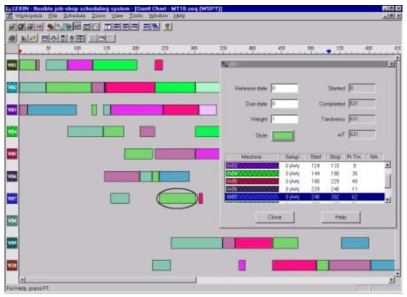
This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Large Scale Data Analysis for Policy
90-866, Fall 2012

Lecture 5: Representation and Heuristic Search

Representation and search

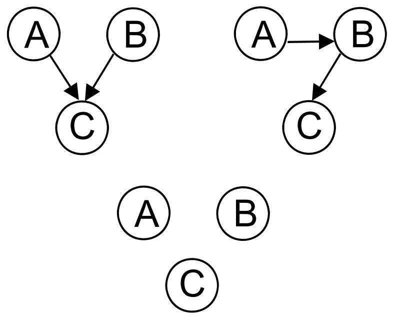
"Intelligent action, in either human or machine, is achieved through:
- **Representation** of a problem domain by patterns or symbols
- **Operations** on these patterns to generate potential solutions to problems
- **Search** to select among possible solutions"   (Newell and Simon, 1976)


Scheduling

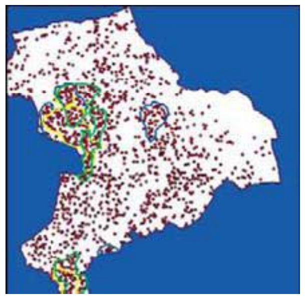
Route planning

Facility location

Autonomous navigation

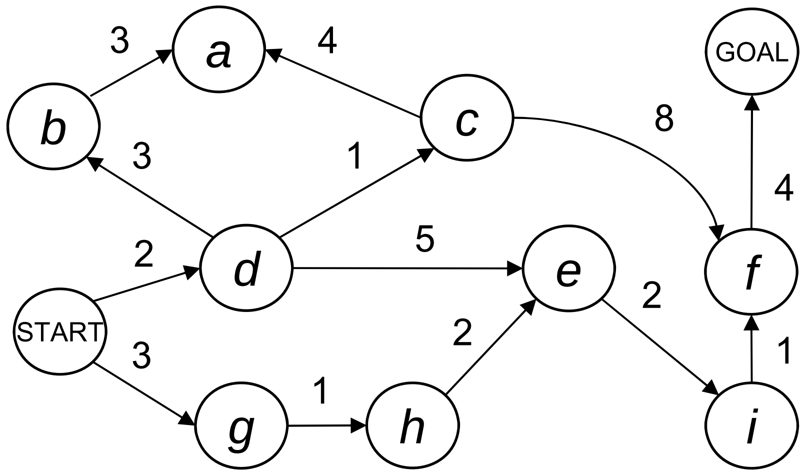


Modeling


Detection

Goal-directed search

… a traditional AI problem



How to get from START to GOAL with the least total cost?

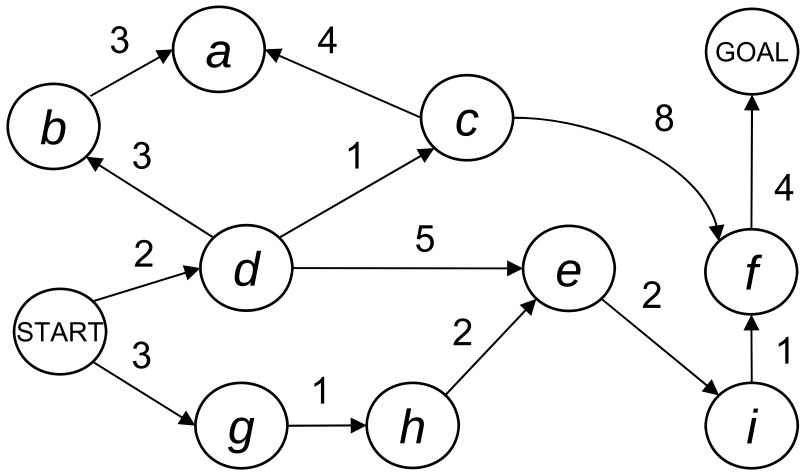
Route planning
Start = Squirrel Hill
Goal = CMU
Cost = driving time

Problem solving
Start = all on 2
Goal = all on 3
Cost = # of moves

Typical AI courses spend lots of time solving this problem, but many policy questions fall outside this simple framework, so we'll only discuss it briefly.

Goal-directed search is useful when we have a known goal state, known transitions and costs, and we wish to find the lowest-cost path to the goal.

Priority-based search



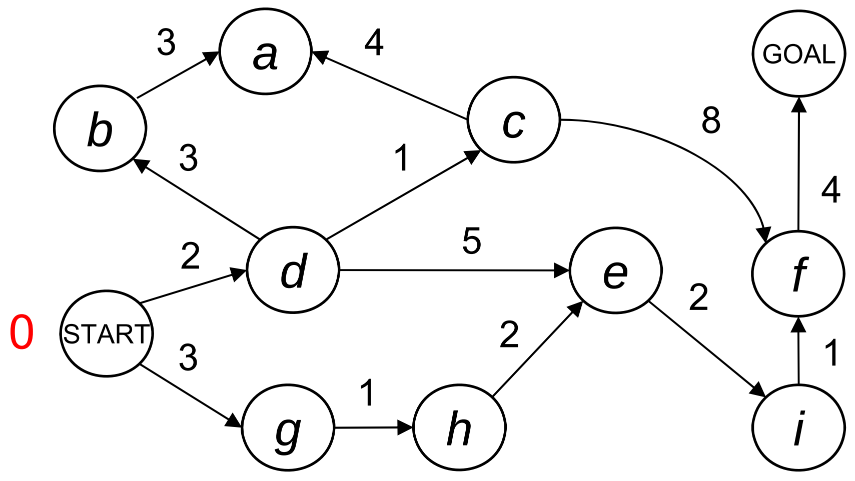
How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

Priority-based search



How to get from START to GOAL with the least total cost?

Queue:
(START, 0)

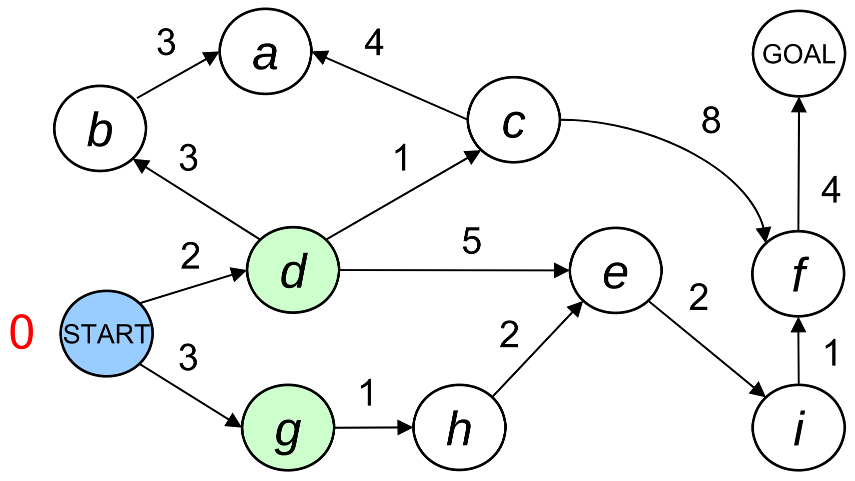
<u>One answer: priority-based search</u>

Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

Priority-based search



Queue:
(START, 0)

How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

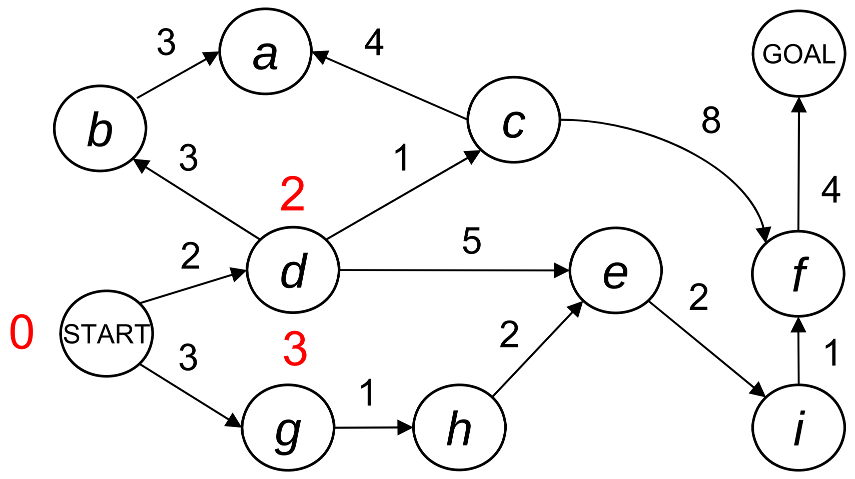
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(d, 2)
(g, 3)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

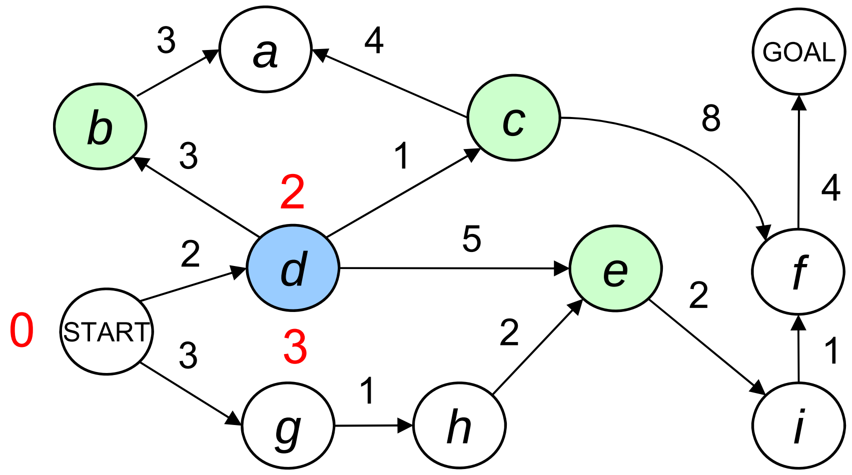
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(d, 2)
(g, 3)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

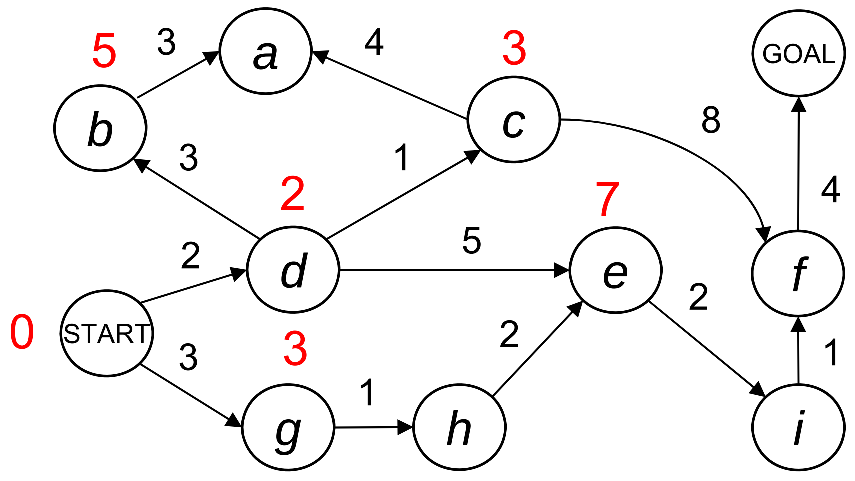
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

Queue:
(c, 3)
(g, 3)
(b, 5)
(e, 7)

One answer: priority-based search

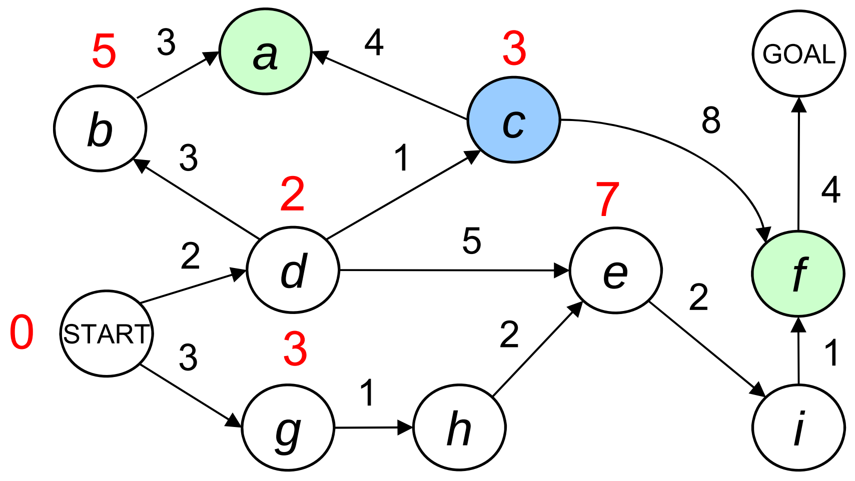
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



<u>Queue</u>:
(c, 3)
(g, 3)
(b, 5)
(e, 7)

How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

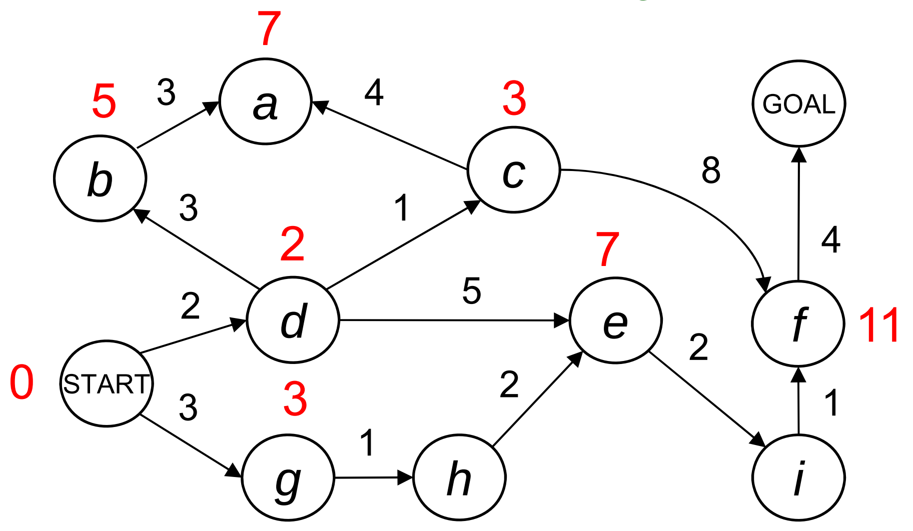
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(g, 3)
(b, 5)
(a, 7)
(e, 7)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

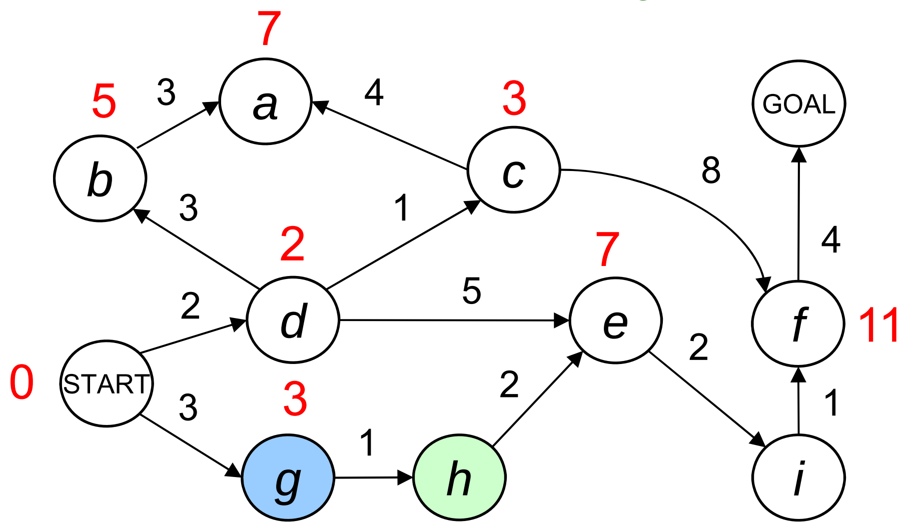
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(g, 3)
(b, 5)
(a, 7)
(e, 7)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

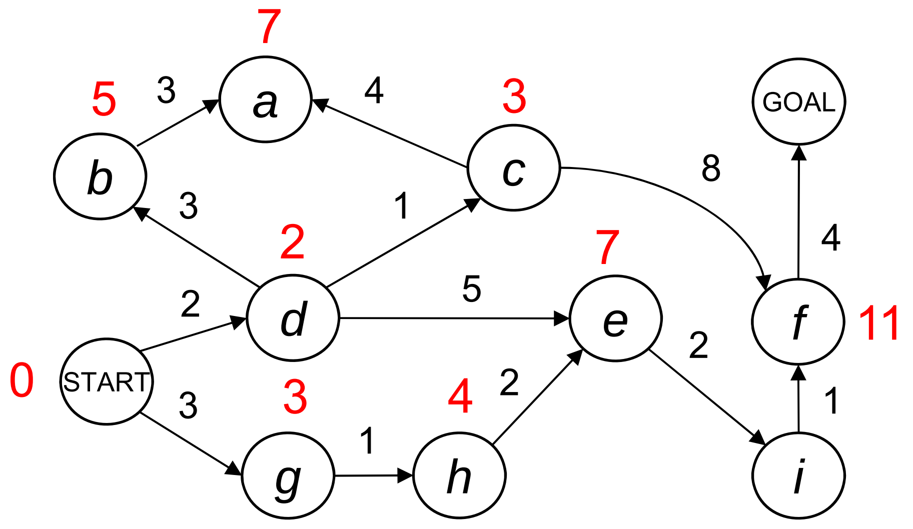
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(h, 4)
(b, 5)
(a, 7)
(e, 7)
(f, 11)

How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

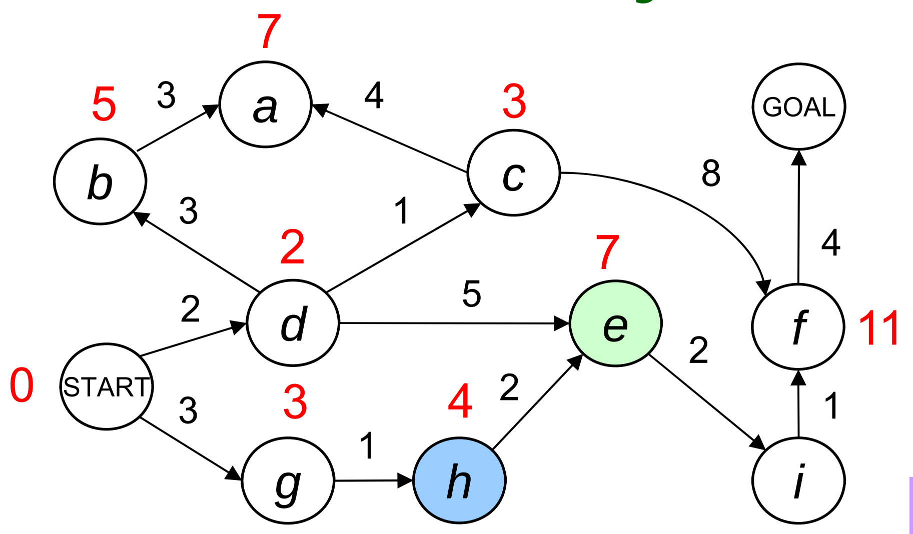
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

Queue:
(h, 4)
(b, 5)
(a, 7)
(e, 7)
(f, 11)

Notice that e is already on the queue, but we have found a lower-cost path.

One answer: priority-based search

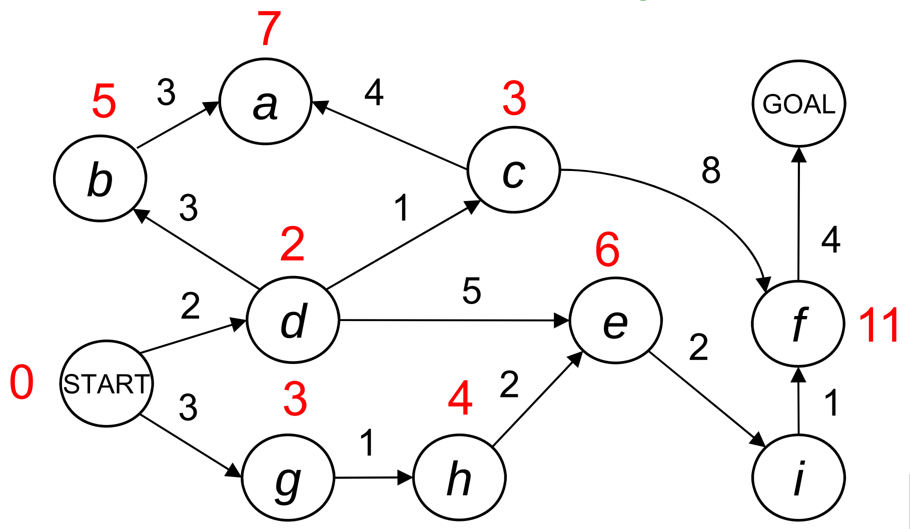
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

Queue:
(b, 5)
(e, 6)
(a, 7)
(f, 11)

We reduce the cost of node e from 7 to 6, and move it ahead of node a.

One answer: priority-based search

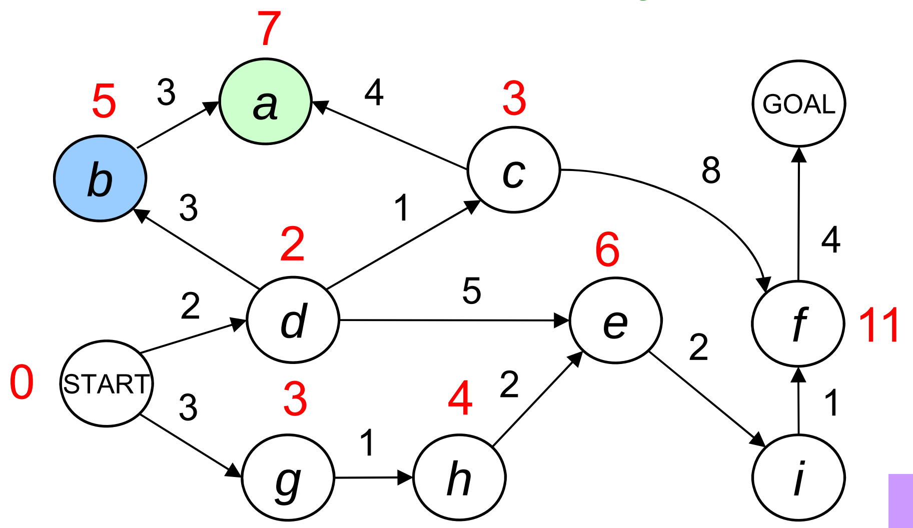
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

Queue:
(b, 5)
(e, 6)
(a, 7)
(f, 11)

Notice that a is already on the queue, but we have not found a lower-cost path, so we leave its priority unchanged.

One answer: priority-based search

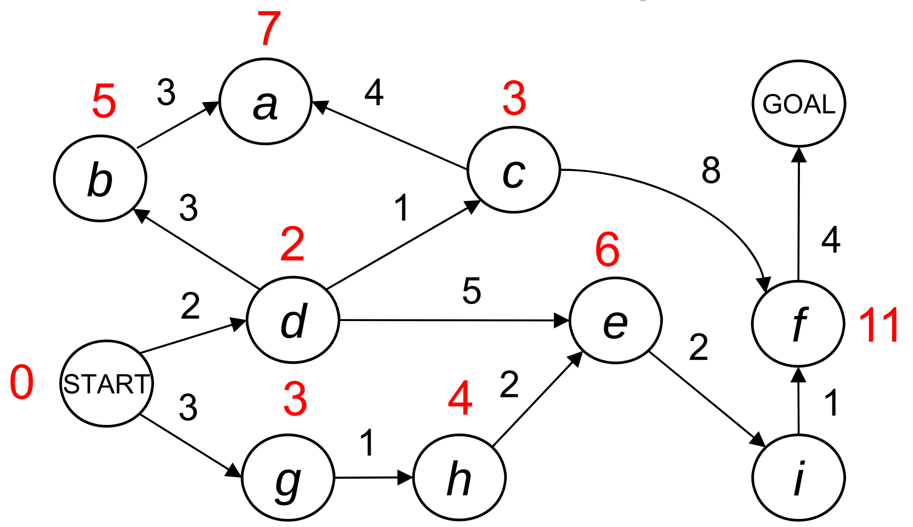
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(e, 6)
(a, 7)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

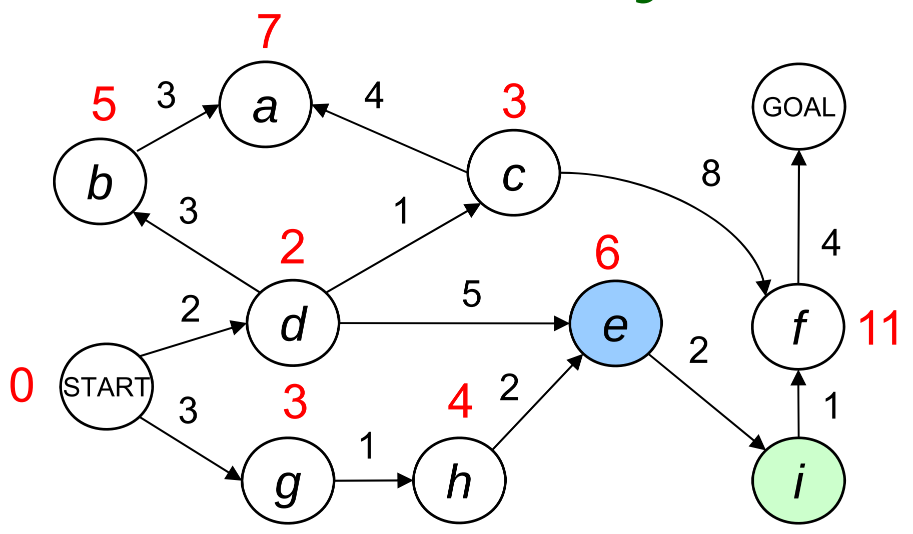
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(e, 6)
(a, 7)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

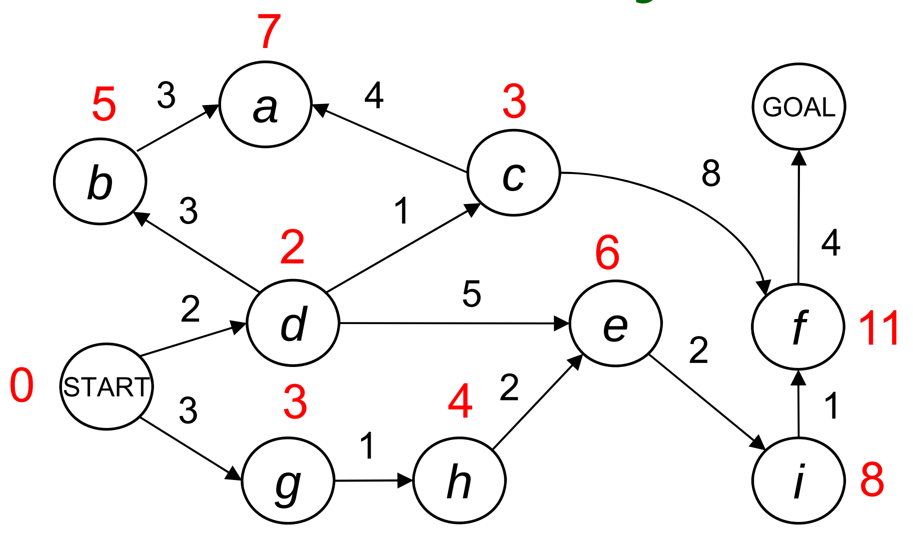
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(a, 7)
(i, 8)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

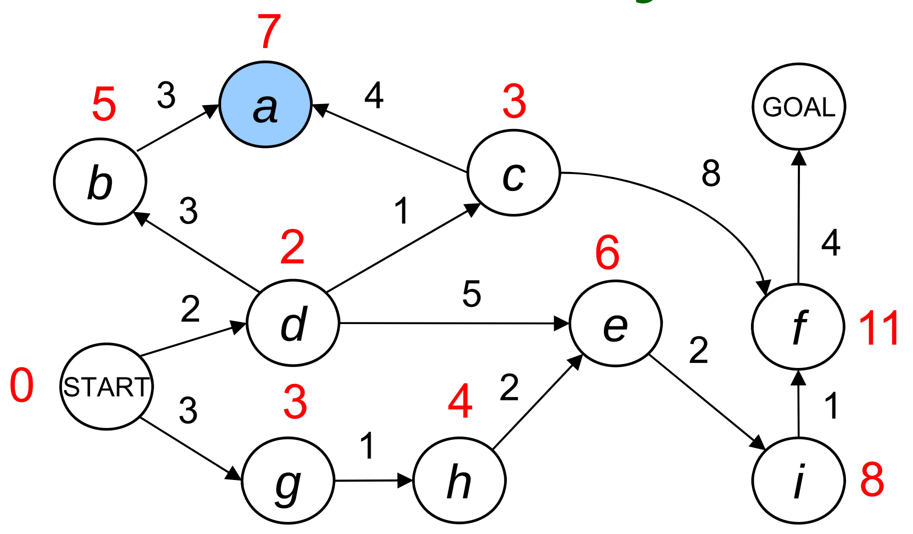
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(a, 7)
(i, 8)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

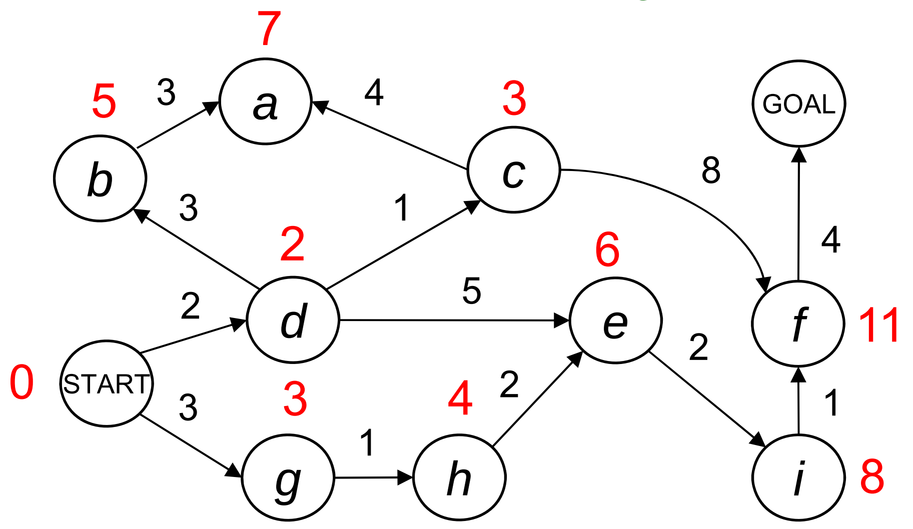
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(i, 8)
(f, 11)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

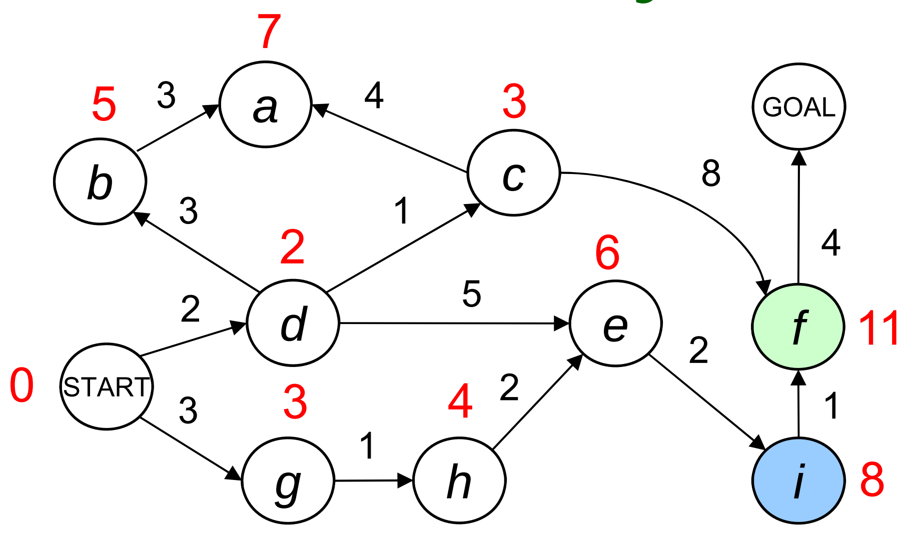
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

<u>Queue</u>:
(i, 8)
(f, 11)

We can reduce the cost of node f from 11 to 9.

