints of an edge in G.

Graph

Definitions

Adjacent vertices in a directed graph [1]

In a directed graph G, the vertex v_1 is adjacent to the vertex v_2 if they are endpoints of a directed edge $e(v_1, v_2)$ in G. Or, it can be said that v_2 is adjacent from v_1.



Graph

Definitions

An edge of an undirected graph G is incident with the vertices that are endpoints of this edge.

Graph

Definitions

Degree of a vertex in an undirected graph [1]

The degree of a vertex v in an undirected graph G, $deg(v)$ is equal to the number of edges incident with the vertex v, where a loop is equivalent to two edges.

Graph
Definitions

Given an undirected graph $G(V, E)$

$\sum_{v \in V} deg(v) = 2|E|$

Graph

Definitions

Degree of a vertex in a directed graph [1]

The indegree(outdegree) of a vertex v in a directed graph G, $deg^{-}(v)(deg^{+}(v))$ is equal to the number of edges with v as their terminal(initial) vertex.

Graph

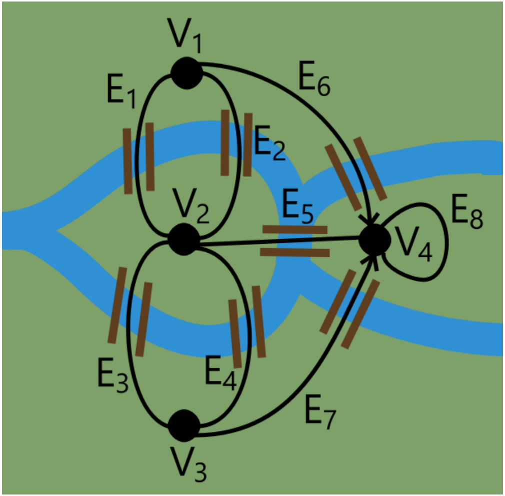
Definitions

Given a directed graph $G(V, E)$

$\sum_{v \in V} deg^{-}(v) = \sum_{v \in V} deg^{+}(v) = |E|$

Graph

A mixed graph



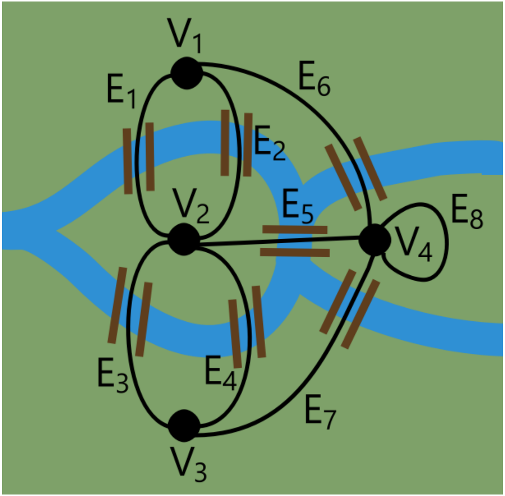
Graph

Adjacency matrix

$$
\begin{array}{ccccc}
 & V_1 & V_2 & V_3 & V_4 \\
V_1 & 0 & 2 & 0 & 1 \\
V_2 & 2 & 0 & 2 & 1 \\
V_3 & 0 & 2 & 0 & 1 \\
V_4 & 0 & 1 & 0 & 1
\end{array}
$$

Graph

A pseudograph



Graph

Incidence matrix

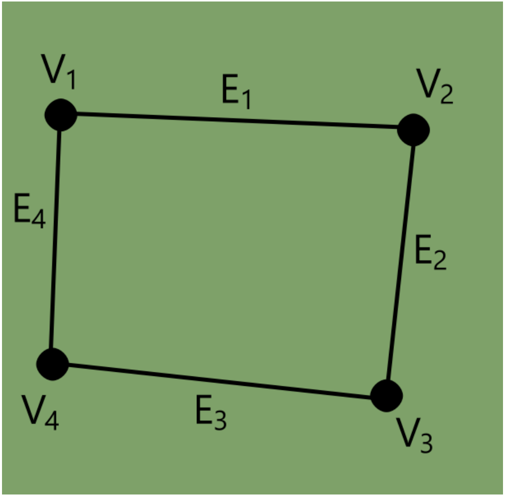
	E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8
V_1	1	1	0	0	0	1	0	0
V_2	1	1	1	1	1	0	0	0
V_3	0	0	1	1	0	0	1	0
V_4	0	0	0	0	1	1	1	1

Isomorphism of graphs [1]

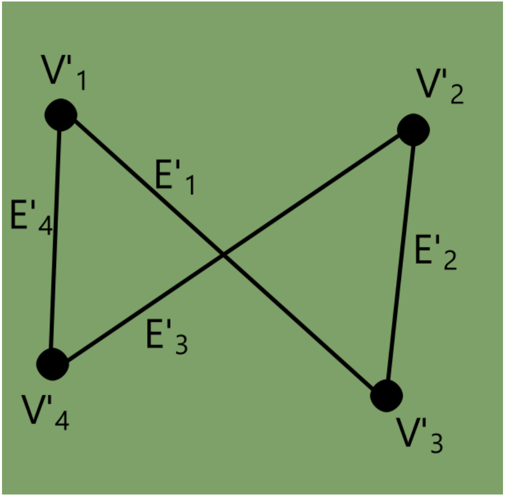
Two simple graphs $G(V, E)$ and $G'(V', E')$ are isomorphic if and only if there exists a permutation of V', denoted as V'^p, leading to $G'^p(V'^p, E')$, where G and G'^p have the same adjacency matrix.

Graph

$G(V, E)$



Graph
$G'(V', E')$



Walk [2]

A walk is a series of vertices that are connected to each other by means of edges.

Simple walk (trail) [1]

A simple walk (trail) is a walk that does not contain the same edge more than once.

Directed walk [2]

A directed walk is a series of vertices that are connected to each other by means of edges in a way that respects the edge directions.

Path [2], [1]

A path is a walk that visits each vertex in the walk only once. A path is also a trail.

Directed path [2]

A directed path is a directed walk that visits each vertex in the directed walk only once.

Circuit [2], [1]

A circuit(closed walk) is a walk of length strictly positive that starts and ends at the same vertex. A simple circuit does not contain the same edge more than once.

Cycle [2]

A cycle is a closed path.

Directed circuit

A directed circuit(closed directed walk) is a directed walk of length strictly positive that starts and ends at the same vertex. A simple directed circuit does not contain the same edge more than once.

Directed cycle [2]

A directed cycle is a directed closed path.

Connected [1]

An undirected graph $G(V, E)$ is connected when a walk exists between each pair of vertices $v, v' \in V^2$ and $v \neq v'$.

Connected [1]

An directed graph $G(V, E)$ is strongly connected when a directed walk exists between each pair of vertices $v, v' \in V^2$ and $v \neq v'$. Let $G'(V, E')$ be the undirected graph obtained by replacing directed edges of G with undirected edges. G is weakly connected if G' is connected.

Graph

Definitions

Network [2]

A network is a graph where vertices and edges have associated properties in the form of numerical values.

The length of a walk [1]

The length of a walk is equal to the sum of the weights of its edges.

The number of walks [1]

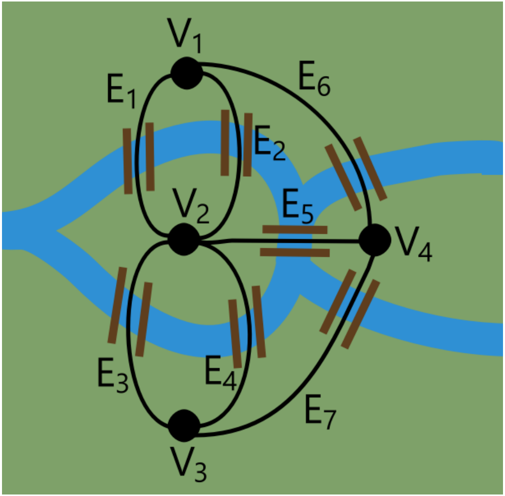
Let A be the adjacency matrix of a graph $G(V, E)$, then the cell with index (i, j) of the matrix A^d is equal to the number of walks of length $d \in \mathbb{Z}^+$ from v_i to v_j, where $v_i, v_j \in V^2$.

Euler walk and circuit [1]

A simple circuit traversing all edges of a graph G is an Euler circuit.
Similarly, a simple walk traversing all edges of a graph G is an Euler walk.

Graph

Can you find an Euler circuit in this multigraph?



An Euler circuit exists..[1]

An Euler circuit exists in a connected multigraph $G(V, E)$ with $|V| \geq 2$ if and only if $\forall v \in V,\ deg(v) \equiv 0 \,(mod\, 2)$.

An Euler walk exists..[1]

An Euler walk but not an Euler circuit exists in a connected multigraph $G(V, E)$ if and only if $\exists\, v', v'' \in V^2$, $v' \neq v''$, $deg(v') \equiv 1 \,(mod\, 2)$, $deg(v'') \equiv 1 \,(mod\, 2)$, and $\forall v \in V \setminus \{v', v''\}$, $deg(v) \equiv 0 \,(mod\, 2)$.

Graph

Definitions

Chinese postman (route inspection) problem

Chinese postman problem looks for the shortest circuit traversing every edge of a connected multigraph at least once.

