Tears, Sweat, and Blood: The Cable-Trench Problem and Some Applications

Eric Landquist
elandqui@kutztown.edu

Department of Mathematics
Kutztown University of Pennsylvania

October 21, 2015

Outline

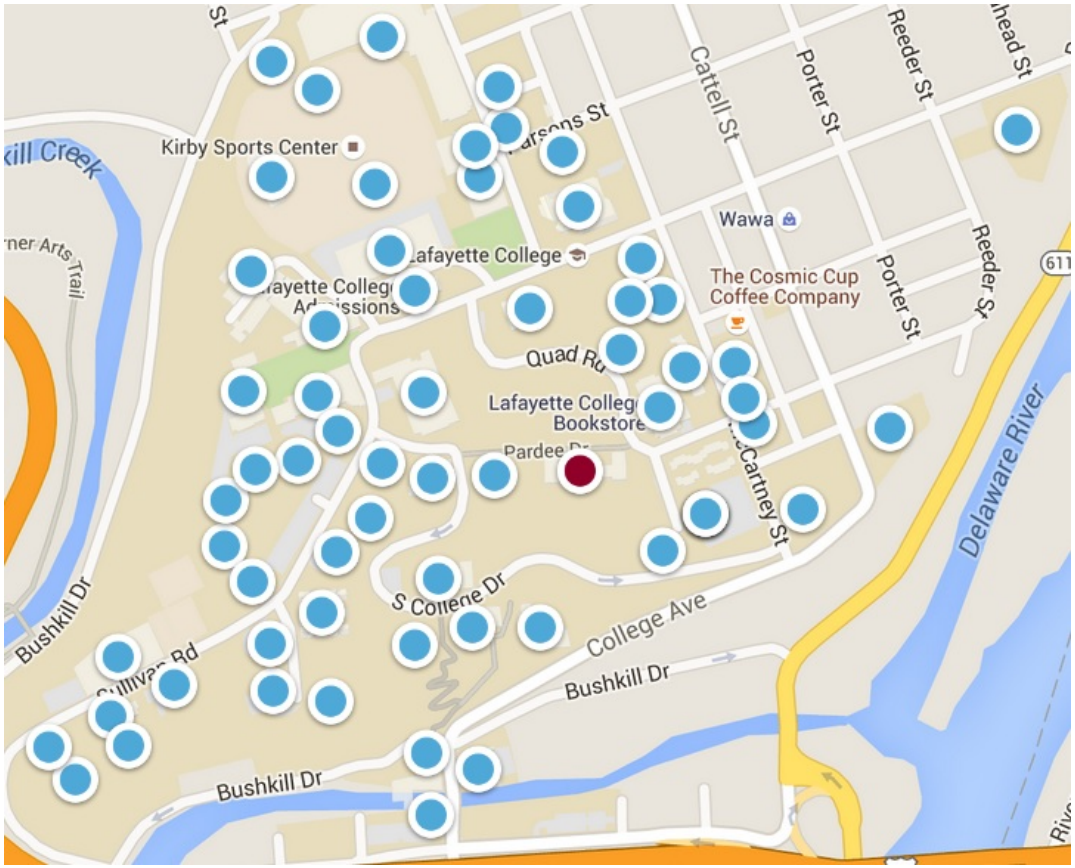
- Background: The Cable-Trench Problem (CTP)
- Solution Approaches
- A Couple Variants of the CTP
- Applications
- Future Work and Open Questions

Background: The Cable-Trench Problem

- Suppose you needed to connect each building on a campus to a hub (IT building) with its own (underground) internet cable.
- How should you dig the trenches between buildings to minimize the total cost to dig the trenches and lay the cable?
- We call this problem the **Cable-Trench Problem**.
- If the cables are free, then the solution is a **minimum spanning tree (MST)**.
- If the trenches are free, then the solution is a **shortest path tree (SPT)**.
- In the real world, both cables and trenches have a cost.
- The CTP establishes a continuum between the SPT and MST problems on a weighted graph.

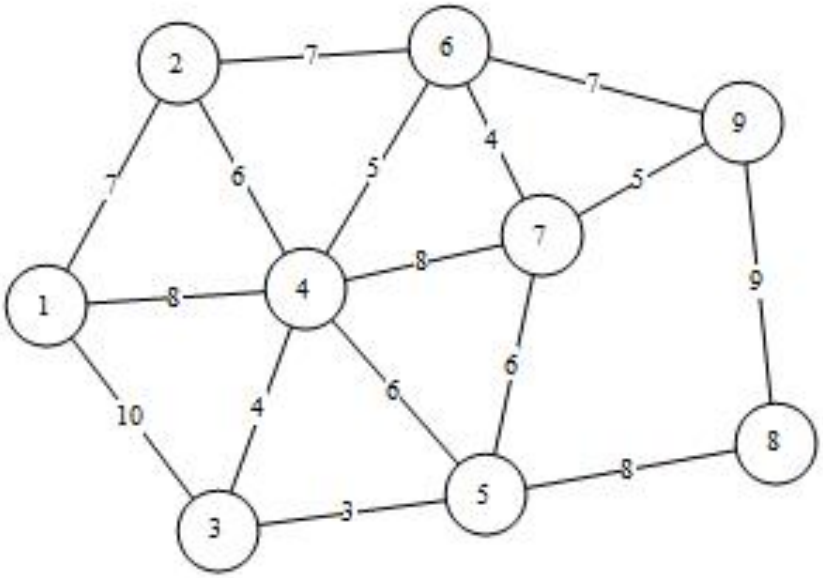
Look familiar?

Pardee Hall is the hub of campus (of course). How would you connect the campus with cables if:
Trenches cost \$5 per foot and cables are \$1 per foot?



Background: Graph Theory

In this context, a **graph** is not a visual representation of a function on the xy-plane, but a collection of **vertices** (or **nodes**) and **edges** that connect two distinct vertices.



- Circles represent vertices, numbered 1 to 9 in this example.
- Line segments represent edges.
- The numbers in the edges are **weights**.
- A weight can represent length, for example.

Background: Graph Theory

Definition

A **graph** $G = (V, E)$ is an ordered pair of a set of **vertices** (or **nodes**), V, and a set of **edges**, E, in which each edge connects exactly two vertices.

Definition

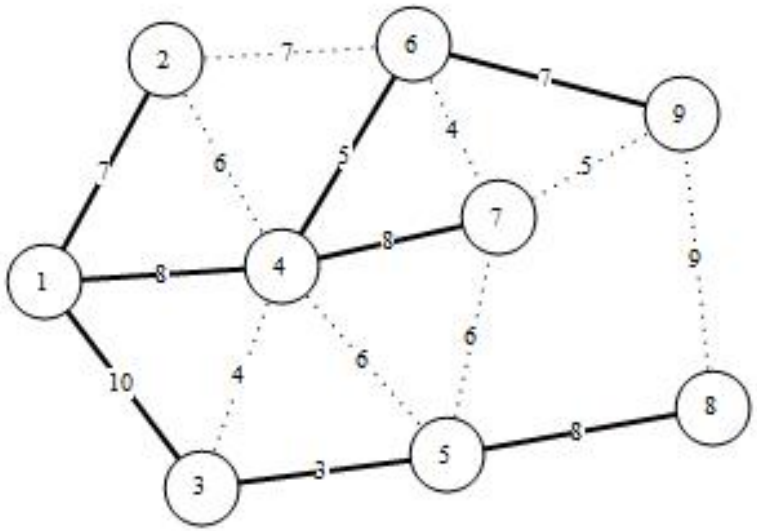
A **weighted graph** is a graph in which each edge, $e \in E$, has a number, denoted $|e|$, called a **weight**, assigned to it.

The weight on an edge can represent cost, length, or another physical attribute.

Background: Spanning Trees

Definition

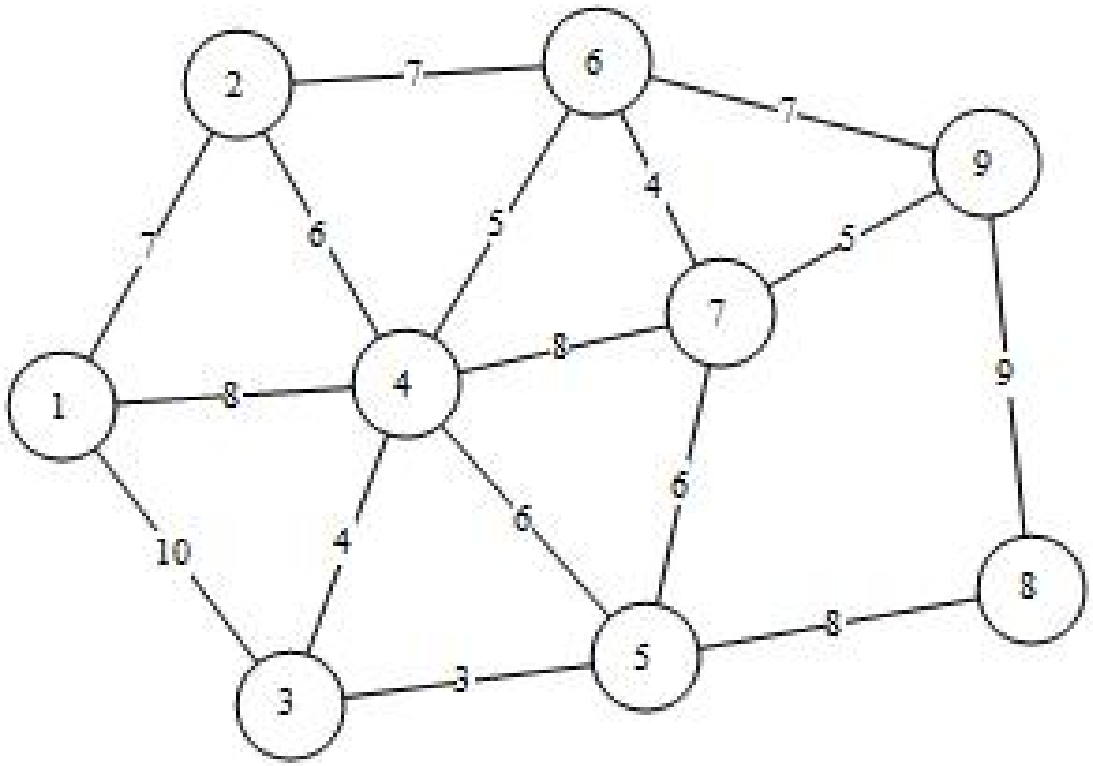
A **spanning tree**, T, of a graph $G = (V, E)$ is a graph with the same vertex set as G, but with no **cycles** or **loops** in the graph. $T = (V, E_T)$.



- Bold edges are in T, dotted edges are in G.
- The terminology for trees is rather intuitive.
- Trees have **branches** and **leaves**,
- and sometimes a **root**.
- We use the vertex 1 (or v_1) as the root node.

An Example: SPT vs. MST

Circles (nodes, vertices) represent the buildings; Node 1 is the hub.
Edges represent allowable routes for digging trenches.
Edge labels (weights) represent distances.



Background: Minimum Spanning Trees

Definition

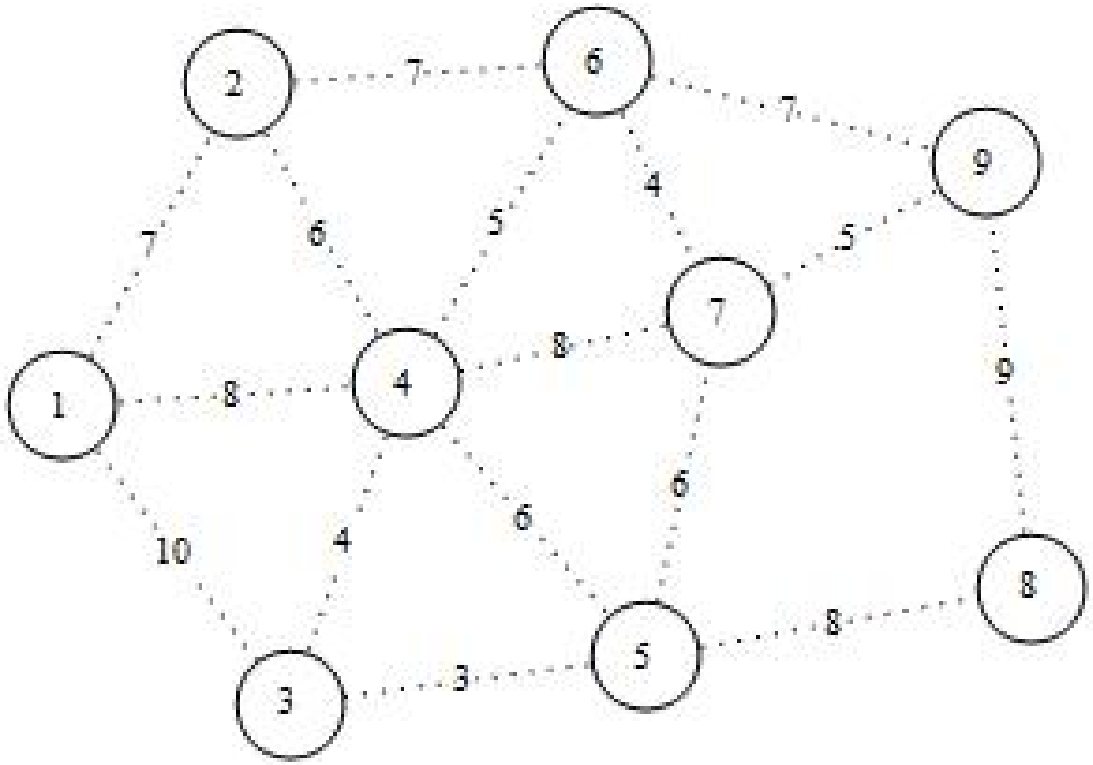
A **minimum spanning tree** (MST) of a weighted graph $G = (V, E)$ is a spanning tree of G with minimum total weight. If $|T|_t$ is the total weight of the edge set, E_T, of T, then

$$|T|_t = \sum_{e \in E_T} |e| \ .$$

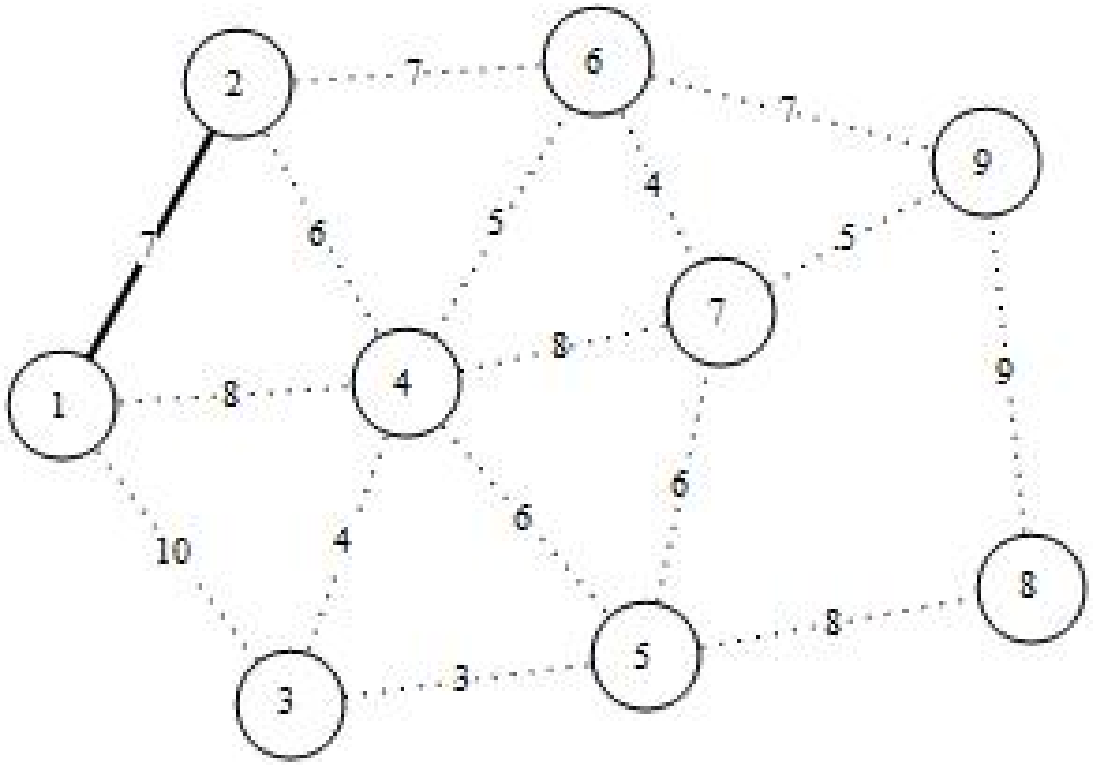
Background: Prim's Algorithm

- Prim's algorithm is one method to find a MST, T, when one vertex of the graph $G = (V, E)$ has been designated as a root node.
- Prim's algorithm is a **greedy** algorithm.
- Begin with $T = (\{v_1\}, \{\})$: the root node and no edges.
- Repeat:
  - Choose an edge $e \in E$ with one vertex in T and one not in T of smallest weight and add the edge and associated vertex to T.
  - Stop when the vertex set of T is V.

