Heuristics

- What are they?
- Why use them?
- Different kinds of Heuristics
- Examples of strengths and weaknesses

What is a Heuristic?

- Any method that is not mathematically proven to find an optimal solution.
- Does not guarantee an optimal solution
- Generally give answers
  - faster than exact methods
  - or with less coding effort

Why Use Heuristics

- Obvious reasons
  - Faster answers, less work
- Less Obvious reasons
  - Data is approximate or uncertain
  - Model is approximate
  - Gaps between recommendation and implementation
    - Why get exact answers to an inexact problem?

Why Not Solve Exactly

- Why not solve inexact problem exactly
  - Exact optimum may be very sensitive to the data
    - if the data are off a little, the exact answer could be off by a lot
  - Exact optimum may be more sensitive to disruption
    - Exact optimum may fail dramatically when faced with a local disruption

We always use heuristics

- Optimizing component behavior rarely optimizes system behavior
- Example:
  - Purchasing minimizing piece cost
  - Transportation minimizing freight cost
  - Manufacturing minimizing changeover costs
  - …

Fundamental Heuristics

- Hierarchical approach to systems
  - Strategic: Setting Direction
  - Tactical: Plotting a course
  - Operational: Staying on course
- Hierarchical solution strategies
  - Ford Finished Vehicles
    - Allocate dealers to ramps
    - Route vehicles to ramps
  - Airlines
    - Assign body types to flight legs
    - Build routes for the planes
    - Build routes for the crews

Basic Forms

- Construction heuristics
  - Build a solution by making a sequence of good (local) decisions
- Improvement heuristics
  - Improve an existing solution by making a sequence of good (local) modifications to an existing solution

Examples

- Traveling Salesman Problem
  - Visit each city once and return to the starting point at minimum cost
  - Applications:
    - Routing
    - Robotic assembly
    - Production scheduling
    - …

Construction Heuristics

- Unalloyed Greed: Nearest Neighbor
  - Start somewhere
  - Go next to the closest as yet unvisited city
  - Nearest Neighbor Example
    (http://itp.nat.uni-magdeburg.de/~mertens/TSP/)
  - Fast, easy to implement, but
  - Myopic behavior leads to problems

Construction Heuristics

- Tempered Greed: Farthest Insertion
  - Start with a circumnavigating tour
  - Adjust the tour in the best way to include the worst excluded city
  - Farthest Insertion Example

    (http://itp.nat.uni-magdeburg.de/~mertens/TSP/)
  - Fast, easy to implement,
  - Tends to do better

Idea

- Construction heuristics are fast
- Temper myopic behavior with global perspective
- Dangers
  - It's not always easy to find a feasible solution!
  - Obvious improvements possible

Improvement Heuristics

- Start with a feasible solution and iteratively improve it.
- Generally hard to spot improvements to the resulting solutions
- Generally fast, but this depends
- Not always best to begin with a good solution

Improvement Heuristics

- 2-Opt: Break the tour and glue it back in a better way
  - Start with a tour
  - Break two edges
  - Put the pieces together in the best way
  - 2-Opt Example

(http://itp.nat.uni-magdeburg.de/~mertens/TSP)

Advanced Improvement

- Finding an improvement might be a (smaller, faster, easier) optimization problem:
  - Machine Scheduling:
    - Start with a schedule
    - Remove one job and reinsert it into the schedule in the best way
    - Repeat

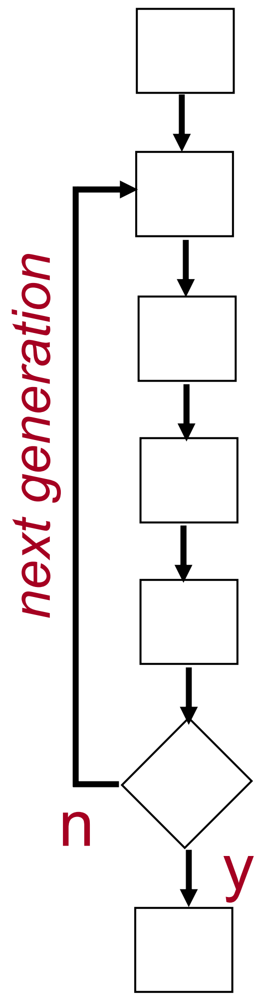
Local Optima

- Construction and Improvement Heuristics find local optima.
- Increasing the neighborhood to look over avoids some of these but increases the time required
  - 3-Opt
  - K-Opt
  - Reschedule 2 jobs at a time, …

