Beyond NP – The complexity of uncertainty

IPA Herfstdagen
November 29th 2007

Johan Kwisthout
Algorithmic Systems
Utrecht University

Universiteit Utrecht

Who am I?

- PhD student at Utrecht University at NWO-project: "Algorithmic Complexity of Probabilistic Networks"
- Working on complexity issues that are related to probabilistic (Bayesian) networks
  - Hardness proofs
  - New or improved algorithms
  - Lower bounds ("soft")
- Frontier between "Theoretical Computer Science"-community and "Uncertainty-community"
  - UU: Algorithmic Systems vs. Decision Support Systems group
  - ICALP/FCT/MFCS/STOC/FOCS vs. UAI/PGM/ECSQARU/ECAI

Universiteit Utrecht

Overview

Dealing with Uncertainty
- Probability Theory
- Probabilistic Networks
- Stochastic Satisfiability

- The Computational Complexity of Uncertainty
  - Probabilistic Turing Machines
  - The class PP and its properties
  - The Counting Hierarchy CH

- Problems, classes, and proof techniques
  - Probabilistic Inference
  - Network Tuning
  - Other results

- Conclusion

Universiteit Utrecht

Dealing with uncertainty

- In real life, we are forced to reason with imperfect knowledge and bounded resources
  - We do not know all the relevant facts
  - Which facts are relevant, anyway?
  - We haven't got time to take everything into account
  - Our information is inconsistent, vague, or imprecise
- To be helpful, computer programs that assist us in decision making need to deal with uncertainty
  - Determining the probability of a patient having a particular disease, given observations and clinical evidence
  - Finding a plan or schedule even when not all facts are known
  - Determining a weather forecast
  - Dealing with inconsistent sensor input in robots

Universiteit Utrecht

Probability Theory

- Mathematical theory of uncertainty: probability theory
- Pr(H|E) denotes probability of hypothesis H given evidence E

- What is the probability that a dice shows a '6'?
- Pr(H=6) = 1/6 (*prior probability*)
- What is the probability that the dice shows a '6' if someone tells us that the number is even?
- Pr(H=6|E=even) = 1/3 (*conditional probability*)

Universiteit Utrecht

Probability Theory

- Variables can be independent or not:
- Pr(Dice1 = 6 & Dice2 = 6) = Pr(Dice1 = 6) x Pr(Dice2 = 6)
- Pr(Rain = yes & Umbrella = yes) != Pr(Rain = yes) x Pr(Umbrella = yes)
- For *n* variables, we need to specify 2^n *joint probabilities*
- Try to use independence wherever possible!

Universiteit Utrecht

Probability Theory

- Conditioning
  $Pr(A=a_1) = Pr(A=a_1|B=b_1) \times Pr(B=b_1) +$
  $Pr(A=a_1|B=b_2) \times Pr(B=b_2) +$
  ...
- Marginalizing
  $Pr(A=a_1) = Pr(A=a_1 \wedge B=b_1) +$
  $Pr(A=a_1 \wedge B=b_2) +$
  ...
- Bayes' Theorem
  $Pr(A=a_1|B=b_1) = \frac{Pr(B=b_1|A=a_1) \times Pr(A=a_1)}{Pr(B=b_1)}$

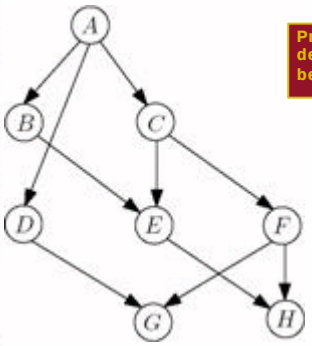
Universiteit Utrecht

Probabilistic Networks

- Often, probabilistic networks are used to represent variables in a particular domain, probabilities and independencies between variables
- Used in decision support systems, diagnosis, expert systems etc.
- Using network structure, conditional probabilities and reasoning rules, all sort of computations can be done:
  - Likeliness of variable having a particular value given evidence
  - Finding the most likely variable setting
  - Determining whether relations are monotone