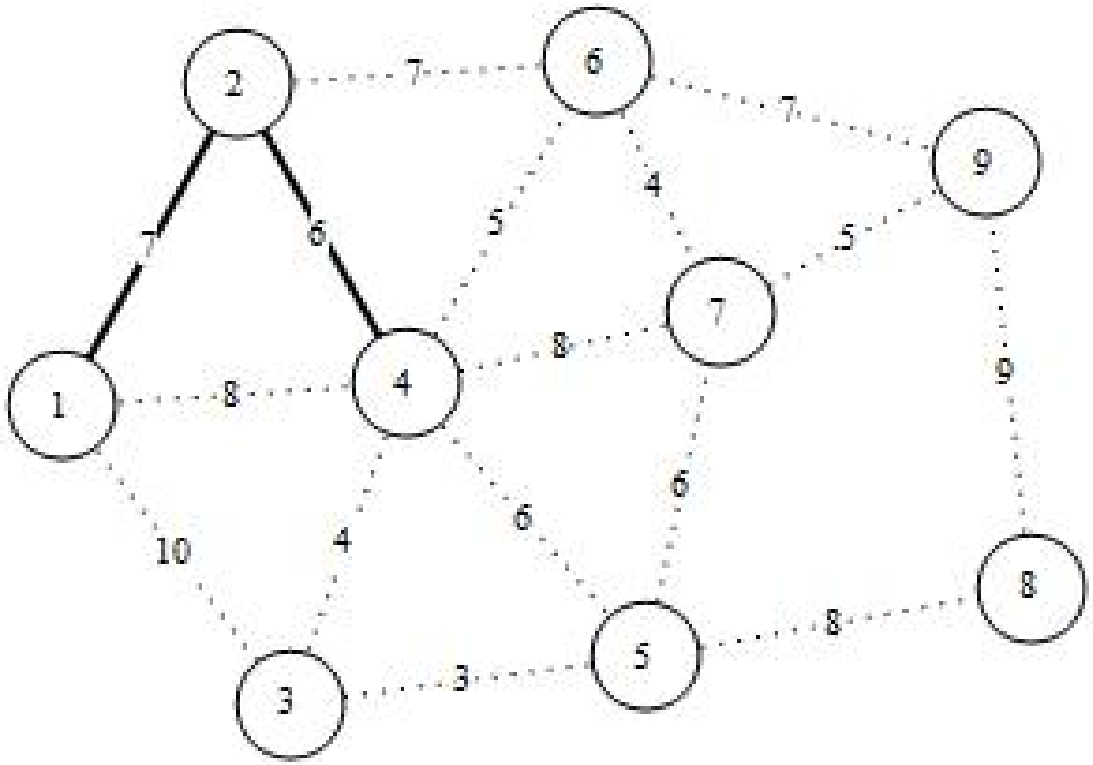
Background: Prim's Algorithm in Action



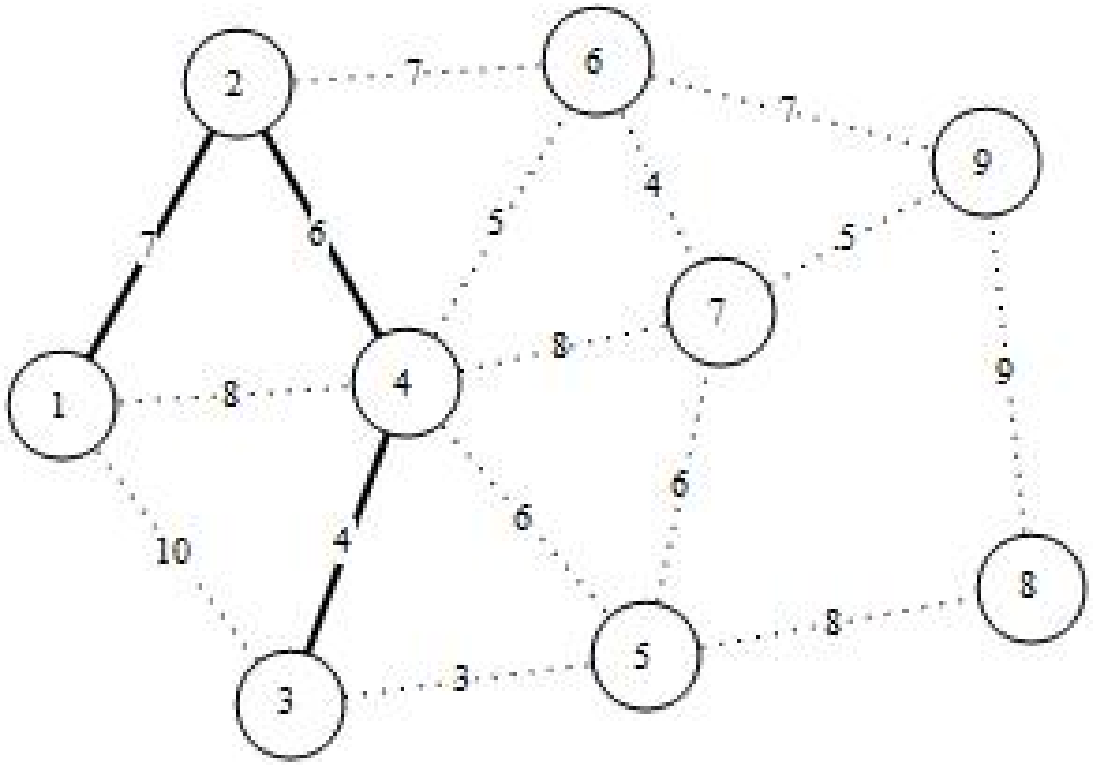
Background: Prim's Algorithm in Action



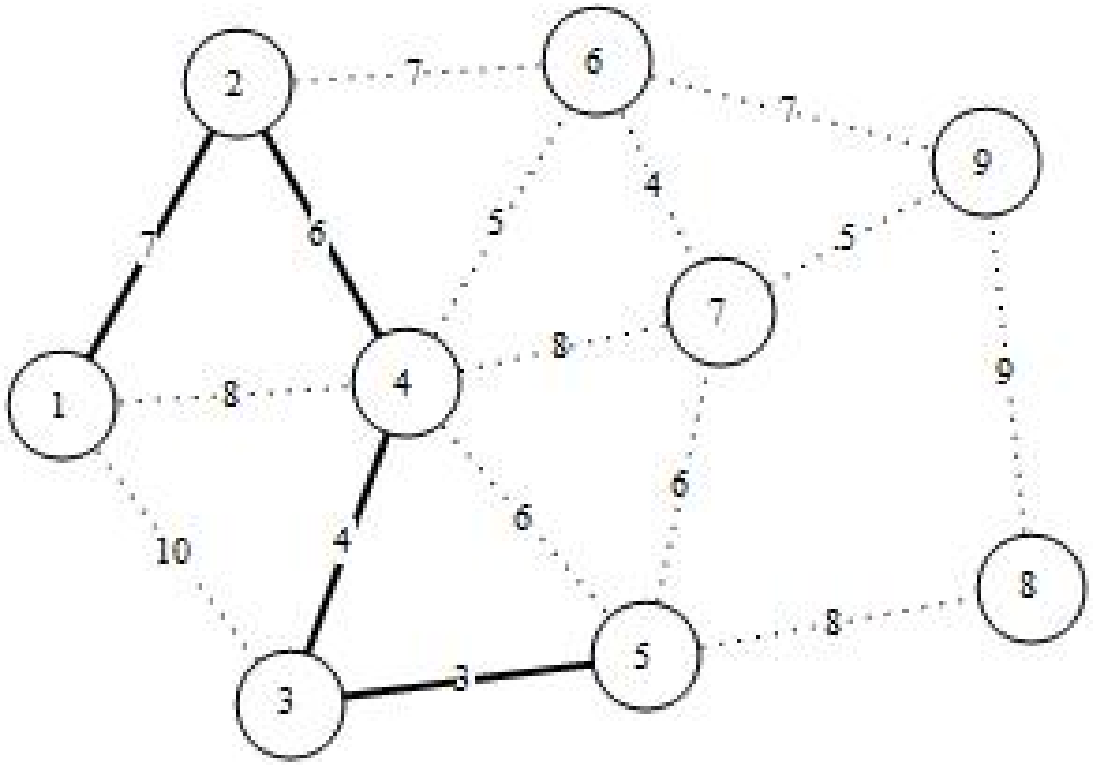
Background: Prim's Algorithm in Action



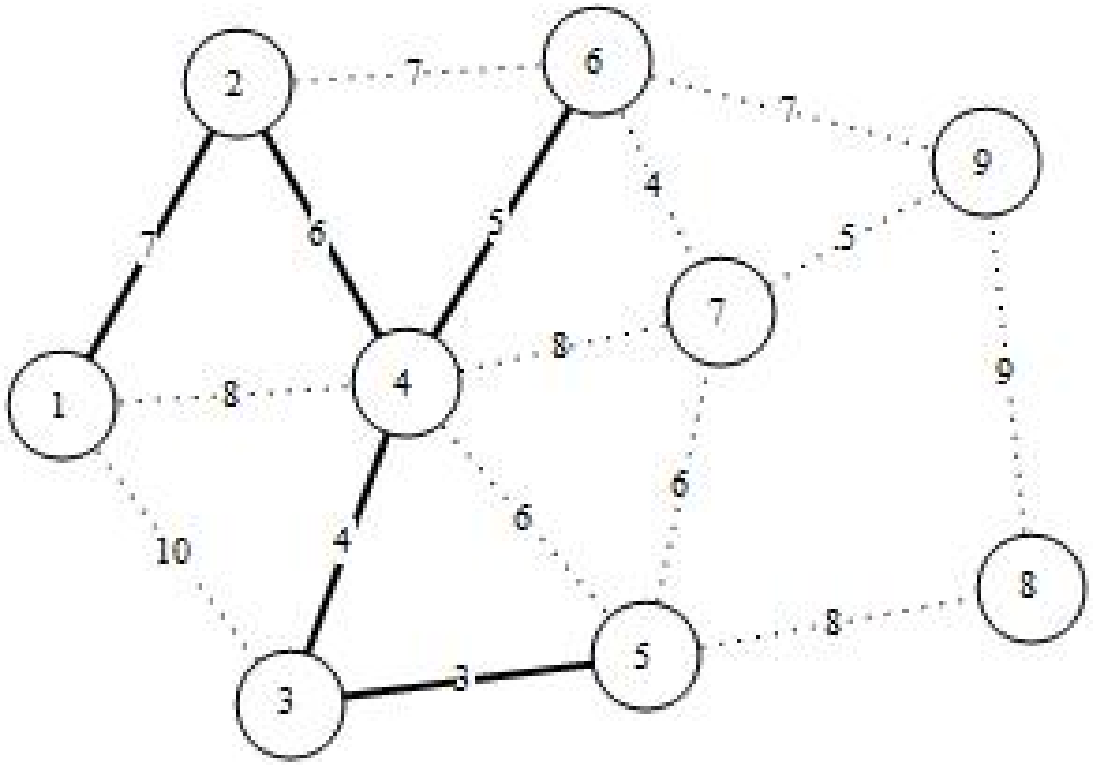
Background: Prim's Algorithm in Action



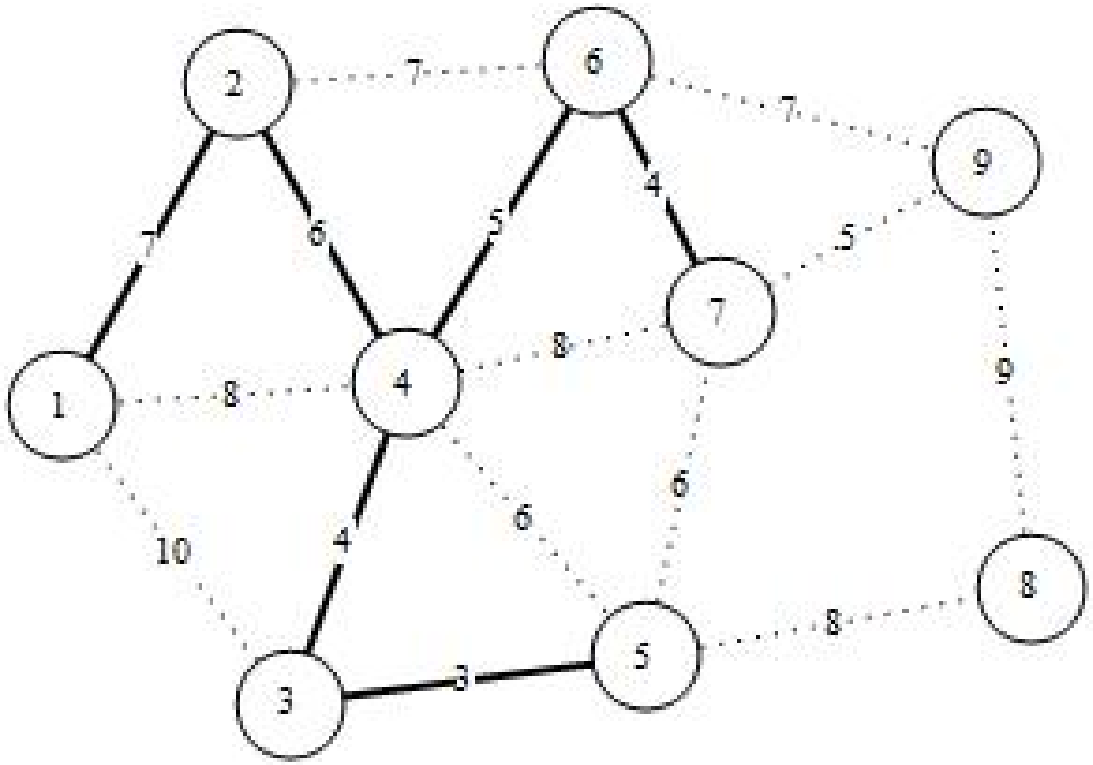
Background: Prim's Algorithm in Action



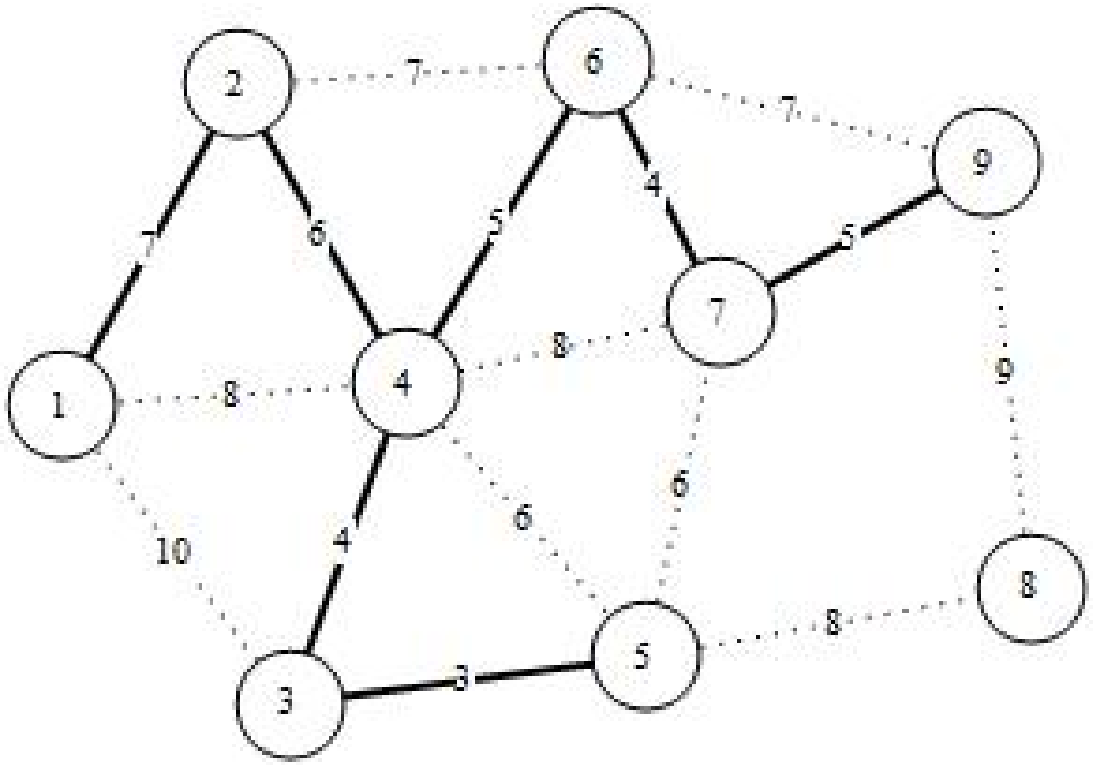
Background: Prim's Algorithm in Action



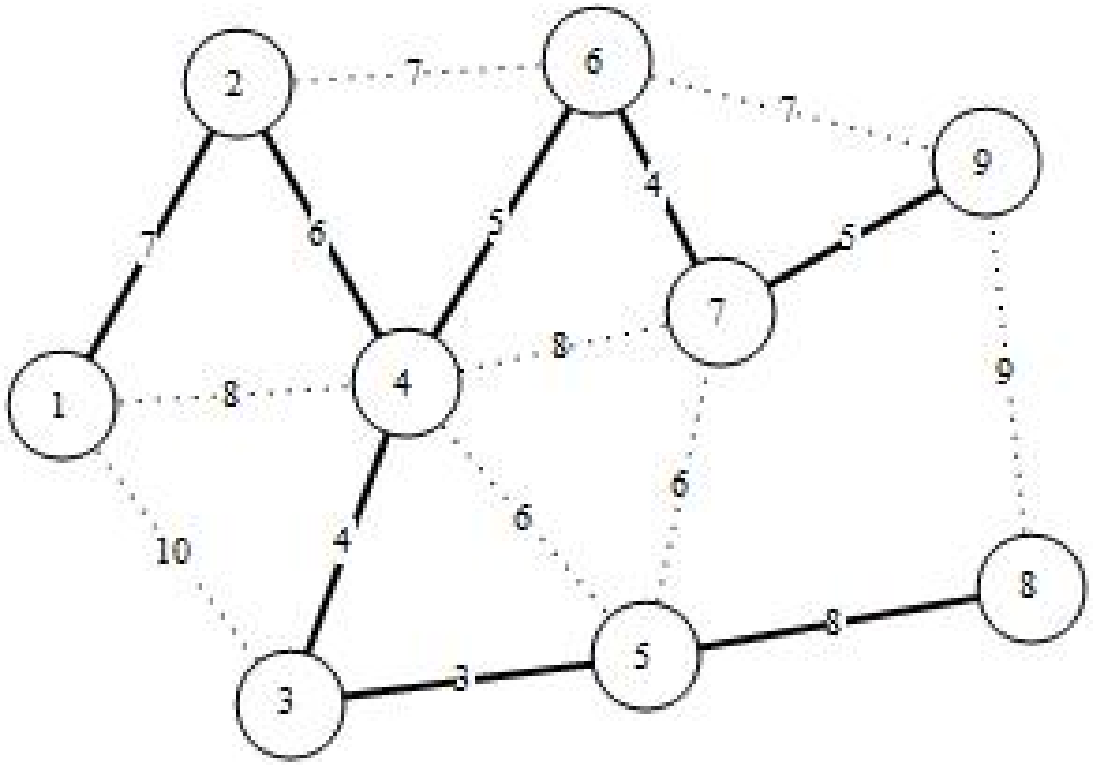
Background: Prim's Algorithm in Action



Background: Prim's Algorithm in Action



Background: Prim's Algorithm in Action



“La, la, la, la, connect the dots!”

