The Computational Complexity of Probabilistic Networks

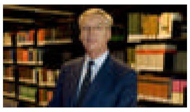
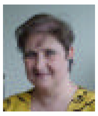
Research Seminar Logic and Automata
RWTH Aachen
March 12th 2009

Johan Kwisthout
Algorithmic Systems
Utrecht University

Universiteit Utrecht

About me

- MSc in Computer Science, 2005 (Open University, Heerlen)
- MSc in Artificial Intelligence, 2006 (Nijmegen University)
- PhD in Computer Science, defense July 1st 2009
  1st promotor: Jan van Leeuwen (algorithms & complexity); 2nd promotor: Linda van der Gaag (probabilistic networks)

- My work: cooperative project between Algorithmic Systems Group and Decision Support Systems Group

Universiteit Utrecht

Our group: Algorithmic Systems

- New Models of Computing
  - Interactive Turing Machines (van Leeuwen)
  - Evolving Systems (van Leeuwen, Verbaan)
- Network Algorithms
  - Treewidth (Bodlaender)
  - Fixed Parameter Tractability (Bodlaender)
  - Kernelization (Penninckx, Bodlaender)
  - Network Flow in Sensor Networks (van Dijk)
- Exact algorithms for NP-complete graph problems
  - Inclusion-Exclusion (Nederlof, van Rooij)
  - Measure-and-Conquer (van Rooij)
- Operations Research
  - Column Generation (Hoogeveen, Diepen)
  - Scheduling and Timetabling (van den Akker)

Universiteit Utrecht

Take home - message

- Probabilistic Networks are an interesting subject to study in a complexity-theoretical sense: many problems related to these networks are complete for complexity classes that have few "real world" complete problems
  - Tunable Monotonicity: $NP^{NP^{PP}}$-complete
  - Enumerating MAP: $P^{PP^{PP}}$-complete
- This gives us insight in general in problems that combine selecting, verifying properties, enumeration, and stochastic reasoning
- Determining the exact complexity (rather than 'NP-hard') of such problems is important to know which restrictions are needed to obtain feasible algorithms

Universiteit Utrecht

Overview

- Probabilistic Networks – usage and definitions
- Complexity of Inference
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- Lower bound on inference running time
- Oracles and the Counting Hierarchy
- Interesting Problems in PNs and their complexity
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- How about other formalisms like games?

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

Probabilistic Networks

- Often, probabilistic networks are used to represent **stochastic variables** in a particular domain, **probabilities** and **independencies** between variables using directed acyclic graphs
- 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 values of a set of variables
  - Determining whether relations are monotone
  - Determining whether variables are sensitive to small changes

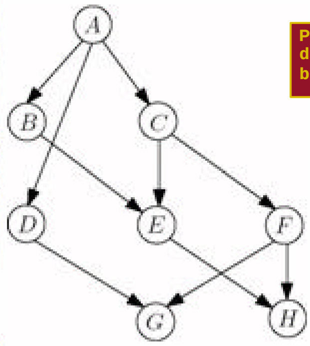
Probabilistic Networks

- Formal definition: **B** = (**G**, Γ), where **G** = (**V**, **A**) is an directed acyclic graph, and Γ denotes the set of conditional probability distributions.
- Each V in **V** is a stochastic variable; arcs in **A** denote dependencies between variables
- For each V a conditional probability table is defined, giving the probability distribution of a variable, given a value assignment to its **parents** in the network
- All probabilities of interest can be calculated from this structure using well-known properties of probability theory

Probability Theory

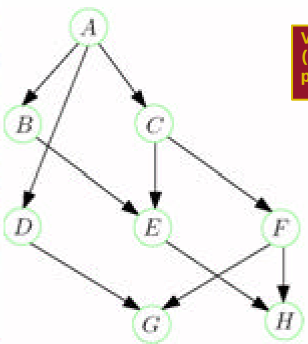
- **Conditioning**
  $Pr(A=a_1) = Pr(A=a_1|B=b_1) \times Pr(B=b_1) +$
  $Pr(A=a_1|B=b_2) \times Pr(B=b_2) +$
  ...
- **Marginalizing**
  $Pr(A=a_1) = Pr(A=a_1 \wedge B=b_1) +$
  $Pr(A=a_1 \wedge B=b_2) +$
  ...
- **Chain Rule**
  $Pr(x_1,\ldots,x_n) = Pr(x_n|x_1,\ldots,x_{n-1}) \times \ldots \times Pr(x_2|x_1) \times Pr(x_1)$

Probabilistic Networks



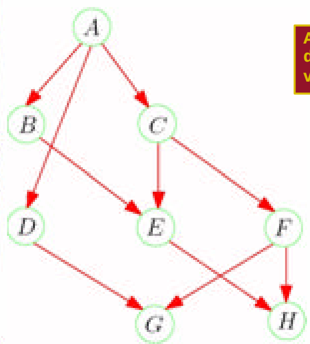
Probabilistic networks denote (in-)dependencies between variables

