Inference in Bayesian Networks

Inference in Bayesian networks

- Given a Bayesian network B (i.e., DAG and CPTs) , calculate P(**X**|**e**) where **X** is a set of query variables and **e** is an instantiaton of observed variables **E** (**X** and **E** separate).
- There is always the way through marginals:
  - normalize $P(\mathbf{x},\mathbf{e}) = \Sigma_{\mathbf{y} \in dom(\mathbf{Y})} P(\mathbf{x},\mathbf{y},\mathbf{e})$, where dom(**Y**), is a set of all possible instantiations of the unobserved non-query variables **Y**.
- There are much smarter algorithms too, but in general the problem is NP hard (more later).

How to generate random vectors from a Bayesian network

	Cloudy=no	Cloudy=yes
	0.5	0.5

Cloudy	Sprinkler=on	Sprinkler=off
no	0.5	0.5
yes	0.9	0.1

Cloudy	Rain=yes	Rain=no
no	0.2	0.8
yes	0.8	0.2

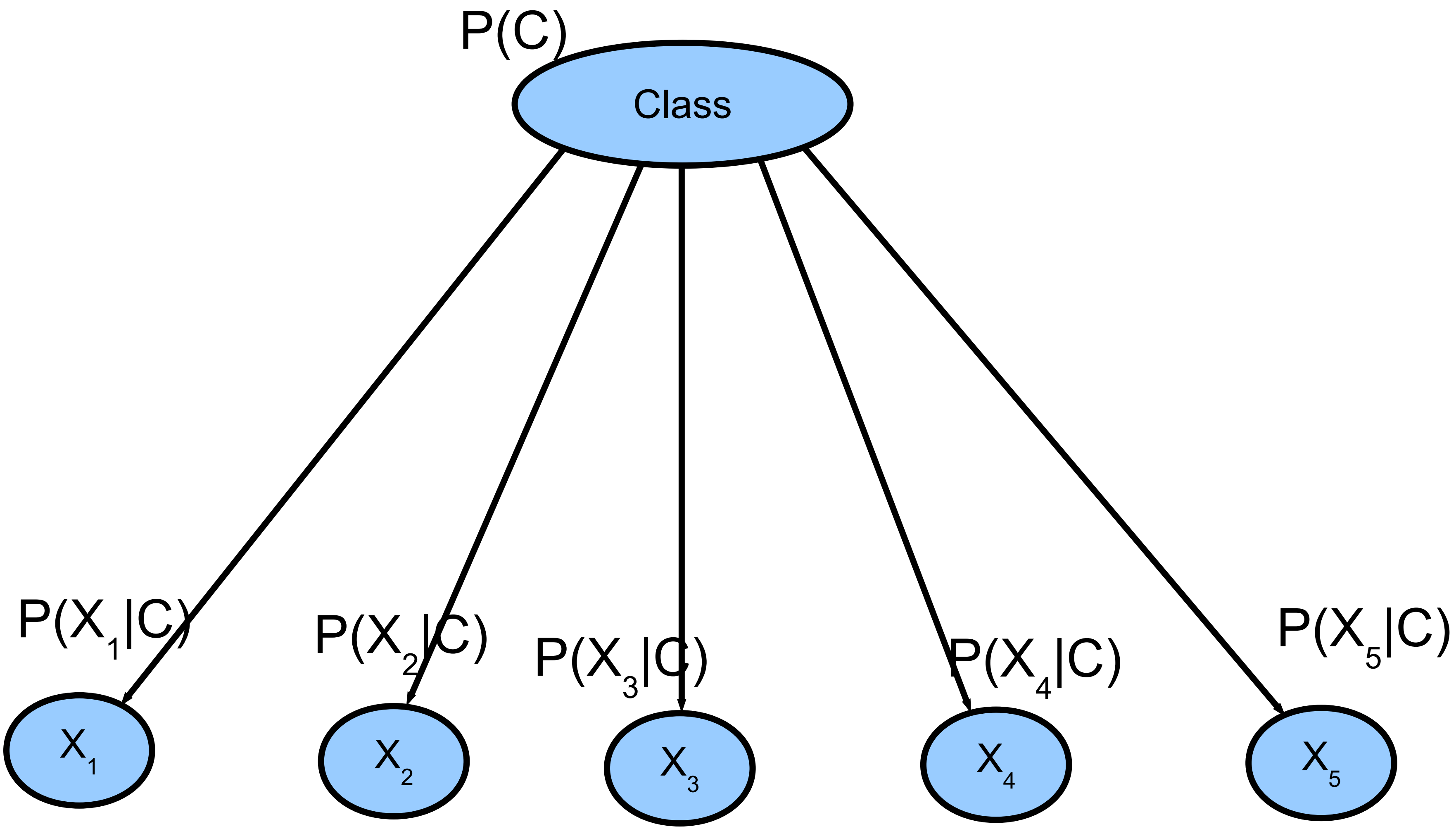
Sprinkler	Rain	WetGrass=yes	WetGrass=no
on	no	0.90	0.10
on	yes	0.99	0.01
off	no	0.01	0.99
off	yes	0.90	0.10

- Sample parents first
  - P(C)
    - (0.5, 0,5) → yes
  - P(S|C=yes)
    - (0.9, 0.1) → on
  - P(R | C=yes)
    - (0.8, 0.2) → no
  - P(W | S=on, R=no)
    - (0.9, 0.1) → yes
- P(C,S,R,W) = P(yes,on,no,yes) = 0.5 x 0.9 x 0.2 x 0.9 = 0.081

Some famous (simple) Bayesian network models

- Naïve Bayes classifier
- Finite mixture model
- Tree Augmented Naïve Bayes
- Hidden Markov Models (HMMs)

Naïve Bayes classifier



- X_i are called predictors or indicators

Naïve Bayes Classifier

- Structure tailored for efficient diagnostics $P(C|x_1,x_2,...,x_n)$.
- Unrealistic conditional independence assumptions, but OK for the particular query $P(C|x_1,x_2,...,x_n)$.
- Because of wrong independence assumptions, NB is often poorly calibrated:
  - Probabilities $P(C|x_1,x_2,...,x_n)$ way off, but $\operatorname{argmax}_c P(c|x_1,x_2,...,x_n)$ still often correct.

Calculating $P(C|x_1,x_2,\ldots,x_n,NB)$

- Boldly calculate through joint probability

$$P(C|x_1,\ldots,x_n) \propto P(C,x_1,\ldots,x_n) = P(C)\prod_{i=1}^{n} P(x_i|C)$$

- No need to have all the predictors. Having just set X_A of predictors (and not X_B):

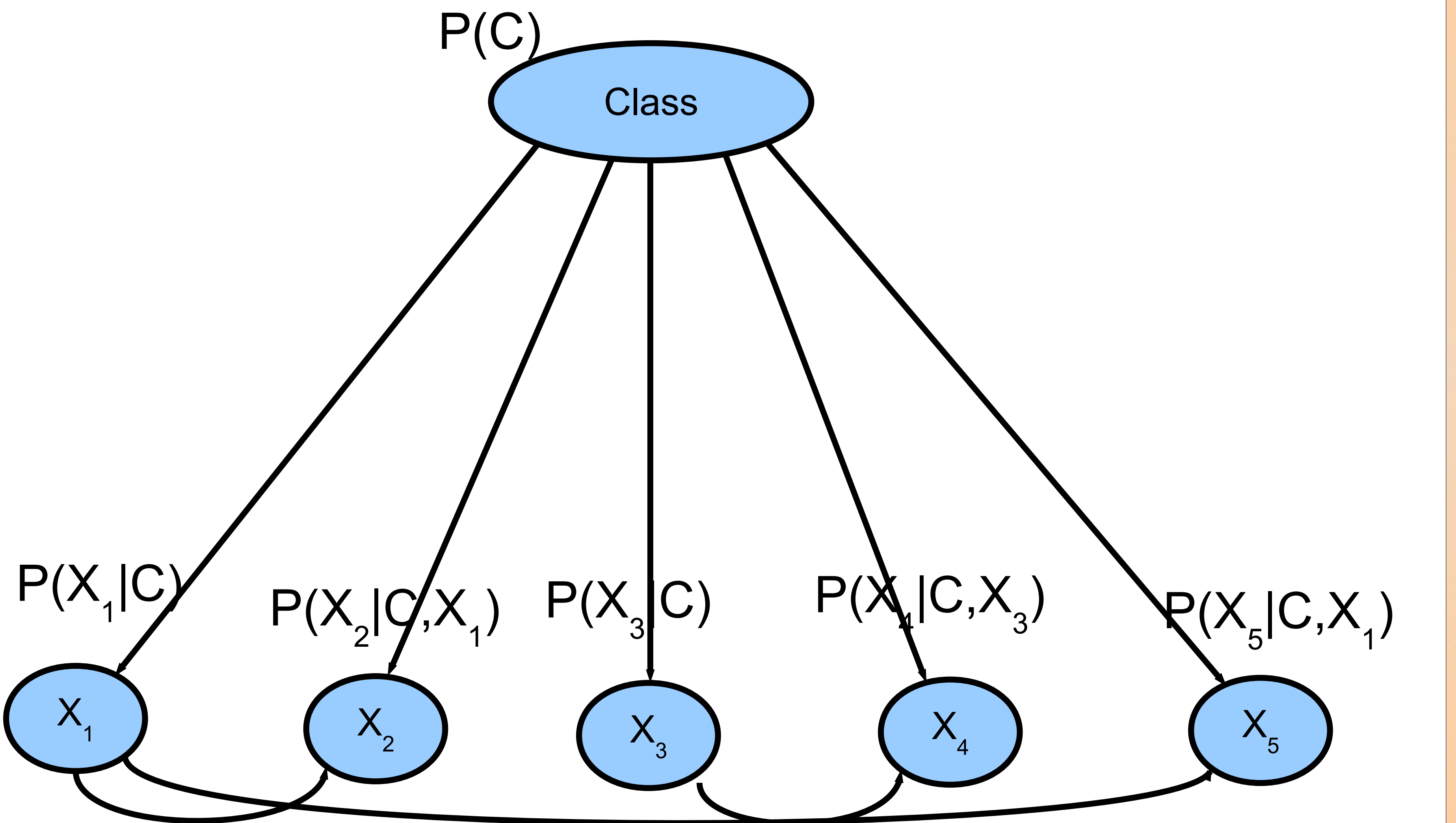
$$P(C|x_A) \propto P(C,x_A) = \sum_{x_B} P(C,x_A,x_B)$$

$$= \sum_{x_B} P(C)\prod_{i\in A} P(x_i|C)\prod_{j\in B} P(x_j|C)$$

$$= P(C)\prod_{i\in A} P(x_i|C)\sum_{x_B}\prod_{j\in B} P(x_j|C)$$

$$= P(C)\prod_{i\in A} P(x_i|C)\prod_{j\in B}\sum_{x_j} P(x_j|C) = P(C)\prod_{i\in A} P(x_i|C)$$

Tree Augmented Naïve Bayes (TAN)



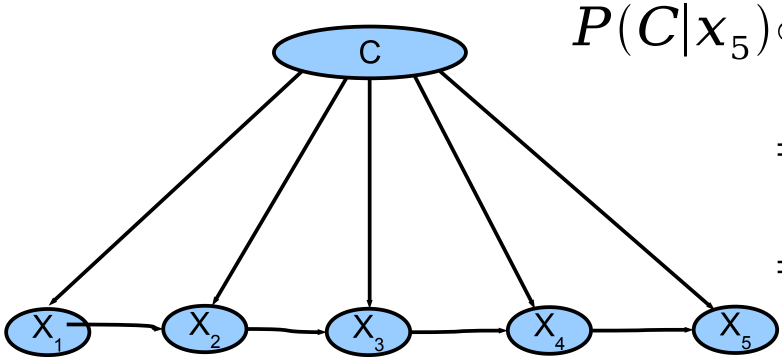
- X_i may have at most one other X_j as an extra parent.

Calculating $P(C|x_1,x_2,\ldots,x_n,TAN)$

- Again, boldly calculate via joint probability

$$P(C|x_1,\ldots,x_n) \propto P(C,x_1,\ldots,x_n) = P(C)\prod_{i=1}^{n} P(x_i|C,Pa(x_i))$$

- But missing predictors may hurt more. For example:



$$\begin{aligned}P(C|x_5) &\propto P(C)P(x_5|C) = P(C)\sum_{x_4} P(x_4|C)P(x_5|x_4,C)\\ &= P(C)\sum_{x_4} P(x_5|C,x_4)P(x_4|C)\\ &= P(C)\sum_{x_4} P(x_5|C,x_4)\sum_{x_3} P(x_4|C,x_3)P(x_3|C)\\ &= \ldots\end{aligned}$$

NB as a Finite Mixture Model

- When NB structure is right, it also makes a nice (marginal) joint probability model $P(X_1,X_2,\ldots,X_n)$ for "predictors".
- A computationally effective alternative for building a Bayesian network for $X_1,X_2,\ldots,X_n$.
- Joint probability $P(X_1,X_2,\ldots,X_n)$ is represented as a mixture of K joint probability distributions $P_k(X_1,X_2,\ldots,X_n) = P_k(X_1)P_k(X_2)\ldots P_k(X_n)$, where $P_k(\cdot) = P(\cdot|C=k)$.