<u>One answer: priority-based search</u>

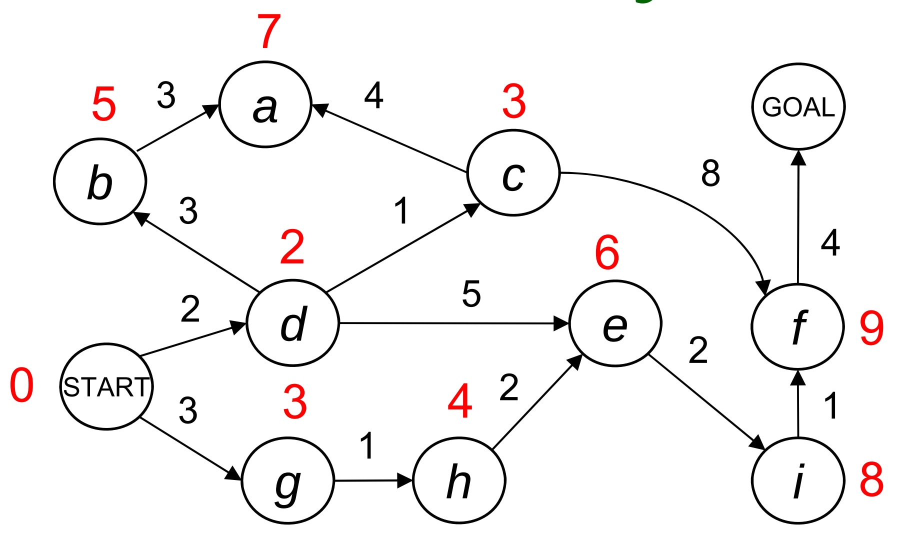
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(f, 9)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

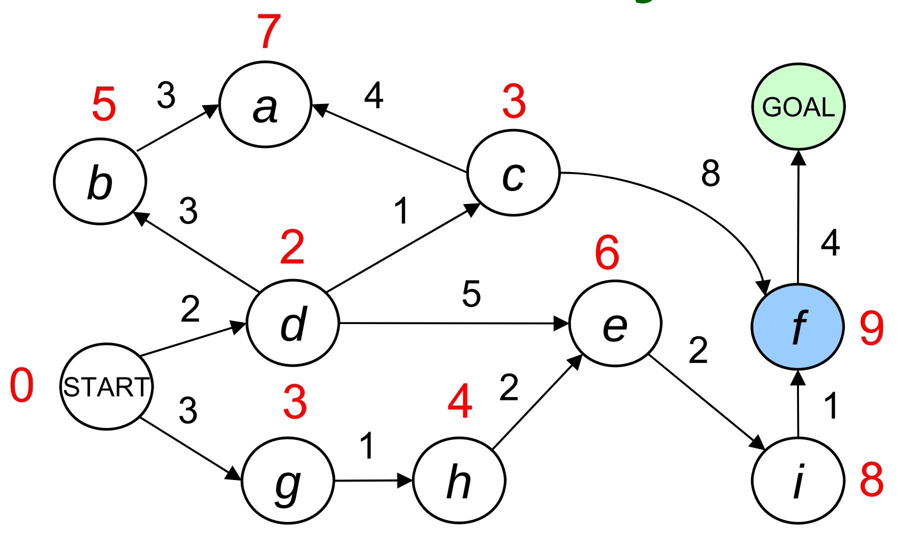
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

Keep a record of the least total cost needed to reach each node.

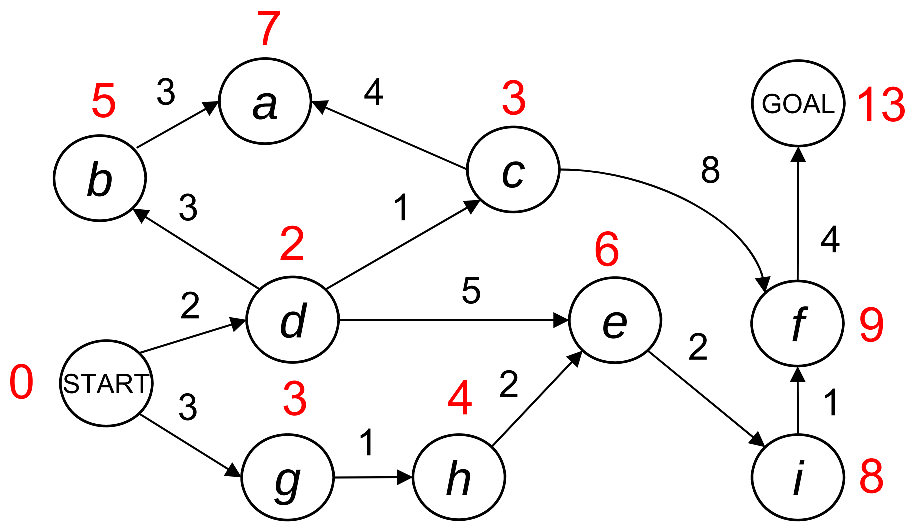
Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

<u>Queue</u>:
(f, 9)

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



Queue:
(GOAL, 13)

How to get from START to GOAL with the least total cost?

One answer: priority-based search

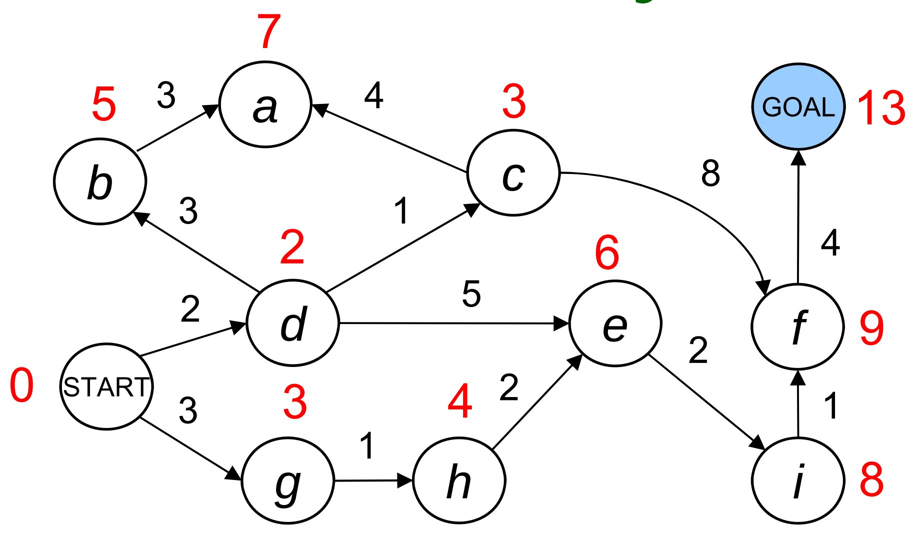
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. Repeat: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

<u>Queue</u>:
(GOAL, 13)

We can stop searching when the GOAL node is at the top of the queue.

<u>One answer: priority-based search</u>

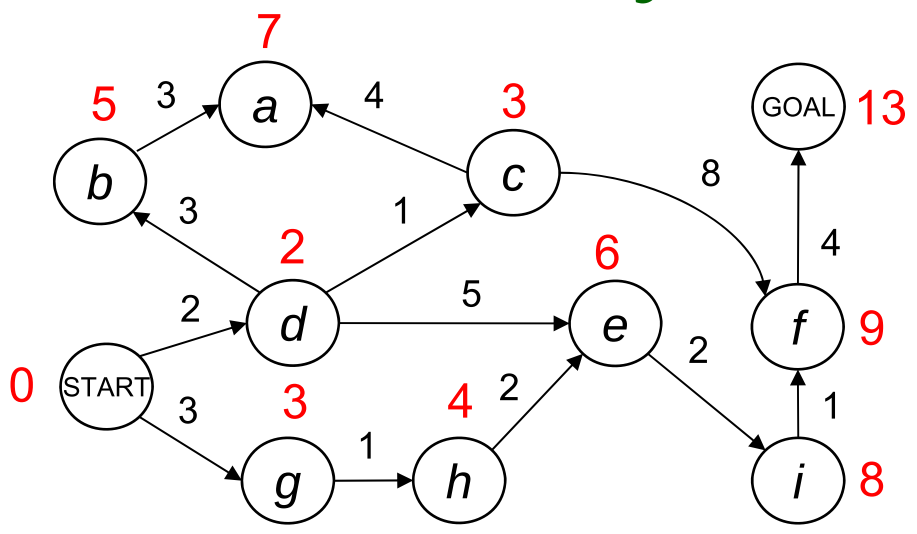
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

<u>One answer: priority-based search</u>

Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

Total cost to reach goal: 13

Obtain path to goal by backtracking: START → g → h → e → i → f → GOAL

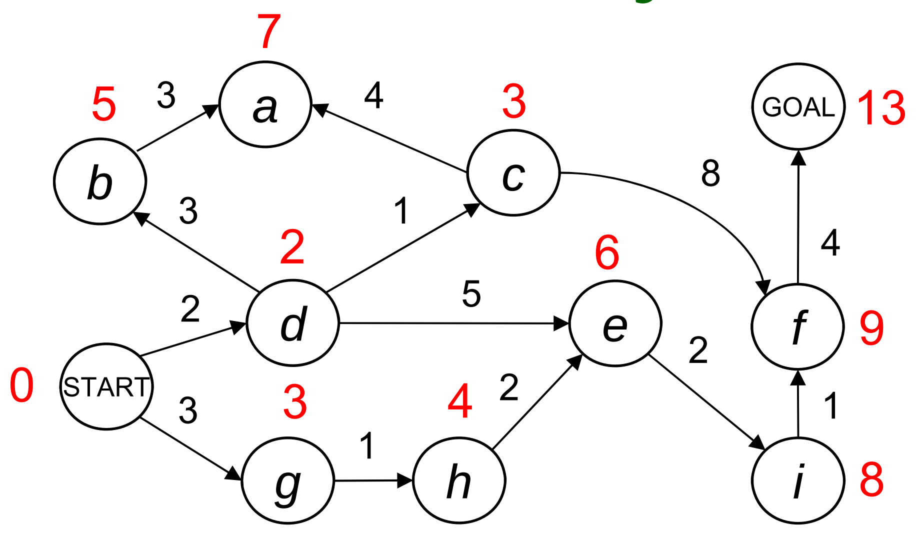
<u>Good news</u>: this method always finds the optimal (lowest-cost) path.

<u>Bad news</u>: it has to search many nodes before it reaches the goal.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

Priority-based search



How to get from START to GOAL with the least total cost?

To do better, we need a heuristic estimate of each node's lowest cost path to the goal.

We will choose the node n with the lowest sum of distances d(start, n) + d(n, goal).

This is called **A* search**.

<u>One answer: priority-based search</u>

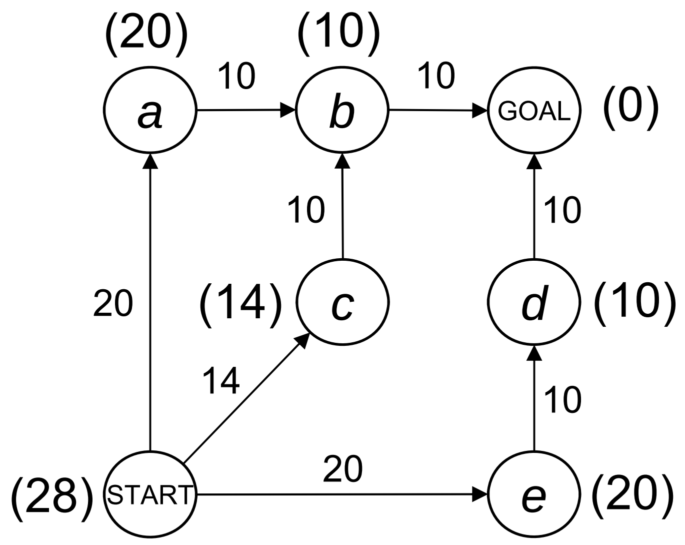
Keep a record of the least total cost needed to reach each node.

Also keep a list of nodes to search, sorted from lowest to highest cost. This is called a **priority queue**.

1. Put the START node on the queue, with cost = 0.

2. <u>Repeat</u>: remove the lowest-cost node from the queue, and add its children.

A* search



How to get from START to GOAL with the least total cost?

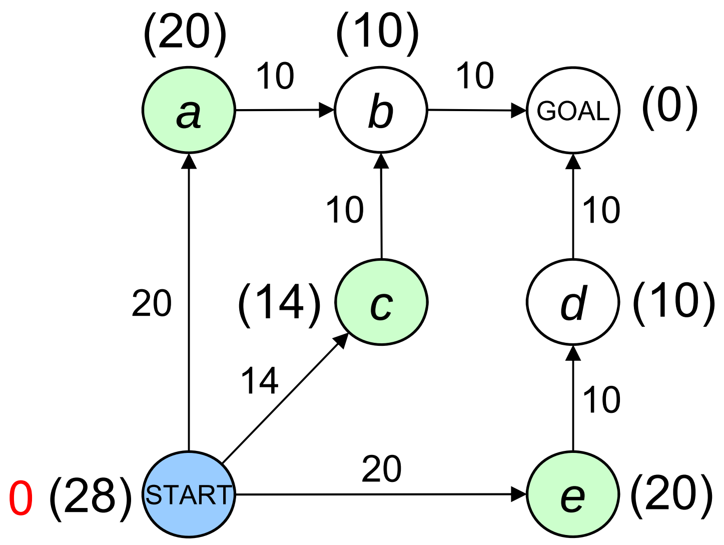
For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Let's consider a simple example of A* search, where the heuristic is Euclidean distance to the goal.

Notice that the heuristic value of each node is less than or equal to its lowest cost path to the goal.

This property must be true in order to guarantee that A* will find the optimal path.

A* search

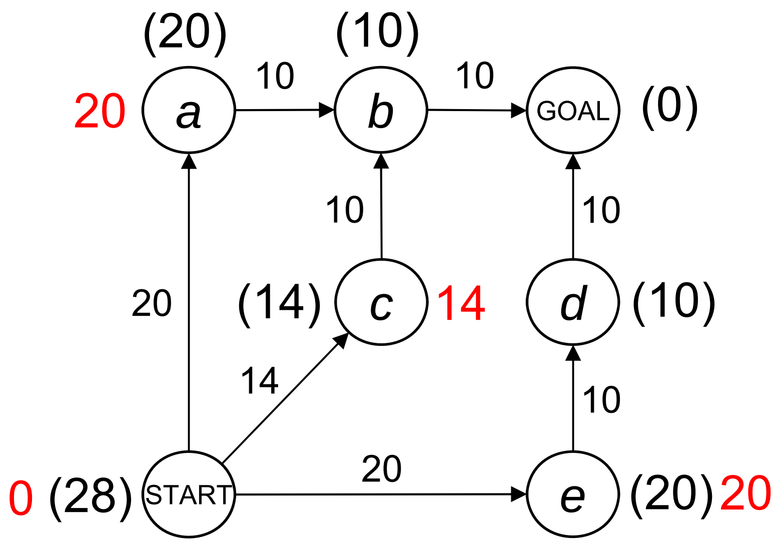


How to get from START to GOAL with the least total cost?

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Queue:
(START, 0+28)

A* search

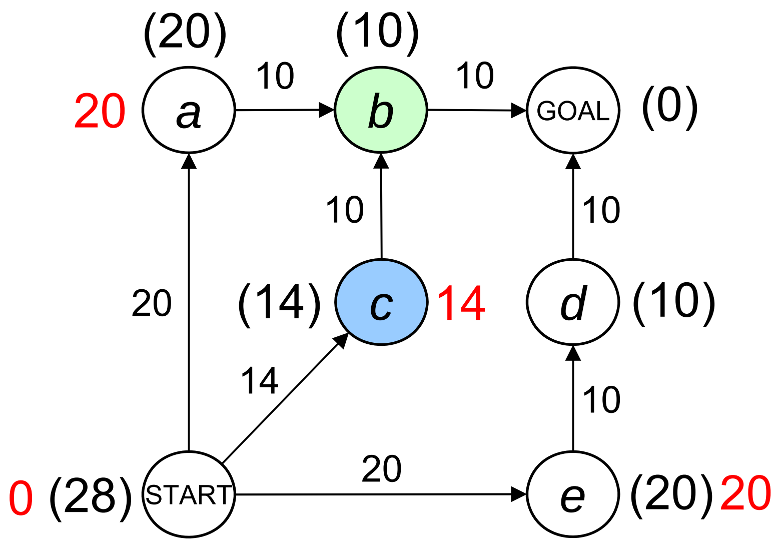


How to get from START to GOAL with the least total cost?

Queue:
(c, 14+14)
(a, 20+20)
(e, 20+20)

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

A* search

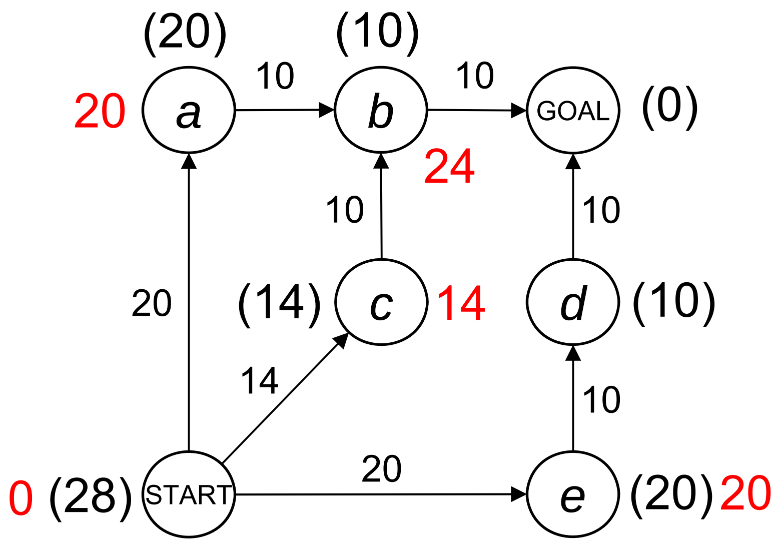


How to get from START to GOAL with the least total cost?

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Queue:
(c, 14+14)
(a, 20+20)
(e, 20+20)

A* search

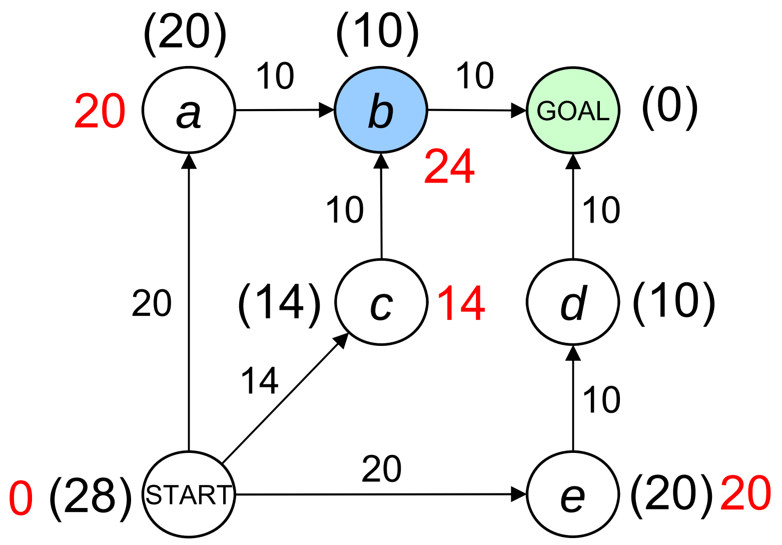


How to get from START to GOAL with the least total cost?

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Queue:
(b, 24+10)
(a, 20+20)
(e, 20+20)

A* search

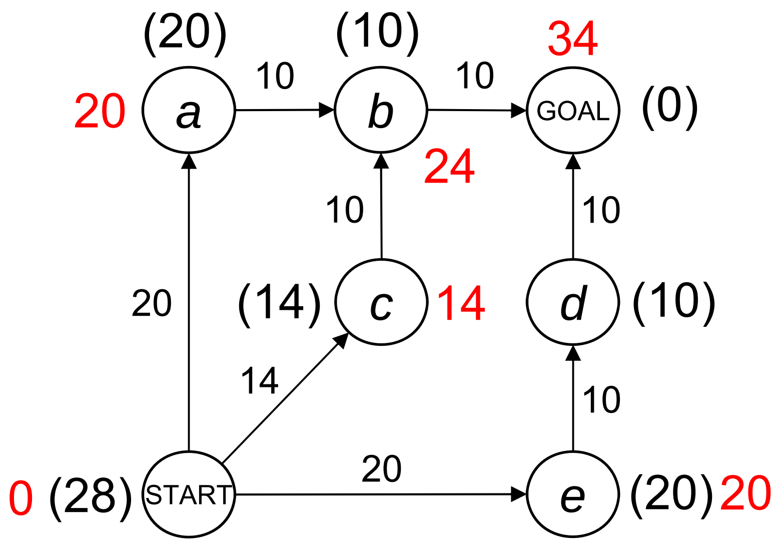


How to get from START to GOAL with the least total cost?

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Queue:
(b, 24+10)
(a, 20+20)
(e, 20+20)

A* search

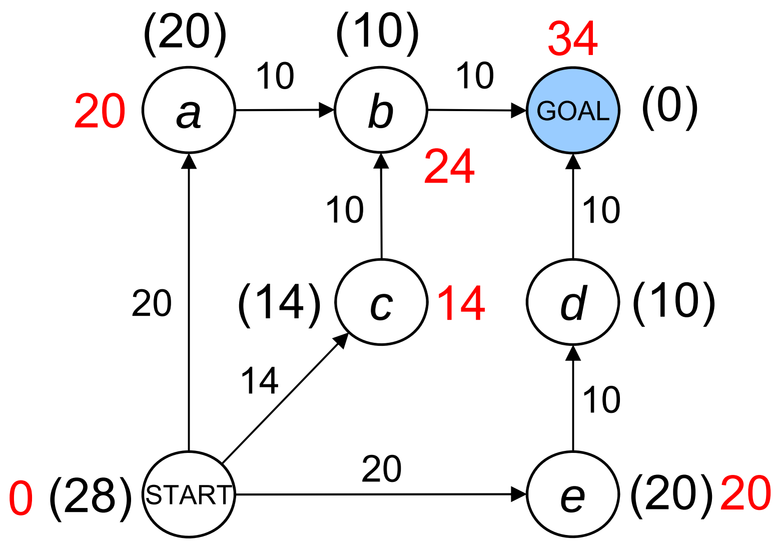


How to get from START to GOAL with the least total cost?

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).

Queue:
(GOAL, 34+0)
(a, 20+20)
(e, 20+20)

A* search



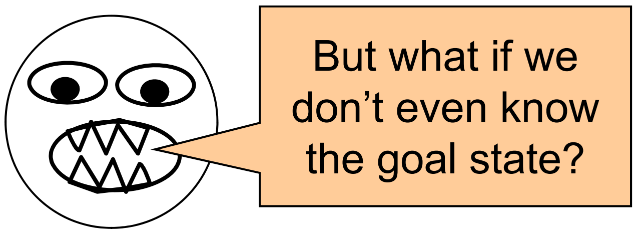
How to get from START to GOAL with the least total cost?

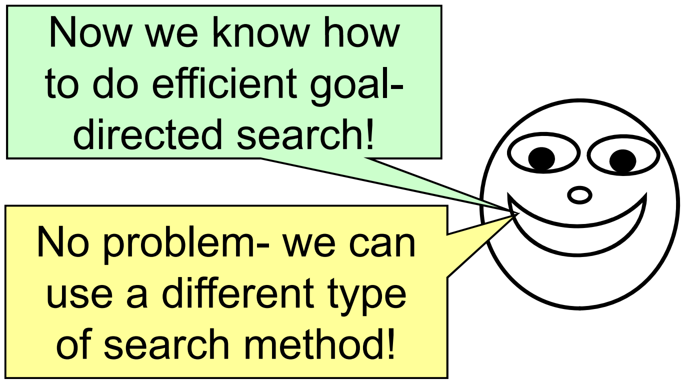
Queue:
(GOAL, 34+0)
(a, 20+20)
(e, 20+20)

We've now found the lowest-cost path to the goal, without looking at the transitions from nodes a, d, or e.

Without a heuristic, we would have had to look at every single node.

For A*, use a priority queue sorted by cost = d(start, n) + d(n, goal).





Now we know how to do efficient goal-directed search!
No problem- we can use a different type of search method!

State-space search

Typical scenario: we must consider a huge number of possible states of the world.

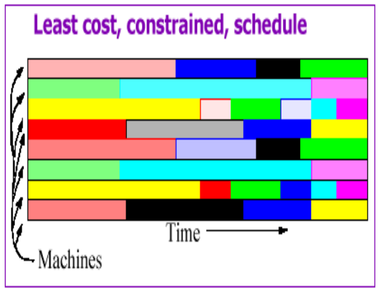
We are given a **score function** that maps each state to a real number, and our goal is to find the highest-scoring state.

However, there are far too many states for us to simply evaluate each one.

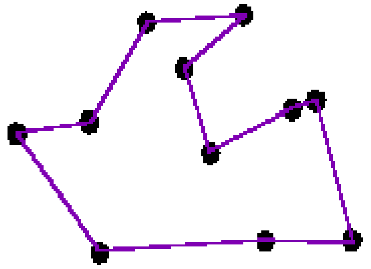
The state space could be combinatorial or continuous.