Chinese postman problem

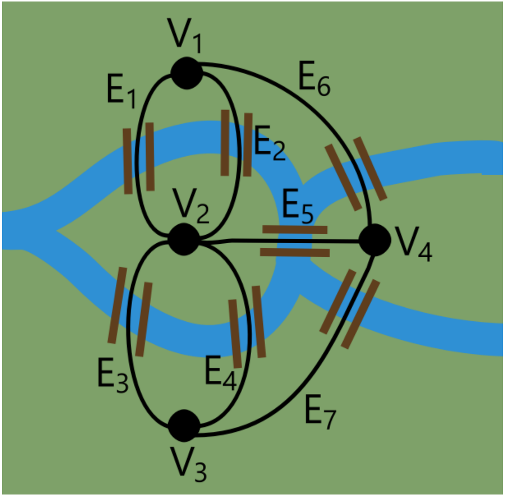
What if an Euler circuit exists in a connected multigraph?

Hamilton path and cycle [1]

A simple circuit visiting every vertex of a graph G exactly once is an Hamilton cycle. Similarly, a simple walk visiting every vertex of a graph G exactly once is an Hamilton path.

Graph

Can you find an Hamilton cycle in this multigraph?



Graph

Definitions (Dirac's theorem)

An Hamilton cycle exists..[1]

An Hamilton cycle exists in a graph $G(V, E)$ if G is a simple graph with $|V| \geq 3$ and $\forall v \in V$, $deg(v) \geq \frac{|V|}{2}$.

Traveling salesman problem

Traveling salesman problem looks for the shortest circuit visiting every vertex of a connected graph exactly once.

Traveling salesman problem

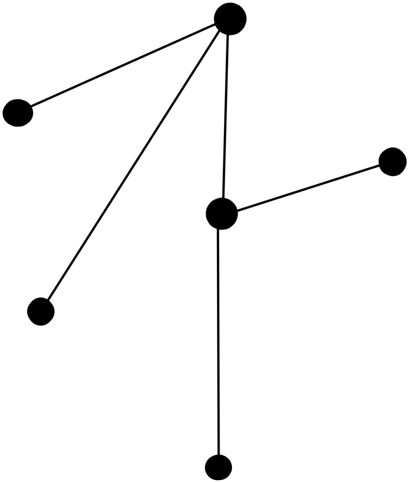
What about the feasible solutions of a traveling salesman problem if it is defined on a complete simple graph with more than 3 vertices? Is this problem feasible?

Tree [2]

A connected graph that contains no cycle is called tree.

Graph

A tree

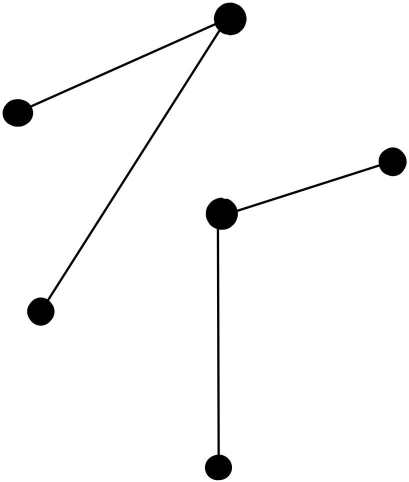


Forest [2]

A collection of trees is called forest.

Graph

A forest



Graph
Definitions

The number of edges in a tree

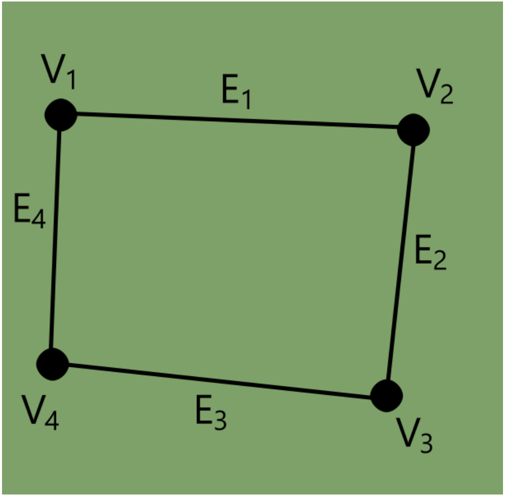
If the graph $G(V, E)$ is a tree than $|E| = |V| - 1$

Planar graph [1]

A planar graph can be drawn in two dimensions without any edges intersecting each other.

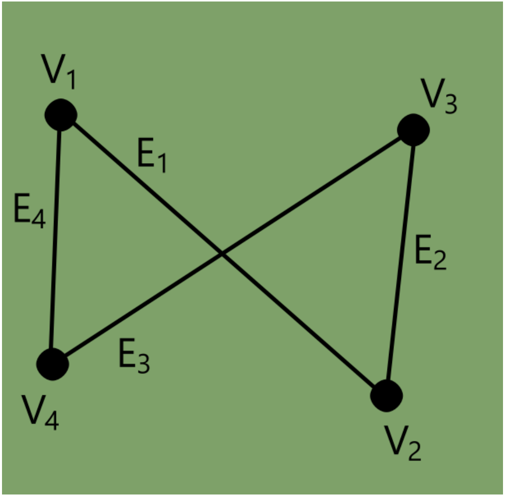
Graph

Planar representation of a planar graph



Graph

Non-planar representation of a planar graph

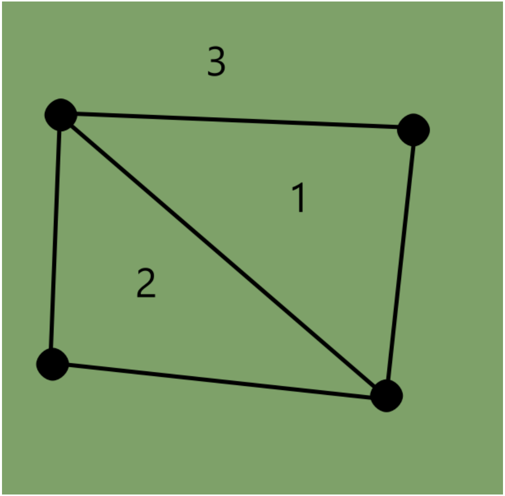


Euler's formula [1]

A connected planar simple graph $G(V, E)$ has $|E| - |V| + 2$ regions.

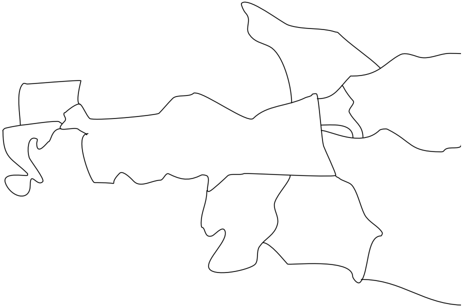
Graph

$3(= 5 - 4 + 2)$ regions of a planar graph



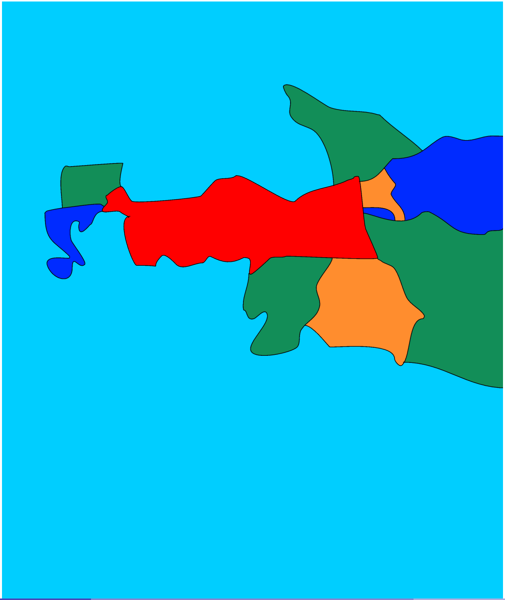
Graph

Map coloring example



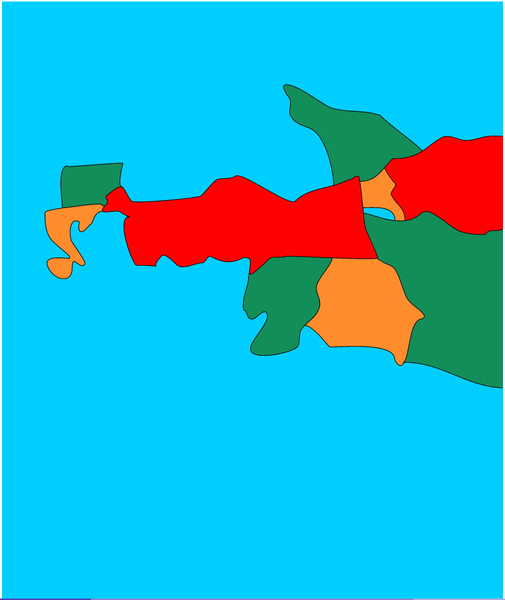
Graph

Map coloring example I (5 colors)



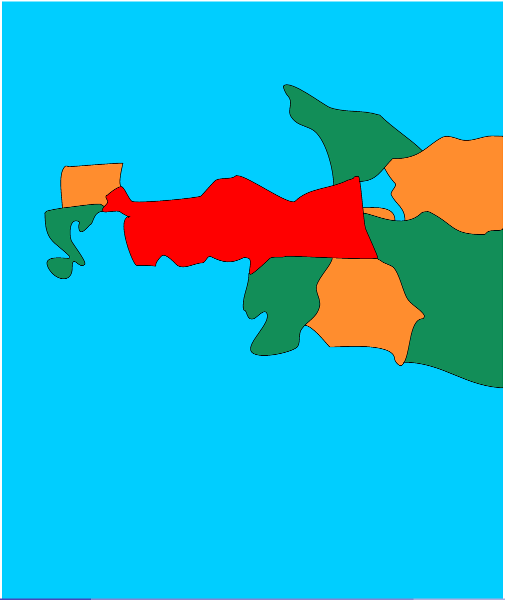
Graph

Map coloring example II (4 colors)



