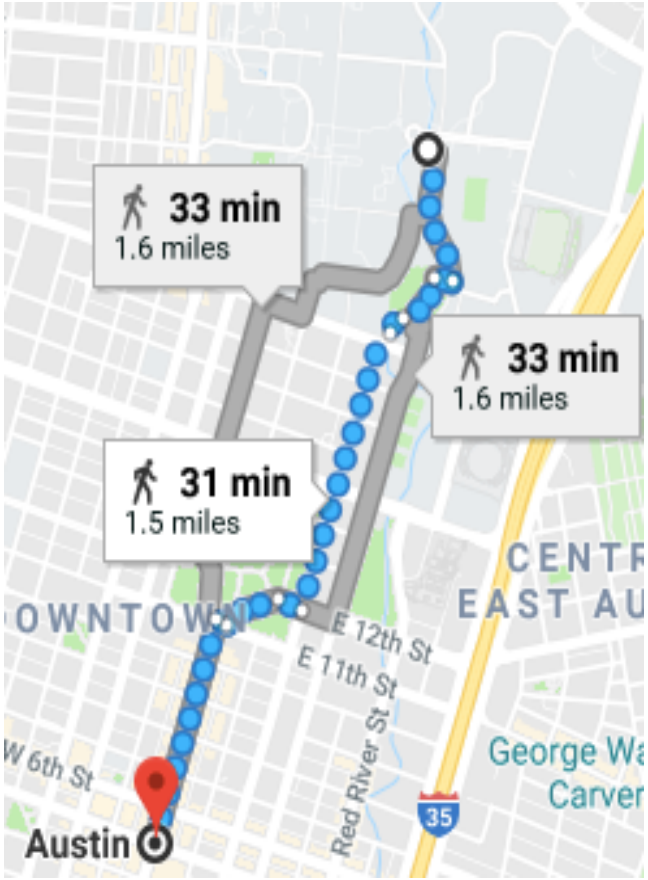
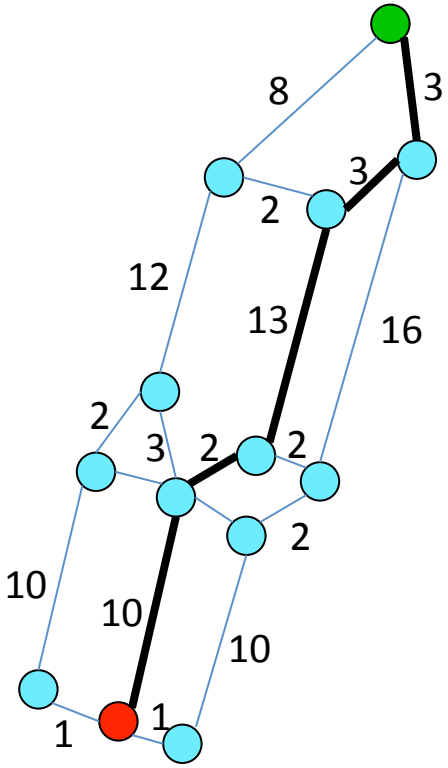
Algorithms, Game Theory and Risk-averse Decision Making*

Evdokia Nikolova
University of Texas at Austin

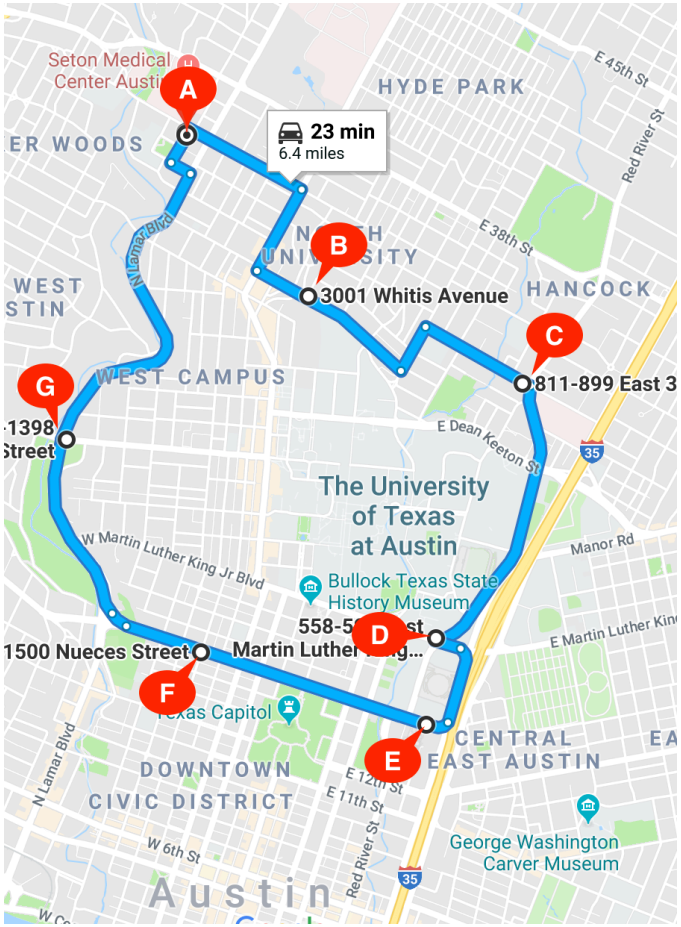
*Introductory lecture for Simons Real-time Decision Making Bootcamp, January 24, 2018

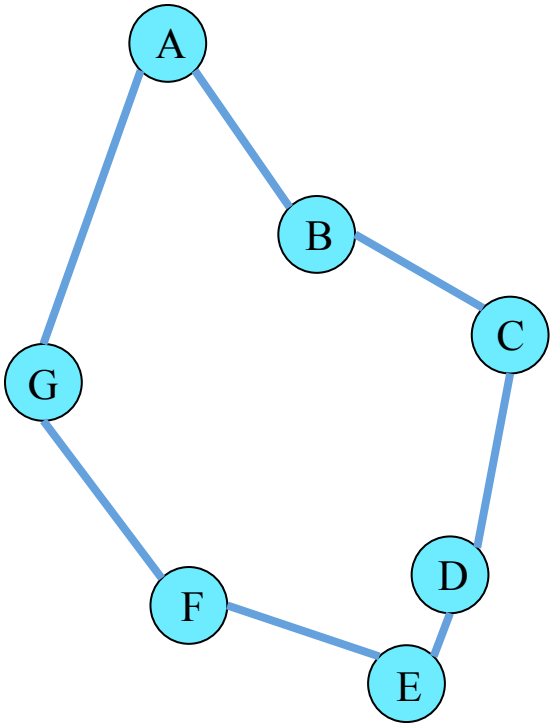
Real-time decision making examples





Real-time decision making examples

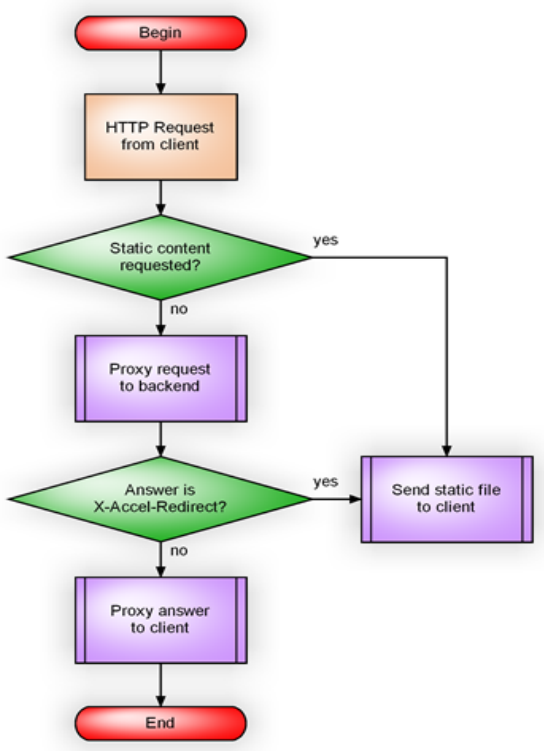




Part I: Algorithms

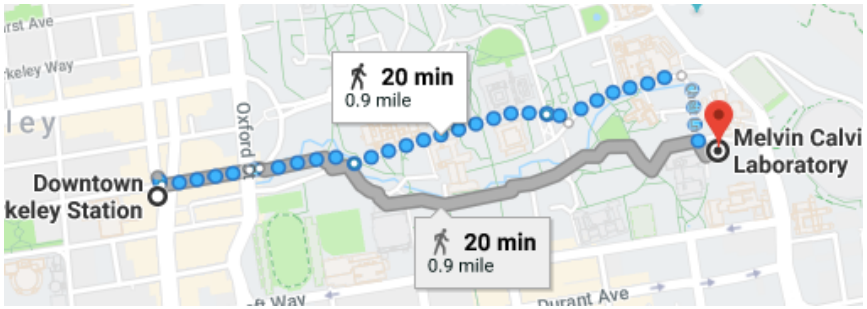
Algorithms: the basics

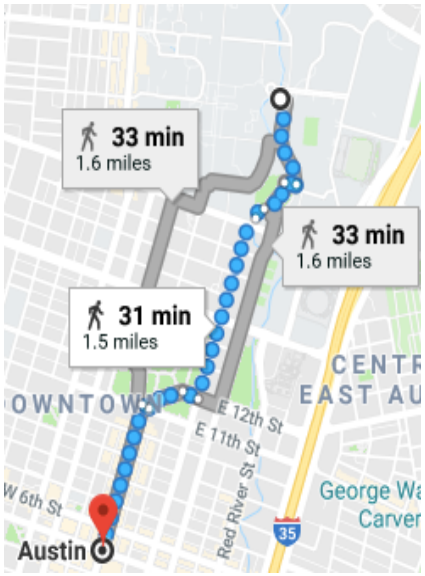
- What is an algorithm?
  - a step-by-step procedure to solve a problem
  - every computer program is the instantiation of some algorithm



Shortest paths example

- Solves a general, well-specified problem
  - given a graph G = (V, E), a source node s and a destination node *t*, and edge costs $c_1,\ldots,c_n$, as input, produce as output a shortest *st*-path, namely an *st*-path with smallest total edge cost
- Problem has specific instances





- Algorithm takes every possible instance and produces output with desired properties
  - Dijkstra, Bellman-Ford, Floyd-Warshall shortest path algorithms

Modeling the real-world

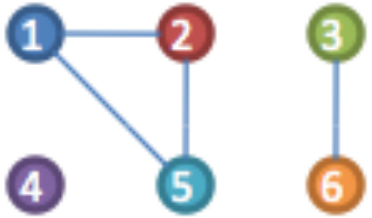
- Cast your application in terms of well-studied abstract data structures

Concrete	*Abstract*
arrangement, tour, ordering, sequence	permutation
cluster, collection, committee, group, packaging, selection	subsets
hierarchy, ancestor/descendants, taxonomy, radial network	trees
network, circuit, web, relationship	graph
sites, positions, locations	points
shapes, regions, boundaries	polygons
text, characters, patterns	strings

Graph terminology

- A **directed graph** (or **digraph**) G is a pair (V, E) where V is a finite set (of "vertices") and E (the "edges") is a subset of $V \times V$.

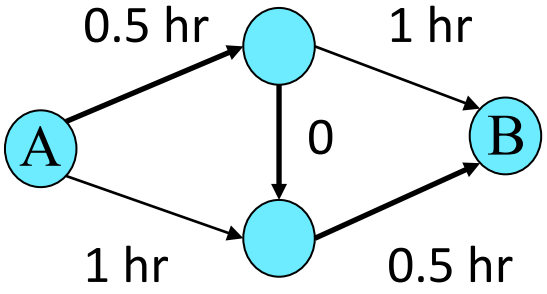
- An **undirected graph** G is a pair (V, E) where V is a finite set (of "vertices") and E (the "edges") is a set of unordered pairs of edges $\{u, v\}$, where $u \neq v$.



Graph = Network
Nodes = Vertices
Edges = Links = Arcs

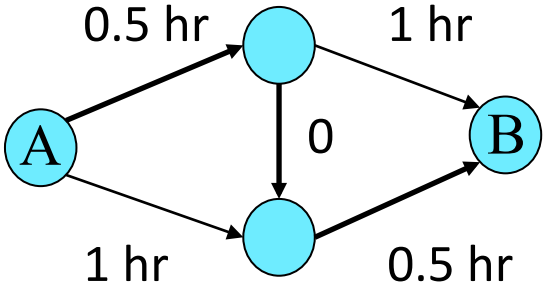
Graph examples

- Abstraction of transportation graph (Braess paradox graph)

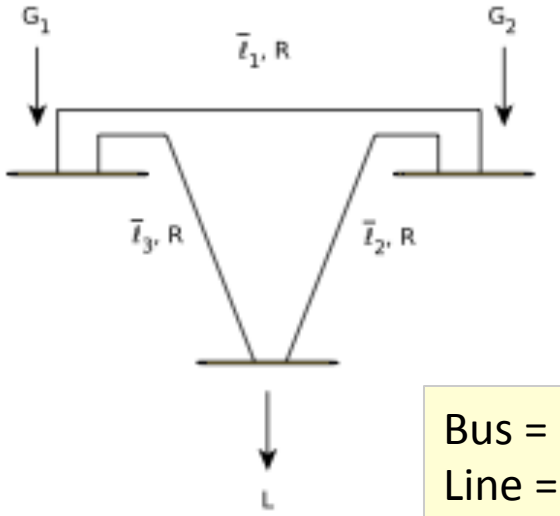


Graph examples

- Abstraction of transportation graph (Braess paradox graph)



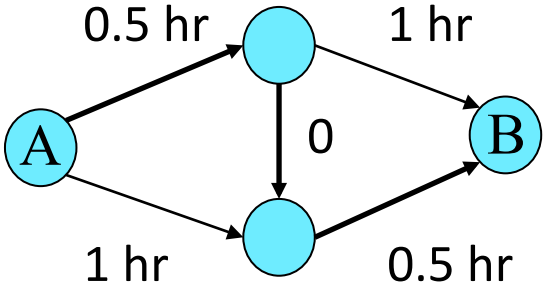
- Abstraction of electricity graph:



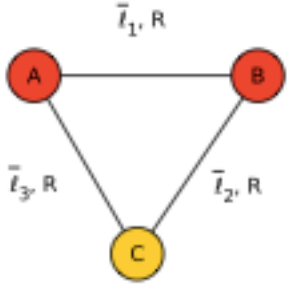
Bus = Node
Line = Edge?