Definition

A **(single-source) shortest path spanning tree** (SPT) of a weighted graph $G = (V, E)$ is a spanning tree of G in which the path distance from any vertex to the root in T is the shortest path between those two vertices in G. If $|T|_c$ is the sum of the weights of the edges of each path from each $v \in V$ to the root node of T, then

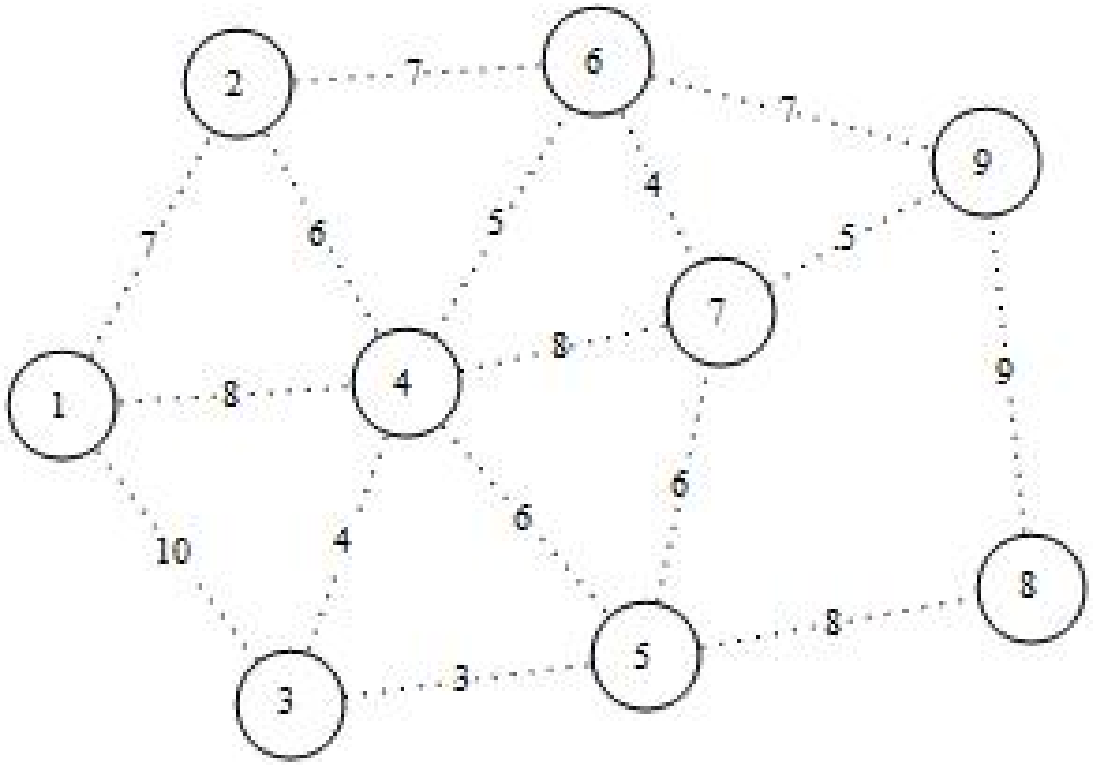
$$|T|_c = \sum_{j=2}^{n+1} \sum_{e \in \mathcal{P}(v_1, v_j)} |e| \ ,$$

where $\mathcal{P}(v_1, v_j)$ is the path from the root node, v_1, to v_j.

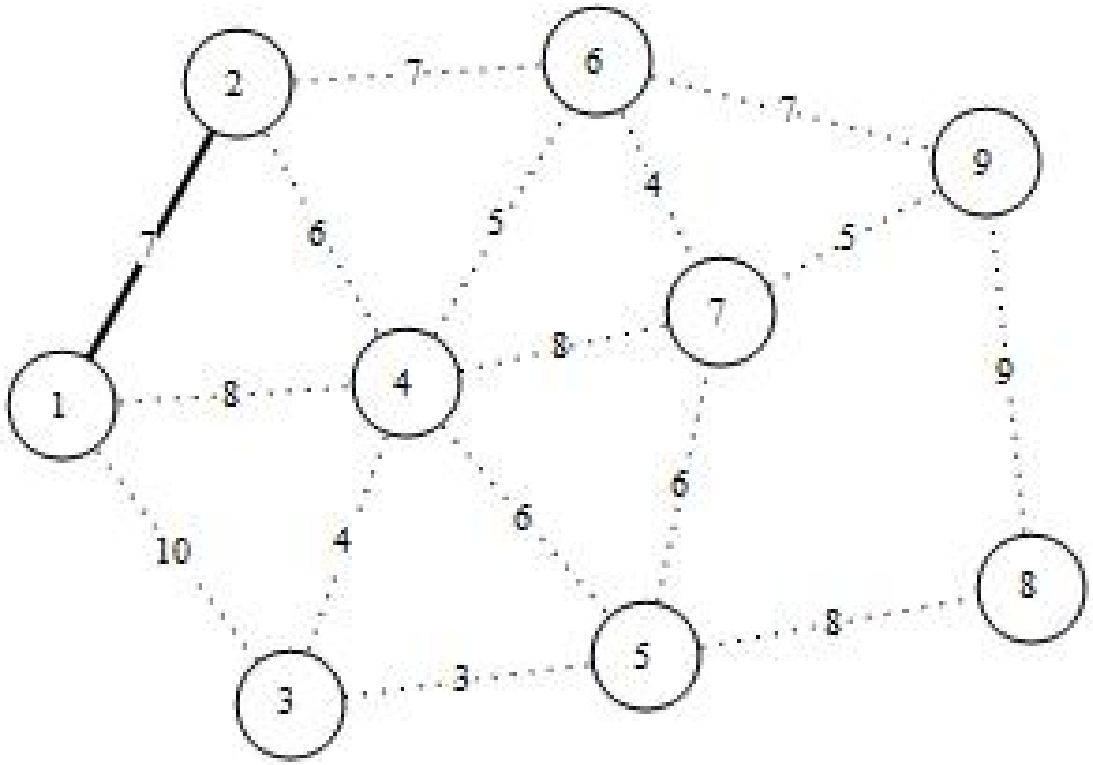
Background: Dijkstra's Algorithm

- Dijkstra's algorithm finds a SPT, T, of $G = (V, E)$, with root node v_1.
- Dijkstra's algorithm is also a greedy algorithm.
- Begin with $T = (\{v_1\}, \{\})$: the root node and no edges.
- Repeat:
  - For each $v \in V$ that neighbors T, find the shortest distance (the sums of the weights) from v to v_1.
  - Add the vertex with shortest distance to v_1 to T, along with the appropriate edge.
  - Stop when the vertex set of T is V.

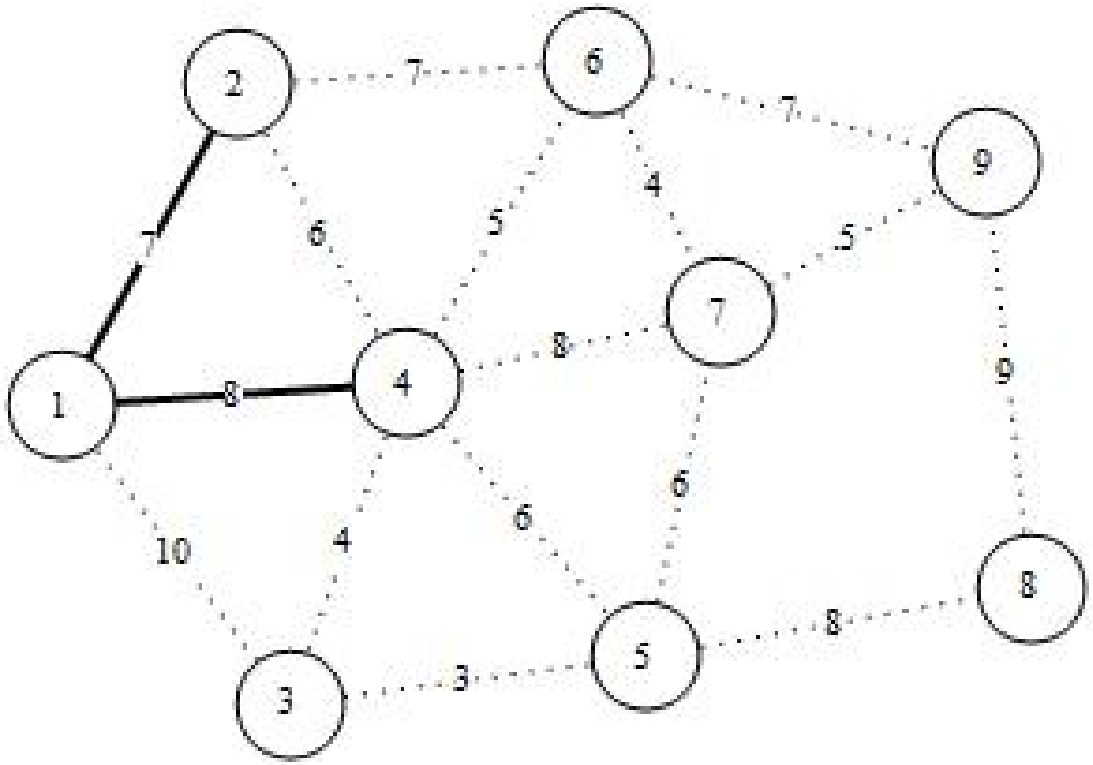
Background: Dijkstra's Algorithm in Action



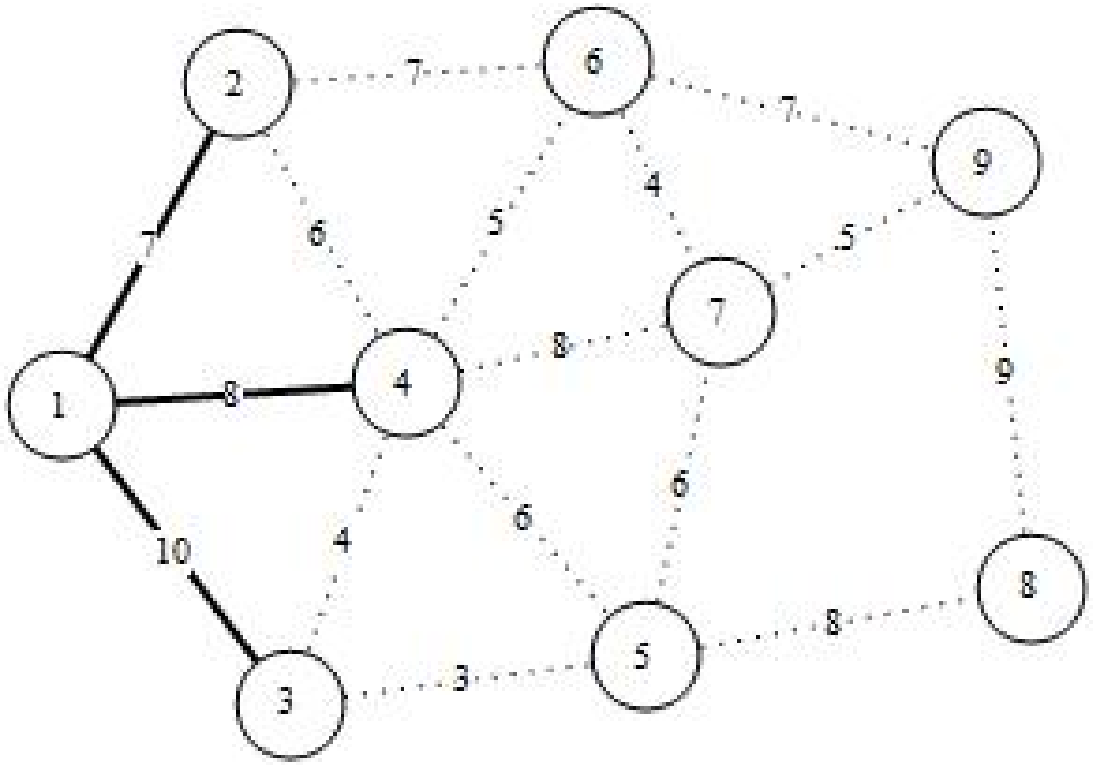
Background: Dijkstra's Algorithm in Action



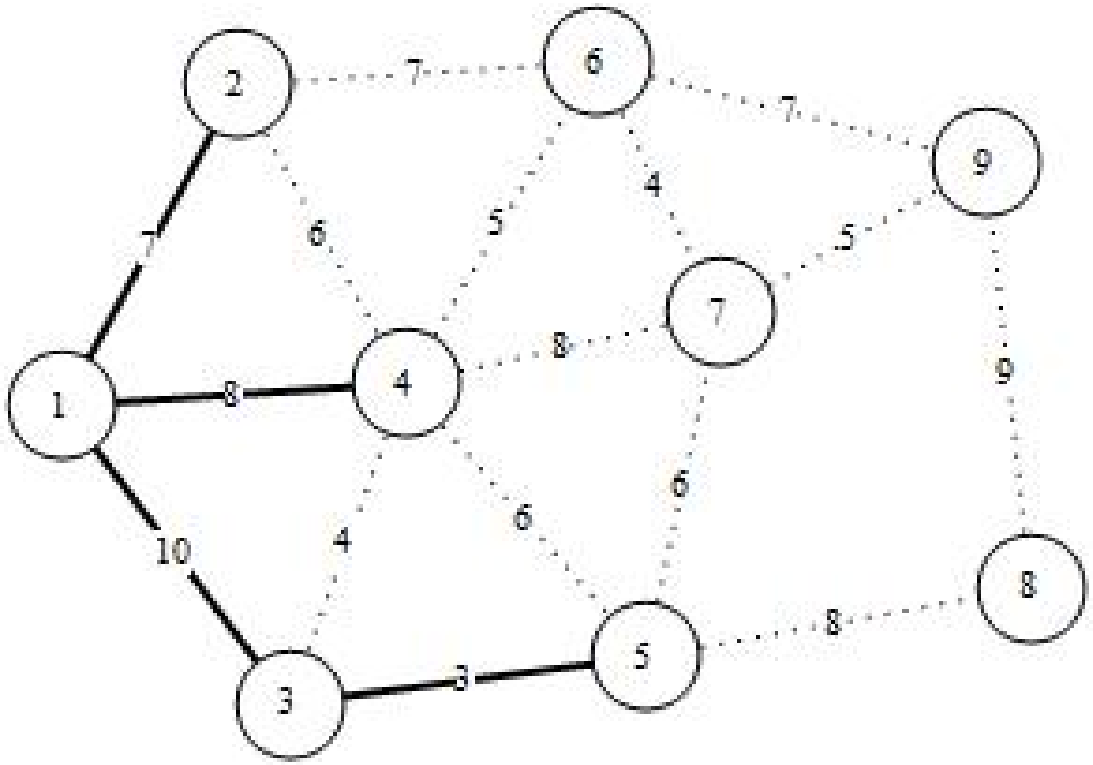
Background: Dijkstra's Algorithm in Action



