FORMAL MODELLING

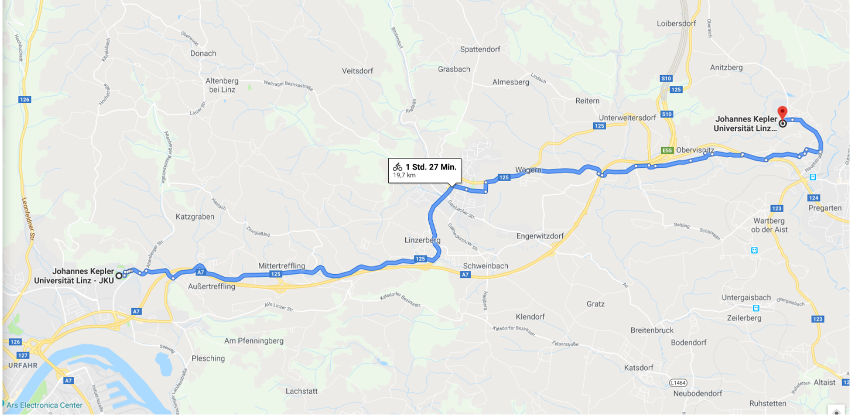
Modelling Problems in Geometry and Discrete Mathematics

Wolfgang Windsteiger
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
Wolfgang.Windsteiger@risc.jku.at

JOHANNES KEPLER
UNIVERSITY LINZ

RESEARCH INSTITUTE FOR
SYMBOLIC COMPUTATION

THE SHORTEST PATH PROBLEM



BASIC CONCEPTS

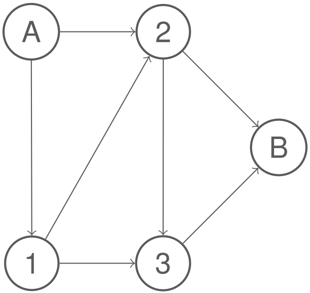
Graphs $\rightsquigarrow$ appropriate mathematical model for a “network of streets”.

Basic entities: vertices with connections between them, called edges or arcs.

Different aspects:

- Connections can be oriented?
- Can there be more than one edge between to vertices?
- Must vertices connected by an edge be distinct?
- Can edges have values associated?
- Can vertices have values associated?
- etc. etc.

TYPICAL EXAMPLE



BASIC DEFINITIONS

Definition (Undirected Simple Graph, Directed Simple Graph)

Let V be a set. The pair $G = (V, E)$ is called an undirected simple graph iff

$$E \subseteq P(V) \text{ and } \underset{a \in E}{\forall} |a| = 2.$$

The pair (V, E) is called a directed simple graph iff

$$E \subseteq V^2 \text{ and } \underset{a \in E}{\forall} a_1 \neq a_2.$$

We call G a simple graph if and only if it is an undirected or a directed graph. If G is a graph, then $V(G) := G_1$ and $E(G) := G_2$ are called the vertices and edges of G, respectively.

MORE THAN ONE EDGE BETWEEN TWO VERTICES

E can be defined as a multiset $E = (A, m)$, where

$$A = \{a \in P(V) \mid |a| = 2\} \quad \text{or} \quad A = \{a \in V^2 \mid a_1 \neq a_2\}$$

for undirected graphs or directed graphs, respectively, and

$$m\colon A \longrightarrow \mathbb{N}_0.$$

$A \rightsquigarrow$ the set of potential edges.

$m \rightsquigarrow$ the multiplicity of each edge, including the case $m(a) = 0$ meaning that the graph does not contain the edge a.

For a multiset $E = (A, m)$ we write $a \in E$ if and only if $a \in A$ and $m(a) \geq 1$.

DISTINGUISHABLE EDGES

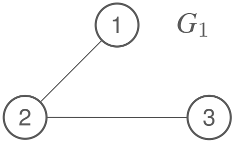
E can be defined as $E = (X, e)$, where X is a set and

$$e\colon X \longrightarrow A \qquad \text{with } A \text{ as above.}$$

$X \rightsquigarrow$ the names for the edges.