Universiteit Utrecht

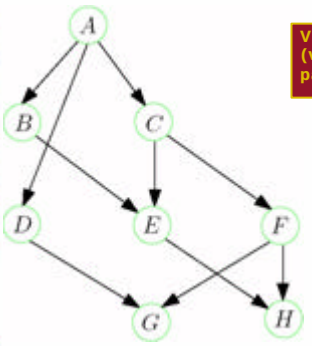
Probabilistic Networks



Probabilistic networks denote (in-)dependencies between variables

Universiteit Utrecht

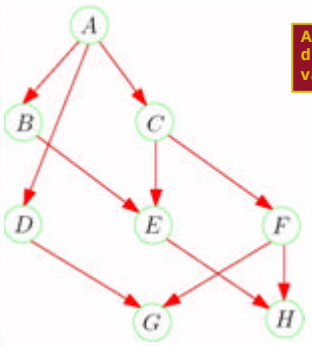
Probabilistic Networks



Variables *V* have values (v_1, v_2, ..., v_n) denoting particular states

Universiteit Utrecht

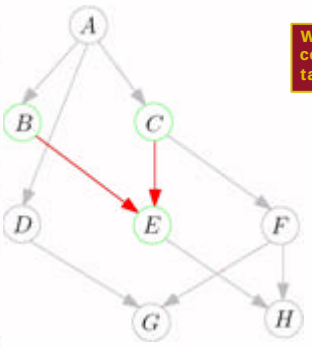
Probabilistic Networks



Arcs (*V*,*W*) denote dependencies between variables *V* and *W*

Universiteit Utrecht

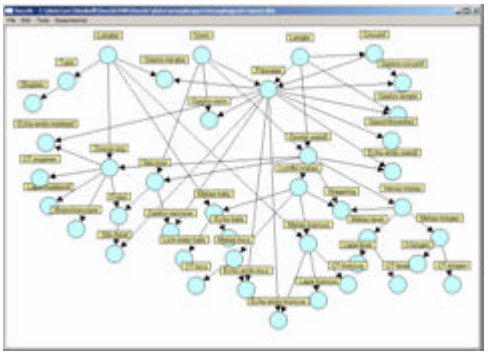
Probabilistic Networks



With each variable, a conditional probability table is associated

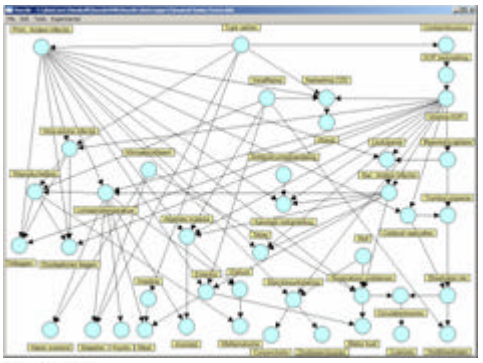
Universiteit Utrecht

Oesophageal Cancer Network



Universiteit Utrecht

Classical Swine Fever Network



Universiteit Utrecht

Stochastic Satisfiability

- Tautology f = (p ∨ :p):
  all instantiations satisfy this formula
- Contradiction f = (p ∧ :p):
  no instantiations satisfy this formula
- how *likely* will an arbitrary instantiation satisfy a particular formula?
- f = (p ∨ q): Pr(f = T) = 0.75
- f = (p ∧ q): Pr(f = T) = 0.25

Universiteit Utrecht

Stochastic Satisfiability

- Introduce *Random* operator ∀

$\forall x_i$ means: for *all* instantiations of x_i
$\exists x_i$ means: there *exists* an instantiation of x_i
$\forall x_i$ means: for a *random* instantiation of x_i

- Introduce *Expected value* E(X), with E(T)=1, E(F)=0

$\forall p,q\ E(p \vee q) = 0.75$

- SSAT (Stochastic Satisfiability)

$\exists x_1 \forall x_2 \forall x_3\ E((x_1 \vee x_2) \wedge (x_1 \vee x_3)) \geq q$

Universiteit Utrecht

Stochastic Satisfiability

- SSAT is a general problem with well known sub-problems when the quantifiers are restricted:
  - SATISFIABILITY – Only existential operators
  - TAUTOLOGY – Only universal operators
  - MAJSAT – Only random operators and E(f)
  - Q-SAT$_k$ – *k* alternating existential and universal operators
  - E-MAJSAT $X_{1\ldots k}$ bounded by existential operators, $X_{k+1\ldots n}$ bounded by random operators
  - A-MAJSAT $X_{1\ldots k}$ bounded by universal operators, $X_{k+1\ldots n}$ bounded by random operators
  - Two-player strategies or games against nature: alternating existential and universal (or random) operators

