Navigating in a graph by aid of its spanning tree

- Feodor Dragan
  - Kent State University, USA.
- Martín Matamala
  - Universidad de Chile, Chile.

Navigating in a graph

- Rules to advance in a graph from a given vertex towards a target vertex along a path close to shortest.

- Communication networks: a mechanism that can deliver packets of information from any vertex of a network to any other vertex.

Routing from v to y.

- A vertex *v* needs to decide:
  - If the packet has reached its destination *y*.
  - And if not, to which of its neighbors *v** forward the packet.

- Information locally available at *v*.
  - Full knowledge of its neighborhood.
  - A piece of global information; A sense of direction to each destination.
  - Address of the destination vertex *y*.

Routing Strategies

- *Full-tables.*
  - For each destination *y* the next vertex *v** is known.
  - *Routing is along shortest paths.*
  - *$O(n \log(\Delta))$ local memory requierement.*
- *Routing along shortest* *needs* $\Omega(n \log(\Delta))$.
- *Low local memory* requires
  - *Restricted classes of graphs.*
  - *Routing along sub-optimal paths.*

Routing Strategies

- *(greedy) Geographic routing* from *v* to *y*
  - *v** is chosen as the geographically closest to *y*.
  - Coordinates in the underlying physical space are known.
  - No delivery guarantee (existence of lakes).

Routing Strategies

- *Virtual geographic routing in a metric space (X,d).*
  - *Delivery guaranted when G admits a greedy embedding:*
    - *a function f such that for every x and y, there is a neighbor u of x such that $d(f(u),f(y))<d(f(x),f(y))$.*
  - *Routing using suboptimal paths.*

Routing Strategies

Greedy embedding in a metric space (X,d)

X a d-dimensional normed vector spaces.

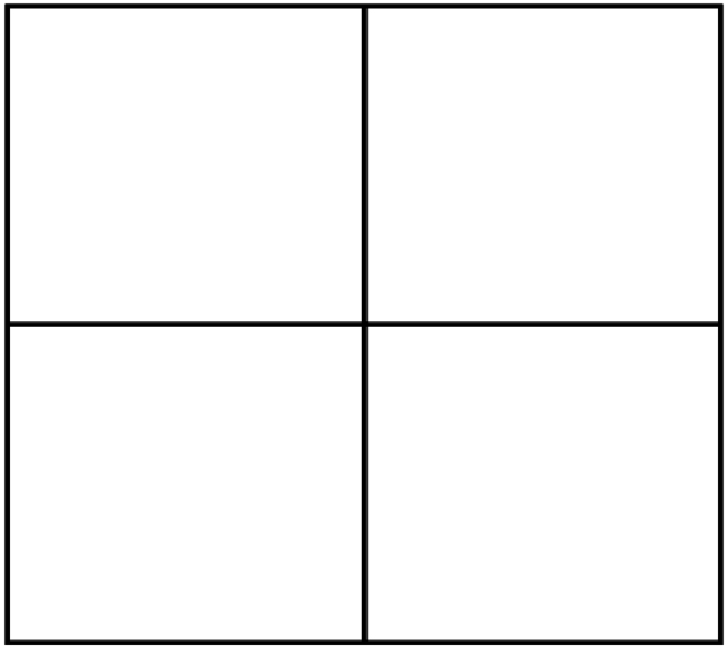
- *Euclidean plane:* some simple graphs have not g.e.
- $d=c \log(n)$ allows greedy embedding for all graph on n vertices (tight up to a mutiplicative factor).

X the *Hyperbolic plane*.

- Every graph admits a greedy embedding.

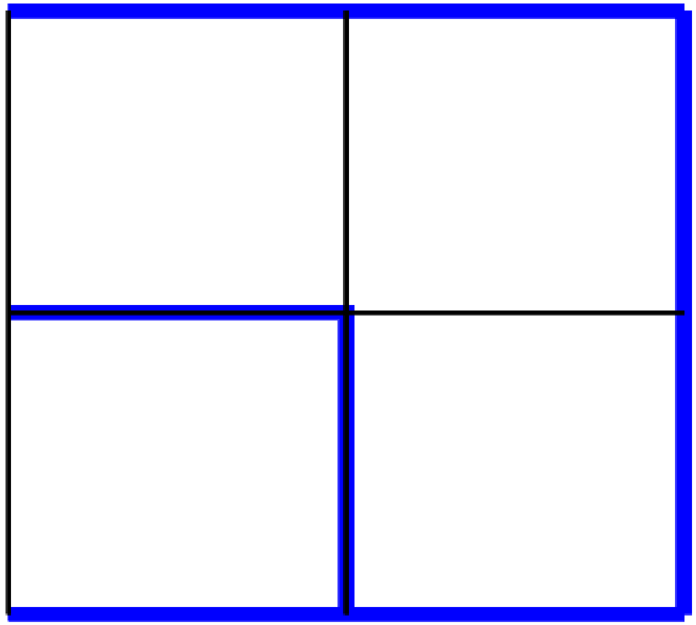
Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G*



Our approach: Greedy routing by aid of a spanning tree

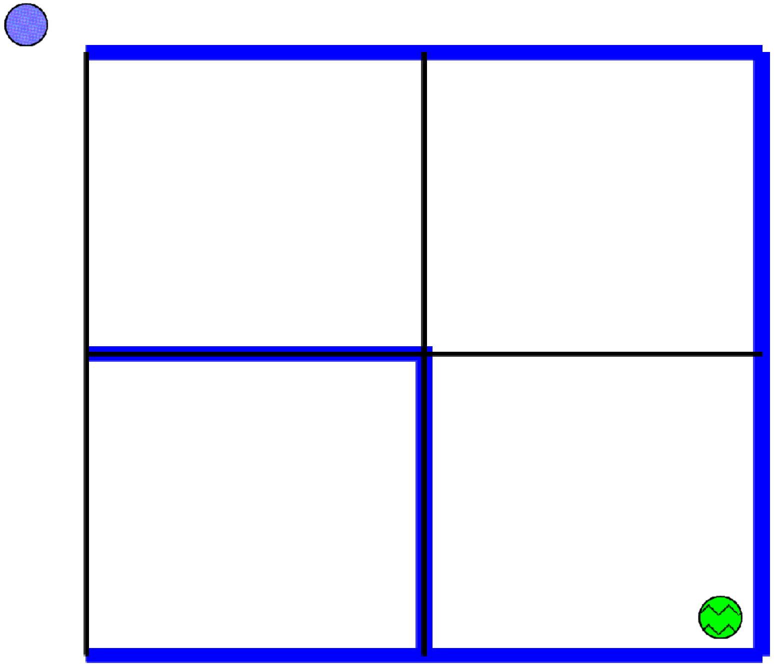
- Given a graph *G* and a spanning tree *T* of *G*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