Challenge: How can we find an optimal (or nearly optimal) state in a reasonable time?

Assumption: Similar states have similar scores.

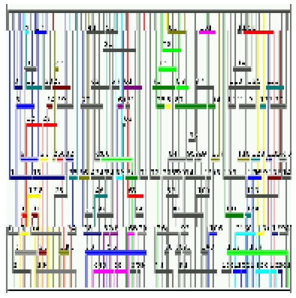


Job scheduling



Traveling salesman/ Meals on Wheels

Aerodynamic design



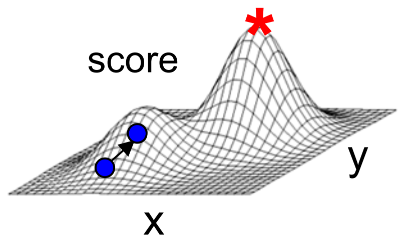
Microprocessor design

(Many other applications… urban planning, supply chains, etc.)

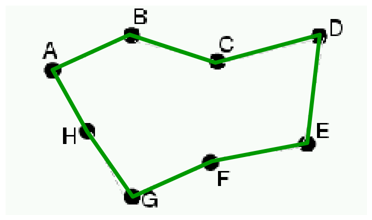
Hill-climbing

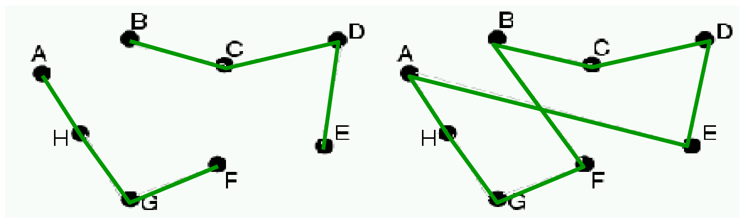
Imagine that the states of the world are laid out on the surface of a landscape, and we want to find the highest point.

For each state, we have a set of allowable moves to other states.



Simple 2-D example:
(x, y) → (x + Δx, y + Δy)





Traveling salesman problem:
cut and reconnect two edges

Hill-climbing algorithm:
1. Pick a random starting state.
2. On each step, move "uphill" to a state with higher score.
3. Stop when you're at a peak.

i.e. no allowable move can increase score

Hill-climbing

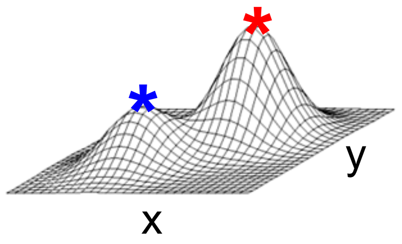
Question 1: How to choose the new state?

Deterministic hill-climbing: evaluate all allowable moves, and choose the state with the highest score.

Randomized hill-climbing: pick an allowable move at random, but only move if it improves the score.

Question 2: Does hill-climbing always find the optimal state?

Answer: No, it only finds a **local optimum** (a peak, but not necessarily the highest)



Hill-climbing algorithm:

1. Pick a random starting state.
2. On each step, move "uphill" to a state with higher score.
3. Stop when you're at a peak.

One solution: Run hill-climbing many times with different starting states, and choose the best local optimum.

Another solution: simulated annealing (allow some downhill moves as well)

Simulated annealing

Pick an allowable move at random. Make the move with probability 1 if it improves the score, or with some smaller probability if it decreases the score.

Here's the tricky part: we let the probability of allowing a downhill move decrease gradually over time.

This may make the search fall out of poor local maxima and into better local maxima.

$$\Pr(\text{accept}) = \exp\left(\frac{\Delta\text{score}}{T}\right)$$

"Temperature" T decreases with time

<u>Large T</u>: random walk (all moves accepted)

<u>Small T</u>: hill-climbing (only uphill moves)

<u>Hill-climbing algorithm</u>:

1. Pick a random starting state.
2. On each step, move "uphill" to a state with higher score.
3. Stop when you're at a peak.

We can prove that, with an infinitely slow cooling rate, SA converges to the global optimum.

More practically, it often finds better solutions than hill-climbing when there are many local optima.

Issues in state-space search

- Hill-climbing and simulated annealing are useful tools for general search problems.
- The challenging part is designing a good representation of the problem domain, and choosing a good moveset.
- <u>Advantages</u>: very easy to implement, can save a great deal of programming effort, only need to remember the current state, can be parallelized easily.
- <u>Disadvantage</u>: No guarantee that you will find the optimal solution in a reasonable amount of time.
- DON'T solve a problem with hill-climbing or simulated annealing if you can find an exact solution (e.g. by linear programming or A* search).
- In the remainder of the course, we'll often treat modeling and detection as state-space search problems, searching over the set of possible models and over all potentially interesting subsets of the data respectively.

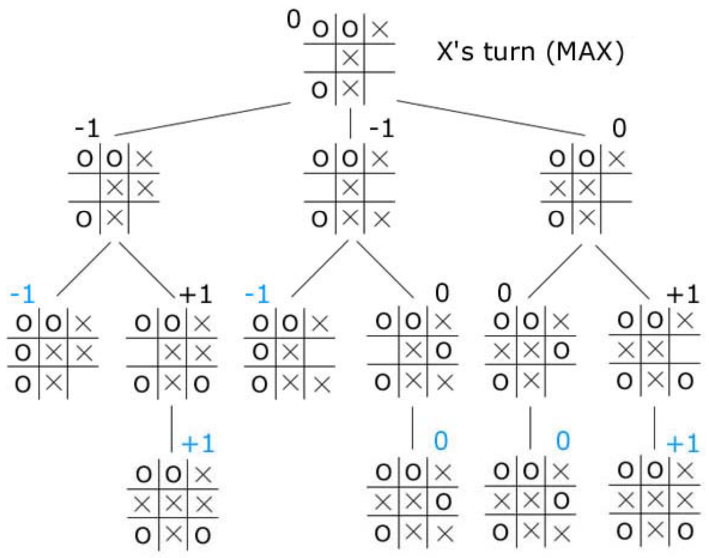
Other search techniques

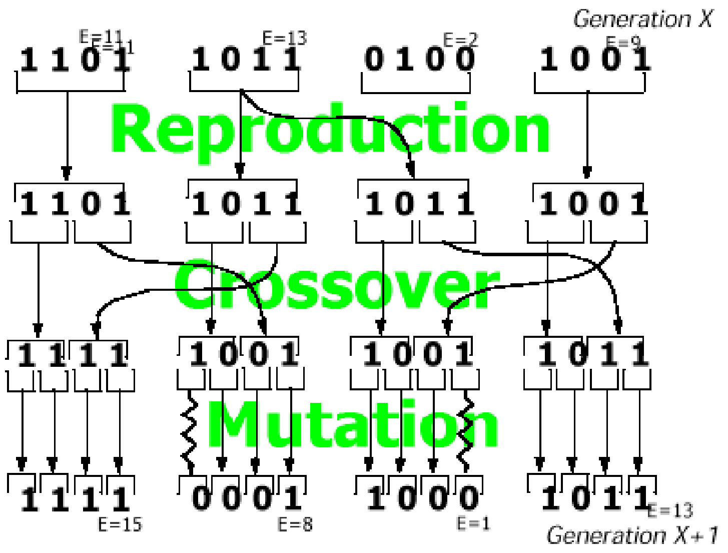
Greedy search: if you want to quickly find a "decent" solution, but don't care about finding the optimal solution, just make the best myopic choice at each decision point.

Example: Greedy algorithm for TSP
Pick a random starting location
Repeat: choose closest unvisited location
When all locations visited, return to start

Minimax search: if you have an opponent, assume that the opponent will play optimally, and choose a best response at each decision point.

Genetic algorithms: Initialize a large population of states, and let them evolve by natural selection (random mutation and cross-over).





References

- N. Nilsson. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill,1971.
- J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley, 1984.
- S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Chapters 3-4.

This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Large Scale Data Analysis for Policy 90-866, Fall 2012

Lecture 6: Methods for Clustering Data

Clustering data

Main goal: Given a dataset of N records, we wish to partition the dataset into $k << N$ groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

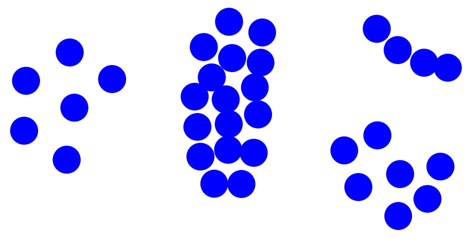
Given a population of individuals, we want to **identify** and **characterize** the underlying subgroups (or “clusters”) of individuals, and possibly use these subgroups for **prediction**.

We want to explain the data in terms of its natural groupings.

How many groups are there?
Who belongs to which group?
Characteristics of each group?

Example: identify congressional voting blocs.

How well do blocs correspond to party affiliation?
Are there relevant blocs within a party?
Are there “mavericks” that vote across party lines?
How much do blocs vary with proposed legislation?



2-D example

Clustering data

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Given a population of individuals, we want to **identify** and **characterize** the underlying subgroups (or “clusters”) of individuals, and possibly use these subgroups for **prediction**.

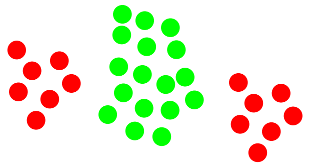
How does clustering differ from prediction?

There may not be any single output that we are trying to predict.

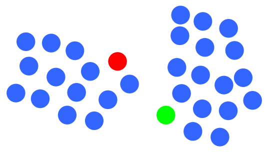
We are interested in an underlying **structure** that explains or predicts many characteristics of the population.

Clustering can improve prediction performance.

Improve Bayesian classification by learning multiple models per class.



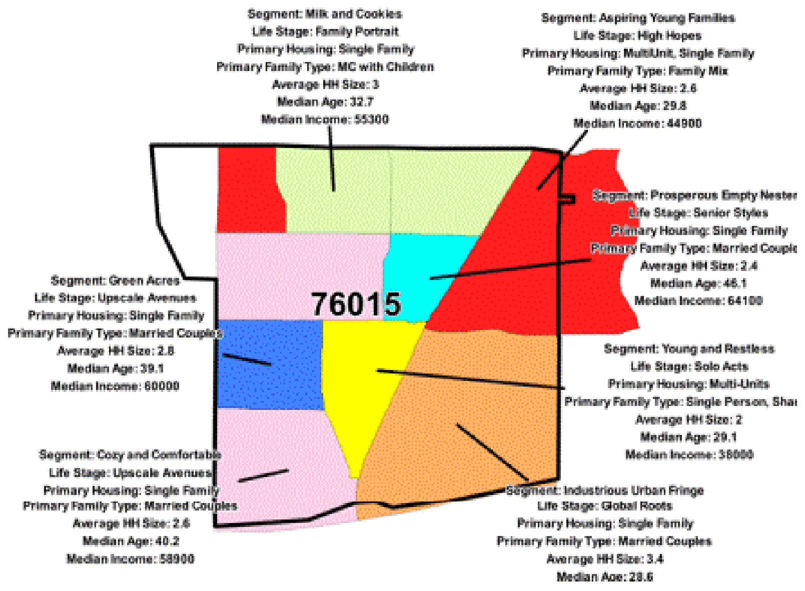
We may have little or no labeled training data for learning class models, but lots of unlabeled data.



Applications of clustering

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

An important application of clustering is to improve **prediction** of an individual's characteristics or behavior, using information obtained from other members of their inferred group.



Segment: Milk and Cookies
Life Stage: Family Portrait
Primary Housing: Single Family
Primary Family Type: MC with Children
Average HH Size: 3
Median Age: 32.7
Median Income: 55300
Segment: Aspiring Young Families
Life Stage: High Hopes
Primary Housing: MultiUnit, Single Family
Primary Family Type: Family Mix
Average HH Size: 2.6
Median Age: 29.8
Median Income: 44900
Segment: Prosperous Empty Nester
Life Stage: Senior Styles
Primary Housing: Single Family
Primary Family Type: Married Couple
Average HH Size: 2.4
Median Age: 46.1
Median Income: 64100
76015
Segment: Green Acres
Life Stage: Upscale Avenues
Primary Housing: Single Family
Primary Family Type: Married Couples
Average HH Size: 2.8
Median Age: 39.1
Median Income: 60000
Segment: Young and Restless
Life Stage: Solo Acts
Primary Housing: Multi-Units
Primary Family Type: Single Person, Sha
Average HH Size: 2
Median Age: 29.1
Median Income: 38000
Segment: Cozy and Comfortable
Life Stage: Upscale Avenues
Primary Housing: Single Family
Primary Family Type: Married Couples
Average HH Size: 2.6
Median Age: 40.2
Median Income: 58900
Segment: Industrious Urban Fringe
Life Stage: Global Roots
Primary Housing: Single Family
Primary Family Type: Married Couples
Average HH Size: 3.4
Median Age: 28.6

Customer database segmentation: To which subgroups should I market my product, and how should I target them? (Similarly, voter database segmentation)

Patient database segmentation: Different subgroups of patients may benefit from different treatment regimens.

Applications of clustering

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Group-based trajectory modeling identifies subgroups of the population, and uses these subgroups to predict how an individual's behavior will change over the course of his/her life.

Prof. Daniel Nagin developed these methods and applied them to **predict** juvenile delinquency and criminal behavior.

This figure shows the predicted and actual trajectories of an individual's number of criminal convinctions, varying with age. Three trajectory groups were identified: never (71%), limited to adolescence (22%), and chronic offenders (7%).

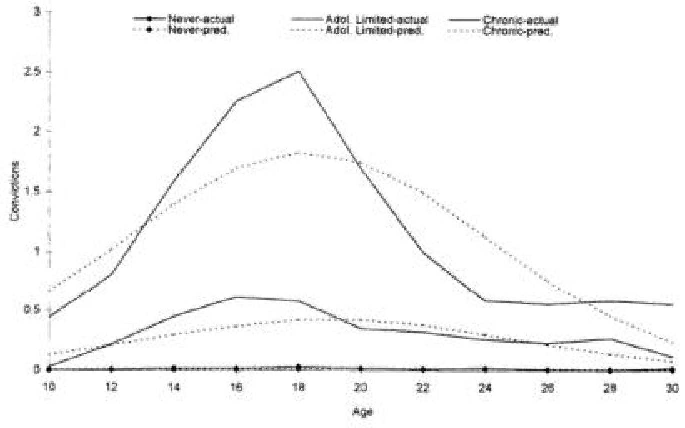


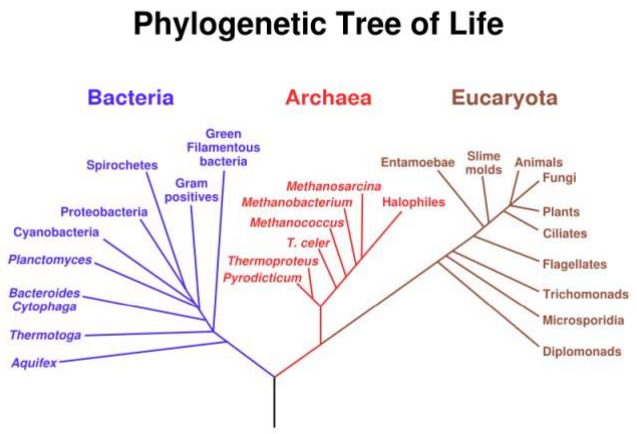
Figure 1. Trajectories of number of convictions (Cambridge sample). Adol. = adolescent; pred. = predicted.

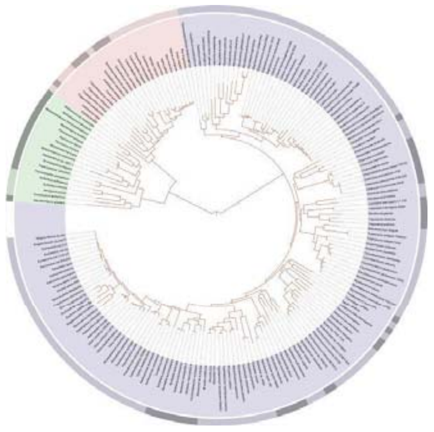
Nagin, D. S. 1999. "Analyzing Developmental Trajectories: A Semi-parametric, Group-based Approach." *Psychological Methods*, 4: 139-177.

Applications of clustering

Main goal: Given a dataset of N records, we wish to partition the dataset into k << N groups such that records in the same group are similar to each other, and records in different groups are dissimilar.

Clustering is also very commonly used in **evolutionary biology** to create “phylogenetic trees” relating different species and showing when the species diverged from a common ancestor.



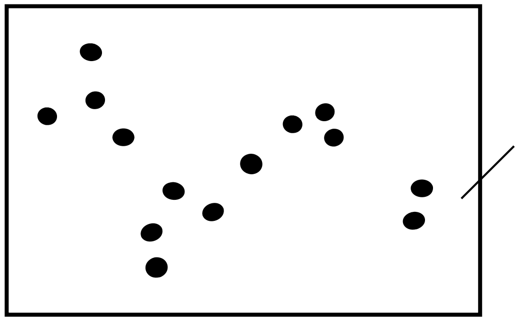


Here we are interested in learning the **hierarchy** of clusters!

Here is a simple phylogenetic tree developed manually by evolutionary biologists.

Now much more detailed trees can be generated automatically by clustering DNA sequences, enhancing our understanding of evolution.

Hierarchical clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

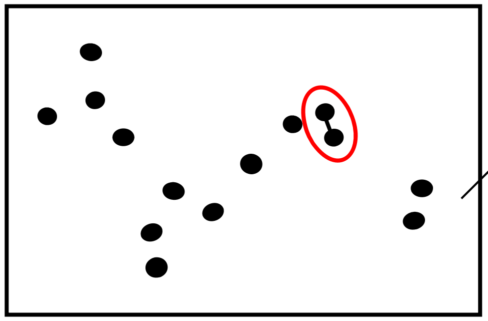
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

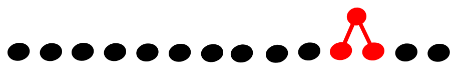
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

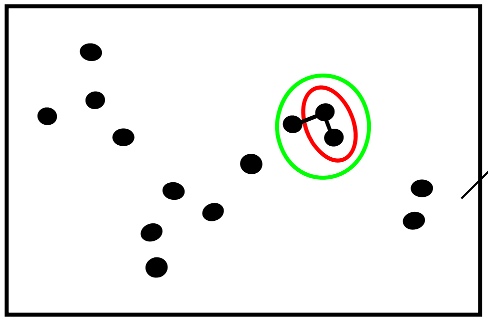
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 1 merge)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

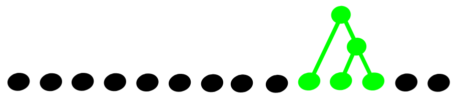
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

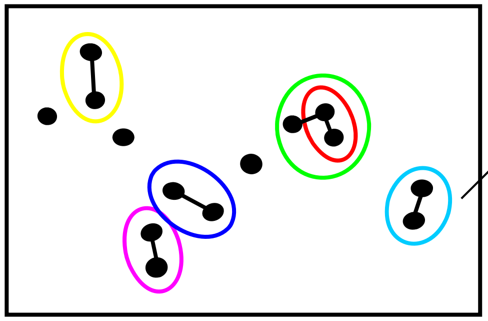
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 2 merges)



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

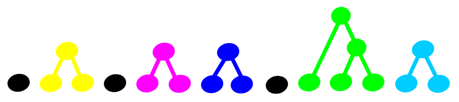
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

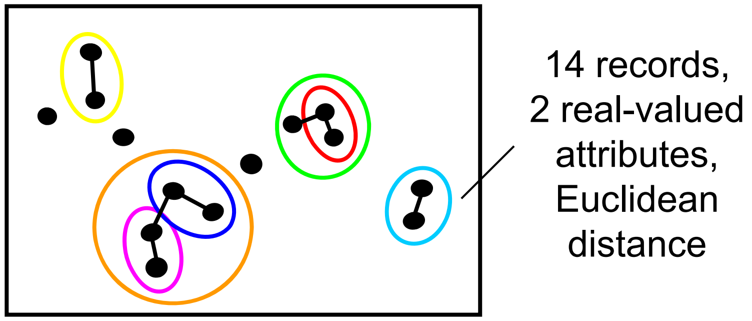
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 6 merges)



Single-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

<u>Bottom-up hierarchical clustering</u>

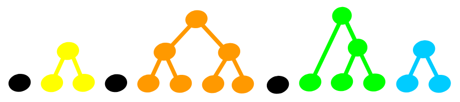
- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

How to define "nearest" clusters?

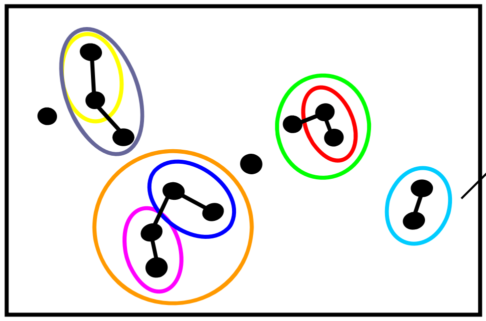
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

(After 7 merges)



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

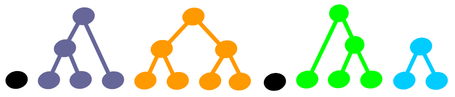
We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

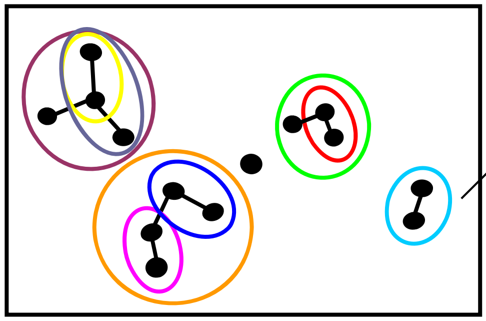
$$D(C,C') = \min_{x \in C,\, x' \in C'} d(x,x')$$

Single-link clustering

(After 8 merges)



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

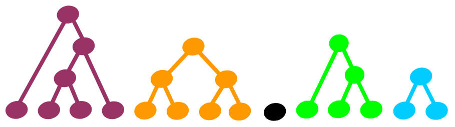
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

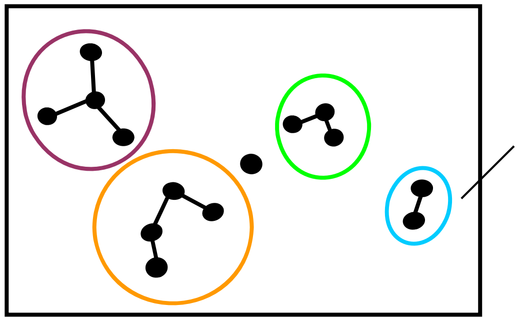
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 9 merges)



Single-link clustering



Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

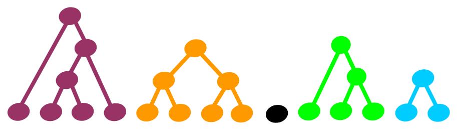
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

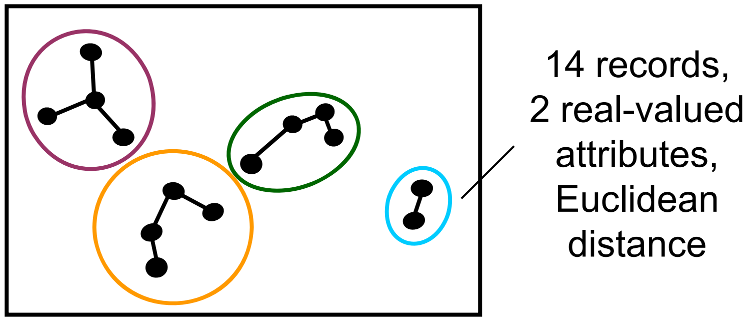
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 9 merges)



Single-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

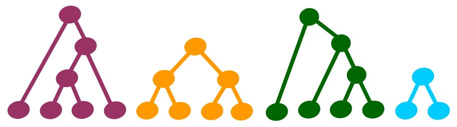
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

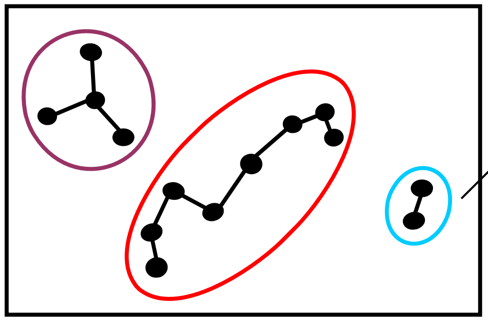
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 10 merges)



Single-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

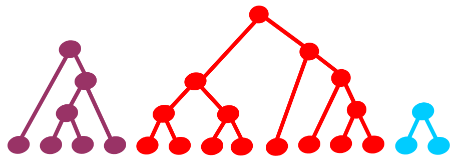
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

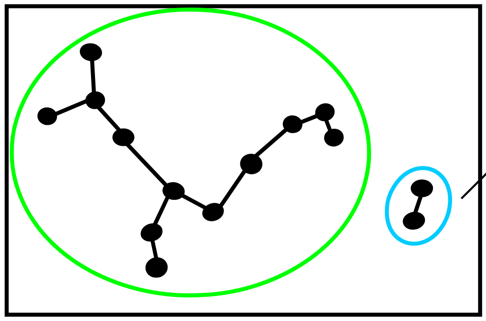
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 11 merges)



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

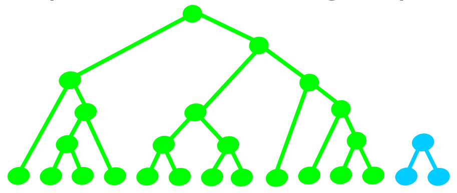
$$D(C, C') = \min_{x \in C, x' \in C'} d(x, x')$$

Single-link clustering

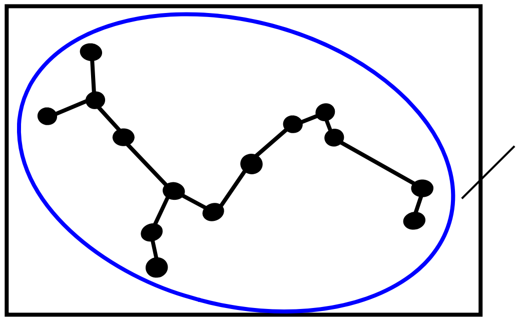
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

(After 12 merges)



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

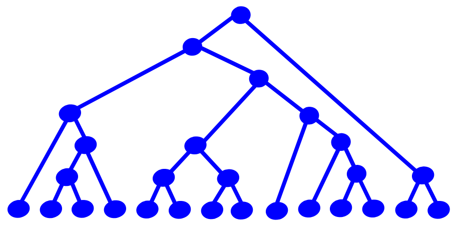
$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

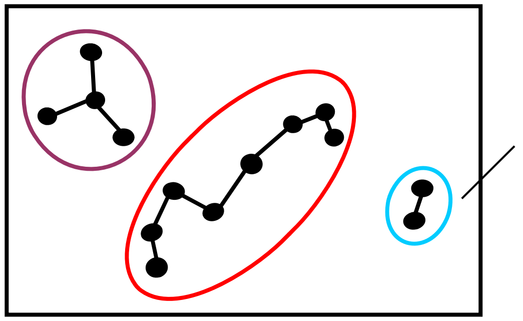
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Done!



Single-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C,C') = \min_{x \in C, x' \in C'} d(x,x')$$

Single-link clustering

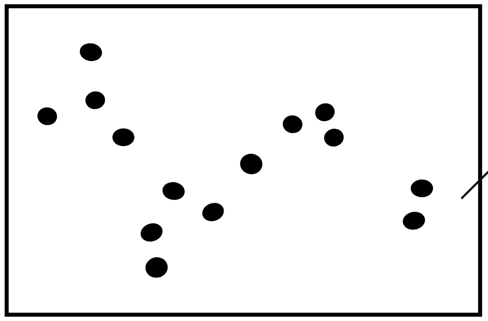
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$
and a distance metric $d(x_i, x_j)$.

Records can have real and
discrete-valued attributes.

We will create a **hierarchy** of
clusters by merging similar records.