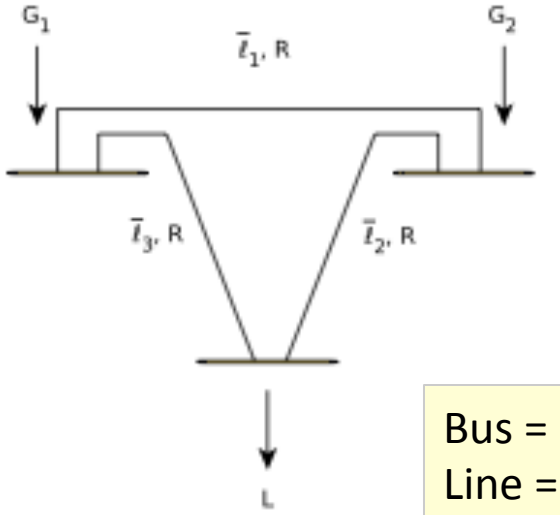
Graph examples

- Abstraction of transportation graph (Braess paradox graph)



- Abstraction of electricity graph:

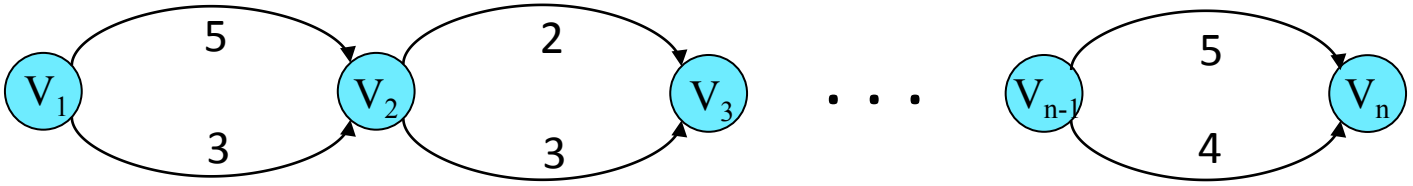




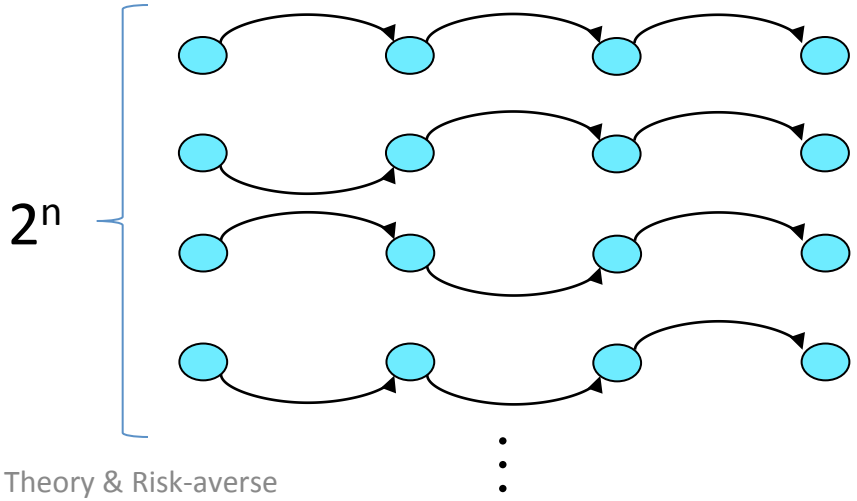
Bus = Node
Line = Edge?

Back to shortest paths

- Algorithmic challenge: exponentially many paths

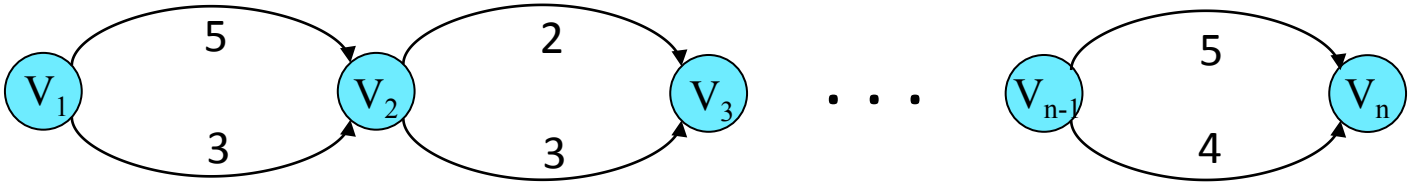


- Brute force: enumerate all possible paths; output best one. Brute force algorithm has exponential running time



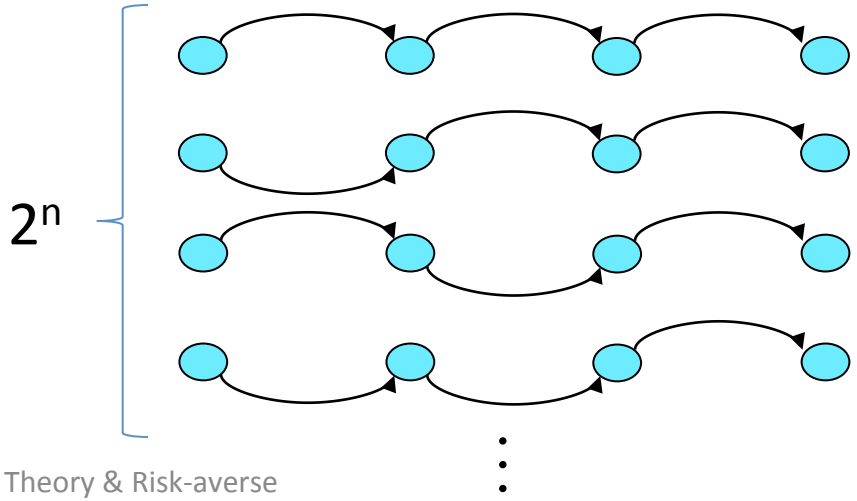
Back to shortest paths

- Algorithmic challenge: exponentially many paths



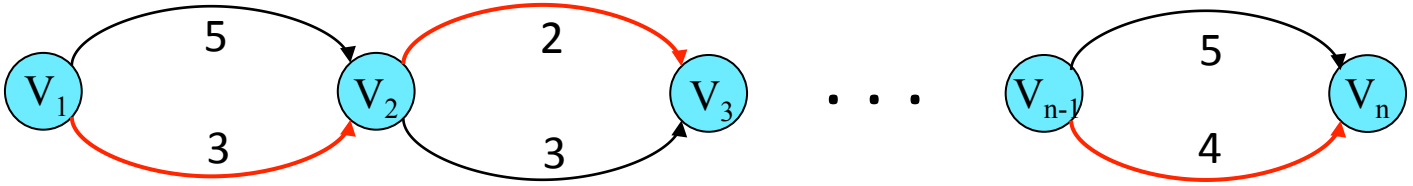
- Brute force: enumerate all possible paths; output best one. Brute force algorithm has exponential running time

10^{67}:
number of atoms in our galaxy

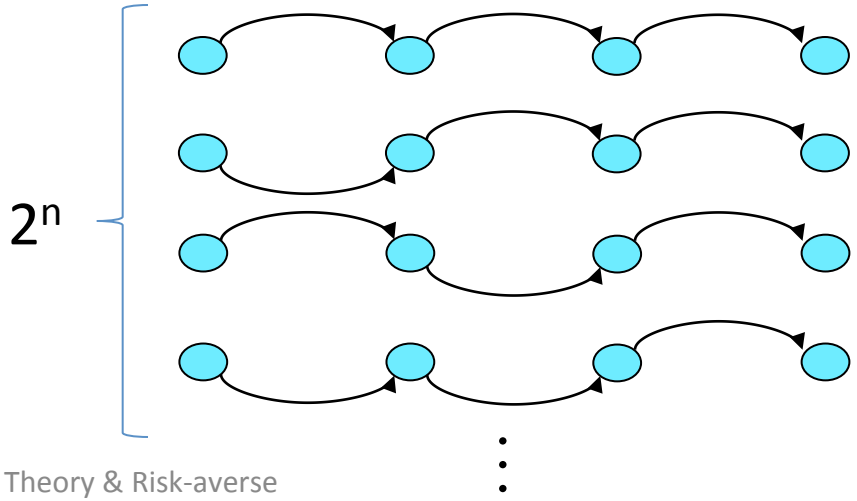


Back to shortest paths

- Algorithmic challenge: exponentially many paths

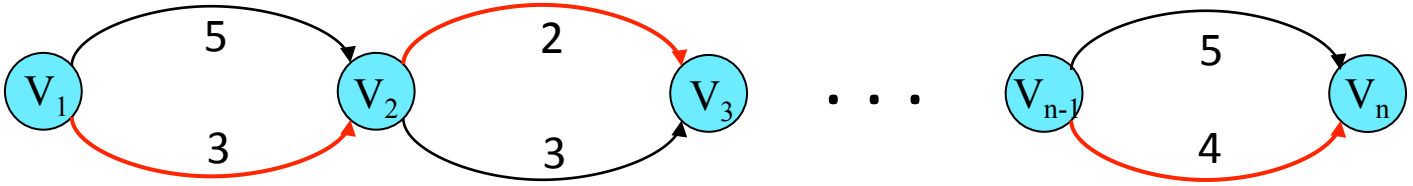


- Brute force: enumerate all possible paths; output best one. Brute force algorithm has exponential running time

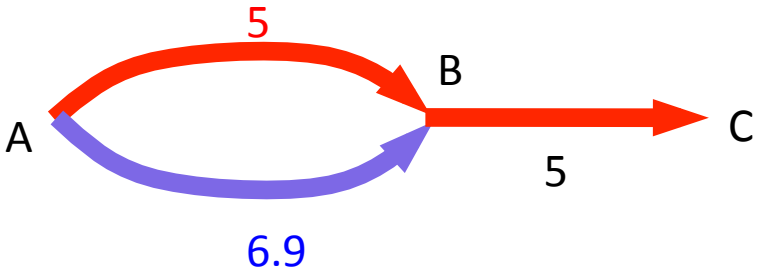


Back to shortest paths

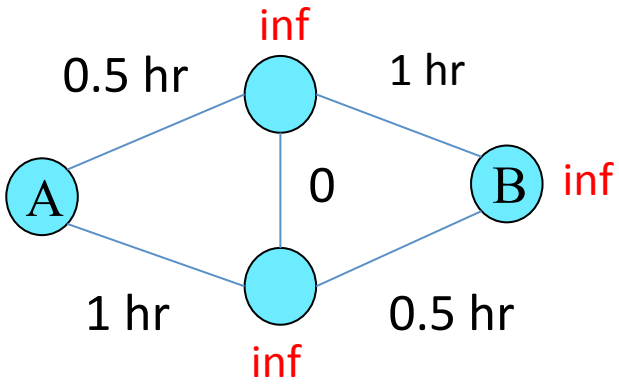
- Algorithmic challenge: exponentially many paths



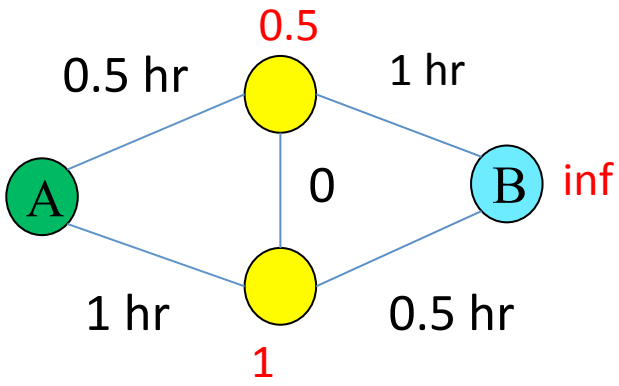
- In shortest paths, *optimal substructure* property *(part of the optimal solution remains optimal on the subproblem)* allows for efficient dynamic programming algorithms (Dijkstra, Bellman-Ford, etc)



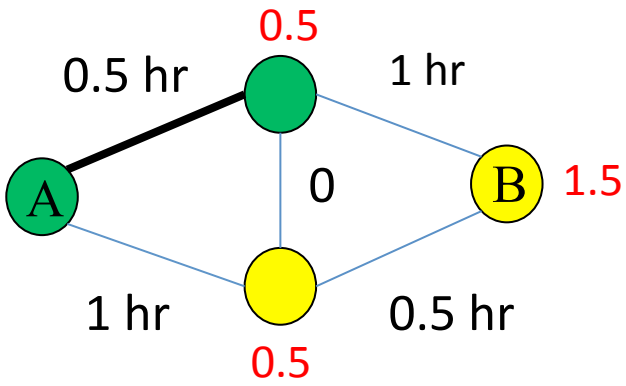
Dijkstra shortest path algorithm



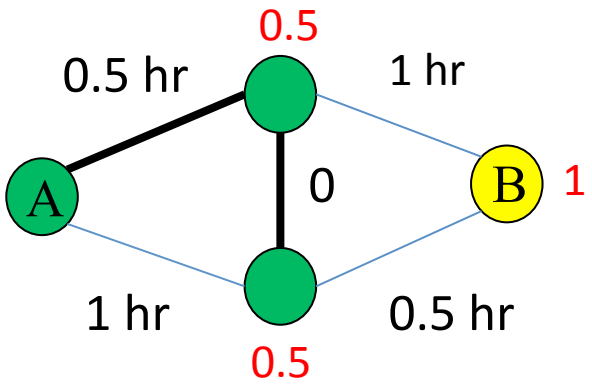
Dijkstra shortest path algorithm



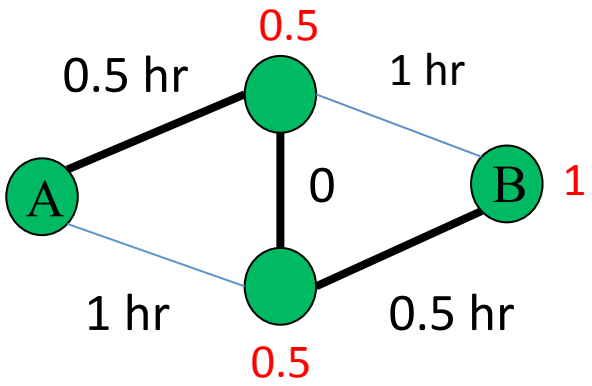
Dijkstra shortest path algorithm



Dijkstra shortest path algorithm



Dijkstra shortest path algorithm



(Basic) Algorithm Design Techniques

- Brute Force & Exhaustive Search
  - follow definition / try all possibilities
- Divide & Conquer
  - break problem into distinct subproblems
- Transformation
  - convert problem to another one
- Dynamic Programming
  - break problem into overlapping subproblems
- Greedy
  - repeatedly do what is best now
- Randomization
  - use random numbers
- Linear programming

Algorithm running time

- We would like a definition of algorithm efficiency that is:
  - platform-independent
  - instance-independent
  - of predictive value with respect to increasing instance (input) sizes (e.g., for a graph, input size is the number of nodes *n* and edges *m*)

- **An algorithm is efficient if it has polynomial running time.**
  - $f(n) = a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ is a polynomial of degree k.