Calculating with $P(X_1,X_2,\ldots,X_n|NB)$

- Joint probability a simple marginalization:

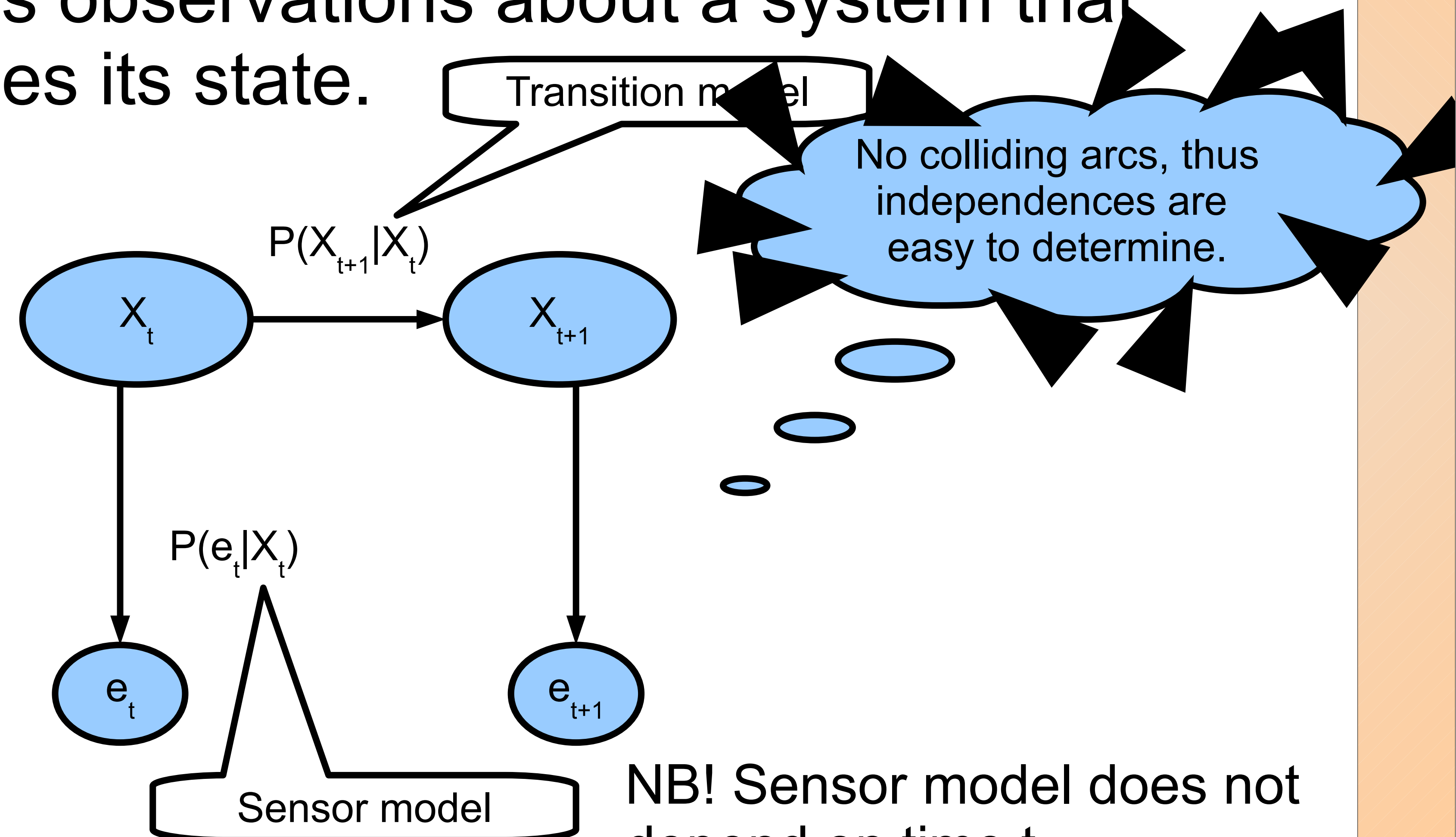
$$P(X_1,\ldots,X_n)=\sum_{k=1}^{K} P(X_1,\ldots,X_n,C=k)$$
$$=\sum_{k=1}^{K} P(C=k)\prod_{i=1}^{n} P(X_i|C=k)$$

- Inference

$$P(X|e)\propto P(e,X)=\sum_{k=1}^{K} P(e,X,C=k)$$
$$=\sum_{k=1}^{K} P(C=k)P(e,X|C=k)$$
$$=\sum_{k=1}^{K} P(C=k)\prod_{X_i\in X} P(X_i|C=k)\prod_{e_i\in e} P(e_i|C=k)$$

Hidden Markov Models

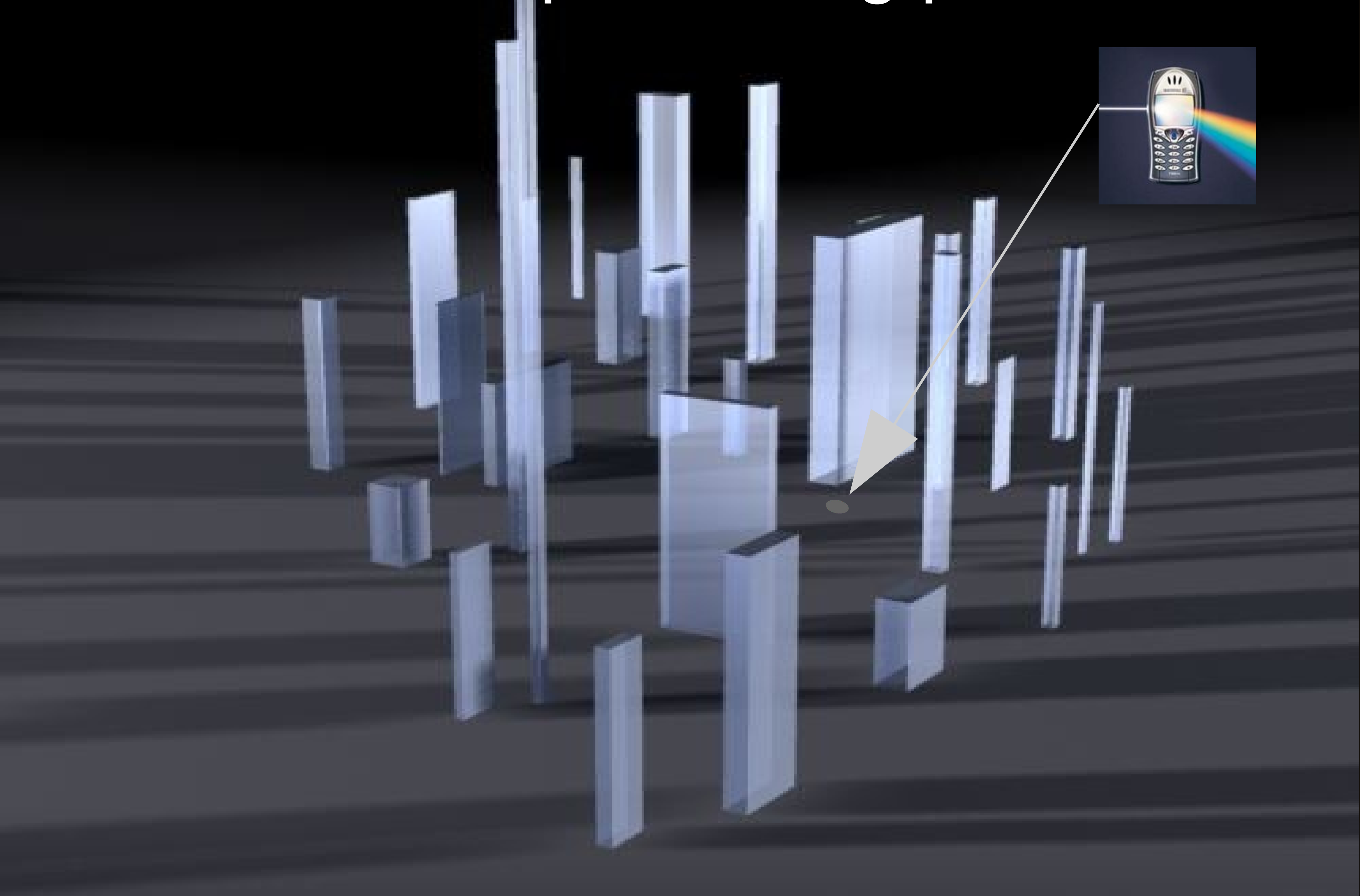
- Models observations about a system that changes its state.



NB! Sensor model does not depend on time t.

Graphical models on Manhattan – A probabilistic approach to mobile device positioning