How to define "nearest" clusters?

$$D(C, C') = \max_{x \in C,\, x' \in C'} d(x, x')$$

Complete-link clustering

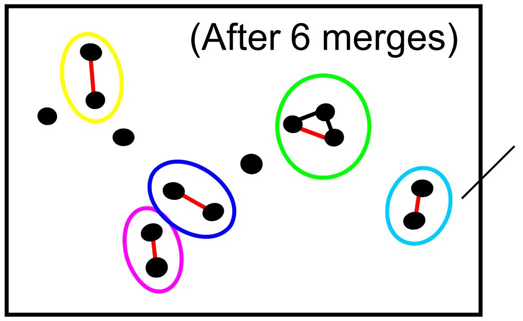
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define “nearest” clusters?

$$D(C,C') = \max_{x \in C,\, x' \in C'} d(x,x')$$

Complete-link clustering

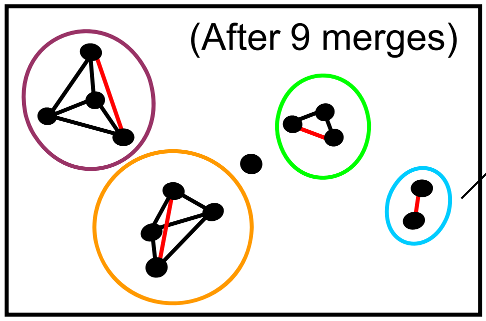
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two “nearest” clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or “chains”) as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records, 2 real-valued attributes, Euclidean distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C,C') = \max_{x \in C, x' \in C'} d(x,x')$$

Complete-link clustering

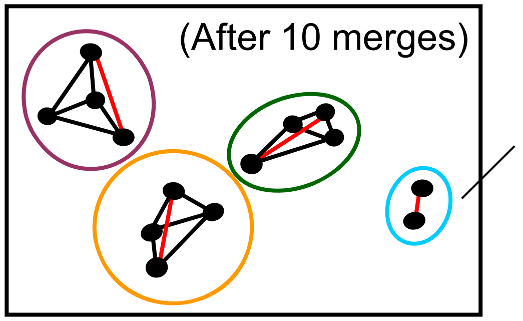
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C,C') = \max_{x \in C, x' \in C'} d(x,x')$$

Complete-link clustering

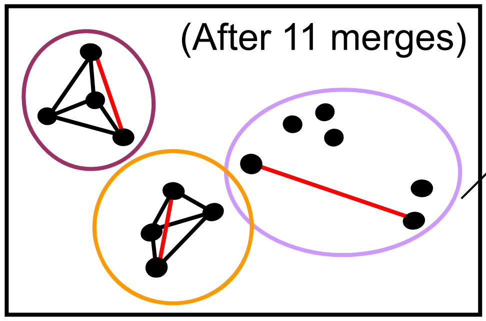
<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C, C') = \max_{x \in C, x' \in C'} d(x, x')$$

Complete-link clustering

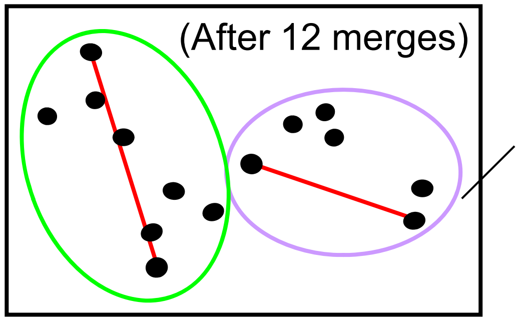
Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Complete-link clustering



14 records,
2 real-valued
attributes,
Euclidean
distance

Given a set of records $x_1..x_N$ and a distance metric $d(x_i, x_j)$.

Records can have real and discrete-valued attributes.

We will create a **hierarchy** of clusters by merging similar records.

How to define "nearest" clusters?

$$D(C,C') = \max_{x \in C,\, x' \in C'} d(x,x')$$

Complete-link clustering

<u>Bottom-up hierarchical clustering</u>

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Note that single-link clustering tends to produce highly elongated groups (or "chains") as shown above.

If we want to produce more spherical groups, we can instead consider the **maximum** distance between clusters.

Hierarchical clustering

Top-down hierarchical clustering

- Start with one cluster containing all N records.
- Choose the "worst" cluster, and optimally partition it into two distinct clusters.
- Repeat until all points are in separate clusters.

Bottom-up hierarchical clustering

- Start with N clusters, each containing one record.
- Choose the two "nearest" clusters, and merge them into a single cluster.
- Repeat until all points are merged into a single cluster.

Top-down clustering is hard, so we focus on the bottom-up approach.

Some fancy hierarchical clustering algorithms alternate bottom-up and top-down stages.

Advantages of hierarchical clustering

You get an entire cluster hierarchy, not just a single clustering.

Easy to compare different numbers of clusters k: just cut the longest k-1 links.

(Any disadvantages?)

K-means clustering

Hierarchical clustering is a simple, useful clustering method, but it gives you no guarantees on the quality of the resulting groups.

An alternative is to define some <u>objective function</u> that describes the quality of the groups, and attempt to optimize that function.

Assume that all attributes are real-valued, so we can compute the <u>centroid</u> of any cluster (i.e. the mean value of each attribute)

Possible objective:
minimize distortion

$$\sum_{C_k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

This is the sum of squared errors for each data point x_i, assuming that each x_i is mapped to the closest cluster center μ_k.

<u>Possible moves for state-space search</u>
Change position of cluster center μ_k
Change mapping of point x_i to center μ_k
Change number of clusters K

How to perform this search efficiently?

K-means clustering

Hierarchical clustering is a simple, useful clustering method, but it gives you no guarantees on the quality of the resulting groups.

An alternative is to define some objective function that describes the quality of the groups, and attempt to optimize that function.

Assume that all attributes are real-valued, so we can compute the centroid of any cluster (i.e. the mean value of each attribute)

Possible objective:
minimize distortion

$$\sum_{C_k} \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

This is the sum of squared errors for each data point x_i, assuming that each x_i is mapped to the closest cluster center μ_k.

Possible moves for state-space search

Change position of cluster center μ_k $\leftarrow$ 1. Moving μ_k to centroid of all points in C_k reduces distortion

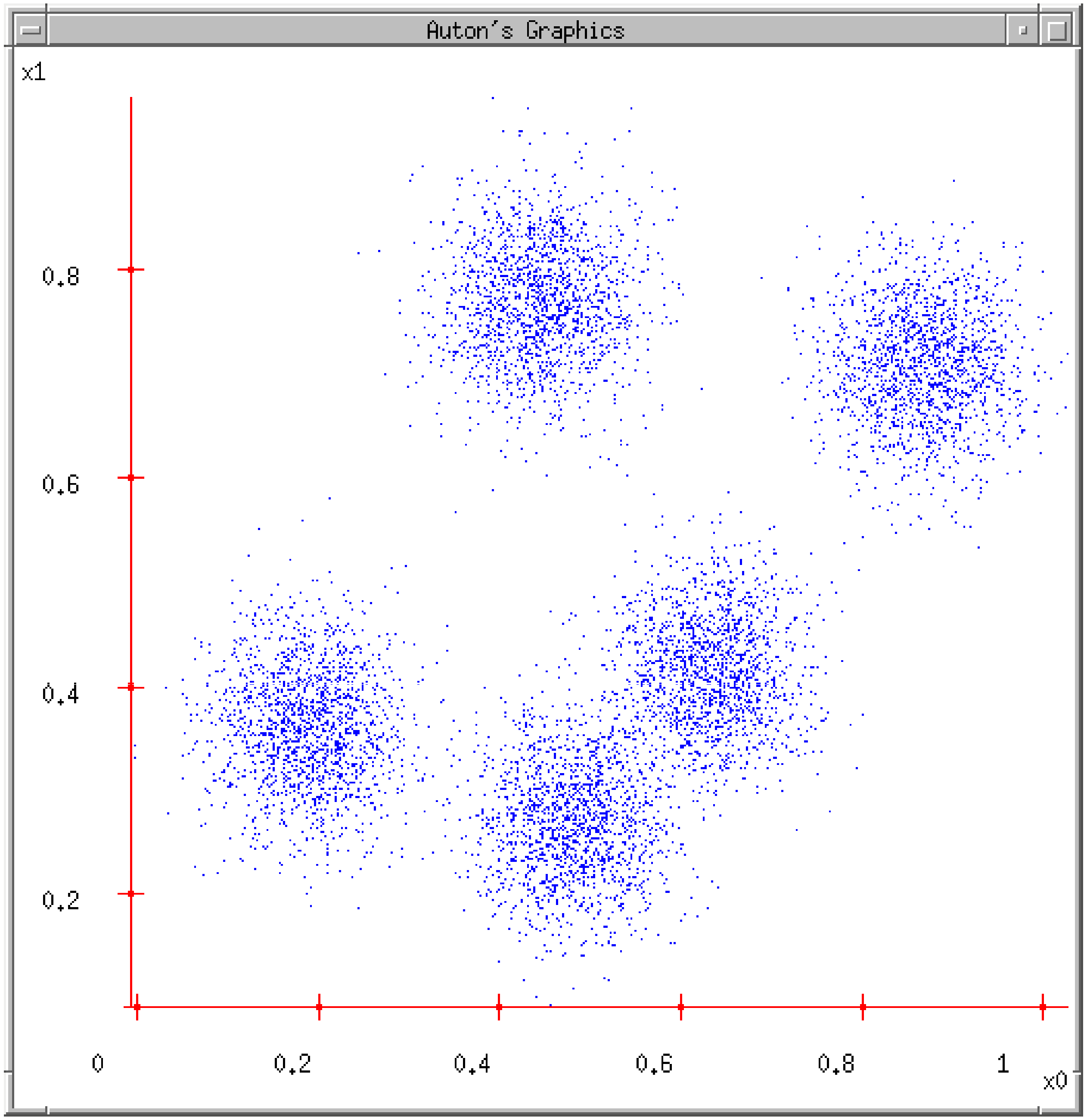
Change mapping of point x_i to center μ_k $\leftarrow$ 2. Mapping point x_i to nearest center μ_k reduces distortion.

Change number of clusters K

Keep number of centers fixed, and alternate between these two moves.

K-means

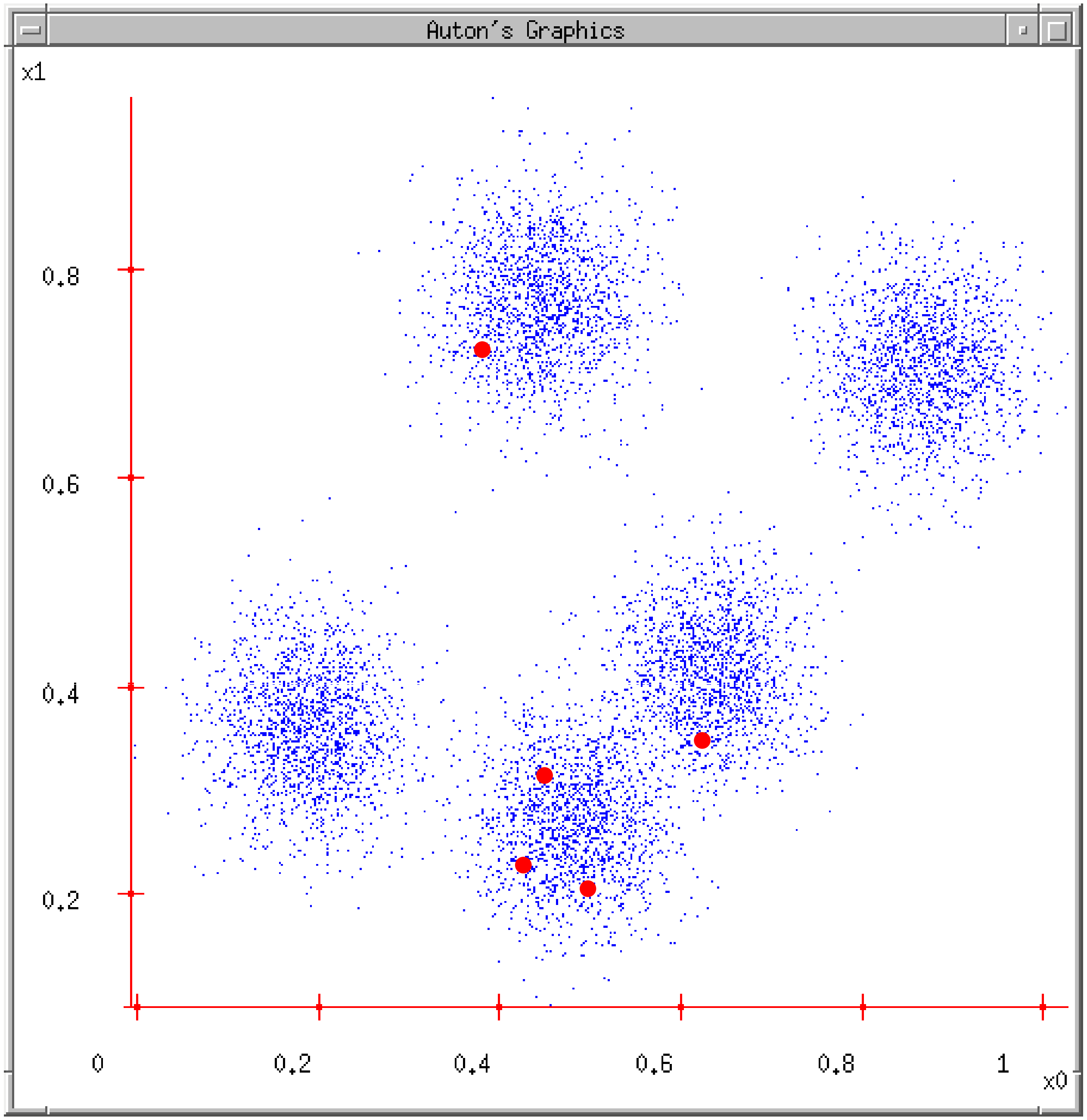
1. Ask user how many clusters they'd like. (e.g. k = 5)



Thanks to Andrew Moore for providing this example.

K-means

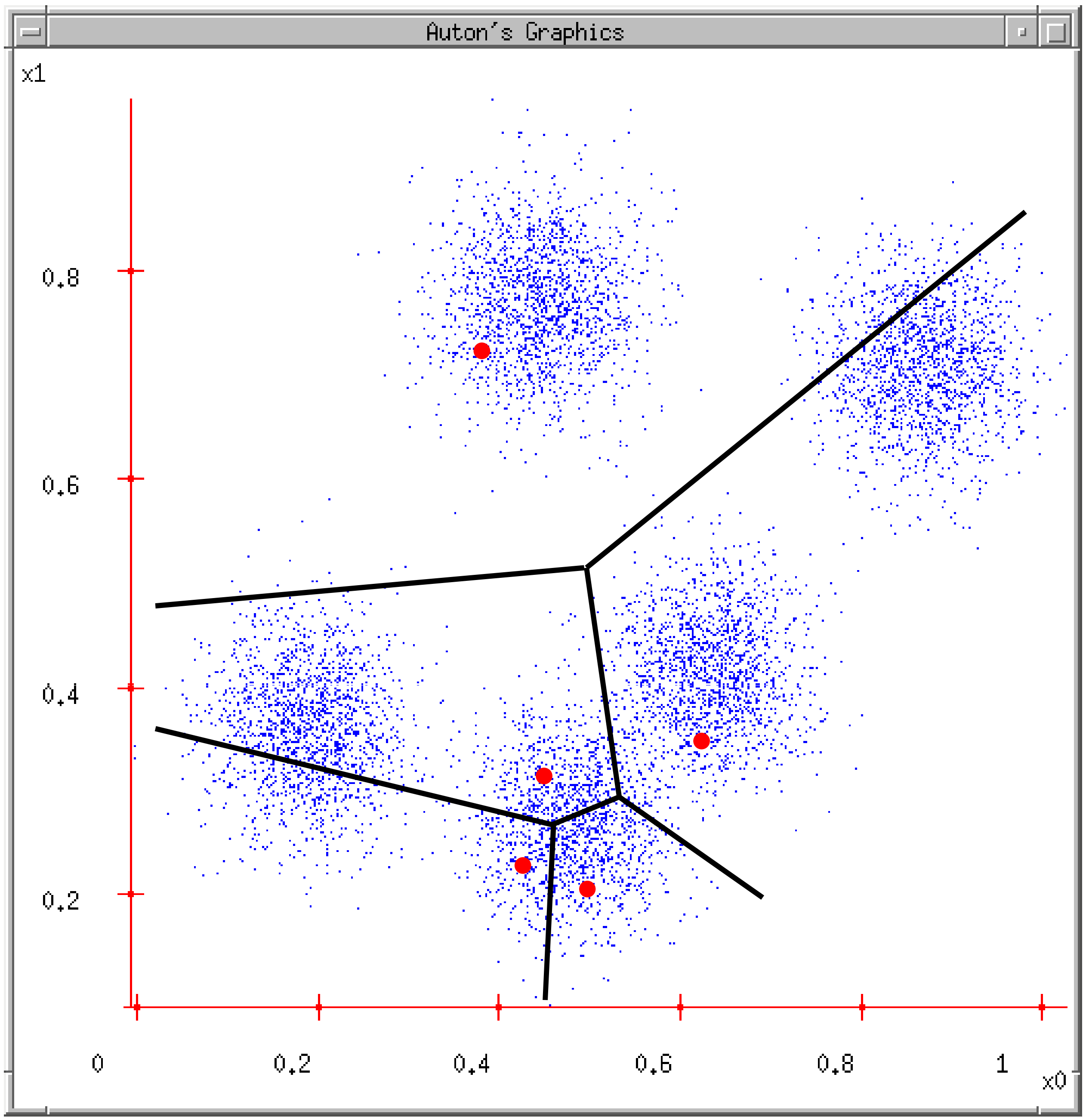
1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations



Thanks to Andrew Moore for providing this example.

K-means

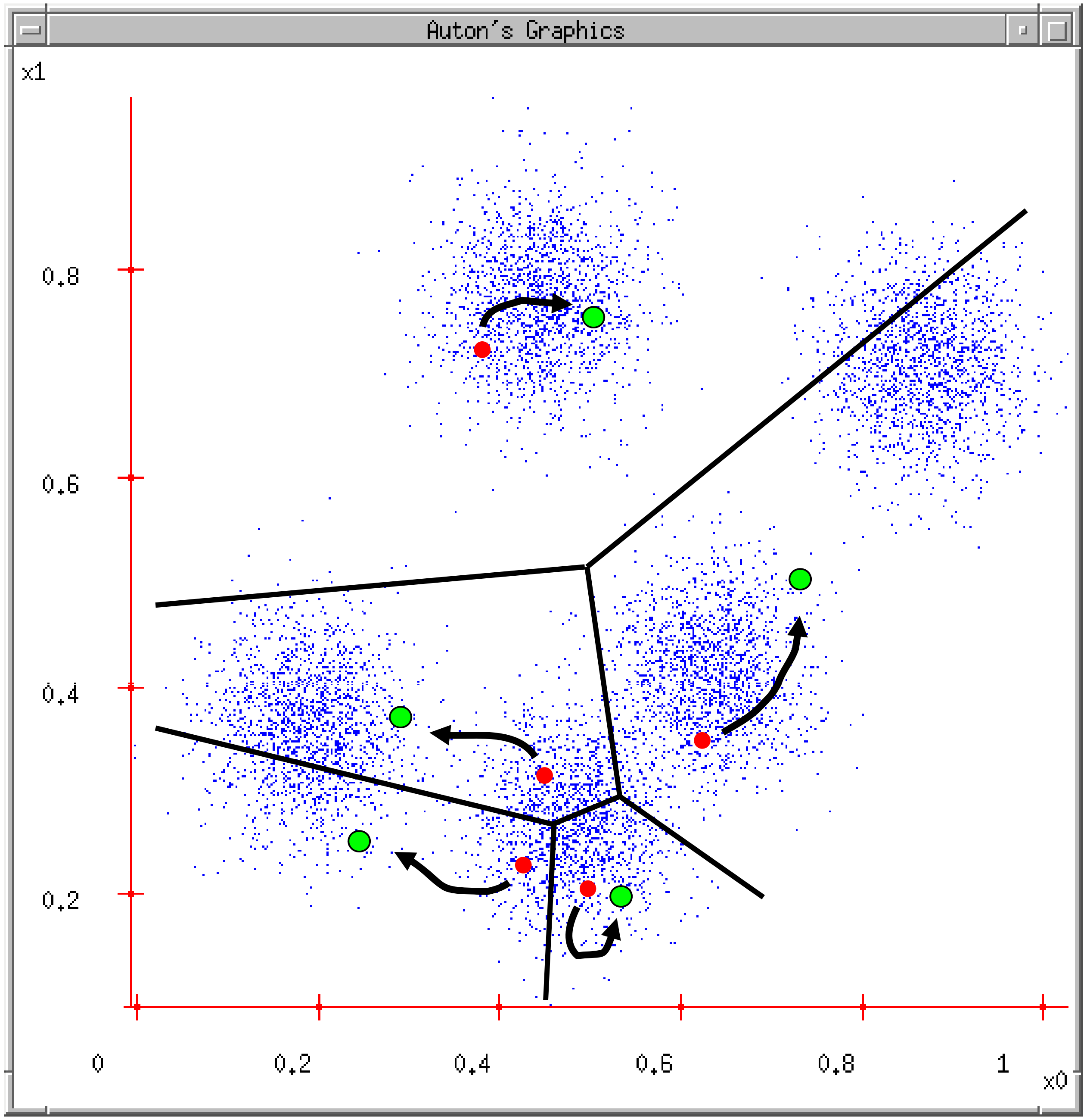
1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations
3. Each datapoint finds out which center it's closest to.



Thanks to Andrew Moore for providing this example.

K-means

1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations
3. Each datapoint finds out which center it's closest to.
4. Each center finds the centroid of the points it owns, and moves there.

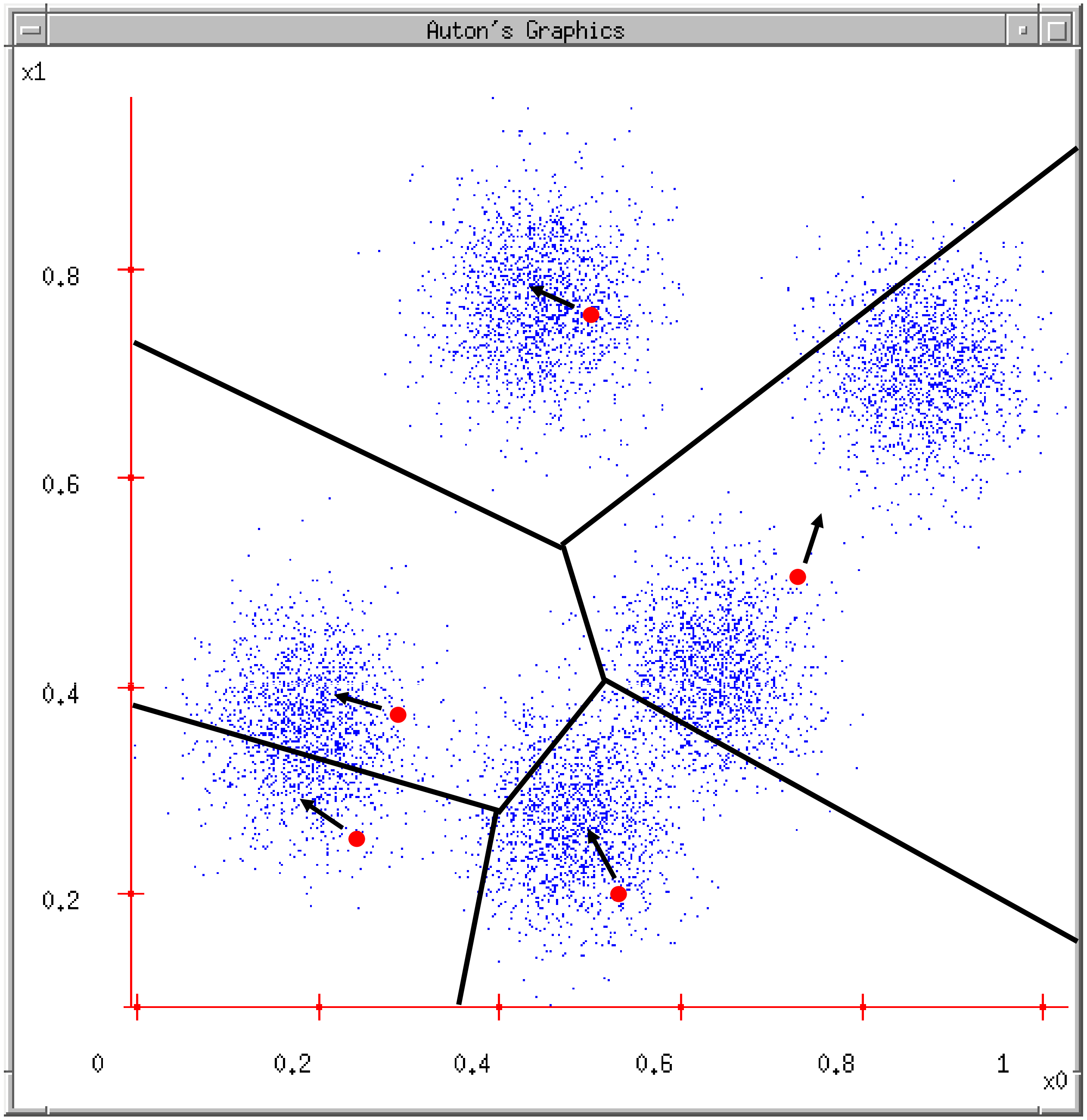


Thanks to Andrew Moore for providing this example.

K-means

1. Ask user how many clusters they'd like. (e.g. k = 5)
2. Randomly guess k cluster center locations
3. Each datapoint finds out which center it's closest to.
4. Each center finds the centroid of the points it owns, and moves there.

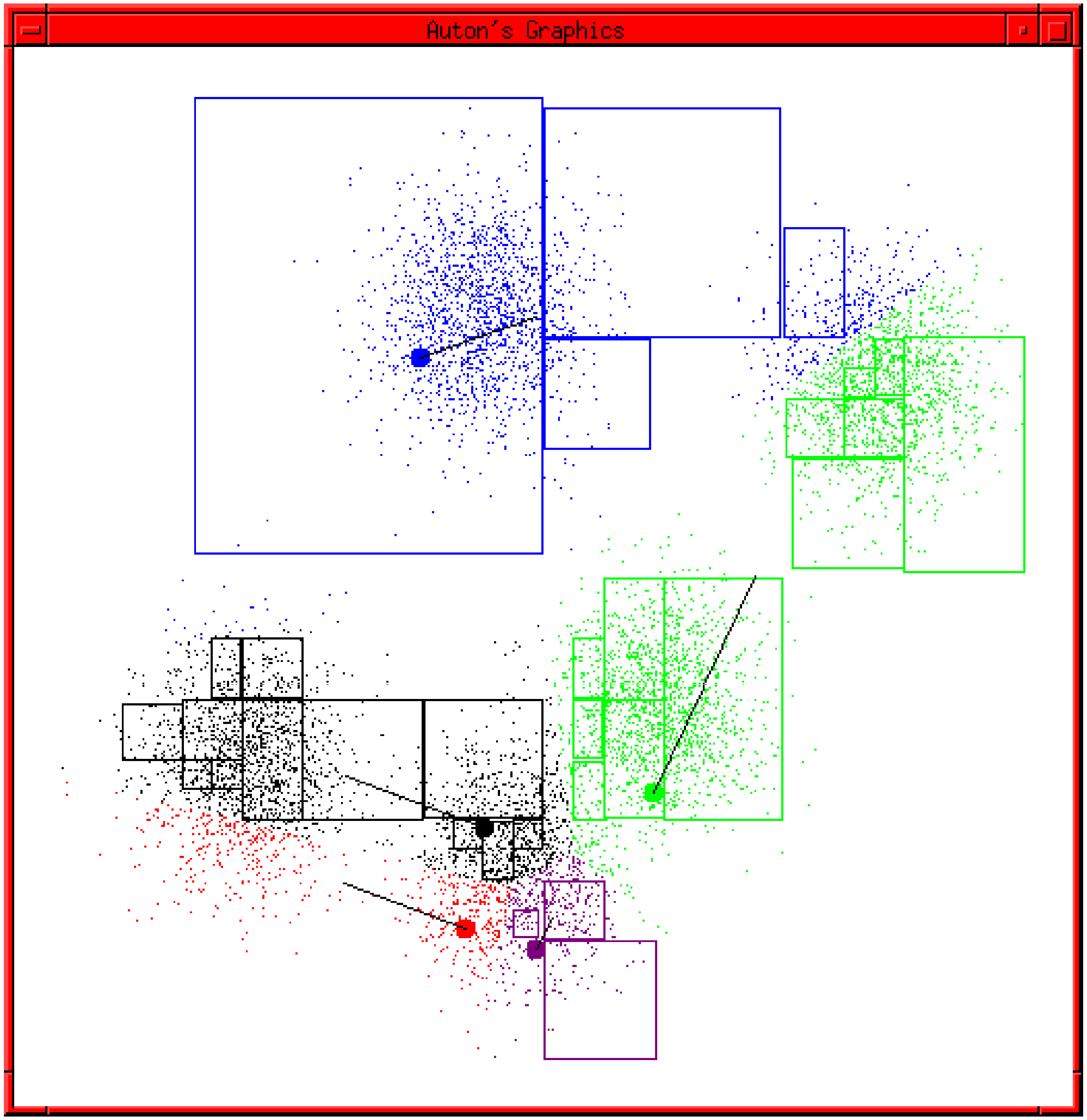
Repeat steps 3-4 until convergence!



Thanks to Andrew Moore for providing this example.