Algorithm running time

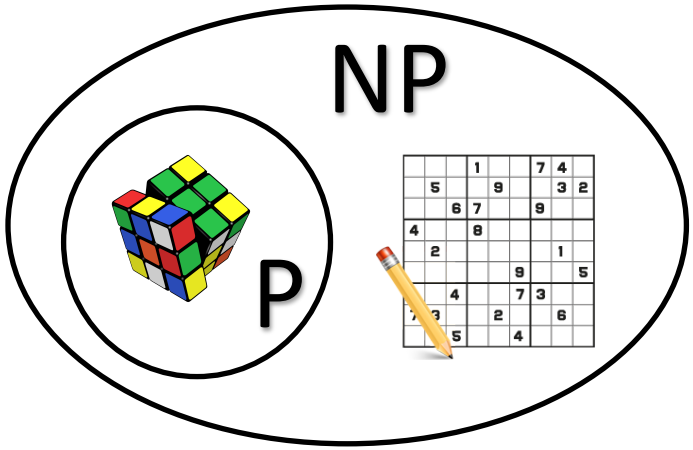
- **An algorithm is efficient if it has polynomial running time.**
  - f(n) = $a_k n^k + a_{k-1} n^{k-1} + \ldots + a_1 n + a_0$ is a polynomial of degree k.
  - We care about highest order term and no coefficients: algorithm polynomial runtime is $O(n^k)$, where k is a constant

- Of course, running time of n^{100} is clearly not great, and a running time of $n^{1+.02(\log n)}$ is not clearly bad. But in practice, polynomial time is generally good.

- In addition to being precise, this definition is also *negatable*.

P vs NP

- P is the class of problems which can be solved in polynomial time
- NP ("nondeterministic polynomial time") is the class of problems for which a candidate solution can be verified in polynomial time
  - Note: NP does ***not*** stand for "not polynomial"
  - P is a subset of NP



P vs NP

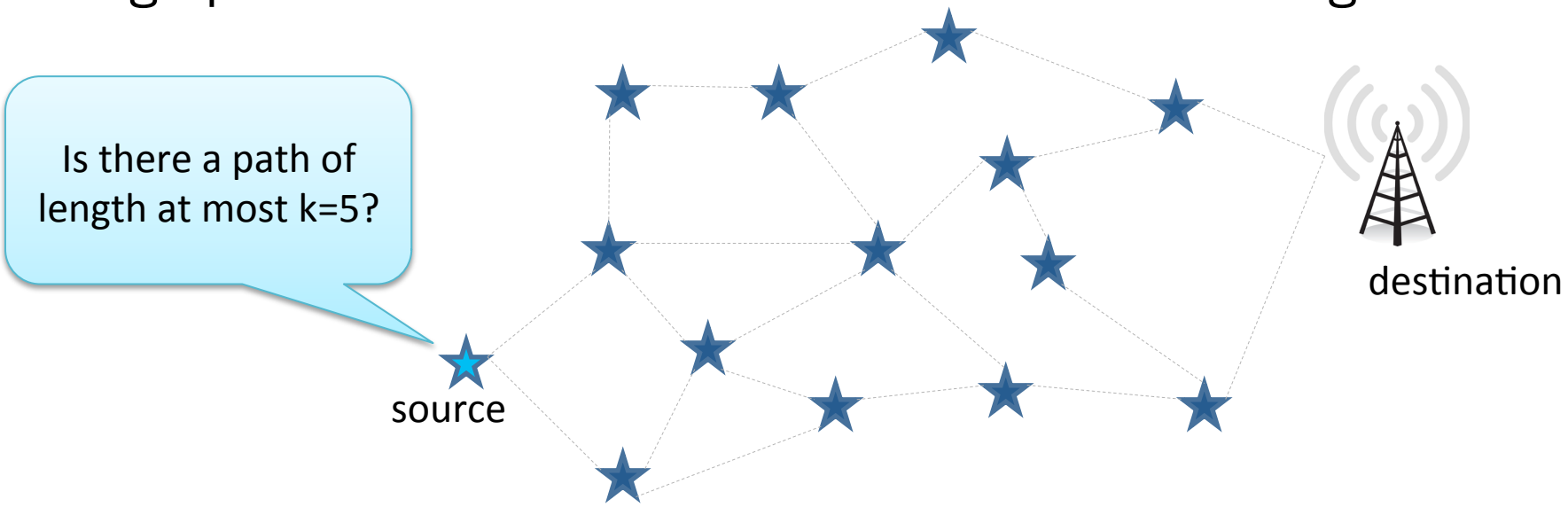
Difference between solving a problem and verifying a candidate solution:

- Solving a problem: is there a path in graph G from vertex u to vertex v with at most k edges?
- Verifying a candidate solution: is $v_0, v_1, \ldots, v_\ell$ a path in graph G from vertex u to vertex v with at most k edges?

P vs NP

Difference between solving a problem and verifying a candidate solution:

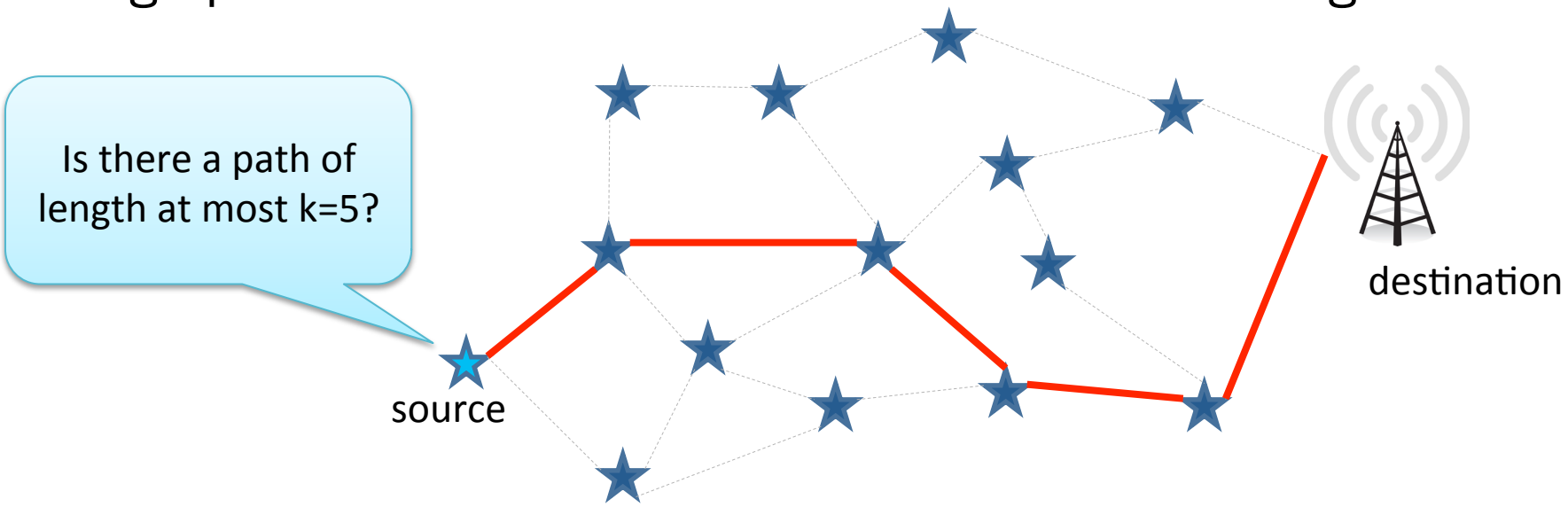
- Solving a problem: is there a path in graph G from vertex u to vertex v with at most k edges?
- Verifying a candidate solution: is $v_0, v_1, \ldots, v_\ell$ a path in graph G from vertex u to vertex v with at most k edges?



P vs NP

Difference between solving a problem and verifying a candidate solution:

- Solving a problem: is there a path in graph G from vertex u to vertex v with at most k edges?
- Verifying a candidate solution: is $v_0, v_1, \ldots, v_\ell$ a path in graph G from vertex u to vertex v with at most k edges?

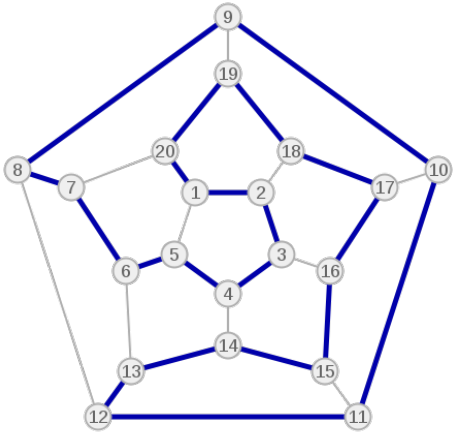


P vs NP

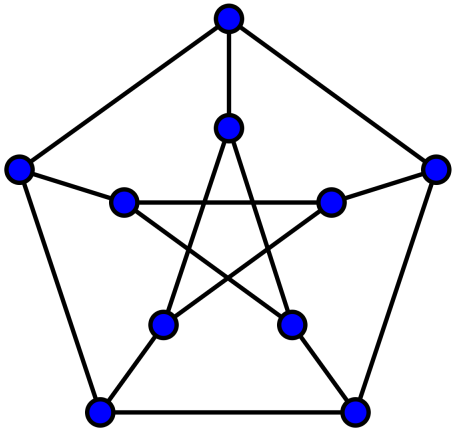
- A Hamiltonian cycle in an undirected graph is a cycle that visits every vertex exactly once.
- Solving a problem: is there a Hamiltonian cycle in graph G?
- Verifying a candidate solution: is $v_0, v_1, \ldots, v_\ell$ a Hamiltonian cycle of graph G?

P vs NP

- A Hamiltonian cycle in an undirected graph is a cycle that visits every vertex exactly once.
- Solving a problem: is there a Hamiltonian cycle in graph G?
- Verifying a candidate solution: is $v_0, v_1, \ldots, v_\ell$ a Hamiltonian cycle of graph G?



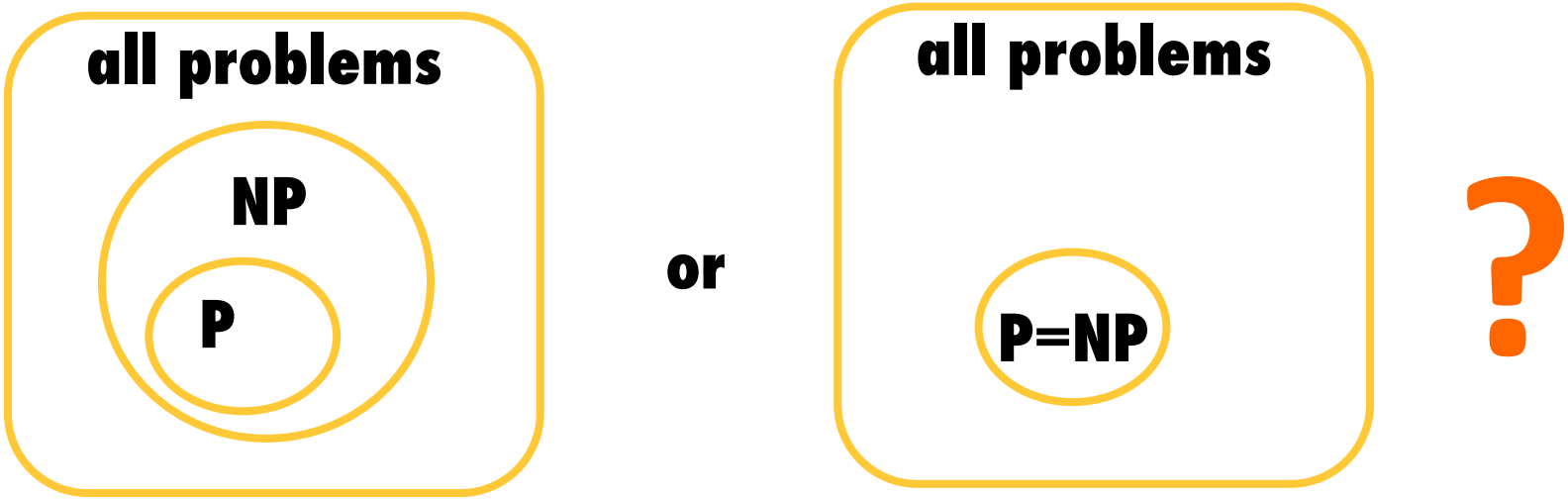
YES



NO

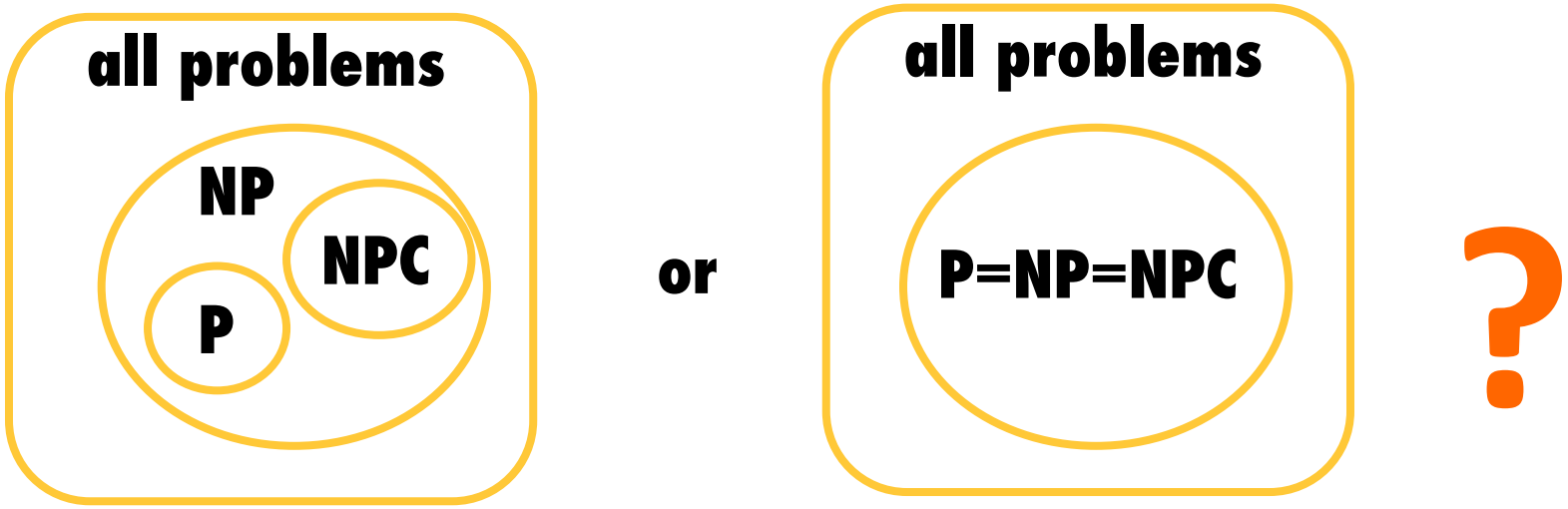
P vs NP

- Although poly time verifiability *seems* like a weaker condition than poly time solvability, no one has been able to prove that it is weaker (describes a larger class of problems)
- So it is unknown whether P = NP.