Avoiding Local Optima

- Randomized methods
- Simulated annealing
  - move to a worse solution with some probability
  - this probability declines with time
- Genetic Algorithms
  - Using a population of feasible solutions

Genetic Algorithms



Initialize Population (initialization)

Select individual for mating (selection)

Mate individuals and produce children (crossover)

Mutate children (mutation)

Insert children into population (insertion)

Are stopping criteria satisfied ?

Finish

Genetic Algorithms

- GA Example
  (http://cs.felk.cvut.cz/%7Exobitko/ga/index.html)
- Quickly becoming an off-the-shelf tool
- Key to success for heuristics – quick implementation
- An implementation included with Premium Solver from Frontline Systems, Inc. (Included with text).

Eliminate the Problem!

- Self organizing mechanisms
- Examples:
  - Honey bee forager allocation
    - No one is in charge. The process works
  - Bucket brigades
  - http://www.isye.gatech.edu/people/faculty/John_Bartholdi/bucket-brigades.html

Honey Bees

- Allocate Foragers to flower patches
- Quality and locations change
- Survival depends on good answers
- No
  - Satellites
  - Cell phones
  - Supercomputers
  - Centralized control mechanism

Like the Ants

- Waggle dance advertises site
- Faster bees dance more often
- Assume:
  - More bees gather more quickly in total
  - But more slowly on average
- Converge to equal time allocation
  - Every active site takes the same time to collect from

This Answer is Good

- In the worst case, with satellites, cell phones and supercomputers, bees could do at most twice as well
- In the likely case (specific reasonable models of cost) it's more like 5% better
- The analytical approach of setting derivatives to 0 can be arbitrarily bad