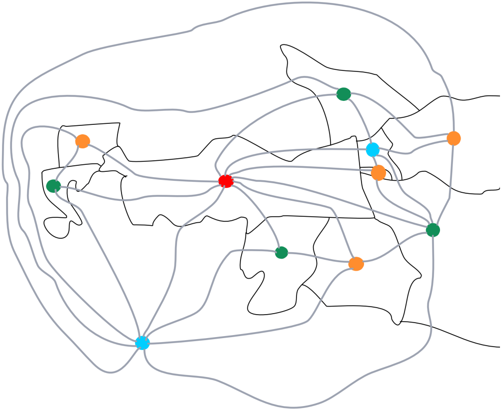
Graph

Map coloring example III (4 colors)



Graph

Dual graph (III) (4 colors)

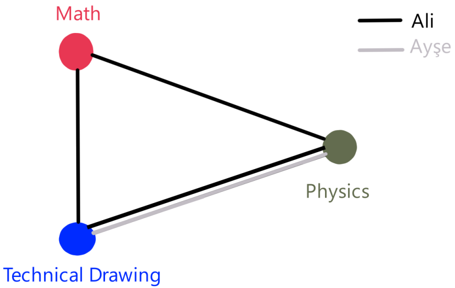


The four color theorem [1]

The chromatic number (minimum number of colors) of a planar simple graph ≤ 4.

Graph

Graph coloring example



Network Problems

Minimum cost flow problem

Minimum cost flow problem

Let $G(V, E)$ be a directed graph with costs $c_{vv'}$ and capacities $u_{vv'}$ defined on edges $vv' = e \in E$, where $v \neq v'$, v and $v' \in V$. Let $b_v > 0$ be the supply and $b_v < 0$ be the demand associated with each vertex $v \in V$. Moreover, $x_{vv'}$ denotes the amount of flow from a vertex v to another vertex v'. Then, minimum cost flow problem minimizes the total cost incurred from all flows in G satisfying both flow conservation constraints and flow limits.

Minimum cost flow problem

$$
\begin{aligned}
min & \sum_{vv' \in E} c_{vv'} x_{vv'} & \\
s.t. & \sum_{v': vv' \in E} x_{vv'} - \sum_{v': v'v \in E} x_{v'v} = b_v & \forall v \in V \\
& 0 \leq x_{vv'} \leq u_{vv'} & \forall vv' \in E
\end{aligned}
$$

Network Problems

Minimum cost flow problem

Assumptions [2]

- $\forall e \in E,\ c_e \in \mathcal{Z}_0^+$
- $\forall v \in V,\ b_v \in \mathcal{Z}$ and $\sum_v b_v = 0$
- $\forall e \in E,\ u_e \in \mathcal{Z}_0^+$
- $\forall v, v' \in V^2$, $\exists$ an uncapaciated directed path from v to v'

Network Problems
Definitions

Polynomial time algorithm

A polynomial time algorithm has a running time polynomial in the length (number of bits) of the input.

Pseudo-polynomial time algorithm

A pseudo-polynomial time algorithm has a running time polynomial in the numeric value (largest value) of the input.

Pseudo-polynomial time algorithms [2]

- Cycle-canceling with $\mathcal{O}(|E|CU)$ iterations
- Successive shortest path with $\mathcal{O}(|V|U)$ iterations
- Primal-dual algorithm with $\mathcal{O}(min(|V|U, |V|C))$ iterations
- Out-of-kilter with $\mathcal{O}(|V|U)$ iterations
- Relaxation

where, $c_e \leq C, \forall e \in E$ and $u_e \leq U, \forall e \in E$

Network Problems

Minimum cost flow problem

Complexity of some minimum cost flow algorithms [3]

- Ford and Fulkerson, $\mathcal{O}(|V|^4 CU)$
- Out-of-kilter, $\mathcal{O}(|E|^3 U)$
- Successive shortest path, $\mathcal{O}(|V|^2|E|U)$
- Cycle-cancelling, $\mathcal{O}(|V||E|^2 CU)$
- Cost-scaling (generic), $\mathcal{O}(|V|^2|E|log(|V|C))$
- Cancel-and-tighten (dynamic trees), $\mathcal{O}(|V||E|log(|V|)min(log(|V|C, |E|log(|V|))))$
- Primal network simplex (dynamic trees), $\mathcal{O}(|V||E|log(|V|)min(log(|V|C, |E|log(|V|))))$
- Dual network simplex (Orlin), $\mathcal{O}(|E|(|E| + |V|log|V|)min(log(|E|U), |E|log(|V|)))$
- Dual network simplex (Armstrong and Jin), $\mathcal{O}(|V||E|log|V|(|E| + |V|log|V|))$

Study of minimum cost flow algorithms [3]

Cost-scaling and primal network simplex were both efficient and robust.

Study of seven state-of-the-art algorithms [4]

- Simple cycle canceling
- Minimum mean cycle canceling
- Cancel and tighten
- Successive shortest path
- Capacity scaling
- Network simplex
- Cost scaling

where, network simplex was the fastest algorithm in $\approx 75\%$ of the studied cases

Maximum flow problem

Let $G(V, E)$ be a directed graph with capacities $u_{vv'} \geq 0$ defined on edges $vv' = e \in E$, where $v \neq v'$, v and $v' \in V$. Let $b_v > 0$ be the supply and $b_v < 0$ be the demand associated with each vertex $v \in V$. Moreover, $x_{vv'}$ denotes the amount of flow from a vertex v to another vertex v'. Then, maximum flow problem maximizes the amount of flow from the source vertex $s \in V$ to the sink vertex $t \in V$, $s \neq t$, and all flows in G satisfy both flow conservation constraints and flow limits.

Maximum flow problem

$$
\begin{aligned}
& max \quad \alpha \\
& s.t. \quad \sum_{v':vv' \in E} x_{vv'} - \sum_{v':v'v \in E} x_{v'v} = \begin{cases} \alpha & for \quad v = s \\ 0 & \forall v \in V \setminus \{s, t\} \\ -\alpha & for \quad v = t \end{cases} \\
& \qquad 0 \le x_{vv'} \le u_{vv'} \qquad \forall vv' \in E
\end{aligned}
$$

Network Problems

Maximum flow problem

Special case of minimum cost flow problem

- Maximum flow problem from s to t on $G(V, E)$
- Add $b_v = 0, \forall v \in V$
- Add $c_e = 0, \forall e \in E$
- Add a new edge ts with $c_{ts} = -1$ and $u_{ts} = \infty$
- $E' = E \cup \{ts\}$
- Minimum cost flow problem on $G'(V, E') \equiv$ Maximum flow problem on $G(V, E)$

Network Problems

Maximum flow problem

Assumptions [2]

- $\forall e \in E,\ u_e \in \mathcal{Z}_0^+$
- $\nexists$ an uncapaciated directed path from s to t
- If $vv' \in E$ than $v'v \in E$
- No multiple edges

Running times of maximum flow algorithms [2]

- Labeling, $\mathcal{O}(|V||E|U)$
- Capacity scaling, $\mathcal{O}(|V||E|log(U))$
- Successive shortest path, $\mathcal{O}(|V|^2|E|)$
- Generic preflow-push, $\mathcal{O}(|V|^2|E|)$
- FIFO preflow-push, $\mathcal{O}(|V|^3)$
- Highest-label preflow-push, $\mathcal{O}(|V|^2\sqrt{|E|})$
- Excess scaling, $\mathcal{O}(|V||E| + |V|^2 log(U))$

Running time of an almost linear time algorithm [5] for minimum cost flows and maximum flows

- Demands, costs and capacities are bounded polynomially
- Demands, costs and capacities are integral
- Runs in $m^{1+\mathcal{O}(1)}$ time

Network Problems

Maximum flow problem

Feasible flow problem

$$\sum_{v':vv'\in E} x_{vv'} - \sum_{v':v'v\in E} x_{v'v} = b_v \qquad \forall v$$

$$0 \leq x_{vv'} \leq u_{vv'} \qquad \forall vv' \in E$$

Procedure to create a transformed network $G'(V', E')$ [2]

- Add the vertex s
- $\forall v \in V$ with $b_v > 0$, add the edges sv with $u_{sv} = b_v$
- Add the vertex t
- $\forall v \in V$ with $b_v < 0$, add the edges vt with $u_{vt} = -b_v$
- $V' = V \cup \{s, t\}$
- $E' = E \cup \{sv : v \in V,\ b_v > 0\} \cup \{vt : v \in V,\ b_v < 0\}$

Maximum flow problem on the transformed network $G'(V', E')$

$$\max \quad \alpha$$

$$s.t. \quad \sum_{v':vv' \in E'} x_{vv'} - \sum_{v':v'v \in E'} x_{v'v} = \begin{cases} \alpha & \text{for} \quad v = s \\ 0 & \forall v \in V' \setminus \{s, t\} \\ -\alpha & \text{for} \quad v = t \end{cases}$$

$$0 \le x_{vv'} \le u_{vv'} \qquad \forall vv' \in E'$$

Network Problems

Maximum flow problem

Feasible flow problem

If α^* of the maximum flow problem on the transformed network $G'(V', E')$ is equal to $\sum_{v \in V,\, b_v > 0} b_v$ than the flow problem is feasible.