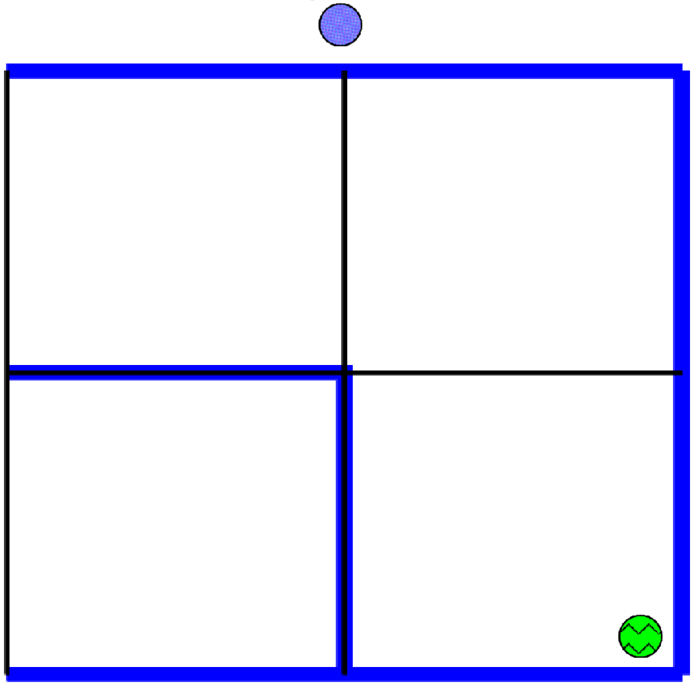
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: *Greedy routing by aid of a spanning tree*

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

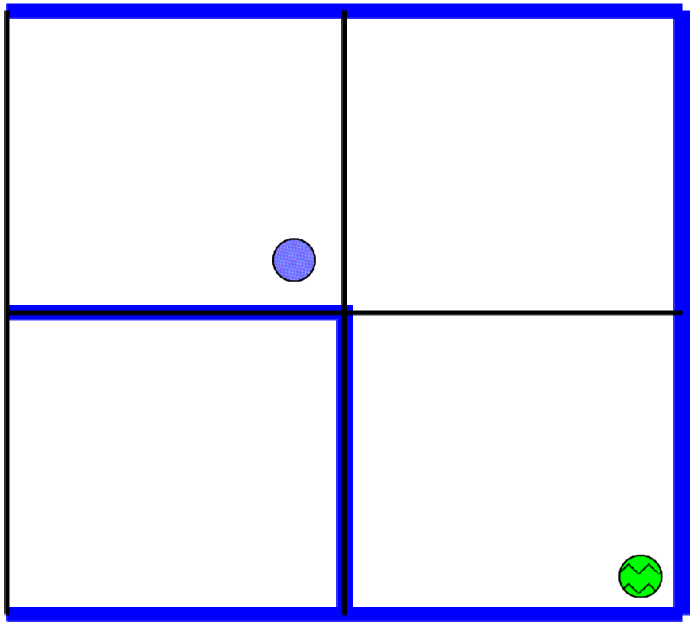
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

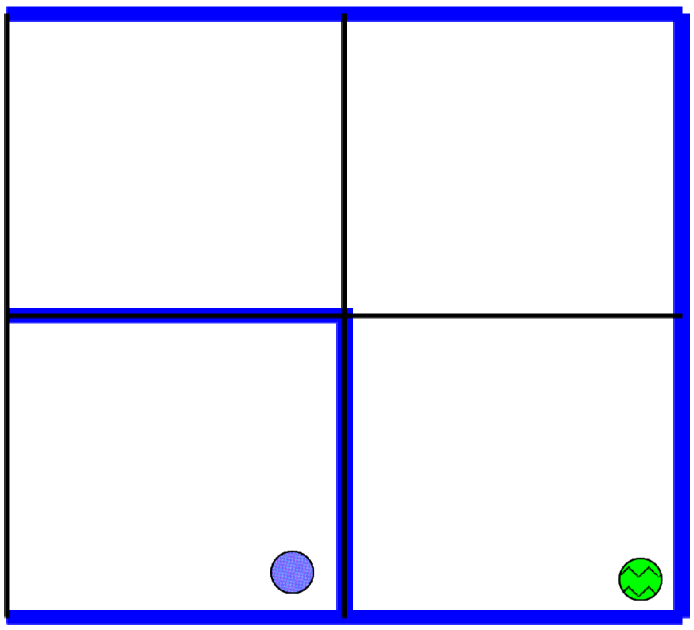
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

go to a neighbor of *v* in *G* closest to *y* in *T*.



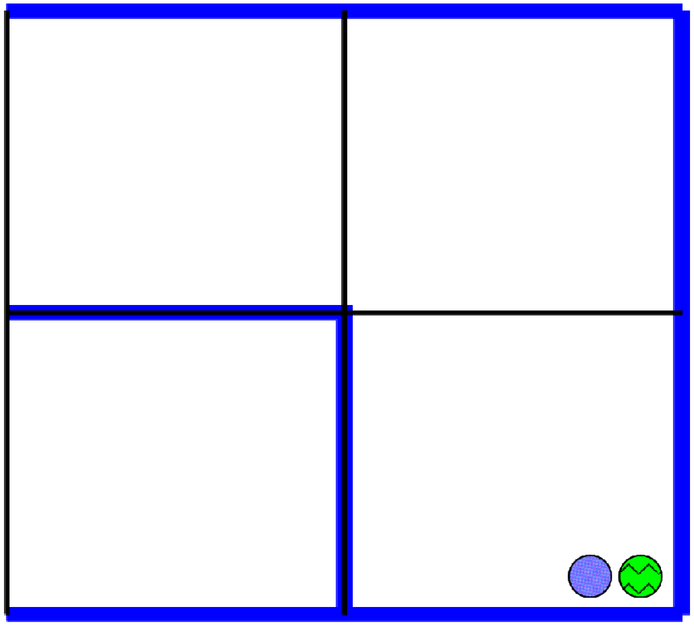
Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

Routing stops when *v*=*y*.