For $E = (X, e)$ we write $a \in E$ if and only if $\underset{x \in X}{\exists}\ e(x) = a$.

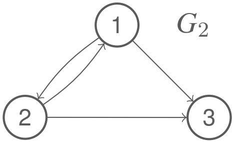
EXAMPLES



An undirected simple graph

$$G_1 = (\{1, 2, 3\}, \{\{1, 2\}, \{2, 3\}\}),$$

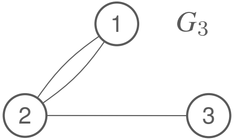
EXAMPLES



A directed simple graph

$$G_2 = (\{1, 2, 3\}, \{(1, 2), (2, 1), (1, 3), (2, 3)\}).$$

EXAMPLES



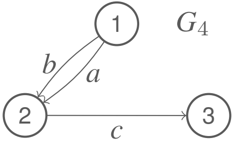
Not an undirected simple graph: contains a double edge between 1 and 2. Can be defined as

$$G_3 = (\{1,2,3\}, (\{\{1,2\},\{1,3\},\{2,3\}\}, m)) \quad \text{with } m \text{ defined by}$$

a	$m(a)$
$\{1,2\}$	2
$\{1,3\}$	0
$\{2,3\}$	1

Note that one cannot distinguish the two edges between 1 and 2.

EXAMPLES



Not a simple directed graph: contains two distinct edges a and b from 1 to 2. Can be defined as

$G_4 = (\{1, 2, 3\}, (\{a, b, c\}, e))$ with e defined by

x	$e(x)$
a	$(1, 2)$
b	$(1, 2)$
c	$(2, 3)$

LOOPS

By definition, a simple graph cannot contain loops, i.e. edges connecting a vertex with itself.

"G is a graph" $\rightsquigarrow$ G is a graph in one of the representations mentioned above.

- G undirected: uv as an abbreviation for an edge $\{u, v\} \in E(G)$.
- G directed: uv as an abbreviation for an edge $(u, v) \in E(G)$.

NEIGHBOUR, NEIGHBOURHOOD

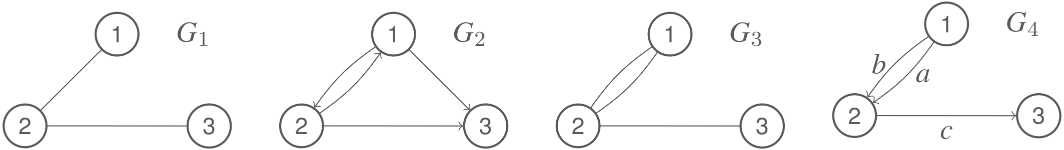
Definition

Let G be a graph. The vertex v is a neighbour of vertex u if and only if $uv \in E(G)$. Furthermore, we call

$$N_G(u) := \{v \in V(G) \mid v \text{ is a neighbour of } u\}$$

the neighbourhood of u (in G).

EXAMPLES



Example

$$N_{G_1}(1) = \{2\} \qquad N_{G_1}(2) = \{1,3\} \qquad N_{G_1}(3) = \{2\}$$
$$N_{G_2}(1) = \{2,3\} \qquad N_{G_2}(2) = \{1,3\} \qquad N_{G_2}(3) = \{\}$$
$$N_{G_3}(1) = \{2\} \qquad N_{G_3}(2) = \{1,3\} \qquad N_{G_3}(3) = \{2\}$$
$$N_{G_4}(1) = \{2\} \qquad N_{G_4}(2) = \{3\} \qquad N_{G_4}(3) = \{\}$$

WALK

Definition

Let G be a graph, $n \geq 1$, and $a, b \in V(G)$. A finite sequence

$$w\colon \mathbb{N}_{1,2n+1} \longrightarrow V(G) \cup E(G)$$

is called a walk of length n from a to b in G if and only if

$$\underset{0 \leq i \leq n}{\forall} w_{2i+1} \in V(G) \qquad \underset{1 \leq i \leq n}{\forall} w_{2i} \in E(G)$$

$$w_1 = a, w_{2n+1} = b \qquad \underset{1 \leq i \leq n}{\forall} w_{2i} = w_{2i-1}w_{2i+1}.$$