Universiteit Utrecht

Overview

- Dealing with Uncertainty
  - Probability Theory
  - Probabilistic Networks
  - Stochastic Satisfiability
- **The Computational Complexity of Uncertainty**
  - Probabilistic Turing Machines
  - The class PP and its properties
  - The Counting Hierarchy CH
- Problems, classes, and proof techniques
  - Probabilistic Inference
  - Network Tuning
  - Other results
- Conclusion

Universiteit Utrecht

Probabilistic Turing Machines

- A Probabilistic Turing Machine is a Non-Deterministic Turing Machine that branches according to a particular probability distribution
- Alternative definition: it has an additional *random tape* randomly filled with bits that is used during computation
- Complexity classes are defined based on a particular notion of *acceptance* on a Probabilistic Turing Machine
- Interesting classes are e.g.: ZPP, BPP, PP
- Also NP can be defined in such a way by forgetting about the probability distribution in each branch

Universiteit Utrecht

ZPP – zero-error polynomial time

- ZPP contains languages accepted by a Probabilistic Turing Machine **M**, *always* yielding correct results and *on average* polynomial running time (sometimes more)
- Alternative definition uses certificates (like NP): a string is in ZPP if, given a polynomial-length certificate, a deterministic Turing Machine can *verify* that the language is accepted by a Probabilistic Turing Machine
- While membership of NP assumes *existence* of such a certificate, membership of ZPP assumes that a *randomly constructed certificate* will do with probability > 0.5
- Typical algorithms: Las Vegas algorithm. Problems with efficient Las Vegas algorithms are in ZPP

Universiteit Utrecht

BPP – bounded polynomial time

- BPP contains languages that are accepted *by bounded majority* on a Probabilistic Turing Machine **M**
- This means, there exists a *c* such that, for all inputs, **M** accepts YES-instances with probability 0.5+c and rejects NO-instances with probability 0.5+c
- The error bound can be anywhere between 0 and 0.5, but it must be *independent* of the input (or input size)
- Typical algorithms: Monte Carlo algorithms. Problems with efficient Monte Carlo algorithms are in BPP
- Known: ZPP µ BPP. Conjectured: ZPP = BPP = P.

Universiteit Utrecht

PP – probabilistic polynomial-time

- PP contains languages that are accepted *by any majority* on a Probabilistic Turing Machine **M**
- This majority may depend on the input and may be exponentially small, hence BPP µ PP. This 'trivial' distinction between BPP and PP makes PP a very powerful class, including NP
  - Let f be a SATISFIABILIY instance with variables x_1 to x_n. Define ? = f $\vee x_{n+1}$. The majority of the instantiations to $x_{1\ldots n+1}$ accept ? iff f is satisfiable. Thus, NP µ PP.
- PP has complete problems (BPP and ZPP have not), the canonical complete problem is MAJSAT: given a Boolean formula f , does the majority of the truth assignments to its variables accept f ?

Universiteit Utrecht

PP – properties

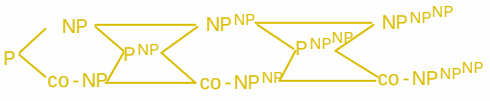
- PP is closed under complement, thus if a problem A 2 PP, then its complement is as well. Since NP µ PP, also co-NP µ PP
- PP µ PSPACE (just try all paths and count them, can be done in polynomial space)
- PP is a very powerful oracle (Toda's theorem):

  PH µ P^{PP}, i.e. P with a PP oracle contains the entire polynomial hierarchy
- PP (majority) relates to #P (exact solution counting): in fact Toda also proved that $P^{PP} = P^{\#P}$

Universiteit Utrecht

Oracles and the Counting Hierarchy

- Recall the polynomial hierarchy



- This hierarchy can be characterized using existential and universal operators (at least the NP and co-NP tracks)
- When adding PP, P^{PP}, NP^{PP}, co-NP^{PP} and PP^{PP} (and further on) we get the *Counting Hierarchy CH*
- PP µ P^{PP} µ co-NP^{PP}/NP^{PP} µ $P^{NP^{PP}}$ µ … µ PP^{PP}