Maximum flow problem with lower bounds on $G(V, E)$

$$
\begin{aligned}
& max \quad \alpha \\
& s.t. \quad \sum_{v':vv' \in E} x_{vv'} - \sum_{v':v'v \in E} x_{v'v} = \begin{cases} \alpha & for \quad v = s \\ 0 & \forall v \in V \setminus \{s, t\} \\ -\alpha & for \quad v = t \end{cases} \\
& \qquad l_{vv'} \leq x_{vv'} \leq u_{vv'} \qquad \forall vv' \in E
\end{aligned}
$$

Network Problems

Maximum flow problem

Procedure to create a circulation network $G^c(V, E^c)$ [2]

- Add the edge ts with $u_{ts} = \infty$
- $E^c = E \cup \{ts\}$

so that it is possible to send the flow from s to t back to s from t by using the edge ts with $u_{ts} = \infty$.

Network Problems

Maximum flow problem

Circulation problem (a feasible flow of the maximum flow problem with lower bounds) [2]

$$\sum_{v':vv'\in E^c} x_{vv'} - \sum_{v':v'v\in E^c} x_{v'v} = 0 \qquad \forall v \in V$$

$$l_{vv'} \leq x_{vv'} \leq u_{vv'} \qquad \forall vv' \in E^c$$

Network Problems

Maximum flow problem

Transformed ($x_{vv'} = x'_{vv'} + l_{vv'}$) circulation problem [2]

$$\sum_{v':vv' \in E^c} x'_{vv'} - \sum_{v':v'v \in E^c} x'_{v'v} = b_v \qquad \forall v \in V$$

$$0 \leq x'_{vv'} \leq u_{vv'} - l_{vv'} \qquad \forall vv' \in E^c$$

where $b_v = \sum_{v':v'v \in E^c} l_{v'v} - \sum_{v':vv' \in E^c} l_{vv'} \qquad \forall v \in V$

Feasible flow problem

A feasible flow can be found by solving a maximum flow problem on the transformed network ${G^c}'(V', {E^c}')$.

Network Problems

Maximum flow problem

Residual capacities on $G(V, E)$ [2]

A residual capacity of an edge vv' is denoted as $r_{vv'} = (u_{vv'} - x_{vv'}) + (x_{v'v} - l_{v'v})$, where $x_{vv'}$'s and $x_{v'v}$'s are the feasible flows found in the previous step.

Maximum flow problem with residual capacities on $G(V, E)$

Solve the maximum flow problem with residual capacities on $G(V, E)$. Note that the residual capacity $r_{vv'}$ denotes the maximum possible increase in flow for the edge vv'.

Find the solution of the maximum flow problem with lower bounds

Find the solution of the maximum flow problem with lower bounds on $G(V, E)$ by increasing feasible flows found in the feasible flow problem by values from the maximum flow problem with residual capacities.

Minimum value problem [2] with lower bounds on $G(V, E)$

$$
\begin{aligned}
&min \quad \alpha \\
&s.t. \quad \sum_{v':vv' \in E} x_{vv'} - \sum_{v':v'v \in E} x_{v'v} = \begin{cases} \alpha & for \quad v = s \\ 0 & \forall v \in V \setminus \{s, t\} \\ -\alpha & for \quad v = t \end{cases} \\
&\qquad l_{vv'} \leq x_{vv'} \leq u_{vv'} \qquad \forall vv' \in E
\end{aligned}
$$

Solution method for minimum value problem

First find a feasible flow. Than solve the maximum flow problem, where capacities $r^{inv}_{vv'}$ are equal to $(x_{vv'} - l_{vv'}) + (u_{v'v} - x_{v'v})$. Note that the capacity $r^{inv}_{vv'}$ denotes the maximum possible decrease in flow for the edge vv'. Finally, the solution of the minimum value problem with lower bounds on $G(V, E)$ can be found by decreasing feasible flows by values from the maximum flow problem with capacities $r^{inv}_{vv'}$.

Shortest path problem

Let $G(V, E)$ be a directed graph with costs $c_{vv'}$ defined on edges $vv' = e \in E$, where $v \neq v'$, v and $v' \in V$. Let $b_v > 0$ be the supply and $b_v < 0$ be the demand associated with each vertex $v \in V$. Moreover, $x_{vv'}$ denotes the amount of flow from a vertex v to another vertex v'. Then, shortest path problem minimizes the lengths of directed paths from a vertex s to all other vertices $t \in V$, $t \neq s$. Equivalently, shortest path problem minimizes the cost of sending an amount of unit flows from vertex s to all other vertices $t \in V$, $t \neq s$, where all flows in G are positive and satisfy the flow conservation constraints.

Shortest path problem

$$
\begin{aligned}
min \quad & \sum_{vv' \in E} c_{vv'} x_{vv'} \\
s.t. \quad & \sum_{v' : vv' \in E} x_{vv'} - \sum_{v' : v'v \in E} x_{v'v} = \begin{cases} |V - 1| & for \quad v = s \\ -1 & \forall v \in V \setminus \{s\} \end{cases} \\
& 0 \leq x_{vv'} \qquad \forall vv' \in E
\end{aligned}
$$

Special case of minimum cost flow problem

- Shortest path problem from vertex s to other vertices on $G(V, E)$
- Add $u_e = \infty$, $\forall e \in E$
- Minimum cost flow problem (with u_e) on $G(V, E) \equiv$ Shortest path problem from vertex s to other vertices on $G(V, E)$

Assumptions [2]

- $\forall e \in E$, $c_e \in \mathcal{Z}$
- $\exists$ a directed path from vertex s to any vertex t, $t \in V$, $t \neq s$
- $\nexists$ a negative cycle

Network Problems

Shortest path problem

Label-setting algorithms

- Once labels are set they are not allowed to be changed

Some graph features for label-setting algorithms [2]

- $G(V, E)$ is a directed acyclic (does not contain any directed cycle) network with possibly negative c_e's, $e \in E$
- or $G(V, E)$ is a network with $c_e \geq 0$, $e \in E$

Label-correcting algorithms

- Less restrictive problem formulations
- Less efficient than label-setting algorithms

Network Problems

Shortest path problem

Breadth-First Search

- It is a label-setting algorithm
- $\forall e \in E,\ c_e = 1$
- Runs in $\mathcal{O}(|V| + |E|)$ time [6]

Directed-acyclic graph algorithm

- It is a label-setting algorithm
- Runs in $\mathcal{O}(|V| + |E|)$ time [6]

Network Problems

Shortest path problem

Dijkstra's algorithm

- It is a label-setting algorithm
- $\forall e \in E,\ c_e \geq 0$
- Original implementation runs in $\mathcal{O}(|V|^2)$ time [2]

Network Problems

Shortest path problem

Running times of variants [2] of Dijkstra's algorithm

- Dial, $\mathcal{O}(|E| + |V|C)$
- d-Heap, $\mathcal{O}(|E|log_d(|V|))$, $d = \frac{|E|}{|V|}$
- Fibonacci heap implementation, $\mathcal{O}(|E| + |V|log(|V|))$
- Radix heap implementation, $\mathcal{O}(|E| + |V|log(|V|C))$

Network Problems

Shortest path problem

Bellman-Ford-Moore algorithm

- It is a label-correcting algorithm
- $\exists e \in E,\ c_e < 0$
- FIFO implementation runs in $\mathcal{O}(|V||E|)$ time [7]

Network Problems

Shortest path problem

Running times of label-correcting algorithms [2]

- Generic, $\mathcal{O}(min(|V|^2|E|C, |E|2^{|V|}))$
- Modified, $\mathcal{O}(min(|V||E|C, |E|2^{|V|}))$
- Modified FIFO, $\mathcal{O}(|V||E|)$
- Modified Dequeue, $\mathcal{O}(min(|V||E|C, |E|2^{|V|}))$

A shortest path simplex algorithm [8]

- Pseudo permanent labels
- Multiple pivot rule
- Runs in $\mathcal{O}(|V||E|)$ time

Network Problems

Shortest path problem

Floyd-Warshall algorithm

- It is an all-pairs (not only from one vertex s) label-correcting algorithm [2]
- Runs in $\mathcal{O}(|V|^3)$ time [2]

Johnson's algorithm

- It is an all-pairs (not only from one vertex s) label-correcting algorithm [6]
- Runs in $\mathcal{O}(|V|^2 log(|V|) + |V||E|)$ time [6]

Longest path problem

- NP-hard (non-deterministic polynomial-time)

Longest path problem

- $G(V, E)$ is a directed acyclic graph
- Let $E' = E$
- $\forall e' \in E'$, $c_{e'} = -c_e$
- Shortest path problem on $G'(V, E') \equiv$ longest path problem on $G(V, E)$

Matching

Let $G(V, E)$ be an undirected graph. A matching $G'(V', E')$ is a subgraph of G and furthermore G' satisfies the following condition: $\forall v \in G'$, $deg(v) \leq 1$.

Bipartite (cardinality) matching problem

Let $G(V, E)$ be a bipartite undirected graph. Bipartite matching problem in G looks for a matching that has the maximum cardinality.