NP-Complete problems

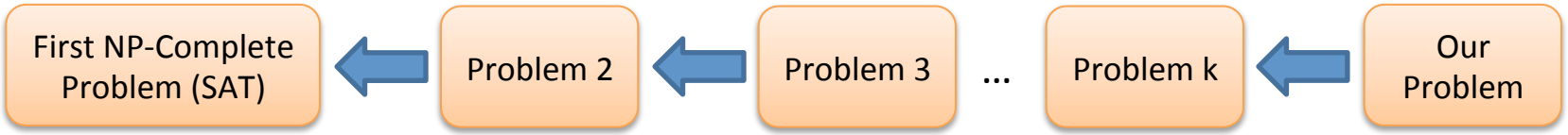
- NP-complete problems is class of "hardest" problems in NP.
- They have the property that if any NP-complete problem can be solved in poly time, then all problems in NP can be, and thus P = NP.



NPC = NP-complete

NP-Complete problems

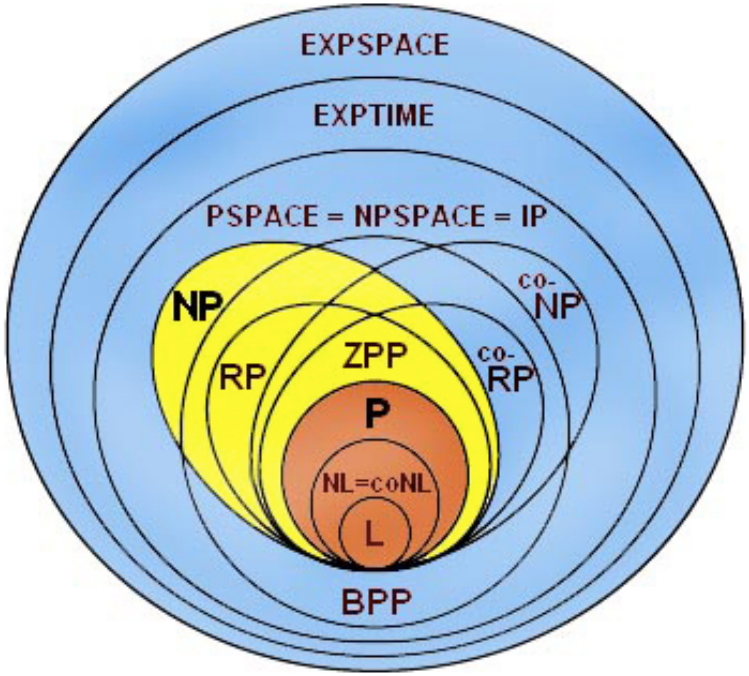
- NP-complete problems is class of "hardest" problems in NP.
- A decision problem is NP-complete if
  - It is in NP
  - It is “at least as hard as” some known NP-complete problem



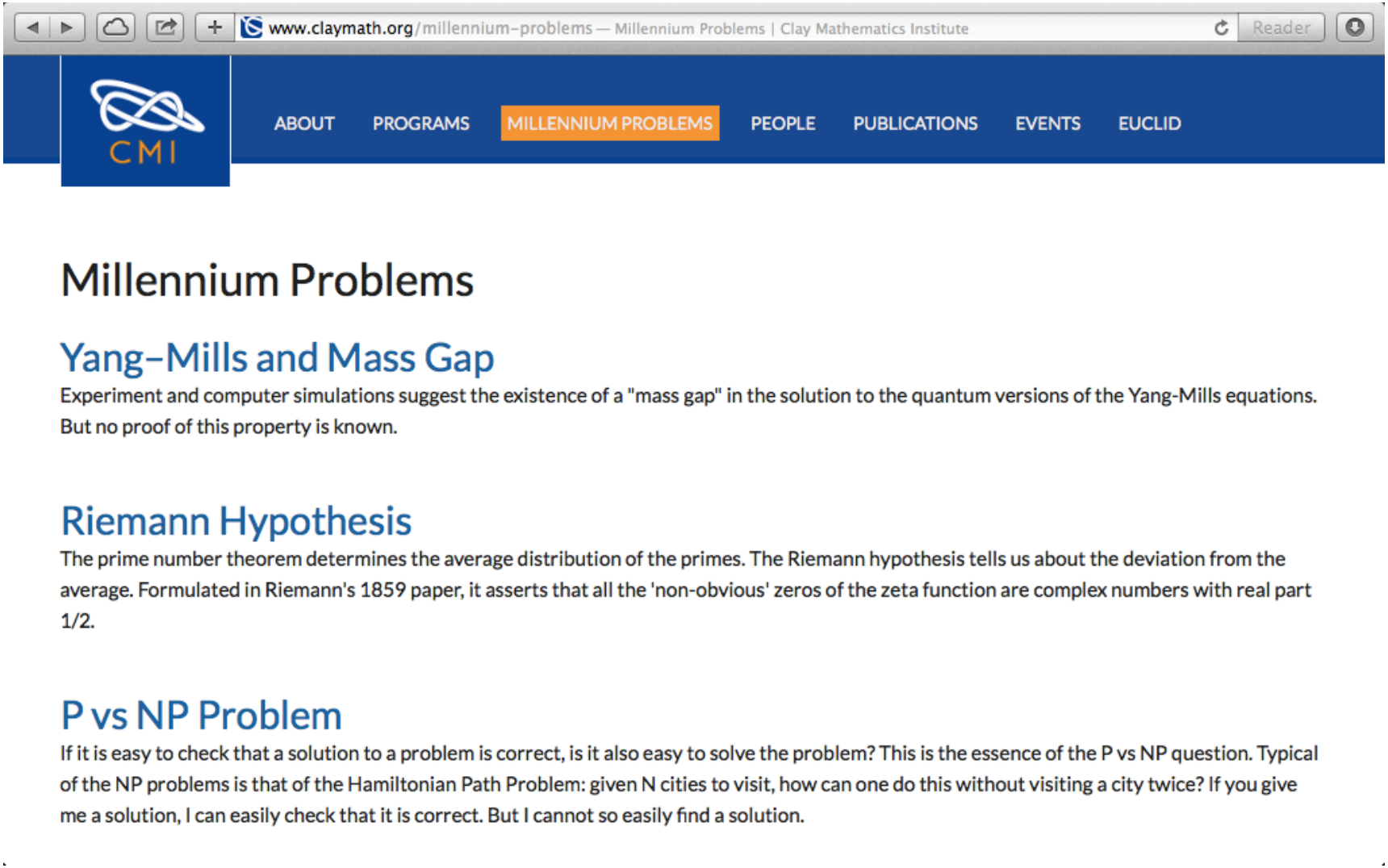
- When given a new problem, a computer science theorist first wants to understand whether the problem is in P, or NP-complete, or another complexity class.

NP-Complete problems

- NP-complete problems is class of "hardest" problems in NP.
- They have the property that if any NP-complete problem can be solved in poly time, then all problems in NP can be, and thus P = NP.
- Other complexity classes:



P vs NP problem



P vs NP problem

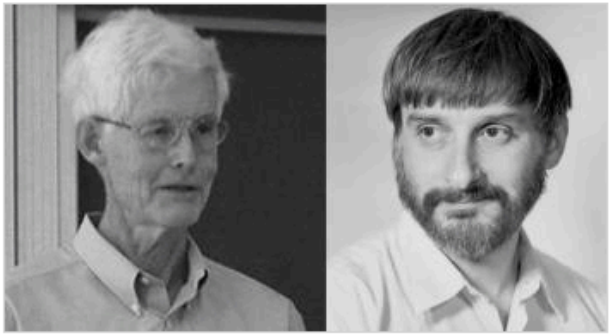
www.claymath.org/millennium-problems/p-vs-np-problem — P vs NP Problem | Clay Mathematics Institute

Reader

CMI

ABOUT PROGRAMS MILLENNIUM PROBLEMS PEOPLE PUBLICATIONS EVENTS EUCLID

P vs NP Problem



Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether

Rules:

Rules for the Millennium Prizes

Related Documents:

Official Problem Description

Minesweeper

Related Links:

P vs NP problem

- Open question since about 1970
- One of the seven "Millennium Prize Problems" by the Clay Mathematics Institute ($1 million prize for solving it)*
- Great theoretical interest
- Great practical importance:
  - If your problem is NP-complete, then don't waste time looking for an efficient algorithm
  - Instead look for efficient approximations, heuristics, etc.

* http://www.claymath.org/millennium-problems

Approximation algorithms*

- If we do not know how to solve a problem in polynomial time, can we find a near-optimal solution in polynomial time?
- An α-approximation algorithm for an optimization problem
  - runs in polynomial time
  - always returns a candidate solution
  - cost of returned solution is at most α times the cost of the optimal solution (for a minimization problem)

*This semester: CS294-145 "Approximation Algorithms" instructor David Williamson. Tue/Thu 3:30-5:00 in 310 Soda Hall.

Traveling Salesman Problem

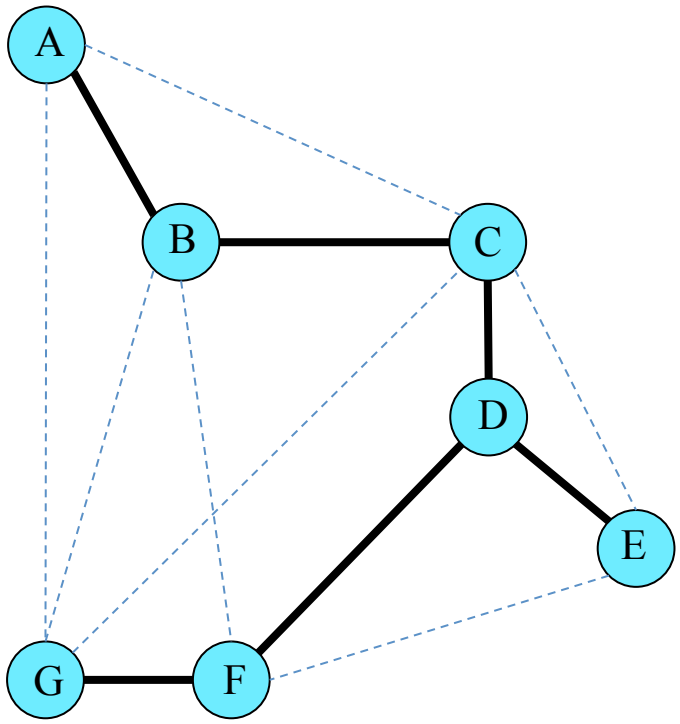
- Given a set of cities, distances between all pairs of cities, and a bound B, does there exist a tour (sequence of cities to visit that returns to the start and visits each city exactly once) that requires at most distance B to be traveled?
- TSP is in NP:
  - given a candidate solution (a tour), add up all the distances and check if total is at most B

TSP approximation algorithm

- **Input:** set of cities and distances b/w them that satisfy the triangle inequality

TSP approximation algorithm

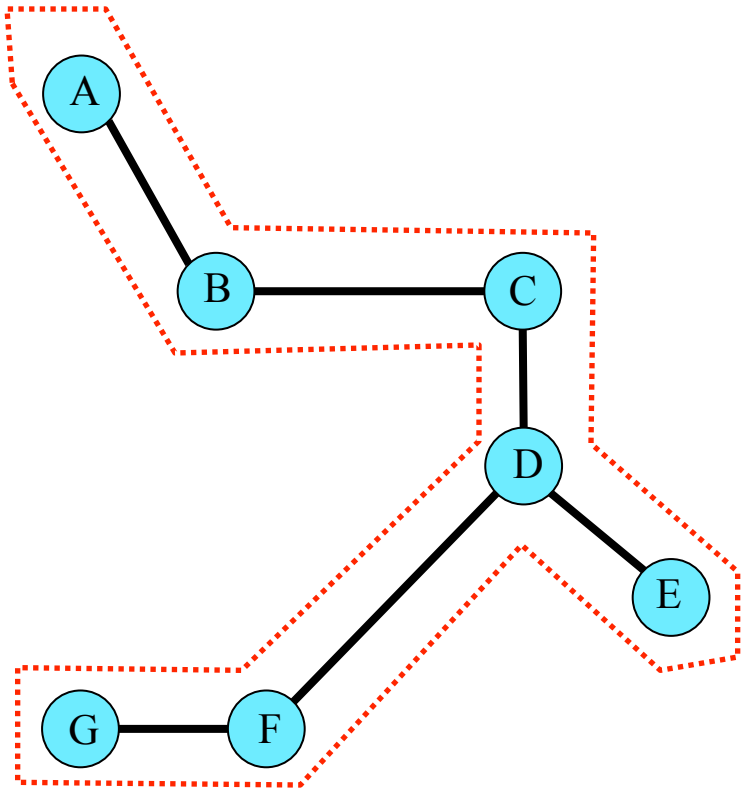
- **Input:** set of cities and distances b/w them that satisfy the triangle inequality



1) Compute MST

TSP approximation algorithm

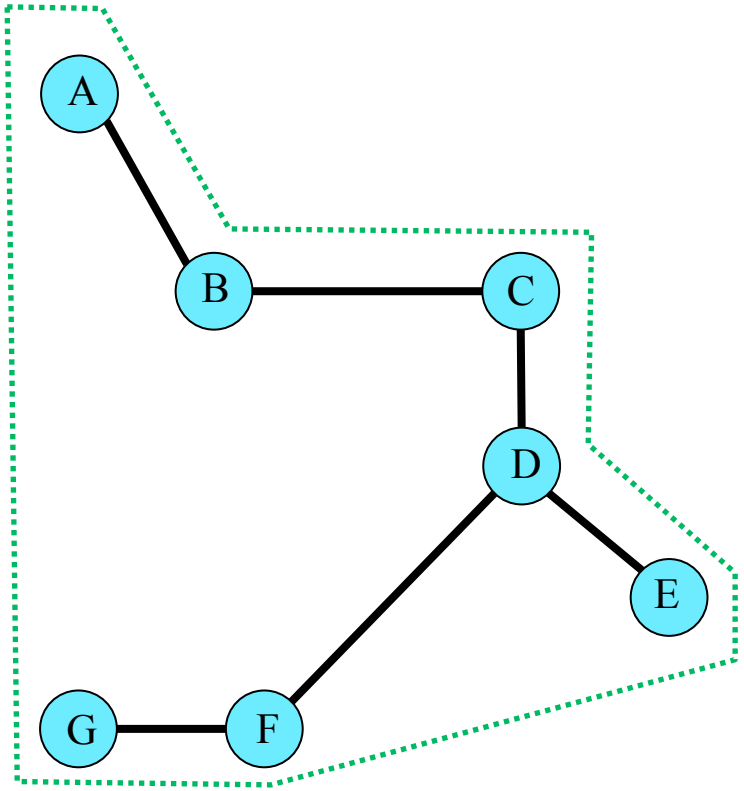
- **Input:** set of cities and distances b/w them that satisfy the triangle inequality