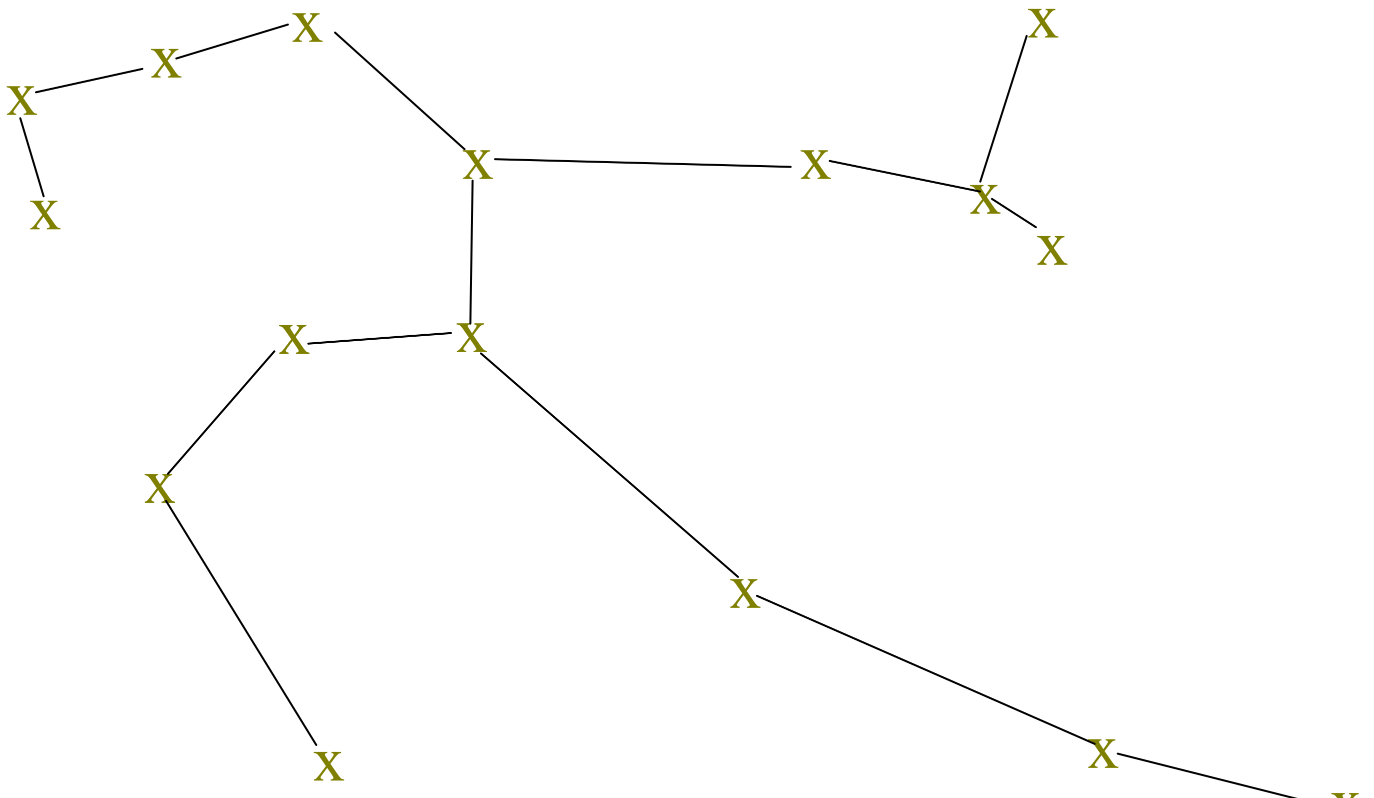
Performance Guarantees

- Some heuristics do come with guarantees
- The answer from the heuristic is within X% of the best answer
- Value: reassuring
- Disadvantage: Require assumptions or simplification

A Heuristic with guaranteed performance for The TSP

- A Spanning Tree
  - Use the Greedy Algorithm
    - Add edges in increasing order of length
    - Discard any that create a cycle
- Is a Lower bound on the TSP
  - Drop one edge from the TSP and you have a spanning tree
  - It must be at least as long as the minimum spanning tree