Network Problems

Matching problem

Bipartite matching as maximum flow problem [2]

- $G(V, E)$ is a bipartite undirected graph
- V_1 and V_2 are a partition of V
- $V' = V \cup \{s, t\}$
- $E' = \{vv' : v \in V_1, v' \in V_2\} \cup \{sv : v \in V_1\} \cup \{vt : v \in V_2\}$
- $\forall e \in E'$, $u_e = 1$
- $G'(V', E')$ is a directed graph
- Bipartite matching problem on $G \equiv$ maximum flow problem on G'
- Solvable with the unit capacity flow algorithm in $\mathcal{O}(\sqrt{|V|}|E|)$ time

Network Problems

Matching problem

HopcroftKarp algorithm [9]

- Solves the bipartite matching problem
- Runs in $\mathcal{O}(|V|^{\frac{5}{2}})$ time

Bipartite weighted matching problem

Let $G(V, E)$ be a bipartite directed graph with weights c_e, $e \in E$. Moreover $\forall vv' \in E$, $v \in V_1$ and $v' \in V_2$. Bipartite weighted matching problem in G looks for a matching that has minimum weight.

Network Problems

Matching problem

Bipartite weighted matching (assignment) problem

$$\begin{aligned} min \quad & \sum_{vv' \in E} c_{vv'} x_{vv'} & \\ s.t. \quad & \sum_{v' : vv' \in E} x_{vv'} = 1 & \forall v \in V_1 \\ & \sum_{v' : v'v \in E} x_{v'v} = 1 & \forall v \in V_2 \\ & 0 \leq x_{vv'} & \forall vv' \in E \end{aligned}$$

Network Problems

Matching problem

Running times of algorithms for bipartite weighted matching problem [2]

- Successive shortest path, $\mathcal{O}(|V_1|S(|V|, |E|, C))$
- Hungarian (primal-dual), $\mathcal{O}(|V_1|S(|V|, |E|, C))$
- Relaxation, $\mathcal{O}(|V_1|S(|V|, |E|, C))$
- Cost scaling, $\mathcal{O}(|V||E|log(|V|C))$
- Modified cost scaling, $\mathcal{O}(\sqrt{|V_1|}|E|log(|V|C))$

where $S(|V|, |E|, C)$ is the running time of the shortest path problem with $c_e \geq 0, \forall e \in E$.

Karp algorithm [10]

- Solves the bipartite weighted matching problem
- Runs in $\mathcal{O}(|V||E|log(|V|))$ time

Stable marriage problem [2]

Stable marriage problem is defined on a directed bipartite graph $G(V, E)$, where $|V_1| = |V_2|$, $\forall v \in V_1$ and $\forall v' \in V_2$, $c_{vv'} \in \{1, ..., |V_1|\}$ and $c_{v'v} \in \{1, ..., |V_1|\}$. In addition, $\forall v \in V_1$, if $v' \neq v''$ than $c_{vv'} \neq c_{vv''}$. Furthermore, $\forall v \in V_2$, if $v' \neq v''$ than $c_{vv'} \neq c_{vv''}$. In other words, both $|V_1|$ men and $|V_2|$ women give distinct ranks to their potential mates. An unstable situation arises when an unmarried couple chooses each other over their current spouse.

The propose-and-reject algorithm [2]

- Solves stable marriage problem in $\mathcal{O}(|V_1|^2)$ time
- $\exists$ a stable matching for any set of rankings
- Man-optimal solution if man proposes first

Network Problems

Matching problem

Nonbipartite (cardinality) matching problem

Let $G(V, E)$ be an undirected graph. Nonbipartite matching problem in G looks for a matching that has the maximum cardinality.

Network Problems

Matching problem

Nonbipartite matching algorithm [2]

- Runs in $\mathcal{O}(|V|^3)$ time

However,

Bipartite matching algorithm [2]

- Runs in $\mathcal{O}(|V||E|)$ time
- Slower than the unit capacity flow algorithm which runs in $\mathcal{O}(\sqrt{|V|}|E|)$ time

Edmonds(Gabow) algorithm [11]

- Solves the maximum weight nonbipartite matching problem
- Runs in $\mathcal{O}(|V|^3)$ time

Network Problems

Cut

Cut

A cut is a partition ($V_1 \cup V_2 = V$ and $V_1 \cap V_2 = \emptyset$) of the vertices of a directed graph $G(V, E)$. In particular, a cut is called an $s - t$ cut if $s \in V_1$ and $t \in V_2$.

Network Problems

Capacity of an $s-t$ cut

Capacity of an $s-t$ cut [2]

The capacity of an $s-t$ cut is equal to the maximum possible amount of net flow from V_1 to V_2, where $s \in V_1$ and $t \in V_2$:

$$\sum_{vv':v \in V_1,\, v' \in V_2} u_{vv'} - \sum_{v'v:v \in V_1,\, v' \in V_2} l_{v'v}$$

Minimum $s - t$ cut [2]

A **minimum $s - t$ cut** has the minimum capacity among all possible partitions of the vertices of a directed graph $G(V, E)$ such that $s \in V_1$ and $t \in V_2$.

Generalized max-flow min-cut theorem [2]

The maximum amount of flow from s to t is equal to the capacity of the minimum $s - t$ cut.

Spanning forest

A spanning forest of an undirected graph $G(V, E)$ is an acyclic subgraph of G, denoted as $G'(V, E')$, where $|E'| < |V| - 1$.

Spanning tree

A spanning tree of an undirected graph $G(V, E)$ is a connected acyclic subgraph of G, denoted as $G'(V, E')$, where $|E'| = |V| - 1$.

Minimum spanning tree problem

Minimum spanning tree problem in an undirected graph G looks for a spanning tree that has the minimum total weight.

Network Problems

Minimum spanning tree problem

Minimum spanning tree problem [2]

$$
\begin{aligned}
min \quad & \sum_{e \in E} c_e x_e \\
s.t. \quad & \sum_{e \in E} x_e = |V| - 1 \\
& \sum_{e \in E' = \{e = vv' : v \in V' \text{ and } v' \in V'\}} x_e \leq |V'| - 1 \qquad \forall V' \subseteq V \\
& x_e \in \{0, 1\} \qquad \forall e \in E
\end{aligned}
$$

Cut optimality conditions [2]

A spanning tree in $G(V, E)$ is a minimum spanning tree denoted as $G'(V, E') \Leftrightarrow \forall e \in E'$, $\exists$ a unique cut that can be obtained by removing only edge e from $G'(V, E')$ such that $\forall (v - v') : v \in V_1,\ v' \in V_2,\ (v - v') \in E;\ c_e \leq c_{(v-v')}$.

Path optimality conditions [2]

A spanning tree in $G(V, E)$ is a minimum spanning tree denoted as $G'(V, E') \Leftrightarrow \forall e = (i - j) \in E \setminus E'$, $\exists$ a unique path connecting vertices i and j, denoted as $p(e) = (i - v_0 - v_1 - v_2 ... j)$ whose elements(edges) are in E'; than $\forall e' \in p(e)$, $c_{e'} \leq c_e$.

Running times of algorithms for minimum spanning tree problem [2]

- Kruskal (based on path optimality conditions), $\mathcal{O}(|E| + |V| log(|V|)) + Sort(|E|)$
- Prim (based on cut optimality conditions), $\mathcal{O}(|E| + |V| log(|V|))$
- Sollin (based on cut optimality conditions), $\mathcal{O}(|E| log(|V|))$

All-pairs minimax path problem [2]

The all-pairs minimax path problem wants to determine a path for each pair of vertices such that the maximum edge weights on these paths are minimized. The solution of this problem corresponds to a minimum spanning tree.

Branching

In a directed graph $G(V, E)$, a branching is a directed forest, denoted as $G'(V', E')$ where the indegree (the number of edges with v as their terminal vertex) of a vertex v, $deg^{-}(v) \leq 1$ for all $v \in V'$.

(Rooted) Arborescence

In a directed graph $G(V, E)$, an (rooted) arborescence is a directed tree, denoted as $G'(V', E')$ where all edges are directed away from the root vertex. For an arborescence, the indegree (the number of edges with v as their terminal vertex) of a vertex v, $deg^{-}(v) \leq 1$ for all $v \in V'$.

Minimum(maximum) spanning branching problem

The minimum(maximum) spanning branching problem in a directed graph $G(V, E)$ looks for a branching with minimum(maximum) total weight on the edges, denoted as $G'(V, E')$, where $|E'| \leq |V| - 1$.

Minimum(maximum) spanning arborescence problem

Given a root vertex, the minimum(maximum) spanning arborescence problem in a directed graph $G(V, E)$ looks for an arborescence with minimum(maximum) total weight on the edges, denoted as $G'(V, E')$, where $|E'| = |V| - 1$.

Reduced costs

For each edge $e = (v - v')$ in $G(V, E)$ with $v \in V$, $v' \in V$, $v \neq v'$, the reduced cost $c^{\pi}_{(v-v')} = c_{(v-v')} - \pi(v) + \pi(v')$, where $\pi(v)$ and $\pi(v')$ are the node potentials associated with nodes(vertices) v and v', respectively. Let x_e's be the arc(edge) flows, $\forall e \in E$. Then, $\sum_{e \in E} c^{\pi}_e x_e = (\sum_{e \in E} c_e x_e) - \boldsymbol{\pi}^{\top}\mathbf{b}$, where $\boldsymbol{\pi}$ and $\mathbf{b}$ are column vectors associated with the node potentials and supplies/demands, respectively [2]. Furthermore, $\sum_{e \in Cyc} c^{\pi}_e = \sum_{e \in Cyc} c_e$, where Cyc is for any directed cycle and $\boldsymbol{\pi}$ is for any node potentials [2].

Free and restricted edges(arcs) [2]

An edge e with a feasible flow in $G(V, E)$ is called restricted if its amount of flow is equal to its lower or upper bound. Otherwise, it is called a free edge.