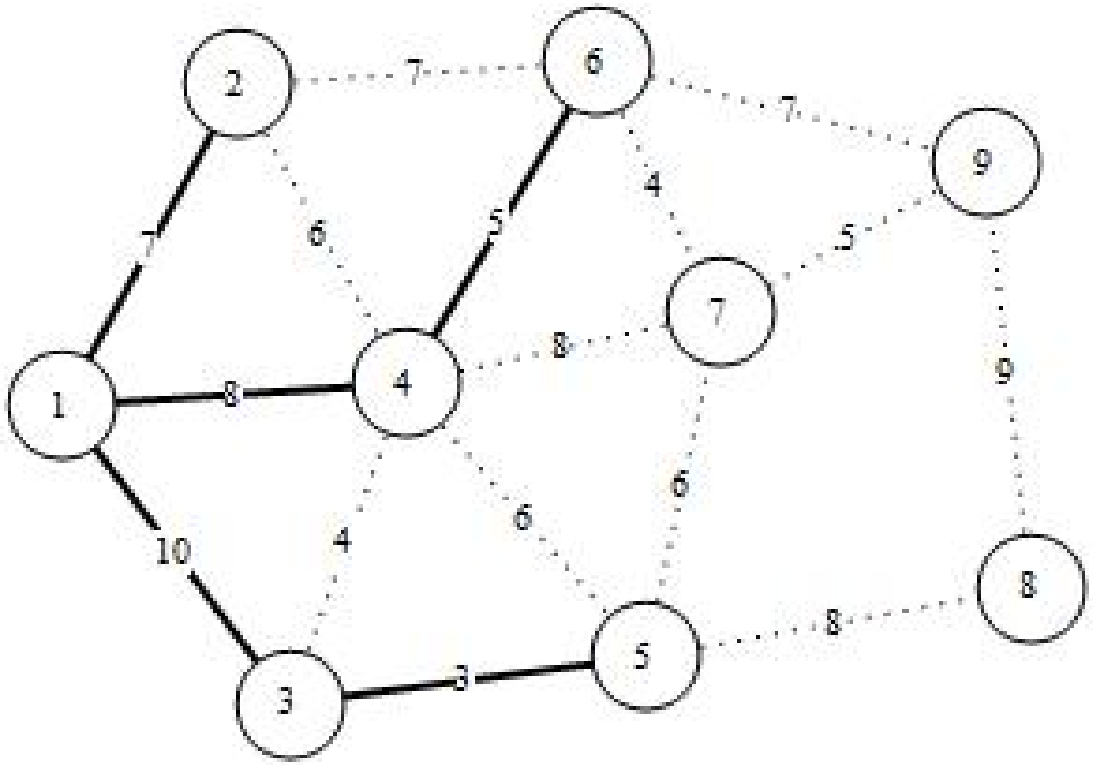
Background: Dijkstra's Algorithm in Action



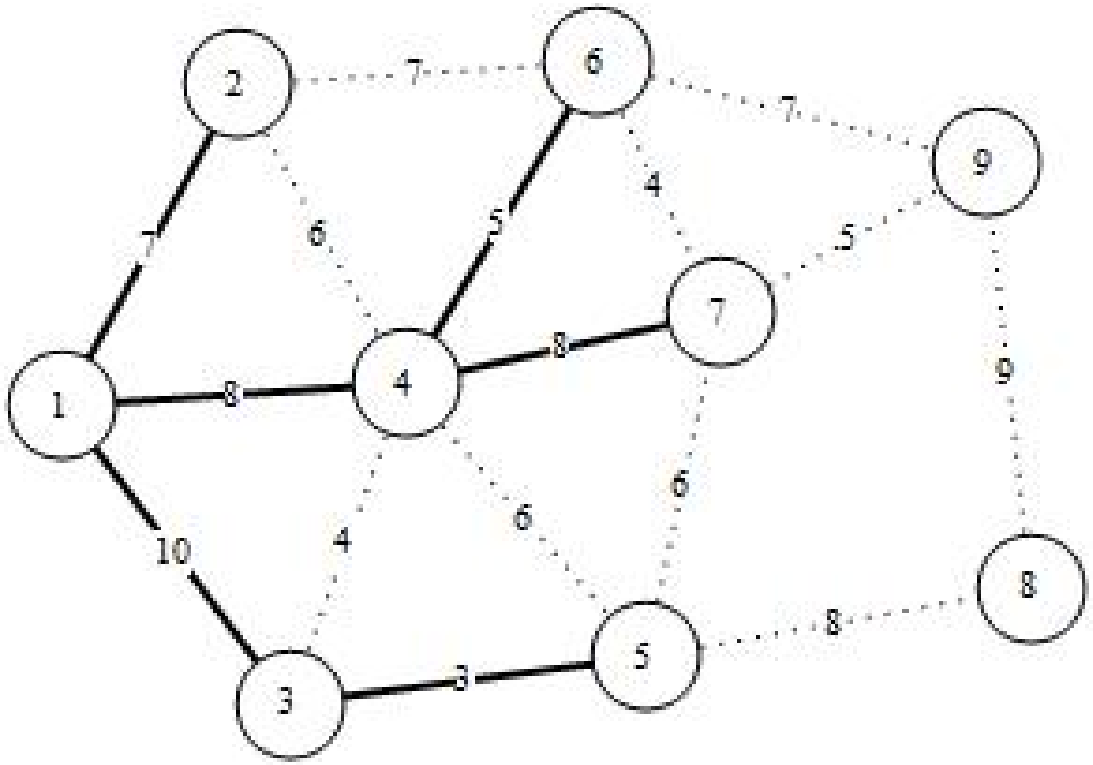
Background: Dijkstra's Algorithm in Action



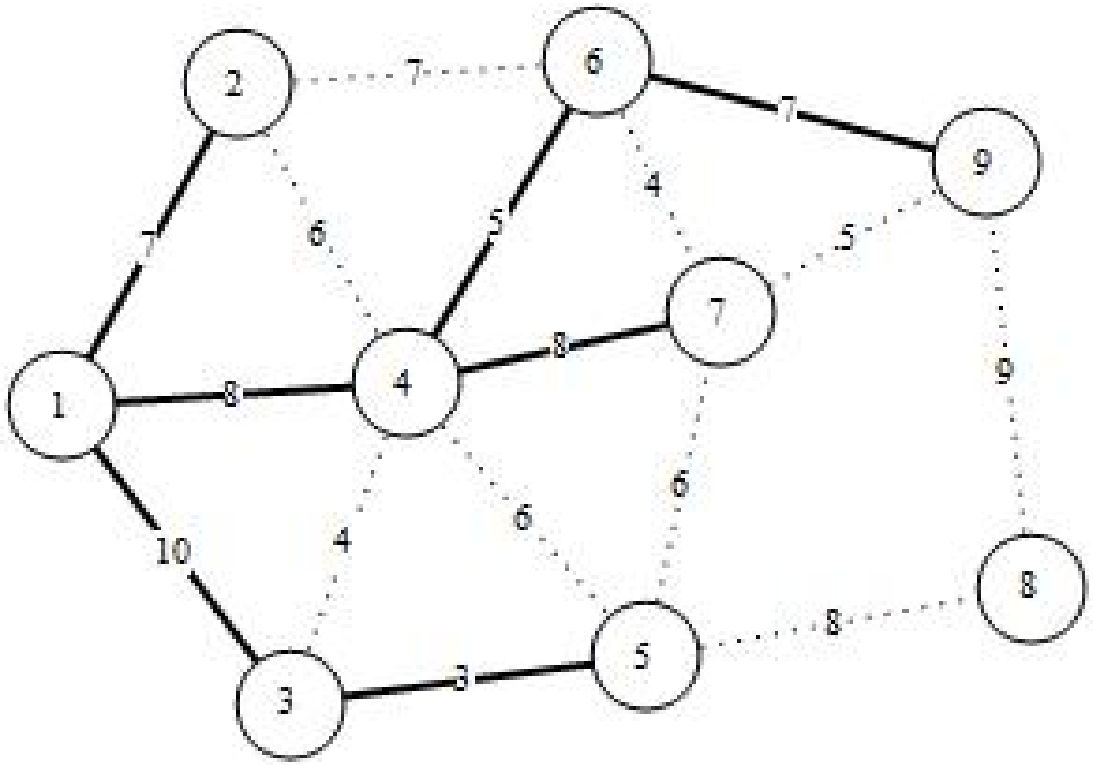
Background: Dijkstra's Algorithm in Action



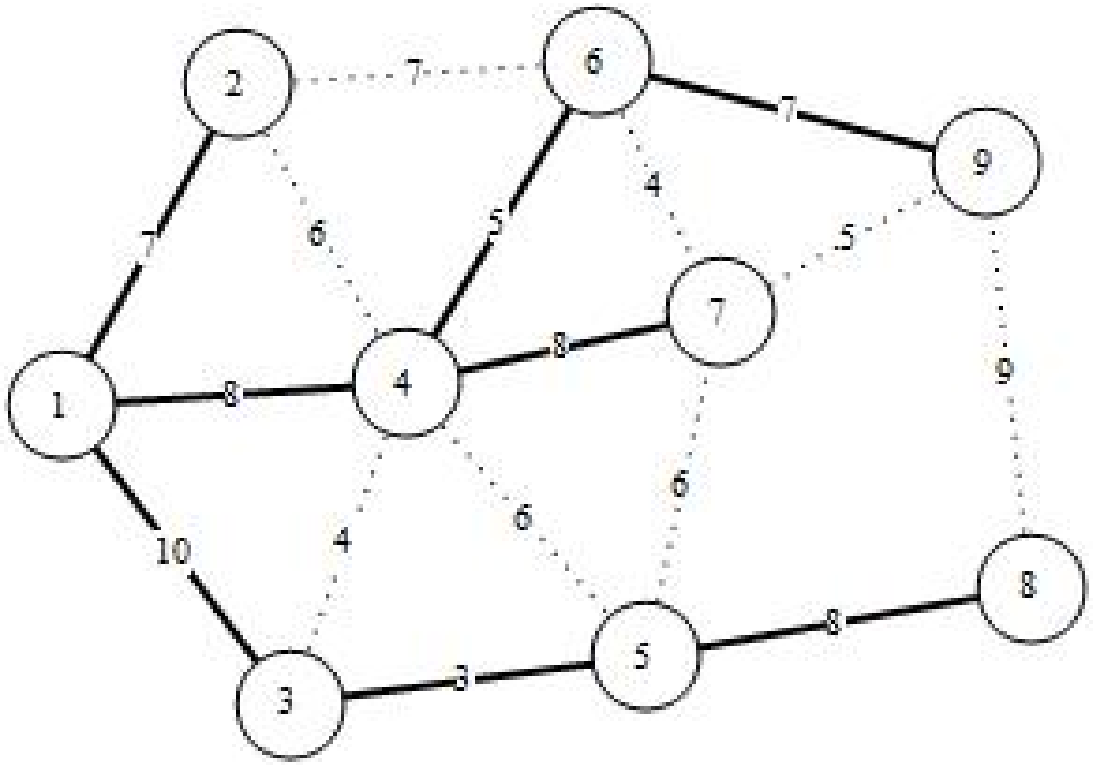
Background: Dijkstra's Algorithm in Action



Background: Dijkstra's Algorithm in Action



Background: Dijkstra's Algorithm in Action



Putting it Together: The Cable-Trench Problem

- Let $G = (V, E)$ be a weighted graph, $V = \{v_1, \ldots, v_n\}$, with root node v_1, and $|e|$ the weight of $e \in E$.
- $|T|_c$ is the total weight of the paths, $\mathcal{P}(v_1, v_j)$, from v_1, to each $v_j \in V$ of a spanning tree T of G: **total cable length**.
- $|T|_t$ is the total weight of T: **total trench length**.
- Let $\gamma \geq 0$ be the **per-unit cable cost** and
- $\tau \geq 0$ the **per-unit trench cost**.

Definition

The **CTP** is the problem of finding a spanning tree $T = (V, E_T)$ of G that minimizes the total cable and trench costs:

$$\gamma|T|_c + \tau|T|_t = \gamma \sum_{j=2}^{n} \sum_{e \in \mathcal{P}(v_1, v_j)} |e| + \tau \sum_{e \in E_T} |e| \ .$$

The Cable-Trench Problem in the Eyes of a Toddler

Further instructions: “Daddy, ...”

A Generalized Cable-Trench Problem

- What if the edge weights don't measure both the trench and cable weights, due to rocks, obstacles, overhead costs etc.?
- For $e \in E$, let $|e|_c$ be the **cable weight** of e and
- $|e|_t$ the **trench weight** of e.
- $|T|_c$ and $|T|_t$ are the total cable length and total trench length of the tree T.
- $\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.

Definition

The **Generalized Cable-Trench Problem** (GCTP) is the problem of finding a spanning tree T of G that minimizes

$$\gamma|T|_c + \tau|T|_t = \gamma \sum_{j=2}^{n} \sum_{e \in \mathcal{P}(v_1, v_j)} |e|_c + \tau \sum_{e \in E_T} |e|_t \ .$$

A Generalized Steiner Cable-Trench Problem

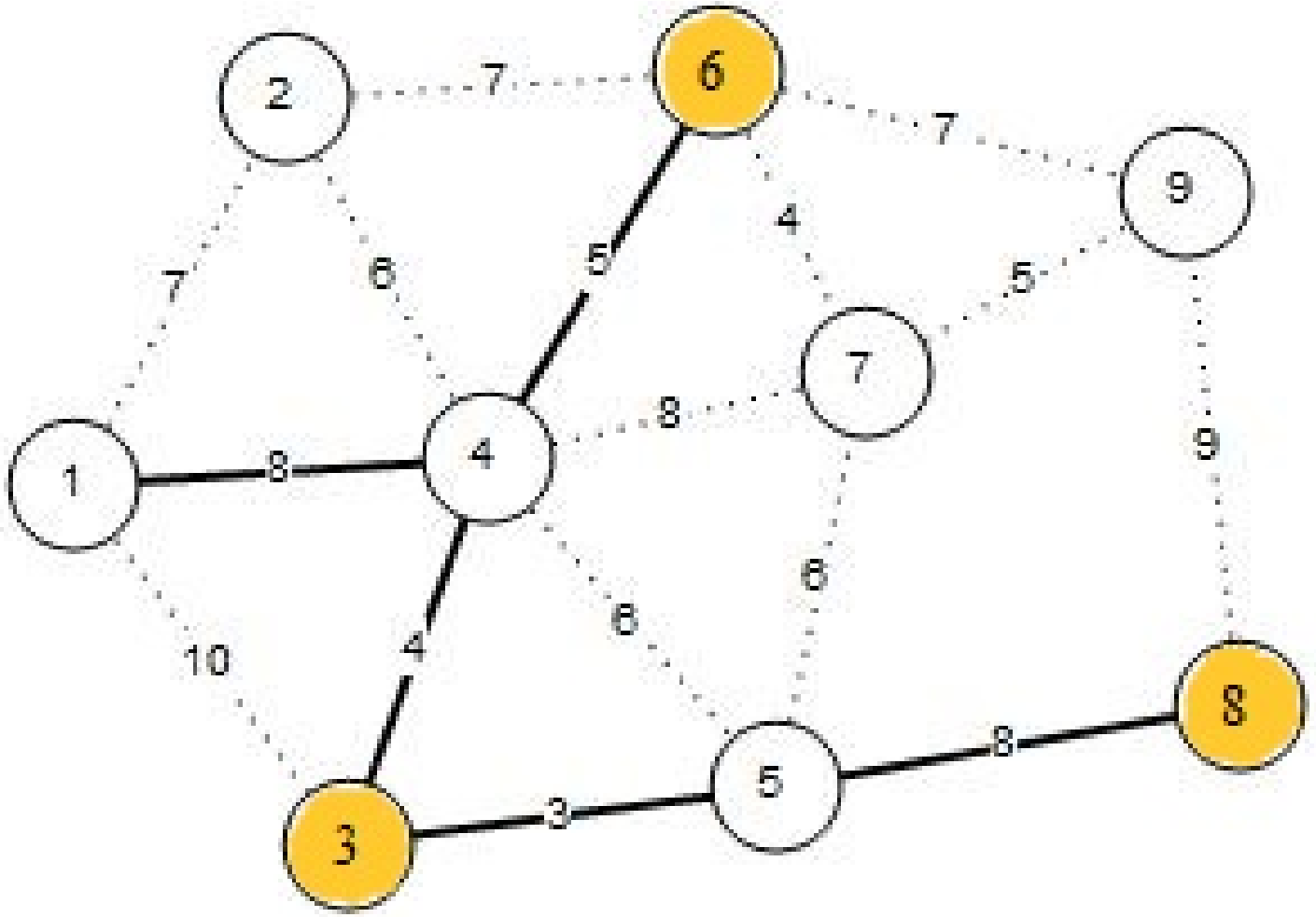
- What if we require the solution tree T to contain a given subset, $S \subseteq V$?
- S is called the set of **terminal nodes**,
- and $N = V \setminus (S \cup \{v_1\})$ the set of **nonterminal nodes**.
- $|T|_c$ and $|T|_t$ are the total cable length and total trench length of the tree T.
- $\gamma, \tau \geq 0$ are the per-unit cable and trench costs.