Universiteit Utrecht

Complete problems in the Counting Hierarchy

Take $F = X_1..X_n$ partitioned in subsets X_A, X_B, X_C of variables:

NP^{PP} - E-MAJSAT: "*Is there* an instantiation to X_A, such that the *majority* of the instantiations to X_B satisfy F ?"

co-NP^{PP} - A-MAJSAT : "*For all* instantiations to X_A, does the *majority* of the instantiations to X_B satisfy F ?"

$NP^{NP^{PP}}$ - EA-MAJSAT: "*Is there* an instantiation to X_A, such that, *for all* instantiations to X_B, the *majority* of the instantiations to X_C satisfy F ?"

co-$NP^{NP^{PP}}$ - AE-MAJSAT: "*For all* instantiations to X_A, *is there* an instantiation to X_B, such that the *majority* of the instantiations to X_C satisfy F ?"

Universiteit Utrecht

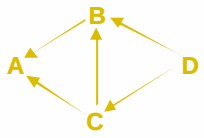
Overview

- Dealing with Uncertainty
  - Probability Theory
  - Probabilistic Networks
  - Stochastic Satisfiability
- The Computational Complexity of Uncertainty
  - Probabilistic Turing Machines
  - The class PP and its properties
  - The Counting Hierarchy CH
- **Problems, classes, and proof techniques**
  - Probabilistic Inference
  - Network Tuning
  - Other problems
- Conclusion

Universiteit Utrecht

Probabilistic Inference

- **x** is a configuration of all variables (e.g., $A = a_1$, $B = b_2$, $C = c_3$, $D = d_1$)
- $Pr(\mathbf{x}) = \prod_{A \in V(G)} Pr(A \mid \pi(A))$
- In this example, $Pr(a_1 b_2 c_3 d_1) = Pr(a_1 \mid b_2 c_3) \cdot Pr(b_2 \mid c_3 d_1) \cdot Pr(c_3 \mid d_1) \cdot Pr(d_1)$



Likewise:

- $Pr(a_1) = \sum_m Pr(a_1 \wedge x_m)$
- $Pr(a_1 \mid e) = \frac{\sum_m Pr(a_1 \wedge e \wedge x_m)}{\sum_m Pr(e \wedge x_m)}$

Universiteit Utrecht

Complexity of Inference

- Formal definition
  Let **B** be a probabilistic network, with C as a variable of interest and c as a particular value of C, and let E denote a set of evidence variables with instantiation e.
  Is $Pr(C=c \mid E=e) = q$?
- Intuitively:
  Randomly guess assignments to all variables with respect to their conditional probabilities: what is the probability of ending in an assignment consistent with C=c and E=e?
- Thus: conjectured complexity class is PP

Universiteit Utrecht

Probabilistic Inference is PP-Complete

To prove:

1. Reduce MAJSAT to INFERENCE
2. Show that there exists a probabilistic Turing Machine accepting INFERENCE instances in polynomial time

Note:
(2) is often taken *for granted* in complexity proofs, most proofs actually prove PP-hardness

In some cases completeness proofs are wanted (e.g. to separate PP-problems to NP^{PP} problems)

Universiteit Utrecht

PP-Hardness proof of INFERENCE

- Transform a MAJSAT instance to INFERENCE

$$F = \neg(X_1 \vee X_2) \vee \neg X_3$$

- Does the majority of the possible instantiations to X satisfy F ?
- This is a YES-instance, actually (5 out of 8 instances satisfy F)

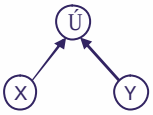
Universiteit Utrecht

Hardness proof constructs

- Variables in F are nodes with values T, F (uniform probability)

Pr(X = T) = 0.5
Pr(X = F) = 0.5

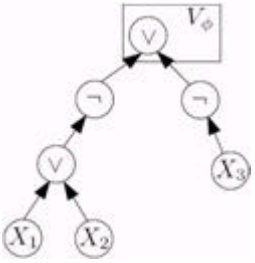
- Operators in F are nodes with values T, F (probability table = truth value of logical component