Cycle free (feasible) solution [2]

A cycle free solution is one with no cycle consisting of only free arcs. Hence, there is at least one restricted edge in an augmenting cycle associated with a cycle free solution. It is then possible to augment flow in only a single direction due to the presence of some restricted edges.

Cycle free property [2]

The optimal solution of a minimum cost flow problem with a bounded objective function over the feasible region is cycle free.

Spanning tree solution [2]

A spanning tree solution is where a feasible solution is associated with a spanning tree, and furthermore, every non-tree edge is restricted while tree edges are allowed to be free or restricted.

Spanning tree property [2]

A spanning tree can always be constructed from the cycle free optimal solution (if required by adding some restricted edges to the spanning forest resulting from the cycle free solution) of a minimum cost flow problem with a bounded objective function over the feasible region.

Spanning tree structure (T,L,U) [2]

- T: tree edges
- L: non-tree edges at their lower bounds
- U: non-tree edges at their upper bounds

Non-degenerate spanning tree [2]

If all tree edges are free than the spanning tree is non-degenerate, otherwise it is called degenerate.

Optimality conditions [2]

$\exists\, \boldsymbol{\pi}$,

- $\forall e \in T,\ c_e^\pi = 0$
- $\forall e \in L,\ c_e^\pi \geq 0$
- $\forall e \in U,\ c_e^\pi \leq 0$

Network simplex algorithm [2]

Find an edge $e \in \{L \cup U\}$ violating the optimality conditions

- add this edge to T in order to obtain a negative cycle
- send maximum amount of flow in this cycle
- remove an edge reaching its bound after augmenting flow (in the direction of this negative cycle) from this cycle

Network Problems

Network simplex algorithm

Calculating the node potentials given (T,L,U) [2]

- set $\pi(v) = 0$ for some node $v \in V$
- given the fact that $\forall e = (v - v') \in T$, $0 = c^{\pi}_{v-v'} = c_{v-v'} - \pi(v) + \pi(v')$, calculate $\pi(v') = \pi(v) - c_{v-v'}$ iteratively

Calculating the edge flows given (T,L,U) [2]

- for an edge $e \in L$, x_e is equal to its lower bound
- for an edge $e \in U$, x_e is equal to its upper bound
- for edges $e \in T$, x_e's are calculated starting from the leaf nodes by taking into account the node supplies/demands

Strongly feasible spanning tree [2]

Given a spanning tree structure (T,L,U) hanging from the root node, this spanning tree is strongly feasible if $\forall e \in T$, if edge e has zero flow than it points towards the root node and if edge e has flow quantity at its capacity than it points away from the root node. By iterating over the adjacent strongly feasible spanning tree structures, the network simplex algorithm runs in a finite number of steps.

Some variants

- Generalized network simplex algorithm
- Minimum cost proportional flow problem with disconnected subnetworks [12]

Network Problems

A few libraries

LEMON Graph Library (C++)

Library for Efficient Modeling and Optimization in Networks

NetworkX

A Python library for graphs and networks

Compressed sparse graph routines (scipy.sparse.csgraph)

Fast graph algorithms

MIP

Mixed integer programming (MIP)

Mixed integer programming (MIP) is a mathematical framework with many applications in the optimization of industrial systems.

Modeling languages/tools

Modeling languages/tools are used to model and analyze MIP formulations. Usually these tools are not a standalone solver.

Examples for modeling languages/tools

- AMPL (algebraic modeling language)
- GAMS (general algebraic modeling system)
- LINGO (for building and solving MIP formulations)
- MiniZinc (constraint modeling language)
- CVXPY (Python-embedded modeling language for convex optimization problems)
- COIN-OR (computational infrastructure for operations research) PuLP (Python library for modeling LP formulations)
- COIN-OR Python MIP (Python tools for modeling MIP formulations)

MIP solvers

MIP solvers are used to solve MIP formulations.

Examples for MIP solvers

- IBM CPLEX
- Gurobi
- SCIP
- MOSEK
- FICO Xpress
- LINDO
- HiGHS
- Cbc (COIN-OR branch and cut)
- GLPK (GNU Linear Programming Kit)

MIP

Integration with industrial systems

Application programming interfaces (API)

- Python API
- C++ API
- Java API
- ...

MIP

Integration with industrial systems

Deployment

- AWS
- Azure
- Google Cloud
- ...

MIP

File formats

File formats for problem exchange

- LP
- MPS (fixed or free)
- ...

Example I

A machine is used to manufacture the required components of a product "z". This machine can work 23 hours a day. To produce product "z", three components "x" and two components "y" are needed. It takes 0.2 hours to produce one product "x", and 0.25 hours to produce one product "y" on this machine. It is not possible to produce "x" and "y" components at the same time. At the end of the day, the machine must be empty so that a 1-hour planned maintenance can be performed. Find the maximum amount of "z" that can be sustainably produced in a day.

LP file format for Example I

```
Maximize
objective: z
Subject To
constraint1: 2 x - 3 y = 0
constraint2: x - 3 z = 0
constraint3: 0.2 x + 0.25 y <= 23
General
x y z
End
```

MIP

Modeling languages and solvers

Use of CVXPY and SCIP for Example I

```
import cvxpy as cp
x = cp.Variable(1, integer=True)
y = cp.Variable(1, integer=True)
z = cp.Variable(1, integer=True)
objective = z
constraints = []
constraints += [2*x - 3*y == 0]
constraints += [x - 3*z == 0]
constraints += [0.2*x + 0.25*y <= 23]
problem = cp.Problem(cp.Maximize(objective), constraints)
problem.solve(solver=cp.SCIP, verbose=False)
```

Example II

In a production line, two models, namely ”x” and ”y” are produced with a profit margin 3 and 2, respectively. Each model has its own setup requirements, so the setup costs of models ”x” and ”y” are 15 and 25 respectively. The production quantity for model ”X” is either 0 or must be between 50 and 250. The production quantity for model ”Y” is either 0 or must be between 100 and 300. Total production quantity for models ”x” and ”y” should equal 300 units.

LP file format for Example II

```
Maximize
objective: 3 x + 2 y - 15 xs - 25 ys
Subject To
constraint1: x + y = 300
constraint2: 50 xs - x <= 0
constraint3: x - 250 xs <= 0
constraint4: 100 ys - y <= 0
constraint5: y - 300 ys <= 0
General
x  y
Binary
xs  ys
End
```

Use of CVXPY and SCIP for Example II

```
import cvxpy as cp
x = cp.Variable(1, integer=True)
y = cp.Variable(1, integer=True)
xs = cp.Variable(1, boolean = True)
ys = cp.Variable(1, boolean = True)
objective = 3*x + 2*y - 15*xs - 25*ys
constraints = []
constraints += [x + y == 300]
constraints += [50*xs - x <= 0]
constraints += [x - 250*xs <= 0]
constraints += [100*ys - y <= 0]
constraints += [y - 300*ys <= 0]
problem = cp.Problem(cp.Maximize(objective), constraints)
problem.solve(solver=cp.SCIP, verbose=False)
```

Traveling salesman problem

The optimal solution of the traveling salesman problem in a connected graph $G(V, E)$ is the shortest circuit visiting every vertex of $G(V, E)$ exactly once or the shortest Hamilton cycle.

Symmetric traveling salesman problem (sTSP)

When the traveling salesman problem is defined on an undirected graph $G(V, E)$, it is called symmetric traveling salesman problem.

MIP

Traveling salesman problem

A MIP formulation for symmetric traveling salesman problem (|V|>2) [13], [14]

$$
\begin{aligned}
min \quad & \sum_{vv' \in V^2,\, v<v'} c_{vv'} x_{vv'} & \\
s.t. \quad & \sum_{v': v' \in V,\, v'<v} x_{v'v} + \sum_{v': v' \in V,\, v'>v} x_{vv'} = 2 & \forall v \in V \\
& \sum_{vv': vv' \in \mathcal{S}^2,\, v<v'} x_{vv'} \leq |\mathcal{S}| - 1 & \forall \mathcal{S} \subset V:\; 3 \leq |\mathcal{S}| \leq |V| - 3 \\
& x_{vv'} \in \{0, 1\} & \forall vv' \in E:\; v < v'
\end{aligned}
$$

Asymmetric traveling salesman problem (aTSP)

When the traveling salesman problem is defined on a directed graph $G(V, E)$, it is called asymmetric traveling salesman problem.