Definition

The **Generalized Steiner Cable-Trench Problem** (GSCTP) is the problem of finding a subtree $T = (V_T, E_T)$ of G such that $\{v_1\} \cup F \subseteq V_T$ that minimizes

$$\gamma|T|_c + \tau|T|_t = \gamma \sum_{v_j \in F} \sum_{e \in \mathcal{P}(v_1, v_j)} |e|_c + \tau \sum_{e \in E_T} |e|_t \ .$$

The Steiner CTP: An Example



Solving the Cable-Trench Problem

- Finding the MST with lowest total path length is NP-hard. (Eppstein)

Solving the Cable-Trench Problem

- Finding the MST with lowest total path length is NP-hard. (Eppstein)
- So the CTP and any generalization are NP-hard.

Solving the Cable-Trench Problem

- Finding the MST with lowest total path length is NP-hard. (Eppstein)
- So the CTP and any generalization are NP-hard.
- The obvious exceptions are the SPT ($\tau = 0$) and MST ($\gamma = 0$).

Solving the Cable-Trench Problem

- Finding the MST with lowest total path length is NP-hard. (Eppstein)
- So the CTP and any generalization are NP-hard.
- The obvious exceptions are the SPT ($\tau = 0$) and MST ($\gamma = 0$).
- (The Sweat): In general, we could check all spanning trees and find an optimal solution.

Must go faster. Must go faster.

Solution Approaches

- Prim's and Dijkstra's algorithms are very similar. Can they be combined to tackle the CTP?

Solution Approaches

- Prim's and Dijkstra's algorithms are very similar. Can they be combined to tackle the CTP?
- Yes!

Solution Approaches

- Prim's and Dijkstra's algorithms are very similar. Can they be combined to tackle the CTP?
- Yes!
- Does this combined algorithm always give you the tree with lowest total cable and trench cost (optimal solution)?

Solution Approaches

- Prim's and Dijkstra's algorithms are very similar. Can they be combined to tackle the CTP?
- Yes!
- Does this combined algorithm always give you the tree with lowest total cable and trench cost (optimal solution)?
- If so, we'll be rich!

Solution Approaches

- Prim's and Dijkstra's algorithms are very similar. Can they be combined to tackle the CTP?
- Yes!
- Does this combined algorithm always give you the tree with lowest total cable and trench cost (optimal solution)?
- If so, we'll be rich!
- Let's find out.

Modified Prim's Algorithm for the CTP (ModPrim)

- Combines Dijkstra's (SPT) and Prim's (MST) algorithms.
- Let $G = (V, E)$ and initialize $T = (\{v_1\}, \{\})$.
- Let v_t represent a vertex in T.
- Iteratively add the vertex $v_k \in V$ and edge $e_k = (v_t, v_k) \in E$ to T so that the added cost, $d(v_k, v_t)$, of
  - "digging the trench" from vertex v_t to v_k plus
  - "laying the cable" from the root node v_1 to v_k is minimized.
- Cost function:

$$d(v_k, v_t) = \text{ cable cost } + \text{ trench cost}$$

$$d(v_k, v_t) = \gamma \sum_{e \in \mathcal{P}(v_1, v_k)} |e| + \tau |e|$$

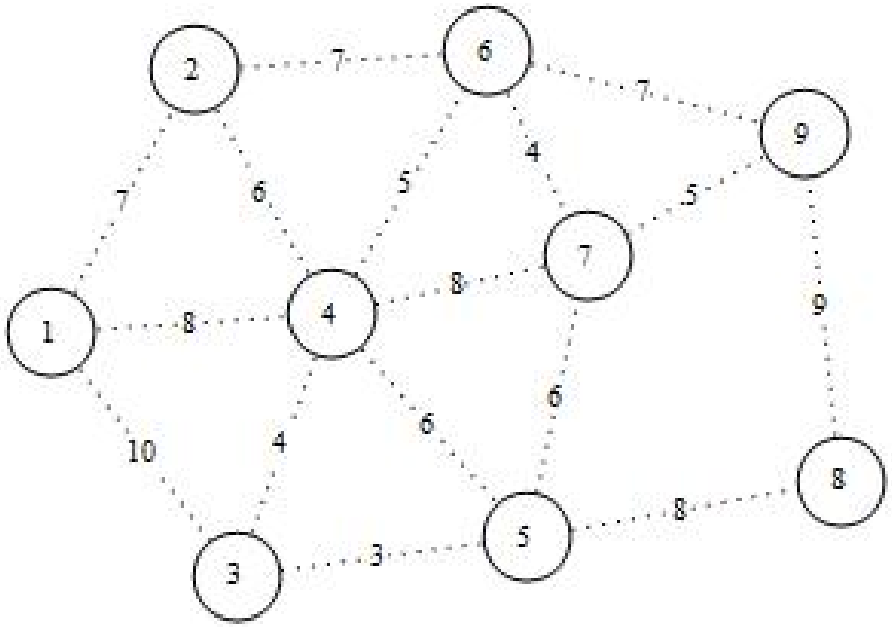
- At worst ($G = K_n$): Quadratic running time, i.e. $O\left(n^2\right)$.

Variations of ModPrim

- All variations require quadratic time.

1. Modify the cost function to encourage the inclusion of terminal nodes and Steiner nodes adjacent to terminal nodes.
   - Why? Certain Steiner nodes are good branching-off points to reach multiple terminal nodes.
2. Semi-Greedy approach: Insert the j^{th} best edge on the first step, $j > 1$, and proceed greedily.
3. Partially Stochastic approach: Insert the first several edges randomly and then proceed in a greedy manner.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

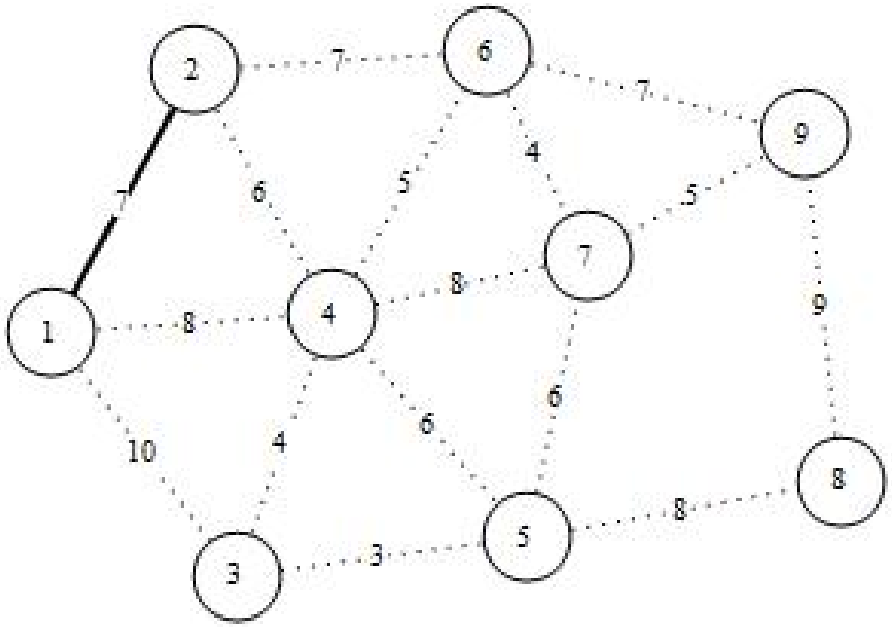


Total weight: 0

$$
\begin{aligned}
d(2,1) &= 7\gamma + 7\tau \\
&= 7 + 35 = 42 \\
d(3,1) &= 10\gamma + 10\tau \\
&= 10 + 50 = 60 \\
d(4,1) &= 8\gamma + 8\tau \\
&= 8 + 40 = 48
\end{aligned}
$$

Include edge $(1, 2)$.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

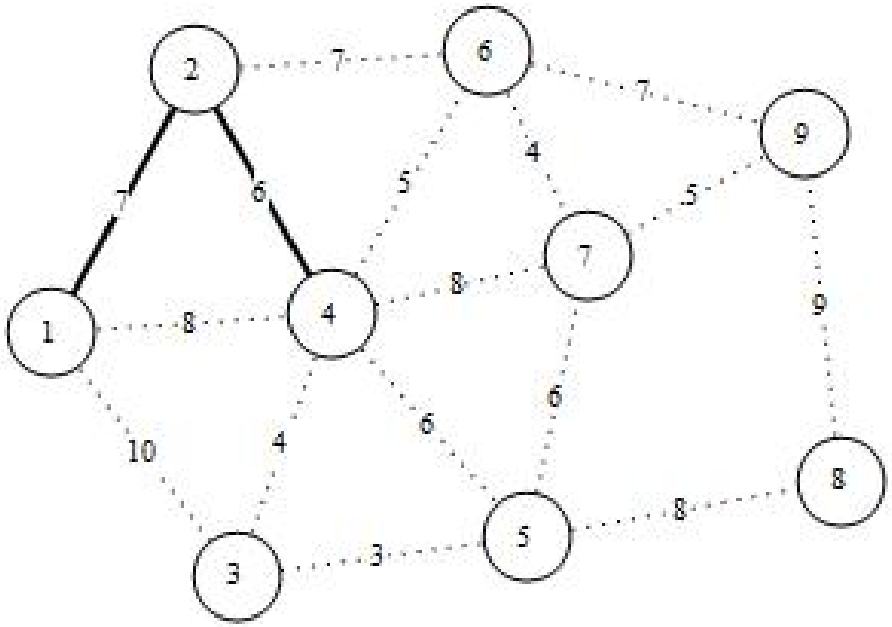


Total weight: $0 + 42 = 42$

$$
\begin{aligned}
d(3,1) &= 60 \\
d(4,1) &= 48 \\
d(4,2) &= 13\gamma + 6\tau \\
&= 13 + 30 = 43 \\
d(6,2) &= 14\gamma + 7\tau \\
&= 14 + 35 = 49
\end{aligned}
$$

Include edge $(1, 4)$.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

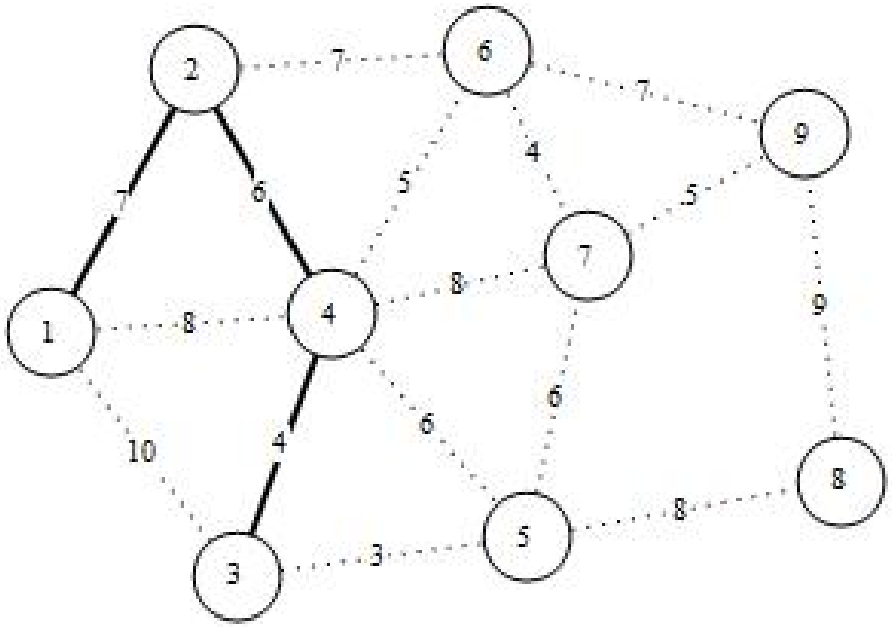


$$d(3,1) = 60$$
$$d(3,4) = 17\gamma + 4\tau$$
$$= 17 + 20 = 37$$
$$d(5,4) = 49$$
$$d(6,2) = 49$$
$$d(6,4) = 43$$
$$d(7,4) = 61$$

Total weight: $42 + 43 = 85$

Include edge $(3,4)$.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

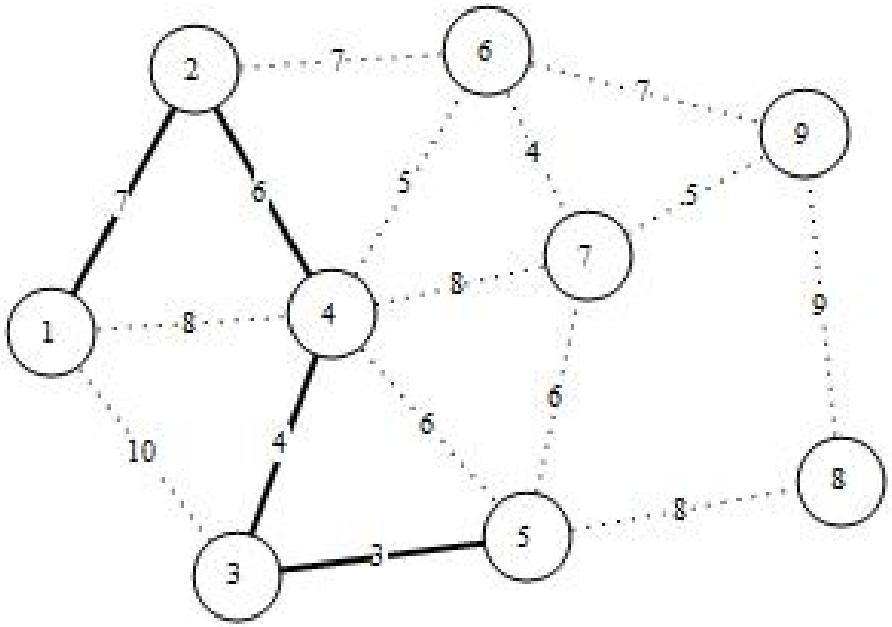


$$d(5,3) = 20\gamma + 3\tau$$
$$= 20 + 15 = 35$$
$$d(5,4) = 49$$
$$d(6,4) = 43$$
$$d(7,4) = 61$$

Include edge $(3,5)$.

Total weight: $85 + 37 = 122$

ModPrim Example with $\gamma = 1$ and $\tau = 5$

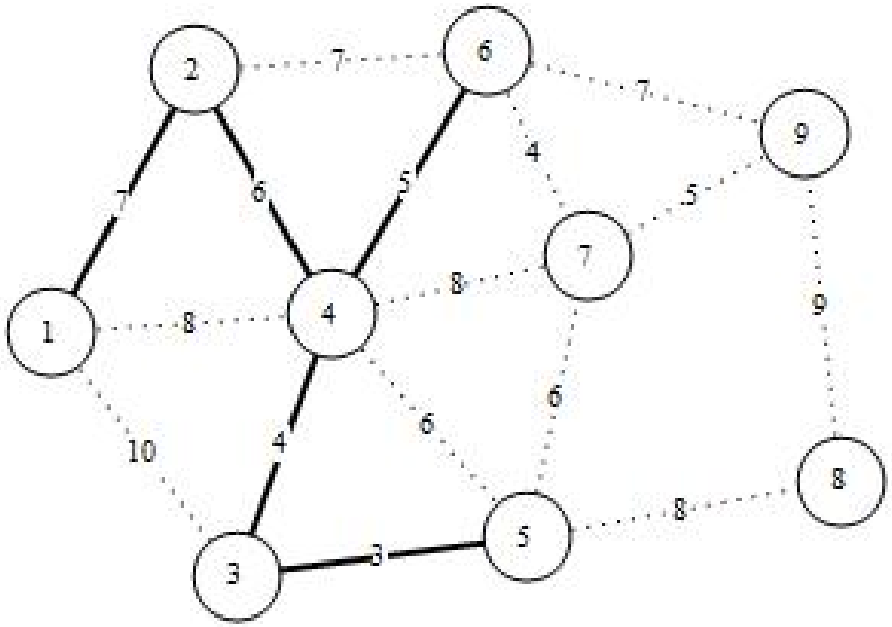


Total weight: $122 + 35 = 157$

$$
\begin{aligned}
d(6,4) &= 43 \\
d(7,4) &= 61 \\
d(7,5) &= 26\gamma + 6\tau \\
&= 26 + 30 = 56 \\
d(8,5) &= 28\gamma + 8\tau \\
&= 28 + 40 = 68
\end{aligned}
$$

Include edge $(4, 6)$.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

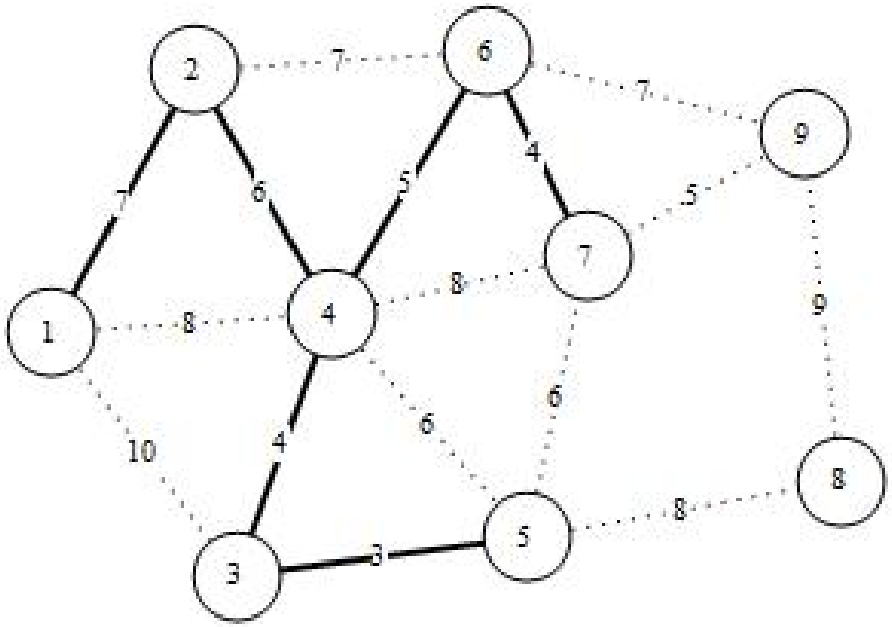


Total weight: $157 + 43 = 200$

$$
\begin{aligned}
d(7,5) &= 56 \\
d(7,6) &= 22\gamma + 4\tau \\
&= 22 + 20 = 42 \\
d(8,5) &= 68 \\
d(9,6) &= 25\gamma + 7\tau \\
&= 25 + 35 = 60
\end{aligned}
$$

Include edge $(6,7)$.