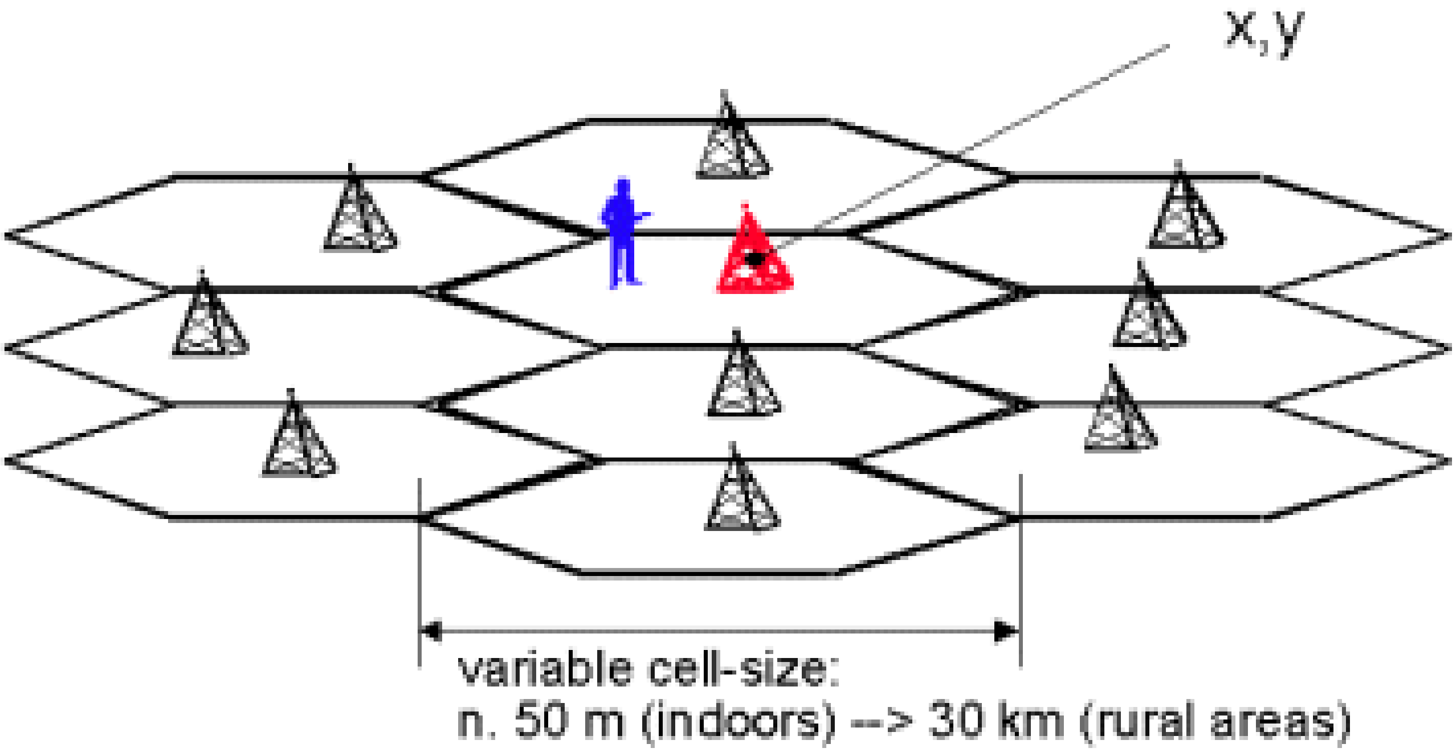
Location positioning problem



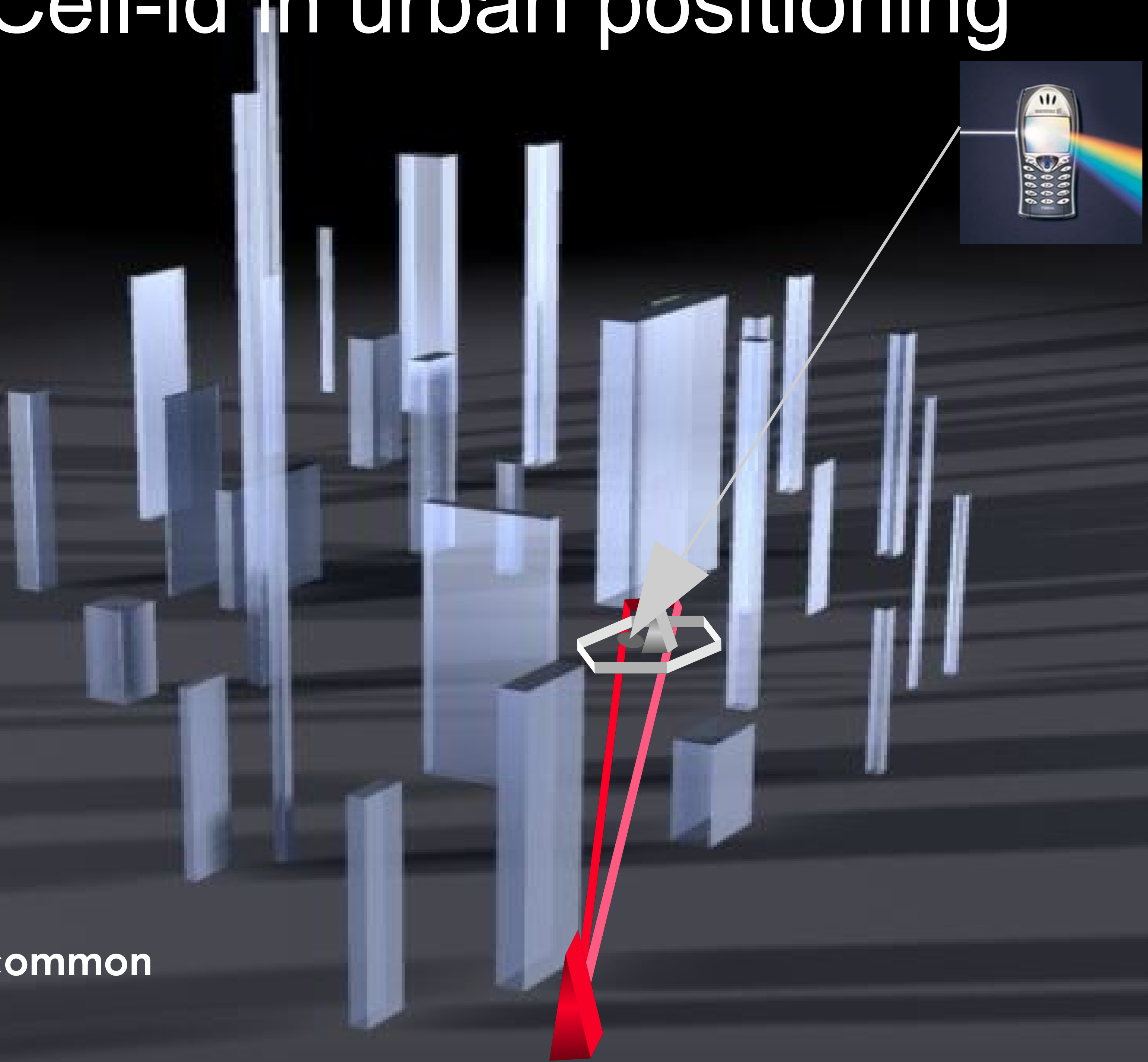
The positioning problem

- Given some location-dependent observations *O*, measured by a mobile device, determine the location *L* of the device
- Why is this a good research problem?
  - The goodness of different solutions is extremely easy to validate (just go to a known location and test)
  - The results have immediate practical applications
    - Location-based services (LBS)
    - FCC Enhanced 911:
      - Network-based solutions: error below 100 meters for 67 percent of calls, 300 meters for 95 percent of calls
      - Handset-based solutions: error below 50 meters for 67 percent of calls, 150 meters for 95 percent of calls