Sequence of vertices of w and the sequence of edges of w:

$$V(w)\colon \mathbb{N}_{1,n+1} \longrightarrow V(G), i \mapsto w_{2i-1} \qquad E(w)\colon \mathbb{N}_{1,n} \longrightarrow E(G), i \mapsto w_{2i}.$$

TRAIL, PATH

Definition

Let G be a graph, $n \geq 1$, and $a, b \in V(G)$.

1. A finite sequence
$$t\colon \mathbb{N}_{1,2n+1} \longrightarrow V(G) \cup E(G)$$
is called a trail of length n from a to b in G if and only if t is a walk of length n from a to b in G and $E(t)$ is injective (from $\mathbb{N}_{1,n}$ to $E(G)$).
2. A finite sequence
$$p\colon \mathbb{N}_{1,2n+1} \longrightarrow V(G) \cup E(G)$$
is called a path of length n from a to b in G if and only if p is a trail of length n from a to b in G and $V(p)$ is injective (from $\mathbb{N}_{1,n+1}$ to $V(G)$).

EXAMPLE

$w = (2, (2, 1), 1, (1, 2), 2, (2, 1), 1, (1, 3), 3)$ is a walk of length 4 from 2 to 3 in G_2.

$$V(w) = (2, 1, 2, 1, 3) \qquad E(w) = ((2, 1), (1, 2), (2, 1), (1, 3)),$$

w is neither a trail nor a path in G_2.

$t = (2, (2, 1), 1, (1, 2), 2, (2, 3), 3)$ is a trail of length 3 from 2 to 3 in G_2.

$$V(t) = (2, 1, 2, 3) \qquad E(t) = ((2, 1), (1, 2), (2, 3)),$$

t is not a path in G_2.

$p = (2, (2, 1), 1, (1, 3), 3)$ is a path of length 2 from 2 to 3 in G_2.

$$V(p) = (2, 1, 3) \qquad E(p) = ((2, 1), (1, 3)).$$

WEIGHTED GRAPH

Definition

The triple $G = (V, E, c)$ is called a weighted graph if and only if (V, E) is a graph and $c \colon E \to \mathbb{R}$. We call c the cost function of G and $c(e)$ the costs of an edge e.

All special properties of the graph (V, E), e.g. being simple, directed, undirected, translate directly to its weighted variant (V, E, c). A sequence is a walk/trail/path in (V, E, c) if and only if it is a walk/trail/path in (V, E).

EXTENDED COST FUNCTION, DISTANCE

Definition

Let $G = (V, E, c)$ be a weighted graph and $F \subseteq E$. Then

$$c(F) := \sum_{e \in F} c(e).$$

Let w be a walk of length n from a to b in G, then

$$c(w) := \sum_{e \in E(w)} c(e).$$

The distance from a to b in G is

$$\text{dist}_G(a, b) := \min(\{c(w) \mid w \text{ is a walk from } a \text{ to } b \text{ in } G\}).$$

ROUTING PROBLEM IN GRAPH THEORY LANGUAGE

Given a network of streets, we define the vertices

$$V := \{C \mid \text{there are streets } s \text{ and } t \text{ crossing at } C\}.$$

Street segment: characterized by its endpoints, i.e. two crossings c_1 and c_2 on the same street such that no other crossing lies between c_1 and c_2.

Street segment between c_1 and c_2 in both directions: $\rightsquigarrow$ undirected edge $\{c_1, c_2\}$.