ModPrim Example with $\gamma = 1$ and $\tau = 5$

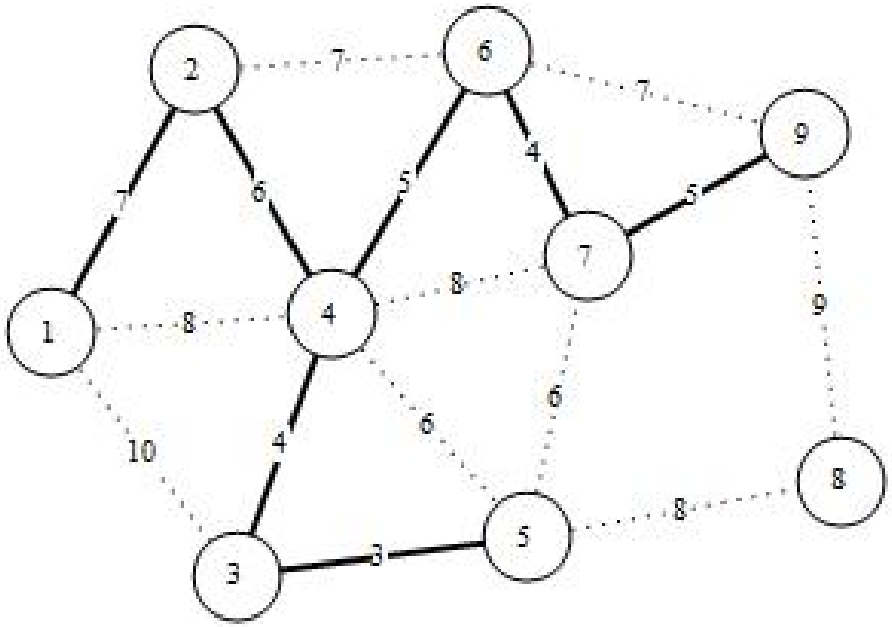


$$d(8,5) = 68$$
$$d(9,6) = 60$$
$$d(9,7) = 27\gamma + 5\tau$$
$$= 27 + 25 = 52$$

Include edge $(7,9)$.

Total weight: $200 + 42 = 242$

ModPrim Example with $\gamma = 1$ and $\tau = 5$

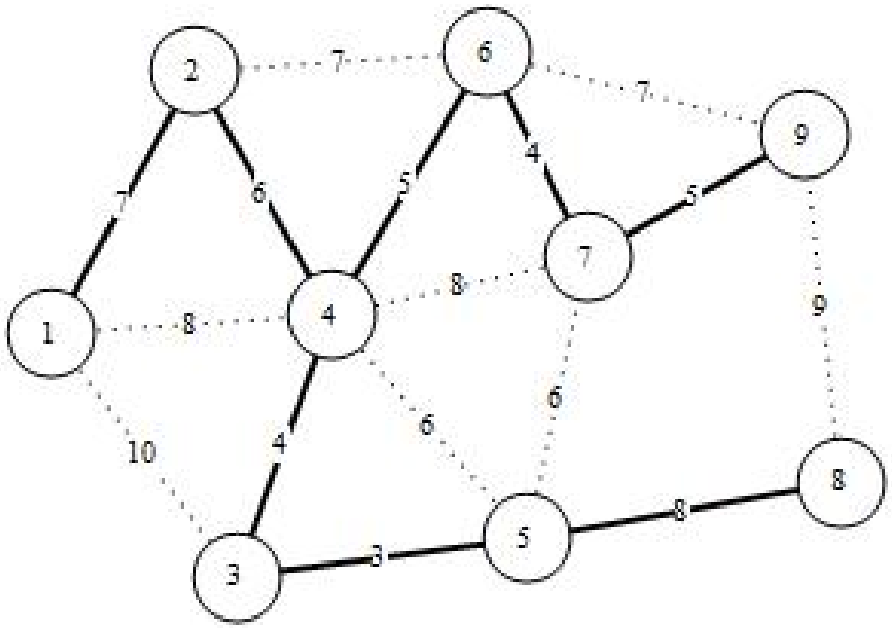


$$d(8,5) = 68$$
$$d(8,9) = 36\gamma + 9\tau$$
$$= 36 + 45 = 81$$

Include edge $(5, 8)$.

Total weight: $242 + 52 = 294$

ModPrim Example with $\gamma = 1$ and $\tau = 5$



Total weight: $294 + 68 = 362$

(The Tears:) *
Optimal Solution: 337
* ModPrim: 362
* Off by about 7.4%.
* If we had picked $(2, 4)$ on the second step, then we would have found the optimal solution.
* True cable costs are hard to predict.

CTP Results on the 9-vertex graph example

τ	Optimum	ModPrim	% Dev.
0.1	113.6	119.7	5.4
0.5	135	141.5	4.8
4	293	322	9.9
10	554	572	3.2
20	984	992	0.8
30	1412	1412	0

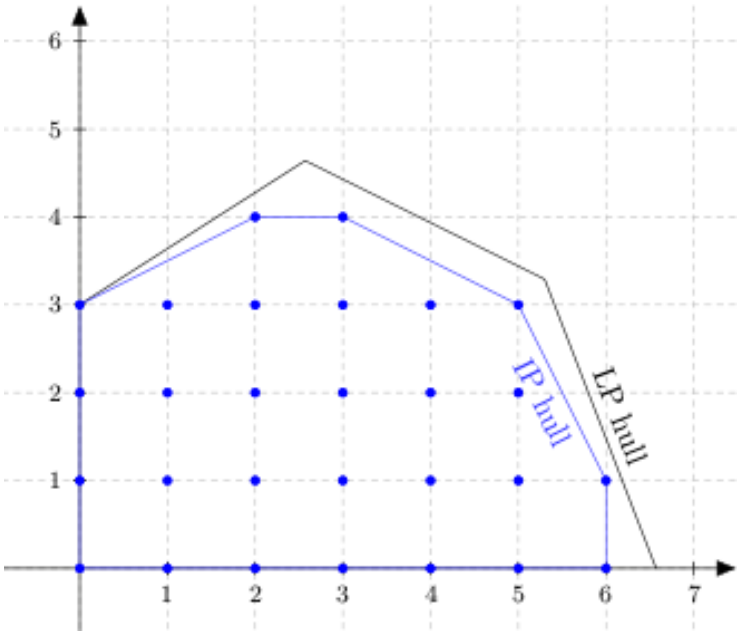
Moral of the story thus far

It may not give you the optimal solution, but at least ModPrim is fast.

"I like fast algorithms. They're kind of like sports cars for nerds." (Nate Wentzel)

Other Solution Approaches

- The CTP can be formulated as an integer linear program.
- The cost function is a linear function.
- How hard can it be to minimize a linear function?



An Integer Programming Formulation of the CTP

Minimize:

$$\gamma \sum_{j=1}^{n} \sum_{i=1}^{n} |(v_i, v_j)| x_{ij} + \tau \sum_{j=1}^{n} \sum_{i=1}^{n} |(v_i, v_j)| y_{ij}$$

subject to the constraints:

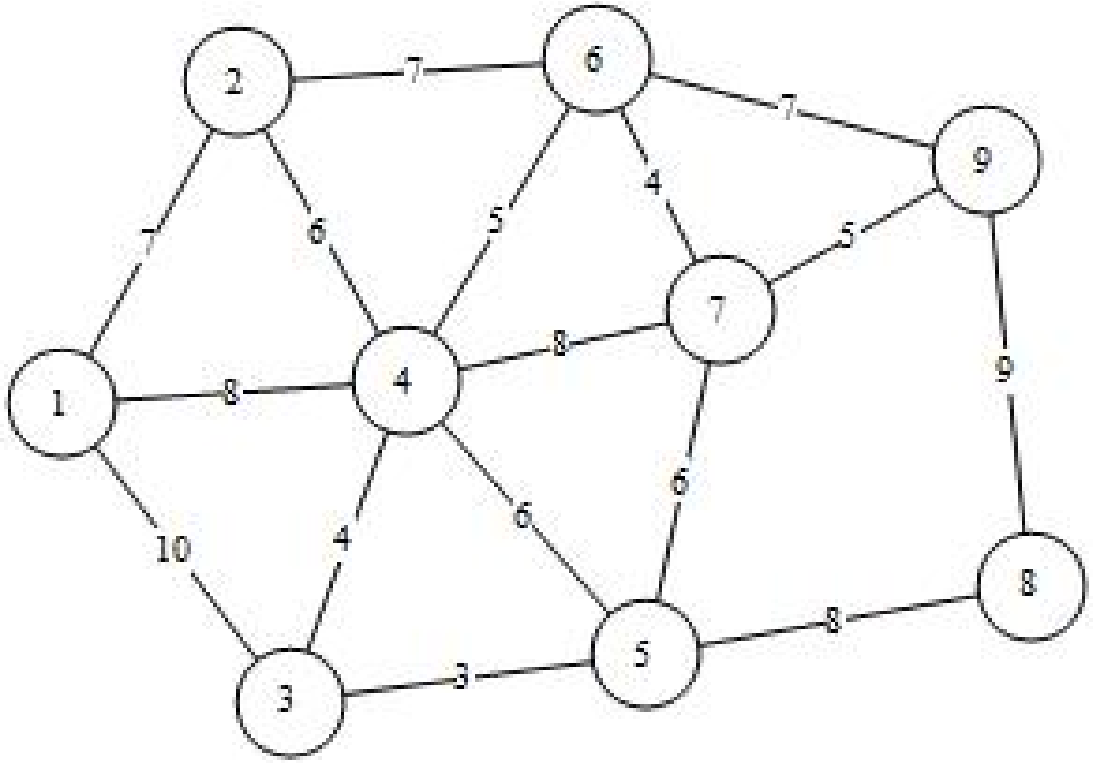
$$\sum_{j=2}^{n} x_{1j} = n - 1 \qquad \sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = -1$$

$$\sum_{j=2}^{n} \sum_{i=1}^{j-1} y_{ij} = n - 1 \qquad (n-1) y_{ij} - x_{ij} - x_{ji} \geq 0$$

$x_{ij} \geq 0 : x_{ij} = \#$ cables in trench (v_i, v_j).
$y_{ij} \in \{0, 1\}$: Dig trench (v_i, v_j)? No: $y_{ij} = 0$, Yes: $y_{ij} = 1$.

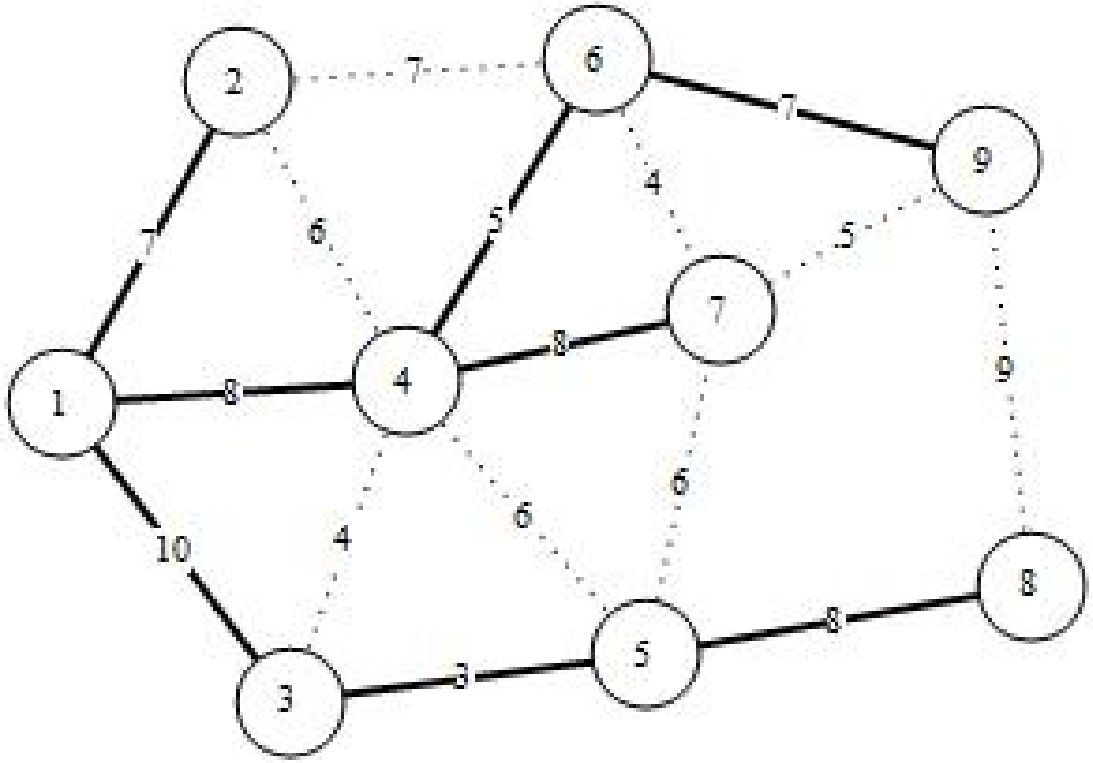
Optimal CTP Solutions Using the Linear Program

What are the optimal CTP solutions?



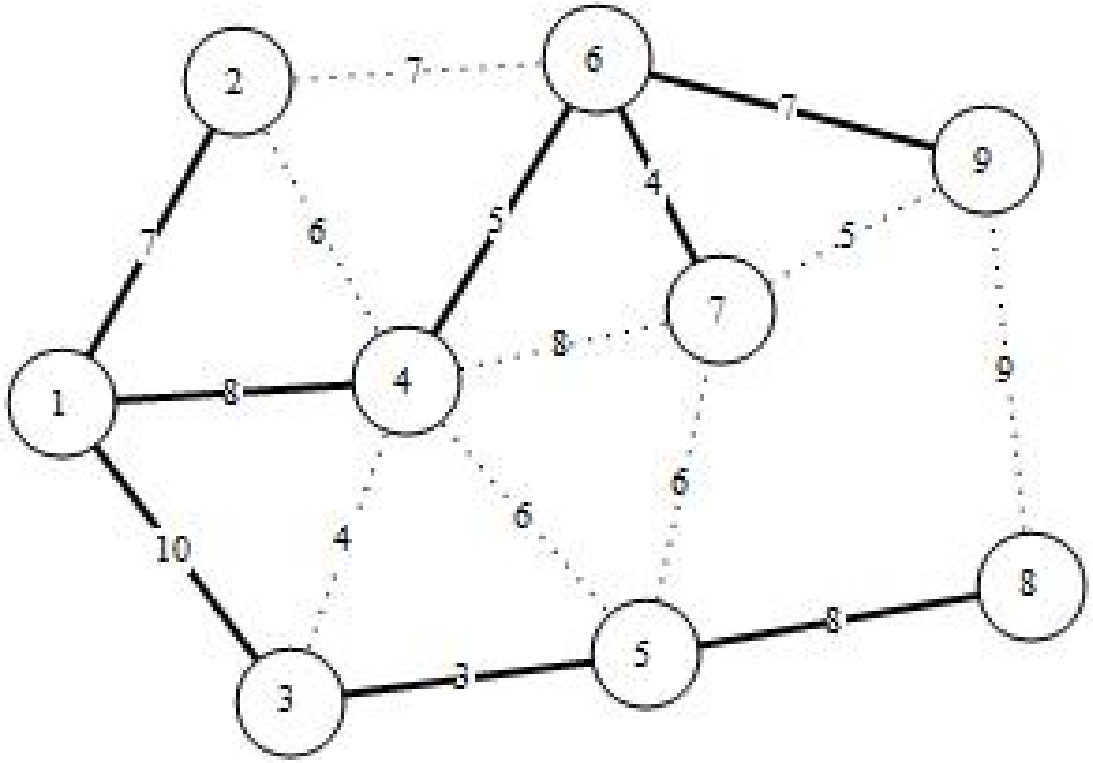
T_1: for $0 \leq \tau/\gamma \leq 1/4$; Cost: $56\tau + 108\gamma$ (SPT)

$\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.