Cell ID

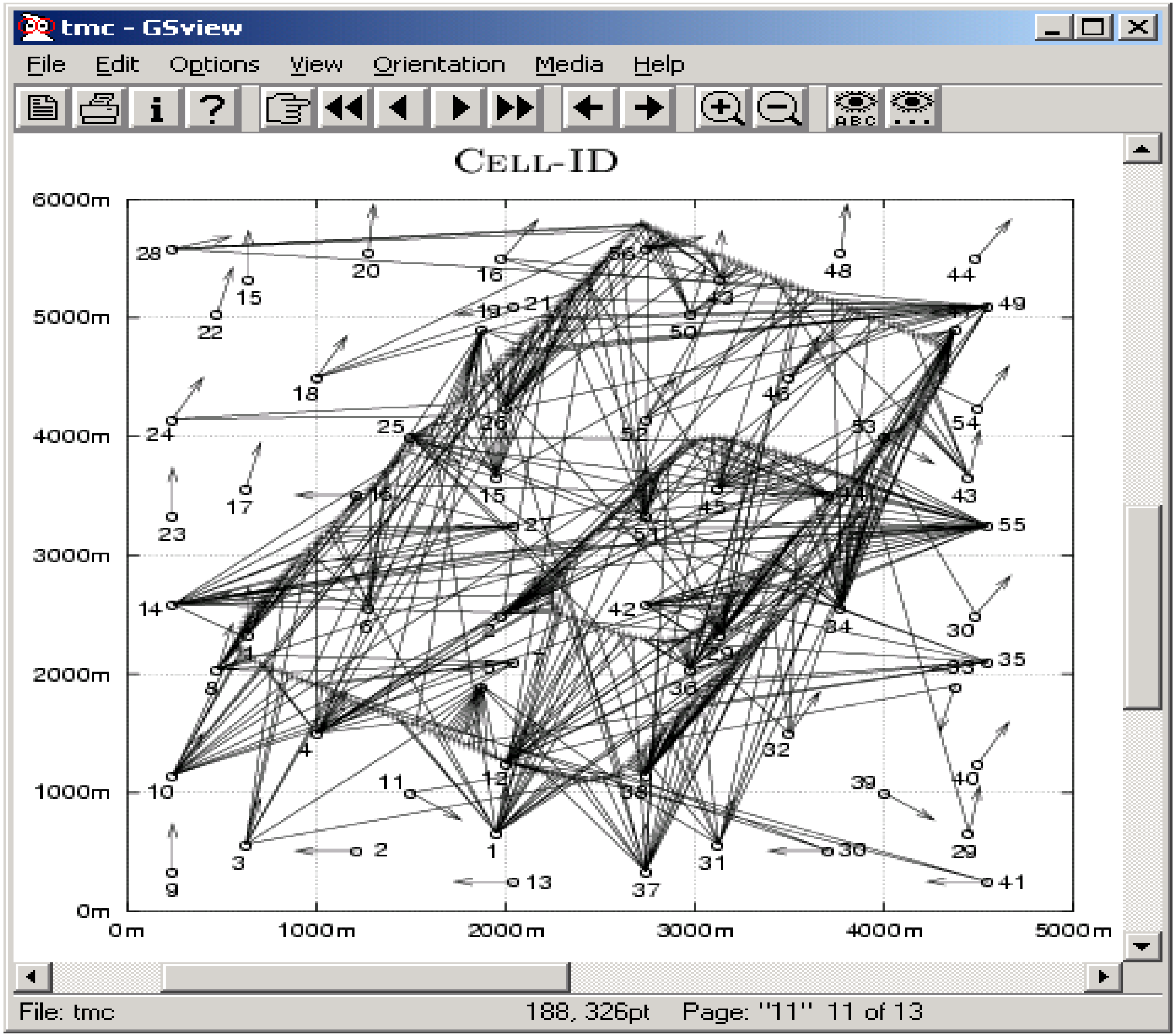


Cell-id in urban positioning

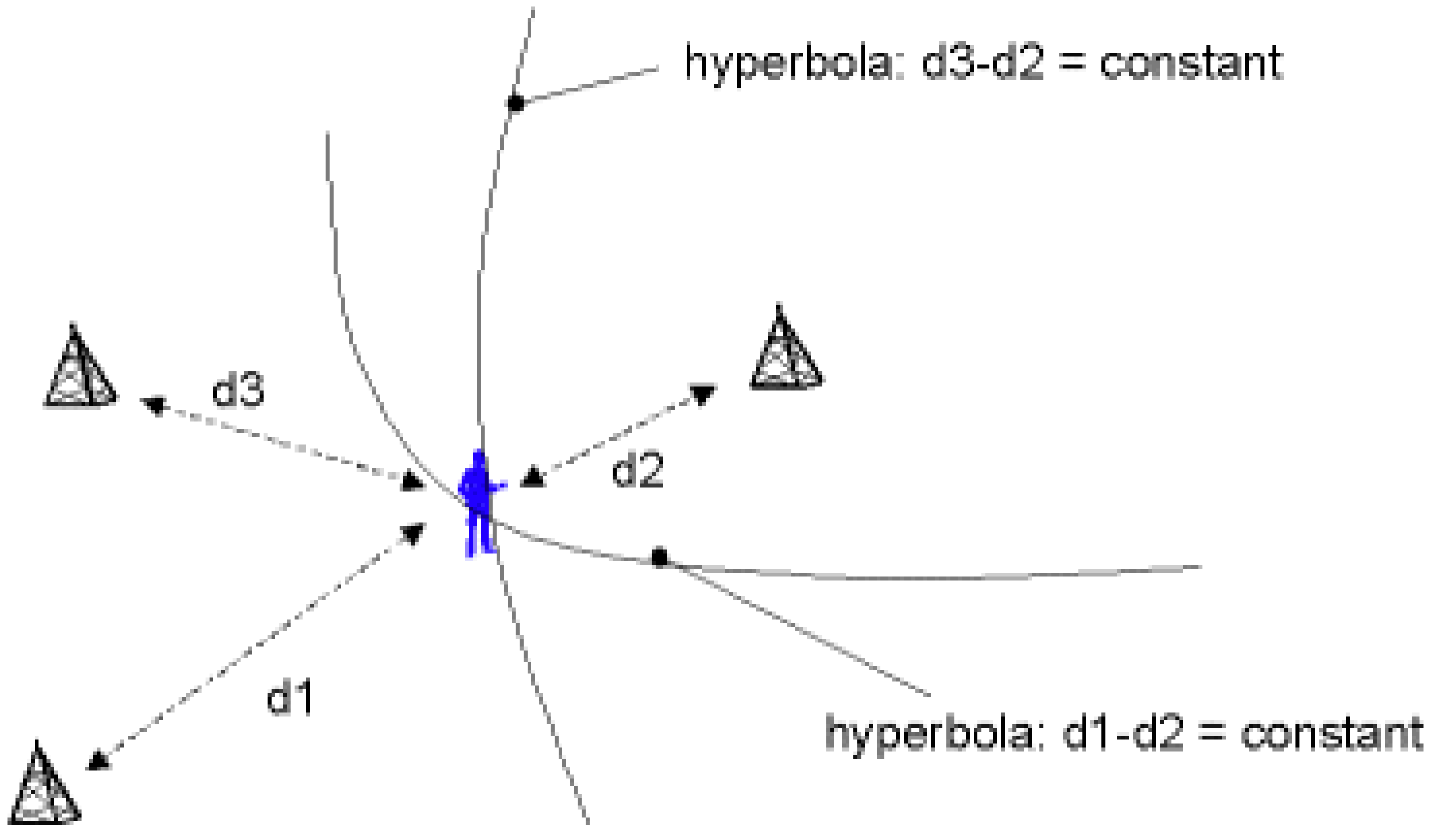


- errors > 500m common
+ simple

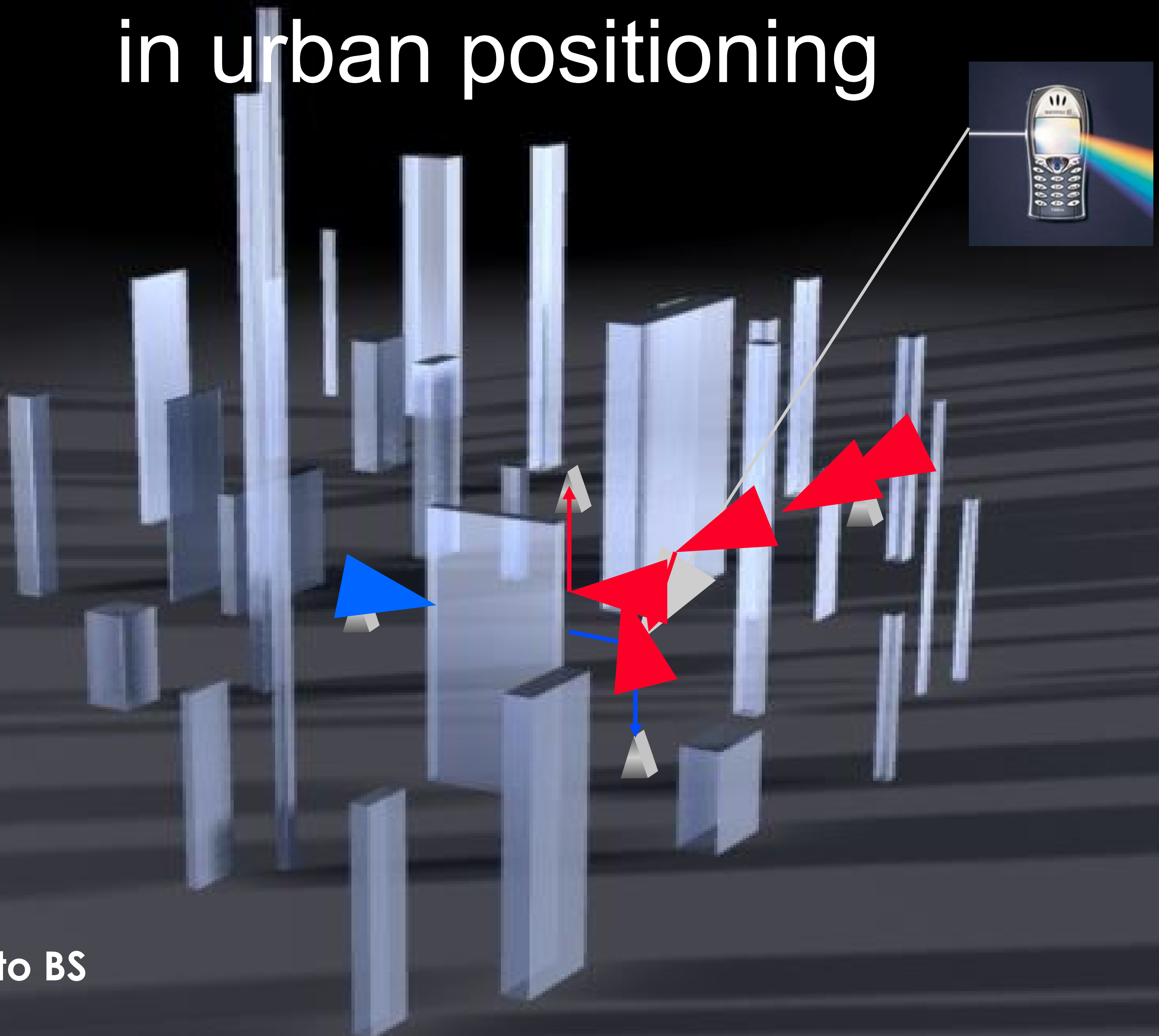
Cell ID errors



Enhanced Observed Time Difference (E-OTD)

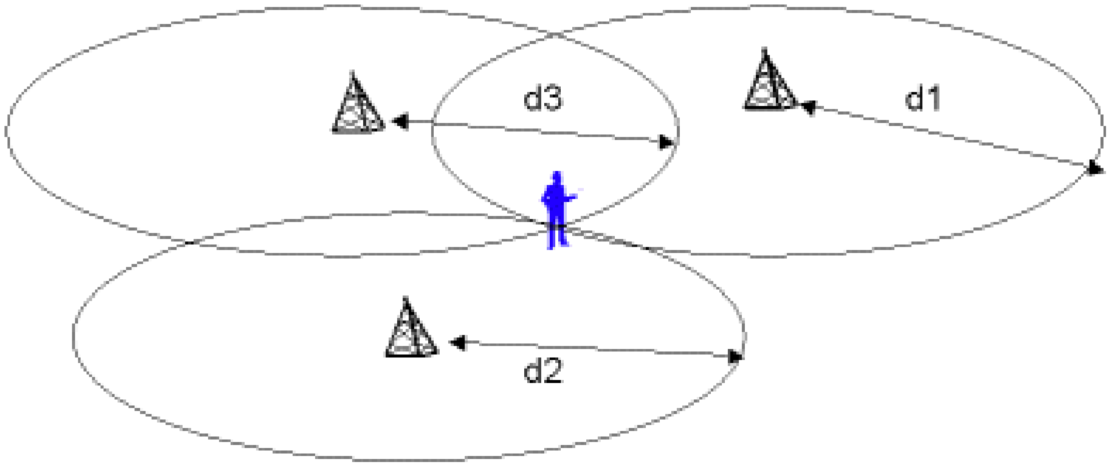


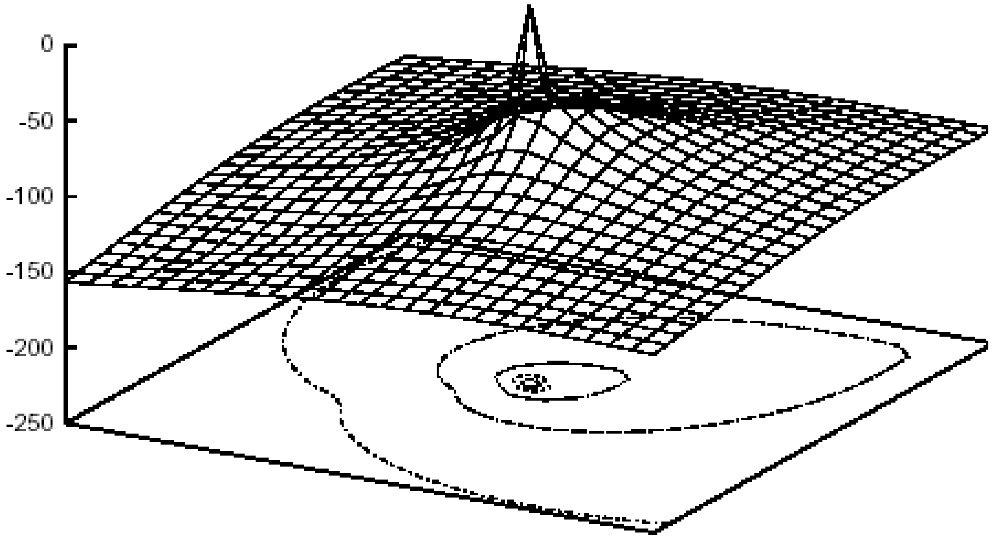
Problems with E-OTD in urban positioning



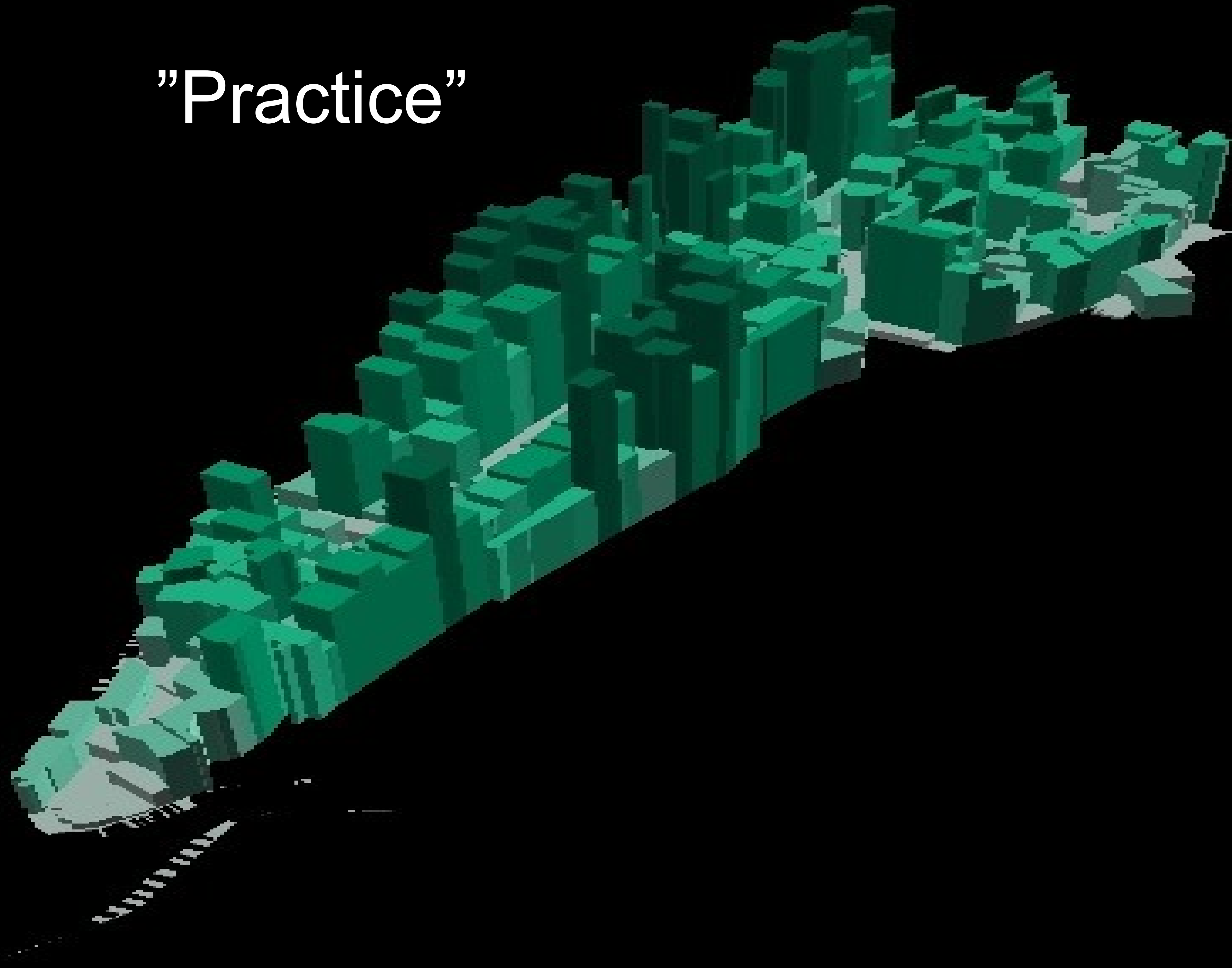
- multi-paths
- no line of sight to BS
- extra hardware

"Theory"



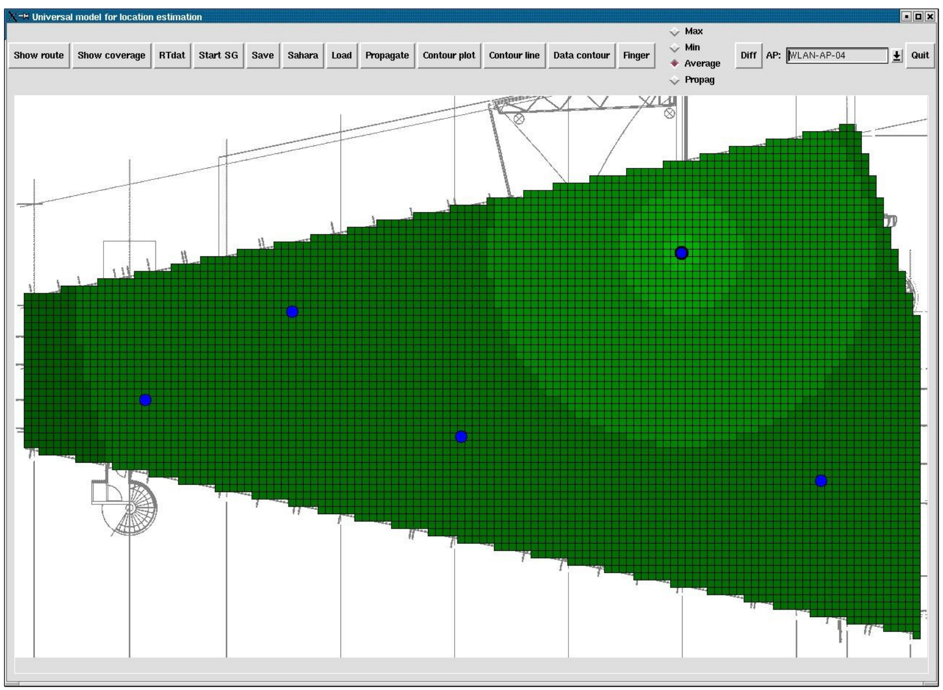


”Practice”