Routing steps =

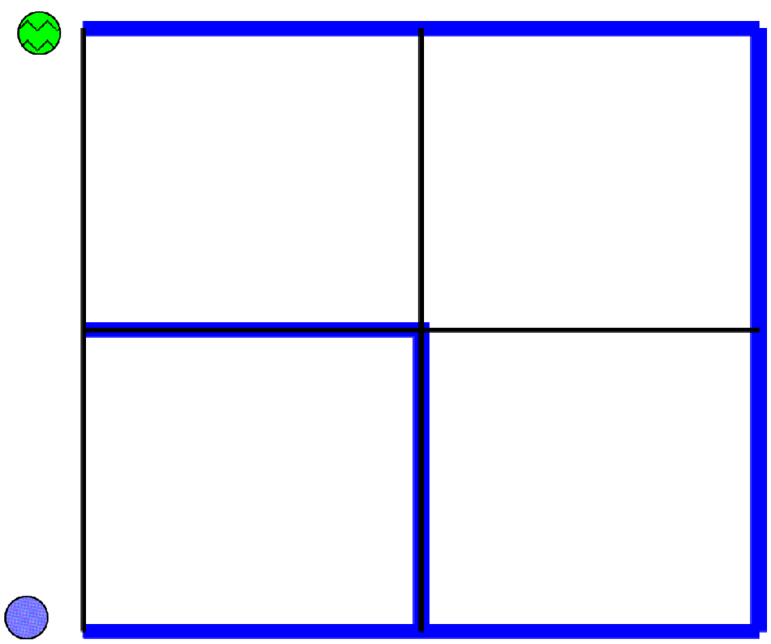
distance in *G*



Our approach: *Greedy routing by aid of a spanning tree*

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

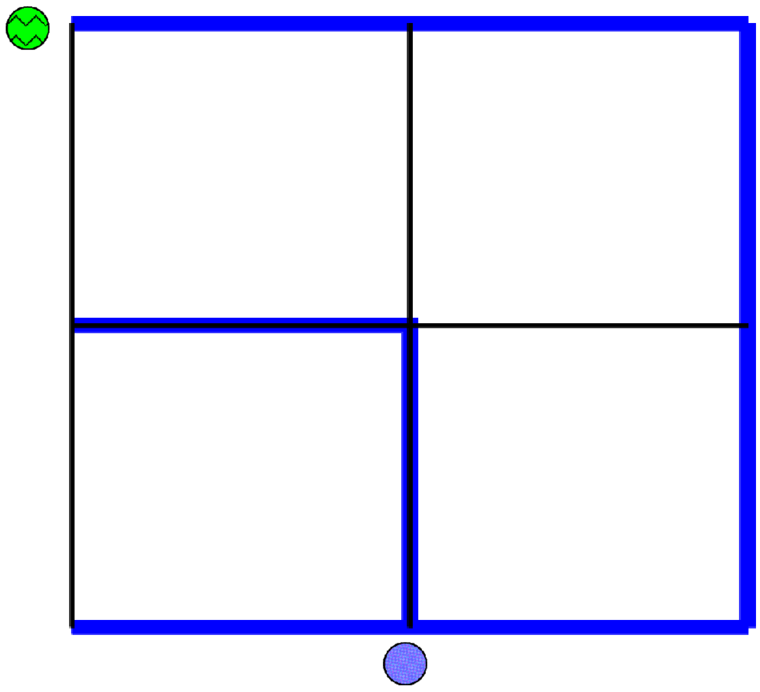
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

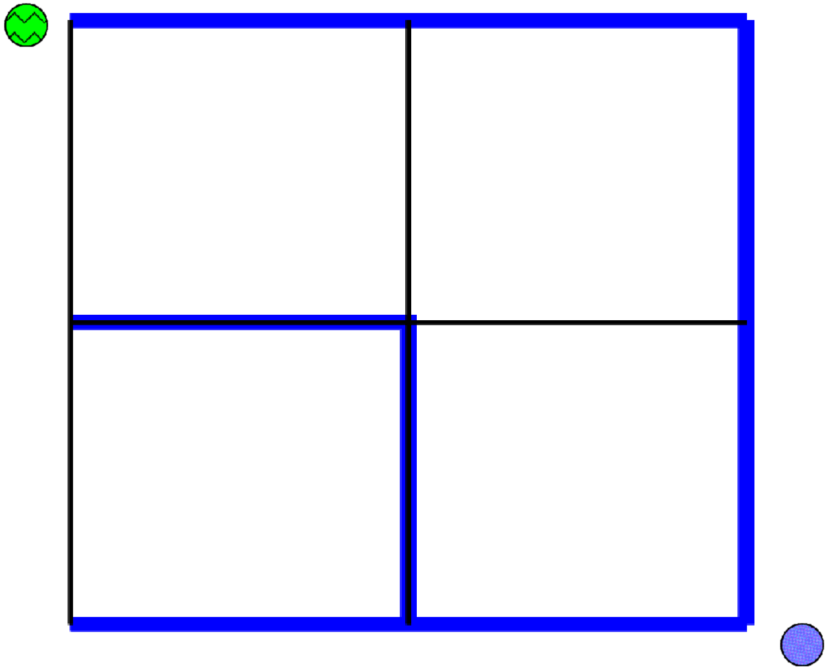
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: *Greedy routing by aid of a spanning tree*

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

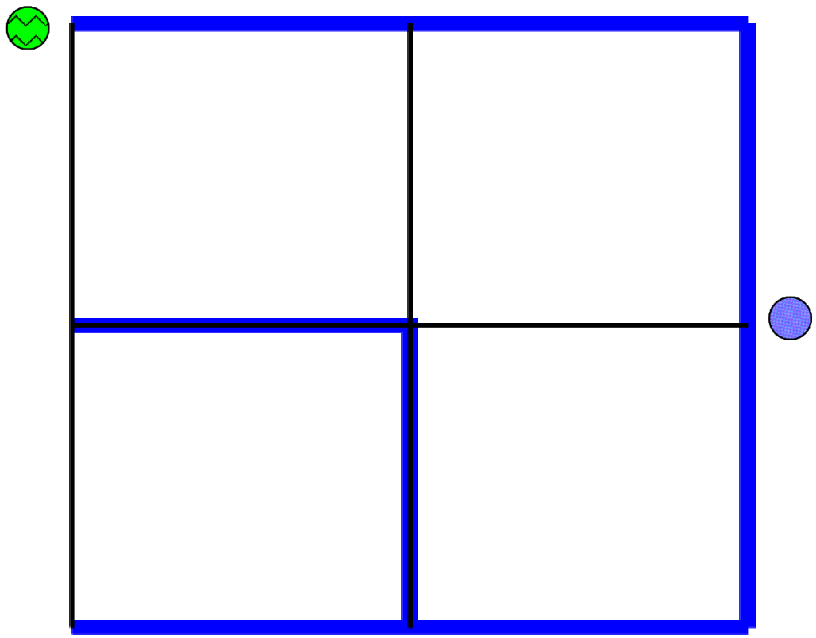
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: *Greedy routing by aid of a spanning tree*

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

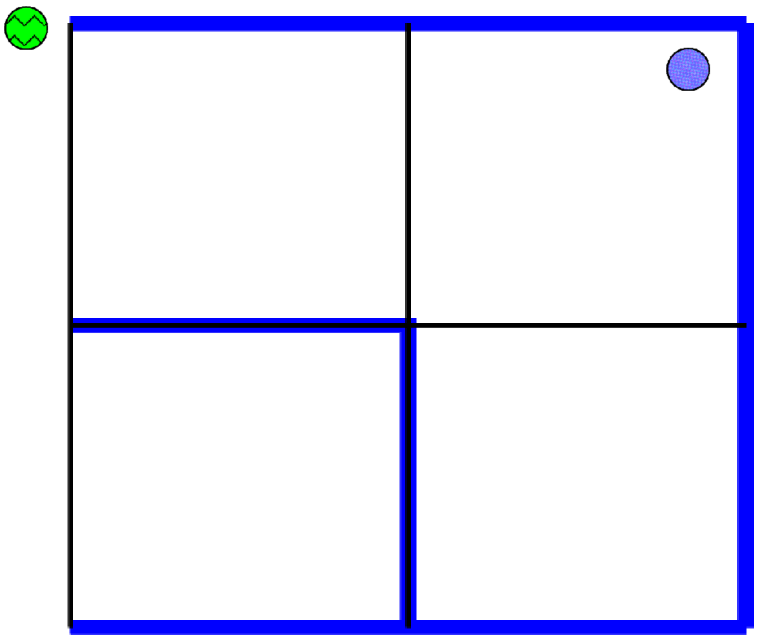
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