1) Compute MST
2) Go around MST to get a tour

TSP approximation algorithm

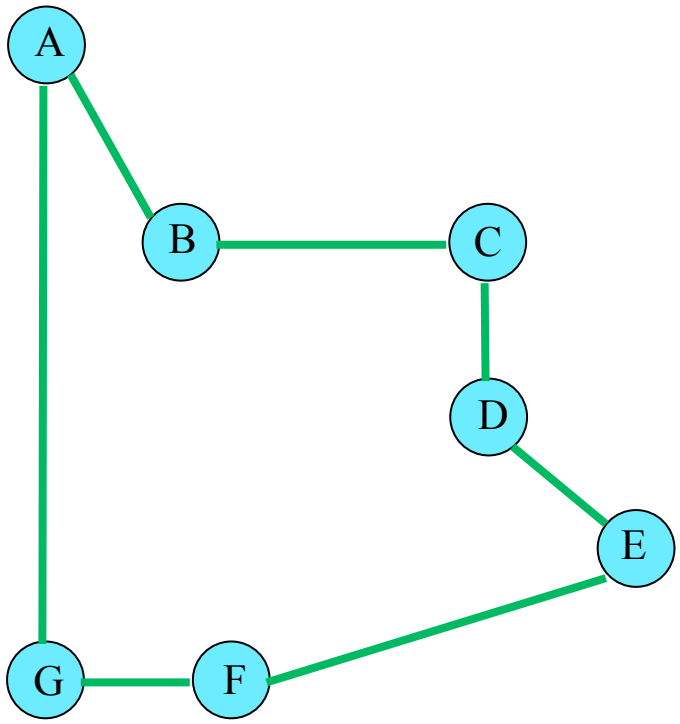
- **Input:** set of cities and distances b/w them that satisfy the triangle inequality



1) Compute MST
2) Go around MST to get a tour
3) Remove duplicate cities

TSP approximation algorithm

- **Input:** set of cities and distances b/w them that satisfy the triangle inequality



1) Compute MST
2) Go around MST to get a tour
3) Remove duplicate cities

- This is a 2-approximation algorithm

Part II: Game Theory

Best route depends on others

Game theory

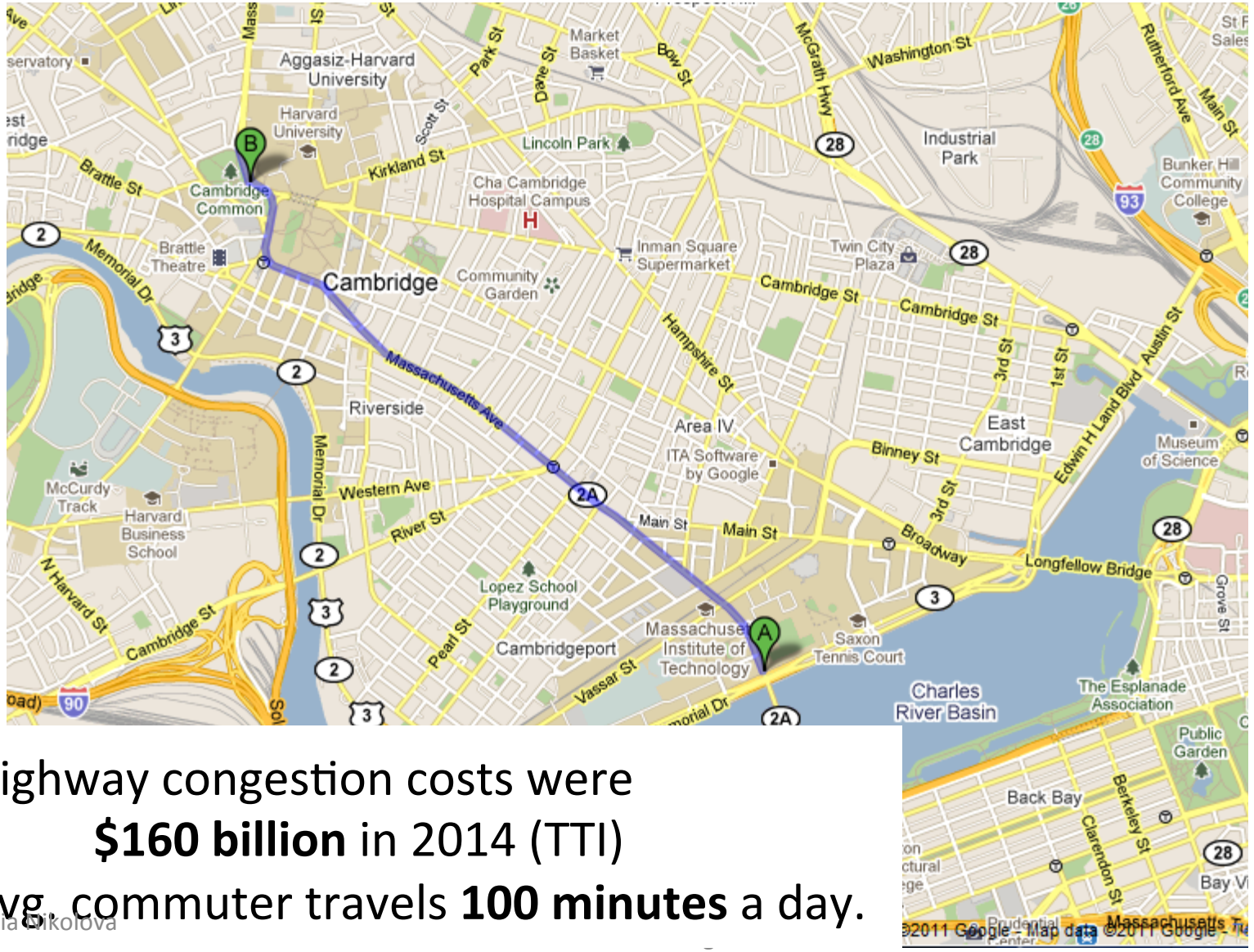
- **Games** are thought experiments to help us learn how to ***predict rational behavior*** in ***situations of conflict***

- ***Situation of conflict:*** Everybody's actions affect others. This is captured by the tabular game formalism.

- ***Rational Behavior:*** The players want to maximize their own expected utility. No altruism, envy, masochism, or externalities (if my neighbor gets the money, he will buy louder stereo, so I will hurt a little myself...).

- ***Predict:*** We want to know what happens in a game. Such predictions are called solution concepts (e.g., Nash equilibrium).

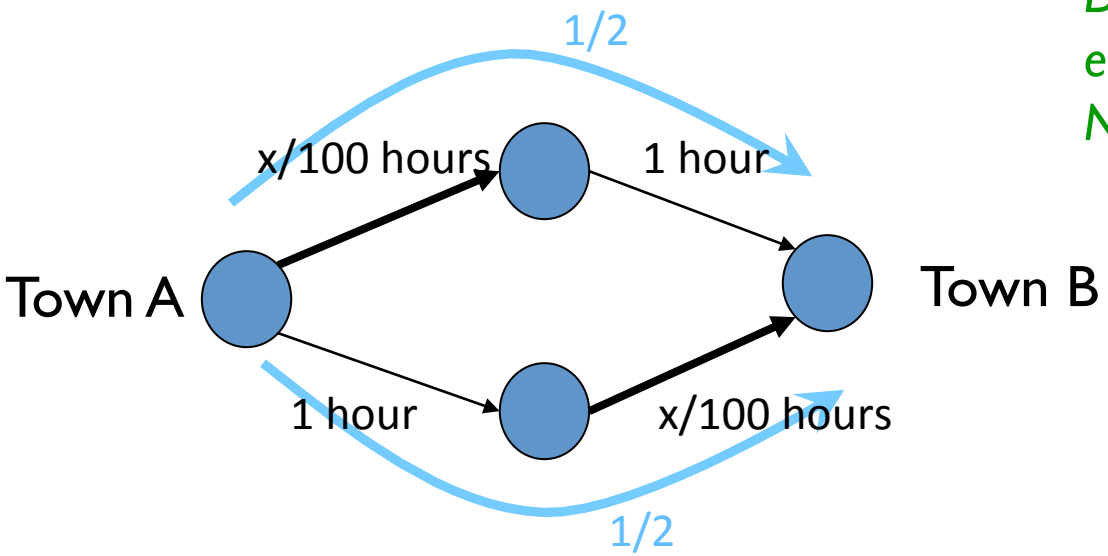
Travel time increases with congestion



- Highway congestion costs were **$160 billion** in 2014 (TTI)
- Avg. commuter travels **100 minutes** a day.

Example: Inefficiency of equilibria

Delay is 1.5 hours for everybody at the unique Nash equilibrium



Suppose 100 drivers leave from town A towards town B.

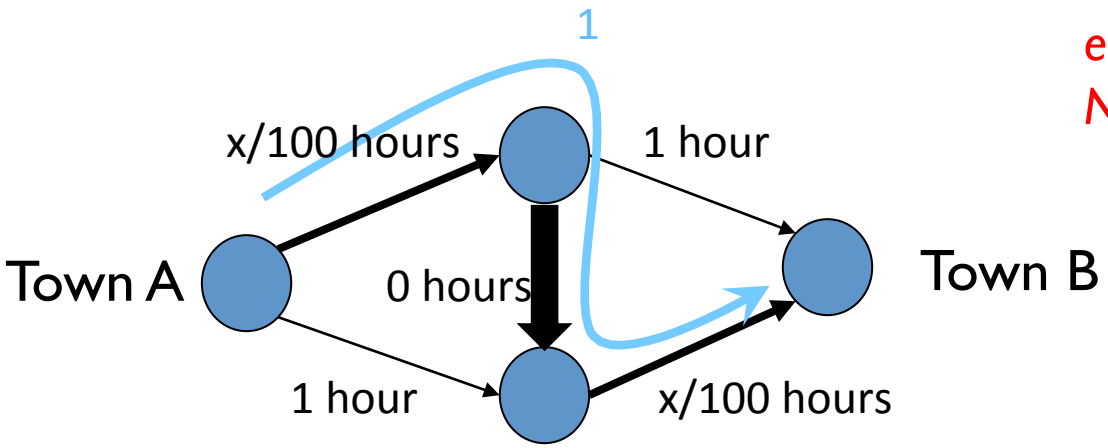
Every driver wants to minimize her own travel time.

What is the traffic on the network?

In any unbalanced traffic pattern, all drivers on the most loaded path have incentive to switch their path.

Example: Inefficiency of equilibria

Delay is 2 hours for everybody at the unique Nash equilibrium

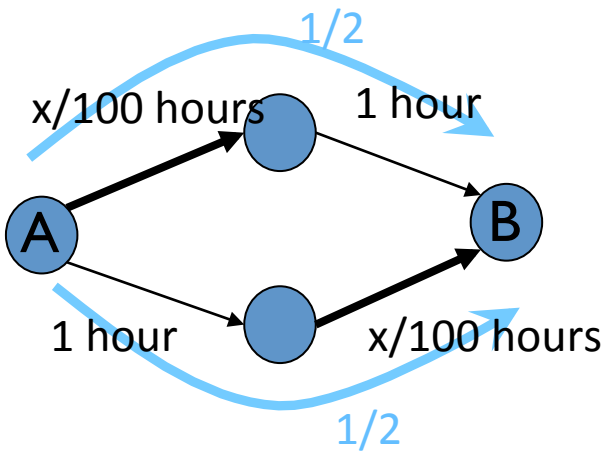


A benevolent mayor builds a superhighway connecting the fast highways of the network.

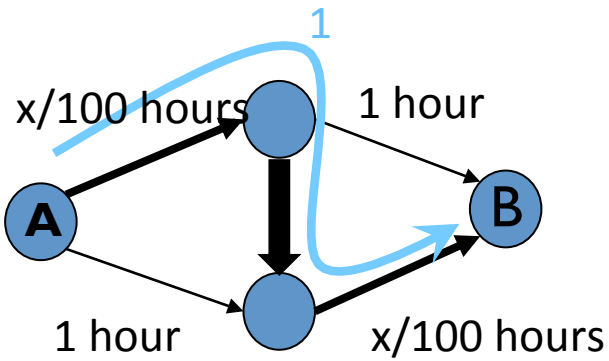
What is now the traffic on the network?

No matter what the other drivers are doing it is always better for me to follow the zig-zag path.

Example: Inefficiency of equilibria



vs



Adding a fast road on a road-network is not always a good idea!

Braess's paradox

4/3

In the RHS network there exists a traffic pattern where all players have delay 1.5 hours.