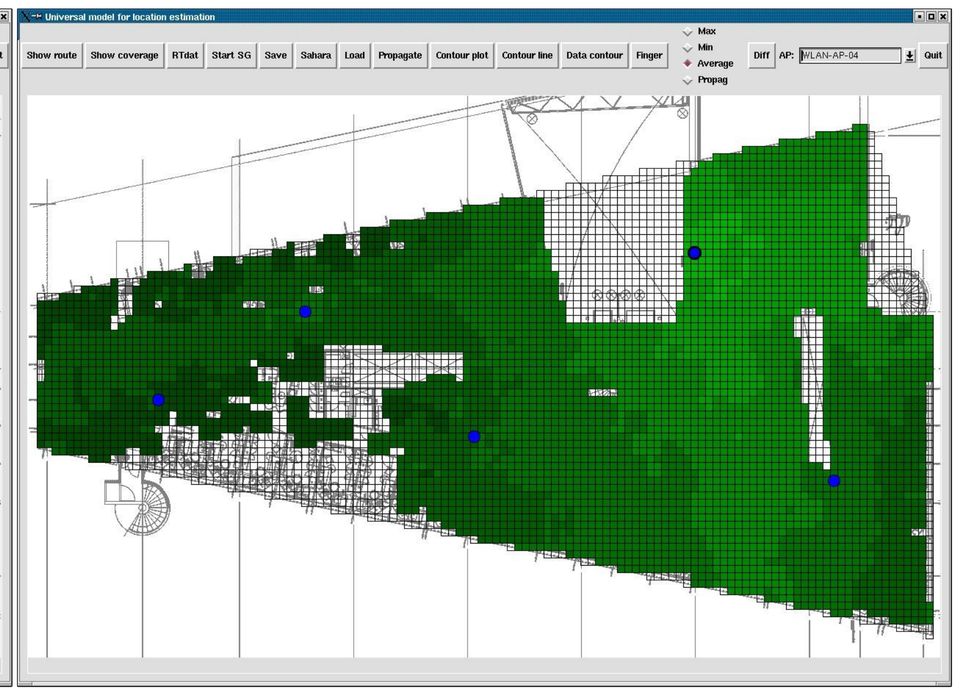


The signal propagation approach

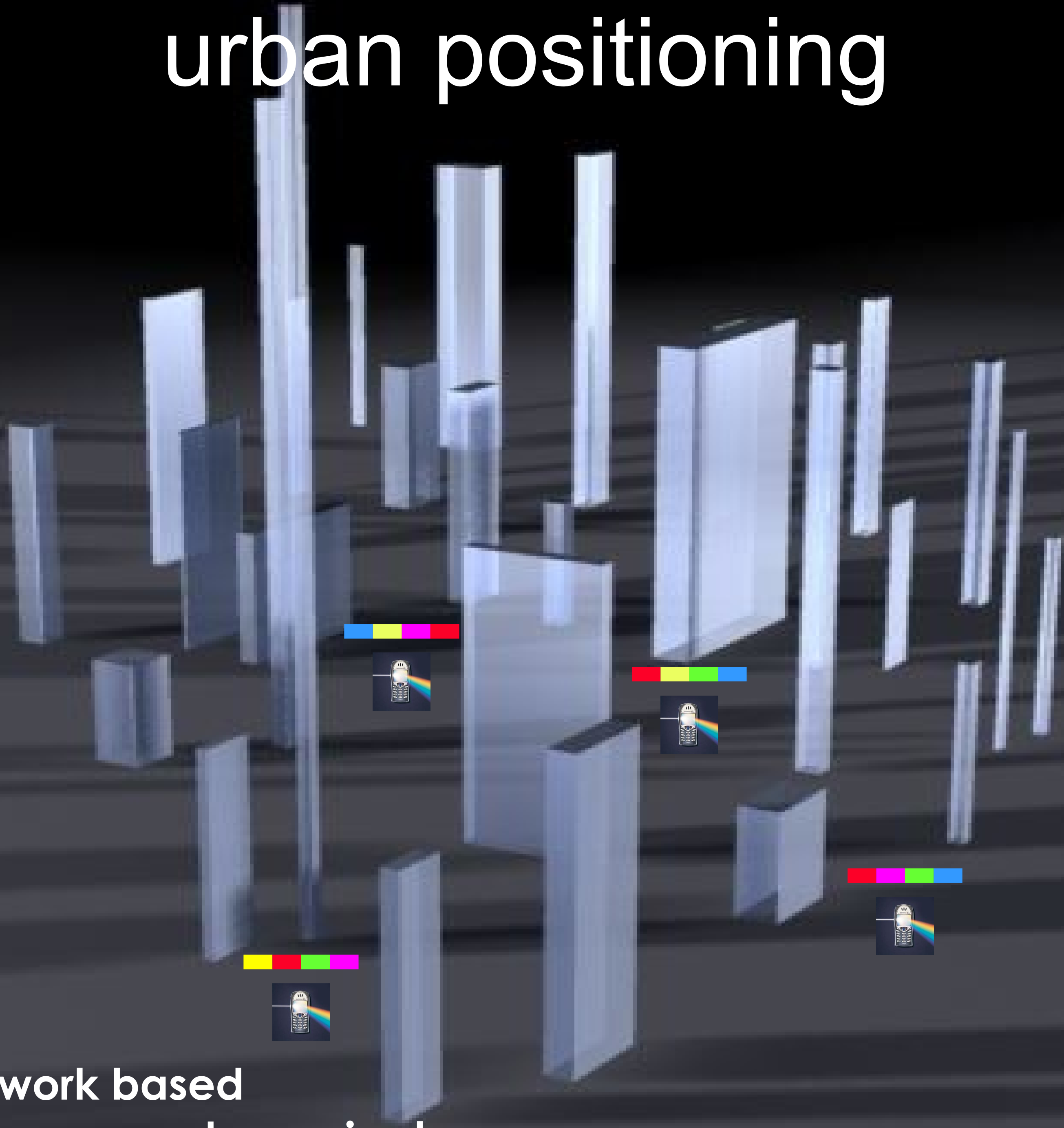
Theory



Reality



Empirical modeling in urban positioning



+accurate
+handset or network based
-calibration measurements required

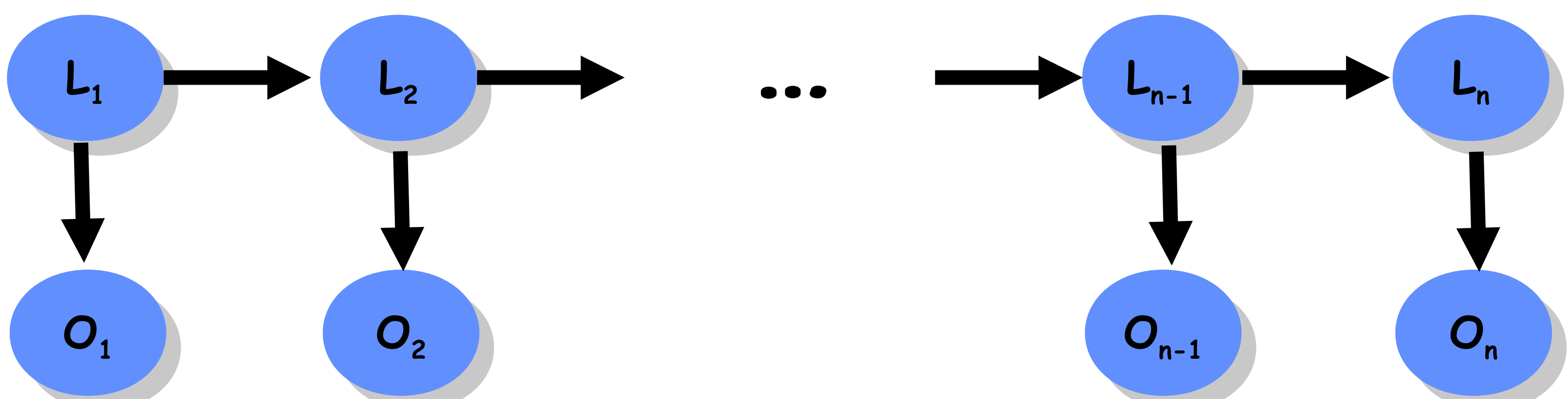
A probabilistic approach to positioning

Bayes rule: $P(L \mid O) = \dfrac{P(O \mid L)\, P(L)}{P(O)}$

- A probabilistic model assigns a probability for each possible location L given the observations O.
  - P(O | L) is the conditional probability of obtaining observations O at location L.
  - P(L) is the prior probability of location O. (Could be used to exploit user profiles, rails etc.)
  - P(O) is just a normalizing constant.
- How to obtain P(O | L)? ⇨ Empirical observations + machine learning

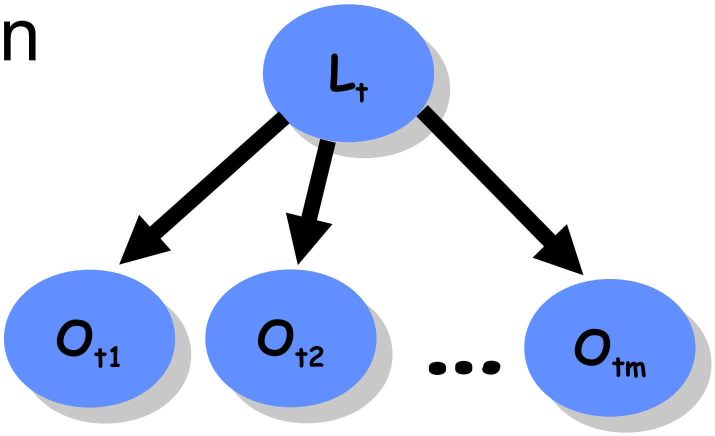
Tracking with Markov models

- Typically we have a sequence (history) of observations $O_1,\ldots,O_n$, and wish to determine $P(L_n \mid O^n)$
- Assumption: $P(O_t \mid L_t)$ are known, and given location L_t, the observation O_t is independent of the rest of the history
- The model: a hidden Markov model (HMM) where the locations L_t are the hidden unobserved states
- The transition probabilities $P(L_t \mid L_{t-1})$ can be easily determined from the physical properties of the moving object



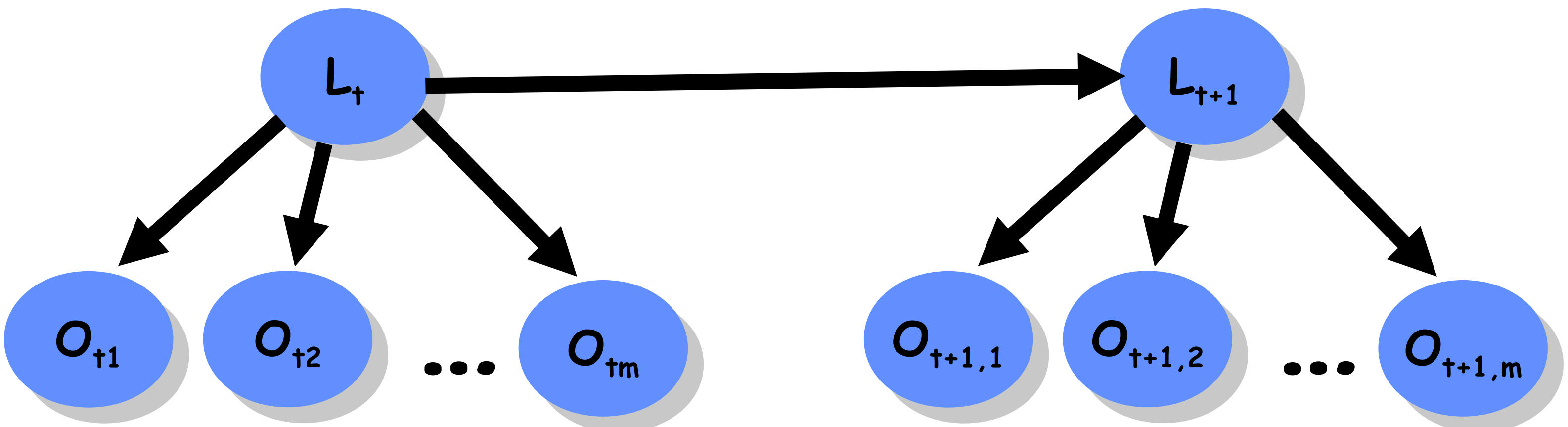
One more assumption

- The observation at time t typically consists of several measurements (e.g., strengths of signals from all the transmitters that can be heard)
- If the wireless network is designed in a reasonable manner (the transmitters are far from each other), it makes sense to assume that the individual observations are independent, given the location
- The “Naïve Bayes” model



The Model

First-order "semi-hidden" Markov model



Tracking as probabilistic inference

- As our hidden Markov model is a tree, we can compute the marginal of any L_t, given the history O^n, in linear time by using a simple forward-backward algorithm
- Alternatively, we can compute the maximum probability path $L_1,\ldots,L_n$ given the history (this is known as the **Viterbi** algorithm)
- **Kalman filter**: all the conditional distributions of the HMM model are normal distributions (linear dependencies with Gaussian noise)