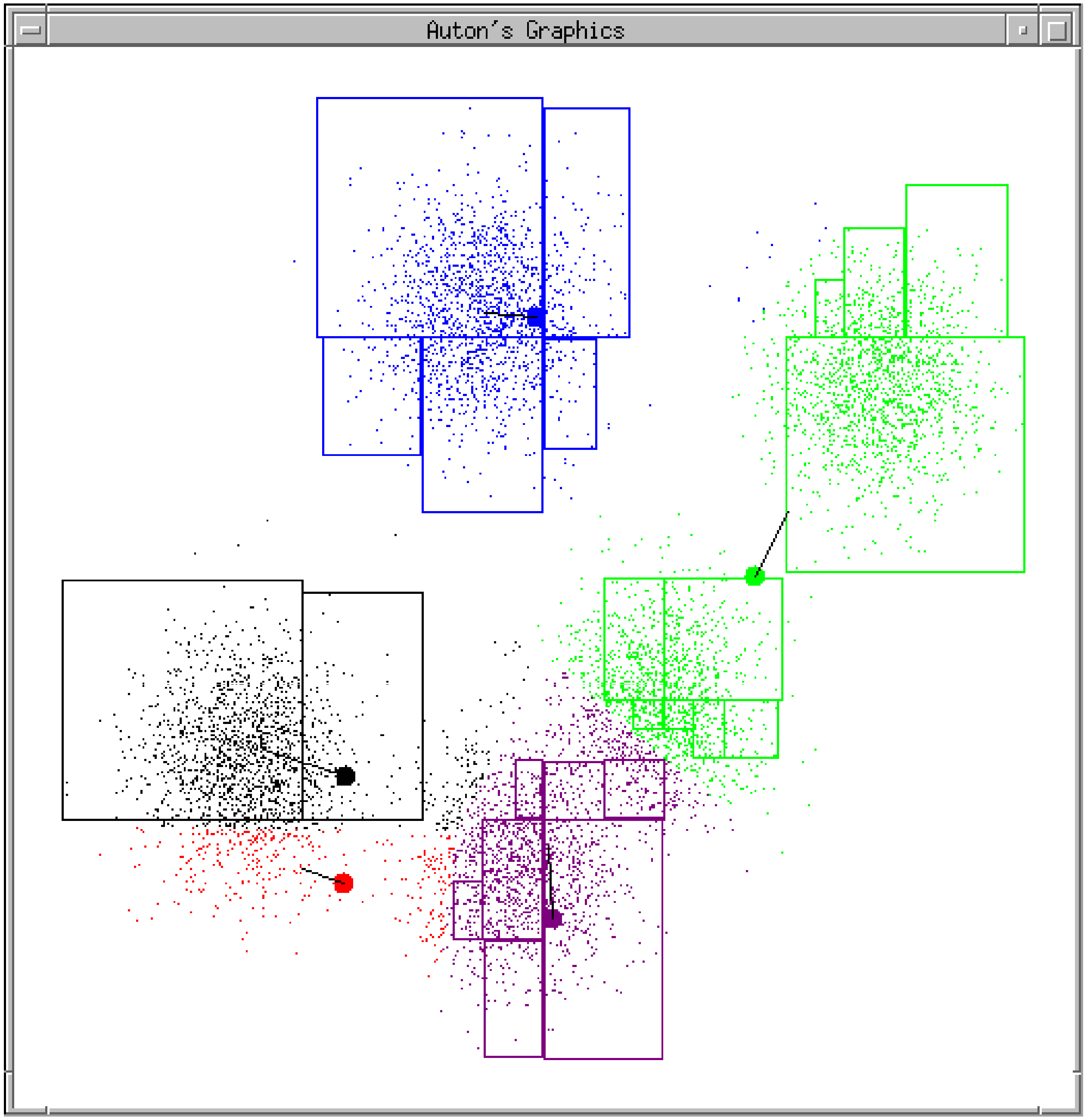
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



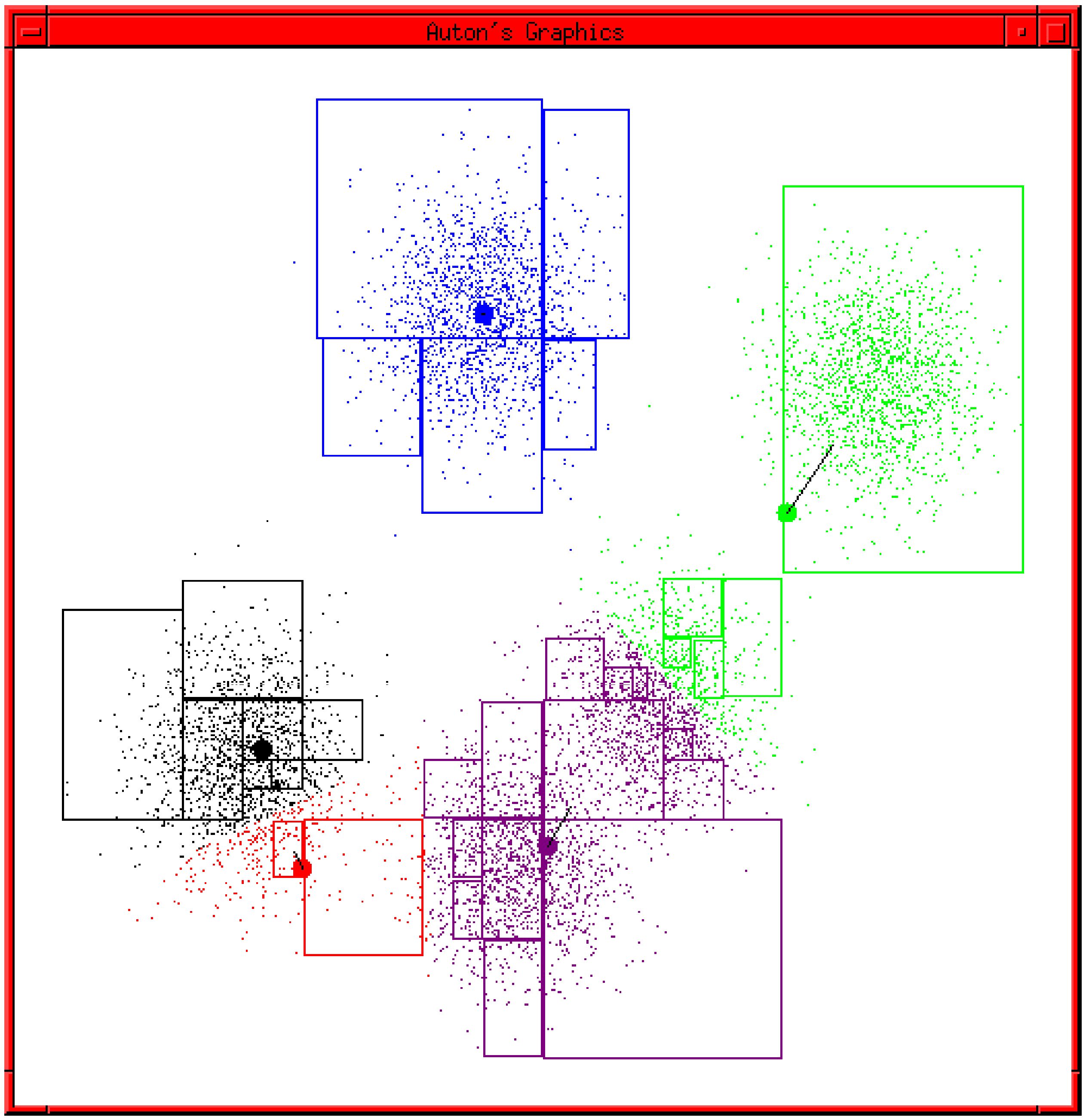
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



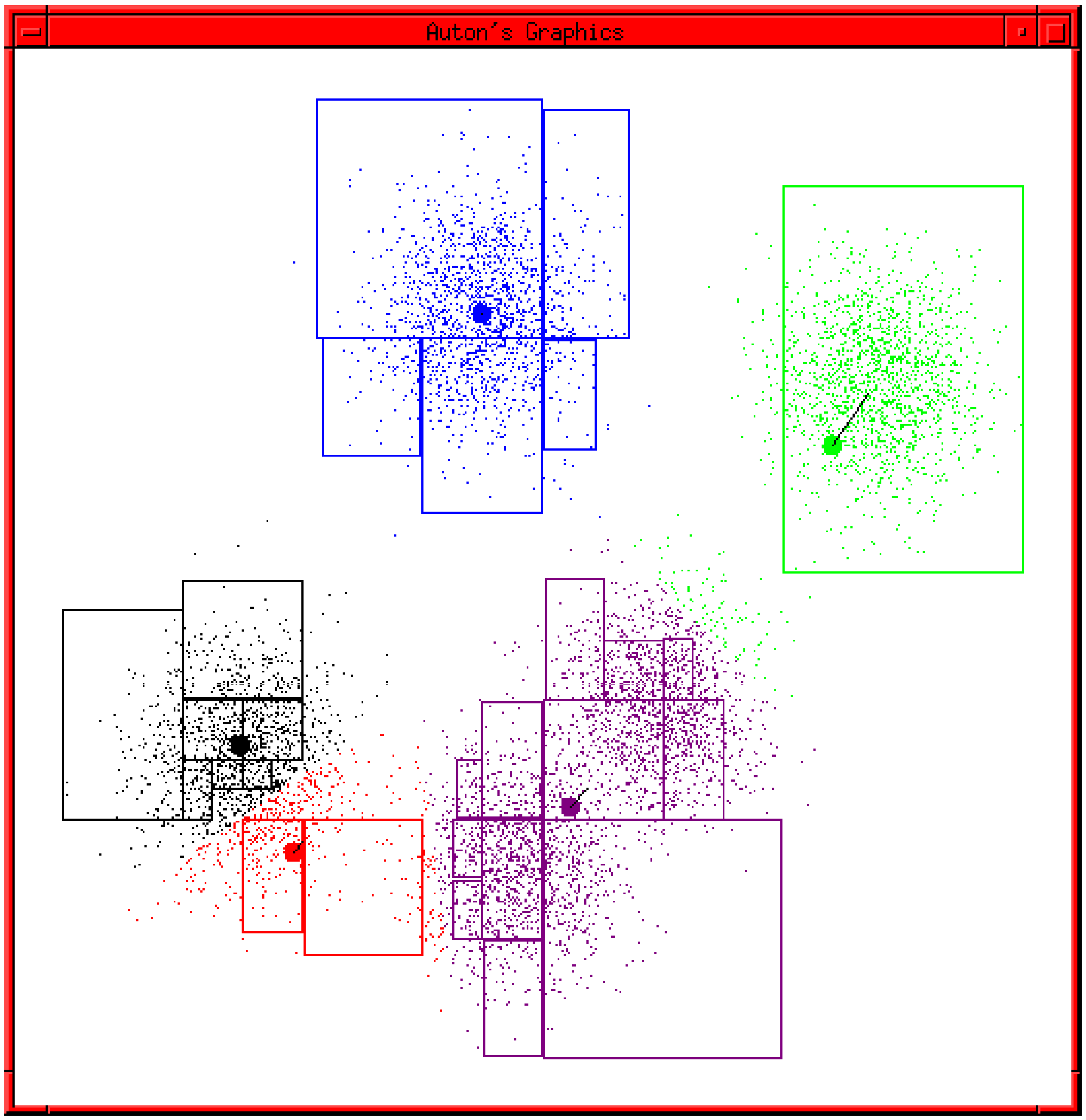
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



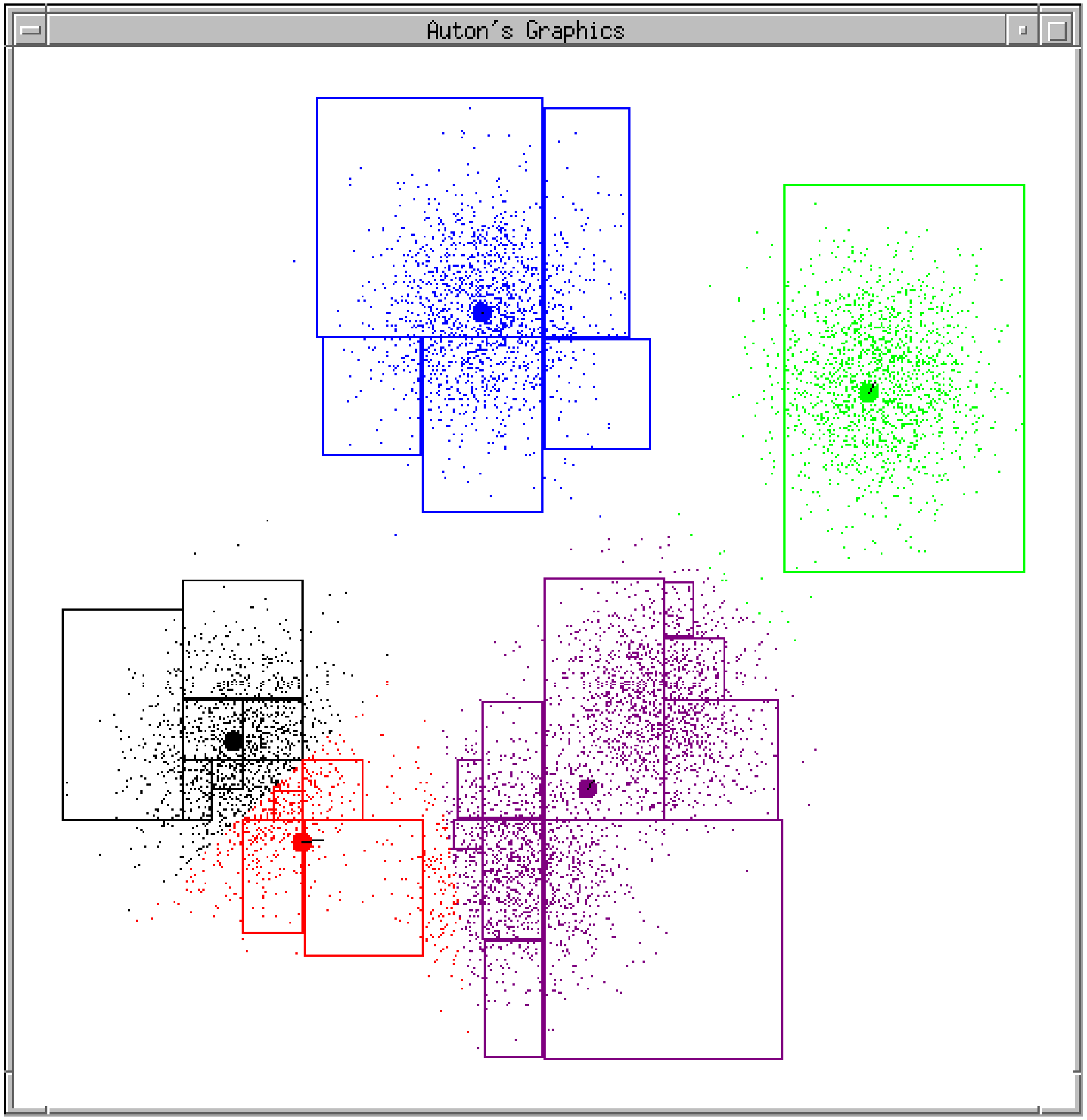
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



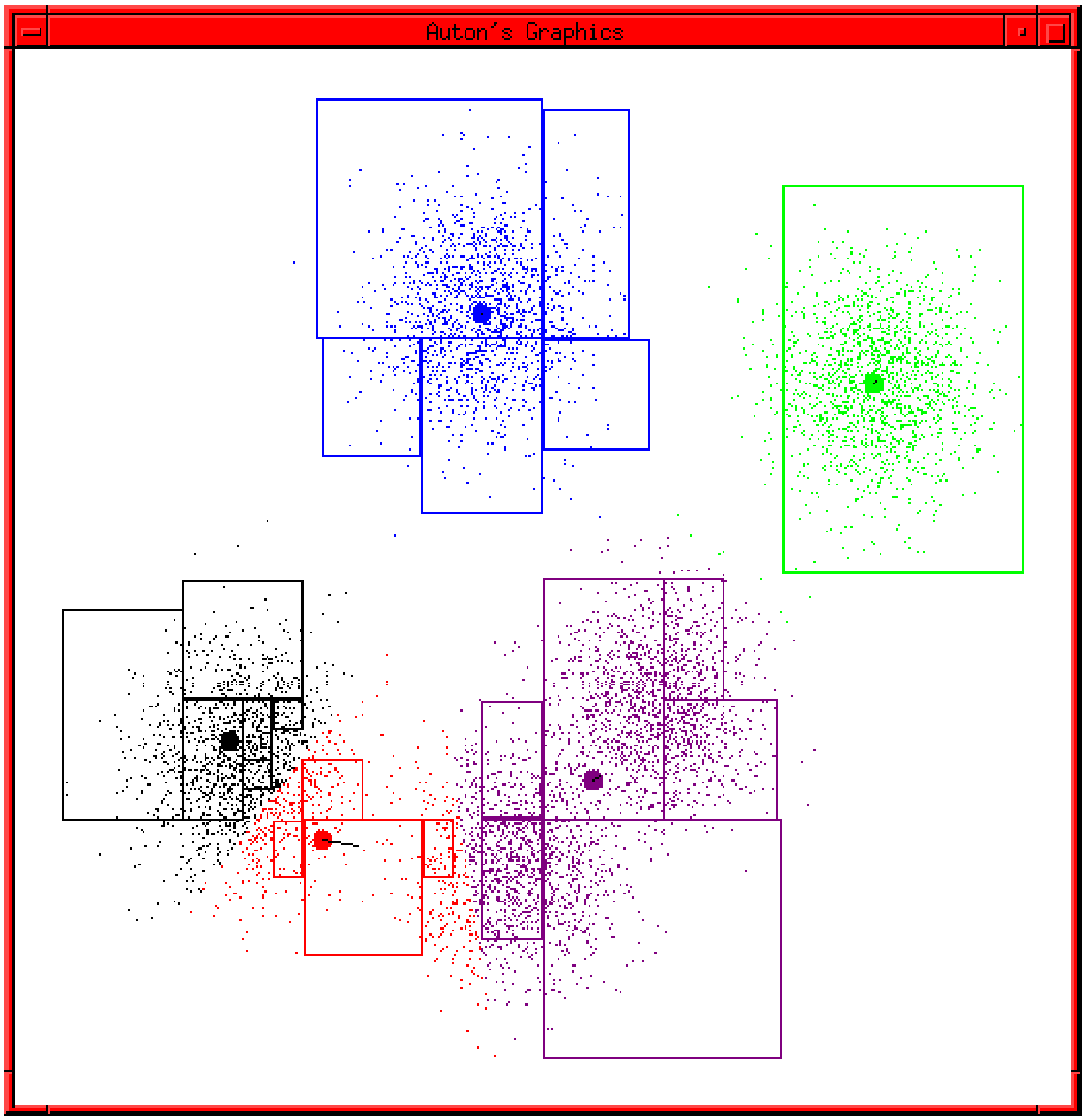
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



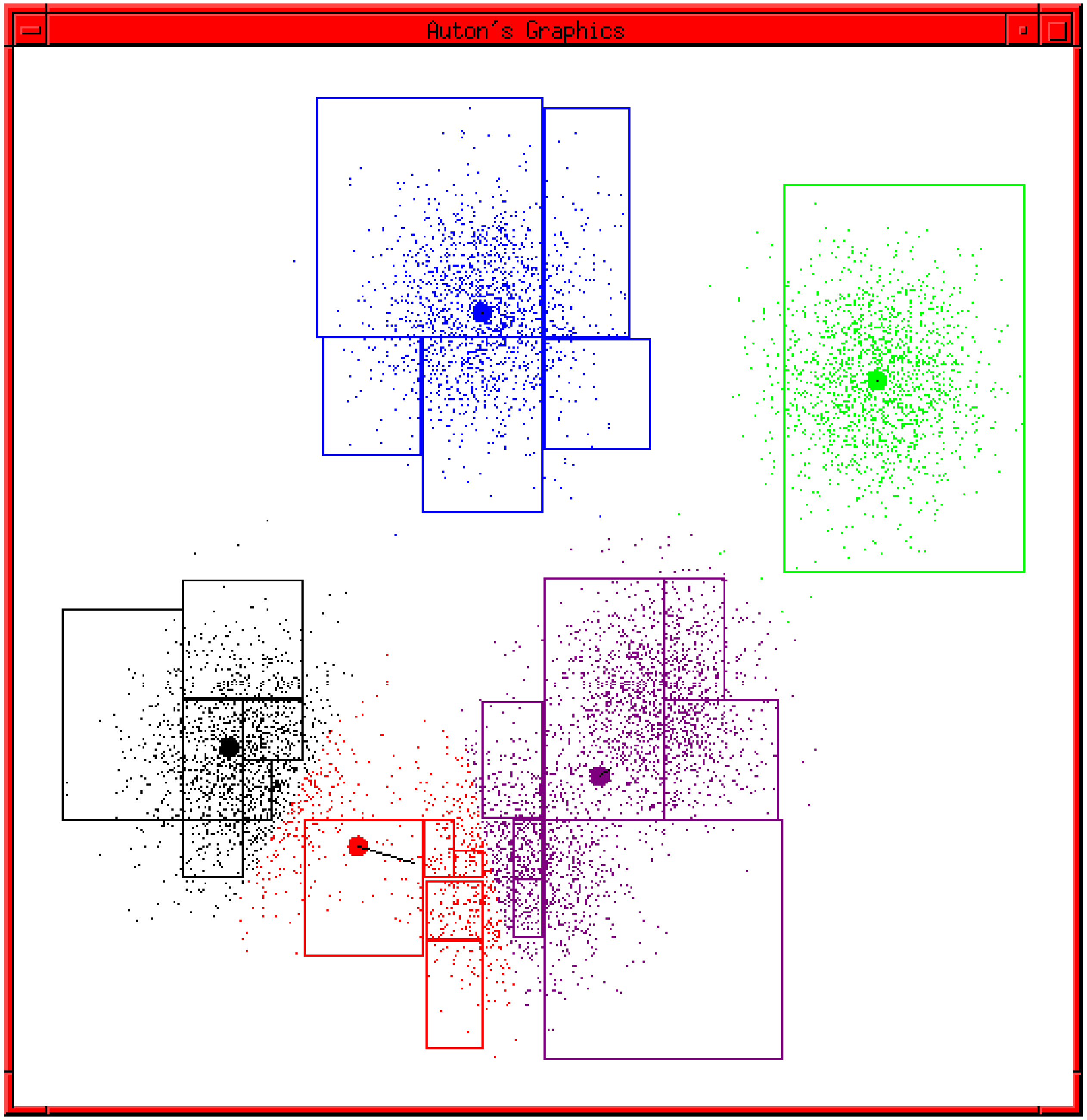
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



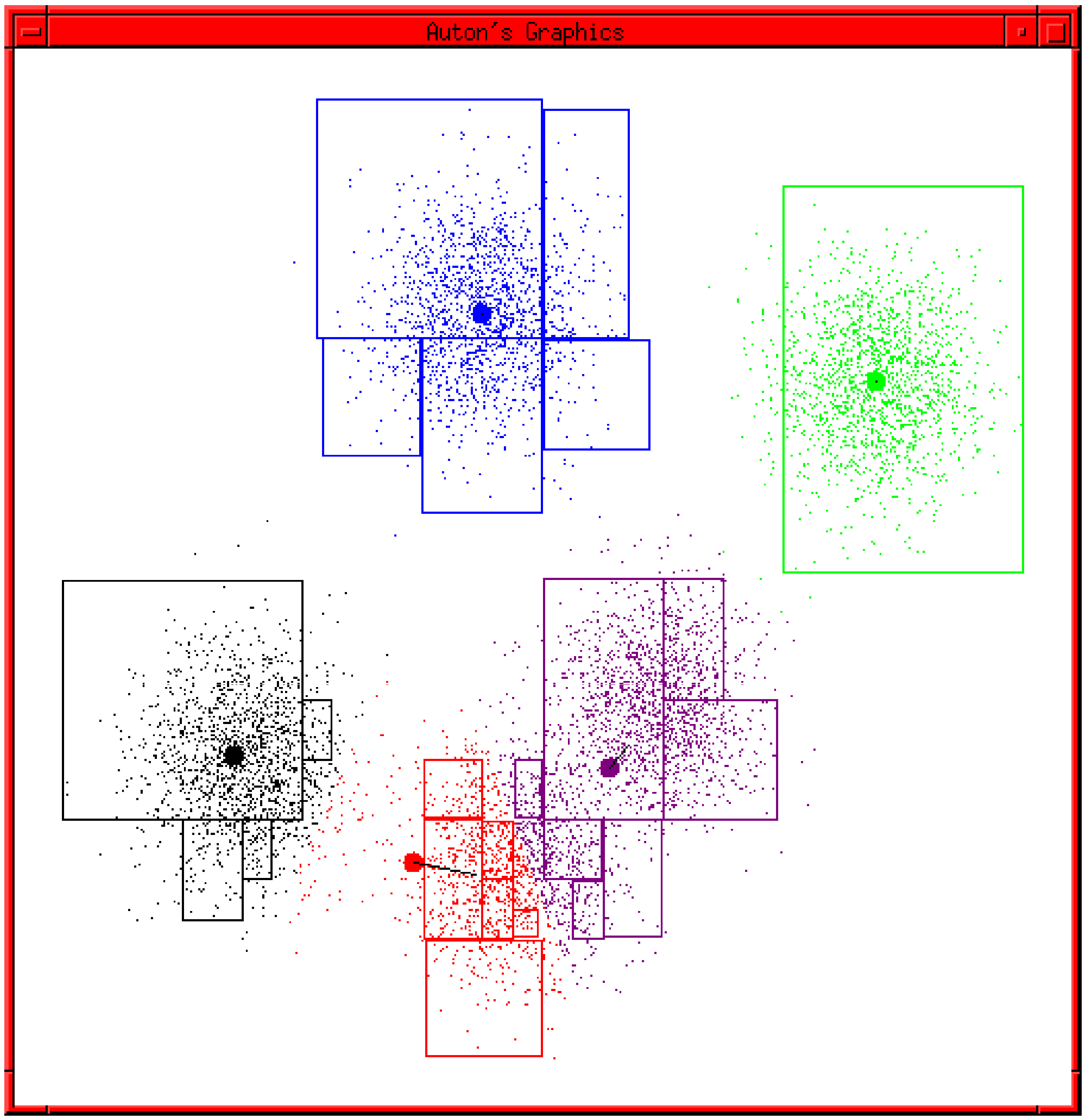
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



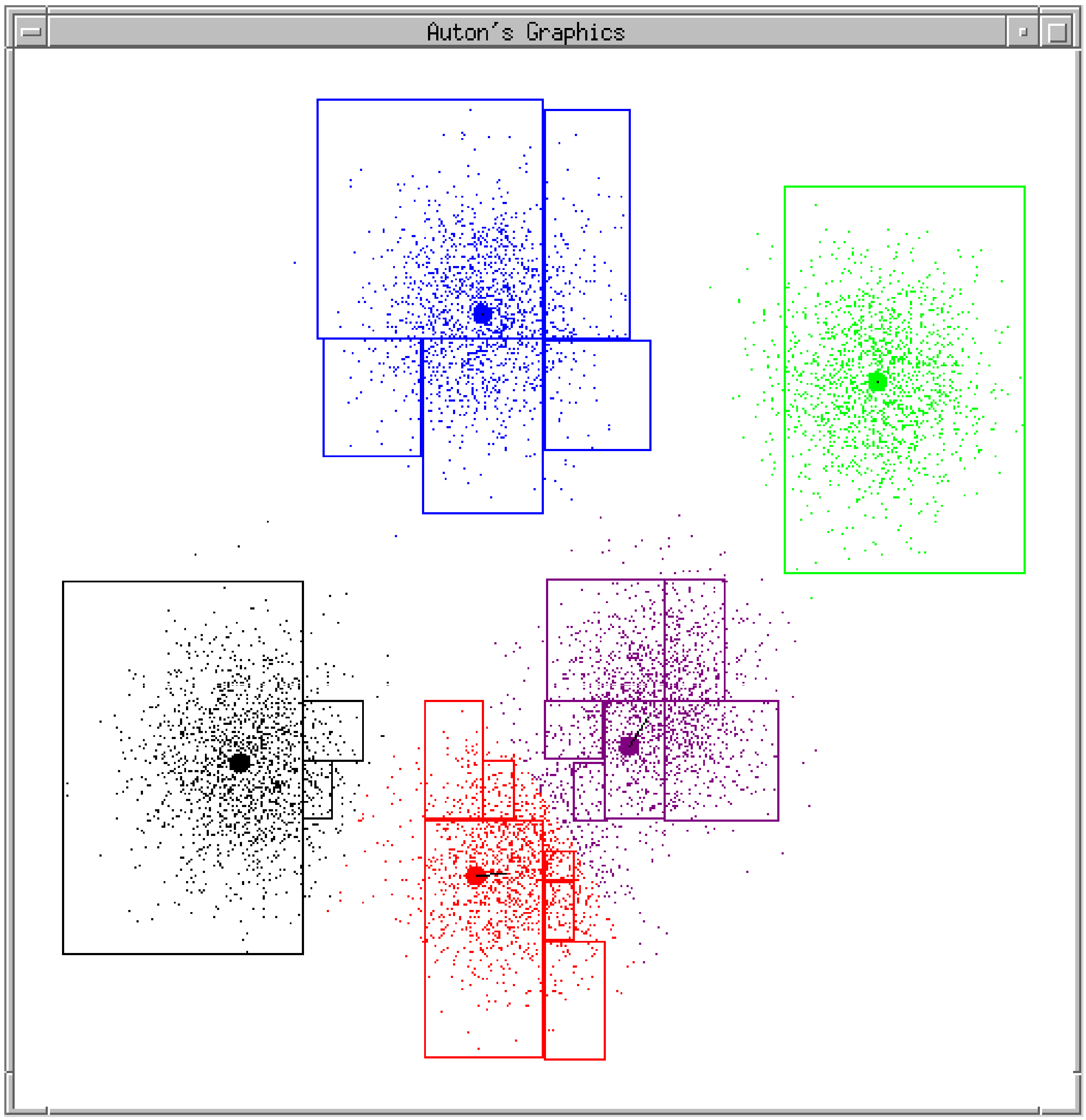
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