Probabilistic Networks



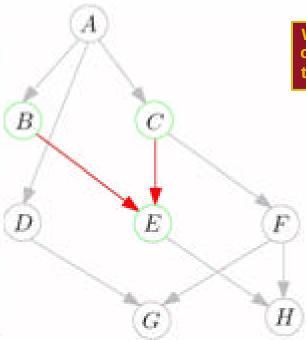
Variables *V* have values (v_1, v_2, ..., v_n) denoting particular states

Probabilistic Networks



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

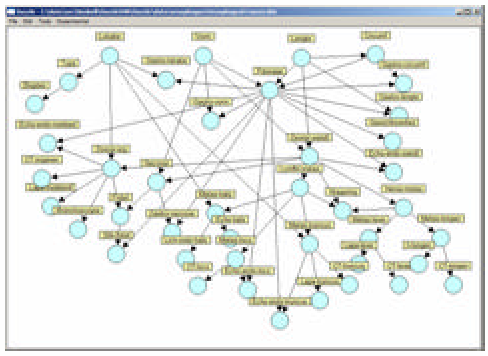
Probabilistic Networks



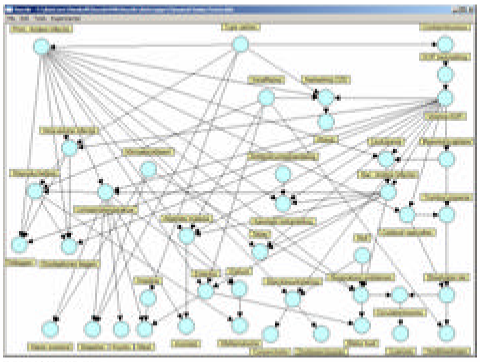
With each variable, a conditional probability table is associated

$Pr(E = e_1 \mid B = b_1 \wedge C = c_1) = 0.15$
$Pr(E = e_1 \mid B = b_2 \wedge C = c_1) = 0.23$
$Pr(E = e_1 \mid B = b_1 \wedge C = c_2) = 0.15$
$\vdots$
$Pr(E = e_4 \mid B = b_4 \wedge C = c_4) = 0.01$

Oesophageal Cancer Network



Classical Swine Fever Network

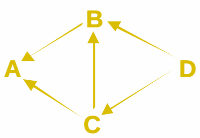


Overview

- Probabilistic Networks – usage and definitions
- **Complexity of Inference**
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- Lower bound on inference running time
- Oracles and the Counting Hierarchy
- Interesting Problems in PNs and their complexity
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- How about other formalisms like games?

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:

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

Probabilistic Inference

- In general, inference takes exponential time in the network size
- Known algorithms often use some form of **clustering** and are exponential only in the treewidth of the (moralised) graph
- Known results (Roth, 1998; Littman, 2001): Inference is #P-complete and has a PP-complete decision variant
- New result (Kwisthout, yet unpublished): no general algorithm can solve arbitrary instances with high treewidth in subexponential time unless the ETH fails

Probabilistic Turing Machines

- A Probabilistic Turing Machine is a Non-Deterministic Turing Machine that branches according to a particular probability distribution
- Complexity classes are defined based on a particular notion of *acceptance* on a Probabilistic Turing Machine
- Interesting classes are e.g.:
  - ZPP (zero error, on **average** polynomial running time)
  - BPP (polynomial running time, **bounded** error)
  - PP (polynomial running time, **unbounded** error)
- Also NP can be defined in such a way by forgetting about the probability distribution in each branch

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 ⊆ 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 ∨ x_{n+1}. The majority of the instantiations to $x_{1...n+1}$ accept ? iff f is satisfiable. Thus, NP ⊆ 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 ?

Probabilistic Inference is PP-Complete

To prove:

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

Note:
(1) 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)

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|E=e) = q?
- Conjectured complexity class is PP
- Intuitively: if we randomly guess assignments to all variables with respect to their conditional probabilities: is the probability of ending in an assignment consistent with C=c and E=e = q?
- eg. $Pr(a_1b_2c_3d_1) = Pr(a_1|b_2c_3) \cdot Pr(b_2|c_3d_1) \cdot Pr(c_3|d_1) \cdot Pr(d_1)$

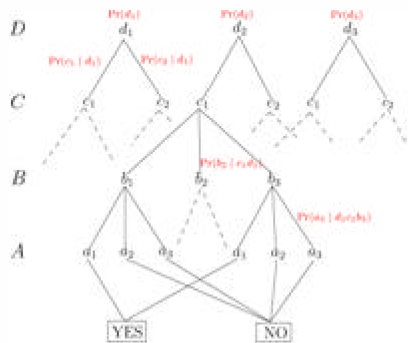
Membership proof INFERENCE

- 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



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)
- We construct a network B_F from an instance F

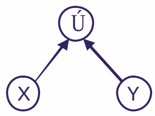
Universiteit Utrecht

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

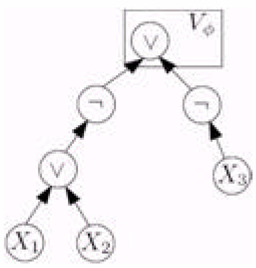


Pr(Ú = T| X = T and Y = T) = 1
Pr(Ú = T| X = T and Y = F) = 1
Pr(Ú = T| X = F and Y = T) = 1
Pr(Ú = T| X = F and Y = F) = 0

Universiteit Utrecht

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: Littman, Majercik, & Pitassi (2001)

Universiteit Utrecht

Overview

- Probabilistic Networks – usage and definitions
- Complexity of Inference
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- **Lower bound on inference running time**
- Oracles and the Counting Hierarchy
- Interesting Problems in PNs and their complexity
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- How about other formalisms?