Recursive tracking

- Assume that $\mathbf{P(L_{n-1} \mid O^{n-1})}$ has been computed.
- Our model defines the transition probabilities $\mathbf{P(L_t \mid L_{t-1})}$ and the local observation probabilities $\mathbf{P(O_t \mid L_t)}$
- Now $\mathbf{P(L_n \mid O^n)}$ $\boldsymbol{\alpha}$ $\mathbf{P(L_n, O^n)}$

$$= \mathbf{P(O_n \mid L_n, O^{n-1})\, P(L_n , O^{n-1})}$$

$$= \mathbf{P(O_n \mid L_n) \Sigma_{L_{n-1}} P(L_n , L_{n-1} , O^{n-1})}$$

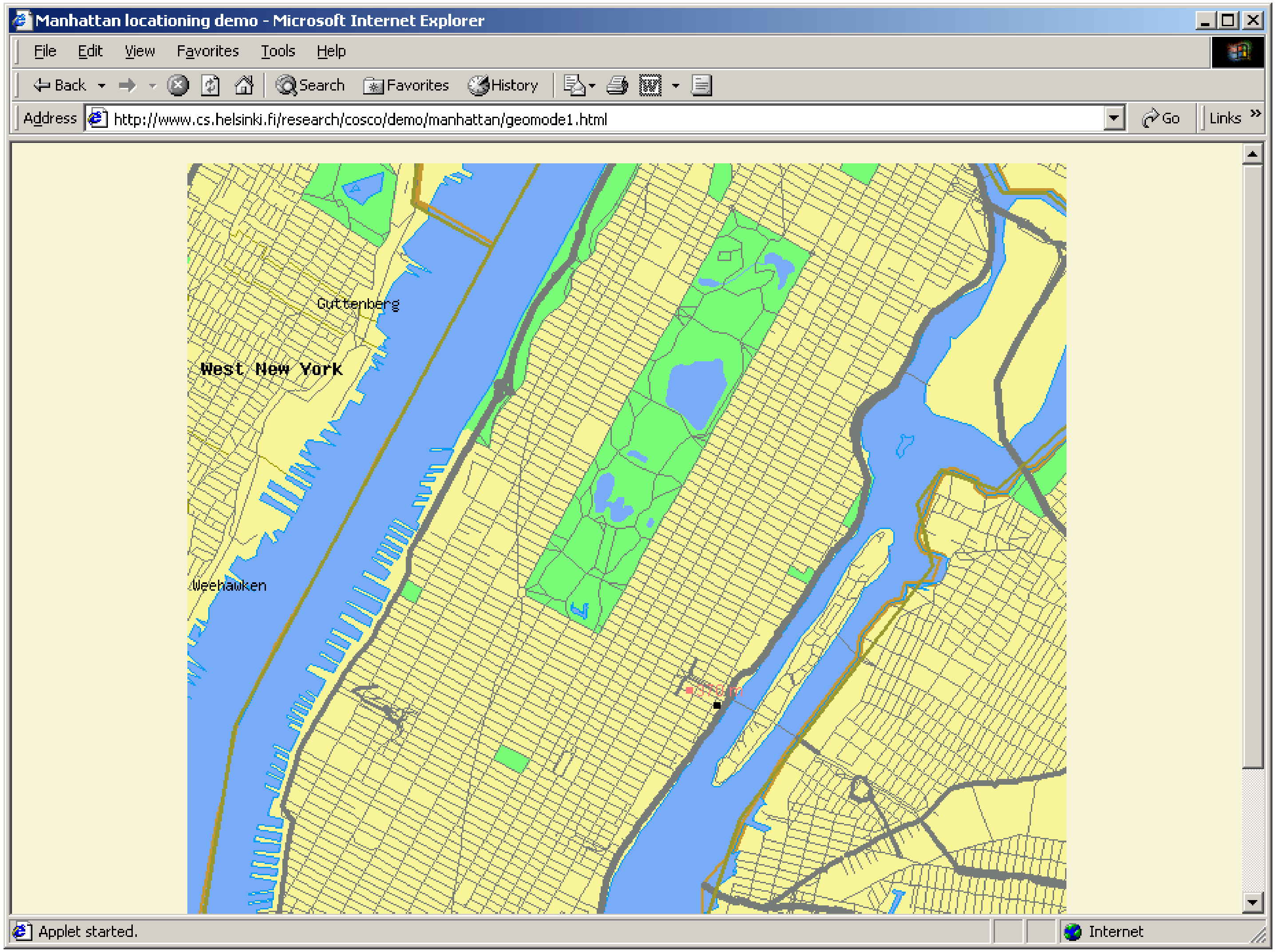
$$\boldsymbol{\alpha}\ \mathbf{P(O_n \mid L_n) \Sigma_{L_{n-1}} P(L_n \mid L_{n-1}) P(L_{n-1} \mid O^{n-1})}$$

- With a Kalman filter, the recursive process operates all the time with Gaussians

GSM-positioning trials

NYC Trial 2001

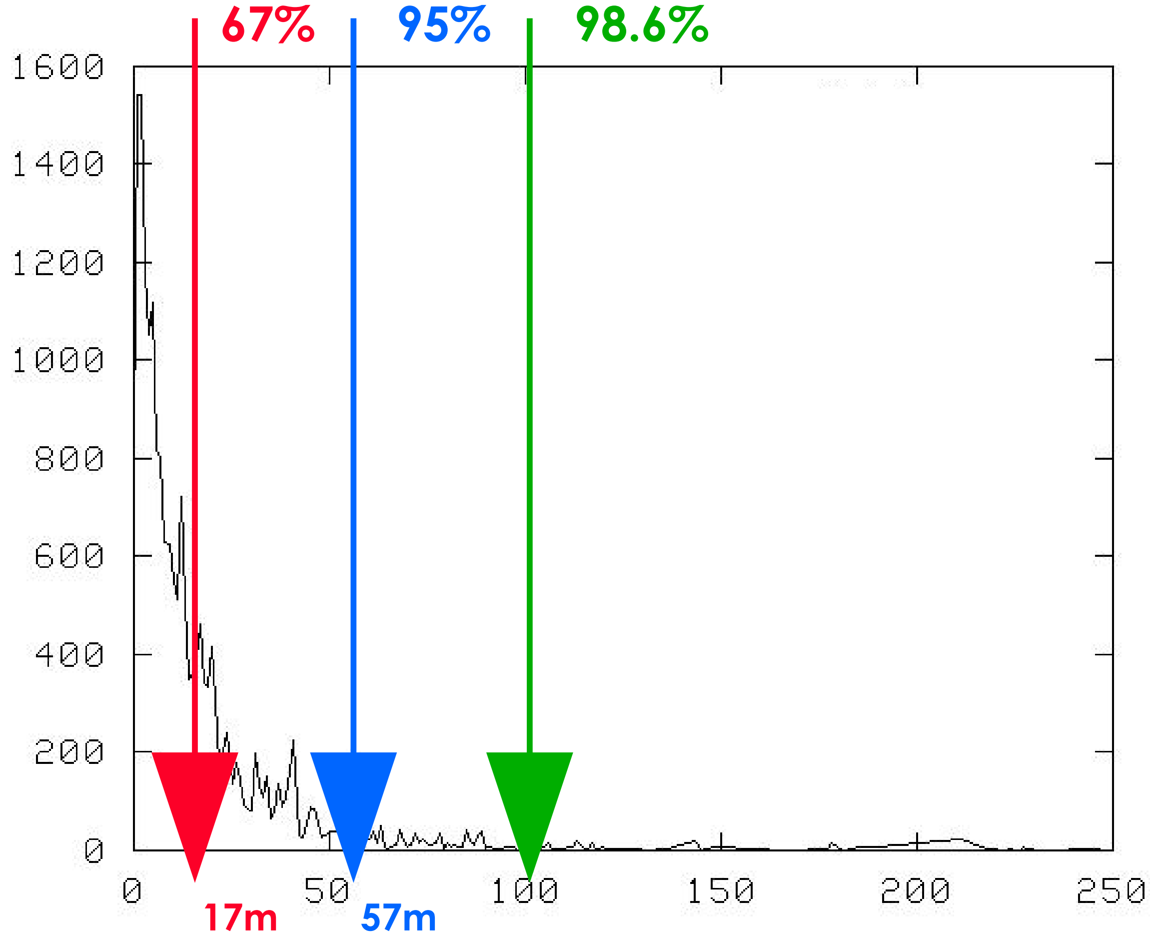
http://cosco.hiit.fi/demo/manhattan/



Details

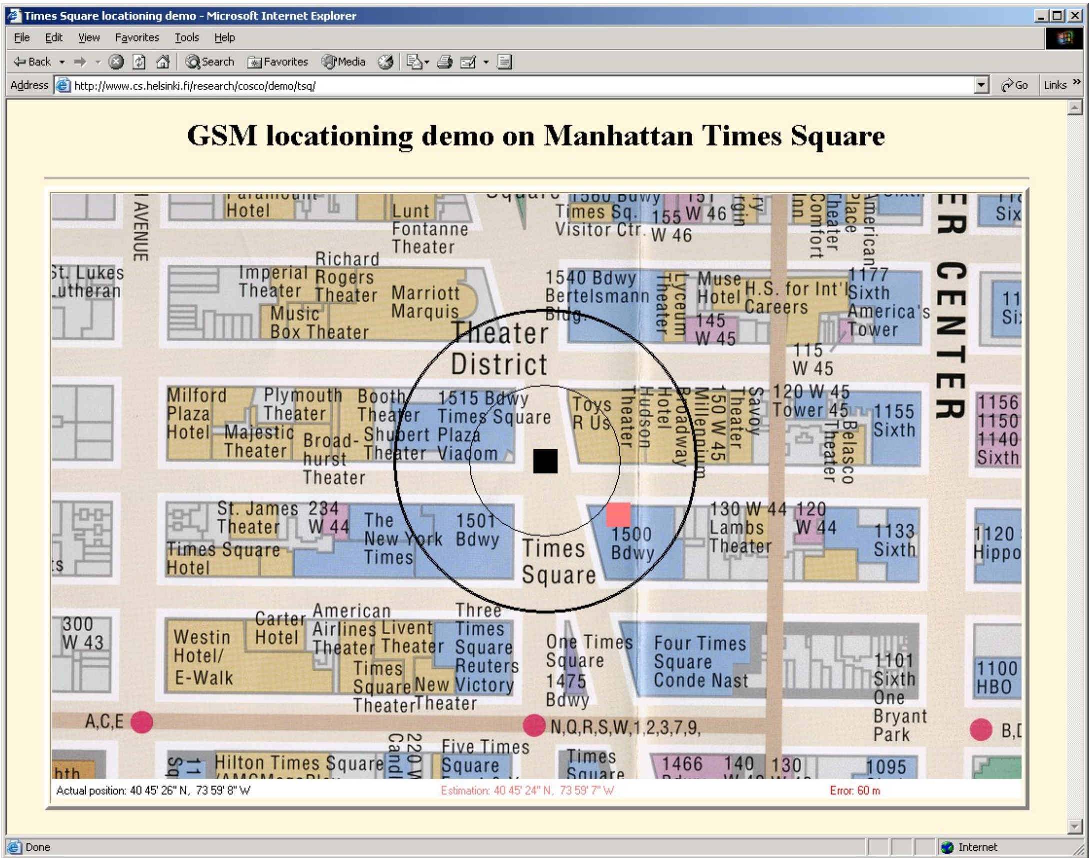
- Covering downtown Manhattan (10th - 114th St)
- Data gathering by car
- Modeling: 10 person days
- Target accuracy: less than 911 handset requirements
- Tests using cars

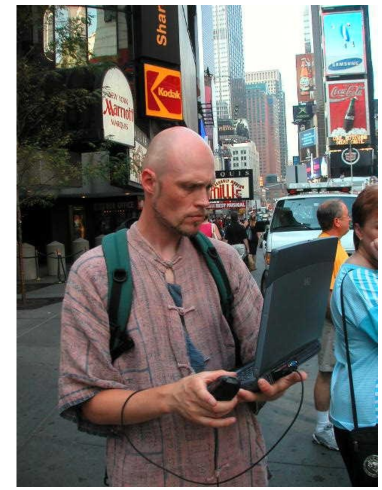
Accuracy of NYC Trial 2001



- 20166 points
- tracking; testing done in a car;