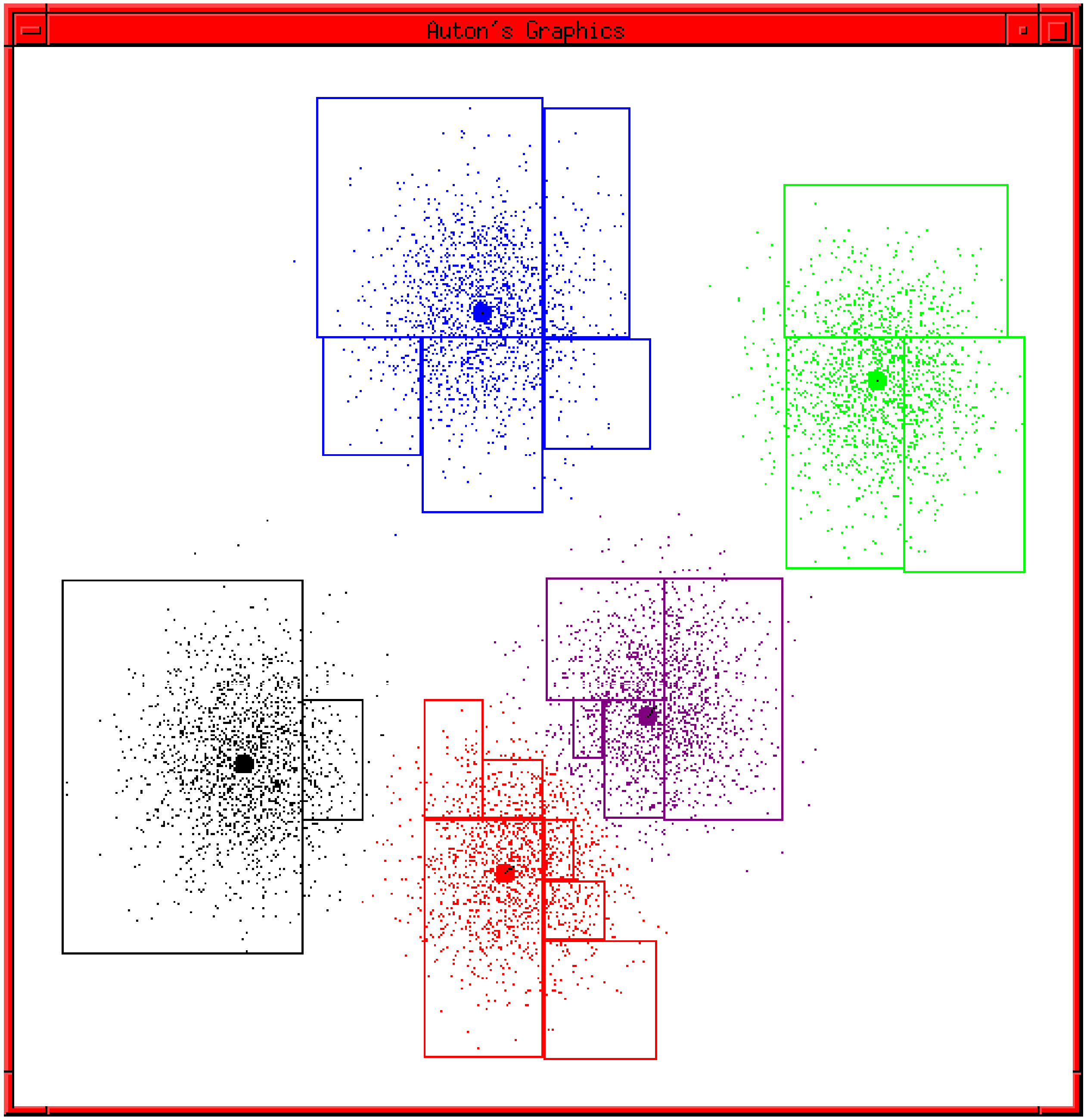
K-means

Here's an example run of k-means, generated by Dan Pelleg's software.



K-means

Here's the final result when the algorithm converges.



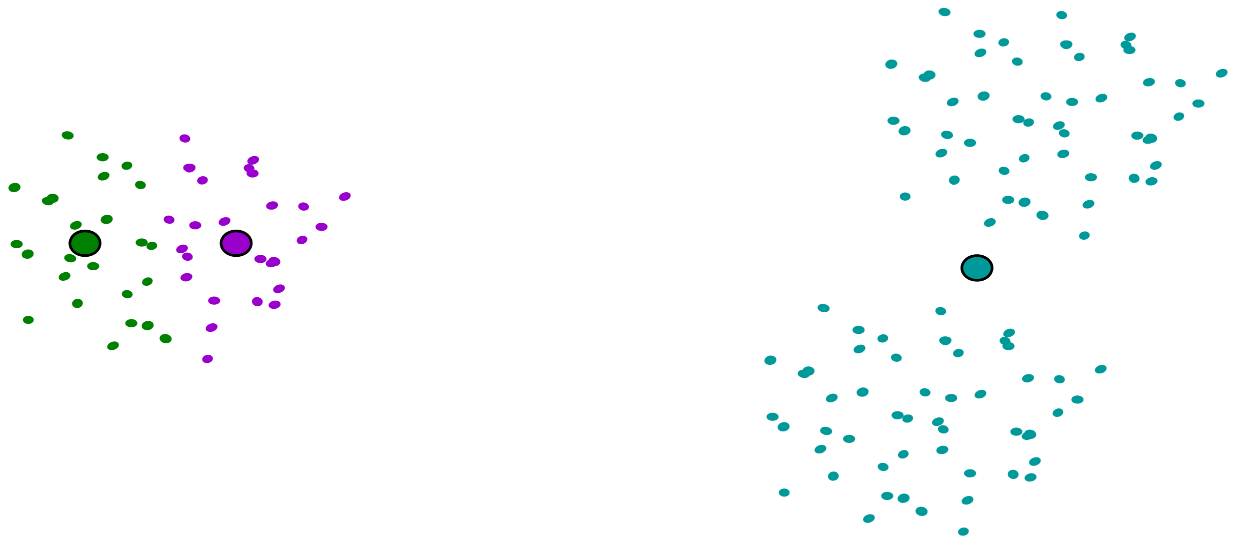
K-means clustering

Question 1: Will k-means always converge to a solution?

Yes! Distortion decreases monotonically, and it will end in a state where neither type of move will further reduce the distortion.

Question 2: Will k-means always find the optimal solution?

No! Here is one example where k-means converges to a locally optimal solution that does not have the global minimum distortion.



K-means clustering

<u>Question 1</u>: Will k-means always converge to a solution?

Yes! Distortion decreases monotonically, and it will end in a state where neither type of move will further reduce the distortion.

<u>Question 2</u>: Will k-means always find the optimal solution?

No! Here is one example where k-means converges to a locally optimal solution that does not have the global minimum distortion.

<u>Question 3</u>: How can we avoid these poor local optima?

K-means is a form of hill-climbing, so we can use many of the same tricks!

1. Run multiple times with different start states, and choose the best result.
2. Allow some moves that increase distortion, as in simulated annealing.
3. Choose a start state that is less likely to result in a poor local optimum.

Center 1 = randomly chosen data point
Center 2 = data point that's farthest from center 1
Center 3 = data point that's farthest from the closest of centers 1 and 2, etc.

K-means clustering

Question 4: How can we choose the number of centers k?

Run k-means multiple times with different values of k, and choose the k with minimum distortion???

Bad idea! Minimum distortion occurs when k = N, and each cluster contains only one point.

Better idea: choose the k that minimizes a measure of distortion with a penalty for more complex models.

Schwarz criterion: minimize (distortion + λk), where λ is a constant, proportional to (# of attributes) x log(# of records).

(Many other criteria have also been considered!)

In general, we can use this criterion to pick the "best" of any set of clusterings, e.g. which links to cut for a given hierarchical clustering.

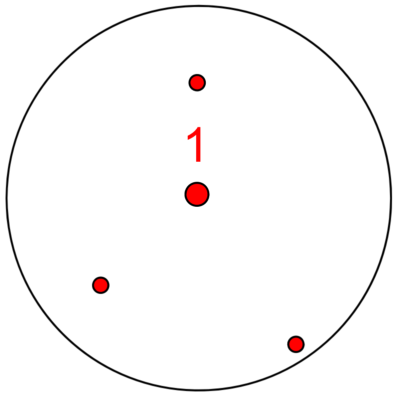
Leader clustering

“massive streaming data”
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points (“leaders”) and summary information about the points similar to each leader.

For each data point x_i: if x_i is within distance T of any leader, add to nearest leader’s group, otherwise make x_i a leader.



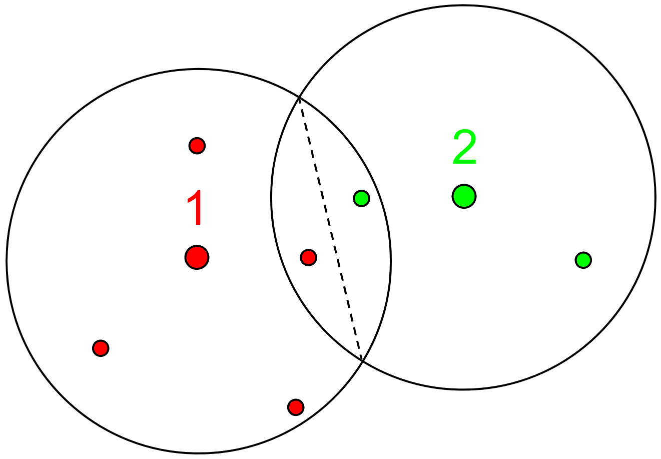
Leader clustering

"massive streaming data"
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points ("leaders") and summary information about the points similar to each leader.

For each data point x_i: if x_i is within distance T of any leader, add to nearest leader's group, otherwise make x_i a leader.



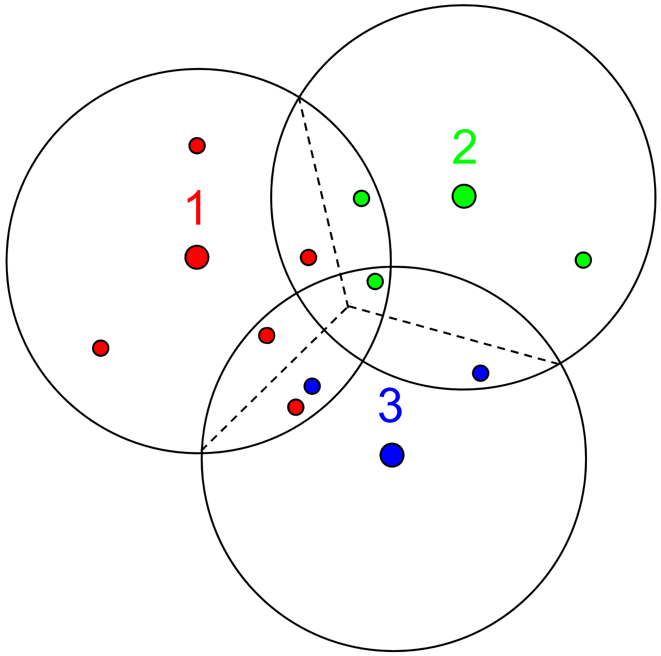
Leader clustering

"massive streaming data"
Billions of sensor readings
for a complex system

What if the dataset is so huge that we can only look at each data point once before discarding it?

We keep only a small subset of representative points ("leaders") and summary information about the points similar to each leader.

For each data point x_i: if x_i is within distance T of any leader, add to nearest leader's group, otherwise make x_i a leader.



Advantages of leader clustering

Very fast: only need to look at leaders for each point

Guarantees group diameter < 2T, group leaders at least T apart

Disadvantages of leader clustering

Order-dependent: first point always a leader, initial clusters tend to be larger

Points may not be assigned to nearest cluster center

The many uses of clustering

Clustering provides a useful **summary** of a large dataset, representing the N data records using only k << N clusters.

This can be used for **exploratory data analysis**, enabling us to understand the underlying sub-structure of the data.

Value of k? Shape? Size? Hierarchy?

Important to choose correct k value

k usually small

We can often improve the performance of model-based prediction by learning separate models for each group.

We can also use clustering to detect **anomalies**, by finding points that are far from any cluster center.

Choose k large as possible

We can use clustering to speed up instance-based prediction (e.g. k-NN) by reducing the training set size.

Finally, we can use clustering to handle massive streaming data by maintaining only the cluster summaries in memory.

References

- A.K Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, 1988.
- A.K. Jain et al. Data clustering: a review. *ACM Computing Surveys* 31(3), 1999.
- Weka has implementations of k-means, EM, and a hierarchical clustering method called Cobweb.
- The Auton Laboratory (www.autonlab.org) also has very fast k-means software, created by Dan Pelleg.

This lecture is partially based on an excellent set of lecture slides by Andrew Moore, available at: http://www.cs.cmu.edu/~awm/tutorials

Large Scale Data Analysis for Policy
90-866, Fall 2012

Lectures 7-8: Bayesian Networks for Knowledge Representation

Why Bayesian networks?

- An easily interpretable graphical representation of the relationships between a set of variables.
- Bayes Nets can be specified manually or learned automatically from data, and enable computationally efficient probabilistic inferences.
- Many practical and successful applications in medicine, manufactuing, failure diagnosis:
  - Diagnosis: infer Pr(problem type | symptoms)
  - Prediction: infer probability distributions for values that are expensive or impossible to measure.
  - Anomaly Detection: detect observations that are very unlikely (i.e. have low probabilities given the model).
  - Active Learning: choose the most informative diagnostic test to perform given these observations.

Introduction to Bayesian networks

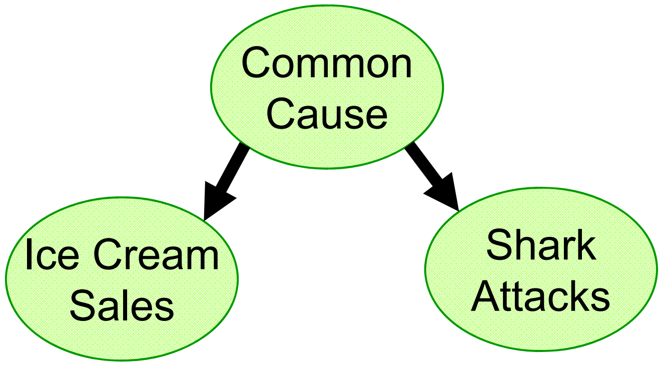
Recent Dow-Jones Change	Number of Ice Creams Sold Today	Number of Shark Attacks Today
UP	3500	4
STEADY	41	0
UP	2300	5
DOWN	3400	4
UP	18	0
STEADY	105	0
STEADY	4	0
STEADY	6310	3
UP	70	0

More ice creams =
More shark attacks!

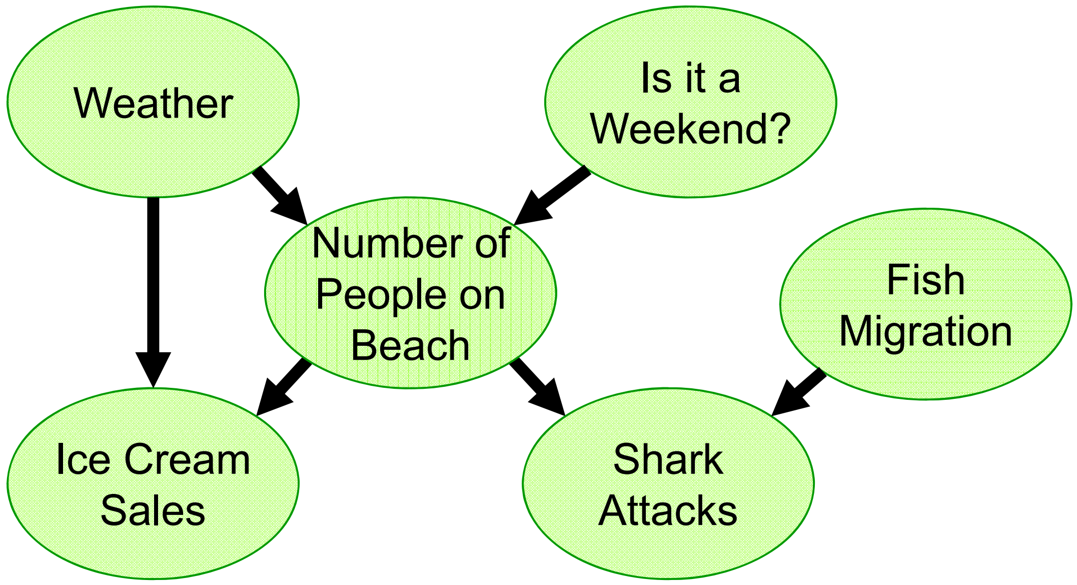
Does eating ice cream
cause shark attacks?

Do shark attacks affect
ice cream consumption?

Perhaps the two events
have a common cause!



Introduction to Bayesian networks



Many other factors may influence some or all of these variables…

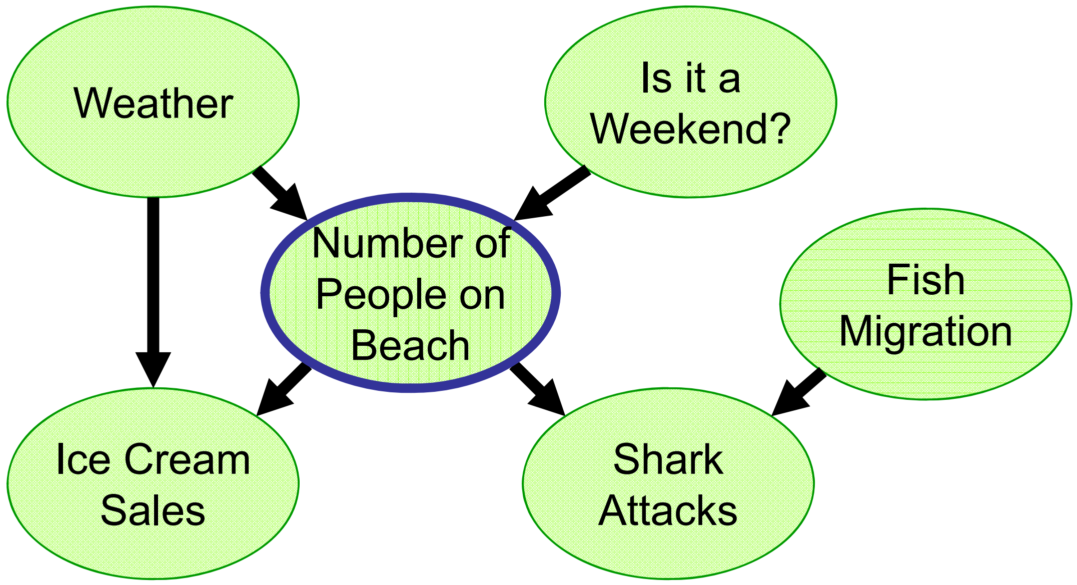
What we have here is a way of representing probabilistic relationships between variables. We call this a **Bayesian network**.

Causal interpretation: Link X → Y means that variable X directly **causes** variable Y.

Probabilistic interpretation: The links encode **conditional dependencies** between variables.

"X and Y are conditionally independent given Z":
Pr(X | Z) = Pr (X | Y, Z) and Pr(Y | Z) = Pr(Y | X, Z)

Introduction to Bayesian networks



Many other factors may influence some or all of these variables…

What we have here is a way of representing probabilistic relationships between variables. We call this a **Bayesian network**.

Each node has a probability distribution that is conditioned on its parents' values.

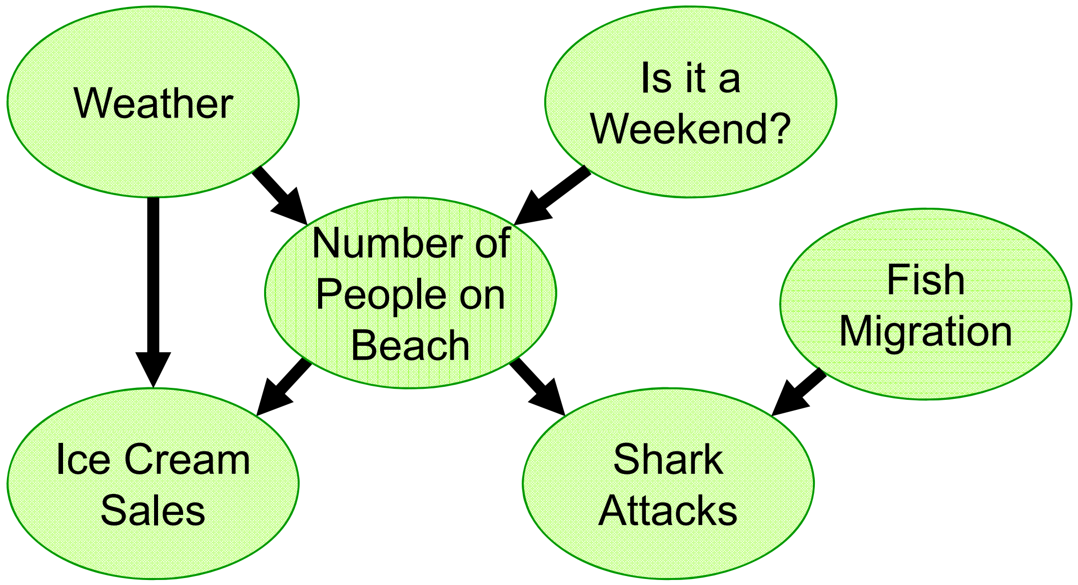
Sunny, Weekend:
Pr (Beach Crowded) = 90%
Sunny, Not Weekend:
Pr (Beach Crowded) = 40%
Not Sunny, Weekend:
Pr (Beach Crowded) = 20%
Not Sunny, Not Weekend:
Pr (Beach Crowded) = 1%

Causal interpretation: Link X → Y means that variable X directly **causes** variable Y.

Probabilistic interpretation: The links encode **conditional dependencies** between variables.

"X and Y are conditionally independent given Z":
Pr(X | Z) = Pr (X | Y, Z) and Pr(Y | Z) = Pr(Y | X, Z)

Introduction to Bayesian networks



Many other factors may influence some or all of these variables…

What we have here is a way of representing probabilistic relationships between variables. We call this a **Bayesian network**.

In a Bayes Net, each node is **conditionally independent** of its non-descendents given its parents.

Causal interpretation: Link X → Y means that variable X directly **causes** variable Y.

Probabilistic interpretation: The links encode **conditional dependencies** between variables.

"X and Y are conditionally independent given Z":
Pr(X | Z) = Pr (X | Y, Z) and Pr(Y | Z) = Pr(Y | X, Z)

For example, if you already know the number of people on the beach and have fish migration data, knowing today's weather doesn't give you any more information about shark attacks.

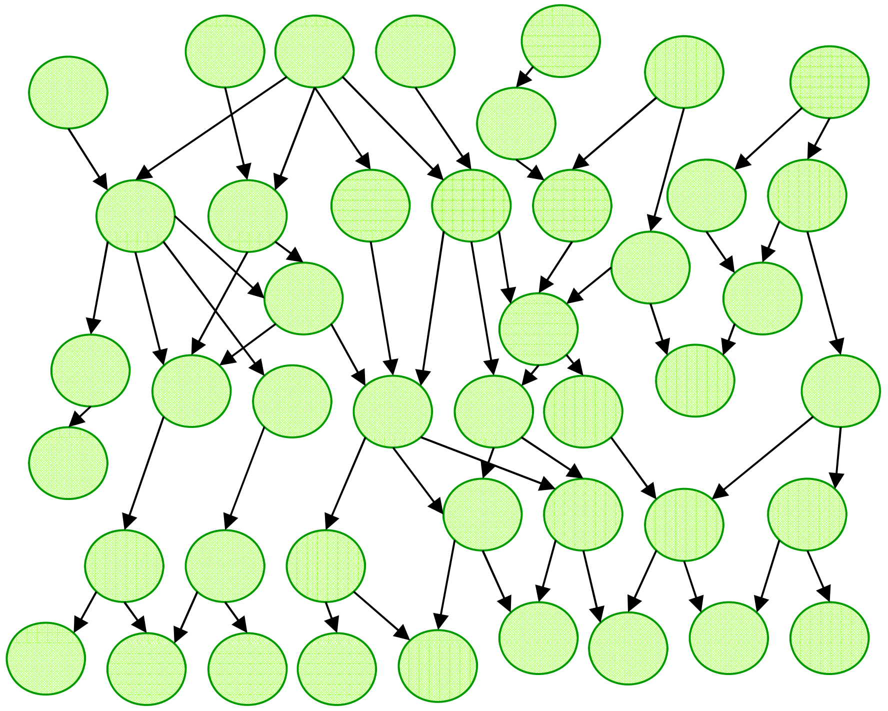
CI(Shark Attacks, Weather | Number of People on Beach, Fish Migration)

Real-world Bayes Nets

Bayesian networks for real-world application domains may have hundreds or thousands of nodes.

They can be built manually, consulting domain experts for the structure and probabilities.

More often, the probabilities (and in some cases, the structure) are **learned** from training data.



Pathfinder (a diagnostic system for lymph-node diseases) uses a Bayes Net with 60 diseases, 100 symptoms, and 14,000 probabilities.

The Bayes Net was constructed manually by human experts: 8 hours to choose variables, 35 hours for structure, 40 hours for probabilities.

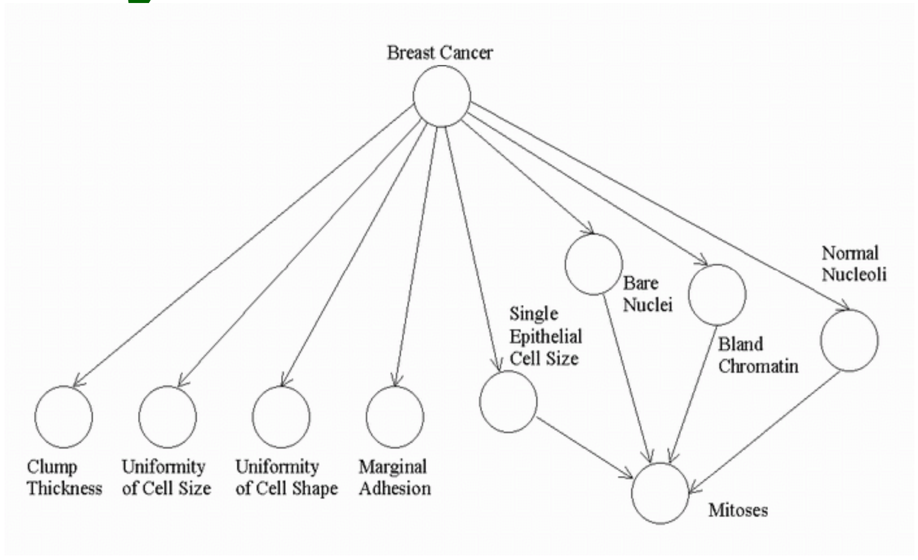
Pathfinder was shown to outperform human experts in diagnosis accuracy, and is currently being extended to many other medical domains.

Real-world Bayes Nets

Bayesian networks for real-world application domains may have hundreds or thousands of nodes.

They can be built manually, consulting domain experts for the structure and probabilities.

More often, the probabilities (and in some cases, the structure) are **learned** from training data.



Medical diagnosis is a common application of Bayes Nets: here's a simple network which can be used to infer Pr(breast cancer | symptoms).

Pathfinder (a diagnostic system for lymph-node diseases) uses a Bayes Net with 60 diseases, 100 symptoms, and 14,000 probabilities.

The Bayes Net was constructed manually by human experts: 8 hours to choose variables, 35 hours for structure, 40 hours for probabilities.