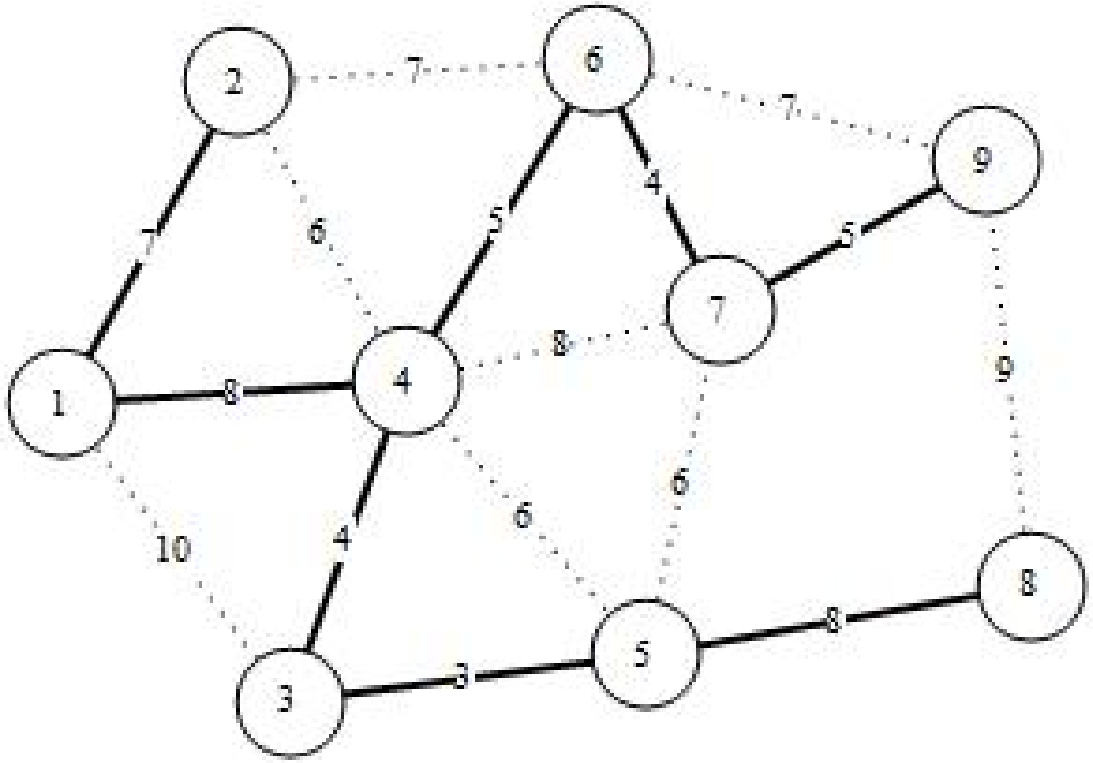
T_2: for $1/4 \leq \tau/\gamma \leq 1$; Cost: $52\tau + 109\gamma$

$\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.



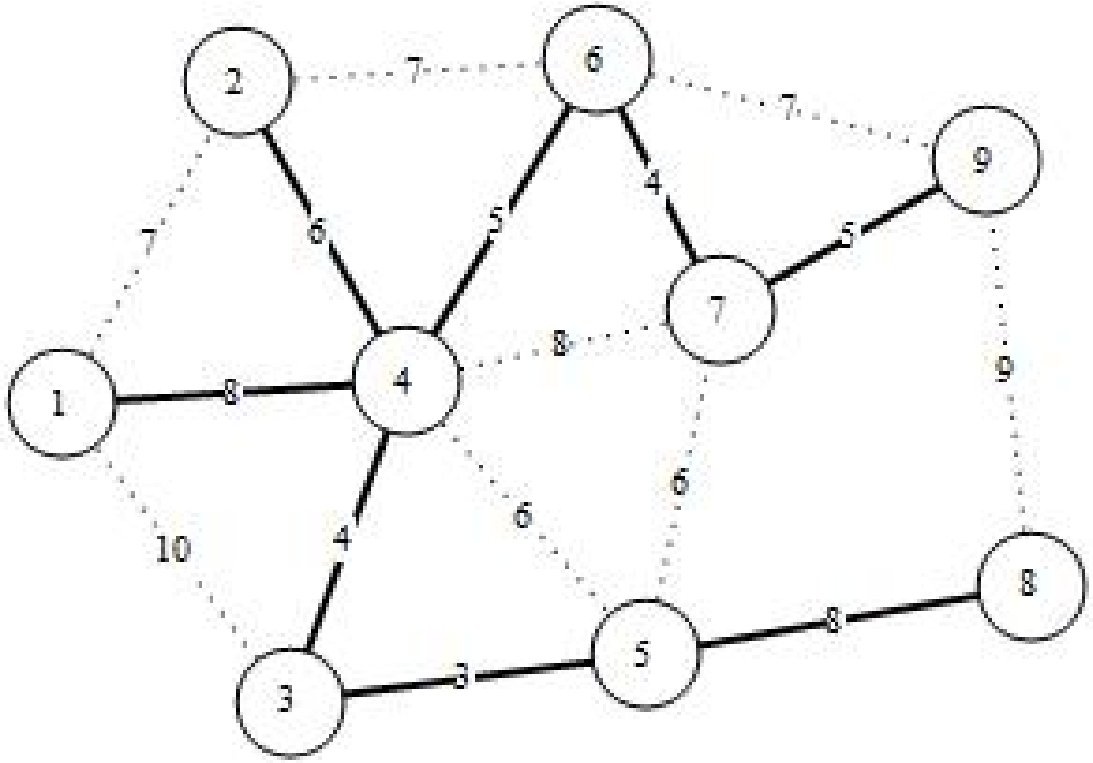
T_3: for $1 \leq \tau/\gamma \leq 7$; Cost: $44\tau + 117\gamma$

$\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.



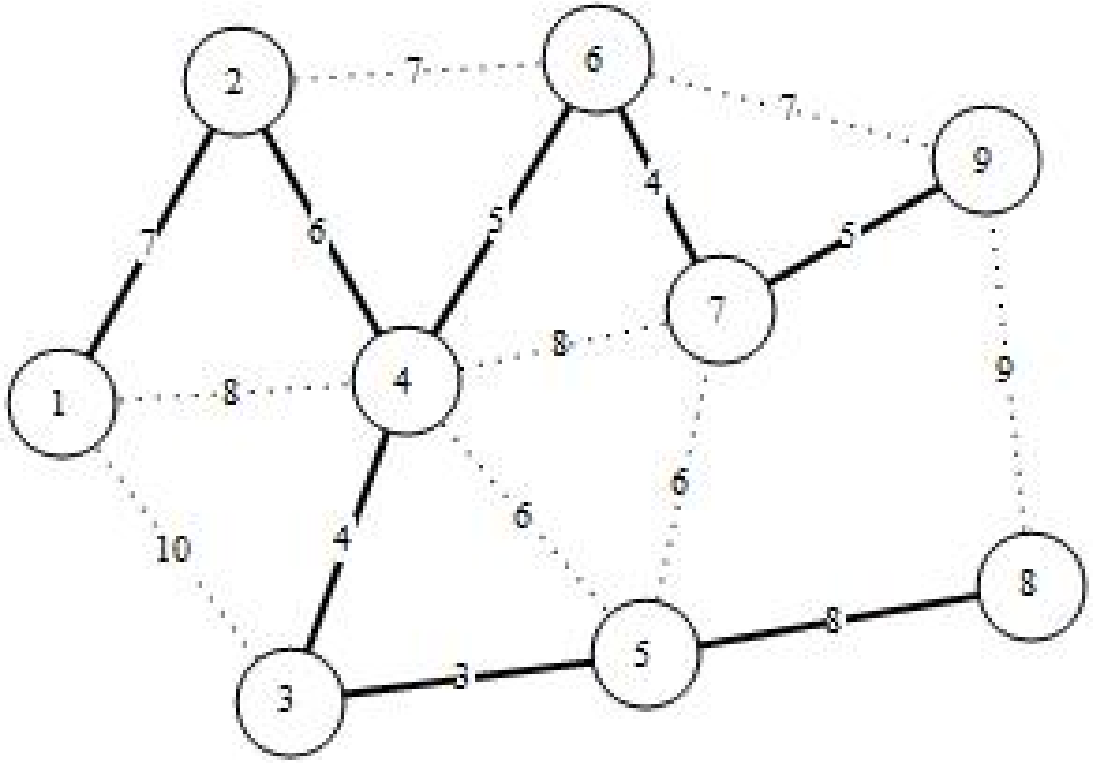
T_4: for $7 \leq \tau/\gamma \leq 28$; Cost: $43\tau + 124\gamma$

$\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.



T_5: for $28 \leq \tau/\gamma$; Cost: $42\tau + 152\gamma$ (MST)

$\gamma, \tau \geq 0$ are the per-unit cable and trench costs, respectively.



Multicommodity Flow Formulation

Minimize:

$$Z = \gamma \sum_{k=2}^{n} \sum_{j=2}^{n} \sum_{\substack{i=1 \\ i \neq j,k}}^{n} s_{ij} x_{ijk} + \tau \sum_{j=2}^{n} \sum_{i=1}^{j-1} t_{ij} y_{ij},$$

subject to the constraints:

$$\sum_{j=2}^{n} x_{1jk} = 1 \qquad \sum_{i=1}^{n} x_{ikk} = 1 \qquad (1)$$

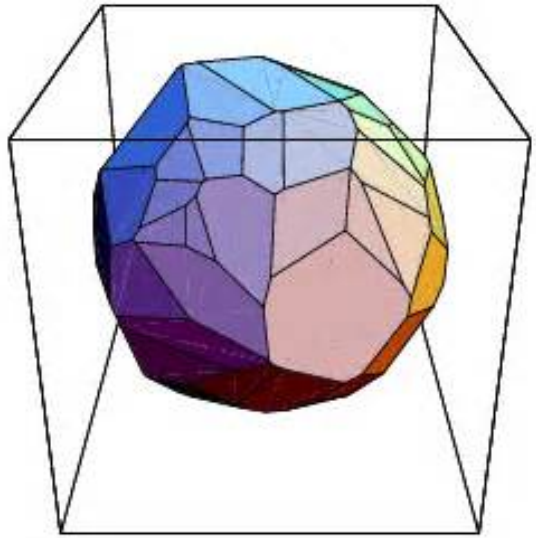
$$\sum_{k=2}^{n} (x_{ijk} - x_{jik}) = 1 \qquad \sum_{j=2}^{n} \sum_{i=1}^{j-1} y_{ij} = n - 1 \qquad (2)$$

$$x_{ijk} + x_{jik} \leq y_{ij} \qquad x_{ijk}, y_{ij} \in \{0, 1\} \qquad (3)$$

X_{ijk} is the number of cables (0 or 1) in trench (v_i, v_j) with destination v_k.

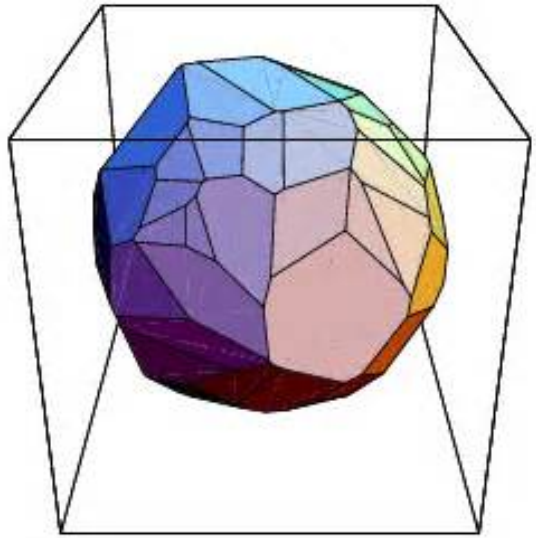
Seriously, it's a Linear Function

- How can optimizing a linear function be NP-hard?



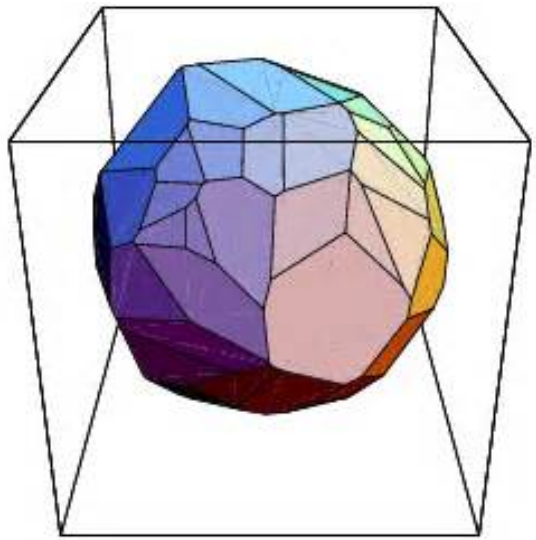
Seriously, it's a Linear Function

- How can optimizing a linear function be NP-hard?
- Yes, it's a linear function, but it's discrete,



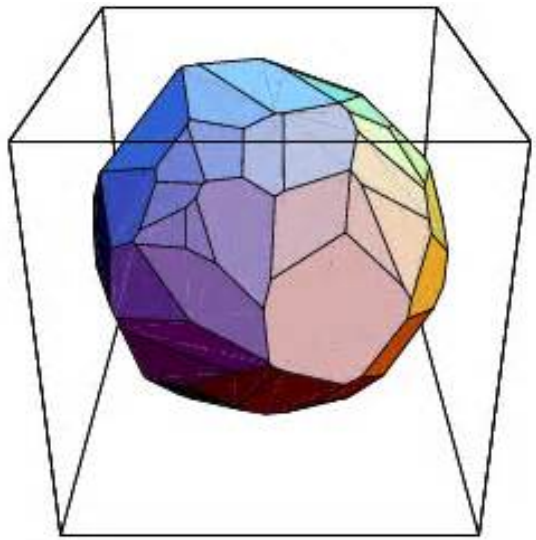
Seriously, it's a Linear Function

- How can optimizing a linear function be NP-hard?
- Yes, it's a linear function, but it's discrete,
- and the domain is a convex $O(|E|)$-dimensional polytope.



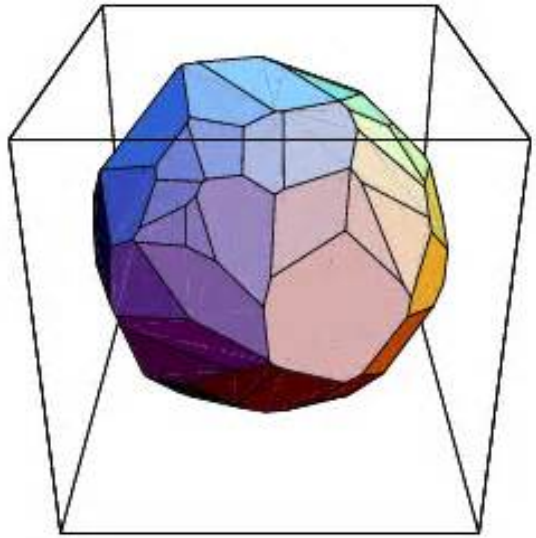
Seriously, it's a Linear Function

- How can optimizing a linear function be NP-hard?
- Yes, it's a linear function, but it's discrete,
- and the domain is a convex $O(|E|)$-dimensional polytope.
- That's a lot of boundaries to check.



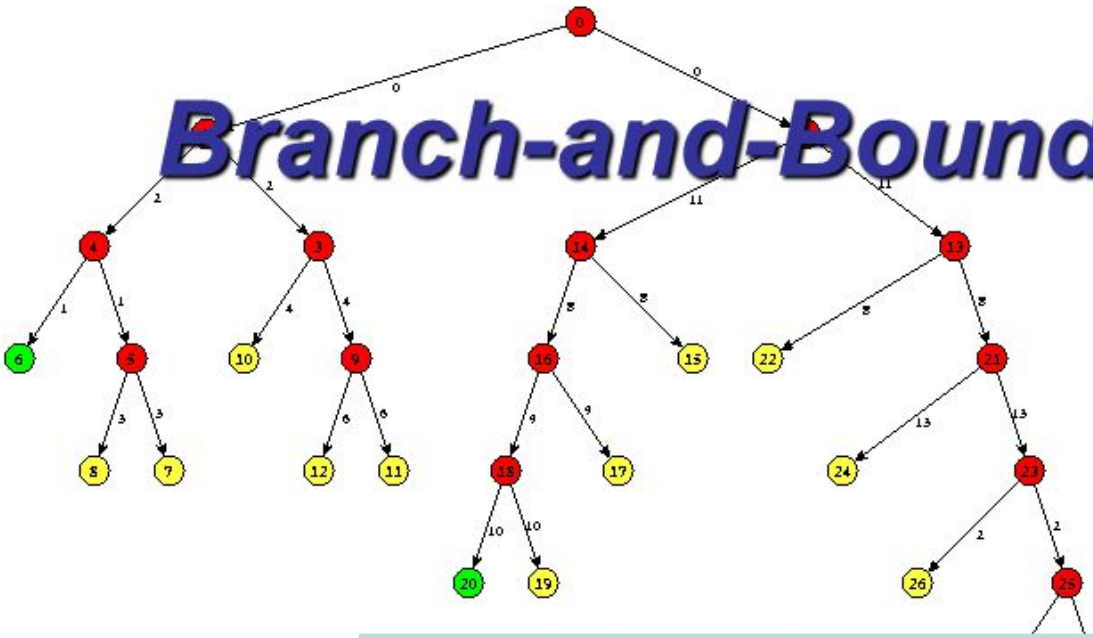
Seriously, it's a Linear Function

- How can optimizing a linear function be NP-hard?
- Yes, it's a linear function, but it's discrete,
- and the domain is a convex $O(|E|)$-dimensional polytope.
- That's a lot of boundaries to check.
- Limitation: Problems can kill memory with only a few hundred vertices.