MIP

Traveling salesman problem

A MIP formulation for asymmetric traveling salesman problem [14]

$$
\begin{aligned}
min \sum_{vv' \in V^2} & c_{vv'} x_{vv'} & \\
s.t. \sum_{v' : v' \in V} & x_{vv'} = 1 & \forall v \in V \\
\sum_{v' : v' \in V} & x_{v'v} = 1 & \forall v \in V \\
\sum_{vv' : vv' \in \mathcal{S}^2} & x_{vv'} \leq |\mathcal{S}| - 1 & \forall \mathcal{S} \subset V : \ 2 \leq |\mathcal{S}| \leq |V| - 2 \\
& x_{vv'} \in \{0, 1\} & \forall vv' \in E
\end{aligned}
$$

The Dantzig, Fulkerson and Johnson formulation of subtour elimination constraints for aTSP [13], [15]

$$\sum_{vv':vv' \in \mathcal{S}^2} x_{vv'} \leq |\mathcal{S}| - 1 \qquad \forall \mathcal{S} \subset V \setminus \{1\} : 2 \leq |\mathcal{S}| \leq |V| - 1$$

MIP

Traveling salesman problem

The Miller, Tucker and Zemlin formulation of subtour elimination constraints for aTSP [15]

$$
\begin{aligned}
& u_v - u_{v'} + (|V| - 1)x_{vv'} \leq |V| - 2 && \forall v \in V, \forall v' \in V \setminus \{1\}, v \neq v' \\
& 1 \leq u_v \leq |V| - 1 && \forall v \in V \setminus \{1\}
\end{aligned}
$$

where u_v's are auxiliary variables used to define the visiting order of vertex v

MIP

Traveling salesman problem

The Desrochers and Laporte formulation of subtour elimination constraints for aTSP [15]

$$
\begin{aligned}
& u_v - u_{v'} + (|V| - 1)x_{vv'} + (|V| - 3)x_{v'v} \leq |V| - 2 && \forall v \in V, \forall v' \in V \setminus \{1\}, \\
& && v \neq v' \\
& 1 + (|V| - 3)x_{v1} + \sum_{v' \in V \setminus \{1\}} x_{v'v} \leq u_v && \forall v \in V \setminus \{1\} \\
& u_v \leq |V| - 1 - (|V| - 3)x_{1v} - \sum_{v' \in V \setminus \{1\}} x_{vv'} && \forall v \in V \setminus \{1\}
\end{aligned}
$$

MIP

Traveling salesman problem

The Gavish and Graves (single commodity flow) formulation of subtour elimination constraints for aTSP [15]

$$\sum_{v':v'\in V} g_{v'v} - \sum_{v':v'\in V\setminus\{1\}} g_{vv'} = 1 \qquad \forall v \in V\setminus\{1\}$$

$$0 \leq g_{vv'} \leq (|V|-1)x_{vv'} \qquad \forall v \in V, \forall v' \in V\setminus\{1\}$$

where $g_{vv'}$'s are auxiliary variables that indicate the number of edges remaining from vertex v' to the vertex 1 in order to complete the optimal tour.

Multiple traveling salesman problem (mTSP)

When the traveling salesman problem has more than one salesman, it is called multiple traveling salesman problem.

MIP

Traveling salesman problem

A MIP formulation for mTSP with *m* salesmen [16]

$$
\begin{aligned}
min \sum_{vv' \in V^2} & c_{vv'} x_{vv'} \\
s.t. \sum_{v':v' \in V \setminus \{1\}} & x_{1v'} = m \\
\sum_{v':v' \in V \setminus \{1\}} & x_{v'1} = m \\
\sum_{v':v' \in V} & x_{vv'} = 1 & \forall v \in V \setminus \{1\} \\
\sum_{v':v' \in V} & x_{v'v} = 1 & \forall v \in V \setminus \{1\} \\
& \ldots
\end{aligned}
$$

MIP

Traveling salesman problem

A MIP formulation for mTSP with m salesmen [16]

$$\sum_{vv':vv' \in \mathcal{S}^2} x_{vv'} \leq |\mathcal{S}| - 1 \qquad \forall \mathcal{S} \subseteq V \setminus \{1\}, \mathcal{S} \neq \emptyset$$

$$x_{vv'} \in \{0, 1\} \qquad \forall vv' \in E$$

The Miller, Tucker and Zemlin formulation of subtour elimination constraints for mTSP [16]

$$u_v - u_{v'} + \mu x_{vv'} \leq \mu - 1 \qquad \forall v \in V, \forall v' \in V \setminus \{1\},\ v \neq v'$$
$$1 \leq u_v \leq \mu \qquad \forall v \in V \setminus \{1\}$$

where u_v's are auxiliary variables used to define the visiting order of vertex v and μ is equal to the maximum number of vertices that can be visited by any salesman

Vehicle routing problem

The vehicle routing problem differs from mTSP in that it takes vehicle capacities into account.

Capaciated vehicle routing problem (CVRP)

Let $G(V, E)$ be a graph. The capacitated vehicle routing problem is defined on the graph G, where a fleet of identical vehicles with indices $m \in M$ and capacity Q based at one central depot deliver goods to meet the demand of customers at different vertices, namely d_v, $\forall v \in V$. If $\forall v, v' \in V$, $c_{vv'} = c_{v'v}$ then CVRP is symmetric (sCVRP), else it is asymmetric (aCVRP). For sCVRP, G is an undirected graph while aCVRP requires to be defined on a directed graph G.

MIP

Vehicle routing problem

The two-index vehicle flow formulation for sCVRP [17], [18]

$$
\begin{aligned}
& min \sum_{vv' \in V^2,\, v<v'} c_{vv'} x_{vv'} \\
& s.t. \sum_{v': v' \in V,\, v'<v} x_{v'v} + \sum_{v': v' \in V,\, v'>v} x_{vv'} = 2 && \forall v \in V \setminus \{1\} \\
& \sum_{v \in V \setminus \{1\}} x_{1v} = 2|M| \\
& \sum_{vv': vv' \in \mathcal{S} \times V \setminus \mathcal{S} \cup V \setminus \mathcal{S} \times \mathcal{S},\, v<v'} x_{vv'} \geq 2 \left\lceil \frac{\sum_{v \in \mathcal{S}} d_v}{Q} \right\rceil && \forall \mathcal{S} \subseteq V \setminus \{1\}, |\mathcal{S}| \geq 2 \\
& x_{vv'} \in \{0,1\} && \forall vv' \in E : v < v', v \neq 1 \\
& x_{1v} \in \{0,1,2\} && \forall 1v \in E : 1 < v
\end{aligned}
$$

MIP

Vehicle routing problem

The three-index vehicle flow formulation for sCVRP [18], [19]

$$min \sum_{vv' \in V^2,\, v<v'} c_{vv'} \sum_{m \in M} x^m_{vv'}$$

$$s.t. \sum_{v':v' \in V,\, v'<v} x^m_{v'v} + \sum_{v':v' \in V,\, v'>v} x^m_{vv'} = 2y^m_v \qquad \forall v \in V \setminus \{1\}, \forall m \in M$$

$$\sum_{vv':vv' \in \mathcal{S} \times V \setminus \mathcal{S} \cup V \setminus \mathcal{S} \times \mathcal{S}, v<v'} x^m_{vv'} \geq 2y^m_{v''} \qquad \forall \mathcal{S} \subseteq V \setminus \{1\}, |\mathcal{S}| \geq 2, \ldots$$

$$\forall v'' \in \mathcal{S}, \forall m \in M$$

$$\ldots$$

The three-index vehicle flow formulation for sCVRP [18], [19]

$$
\begin{aligned}
&\sum_{m \in M} y_1^m = |M| && \\
&\sum_{m \in M} y_v^m = 1 && \forall v \in V \setminus \{1\} \\
&\sum_{v \in V \setminus \{1\}} d_v y_v^m \leq Q && \forall m \in M \\
&x_{vv'}^m \in \{0,1\} && \forall vv' \in E : v < v', v \neq 1, \forall m \in M \\
&x_{1v}^m \in \{0,1,2\} && \forall 1v \in E : 1 < v, \forall m \in M \\
&y_v^m \in \{0,1\} && \forall v \in V, \forall m \in M
\end{aligned}
$$

where y_v^m indicates whether vertex v is visited by vehicle m, and $x_{vv'}^m$ shows whether edge vv' is used by vehicle m.

The three-index vehicle flow formulation for aCVRP [20]

Let $G(V,E)$ be a directed graph. We can extend G by adding another depot that is just a copy of the original to be used for returns. Let $\overline{V} = V \cup \{|V|+1\}$ and $\overline{E} = E \cup_{v \in V, v' = |V|+1} \{vv'\}$. Moreover, let $c_{vv'} = c_{1v}, \forall v \in V \setminus \{1\}$ and $c_{1v'} = 0$, where $v' = |V|+1$. Then, $\overline{G}(\overline{V},\overline{E})$ is the extended graph of G.

MIP

Vehicle routing problem

The three-index vehicle flow formulation for aCVRP [20]

$$
\begin{aligned}
min \sum_{vv' \in \overline{E}} c_{vv'} \sum_{m \in M} x^m_{vv'} & \\
s.t. \sum_{vv' \in \overline{E}, m \in M} x^m_{vv'} = 1 & \quad \forall v \in V \setminus \{1\} \\
\sum_{v'v \in \overline{E}} x^m_{v'v} - \sum_{vv' \in \overline{E}} x^m_{vv'} = 0 & \quad \forall v \in V \setminus \{1\}, \forall m \in M \\
\sum_{1v \in \overline{E}} x^m_{1v} = 1 & \quad \forall m \in M \\
\sum_{vv' \in \overline{E}} x^m_{vv'} = 1 & \quad \forall m \in M, v' = |V| + 1 \\
\ldots &
\end{aligned}
$$

MIP

Vehicle routing problem

The three-index vehicle flow formulation for aCVRP [20]

$$\sum_{v \in V \setminus \{1\}} (d_v \sum_{vv' \in \overline{E}} x^m_{vv'}) \leq Q \quad \forall m \in M$$

$$x^m_{vv'} \in \{0, 1\} \quad \forall vv' \in \overline{E}, \forall m \in M$$

where $x^m_{vv'}$ shows whether edge vv' is used by vehicle m.