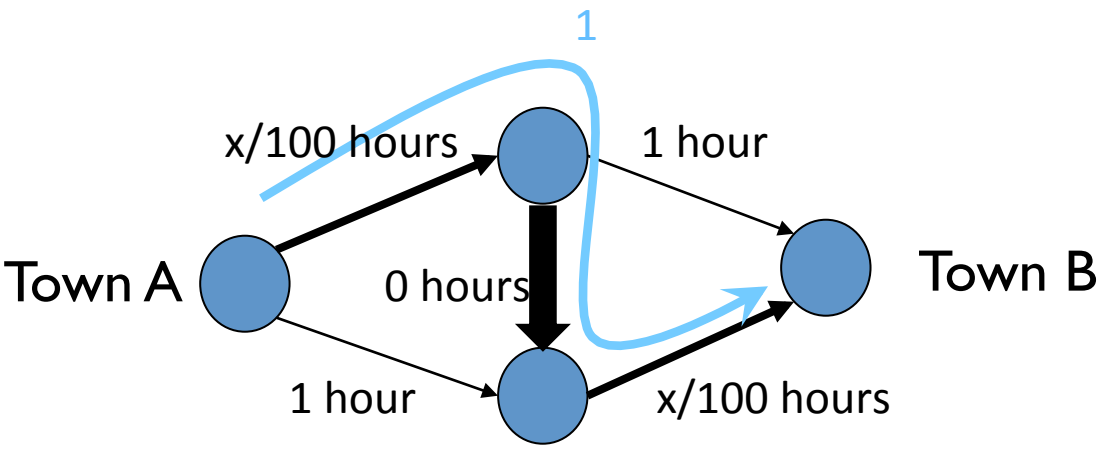
$$\text{PoA} = \frac{\text{performance of system in worst Nash equilibrium}}{\text{optimal performance if drivers did not decide on their own}}$$

Price of Anarchy: measures the loss in system performance due to free-will

Equilibrium

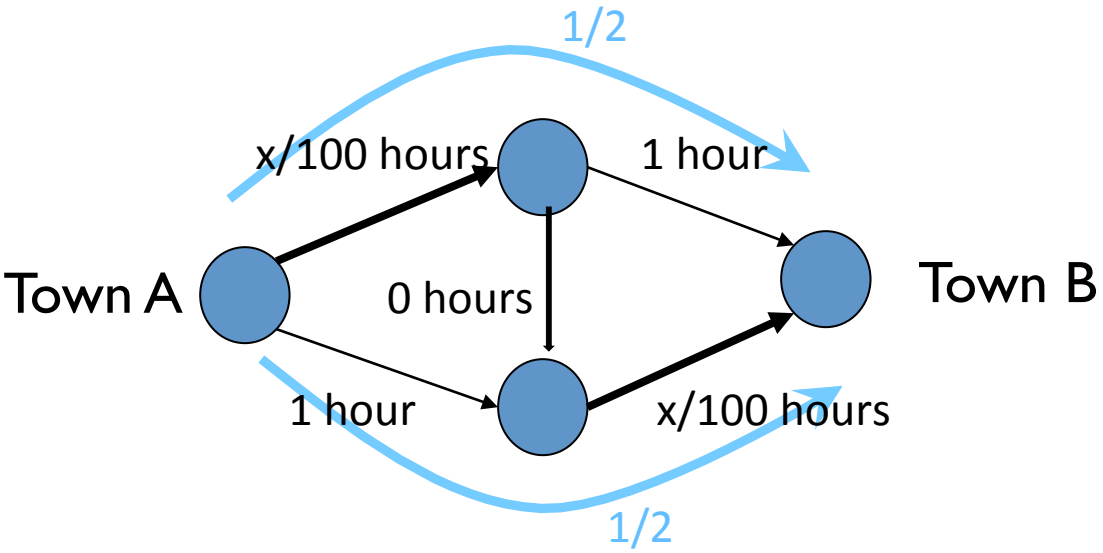
- "Travel times on used routes are equal and no greater than travel times on unused routes."

- Defines Wardrop Equilibrium, also called User Equilibrium or Nash Equilibrium.



Social Optimum

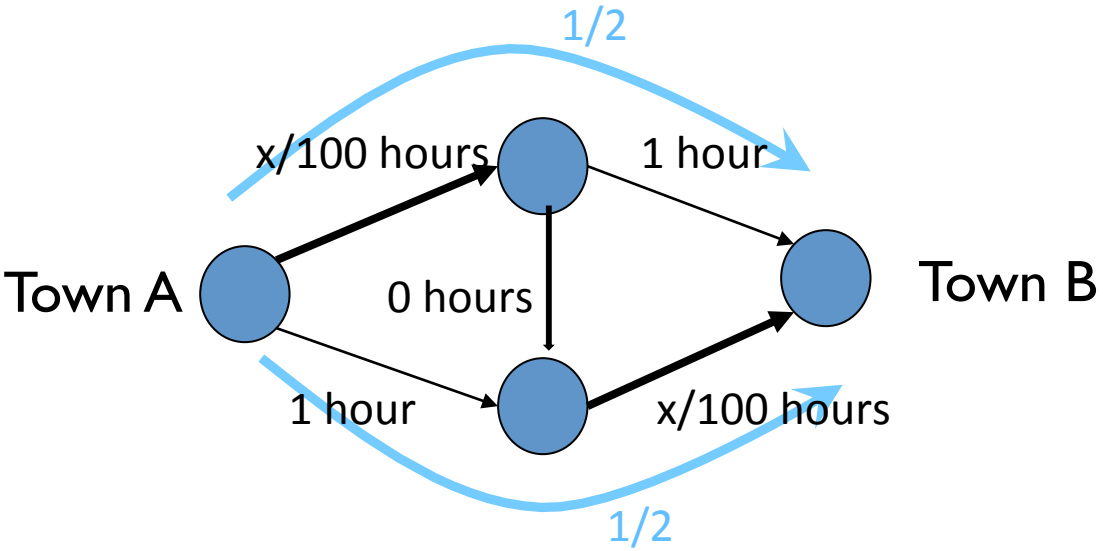
- "The average [total] journey time is minimum."
- Defines Social Optimum (SO)



Social Optimum

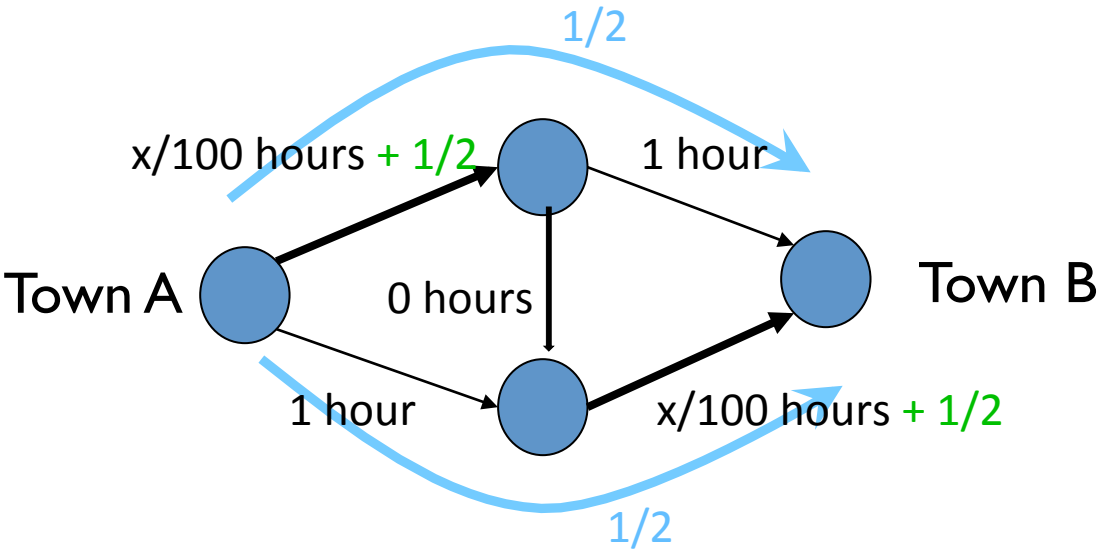
- "[Modified] Travel times on used routes are equal and no greater than travel times on unused routes."

- Defines Social Optimum (SO)



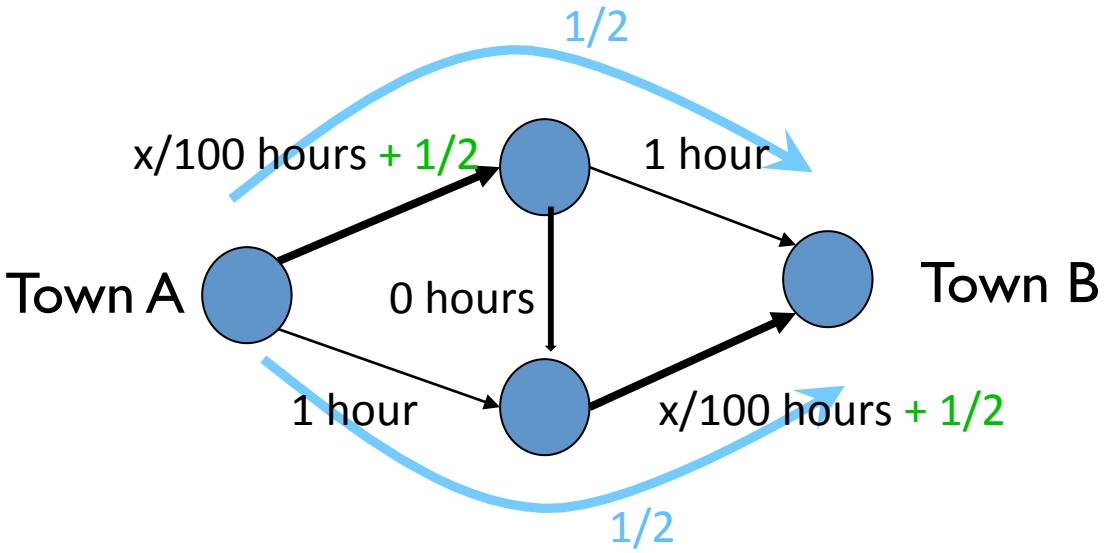
Social Optimum

- "[Modified] Travel times on used routes are equal and no greater than travel times on unused routes."
- Social Optimum (SO) is Equilibrium w.r.t travel times plus tolls!



Social Optimum

- "[Modified] Travel times on used routes are equal and no greater than travel times on unused routes."
- Social Optimum (SO) is Equilibrium w.r.t travel times plus tolls! → Mechanism design



Note: both Equilibrium and Social optimum exist and can be computed efficiently.

Price of Anarchy

- Price of anarchy: (Koutsoupias, Papadimitriou '99)

$$\sup_{\text{problem instances}} \frac{\text{Equilibrium Cost}}{\text{Social Optimum Cost}}$$

- Measures the degradation of system performance due to free will (selfish behavior)

- 4/3 in general graphs, linear delays as function of traffic; 2 for quartic delays (Roughgarden, Tardos '02; Correa, Schulz, Stier-Moses '04, '08)

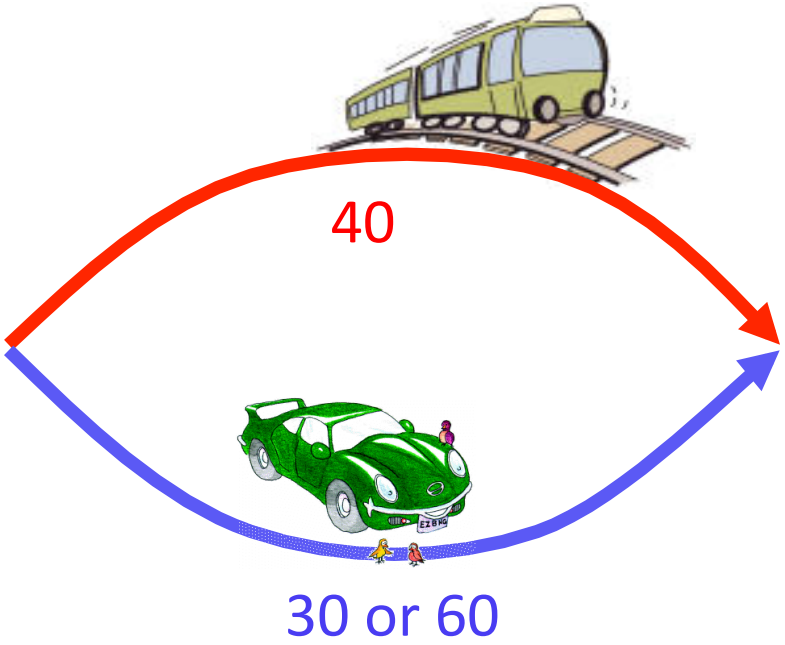
Take-away points on congestion games

- Shortest paths (and algorithmic questions in general) can get complicated due to presence of more people
- Equilibrium and Social Optimum in nonatomic routing games exist and can be found efficiently via convex programs.
- Social optimum is an equilibrium with respect to modified latencies = original latencies plus toll.
- Price of anarchy: 4/3 for linear delays, can be found similarly for more general classes of delay functions.

Part III: Risk-averse decision making

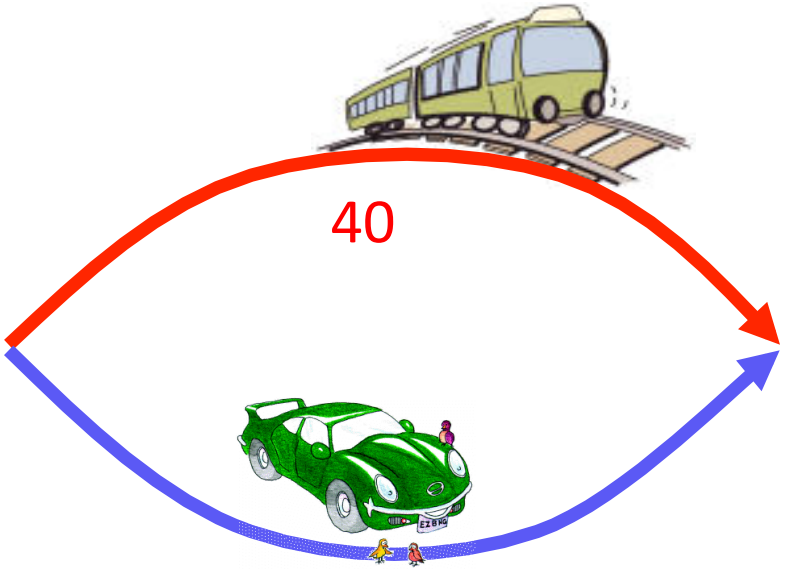
Optimal route?

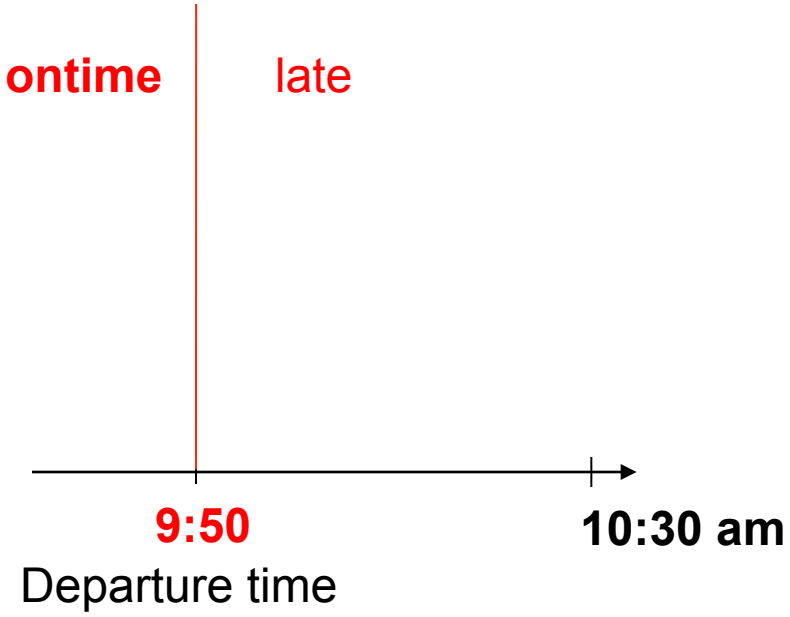
- Optimal route may differ with start time



Optimal route?