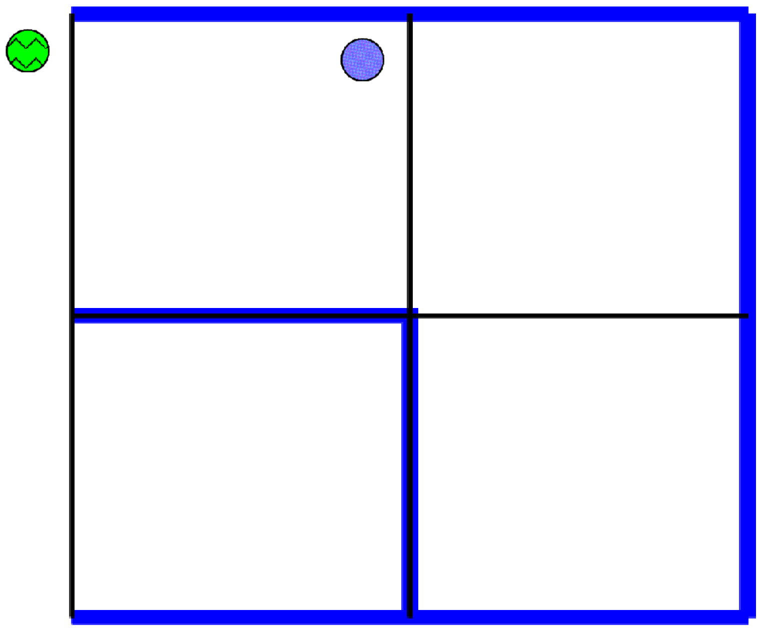
go to a neighbor of *v* in *G* closest to *y* in *T*.



Our approach: Greedy routing by aid of a spanning tree

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

go to a neighbor of *v* in *G* closest to *y* in *T*.

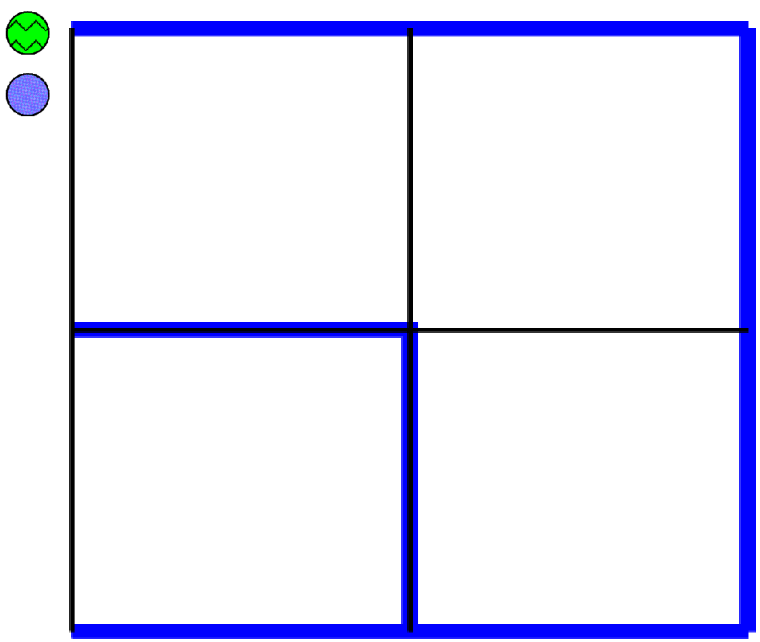


Our approach: *Greedy routing by aid of a spanning tree*

- Given a graph *G* and a spanning tree *T* of *G*.
- To route from *v* to *y* *do the following*

Routing stops when *v*=*y*.

Routing steps ≠ distance in *G*.



Greedy routing by aid of a spanning tree

- *Greedy routing by aid of a spanning tree* is a virtual geographic routing in the metric space *$(V(G),d_T)$*, where *d_T* is the distance defined by the spanning tree *T* of *G*.

Greedy routing by aid of a spanning tree

- *Delivery is guarantee:*
  - Canonical injection is a greedy embedding.
- *Full local knowledge:*
  - Each vertex knows its whole neighborhood.
- *Global (metric) information:*
  - Distances in a spanning tree.

Greedy routing by aid of a spanning tree

- Definition: *Greedy Routing Path (T-GRP)*
  - A path generated by greedy routing by aid of a spanning tree *T.*
- Properties:
  - *T-GRPs* are induced paths.
  - A tale of a *T-GRP* is a *T-GRP.*

Greedy routing by aid of a spanning tree

- Definition: Greedy length: $gr_T(x,y)$.
  - *Length of a longest T-GRP from x to y using tree T.*
- Property:
  - $gr_T(v,y) \le d_T(v,y)$, for every v,y: *Not worse than T.*

Carcasses

- Definition: An spanning tree T is a *r-additive carcass* if

$$gr_T(x,y) \leq d_G(x,y)+r,$$ for every pair x and y.

When $r=0$ the tree T is an *optimal* carcass.

Carcasses

- Theorem

  The following classes of graphs admits additive r-carcass.

- Chordal bipartite graphs: $r=4$.
- 3-sun-free chordal graphs: $r=4$.
- Chordal graphs:
  - $r=w(G)+1$, $w(G)$ size of the maximum clique of G.
  - In particular, k-trees: $r=k+2$.

Carcasses

- Theorem

  The following classes of graphs admits an optimal carcass.

- Distance-hereditary graphs.
- Grids.
- Hypecubes.
- Dually chordal.

Distance Hereditary graphs

Theorem

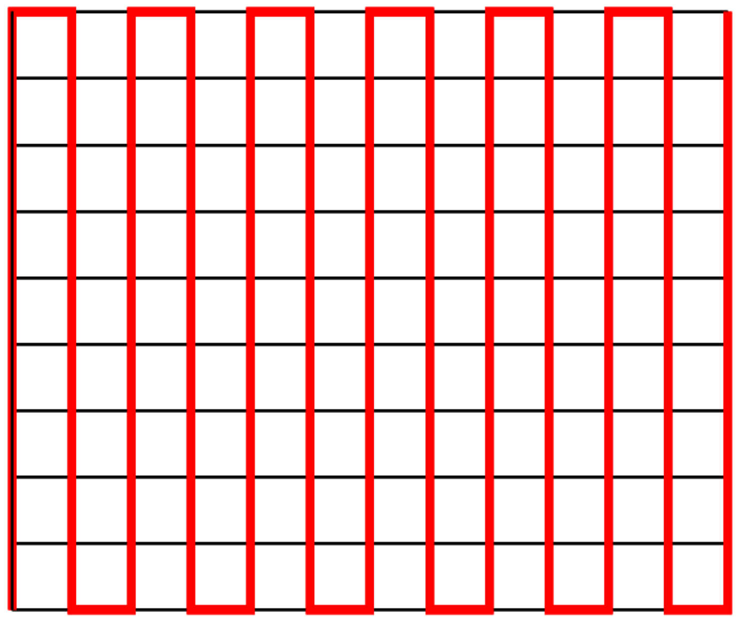
Every spanning tree is an optimal carcass for a distance hereditary graph G.