Trials: Manhattan 2002

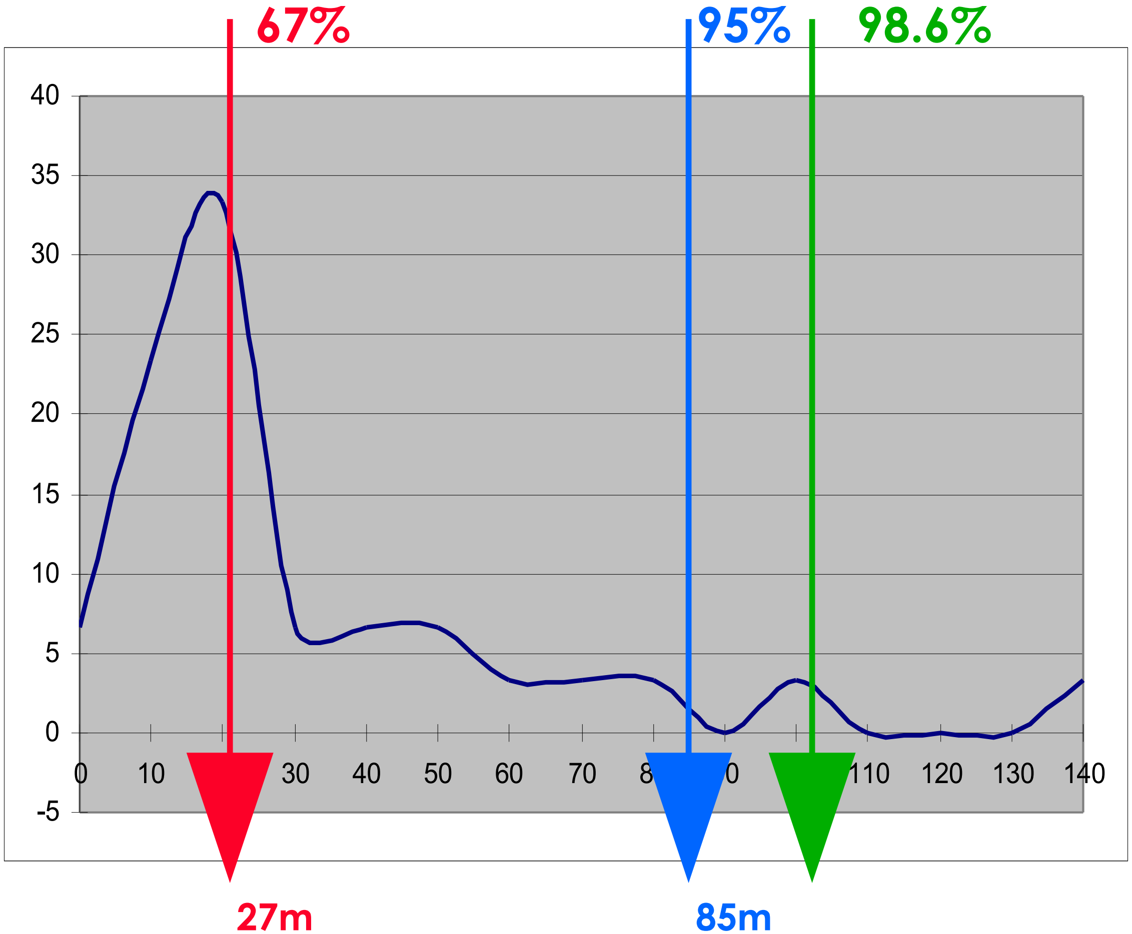




Challenges

- “real 911” simulation
  - No tracking information
  - Only up to 60 seconds of signal measurements
- Target accuracy: “theater level”
- Indoor testing (without indoor modeling)

Accuracy NYC Trial 2002



- 30 points
- static; testing done by walking;

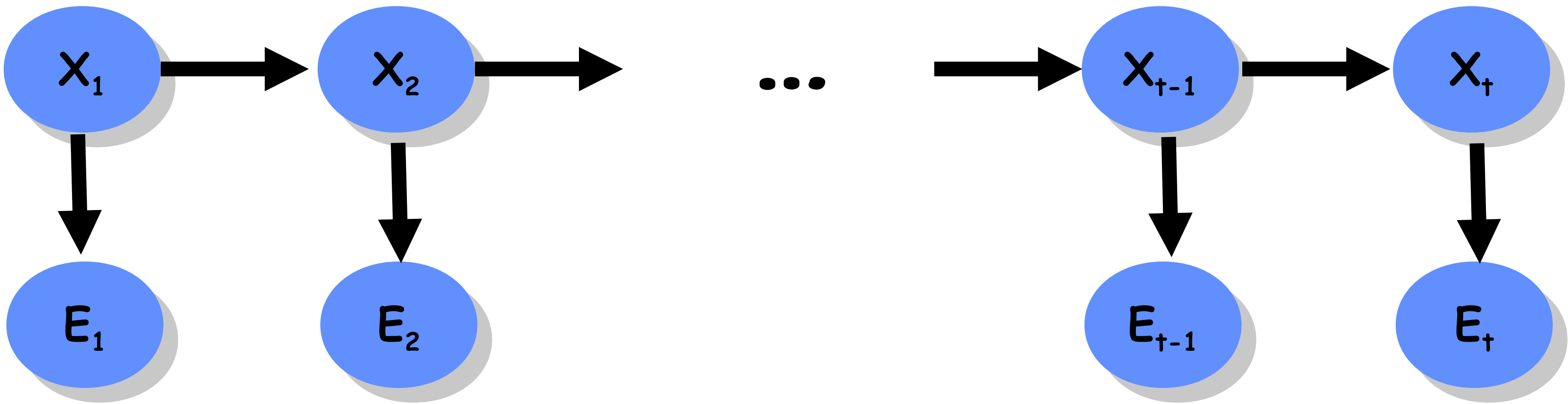
WiFi-positioning

P.Myllymäki, T. Roos, H.Tirri, P.Misikangas and J.Sievänen. *A Probabilistic Approach to WLAN User Location Estimation.* International Journal of Wireless Information Networks, Vol. 9, No. 3, July 2002.

More information: www.ekahau.com

Back to Hidden Markov Models

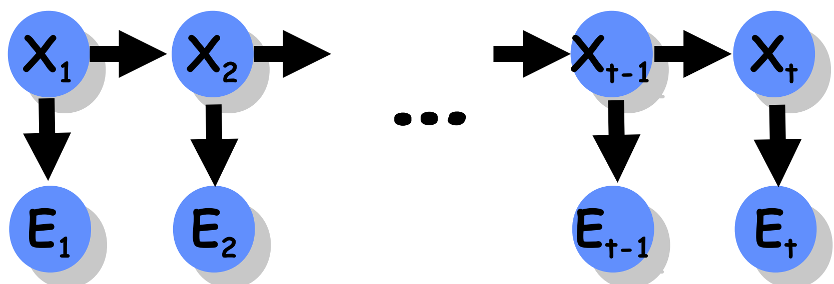
- For inference, easier to think of as a long chain of variables
- (For learning, the two-state model more fitting)



Joint probability of a HMM

- Joint probability factorizes like a BN
  - HMM is a Bayesian network!

$$P(X_0, X_1, E_1, X_2, E_2, \ldots, X_t, E_t) = P(X_0) \prod_{i=1}^{t} P(X_i | X_{i-1}) P(E_i | X_i)$$



- Common inference tasks:
  - Filtering / monitoring: $P(X_t \mid e_{1:t})$
  - Prediction: $P(X_{t+k} \mid e_{1:t})$, $k>0$
  - Smoothing: $P(X_k \mid e_{1:t})$, $k<t$
  - Explanation: $P(X_{1:t} \mid e_{1:t})$

Calculating $P(X_t \mid e_{1:t})$ in HMM

- Lets shoot for a recursive formula:

$$
\begin{aligned}
P(X_{t+1}|e_{1:t+1}) &= P(X_{t+1}|e_{t+1}, e_{1:t}) \\
&\propto P(e_{t+1}|X_{t+1}, e_{1:t}) P(X_{t+1}|e_{1:t}) \\
&= P(e_{t+1}|X_{t+1}) \underline{P(X_{t+1}|e_{1:t})}
\end{aligned}
$$

- and

$$
\begin{aligned}
P(X_{t+1}|e_{1:t}) &= \sum_{x_t} P(X_{t+1}, x_t|e_{1:t}) \\
&= \sum_{x_t} P(X_{t+1}|x_t, e_{1:t}) P(X_t|e_{1:t}) \\
&= \sum_{x_t} P(X_{t+1}|x_t) \underline{P(x_t|e_{1:t})}
\end{aligned}
$$

Forward algorithm for $P(X_t \mid e_{1:t})$

- Combining formulas we get a recursion

$$P(X_{t+1}|e_{1:t+1}) \propto P(e_{t+1}|X_{t+1}) \sum_{x_t} P(X_{t+1}|x_t) \underline{P(x_t|e_{1:t})}$$

- So first calculate

$$P(X_1|e_1) \propto P(e_1|X_1) \sum_{x_0} P(X_1|x_0) P(x_0)$$

- and then

$$P(X_2|e_{1,}e_2) \propto P(e_2|X_2) \sum_{x_1} P(X_2|x_1) P(x_1|e_1)$$

$$P(X_3|e_{1,}e_{2,}e_3) \propto P(e_3|X_3) \sum_{x_2} P(X_3|x_2) P(x_2|e_{1,}e_2)$$

Prediction: $P(X_{t+k} \mid e_{1:t})$, k>0

- $P(X_{t+1} \mid e_{1:t})$ part of the forward algorithm
- and from that on evidence does not count, and one can just calculate forward:

$$P(X_{t+2}|e_{1:t}) = \sum_{x_{t+1}} P(X_{t+2}|x_{t+1}, e_{1:t}) P(x_{t+1}|e_{1:t})$$
$$= \sum_{x_{t+1}} P(X_{t+2}|x_{t+1}) P(x_{t+1}|e_{1:t})$$
$$P(X_{t+3}|e_{1:t}) = \sum_{x_{t+2}} P(X_{t+3}|x_{t+2}, e_{1:t}) P(x_{t+2}|e_{1:t})$$
$$= \sum_{x_{t+2}} P(X_{t+3}|x_{t+2}) P(x_{t+2}|e_{1:t})$$

...

Smoothing: $P(X_k \mid e_{1:t})$, k<t

- Obvious move: divide $e_{1:t}$ to $e_{1:k}$ and $e_{k+1:t}$.

$$\begin{aligned} P(X_k|e_{1:t}) &= P(X_k|e_{1:k}, e_{k+1:t}) \\ &\propto P(X_k|e_{1:k}) P(e_{k+1:t}|X_k, e_{1:k}) \\ &= P(X_k|e_{1:k}) \underline{P(e_{k+1:t}|X_k)} \end{aligned}$$

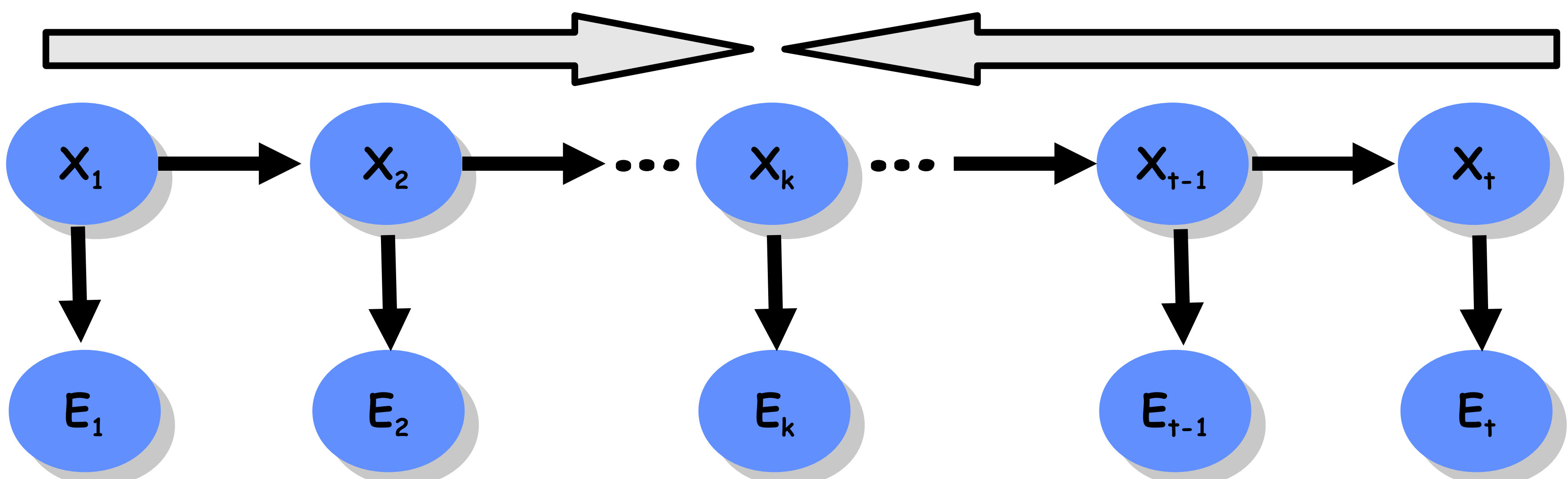
$$\begin{aligned} P(e_{k+1:t}|X_k) &= \sum_{x_{k+1}} P(x_{k+1}, e_{k+1:t}|X_k) \\ &= \sum_{x_{k+1}} P(x_{k+1}|X_k) P(e_{k+1:t}|x_{k+1}, X_k) \\ &= \sum_{x_{k+1}} P(x_{k+1}|X_k) P(e_{k+1}, e_{k+2:t}|x_{k+1}) \\ &= \sum_{x_{k+1}} P(x_{k+1}|X_k) P(e_{k+1}|x_{k+1}) \underline{P(e_{k+2:t}|x_{k+1})} \end{aligned}$$