If it can only be used in one direction: $\rightsquigarrow$ directed edge (c_1, c_2).

$$E := \{(c_1, c_2) \in V^2 \mid \text{there is a street segment from } c_1 \text{ to } c_2\}.$$

ROUTING PROBLEM IN GRAPH THEORY LANGUAGE

Every street segment (c_1, c_2) has (non-negative) costs associated, i.e.

$$c : E \longrightarrow \mathbb{R}_0^+, (x, y) \mapsto \text{“costs” for going from } x \text{ to } y$$

Problem (Shortest Path Problem)

Given: *The graph $G = (V, E, c)$ with appropriate cost function c and $A, B \in V$.*

Find: *$d = \mathit{dist}_G(A, B)$ and p such that p is a path from A to B in G and $c(p) = d$.*

BASIC CONSIDERATIONS

1. Shortest connection is always a path. Assume it was a walk

$$w = (A, e_1, \ldots, e_k, x, \ldots, x, e_l, \ldots, e_n, B)$$

from A to B in G containing vertex x twice. Then take

$$\bar{w} = (A, e_1, \ldots, e_k, x, e_l, \ldots, e_n, B).$$

$\bar{w}$ is also a walk from A to B in G with $c(\bar{w}) \leq c(w)$.

BASIC CONSIDERATIONS

2. We use a simple graph, i.e. a graph without multiple edges and loops. Consider a and b being both edges from x to y and $c(a) \leq c(b)$ and assume a shortest path

$$p = (A, e_1, \ldots, x, b, y, \ldots, e_n, B)$$

from A to B in G. Then take

$$\bar{p} = (A, e_1, \ldots, x, a, y, \ldots, e_n, B).$$

$\bar{p}$ also a path from A to B in G with $c(\bar{p}) \leq c(p)$, such that we can always construct a path with costs at most $c(p)$ avoiding edge b.

BASIC CONSIDERATIONS

3. Loops: Consider a being a loop from x to x and assume a shortest trail

$$p = (A, e_1, \ldots, e_k, x, a, x, e_l, \ldots, e_n, B)$$

from A to B in G. Then take

$$\bar{p} = (A, e_1, \ldots, e_k, x, e_l, \ldots, e_n, B).$$

$\bar{p}$ also a trail from A to B in G with $c(\bar{p}) \leq c(p)$, such that we can always construct a path with costs at most $c(p)$ avoiding loop a.

DIJKSTRA'S ALGORITHM

Solves a slightly more general problem: it computes $\mathrm{dist}_G(A, v)$ for all $v \in V \setminus \{A\}$ and it allows to reconstruct shortest paths from A to v for all $v \in V \setminus \{A\}$.

Basic idea of the algorithm: maintain two sets of vertices C and $O = V \setminus C$, where

- C contains the closed vertices v, for which $\mathrm{dist}_G(A, v)$ is already known and
- O are the remaining open vertices v, for which only a tentative distance $l(v)$ from A is known. In fact, $l(v)$ is the shortest distance from A on a path containing only vertices in C except the final vertex v.

```
C = ∅, O = V, l(A) = 0
for v ∈ V \ {A} do
  l(v) = ∞
end
while O ≠ ∅ do
  v = such an o ∈ O with l(o) ≤ l(x) for all x ∈ O
  C = C ∪ {v}, O = O \ {v}
  dist_G(A, v) = l(v)
  for w ∈ N_G(v) do
    if l(w) > dist_G(A, v) + c(vw) then
      l(w) = dist_G(A, v) + c(vw)
      pre_G(w) = v
    end
  end
end
```

CORRECTNESS OF THE ALGORITHM

The algorithm maintains a loop invariant, namely

$$\forall_{x \in C, u \in N_G(x)} l(u) \leq \mathrm{dist}_G(A, x) + c(xu) \tag{1}$$

$$\forall_{x \in C} l(x) = \mathrm{dist}_G(A, x) \tag{2}$$

$$\forall_{o \in O} l(o) \geq \mathrm{dist}_G(A, o). \tag{3}$$

INITIALIZATION

Before the algorithm enters the while-loop for the first time, (1) and (2) clearly hold due to $C = \emptyset$, and (3) holds because of $O = V$ and $l(A) = 0 = \text{dist}_G(A, A)$ and $l(o) = \infty \geq \text{dist}_G(A, o)$ for all $o \neq A$.

IN THE LOOP ...