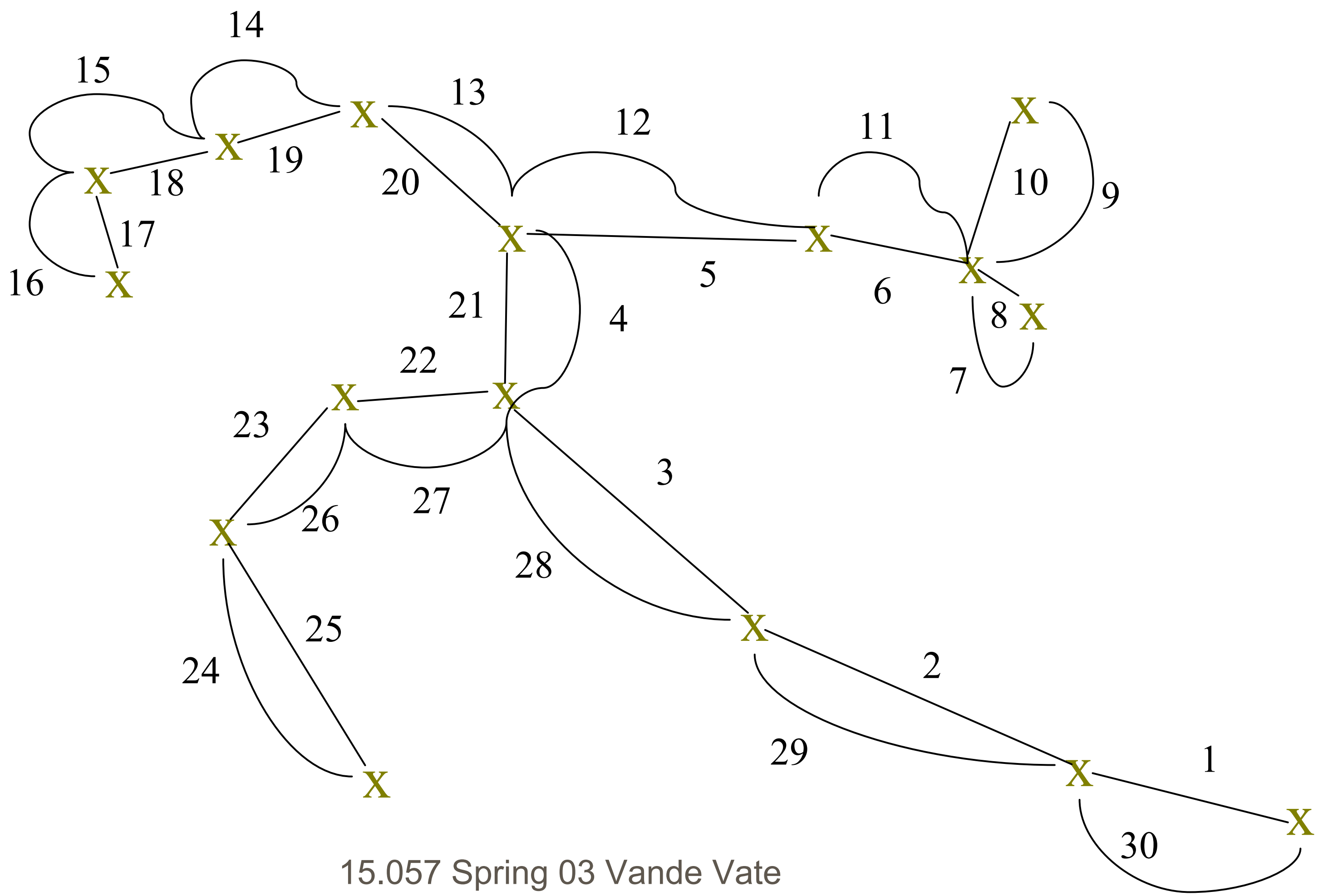
The Spanning Tree



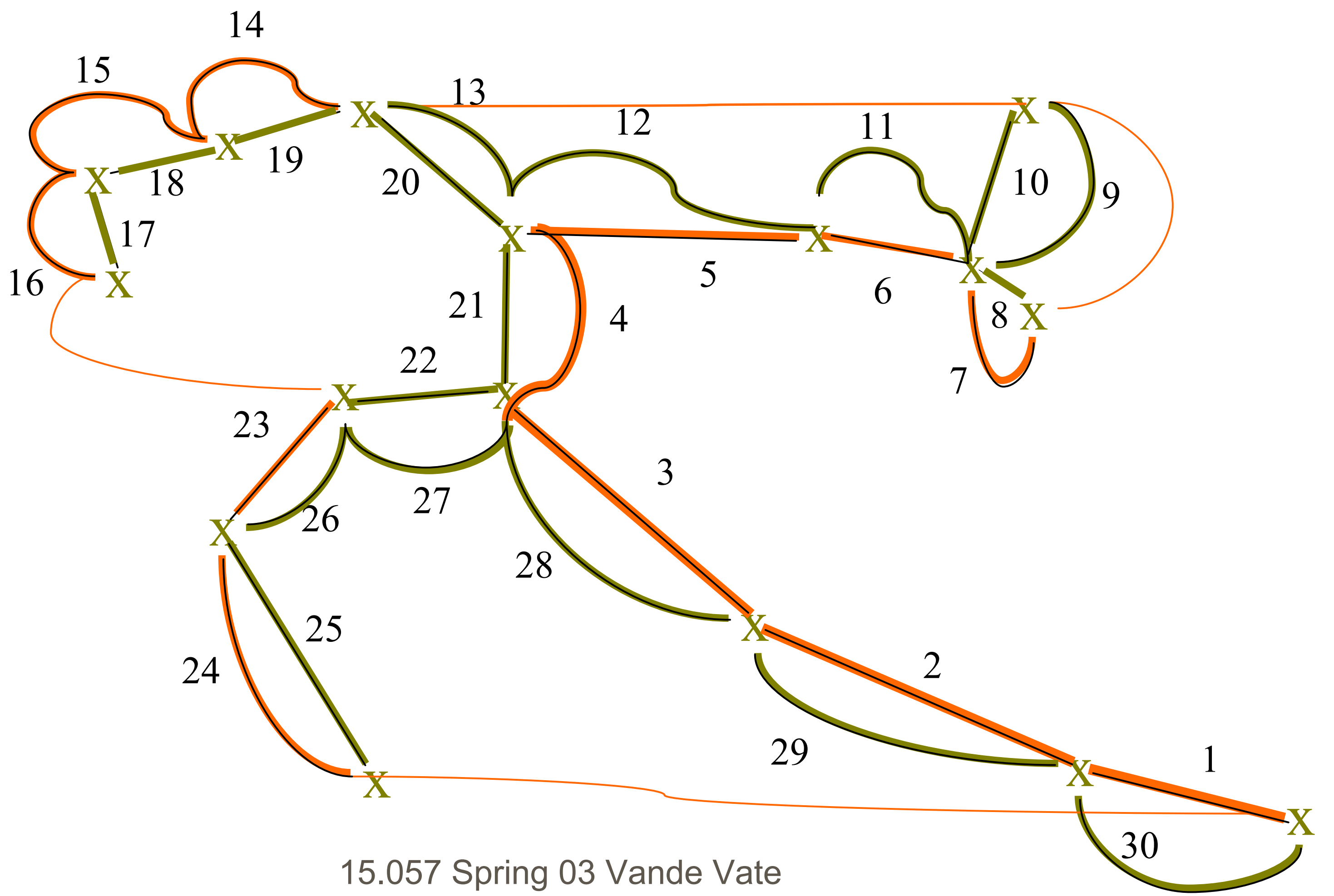
Double the Spanning Tree

- Duplicate each edge in the Spanning Tree
- The resulting graph is connected
- The degree at every node must be even
- That's an Eulerian Graph (you can start at a city, walk on each edge exactly once and return to where you started)
- It's no more than twice the length of the shortest TSP