The Miller, Tucker and Zemlin formulation of subtour elimination constraints for three-index aCVRP

$$
\begin{aligned}
& u_v^m - u_{v'}^m + (|\overline{V}| - 1)x_{vv'}^m \leq |\overline{V}| - 2 && \forall v \in \overline{V}, \forall v' \in \overline{V} \setminus \{1\}, \\
& && v \neq v', \forall m \in M \\
& 1 \leq u_v^m \leq |\overline{V}| - 1 && \forall v \in \overline{V} \setminus \{1\}, \forall m \in M
\end{aligned}
$$

where u_v^m's are auxiliary variables used to determine the visiting order of vertex v by vehicle m

MIP

Vehicle routing problem

The time windows and subtour elimination constraints for three-index aCVRP [20]

$$
\begin{aligned}
& u_v^m - u_{v'}^m + t_{vv'} \leq (1 - x_{vv'}^m)\, L && \forall v \in \overline{V}, \forall v' \in \overline{V} \setminus \{1\}, \\
& && v \neq v', \forall m \in M \\
& s_v \leq u_v^m \leq f_v && \forall v \in \overline{V} \setminus \{1\}, \forall m \in M
\end{aligned}
$$

where u_v^m's are auxiliary variables used to define the visiting time of vertex v by vehicle m. For the vertex v, s_v and f_v denote the start and end of time window, respectively. $t_{vv'}$ is the time required to move from vertex v to vertex v'. Moreover,

$$L = \max_{v \in \overline{V}} f_v - \min_{v \in \overline{V} \setminus \{1\}} s_v$$

The two-commodity flow formulation for sCVRP [18], [21]

Let $G(V, E)$ be an undirected graph. We can extend G by adding another depot that is just a copy of the original to be used for returns. Let $\overline{V} = V \cup \{|V| + 1\}$ and $\overline{E} = E \cup_{v \in V \setminus \{1\}, v' = |V|+1} \{vv'\}$. Moreover, let $c_{vv'} = c_{1v}$, $\forall v \in V \setminus \{1\}$ where $v' = |V| + 1$. Then, $\overline{G}(\overline{V}, \overline{E})$ is the extended graph of G.

MIP

Vehicle routing problem

The two-commodity flow formulation for sCVRP [18], [21]

$$
\begin{aligned}
& min \sum_{vv' \in \overline{E}} c_{vv'} x_{vv'} \\
& s.t. \sum_{v' \in \overline{V}} (y_{v'v} - y_{vv'}) = 2d_v && \forall v \in V \setminus \{1\} \\
& \sum_{v' \in V \setminus \{1\}} y_{1v'} = \sum_{v \in V \setminus \{1\}} d_v \\
& \sum_{v' \in V \setminus \{1\}} y_{v'1} = |M|Q - \sum_{v \in V \setminus \{1\}} d_v \\
& \sum_{v' \in V \setminus \{1\}} y_{vv'} = |M|Q && v = |V| + 1 \\
& \ldots
\end{aligned}
$$

The two-commodity flow formulation for sCVRP [18], [21]

$$
\begin{aligned}
& y_{vv'} + y_{v'v} = Qx_{vv'} && \forall vv' \in \overline{E} \\
& \sum_{vv' \in \overline{E}} x_{vv'} = 2 && \forall v \in V \setminus \{1\} \\
& y_{vv'} \geq 0 && \forall vv' \in \overline{E} \\
& y_{v'v} \geq 0 && \forall vv' \in \overline{E} \\
& x_{vv'} \in \{0, 1\} && \forall vv' \in \overline{E}
\end{aligned}
$$

where $y_{vv'}$ denotes the flow of one commodity from v to v' while $y_{v'v}$ indicates the flow of other commodity from v' to v, and $x_{vv'}$ shows whether undirected edge vv' is used by any vehicle.

MIP

Vehicle routing problem

The set partitioning formulation for CVRP [18], [22]

$$
\begin{aligned}
min \sum_{r \in R} c_r x_r & \\
s.t. \sum_{r \in R} a_{vr} x_r = 1 & \quad \forall v \in V \setminus \{1\} \\
\sum_{r \in R} x_r = |M| & \\
x_r \in \{0, 1\} & \quad \forall r \in R
\end{aligned}
$$

where x_r indicates whether the route r is used by any vehicle, and a_{vr} is a parameter that is equal to 1 if the vertex v is included in the route r. The cost of using route r is equal to c_r.

The set covering formulation for CVRP [18], [23]

$$
\begin{aligned}
min \sum_{r \in R} c_r x_r & \\
s.t. \sum_{r \in R} a_{vr} x_r \geq 1 & \quad \forall v \in V \setminus \{1\} \\
\sum_{r \in R} x_r = |M| & \\
x_r \in \{0,1\} & \quad \forall r \in R
\end{aligned}
$$

The set covering formulation can also be used if $\forall vv' \in E$, $c_{vv'}$ satisfies the triangle inequality [18].

Order batching problem (OBP) [24]

In OBP, customer orders are first grouped into batches and then collected by pickers following a specific routing policy.

MIP

Vehicle routing problem

Some routing policies for OBP [24]

- Traversal
- Return
- Midpoint
- Largest gap
- Composite
- Mixed
- Optimum

References I

[1] K. Rosen, *Discrete Mathematics and Its Applications*. McGraw-Hill, 2007.

[2] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[3] P. Kovács, "Minimum-cost flow algorithms: An experimental evaluation," *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015. DOI: https://doi.org/10.1080/10556788.2014.895828.

[4] P. Herrmann, A. Meyer, S. Ruzika, L. E. Schäfer, and F. von der Warth, "A machine learning based algorithm selection method to solve the minimum cost flow problem,", 2022. arXiv: 2210.02195 [cs.LG].

References II

[5] L. Chen, R. Kyng, Y. P. Liu, R. Peng, M. P. Gutenberg, and S. Sachdeva, "Maximum flow and minimum-cost flow in almost-linear time," in *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, 2022, pp. 612–623. DOI: https://doi.org/10.1109/FOCS54457.2022.00064.

[6] M. Mahmoudi and A. Boloori, "Networks," in *In Graph Theory for Operations Research and Management: Applications in Industrial Engineering*. 2013, pp. 150–178. DOI: https://doi.org/10.4018/978-1-4666-2661-4.ch012.

[7] A. Sedeño-Noda and C. González-Martín, "An efficient label setting/correcting shortest path algorithm," *Computational Optimization and Applications*, vol. 51, pp. 437–455, 2012. DOI: https://doi.org/10.1007/s10589-010-9323-9.

References III

[8] A. Sedeño-Noda and C. González-Martín, "New efficient shortest path simplex algorithm: Pseudo permanent labels instead of permanent labels," *Computational Optimization and Applications*, vol. 43, pp. 437–448, 2009. DOI: https://doi.org/10.1007/s10589-007-9144-7.

[9] J. E. Hopcroft and R. M. Karp, "A $n^{\frac{5}{2}}$ algorithm for maximum matchings in bipartite graphs," in *12th Annual Symposium on Switching and Automata Theory (swat 1971)*, 1971, pp. 122–125. DOI: https://doi.org/10.1109/SWAT.1971.1.

[10] R. Karp, "An algorithm to solve the mxn assignment problem in expected time o (mn log n)," EECS Department, University of California, Berkeley, Tech. Rep. UCB/ERL M78/67, 1978.

[11] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," *ACM Comput. Surv.*, vol. 18, no. 1, pp. 23–38, 1986. DOI: https://doi.org/10.1145/6462.6502.

References IV

[12] U. Bahçeci and O. Feyzioğlu, "A network simplex based algorithm for the minimum cost proportional flow problem with disconnected subnetworks," *OPTIMIZATION LETTERS*, pp. 1173–1184, 2012. DOI: https://doi.org/10.1007/s11590-011-0356-5.

[13] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410, 1954. DOI: https://doi.org/10.1287/opre.2.4.393.

[14] G Laporte, "A concise guide to the traveling salesman problem," *Journal of the Operational Research Society*, vol. 61, no. 1, pp. 35–40, 2010. DOI: https://doi.org/10.1057/jors.2009.76.

[15] T. Öncan, . K. Altınel, and G. Laporte, "A comparative analysis of several asymmetric traveling salesman problem formulations," *Computers & Operations Research*, vol. 36, no. 3, pp. 637–654, 2009. DOI: https://doi.org/10.1016/j.cor.2007.11.008.

References V

[16] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006. DOI: https://doi.org/10.1016/j.omega.2004.10.004.

[17] G. Laporte, Y. Nobert, and M. Desrochers, "Optimal routing under capacity and distance restrictions," *Operations Research*, vol. 33, no. 5, pp. 1050–1073, 1985.

[18] R. Baldacci, P. Toth, and D. Vigo, "Exact algorithms for routing problems under vehicle capacity constraints," *Annals of Operations Research*, vol. 175, no. 1, pp. 213–245, 2010. DOI: https://doi.org/10.1007/s10479-009-0650-0.

[19] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977. DOI: https://doi.org/10.1002/net.3230070203.

References VI

[20] J. Larsen, "Parallelization of the vehicle routing problem with time windows," Ph.D. dissertation, 1999.

[21] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

[22] M. L. Balinski and R. E Quandt, "On an integer program for a delivery problem," *Operations Research*, vol. 12, no. 2, pp. 300–304, 1964.

[23] J. Bramel and D. Simchi-Levi, "4. set-covering-based algorithms for the capacitated vrp," in *The Vehicle Routing Problem*, pp. 85–108. DOI: https://doi.org/10.1137/1.9780898718515.ch4.

References VII

[24] U. Bahçeci and T. Öncan, "An evaluation of several combinations of routing and storage location assignment policies for the order batching problem," *International Journal of Production Research*, vol. 60, no. 19, pp. 5892–5911, 2022. DOI: https://doi.org/10.1080/00207543.2021.1973684.