Now assume (1), (2), and (3) hold at the beginning of one pass through the loop, we will show that (1), (2), and (3) then also hold at the end of that pass, i.e. we have to show

$$\forall_{x \in C \cup \{v\}, u \in N_G(x)} l(u) \leq \mathrm{dist}_G(A, x) + c(xu) \tag{4}$$

$$\forall_{x \in C \cup \{v\}} l(x) = \mathrm{dist}_G(A, x) \tag{5}$$

$$\forall_{o \in O \setminus \{v\}} l(o) \geq \mathrm{dist}_G(A, o). \tag{6}$$

PROOF PART I

Let $x \in C \cup \{v\}$ and $u \in N_G(x)$. In case $x \in C$, $l(u) \leq \text{dist}_G(A, x) + c(xu)$ is true by assumption (1). Now let $x = v$. By the algorithm, we have $l(w) \leq \text{dist}_G(A, v) + c(vw)$ for all $w \in N_G(v)$ after finishing the for-loop, hence $l(u) \leq \text{dist}_G(A, x) + c(xu)$.

PROOF PART II

Let $x \in C \cup \{v\}$. In case $x \in C$, $l(x) = \text{dist}_G(A, x)$ is true by assumption (2). Now let $x = v$. First of all we show $l(x) \leq \text{dist}_G(A, x)$ by contradiction, hence, we assume $l(x) > \text{dist}_G(A, x)$, i.e. there must be a path P from A to x in G with $c(P) = \text{dist}_G(A, x) < l(x)$. P contains at least one vertex in O since $x = v \in O$, hence, there is a minimal i such that $p_i := V(P)_i \in O$. We then have $l(p_i) \leq \text{dist}_G(A, p_i)$ because there are two cases:

PROOF PART II

case $i = 1$: then $p_1 = A$ and $l(p_1) = l(A) = 0 = \text{dist}_G(A, A) = \text{dist}_G(A, p_1)$ and

case $i > 1$: from the minimality of i we get $p_{i-1} := V(P)_{i-1} \in C$ and clearly $p_i \in N_G(p_{i-1})$, hence

$$l(p_i) \overset{(1)}{\leq} \text{dist}_G(A, p_{i-1}) + c(p_{i-1}p_i) = c(P_{1:2i-1}) = \text{dist}_G(A, p_i).$$

Note that the last equality holds, because P is a path with lowest costs from A to x. Therefore, any subpath of P from A to b must be one with lowest costs to b, because otherwise P would not have lowest costs to x.

PROOF PART II

Finally, we have the contradiction

$$l(x) = l(v) \overset{\text{choice of } v}{\leq} l(p_i) \leq \text{dist}_G(A, p_i) \overset{\text{non-negative weights}}{\leq} \text{dist}_G(A, x) < l(x).$$

Thus, $l(x) \leq \text{dist}_G(A, x)$ and together with $l(x) = l(v) \geq \text{dist}_G(A, v) = \text{dist}_G(A, x)$ by assumption (3) since $v \in O$ we have $l(x) = \text{dist}_G(A, x)$.

PROOF PART III

Let $o \in O \setminus \{v\}$. In case $l(o) = \infty$ then $l(o) = \infty \geq \text{dist}_G(A, o)$ is trivial. Otherwise $l(o)$ reflects the costs of a concrete path from A to o, hence $l(o) \geq \text{dist}_G(A, o)$, by definition of $\text{dist}_G(A, o)$.

IMPLEMENTATION ISSUES

1. The algorithm computes the shortest distances from A to all v in G. If one is only interested in the shortest distances from A to B then the while-loop can be terminated as soon as $v = B$ has been chosen and $\mathrm{dist}_G(A, B)$ has been set.
2. For a real implementation of Dijkstra's algorithm one should use special data-structures for storing O such that the choice of v as the $o \in O$ with minimal tentative distance can be performed efficiently. Keywords in this respect are k-heaps or Fibonacci heaps.

A REAL-WORLD PROBLEM

See Mathematica-Demo and Lecture Notes.