Proof

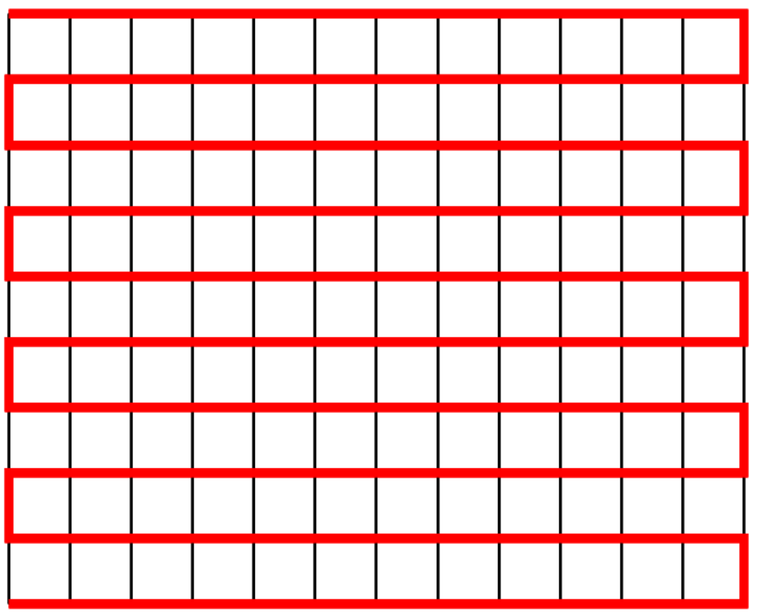
- In *G* every induced path is a shortest path.
- GRPs are induced paths.

Grids

Two special Hamiltonian paths



By columns



By rows

Grids

Theorem: *row and column Hamiltonian paths are optimal carcasses for a grid.*

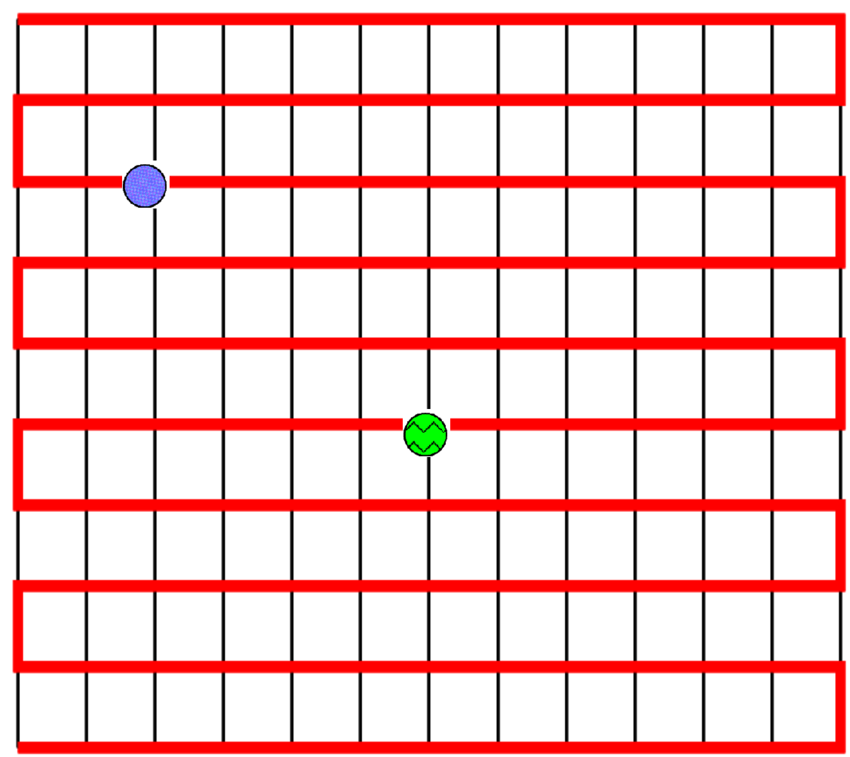
Proof (Row Hamiltonian path)

y and *v* are in different columns.

*v**=

down neighbor of *v* if *y* is below *v*.

up neighbor of *v* if *y* is above *v*.



Grids

Theorem*: row and column Hamiltonian paths are optimal carcasses for a grid.*

Proof (Row Hamiltonian path)

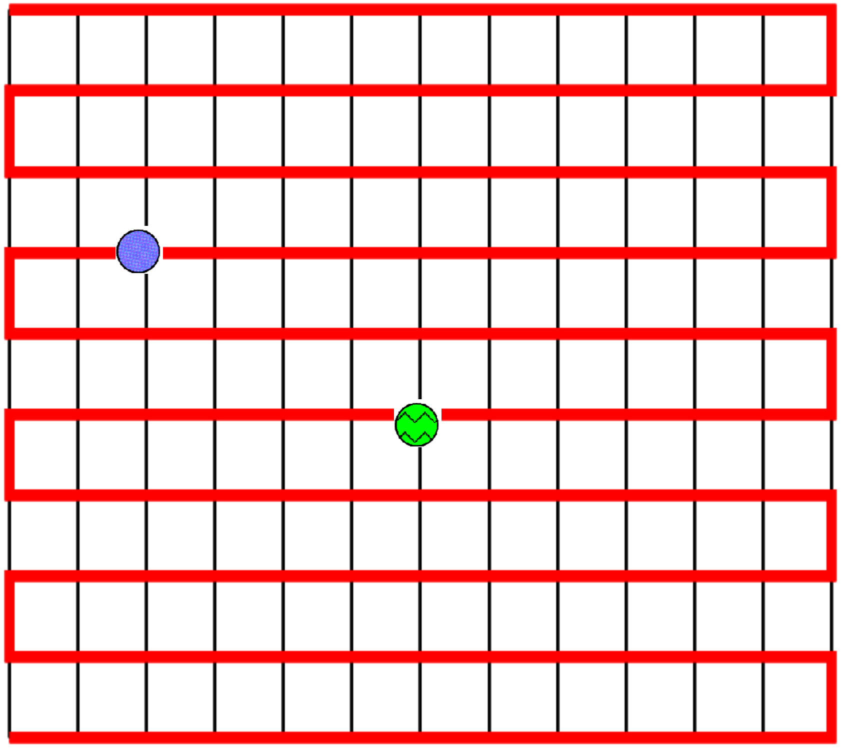
y and *v* are in different columns.

again

*v**=

down neighbor of *v* if *y* is below *v*.

up neighbor of *v* if *y* is above *v*.