Pr(Ú = T| X = T and Y = T) = 1
Pr(Ú = T| X = T and Y = F) = 1
Pr(Ú = T| X = F and Y = T) = 1
Pr(Ú = T| X = F and Y = F) = 0

Hardness proof constructs

$F = \neg(X_1 \vee X_2) \vee \neg X_3$



$Pr(V_F = T| X_1 \wedge X_2 \wedge X_3) = 0$
$Pr(V_F = T| X_1 \wedge X_2 \wedge \neg X_3) = 1$
$Pr(V_F = T| X_1 \wedge \neg X_2 \wedge X_3) = 0$
:
$Pr(V_F = T| \neg X_1 \wedge \neg X_2 \wedge \neg X_3) = 1$

Is $Pr(V_\Phi = T) = 0.5$? Only if the majority of truth assignments satisfies F !

Ref: Litman, Majercik, & Pitassi (2001)

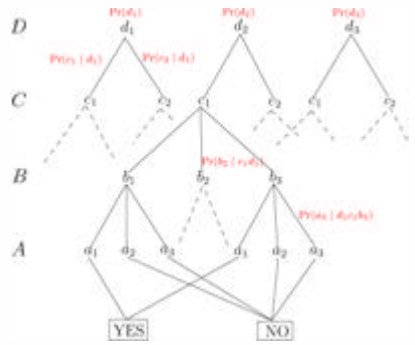
Membership proof INFERENCE

- Construct a *probabilistic Turing Machine* accepting an INFERENCE instance in polynomial time

Example: $Pr(A = a_1) = Pr(a_1|BC) \cdot Pr(B|CD) \cdot Pr(C|D) \cdot Pr(D)$ (summing over all configurations of B, C and D)

- Compute products backwards
- Choose an instantiation *at random* given the probability distribution
- If the configuration is consistent with $A = a_1$, then output YES, else output NO
- The probability of arriving at an accepting output is exactly $Pr(A = a_1)$

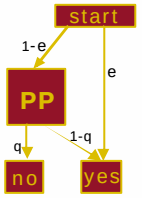
Membership proof INFERENCE



Membership proof INFERENCE

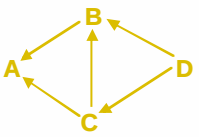
Additional constructs:

- We have shown how to calculate probabilities, but…
- …a probabilistic Turing Machine does not calculate a result, but accepts a string if Pr(YES) > 0.5 on a 'positive' input
- To decide $Pr(A=a_1) = q$, choose paths connecting a_1 to YES with probability 1-q and to NO with probability q
- Add e-path to YES to ensure that Pr(YES) > 0.5
- Now, this machine accepts if and only if $Pr(A=a_1) = q$



Network Tuning

- Assume that, for a particular instantiation of some of the variables in the network below, $Pr(A=a_1) = 0.2$
- When consulted, a domain expert indicates that this probability is too low, given the evidence: some conditional probabilities are not accurate



- Can we *revise* some of the *parameters* (entries in the CPT) such that $Pr(A=a_1) > 0.3$?
- This problem is known as the *Network Tuning* Problem
- Known algorithms are all exponential in the input size

Complexity of Network Tuning

- Formal definition
  Let **B** be a probabilistic network, with C as a variable of interest and c as a particular value of C, and let X be a set of *k* parameters. Is there a set ? of assignments to the parameters in X, such that Pr(C=c) = q?
- Intuitively:
  Non-deterministic guessing of a parameter assignment, using an INFERENCE oracle at the end of each path to test whether this assignment satisfies the constraints
- Thus: conjectured complexity class is NP^{PP}

Universiteit Utrecht

Network Tuning is NP^{PP}-complete

To prove:

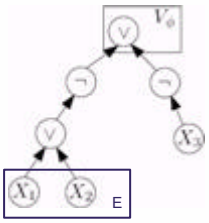
1. Reduce E-MAJSAT to TUNING
2. Show that there exists a non-deterministic Turing Machine that accepts TUNING, given access to a PP-oracle

Note:
(2) Similar to showing that there exist polynomial certificates, so that TUNING can be *verified* in polynomial time, given access to a PP-oracle. This is trivial: a set ? is a polynomial certificate. Since INFERENCE is PP-complete, membership follows.