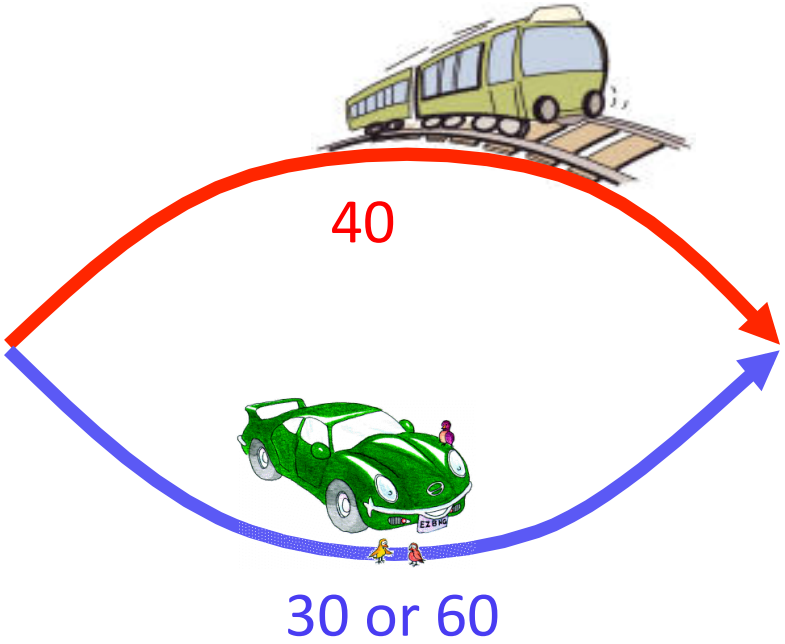
- Optimal route may differ with start time

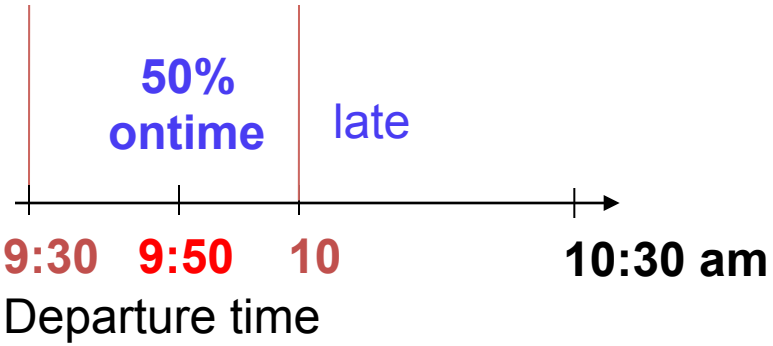




Optimal route?

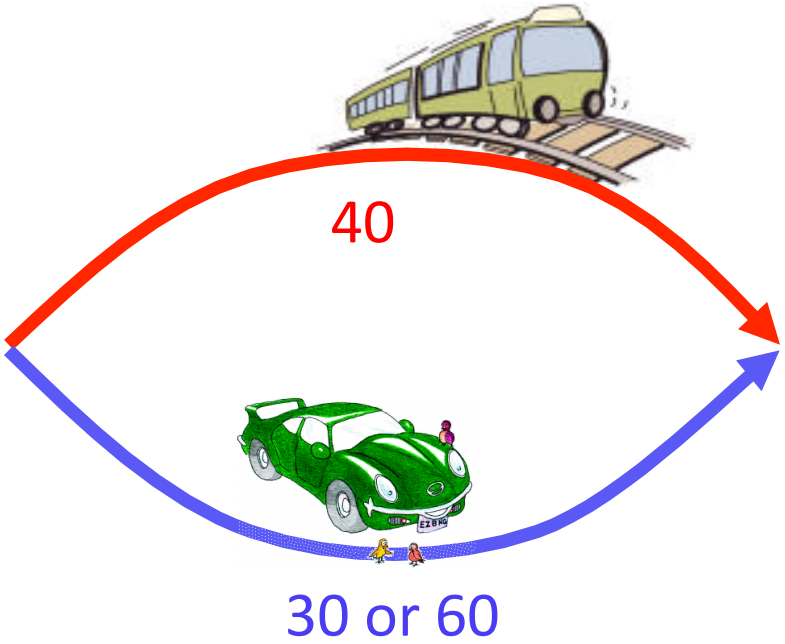
- Optimal route may differ with start time

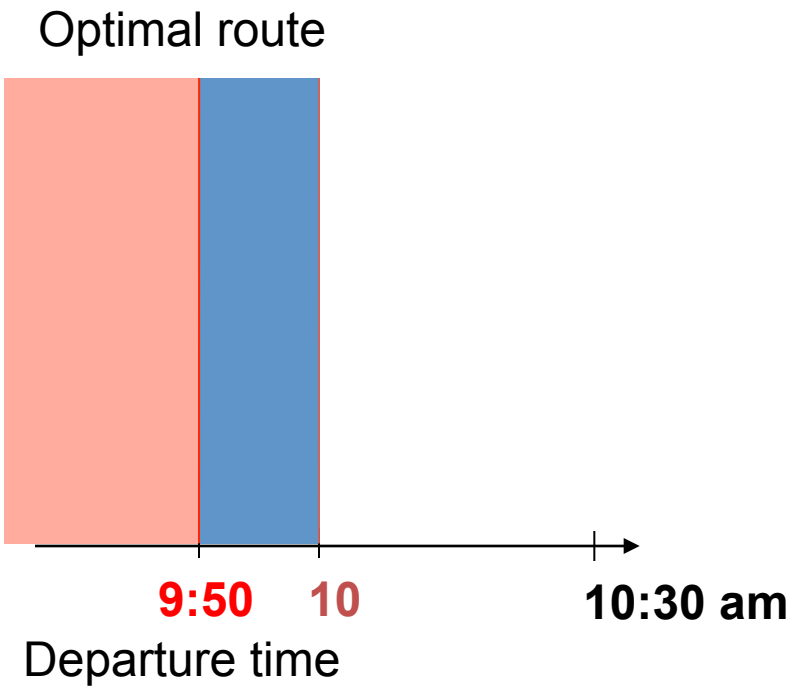




Optimal route?

- Optimal route may differ with start time





Implications of risk attitudes

- Uncertainty and risk can affect our design and analysis of:
- Algorithms
- Algorithmic Game Theory
- Algorithmic Mechanism Design

What is risk?

Three main quantitative approaches to defining risk:

- Economics: Expected utility theory and alternatives
- Finance: Markowitz mean-variance framework
- Modern risk theory: axiomatic approach to defining risk (coherent & convex risk measures)

- Systemic risk: will not talk about it today. See **"A Survey of Systemic Risk Analytics" by Bisias, Flood, Lo, Valavanis, 2012.**
- Dynamic / multi-period risk

Risk I: Expected Utility Theory

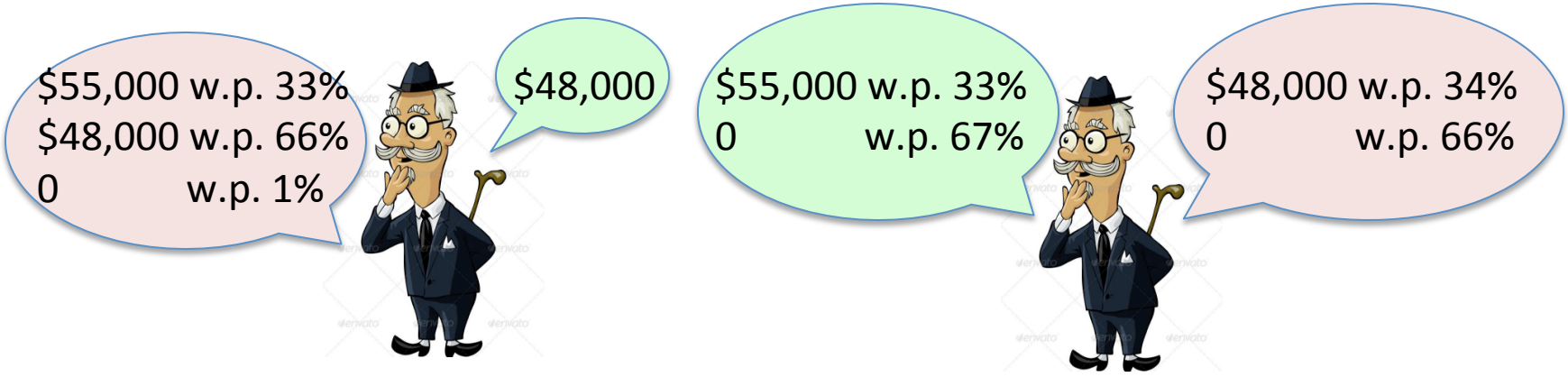
- Express preferences over random variables (lotteries) by mapping each to a single number through a "utility function". [Bernoulli 1738]

Risk I: Expected Utility Theory

- Express preferences over random variables (lotteries) by mapping each to a single number through a "utility function". [Bernoulli 1738]
- Utility exists assuming independence, continuity axioms

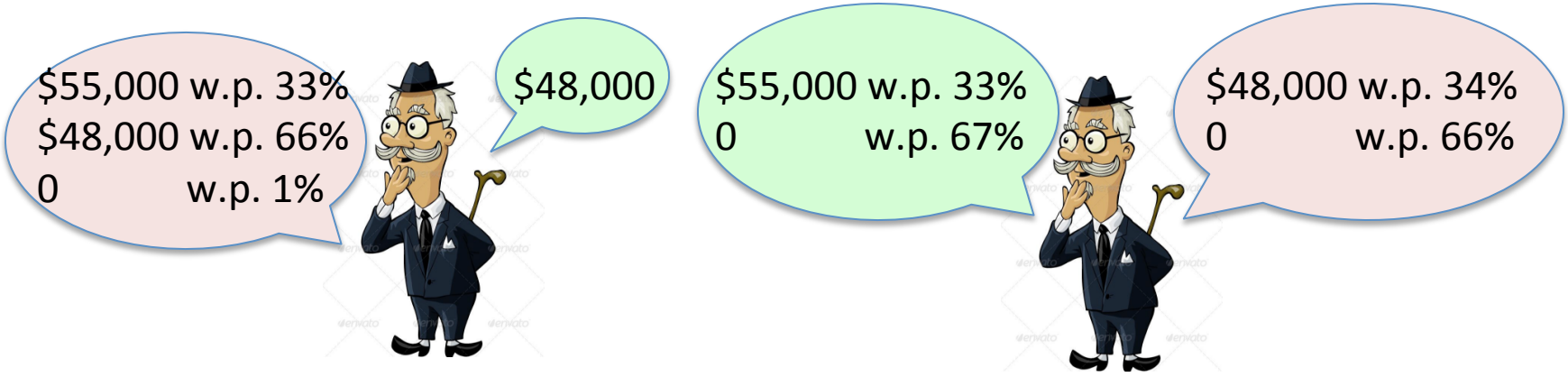
Risk I: Expected Utility Theory

- Express preferences over random variables (lotteries) by mapping each to a single number through a "utility function". [Bernoulli 1738]
- Utility exists assuming independence, continuity axioms
- **Experiment:**



Risk I: Expected Utility Theory

- Express preferences over random variables (lotteries) by mapping each to a single number through a "utility function". [Bernoulli 1738]
- Utility exists assuming independence, continuity axioms
- **Experiment:**



- Convex utility => risk-loving preference
- Concave utility => risk-averse preference

Risk II: Mean-variance framework

- In investment science, Markowitz (1952) identified risk with the variance of a portfolio
- E.g. routing under uncertainty:

  minimize (mean + standard deviation) of travel time

Related:

  minimize variance s.t. mean is at most a threshold

  minimize trip budget s.t. Prob (late) < 5% [value at risk]

- Criticisms
  - Non-monotonicity (may prefer stochastically dominated solutions)
  - Non-convexity (conflicts risk-diversification?)

Risk III: Coherent risk measures

- Axiomatic approach to the construction of risk measures (Artzner et al. 1999)
  - **A1 [Monotonicity]:** $X \geq 0$ implies $R(X) \leq 0$
  - **A2 [Convexity]:** $R(\beta X + (1-\beta) Y) \leq \beta R(X) + (1-\beta) R(Y)$
  - **A3 [Positive homogeneity]:** $R(\beta X) = \beta R(X)$
  - **A4 [Translation invariance]:** $R(X+c) = R(X) - c$ for all real c

- **Example:** Conditional Value at Risk, $CVaR_\alpha(X)$: the conditional expectation of losses that exceed $VaR_\alpha(X)$; X is continuous.

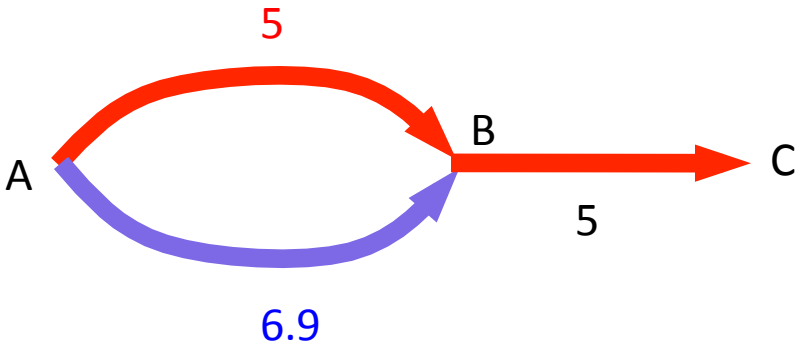
- **Remark**: when X is normally distributed, both $VaR_\alpha(X)$ and $CVaR_\alpha(X)$ are equal to E[X] – k Stdev(X) for appropriate constants k.