Grids

Theorem: *row and column Hamiltonian paths are optimal carcasses for a grid.*

Proof (Row Hamiltonian path)

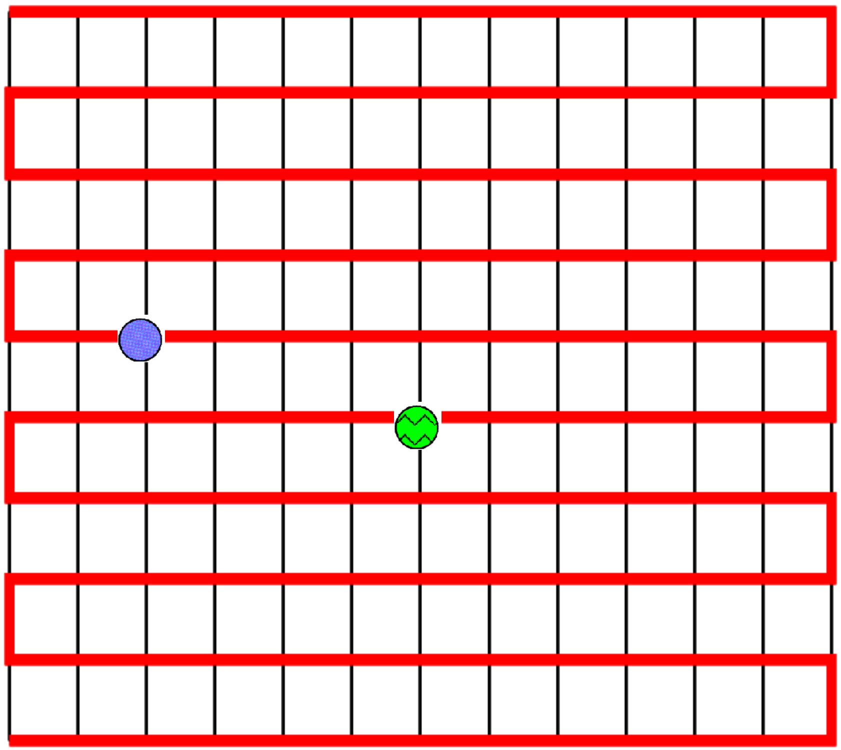
y and *v* are in different columns.

again

*v**=

down neighbor of *v* if *y* is below *v*.

up neighbor of *v* if *y* is above *v*.

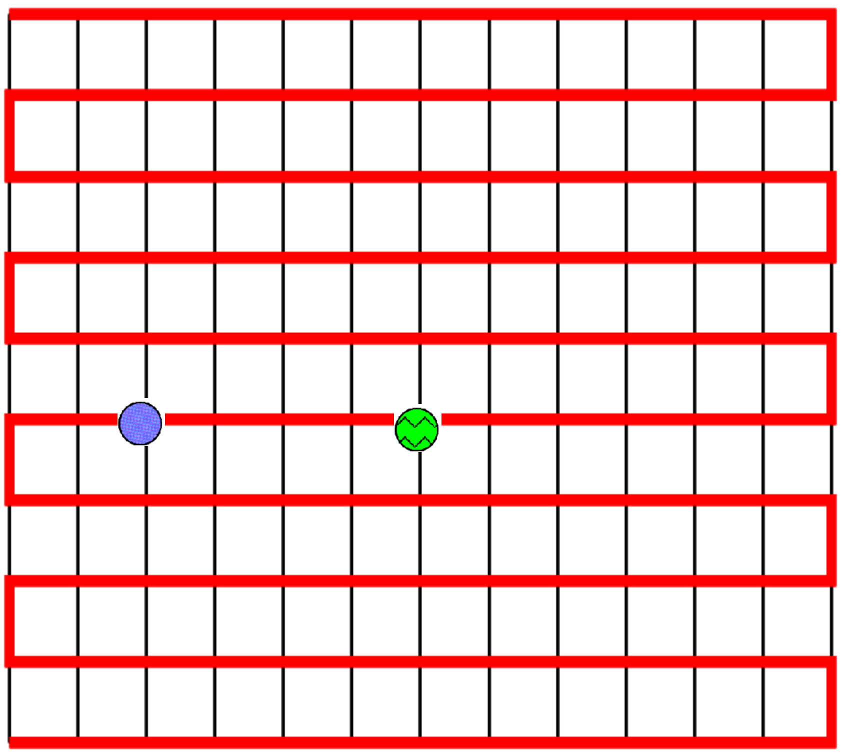


Grids

Theorem: *row and column Hamiltonian paths are optimal carcasses for a grid.*

Proof (Row Hamiltonian path)

y and *v* are in the same column.



*v**=

right neighbor of *v* if *y* is to the *right* of *v*.

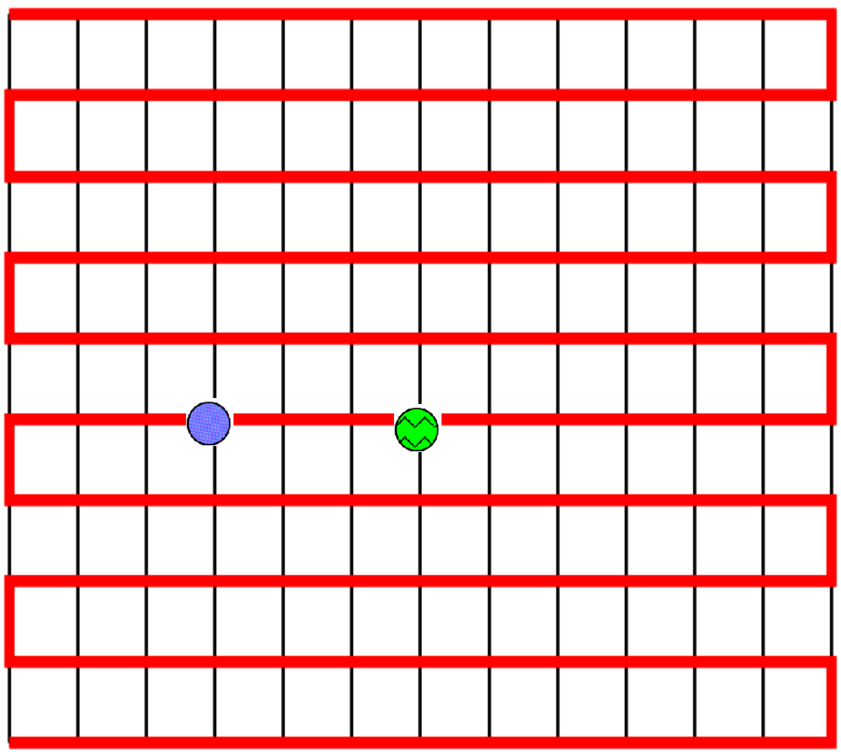
left neighbor of *v* if *y* is to the *left* of *v*.