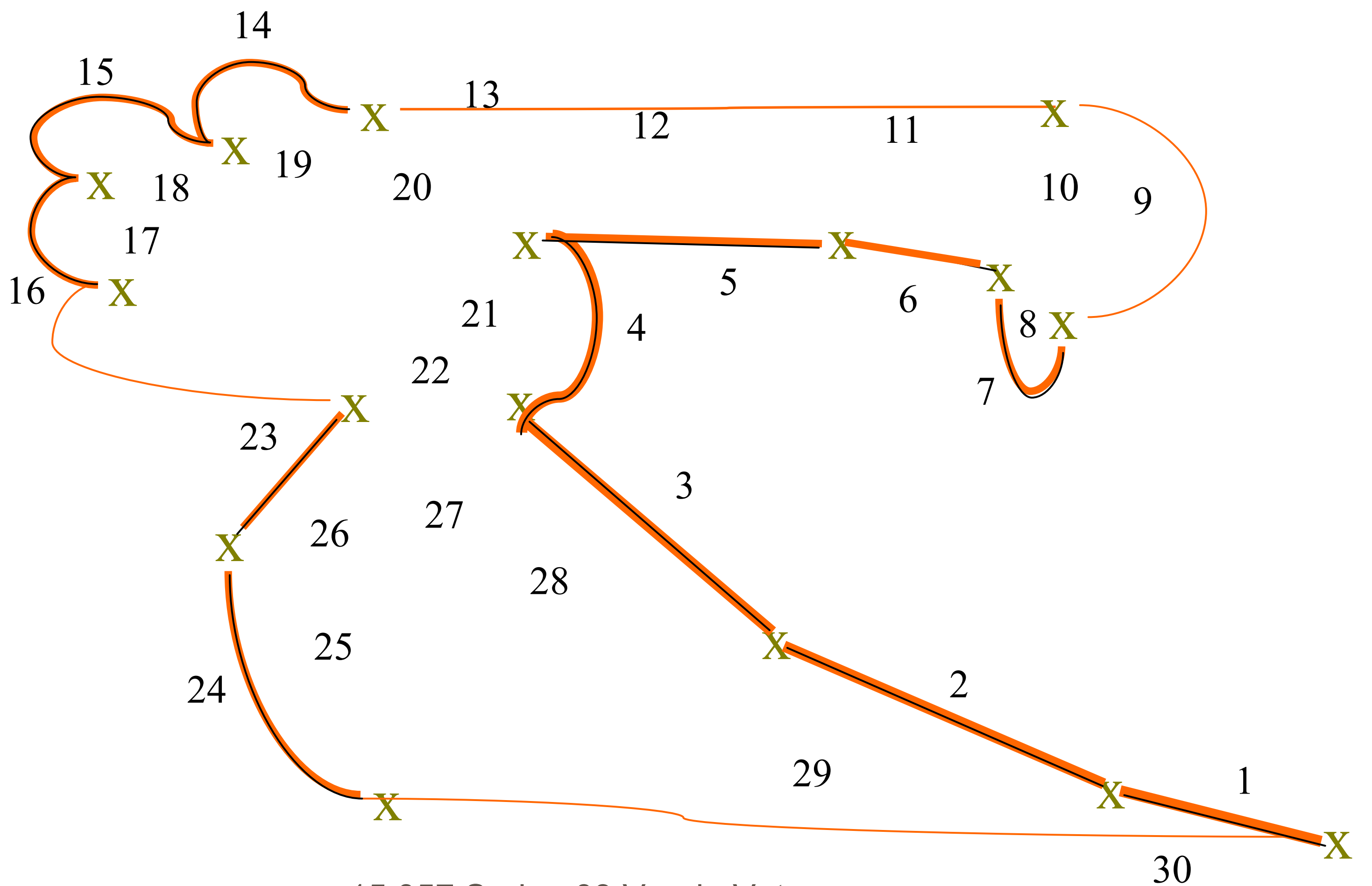
The Doubled Spanning Tree



Short Cut the Eulerian Tour



TSP Tour: at most 2X Optimum



A Novel Engineering Guarantee

- Aircraft loading
- Pallets of different weights and lengths to place on an aircraft.
- Want the center of gravity close to a specified point

The Heuristic

- Heuristic: Place the least dense remaining pallet as far as possible to the target center of gravity given the positions of the previously positioned pallets
- A sort of tempered greed

The Guarantee

- Guarantee: Produces a loading whose center of gravity is either within ½ the length of the longest pallet from the target, or barring that, as close as possible
- Comment: Even an optimal procedure can't give a stronger guarantee across all loads.

Design Implications

- Design for the worst load
- Theoretically, the worst load is ½ the longest pallet
- This approach guarantees it.
- The value of performance guarantees

Eliminate the Problem

- Toyota Sewing Systems
- Balancing workload in low-tech assembly systems
- Partitioning picking area among workers to improve throughput
- Optimization-based approach:
  - Detailed work-content models for each pick
  - Optimization routines to balance work-content among pickers
  - Team of Engineers to maintain the system

Self-Organizing Approach

- Bucket Brigades
- Water source = new work
- Fire = completed work
- Put fastest workers closer to the fire
- Run a bucket brigade
  - When a worker completes a job he walks back and takes over the next job from the previous worker…

Performance

- Self Balancing
  - The workers converge to a partition of the work that balances the workload
  - Performance generally improves significantly
- Bucket Brigades
- http://www.isye.gatech.edu/people/faculty/John_Bartholdi/bucket-brigades.html

Something completely different

- Problems in low dimensions are relatively easy.
- Reduce problems in high dimensions to problems in low dimensions
- Based on fascinating mathematics

Spacefilling Curves

- There are no more points in the unit square than in the interval from 0 to 1!?

"Heuristic" Argument

- Each point (X,Y) on the map
  - X = .165
  - Y = .975
  - Space Filling Number - interleave digits
  - θ(X,Y) = .**196755**

So,...