Just Relax!

- Linear Programming (LP) Relaxation
- Relax the restriction on the integers, and you can quickly find a lower bound on the optimal solution.
- $0 \leq y_{ij} \leq 1$, $0 \leq x_{ij} \in \mathbb{R}$
- If this solution says you need a fraction of a trench dug, try **branch and bound** techniques: decide to dig the trench or not and solve again.
- Another approach is to make **cuts** to simplify the boundary.



1. Solve the original problem using linear programming. If the answer satisfies the integer constraints, we are done.
   If not, this value provides an initial *upper bound* for the objective function.

2. Find any feasible solution that meets the integer constraints for use as a *lower bound*. Usually, rounding down each variable will accomplish this.

Lagrange wants to relax too.

- Lagrangean relaxation.
- Lift a constraint, with a Lagrange multiplier, into the objective fuction.

Minimize:

$$\gamma \sum_{j=1}^{n} \sum_{i=1}^{n} |(v_i, v_j)| x_{ij} + \tau \sum_{j=1}^{n} \sum_{i=1}^{n} |(v_i, v_j)| y_{ij}$$
$$+ \sum_{j=1}^{n} \sum_{i=1}^{j-1} \lambda_{ij} \left(x_{ij} + x_{ji} - (n-1) y_{ij} \right)$$

subject to the constraints:

$$\sum_{j=2}^{n} x_{1j} = n - 1 \qquad \sum_{j=1}^{n} x_{ij} - \sum_{k=1}^{n} x_{ki} = -1$$

$$(n-1) y_{ij} - x_{ij} - x_{ji} \geq 0 \qquad x_{ij} \geq 0, y_{ij} \in \{0, 1\}$$

Applications of the CTP and its Generalizations

- Cable layouts for an array of radio telescopes
- Digitizing micro-CT images of a blood vessel network.
- Car pooling.
- Construction of logging roads and sawmills. (Marianov et al.)
- Construction of irrigation canals and wells. (Marianov et al.)
- Layout of wireless and wired access networks. (Nielsen et al.)
- Laying cables and pipes (of different per-unit costs) in a given trench. (Schwarze)

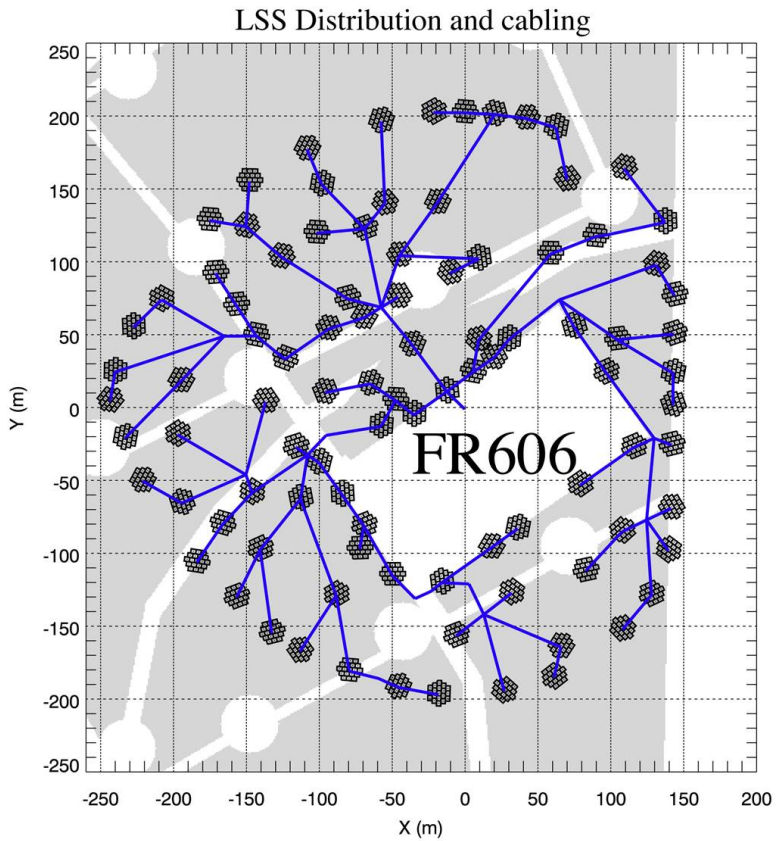
Trivial Application: Radio Telescope Arrays

- LOFAR: Low Frequency Array - The world's largest connected radio telescope, spread out over individual stations in western Europe.
- In Nançay, France, there is a proposed extension of an existing installation of radio telescope arrays.
- 96 mini arrays (of 19 dipole antennas each) will be spread out over a 400×400-meter area.

Overview of one mini-array ↑

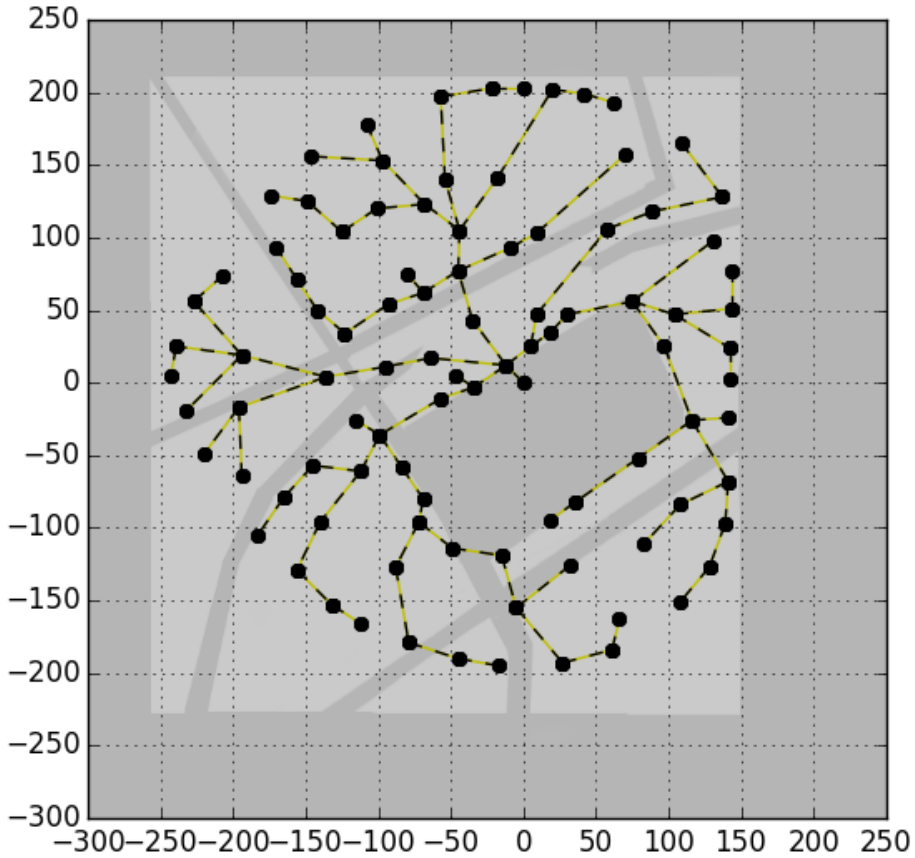
Trivial Application: Radio Telescope Arrays

- White areas represent hard or soft obstactes.
- Trench costs increase when crossing existing cabled areas.

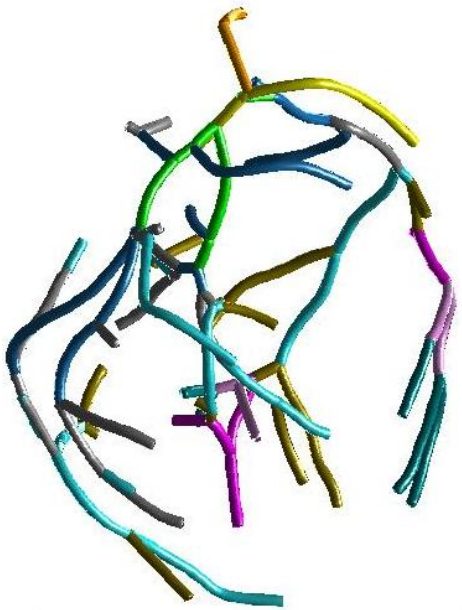


Trivial Application: Radio Telescope Arrays

- A plausible solution: (Zyma, 2015)



A Nontrivial Application: Analyzing Micro-CT Scans



- (Blood)
- Given: Data from a Micro-CT scan (combined microscope and CT scan).
- Digitally reconstruct a blood vessel network (vasculature) from a set of points and vessel radii and
- Automatically filter out errors,
- Accurately and efficiently.
- Study vasculature noninvasively for cancer detection, identifying blood clots, etc.

The Problem: Vascular Image Analysis from a CT Scan

- Example: This scan of a mouse's leg has 27,873 vertices (3D).
- The red dot represents the main artery coming into the leg.
- Can we accurately and efficiently connect the dots to visualize the blood vessel network?

Vascular Image Analysis and Murray's Principle (1926)

- Good solution (previous state-of-the-art): Find the MST of the graph from a Micro-CT scan via Prim's Algorithm, using vessel segment volume as the edge weights.
- Better idea: Incorporate Murray's Minimum Work Principle.
- Work in a circulatory system is primarily due to two factors:
  1. Overcome blood flow resistance (friction; inversely proportional to vessel radius), and
  2. Metabolic support for the blood volume (proportional to vessel segment volume).
- The body minimizes the total work due to these two factors.
- Applied by Jiang, et al. to improve on the MST model. (2011)

The GCTP Model for Vascular Image Analysis

- "Cables:" Work to overcome friction
- "Trenches:" Work due to metabolic support
- Notation:
  - $r(e)$ = Radius of vessel segment $e \in E$
  - $|e|$ = Length of vessel segment $e \in E$
- Total Cable Length:

$$|T|_c = \sum_{j=2}^{n} \sum_{e \in \mathcal{P}(v_1, v_j)} \frac{|e|}{r(e)} \ , \ \text{so} \quad |e|_c = \frac{|e|}{r(e)}$$

- Total Trench Length:

$$|T|_t = \sum_{e \in E_T} |e| \cdot r^2(e) \ , \ \text{so} \quad |e|_t = |e| \cdot r^2(e)$$

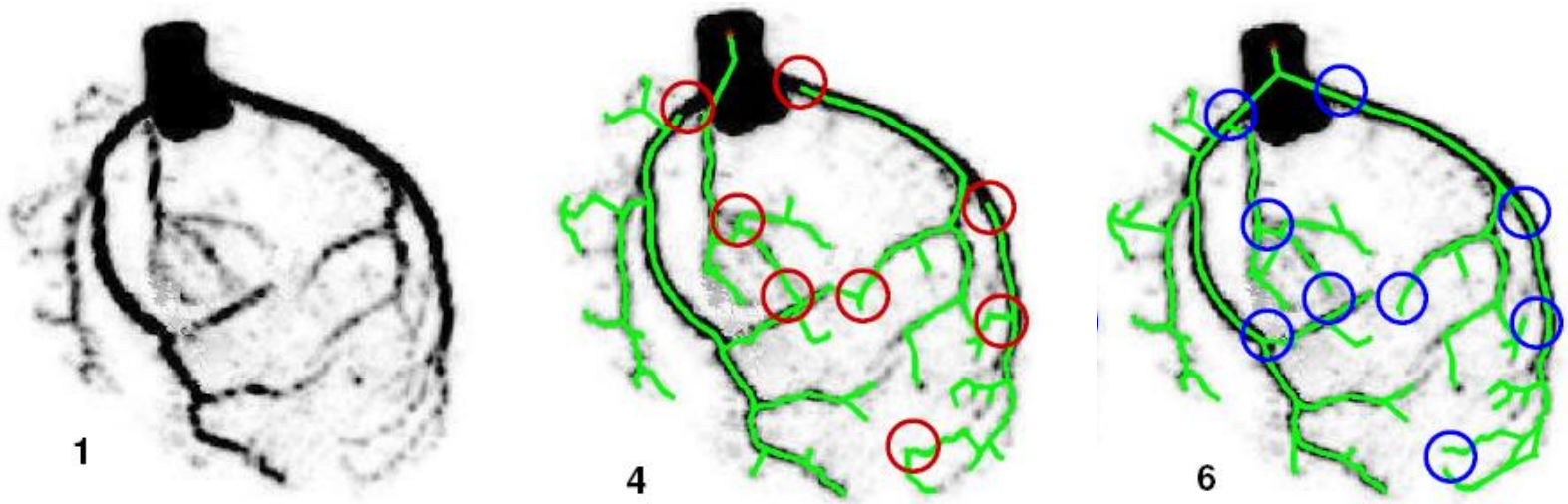
The GCTP Model for Vascular Image Analysis

- Find edges in E to minimize the total work:

$$\gamma|T|_c + \tau|T|_t = \gamma \sum_{j=2}^{n} \sum_{e \in \mathcal{P}(v_1, v_j)} \frac{|e|}{r(e)} + \tau \sum_{e \in E_T} |e| \cdot r^2(e)$$

- $\gamma = 1$ and $35000 \leq \tau \leq 175000$ for this application.

MST Model vs. GCTP Model



- The right-most image used ModPrim.
- Blue circles indicate improvements.
- There is still room for improvement. Can we do better?

Error-correction in Vascular Imaging

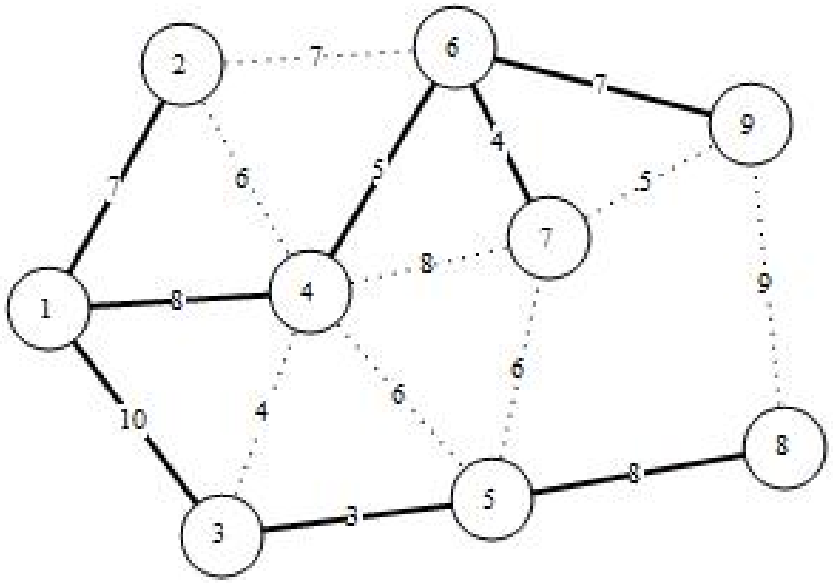
- Problem: There are errors in the imaging process.
- Can we avoid including these false-positive nodes?
- Information: The actual leaf nodes can be determined.
- Solution: Let S be the set of leaf nodes of T.
- Other nodes, called **Steiner** nodes, are optional.
- The resulting tree is called a **Steiner** tree.
- Vascular imaging error-correction is a GSCTP!

Another Nontrivial Application: Carpooling

- The root vertex (1) is the office.
- The other nodes are workers' homes.
- Edges are roads and weights are travel times (minutes).
- The "trench" cost, τ, is the cost to drive one minute.
- The "cable" cost, γ, is the per-minute value of a person's time.

Carpooling: If $1/4 \leq \tau/\gamma \leq 1$

- Worker 2 drives solo.
- Worker 8 picks up workers 5 and 3.
- Worker 7 goes to worker 6's house.
- Worker 9 picks up workers 6, 7, and 4.



Future Work and Open Questions

- Experiment with genetic algorithms.
- Apply concepts from the A^* algorithm. "Light-speed is too slow."
- Implement lower bound and ILP methods.
- Develop parallel algorithms.
- More rigorous statistical analysis.
- Theoretical considerations:
  - How close to optimal is ModPrim overall?
  - When are we guaranteed optimal solutions?
- Are there other fundamental approaches?
- Study other Steiner variants.

Thank you!

elandqui@kutztown.edu