Implications of risk attitudes

- Uncertainty and risk can affect our design and analysis of:
- Algorithms
- Algorithmic Game Theory
- Algorithmic Mechanism Design

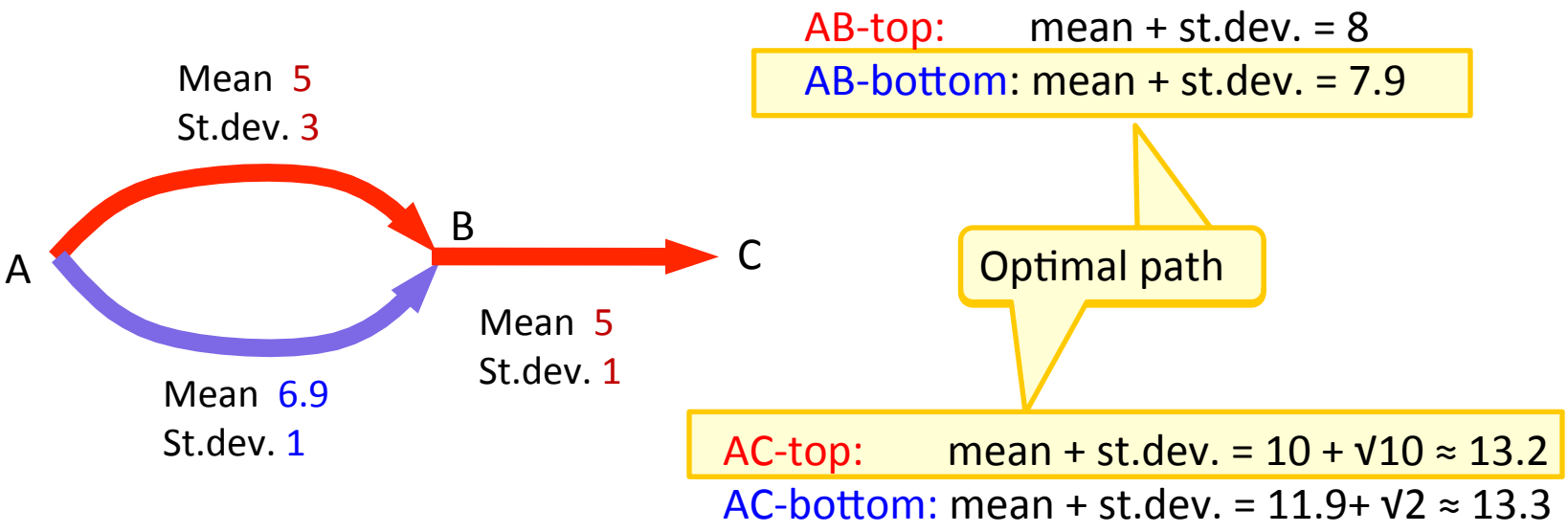
Algorithmic challenges

- E.g., In shortest paths, *optimal substructure (additivity)* property allows for efficient dynamic programming algorithms (Dijkstra, Bellman-Ford, etc)



Algorithmic challenges

- Nonadditivity: in risk-averse shortest paths, *optimal substructure fails.*



- Non-convex (concave) objective

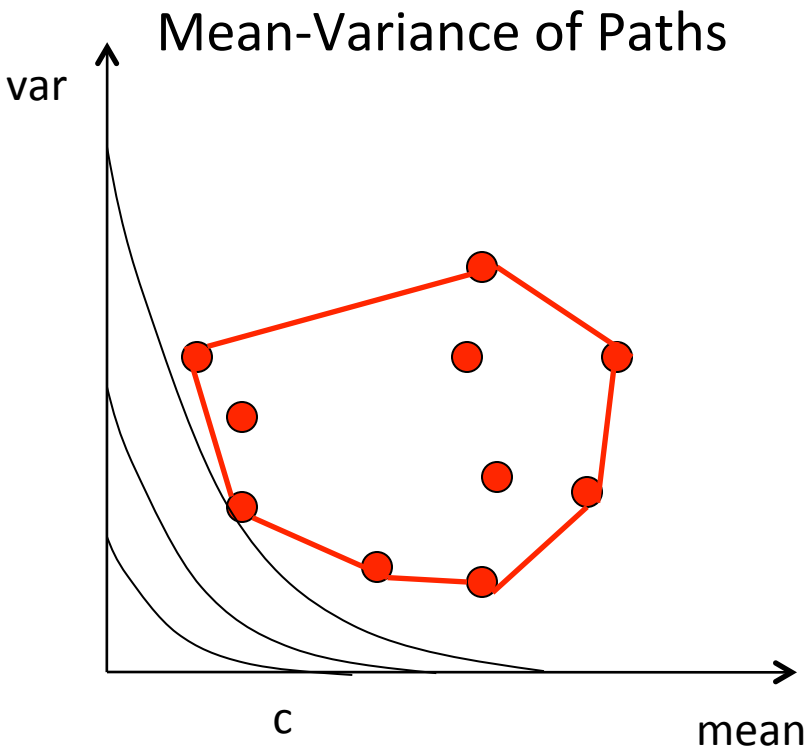
Algorithmic insights

Insight 1: Concave objective

min path mean + r √path var {paths}

Insight 2: Visualize on mean-variance plane

Insight 3: Solution is an extreme point on mean-variance frontier



Exact algorithm for risk-averse shortest path has runtime n^O(log n).
[Nikolova, Kelner, Brand, Mitzenmacher 2006]

OPEN: Is risk-averse shortest path in P? On planar graphs?

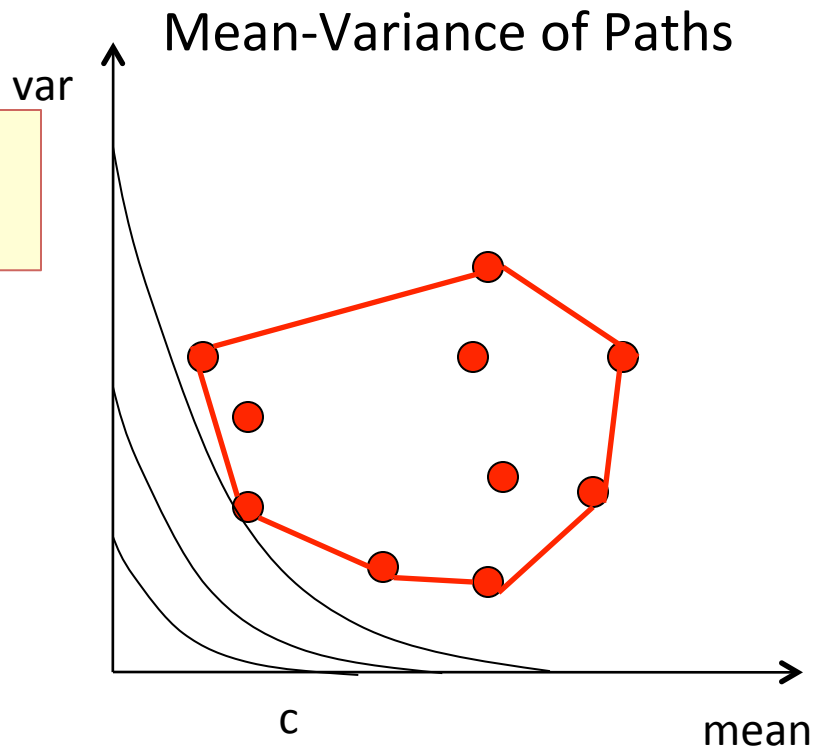
Algorithmic insights

Insight 1: Concave objective

min risk-averse function {feas.set}

Insight 2: Project on mean-variance plane

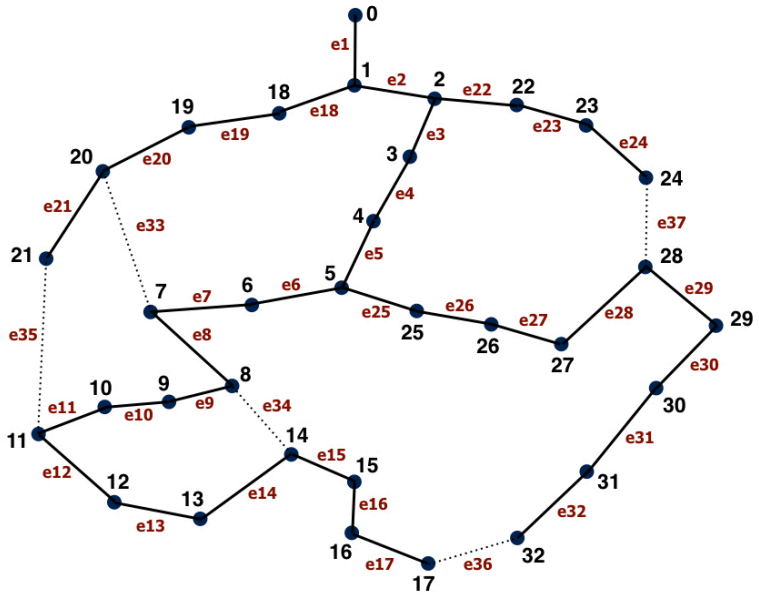
Insight 3: Solution is an extreme point



There is an (1+ε)-approximation algorithm for risk-averse problem that uses poly(|input|, 1/ε) queries to exact algorithm for corresponding deterministic problem. [Nikolova 2010]

Algorithmic challenges

- Challenge is nonlinear objective function over combinatorial feasible set
- Example from electricity grid



$$\min_{ST} \sum_{e \in ST} R_e \left[\left(\sum_{i \in c_e} p_i \right)^2 + \left(\sum_{i \in c_e} q_i \right)^2 \right]$$

Reactive power

Active power

Resistance

Successors of edge e in spanning tree ST

Implications of risk attitudes

- Uncertainty and risk can affect our design and analysis of:
- Algorithms
- Algorithmic Game Theory
- Algorithmic Mechanism Design

Risk sensitivity of price of anarchy

- [Piliouras, Nikolova, Shamma 2013] consider routing games with uncertain delays resulting from "uniform schedulers"
- Price of anarchy of linear congestion games under risk attitudes:
  - Wald's minimax cost 2
  - Savage's minimax regret [4/3, 1]
  - Minimizing Expected cost 5/3
  - Average case analysis 5/3
  - Win-or-Go-Home unbounded
  - Second moment method unbounded

- **Conclusion: Risk critically affects predictions of system performance**

- Related work on risk-aversion in routing games: Ordóñez & Stier-Moses'10, Boyles-Kockelman-Waller'10 (tolls), Nikolova & Stier-Moses'11-'14, '15, Nie'11, Angelidakis-Fotakis-Lianeas'13, Cominetti-Torico'13, Meir-Parkes'15, Lianeas-Nikolova-Stier-Moses'16, etc.

Effect of Risk vs Selfish Behavior?

- Cost of traffic pattern $C(x)$: central planner is risk-neutral (although users are risk-averse), so we consider the sum of *expected travel times*
- Price of Risk Aversion: captures inefficiency introduced by user risk-aversion w.r.t. to risk-neutral users

$$\sup_{\text{problem instances}} \frac{C(x^r)}{C(x^0)}$$

risk-averse equilibrium

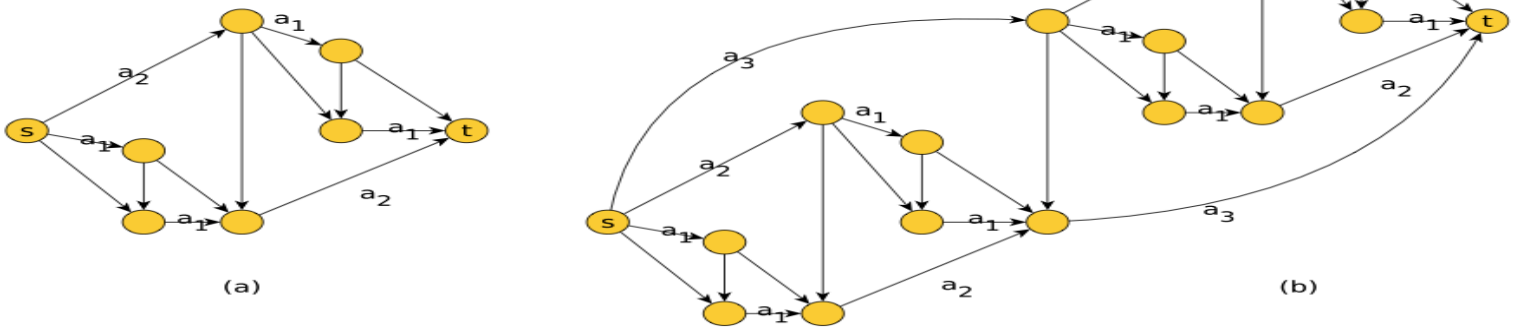
risk-neutral equilibrium

Risk-averse selfish routing*

- In graphs with general mean, variance functions where users minimize (mean + r*variance):

Cost(Risk-averse eq.) ≤ (1+ηrk) Cost(Risk-neutral eq.)

- η=1 for series-parallel graphs, η=2 for Braess graph, η≤ |V|/2 for a general graph
-



* Lianeas, Nikolova, Stier Moses. "**Risk-averse selfish routing.**" Forthcoming in Mathematics of Operations Research.

Risk-averse selfish routing*

- In graphs with general mean, variance functions where users minimize (mean + r*variance):

Cost(Risk-averse eq.) ≤ (1+ηrk) Cost(Risk-neutral eq.)

- $\eta=1$ for series-parallel graphs, $\eta=2$ for Braess graph, $\eta \leq |V|/2$ for a general graph

- Above bound is tight for general graphs.
- Bound can also be found w.r.t latency function classes:

Price of risk aversion **≤ (1+rk) PoA**

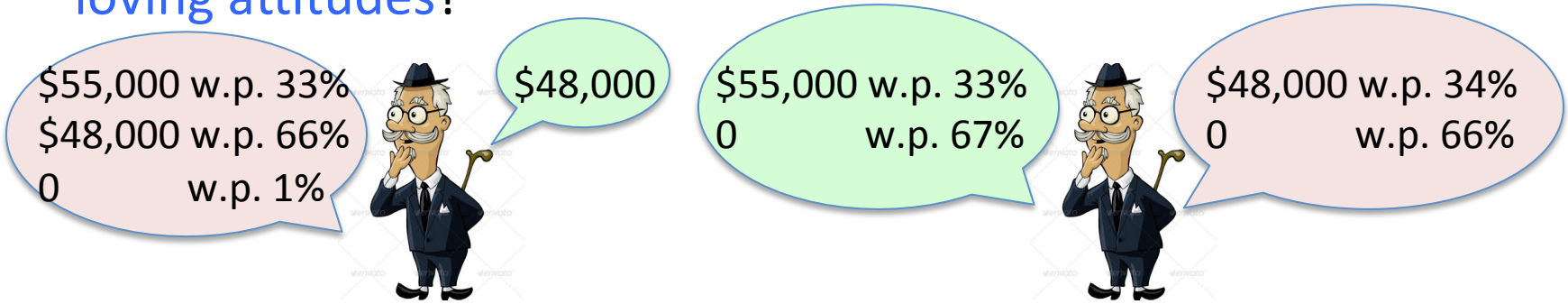
* Lianeas, Nikolova, Stier Moses. "**Risk-averse selfish routing.**" Forthcoming in Mathematics of Operations Research.

Implications of risk attitudes

- Uncertainty and risk can affect our design and analysis of:
- Algorithms
- Algorithmic Game Theory
- Algorithmic Mechanism Design

Effect of risk on mechanism design

- People respond better to lotteries (examples in transportation, energy, recycling, etc.)
- What are the optimal lotteries? I.e. what is the optimal mechanism design if people have given risk-averse or risk-loving attitudes?



- Work in progress on risk-loving mechanism design with Manolis Pountourakis, Ger Yang at UT Austin.
- Want to know more? Talk to Manolis (Simons Fellow)

Conclusion

- CS theorists like to prove theorems (and have guarantees on algorithm runtimes, etc)
- CS theorists like "clean formulations" that abstract away many engineering details so as to find and understand core algorithmic challenges (reduce to toy puzzles)

- Open questions?
  - Dynamic / Adaptive models? (streaming algorithms, learning algorithms for real-time decision making)