- Each pair of points

$$X = .165$$

$$Y = .975$$

maps to a unique point

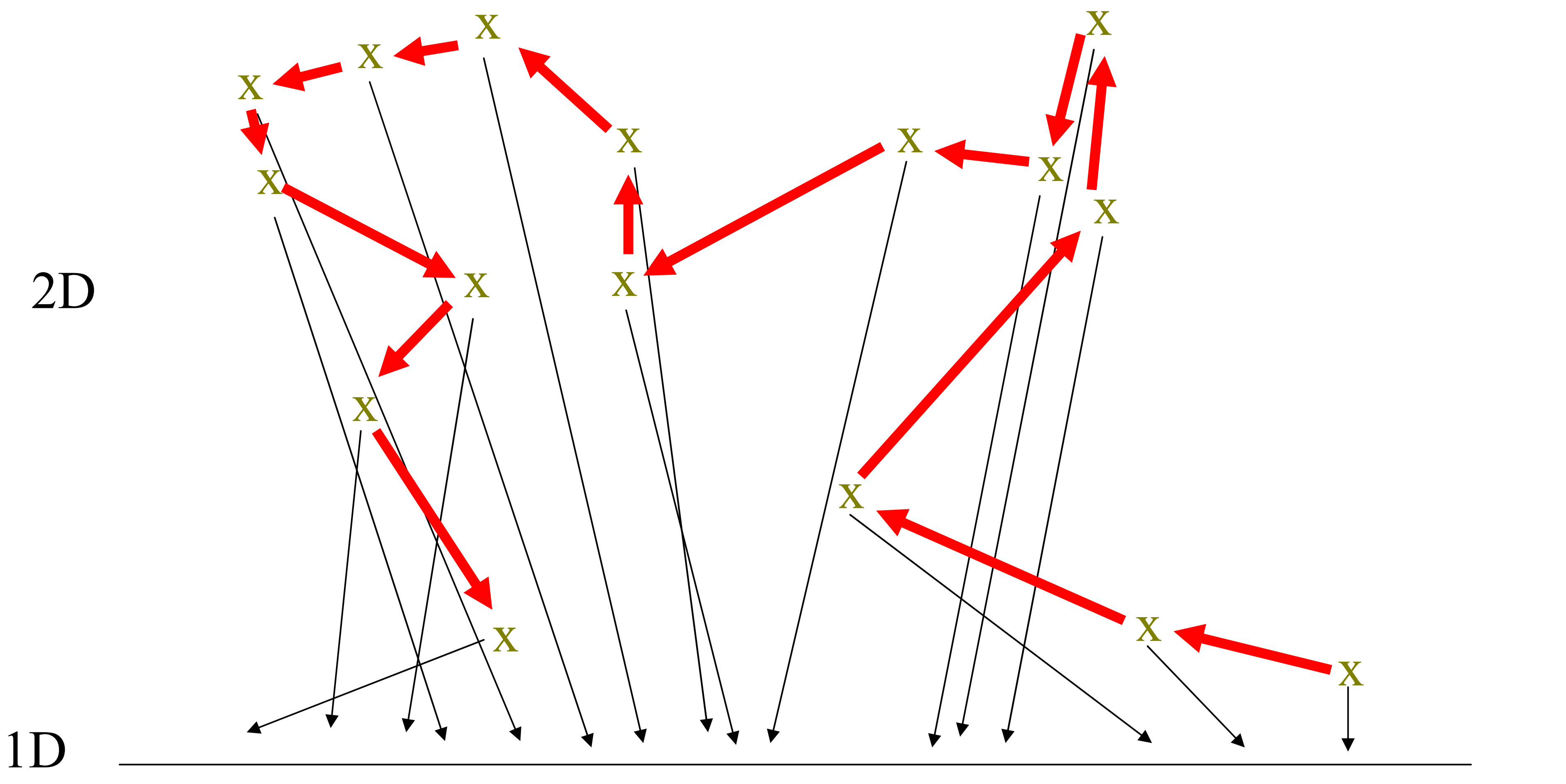
$\theta(X,Y) = .196755$

How to Use this

- Map a higher dimensional problem down to a lower dimensional problem
- Solve the problem in the lower dimensional space
- Interpret the answer in the higher dimensional space

TSP Example

- A mapping of $\theta(X,Y)$ into the unit interval
- Think of this as the inverse mapping of the unit interval onto the square (our super tour)
- For a given customer $\theta(X,Y)$ is the fraction of the way along the super tour where it lies
- Visit the customers in the order of $\theta(X,Y)$ (short cut the super tour to visit our customers)


2D
1D

Applications

- Used to
  - route a plotter pen
  - index data for GIS
  - route drivers for Meals-On-Wheels
- target warheads with missile defense systems
- More on SpaceFilling Curves

  http://www.isye.gatech.edu/~jjb/