Grids

Theorem: *row and column Hamiltonian paths are optimal carcasses for a grid.*

Proof (Row Hamiltonian path)

y and *v* are in the same column.



v^*=

right neighbor of *v* if *y* is to the *right* of *v*.

left neighbor of *v* if *y* is to the *left* of *v*.

Grids

Theorem*: row and column Hamiltonian paths are optimal carcasses for a grid.*

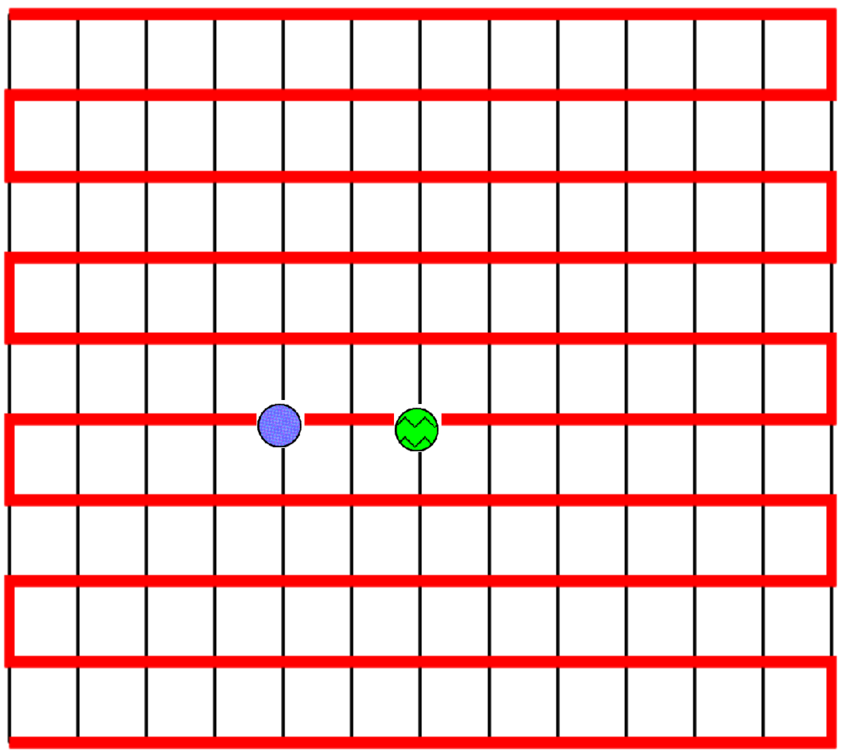
Proof (Row Hamiltonian path)

y and *v* are in the same column.

*v**=

right neighbor of *v* if *y* is to the *right* of *v*.

left neighbor of *v* if *y* is to the *left* of *v*.



Grids

Theorem*: row and column Hamiltonian paths are optimal carcasses for a grid.*

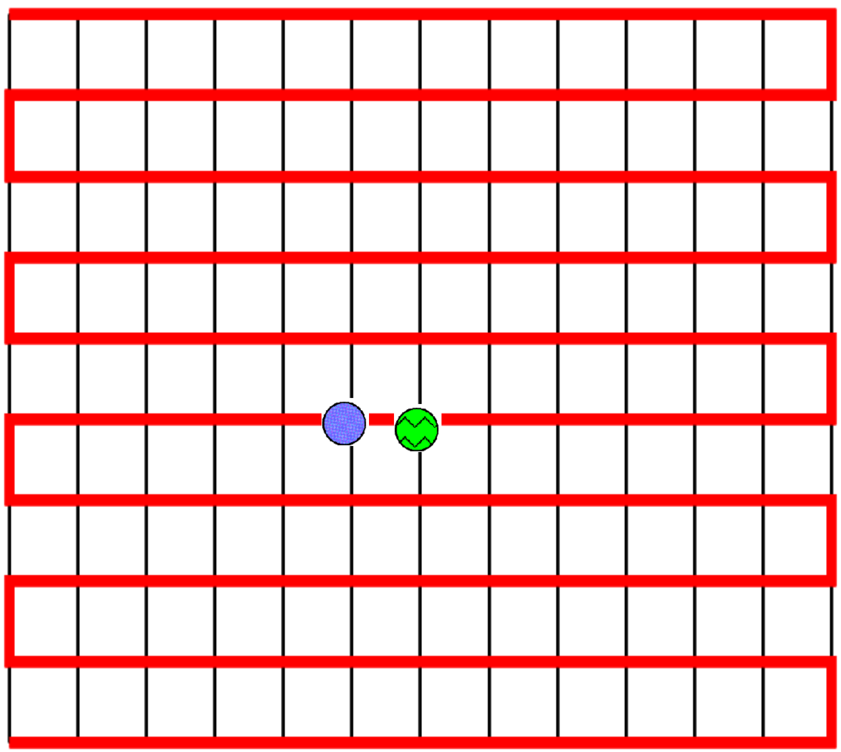
Proof (Row Hamiltonian path)

y and *v* are in the same column.

*v**=

right neighbor of *v* if *y* is to the *right* of *v*.

left neighbor of *v* if *y* is to the *left* of *v*.