- and the first (last) step:

$$\begin{aligned} P(e_t|X_{t-1}) &= \sum_{x_t} P(x_t, e_t|X_{t-1}) = \sum_{x_t} P(e_t|x_t, X_{t-1}) P(x_t|X_{t-1}) \\ &= \sum_{x_t} P(e_t|x_t) P(x_t|X_{t-1}) \end{aligned}$$

Back and forth

- ”Brute-force” smoothing of the whole sequence takes $O(t^2)$ time
- *Forward-backward* algorithm: O(t)
- Finding the most probable sequence works in the same manner (the Viterbi algorithm / Viterbi path)



The Viterbi algorithm

- Want to compute:

$$max_{X_1,\dots X_n} P(X_1,\dots,X_n|e_1,\dots,e_n)$$
$$= max_{X_n} max_{X_1,\dots,X_{n-1}} P(X_1,\dots,X_{n-1},X_n,e_1,\dots,e_n)$$

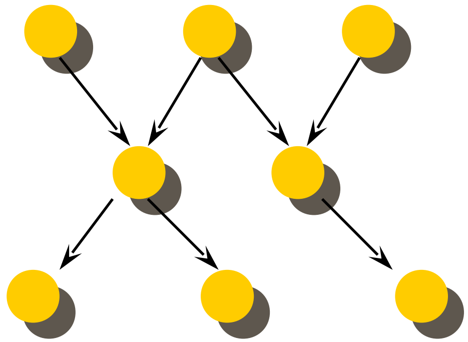
- Recursion:

$$max_{X_1,\dots,X_{n-1}} P(X_1,\dots,X_{n-1},X_n|e_1,\dots,e_n) = max_{X_1,\dots,X_{n-1}} P(X_1,\dots,X_{n-1},X_n,e_1,\dots,e_n)$$
$$= max_{X_1,\dots,X_{n-1}} P(e_n|X_n,X_1,\dots,X_{n-1},e_1,\dots,e_{n-1}) P(X_n,X_1,\dots,X_{n-1},e_1,\dots,e_{n-1})$$
$$= max_{X_1,\dots,X_{n-1}} P(e_n|X_n) P(X_n|X_1,\dots,X_{n-1},e_1,\dots,e_{n-1}) P(X_1,\dots,X_{n-1},e_1,\dots,e_{n-1})$$
$$= P(e_n|X_n) max_{X_{n-1}} P(X_n|X_{n-1}) max_{X_1,\dots,X_{n-2}} P(X_1,\dots,X_{n-2},X_{n-1}|e_1,\dots,e_{n-1})$$

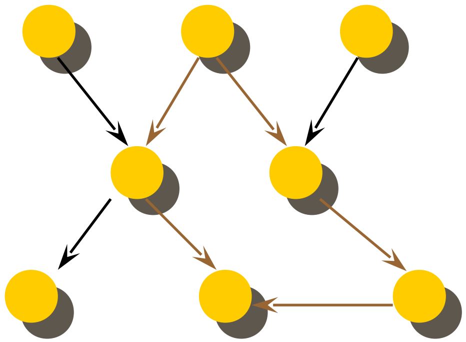
- More:
  - see e.g. Russel & Norvig, Chapter 15.2.

Exact inference in singly-connected BNs

- a singly connected BN = polytree (disregarding the arc directions, no two nodes can be connected with more than one path).

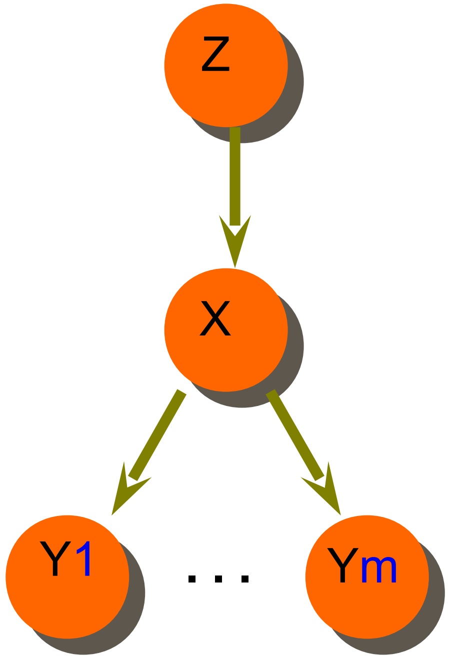


singly-connected



multi-connected

Probabilistic reasoning in singly-connected BNs



$$P(X|E) \propto P(X, E_+, E_-) \propto P(E_-|X) P(X|E_+)$$

$$P(E_-|X) = \prod_Y P(E_{Y-}|X)$$

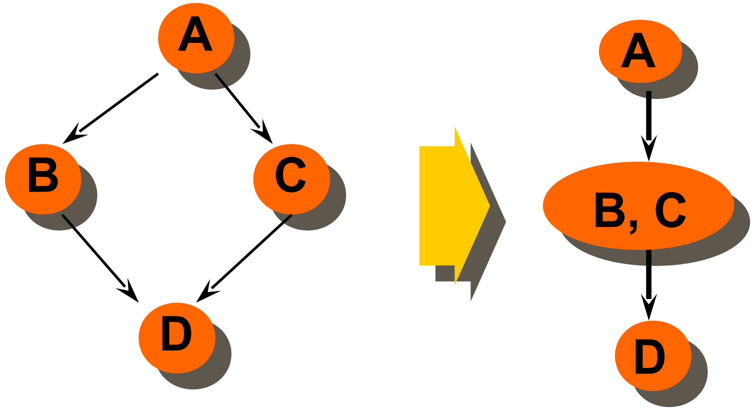
$$P(E_{Y-}|X) = \sum_Y P(E_{Y-}|Y) P(Y|X)$$

$$P(X|E_+) = \sum_Z P(X|Z) P(Z|E_{Z+})$$

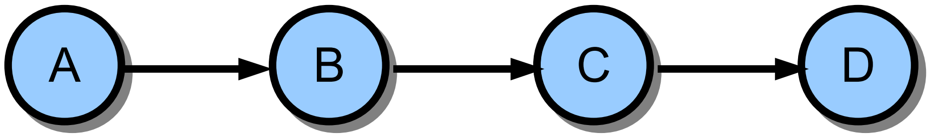
- a computationally efficient **message-passing** scheme: time requirement linear in the number of conditional probabilities in **ϴ**.

Probabilistic reasoning in multi-connected BNs

- generally not computationally feasible as the problem has been shown to be NP-hard (Cooper 1990, Shimony 1994).
- exact methods:
  - clustering
  - conditioning
  - variable elimination
- approximative methods:
  - stochastic sampling algorithms
  - loopy belief propagation



Variable elimination: a simple example



$$P(D) = \sum_{A,B,C} P(A,B,C,D)$$

$$= \sum_{C} \sum_{B} \sum_{A} P(A) P(B|A) P(C|B) P(D|C)$$

$$= \sum_{C} \sum_{B} P(C|B) P(D|C) \sum_{A} P(A) P(B|A)$$

$$= \sum_{C} P(D|C) \sum_{B} P(C|B) \sum_{A} P(A) P(B|A)$$

Approximate inference in Bayesian networks

- How to estimate how probably it rains next day, if the previous night temperature is above the month average?
  - count rainy and non rainy days after warm nights (and count relative frequencies).
- Rejection sampling for P(**X**|**e**) :
  1. Generate random vectors ($\mathbf{x}_r$,$\mathbf{e}_r$,$\mathbf{y}_r$).
  2. Discard those those that do not match **e**.
  3. Count frequencies of different $\mathbf{x}_r$ and normalize.

Rejection sampling, bad news

- Good news first:
  - super easy to implement
- Bad news:
  - if evidence **e** is improbable, generated random vectors seldom conform with **e**, thus it takes a long time before we get a good estimate P(**X**|**e**).
  - With long **E**, all **e** are improbable.
- So called likelihood weighting can alleviate the problem a little bit, but not enough.

P(X|mb(X))?

$$
\begin{aligned}
P(X|mb(X)) &= P(X|mb(x), Rest) \\
&= \frac{P(X, mb(X), Rest)}{P(mb(X), Rest)} \\
&\propto P(All) \\
&= \prod_{X_i \in \mathbf{X}} P(X_i|Pa(X_i)) \\
&= P(X|Pa(X)) \prod_{C \in ch(X)} P(C|Pa(C)) \prod_{R \in Rest \cup Pa(V)} P(R|Pa(R)) \\
&\propto P(X|Pa(X)) \prod_{C \in ch(X)} P(C|Pa(C))
\end{aligned}
$$

Gibbs sampling

- Given a Bayesian network for n variables **X** U **E** U **Y**, calculate P(**X**|**e**) as follows:

```
N = (associative) array of zeros
Generate random vector x,y.
While True:
    for V in X,Y:
        generate v from P(V | MarkovBlanket(V))
        replace v in x,y.
        N[x] +=1
        print normalize(N[x])
```

Why does it work

- All decent Markov Chains q have a unique stationary distribution P* that can be estimated by simulation.
- Detailed balance of transition function q and state distribution P* implies stationarity of P*.
- Proposed q, P(V|mb(V)), and P(**X**|**e**) form a detailed balance, thus P(**X**|**e**) is a stationary distribution, so it can be estimated by simulation.

Markov Chains: stationary distribution

- Defined by transition probabilities $q(x \rightarrow x')$ between states, where x and x' belong to a set of states X.
- Distribution P* over X is called stationary distribution for the Markov Chain q, if $P^*(x') = \sum_x P^*(x) q(x \rightarrow x')$.
- P*(X) can be found out by simulating Markov Chain q starting from the random state x_r.

Markov Chains: detailed balance

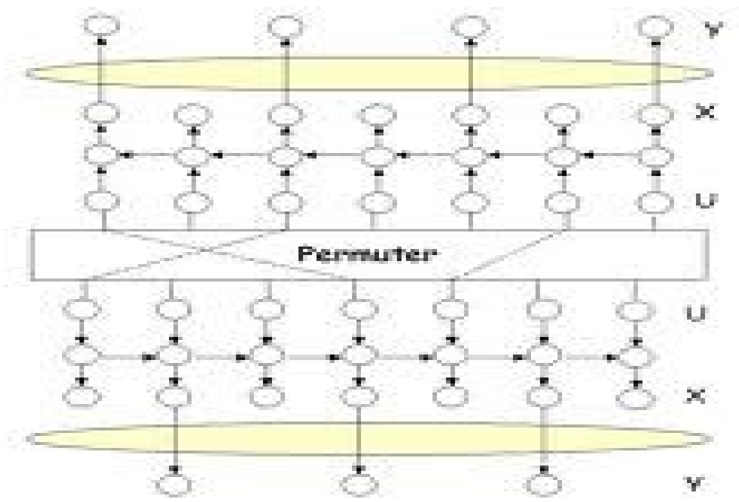
- Distribution P over X and a state transition distribution q are said to form a detailed balance, if for any states x and x', $P(x)q(x \to x') = P(x')q(x' \to x)$, i.e. it is equally probable to witness transition from x to x' as it is to witness transition from x' to x.
- If P and q form a detailed balance, $\sum_x P(x)q(x \to x') = \sum_x P(x')q(x' \to x) = P(x')\sum_x q(x' \to x) = P(x')$, thus P is stationary.

Gibbs sampler as Markov Chain

- Consider $\mathbf{Z}=(\mathbf{X},\mathbf{Y})$ to be states of a Markov chain, and $q((v,\mathbf{z}_{-V}))\rightarrow(v',\mathbf{z}_{-V}))=P(v'|\mathbf{z}_{-V}, \mathbf{e})$, where $\mathbf{Z}_{-V} = \mathbf{Z}-\{V\}$. Now $P^*(\mathbf{Z})=P(\mathbf{Z}|e)$ and q form a detailed balance, thus P* is a stationary distribution of q and it can be found with the sampling algorithm.
  - $P^*(\mathbf{z})q(\mathbf{z}\rightarrow\mathbf{z}') = P(\mathbf{z}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v,\mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v|\mathbf{z}_{-V},\mathbf{e})P(\mathbf{z}_{-V}|\mathbf{e})P(v'|\mathbf{z}_{-V}, \mathbf{e})$
    $= P(v|\mathbf{z}_{-V},\mathbf{e})P(v', \mathbf{z}_{-V}|\mathbf{e}) = q(\mathbf{z}'\rightarrow\mathbf{z})P^*(\mathbf{z}')$, thus balance.

Loopy belief propagation

- What happens if you just keep iterating the message passing algorithm in a multi-connected network?
- In some cases it produces the right results, or at least a good approximation
- Turbo codes



So let us play….