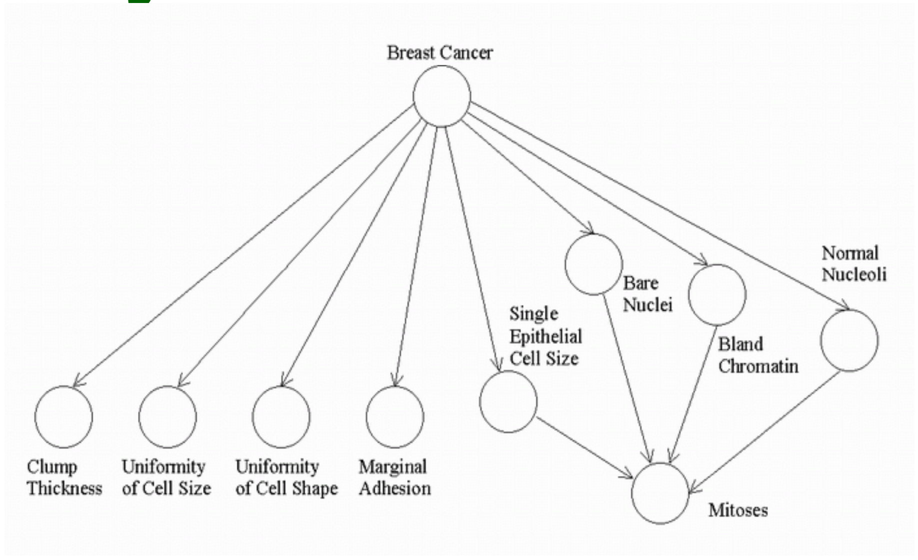
Pathfinder was shown to outperform human experts in diagnosis accuracy, and is currently being extended to many other medical domains.

Real-world Bayes Nets

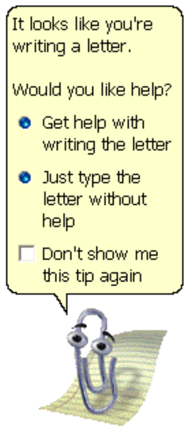
Bayesian networks for real-world application domains may have hundreds or thousands of nodes.

They can be built manually, consulting domain experts for the structure and probabilities.

More often, the probabilities (and in some cases, the structure) are **learned** from training data.



Medical diagnosis is a common application of Bayes Nets: here's a simple network which can be used to infer Pr(breast cancer | symptoms).

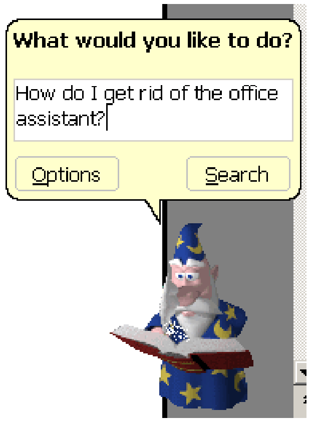

It looks like you're writing a letter.

Would you like help?

- Get help with writing the letter
- Just type the letter without help

☐ Don't show me this tip again

Another common application of Bayes Nets is diagnosis and troubleshooting of failures in computers, networks, circuits, devices, manufacturing systems, etc.


What would you like to do?

How do I get rid of the office assistant?

Example: Microsoft Office Assistant uses Bayes Nets for software technical support.

Joint probability distributions

a.k.a. "Why are Bayes Nets so useful for representing probabilities?"

To create the joint distribution of M discrete-valued variables:

Make a truth table listing all combinations of values of your variables. If there are M binary variables, the table has 2^M rows.

A	B	C
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Joint probability distributions

a.k.a. “Why are Bayes Nets so useful for representing probabilities?”

To create the joint distribution of M discrete-valued variables:

Make a truth table listing all combinations of values of your variables. If there are M binary variables, the table has 2^M rows.

For each combination of values, say how probable it is. These probabilities must sum to 1.

What if we have M = 100 variables: how can we use less than 2^{100} rows?

Use information about conditional independencies between variables to infer a Bayesian network structure!

A	B	C	Prob
0	0	0	0.30
0	0	1	0.05
0	1	0	0.10
0	1	1	0.05
1	0	0	0.05
1	0	1	0.10
1	1	0	0.25
1	1	1	0.10

Joint probability distributions

a.k.a. "Why are Bayes Nets so useful for representing probabilities?"

To create the joint distribution of M discrete-valued variables:

Make a truth table listing all combinations of values of your variables. If there are M binary variables, the table has 2^M rows.

For each combination of values, say how probable it is. These probabilities must sum to 1.

What if we have M = 100 variables: how can we use less than 2^{100} rows?

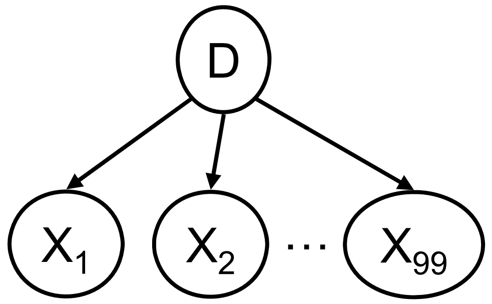
Use information about conditional independencies between variables to infer a Bayesian network structure!

Here's a simple example:

D = "there is a disease outbreak"
$X_1..X_{99}$ = "person i goes to the hospital"

If D is known, also knowing X_i does not change the probability of X_j:

$$\Pr(X_j \mid D, X_i) = \Pr(X_j \mid D)$$



$$\Pr(D, X_1..X_{99}) = \Pr(D) \prod_{i=1..99} \Pr(X_i \mid D)$$

Bayes Nets often allow a much more compact representation of the joint distribution!

(In general, how many probabilities do we need?)

Building a Bayes Net

Small Bayes Nets are easy to build by hand, assuming that we understand the relationships between variables and are able to estimate their conditional probabilities.

Large Bayes Nets may require many person-hours to build, but they can also be **learned** automatically from data.

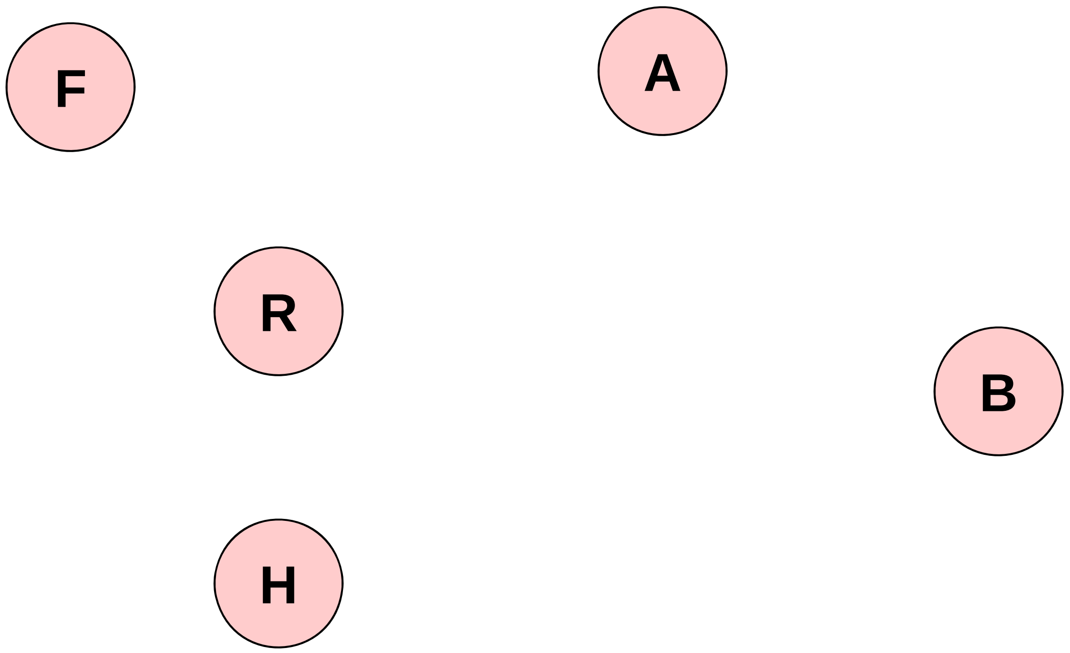
For example, let's assume that we want to build a Bayes Net to determine whether a terrorist anthrax attack has occurred.

1. An anthrax attack is likely to increase the level of respiratory illness.
2. Seasonal influenza is also likely to cause an increase in respiratory illness.
3. The CDC has a **hospital surveillance system** which alerts when the number of ED visits is abnormally high, and has additionally deployed **bio-sensors** for airborne anthrax detection.
4. The hospital surveillance system and the bio-sensors are not perfect: both false alarms and missed outbreaks are possible.

Define the following variables:

F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert

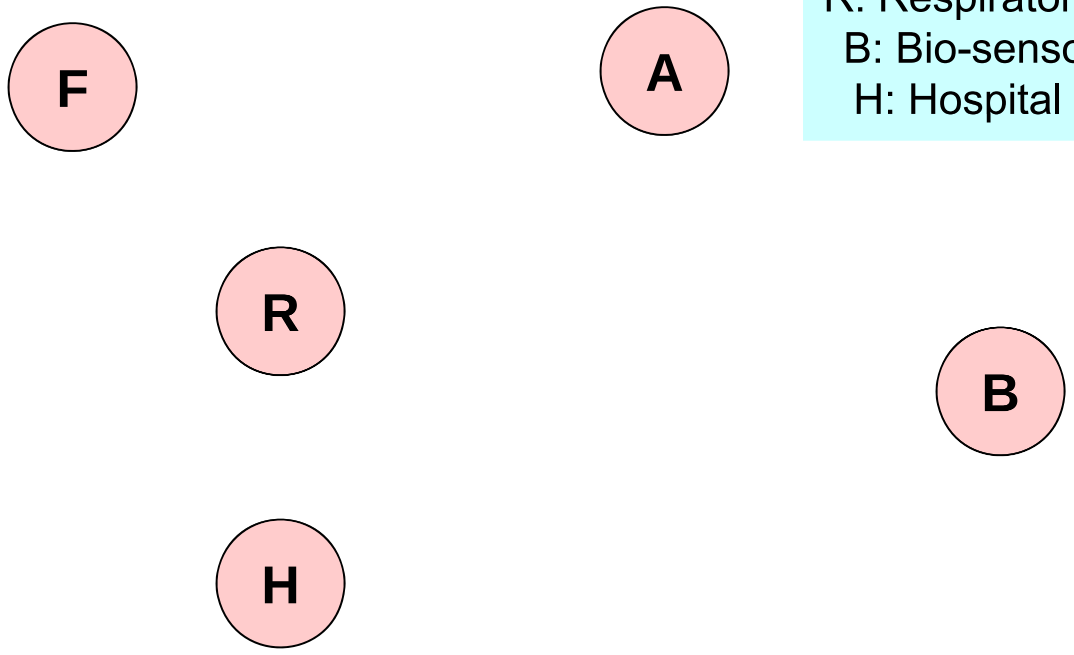
Building a Bayes Net



Step 1: Choose a set of relevant variables, and represent each variable by a node.

Building a Bayes Net

F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert



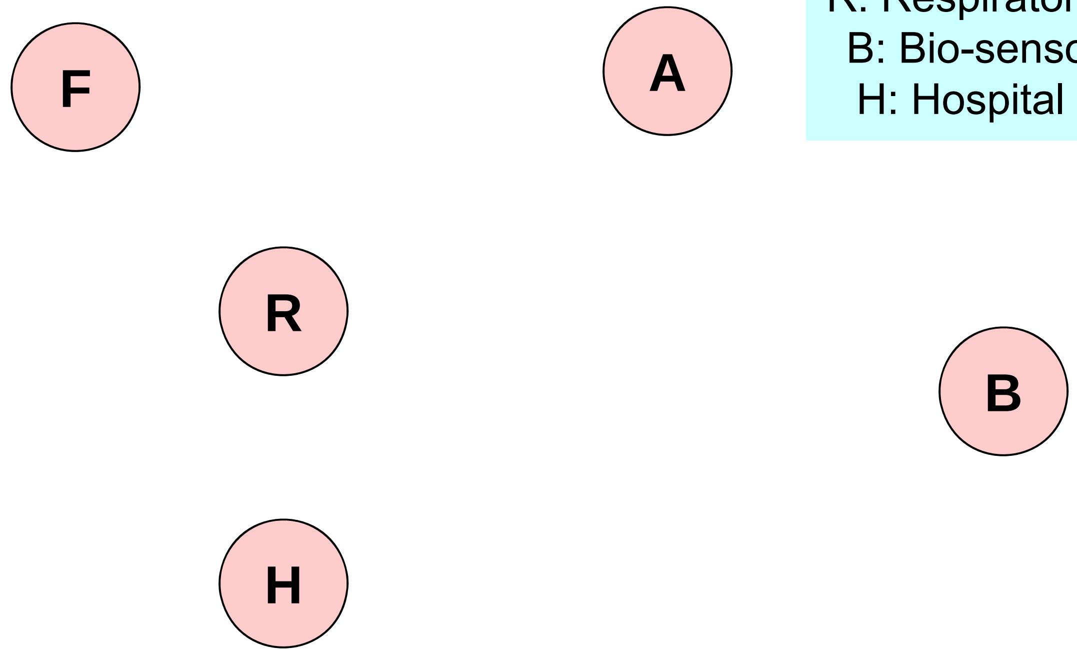
Step 2: Choose an ordering for the variables $X_1..X_M$, such that if X_i influences X_j, then $i < j$.

Hint: put environmental and event variables first, then latent variables, then observations.

Any ordering will produce a valid Bayes Net structure, but using the causal information will produce more compact (fewer links) and more interpretable structures.

Building a Bayes Net

F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert

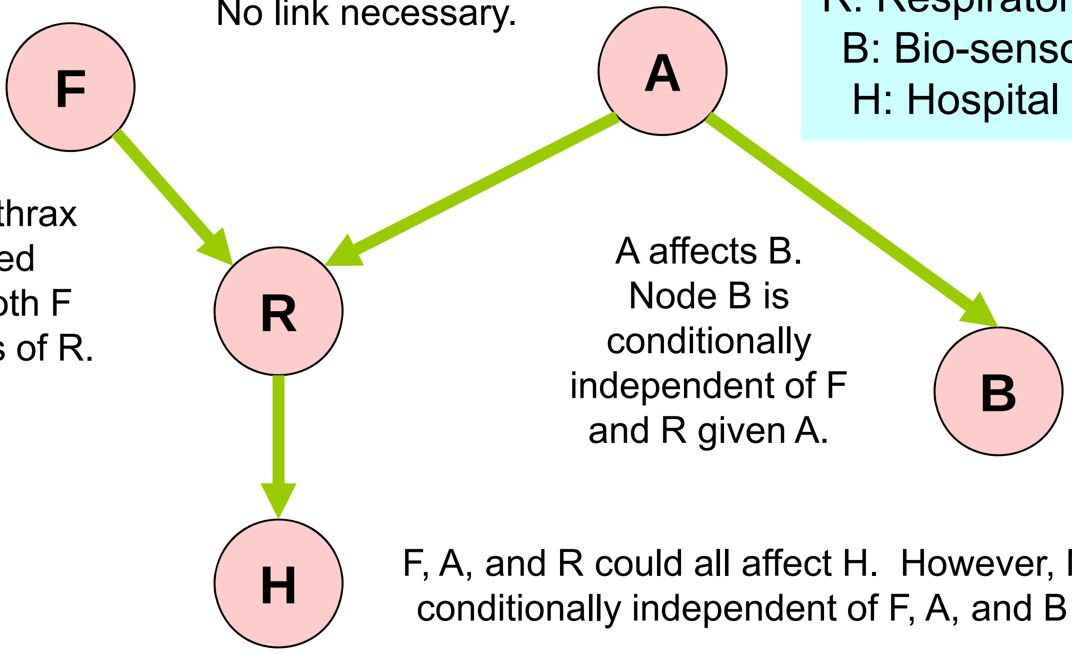


<u>Step 3</u>: Add links.

- The link structure must be acyclic.
- If node X is given parents $Q_1,Q_2,\ldots Q_m$, you are promising that any variable that's a non-descendent of X is conditionally independent of X given $\{Q_1,Q_2,..Q_m\}$

Building a Bayes Net

F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert



F is the first variable, so it cannot have any parents.

Whether an anthrax attack occurs does not depend on flu season. No link necessary.

Since either flu or anthrax can cause increased respiratory illness, both F and A must be parents of R.

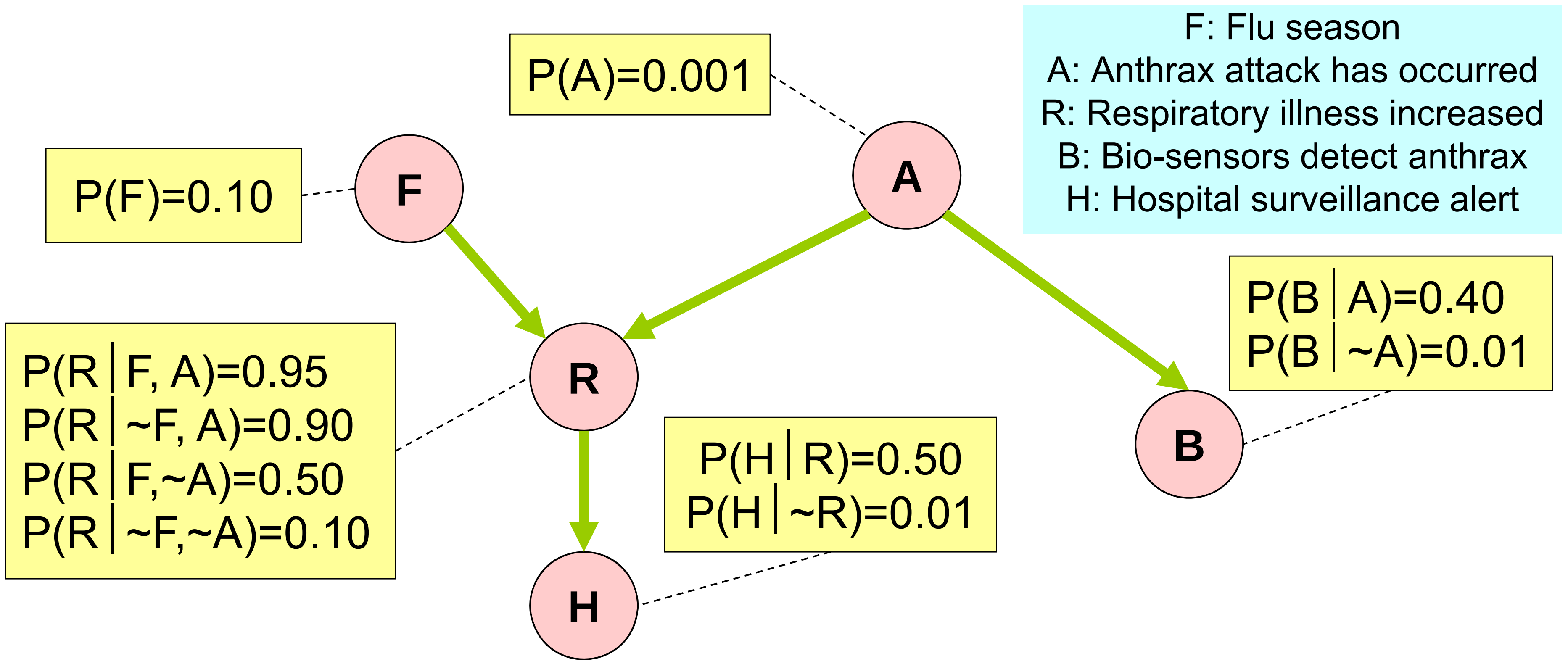
A affects B. Node B is conditionally independent of F and R given A.

F, A, and R could all affect H. However, Node H is conditionally independent of F, A, and B given R.

Step 3: Add links.

- For each variable X_i (for i = 2…M), choose a minimal subset of parents from $X_1..X_{i-1}$, such that X_i is conditionally independent of the rest of $X_1..X_{i-1}$ given its parents.

Building a Bayes Net



Step 4: Add a conditional probability table for each node.

- The table for node X must list Pr(X | Parents(X)) for each value of X and each combination of parent values.
- If X is binary, we know that Pr(~X | Parents) = 1 – Pr(X | Parents), and do not need to write this explicitly.

Building a Bayes Net

Practice problem: suppose that we're building a nuclear power station, and want to report when the core temperature is low.

The gauge is meant to read the temperature of the core, and the alarm is meant to sound if the gauge reads a low temperature.

However, the gauge or the alarm (or both) could be faulty.

Which Bayesian network structure makes the most sense for these five variables?

Define the following variables:
G = Gauge reads low.
C = Core temperature is low.
FG = Gauge is faulty.
FA = Alarm is faulty.
A = Alarm sounds.

Building a Bayes Net

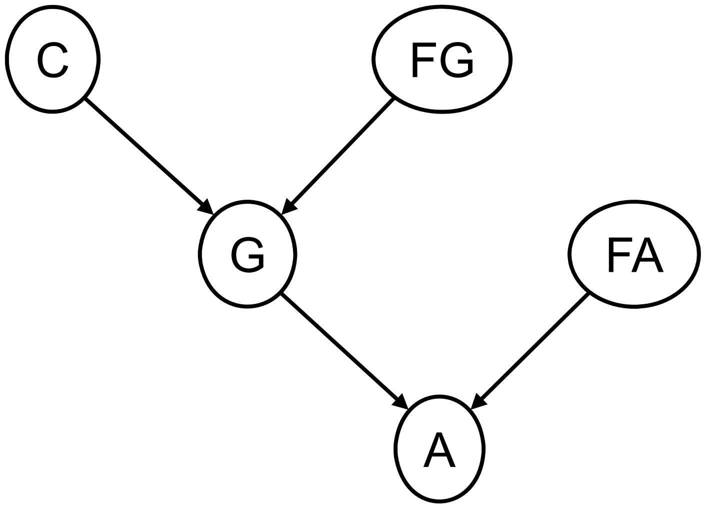
Practice problem: suppose that we're building a nuclear power station, and want to report when the core temperature is low.

The gauge is meant to read the temperature of the core, and the alarm is meant to sound if the gauge reads a low temperature.

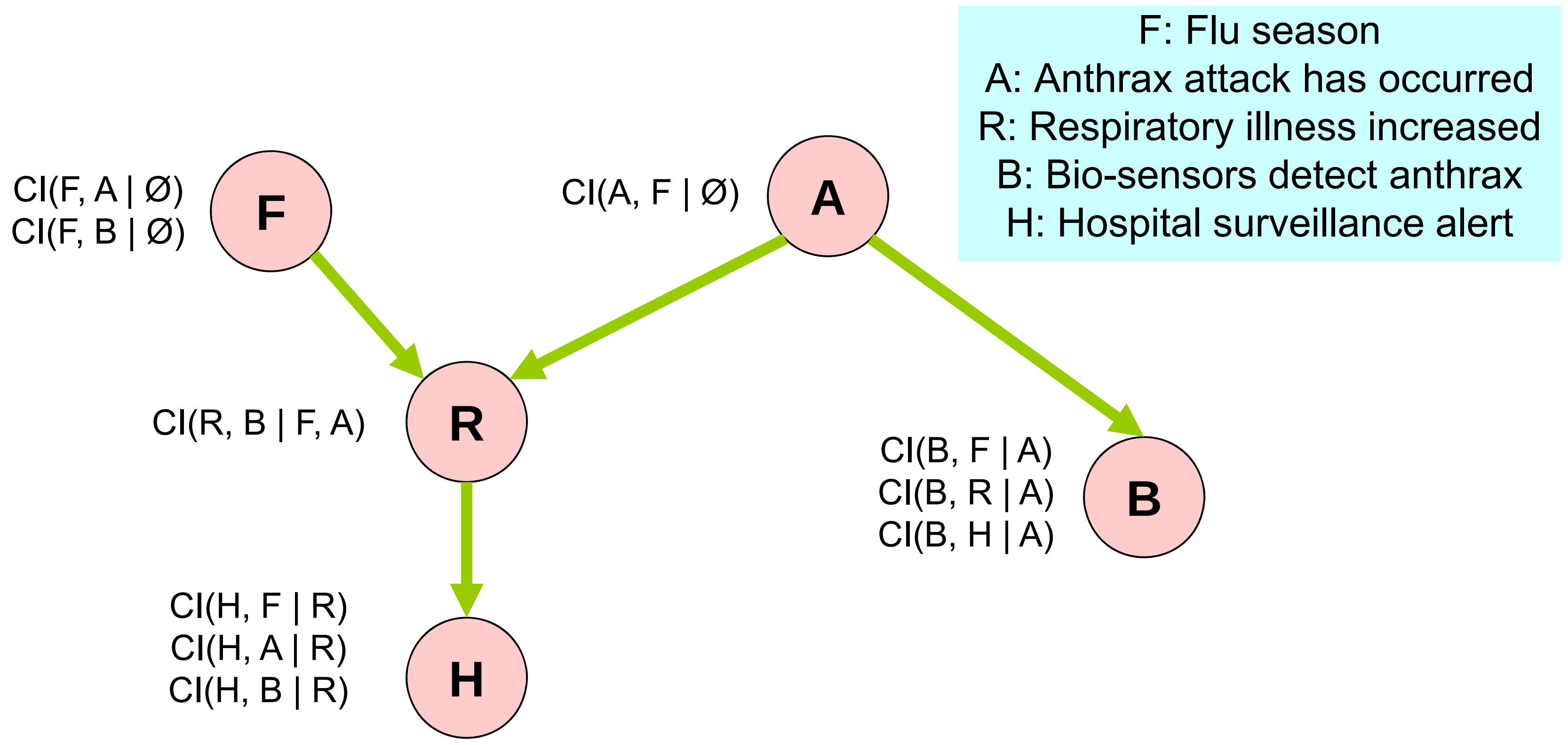
However, the gauge or the alarm (or both) could be faulty.

Which Bayesian network structure makes the most sense for these five variables?

Define the following variables:
G = Gauge reads low.
C = Core temperature is low.
FG = Gauge is faulty.
FA = Alarm is faulty.
A = Alarm sounds.

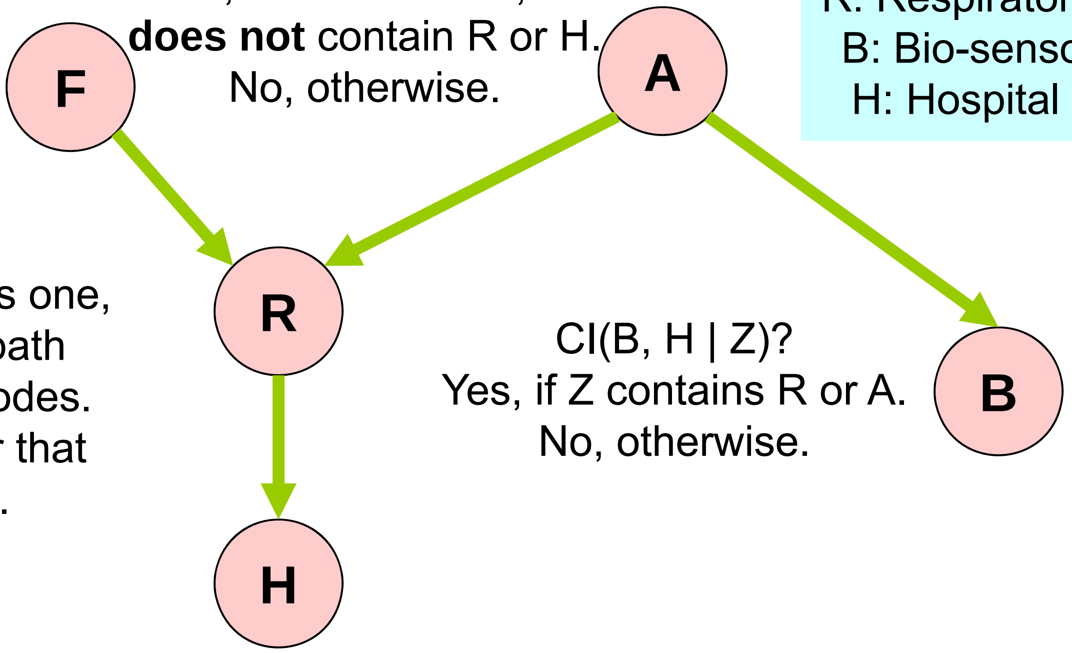

$$\begin{tikzcd} C \arrow[dr] & & FG \arrow[dl] & \\ & G \arrow[dr] & & FA \arrow[dl] \\ & & A & \end{tikzcd}$$

Interpreting a Bayes Net structure



- Key property: each node is conditionally independent of all its non-descendants in the tree, given its parents.
- Two unconnected variables may still be correlated.
- Whether any two variables are conditionally independent can be deduced from a Bayes Net using “d-separation”.

d-separation



CI(F, B | Z)?
Yes, if Z contains A, or **does not** contain R or H.
No, otherwise.

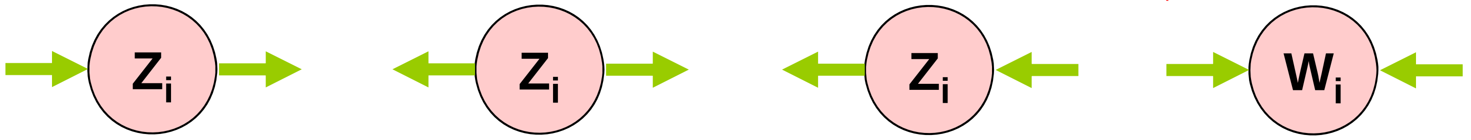
F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert

In a **polytree** like this one, there is only one path between any two nodes. Just check whether that path is blocked.