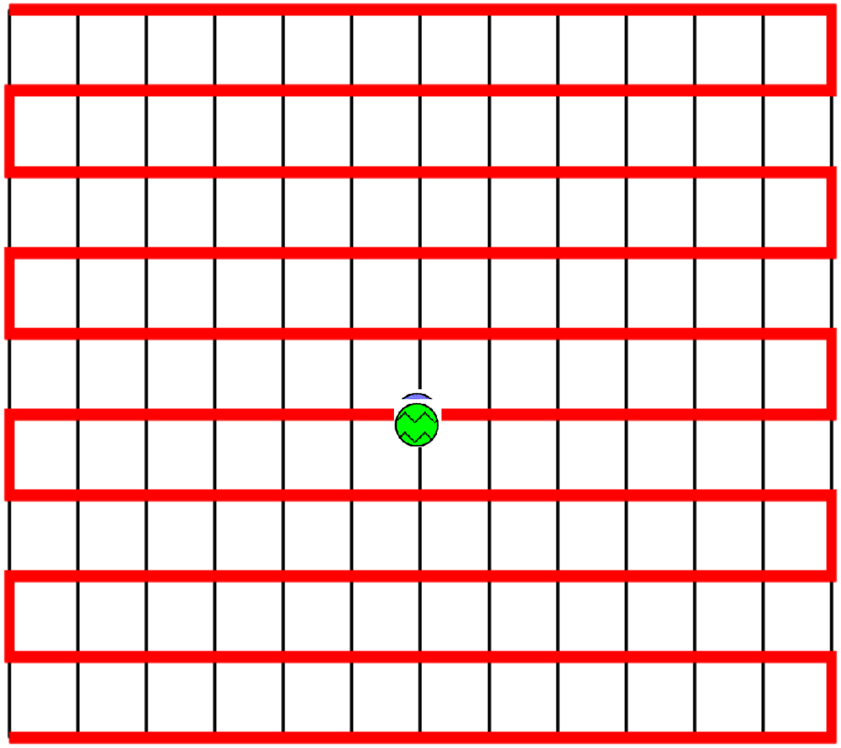


Grids

Theorem: *row and column Hamiltonian paths are optimal carcasses for a grid.*

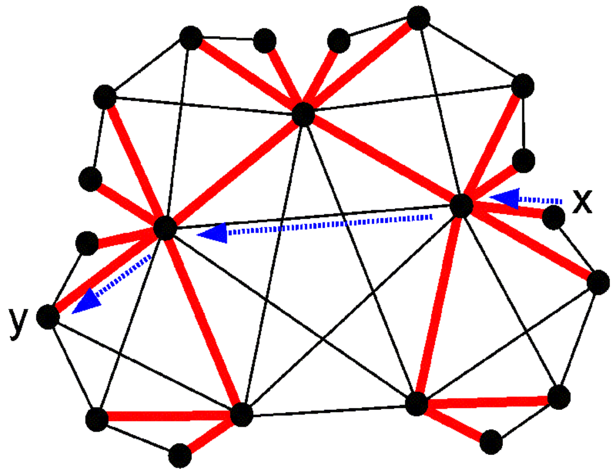
Proof (Row Hamiltonian path)

Routing stops when $v^*=y$



Locally connected spanning trees

An spanning tree *T* is *locally connected* if the neighborhood of each vertex induces a subtree of *T*.



Locally connected spanning trees

Theorem: *Every locally connected spanning tree is an optimal carcass.*

Proof: By induction we prove that *v** belongs to a *v-y* shortest path.

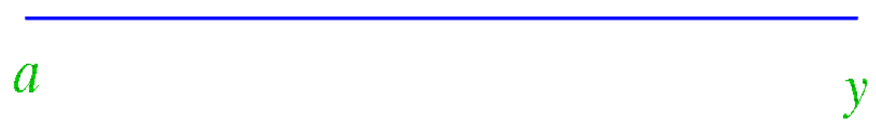
- Let *P=(v,a,b,...,y)* a shortest *v-y* path.
- By assuming that *b** belong to a *b-y* shortest path, we prove that *v*~b*.

Locally connected spanning trees

Theorem: *Every locally connected spanning tree is an optimal carcass.*

Proof: By induction we prove that *v** belongs to a *v-y* shortest path.

- Let *P=(v,a,b,...,y)* a shortest *v-y* path.
- By assuming that *b** belong to a *b-y* shortest path, we prove that *v*~b*.
- By definition of a GRP, *v*,b** are in *aTy*.

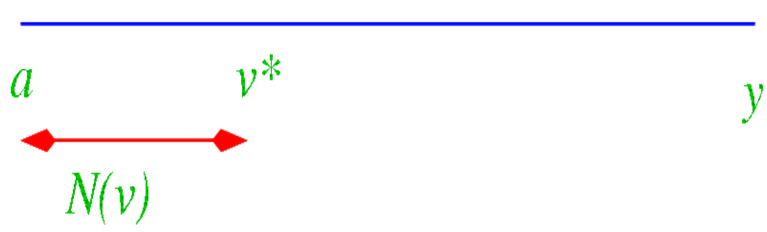


Locally connected spanning trees

Theorem: *Every locally connected spanning tree is an optimal carcass.*

Proof: By induction we prove that *v** belongs to a *v-y* shortest path.

- Let *P=(v,a,b,...,y)* a shortest *v-y* path.
- By assuming that *b** belong to a *b-y* shortest path, we prove that *v*~b*.
- By definition of a GRP, *v*,b** are in *aTy*.
- *b** belongs to *v*Ty:*
  - *a~v* and *v*~v.* Then, *aTv** is included in *N(v).* (*T* is l.c.)

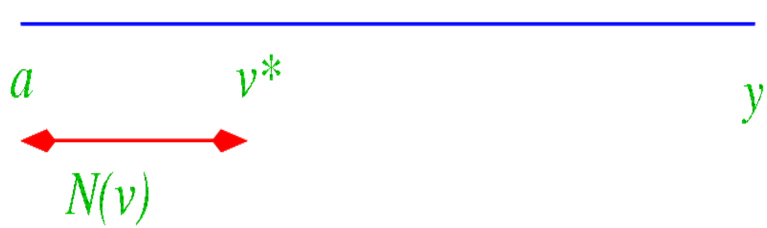


Locally connected spanning trees

Theorem: *Every locally connected spanning tree is an optimal carcass.*

Proof: By induction we prove that v^* belongs to a v-y shortest path.

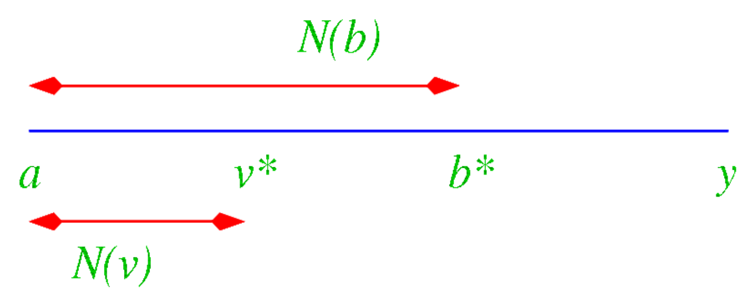
- Let $P=(v,a,b,\dots,y)$ a shortest v-y path.
- By assuming that b^* belong to a b-y shortest path, we prove that $v^*\sim b$.
- By definition of a GRP, v^*,b^* are in aTy.
- b^* belongs to v^*Ty:
  - $a\sim v$ and $v^*\sim v$. Then, aTv^* is included in $N(v)$. (T is l.c.)
  - b^* cannot be a neighbor of v. (P is a shortest path).



Locally connected spanning trees

Theorem: *Every locally connected spanning tree is an optimal carcass.*

Proof: By induction we prove that *v** belongs to a *v-y* shortest path.



- Let *P=(v,a,b,...,y)* a shortest *v-y* path.
- By assuming that *b** belong to a *b-y* shortest path, we prove that *v*~b*.
- By definition of a GRP, *v*,b** are in *aTy*.
- *b** belongs to *v*Ty:*
  - *a~v* and *v*~v.* Then, *aTv** is included in *N(v).* (*T* is l.c.)
  - *b** cannot be a neighbor of *v*. (*P* is a shortest path).
- *a~b, b*~b* and *v** belongs to *aTb*.* Then *v*~b.* (*T* is l.c.)

Dually chordal graphs and locally connected spanning trees

Definition: A graph is dually chordal if it is the intersection graph of maximal cliques of a chordal graph.

Theorem: A graph is dually chordal if and only if it admits a locally connected spanning tree.

Corollary: *Dually chordal graphs admit optimal carcass.*

An invitation to SouthAmerica

LAGOS 2009

Gramados, Brazil,

November 2009.

THANKS!