Universiteit Utrecht

NP^{PP}-hardness proof

- Reduction from E-MAJSAT. Let (f, X_E, X_M) be an instance of E-MAJSAT. Example $F = \neg(X1 \vee X2) \vee \neg X3$
- Parameter set $X = p_i$ for all variables X_i in X_E



- For every X_i, a parameter setting $p_i=1$ would mimic $X_i=T$, and $p_i=0$ corresponds to $X_i=F$
- $Pr(V_f)$ is *multi-linear* in each p_i, so for every parameter a 0 or a 1 would maximize $Pr(V_f)$
- If a TUNING query with $Pr(V_f) = 0.5$ succeeds, the E-MAJSAT instance is satisfiable!

Universiteit Utrecht

k-CPT-Network Tuning is PP-complete

- In general, Network Tuning is NP^{PP}-complete if we allow X to contain parameters of multiple CPTs
- Theorem: Network Tuning is PP-complete if the number of CPTs is *bounded* and also, the nodes have a *bounded* treewidth

To prove:

1. PP-hardness is trivial (take k = 0, this is the INFERENCE problem!)
2. The tricky part here is membership proof of PP...

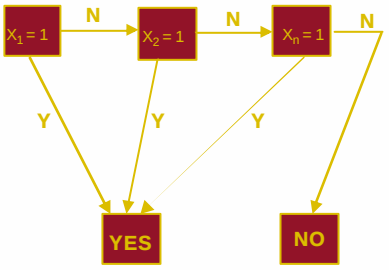
Universiteit Utrecht

Membership proof of *1*-CPT-Network Tuning

- First, let us assume that $k = 1$ and the corresponding node X has no incoming arcs. Let x_i denote the possible values of X and let Pr(C=c) be the output configuration
- $Pr(C=c) = ?_{x_i} Pr(C=c|X=x_i) \cdot Pr(X=x_i)$ (conditioning)
- $?_{x_i} Pr(X=x_i) = 1$ (definition of probabilities)
- Thus, Pr(C=c) is maximal if $x_i = 1$ for a particular x_i and all other parameters $x_j = 0$
- We can build on the INFERENCE membership proof and *chain* probabilistic Turing-Machines together!

Universiteit Utrecht

Membership proof of *k*-CPT-Network Tuning



Universiteit Utrecht

Membership proof of *k*-CPT-Network Tuning

- If the node with n values has m incoming arcs, then there are 2^m possible configurations to consider for each value, so we need $n \cdot 2^m$ chained probabilistic Turing Machines for each node
- Furthermore, if we consider k CPTs then we must consider parameter settings for each combination of values of all CPTs, so we need $(n \cdot 2^m)^k$ chained probabilistic Turing Machines
- For a bounded treewidth and a bounded number of CPTs, this is still a polynomial number, so Network Tuning is in PP under these constraints

Universiteit Utrecht

Some other completeness results

- All sorts of solution counting problems are PP or #P-hard
- Planning under uncertainty: often NP^{PP} or co-NP^{PP} (Littman, Goldsmith, and Mundhenk, 1998)
- Monotonicity in Probabilistic Networks: co-NP^{PP} (see handout), Partial MAP: NP^{PP} (Park & Darwiche, 2004)
- Finding the *median* of the set of feasible solutions: P^{PP} (Toda 1994, for a number of NP-complete problems)
- Generic Numerical Computing using floating points has various completeness results within CH (up to 4th level!)
- Still much interesting work to do!

Universiteit Utrecht

Conclusion

- We looked at uncertainty from a complexity-theoretic point of view
- PP-complete problems are 'harder' than NP-complete problems. Combining 'selecting' and 'guessing' yields problems somewhere in the Counting Hierarchy.
- PP-complete problems in networks have algorithms that are usually exponential in the treewidth of the graph (only). Problems that are higher in the Counting Hierarchy normally remain NP-complete or co-NP-complete even when restricted to poly-trees
- Being able to distinguish between NP, PP, NP^{PP} will give you more insight to algorithms for these sort of problems

Universiteit Utrecht