CI(B, H | Z)?
Yes, if Z contains R or A.
No, otherwise.

- Nodes X_i and X_j are conditionally independent given a set of nodes Z iff every undirected path between X_i and X_j is "blocked" by at least one of the following: $(Z_i \in Z, W_i \notin Z)$

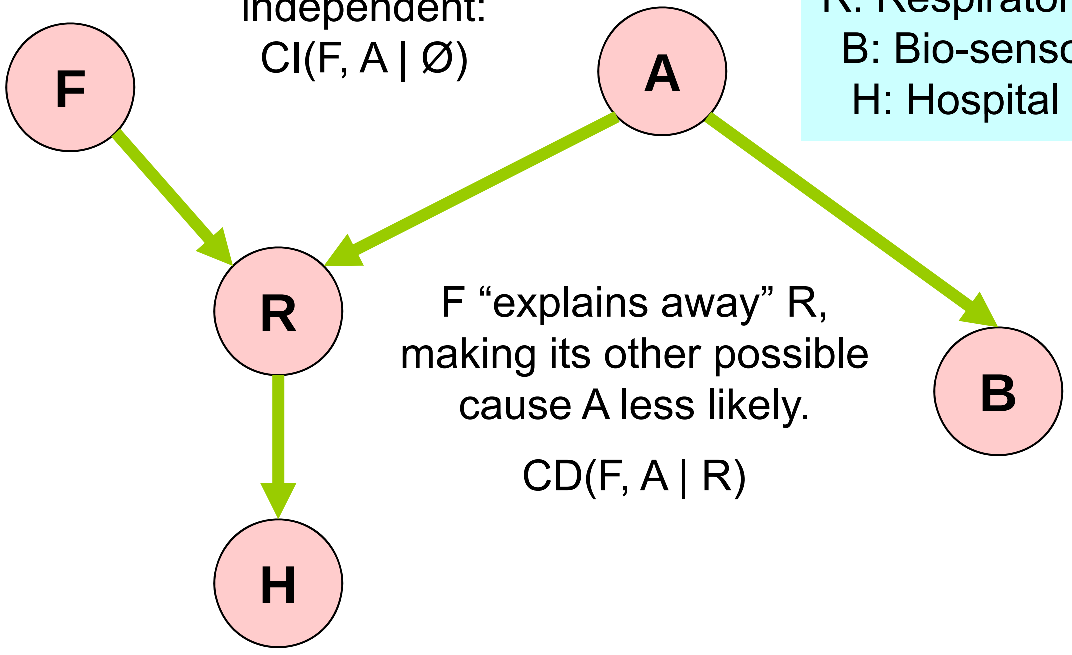
(and no descendent of W_i is in Z)



"Explaining away"

What if we know that respiratory illness has increased, and want to know whether an anthrax attack has occurred?

If it is flu season, the increase in respiratory illness is probably due to flu, not anthrax.



F and A are independent: CI(F, A | Ø)

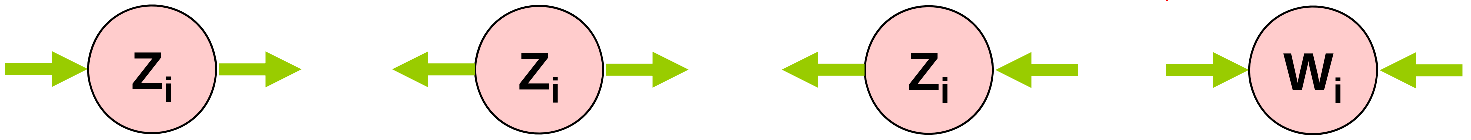
F "explains away" R, making its other possible cause A less likely.

CD(F, A | R)

F: Flu season
A: Anthrax attack has occurred
R: Respiratory illness increased
B: Bio-sensors detect anthrax
H: Hospital surveillance alert

- Nodes X_i and X_j are conditionally independent given a set of nodes Z iff every undirected path between X_i and X_j is "blocked" by at least one of the following: $(Z_i \in Z, W_i \notin Z)$

(and no descendent of W_i is in Z)

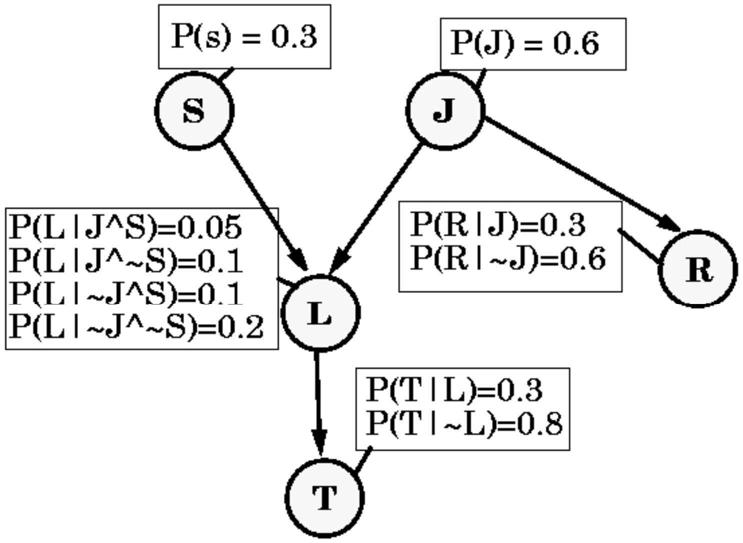


Where are we now?

- We can build a Bayesian network by hand, specifying the structure and the conditional probability tables.
- The network structure represents the conditional dependencies and independencies between variables, and can also have a causal interpretation.
- Bayes Nets are a more compact representation of the joint probability distribution: we only need to store a number of probabilities exponential in the number of parents per node, not the total number of nodes.
- We will now answer two main questions:
  - How can we use Bayes Nets for probabilistic **inference**? ("What is the probability of an anthrax attack, given that the hospital surveillance system and bio-sensors both alerted?")
  - How can we **learn** the Bayes Net structure and parameters automatically, using a large training dataset?

Bayes Net inference

Question 1: How to compute joint probabilities using a Bayes Net?



Compute Pr(S, ~J, L, ~R, T)

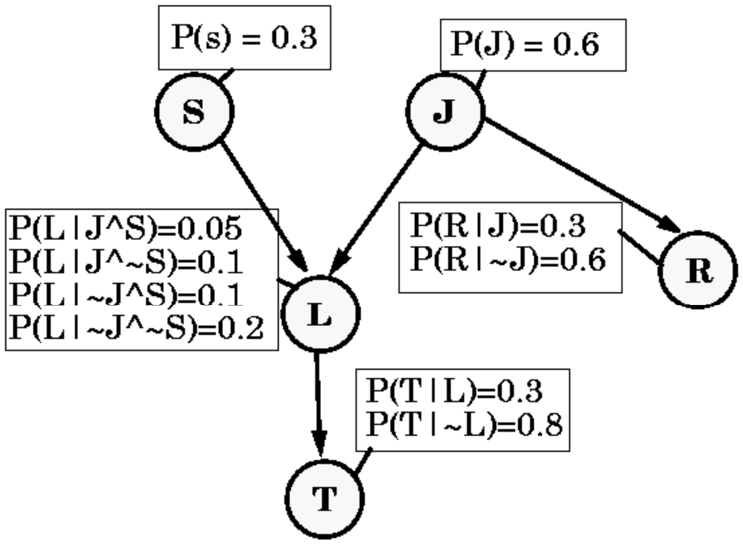
Answer: use conditional independence.
$\Pr(X_1..X_M) = \prod_{i=1..M} \Pr(X_i \mid \text{Parents}(X_i))$

Pr(S, ~J, L, ~R, T) =
Pr(S) Pr(~J) Pr(L | ~J, S) Pr(~R | ~J) Pr(T | L) =
(0.3) (0.4) (0.1) (0.4) (0.3) = 0.00144.

We can efficiently compute the joint probability of any given assignment of values to variables.

Bayes Net inference

Question 2: How to compute arbitrary conditional probabilities?



Compute Pr(S, T | ~J, L)

Express the conditional probability as a ratio:

Pr(S, T | ~J, L) = Pr(S, ~J, L, T) / Pr (~J, L)

Express numerator and denominator as sums of joint probabilities:

Pr(S, ~J, L, T) = Pr(S, ~J, L, R, T) + Pr(S, ~J, L, ~R, T)

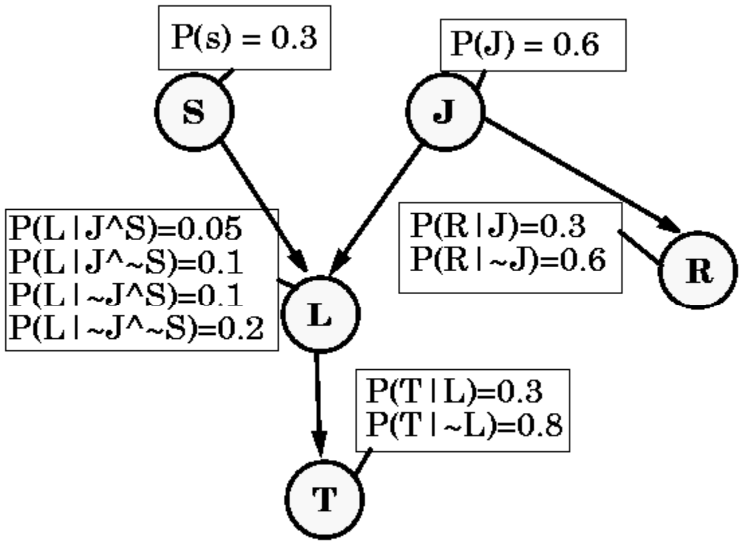
Pr(~J, L) = Pr(S, ~J, L, R, T) + Pr(S, ~J, L, R, ~T) +
Pr(~S, ~J, L, R, T) + Pr(~S, ~J, L, R, ~T) +
Pr(S, ~J, L, ~R, T) + Pr(S, ~J, L, ~R, ~T) +
Pr(~S, ~J, L, ~R, T) + Pr(~S, ~J, L, ~R, ~T)

$$\Pr(X \mid Y) = \frac{\sum_{\text{joint entries matching X and Y}} \Pr(\text{joint entry})}{\sum_{\text{joint entries matching Y}} \Pr(\text{joint entry})}$$

You have M binary variables in your Bayes Net, and expression Y uses k variables. How many rows of the joint do you have to calculate?

Bayes Net inference

Question 2: How to compute arbitrary conditional probabilities?



Compute Pr(S, T | ~J, L)

Express the conditional probability as a ratio:

Pr(S, T | ~J, L) = Pr(S, ~J, L, T) / Pr (~J, L)

Good news: we can sometimes simplify the probability calculations.

Pr(S, ~J, L, T) = Pr(S) Pr(~J) Pr(L | S, ~J) Pr(T | L)

Pr(~J, L) = Pr(~J) Pr(L | ~J)

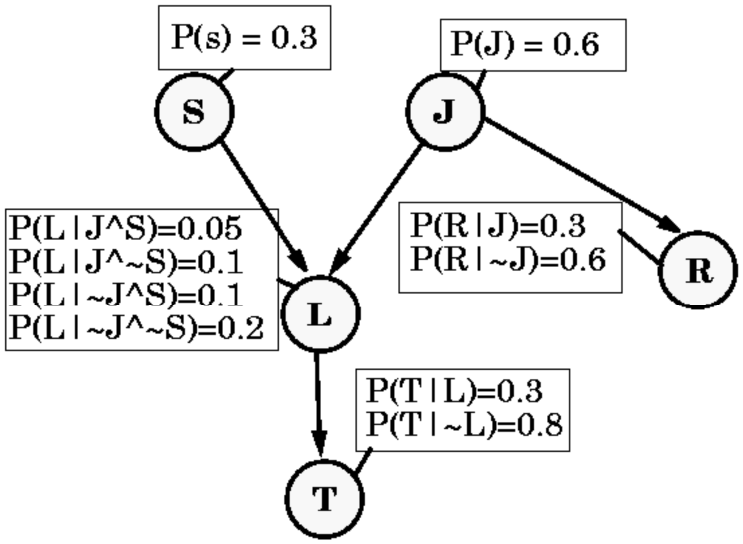
Pr(L | ~J) = Pr(L | ~J, S) Pr(S) + Pr(L | ~J, ~S) Pr(~S)

$$\Pr(X \mid Y) = \frac{\sum_{\text{joint entries matching X and Y}} \Pr(\text{joint entry})}{\sum_{\text{joint entries matching Y}} \Pr(\text{joint entry})}$$

You have m binary variables in your Bayes Net, and expression Y uses k variables. How many rows of the joint do you have to calculate?

Bayes Net inference

Question 2: How to compute arbitrary conditional probabilities?



Express the conditional probability as a ratio:

Pr(S, T | ~J, L) = Pr(S, ~J, L, T) / Pr (~J, L)

Good news: we can sometimes simplify the probability calculations.

Pr(S, ~J, L, T) = Pr(S) Pr(~J) Pr(L | S, ~J) Pr(T | L)

Pr(~J, L) = Pr(~J) Pr(L | ~J)
Pr(L | ~J) = Pr(L | ~J, S) Pr(S) + Pr(L | ~J, ~S) Pr(~S)

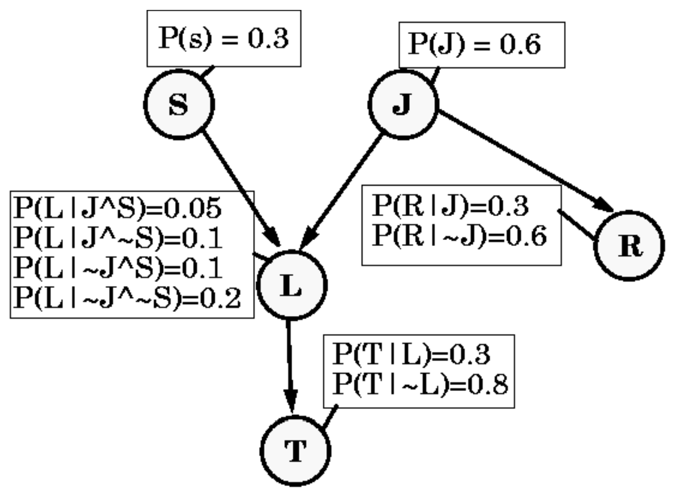
<u>Bad news</u>: doing exact inference for Bayes Nets is computationally hard.

$$\Pr(X \mid Y) = \frac{\sum_{\text{joint entries matching X and Y}} \Pr(\text{joint entry})}{\sum_{\text{joint entries matching Y}} \Pr(\text{joint entry})}$$

But it's tractable in some special cases (e.g. trees). We can also do efficient **approximate** inference.

You have m binary variables in your Bayes Net, and expression Y uses k variables. How many rows of the joint do you have to calculate?

Approximate inference



Compute Pr(S, T | ~J, L)

<u>To sample from the joint distribution of S, J, L, R, T</u>

1. Randomly choose S (True with probability 0.3)
2. Randomly choose J (True with probability 0.6)
3. Randomly choose L. The probability that L is true depends on the assignments of S and J. If steps 1 and 2 produced S = True, J = False, then probability that L is true is 0.1.
4. Randomly choose R (Probability depends on J)
5. Randomly choose T (Probability depends on L)

<u>To estimate any conditional probability Pr(X | Y)</u>

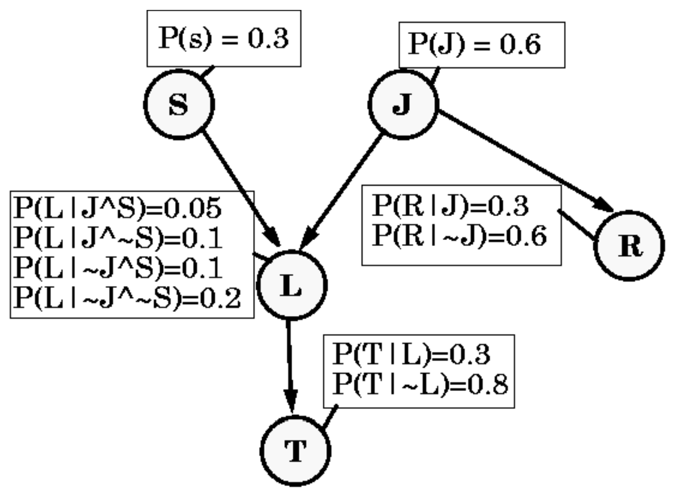
1. Draw N samples from the joint distribution
2. Count N_Y = number of samples where Y is true
3. Count N_{XY} = number of samples where both X and Y are true.
4. Calculate $Pr(X \mid Y) = N_{XY} / N_Y$

<u>Problem</u>: many of these N samples are wasted because Y is false.

<u>Solution</u>: only generate samples where Y is true, but weight them so that this property still holds. $\longrightarrow$

For large N, the ratio of N_{XY} to N_Y converges to the true probability Pr(X | Y).

Likelihood weighted sampling



Compute Pr(S, T | ~J, L)

<u>Choosing a sample from the joint, subject to ~J, L</u>:

0. Set initial weight w = 1.
1. Randomly choose S (True with probability 0.3)
2. Multiply w by Pr(J = False) = 0.4. Set J = False.
3. Multiply w by Pr(L = True) given the current assignments of S and J. For example, if steps 1-2 produced S = True and J = False then multiply w by 0.1. Set L = True.
4. Randomly choose R (True with probability 0.6)
5. Randomly choose T (True with probability 0.3)

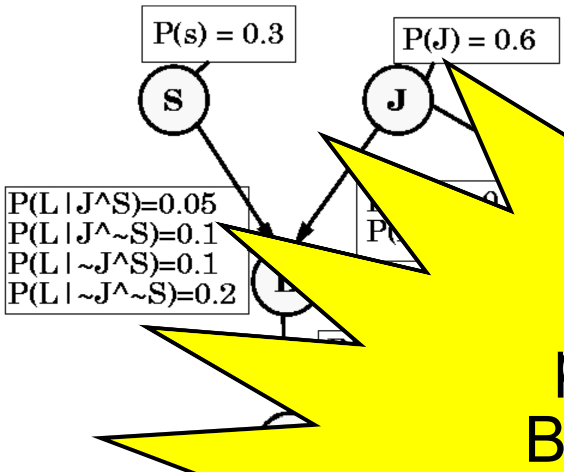
<u>Problem</u>: many of these N samples are wasted because Y is false.

<u>Solution</u>: only generate samples where Y is true, but weight them so that this property still holds.

<u>To estimate any conditional probability Pr(X | Y)</u>

1. Draw N samples from the joint distribution, subject to the constraint Y.
2. For each sample and its weight $w \leq 1$:
   Increment N_Y by w.
   If X is true, increment N_{XY} by w.
3. Calculate $\Pr(X \mid Y) = N_{XY} / N_Y$

Likelihood weighted sampling



C oosin a sa ple fro n the joint, subject to ~J, L:

0 et al ght = 1.

S e with probability 0.3)

.4. Set J = False.

e current

xample if steps

e then

pends on J)

on L)

Y | Y)

ribution,

≤ 1:

Y by

3 = N$_{XY}$ / N$_Y$

Co

Now we know how to perform inference with a Bayes Net. This is great if we already have the network structure and parameters specified by an expert… but what if we want to **learn** the Bayes Net from data?

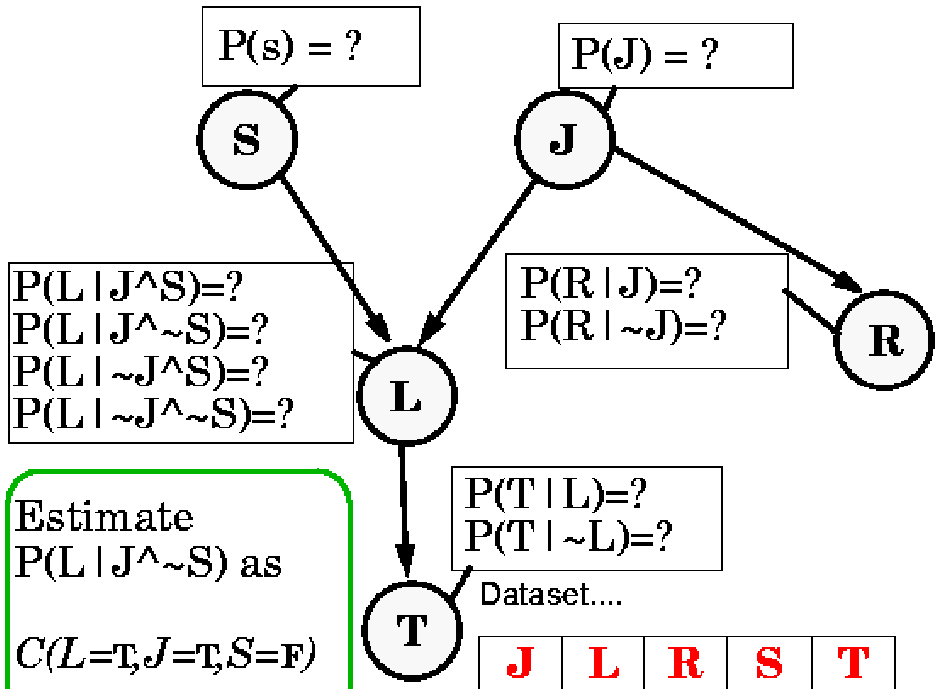
Solution: only gene samples where Y is true, but weight them so that this property still holds.

Bayes Net parameter learning

Given a Bayesian network structure and a training dataset, we can learn the parameters of each node by maximum likelihood.

Given node X with parent nodes $Q_1..Q_m$, we learn the conditional distribution of X for each distinct combination of parent values.

For example, if $Q_1..Q_m$ are all binary, there are 2^m distributions to learn.



Estimate P(L | J^~S) as

$$\frac{C(L=T, J=T, S=F)}{C(J=T, S=F)}$$

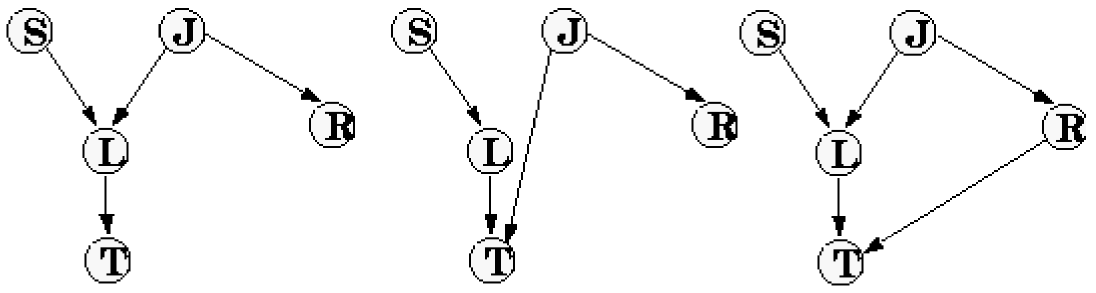
Dataset....

J	L	R	S	T
True	False	False	True	False
True	True	False	True	True
True	False	False	True	False
False	True	True	False	False
True	True	False	True	True
:	:	:	:	:

We learn the distribution $\Pr(X \mid Q_1 = v_1, Q_2 = v_2, \ldots, Q_m = v_m)$ by looking at the subset of training records with the given parent values and computing the proportion with each X value.

Bayes Net structure search

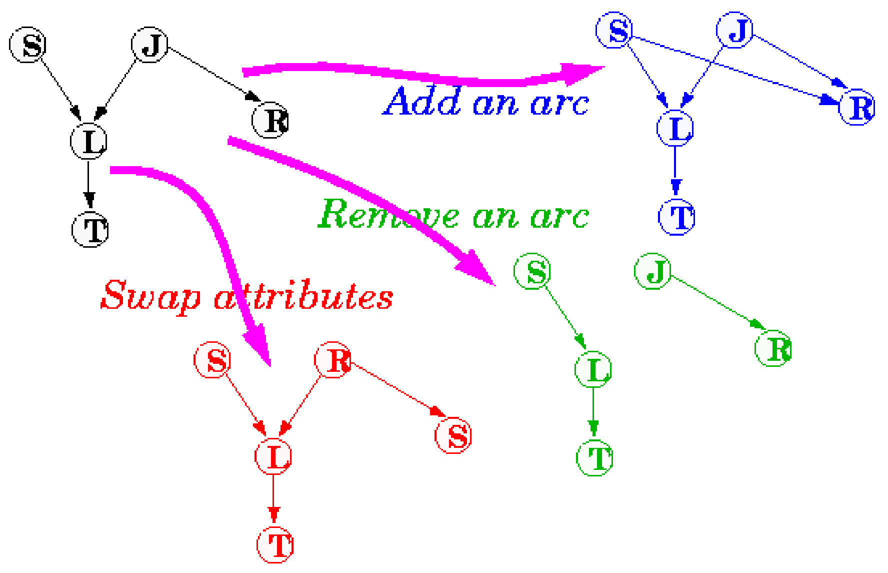
How to automatically find the Bayesian network structure that best fits the data?



This is a hard **state-space search** problem. Use simulated annealing with multiple restarts.

What moveset to use? How to score a structure?

Here's one possible moveset:



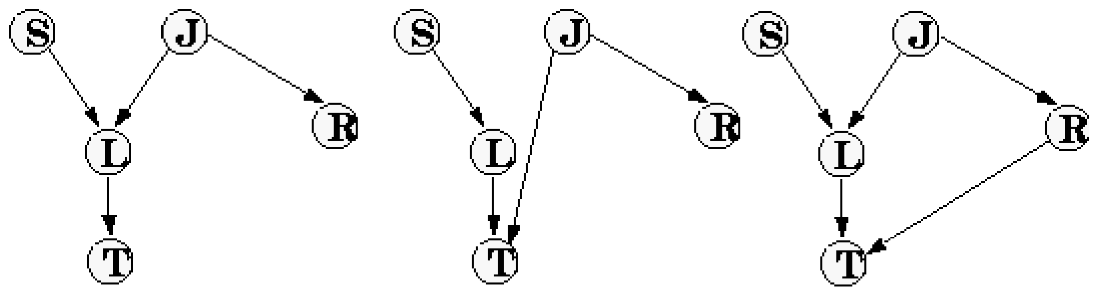
To score a structure, learn all parameters from the training data by maximum likelihood.

Then compute the log-likelihood of the training dataset given the structure and parameters.

Score = log-likelihood – λk, where λ is a constant and k is the total number of parameters.

Bayes Net structure search

How to automatically find the Bayesian network structure that best fits the data?



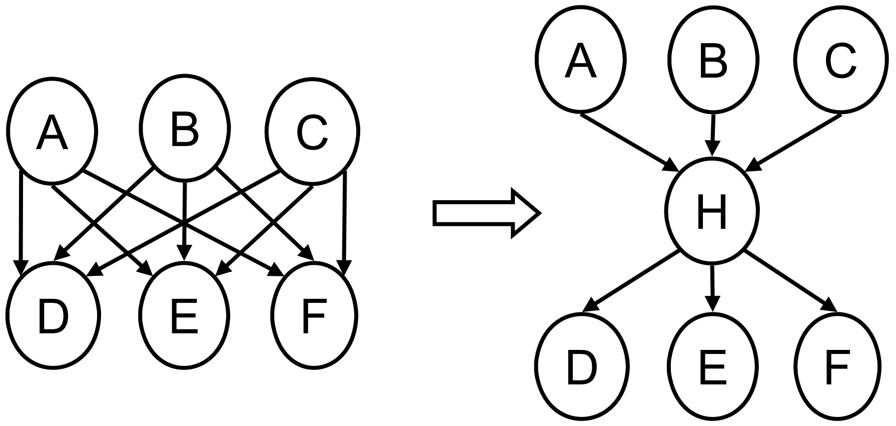
This is a hard **state-space search** problem. Use simulated annealing with multiple restarts.

What moveset to use? How to score a structure?

Some fancy structure learning systems also infer the presence of hidden nodes representing unobserved variables.

This can help to simplify the Bayes Net structure by postulating an unobserved common cause which connects multiple observations.

However, both parameter learning and structure search become much more difficult when hidden nodes are allowed.



The many uses of Bayes Nets

Bayes Nets provide a useful graphical representation of the probabilistic relationships between many variables.

Automatic learning of Bayes net structure can be used for exploratory analysis of datasets with many attributes.

We can often improve the performance of model-based classification by moving from Naïve Bayes to Bayes Nets.

We can also use Bayes Nets to detect **anomalies**, by finding points with low probabilities given the Bayes Net.

Bayes Nets provide a compact structure which enables us to efficiently compute probability distributions for any unobserved variables given observations of other variables.

Medical diagnosis Failure troubleshooting Drug discovery

Environmental modeling Computational biology

References

- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
- S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, Ch. 15.
- Several excellent tutorials on Bayes Nets available at http://www.autonlab.org/tutorials.
- Bayesian networks in Weka: tutorial at http://www.cs.waikato.ac.nz/~remco/weka.bn.pdf
- Auton software for efficient Bayes Net Learning: http://www.autonlab.org/autonweb/10530