Universiteit Utrecht

Lower bound on Inference

- Inference is PP-complete, so polynomial time algorithms are highly unlikely to exist
- Algorithms are known that are exponential in the treewidth of the (moralised) graph
- We prove that these algorithms are optimal up to an logarithmic factor in the exponent, unless the **Exponential Time Hypothesis** fails
- ETH (Impagliazzo): there exists a constant c > 1 such that deciding any 3SAT instance with n variables takes $O(c^n)$ time

Universiteit Utrecht

Lower bound on Inference

- Marx (2007, STOC) proved lower bound on (binary) CSP and Graph Homomorphism for *any* graph with high treewidth, using novel characterization of treewidth and embeddedness of graphs
- We introduce treewidth-preserving many-one reductions to reduce CSP to Inference in polynomial time, where the inference instance has the same treewidth (up to a constant) as the CSP instance
- Hence, if we have an algorithm that can solve **any** arbitrary Inference instance with high treewidth efficiently, then we can also solve the CSP instance with high treewidth efficiently, contradicting the ETH

Universiteit Utrecht

Lower bound on Inference

- Hardness proof with extra constraint:
  - A many-one reduces to B, i.e. $x \in A \rightarrow f(x) \in B$
  - This reduction takes polynomial time
  - **AND** $tw(f(x)) = tw(x) + l(x)$ for a linear function l
- Sketch of reduction
  - Let I = <V, D, C> be a CSP instance with binary constraints
  - Construct B: every variable in I is a node X in B; every relation in I is a node R in B with as parents the two variables involved;
  - Conditional Probability $Pr(R=true|X_i,X_j) = 1$ for a particular value of X_i, X_j if that combination is in C(I)
  - Pr(all Rs=true) > 0 iff. CSP is solvable
  - 'AND'-construction to connect all R nodes in single S-node
  - Here we must take care to guarantee treewidth

Universiteit Utrecht

Lower bound on Inference

- We have shown that no generic algorithm can solve arbitrary Inference instances with high treewidth in subexponential time
- However, algorithms may exist that work only on a particular class of instances – for example, using a particular direction of the arcs – that may run fast
- Yet, the known algorithms are all generic ones that work on all possible networks and have a guarenteed running time that is exponential in the treewidth of the graph.
- Thus, these algorithms are essentially optimal (up to a logarithmic factor in the exponent)

Universiteit Utrecht

Overview

- Probabilistic Networks – usage and definitions
- Complexity of Inference
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- Lower bound on inference running time
- Oracles and the Counting Hierarchy
- Interesting Problems in PNs and their complexity
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- How about other formalisms like games?

Universiteit Utrecht

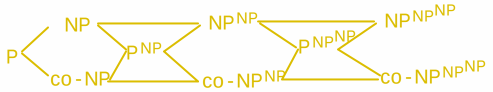
Oracle Turing Machine

- A Turing Machine **M** has *oracle access* to a set A if membership queries of A can be decided in constant time
- **M** puts a string x on its oracle tape, enters the oracle state q_O and in the next timestep, **M** is in state q_{O+} if x is in A, else **M** is in state q_{O-}.
- If A corresponds to a complete problem for a class C, then we will write e.g. NP^{PP} to denote the class of problems decidable by a nondeterministic Turing Machine with oracle access to a Probabilistic Turing Machine

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*
- $PP \subseteq P^{PP} \subseteq co\text{-}NP^{PP}/NP^{PP} \subseteq PP^{PP} \subseteq \ldots \subseteq PSPACE$

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 ?"

P^{PP} - Kth-SAT: "What is the lexicographical *kth* instantiation to X that satisfies F ?"

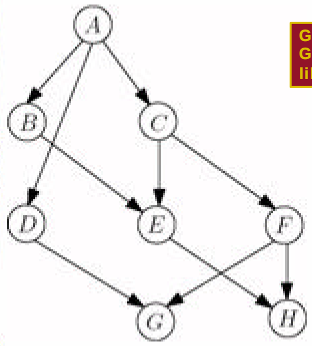
$P^{PP^{PP}}$ - Kth-MAJSAT: "What is the lexicographical *kth* instantiation to X_A, such that the *majority* of the instantiations to X_B satisfy F ?"

Universiteit Utrecht

Overview

- Probabilistic Networks – usage and definitions
- Complexity of Inference
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- Lower bound on inference running time
- Oracles and the Counting Hierarchy
- **Interesting Problems in PNs and their complexity**
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- How about other formalisms like games?

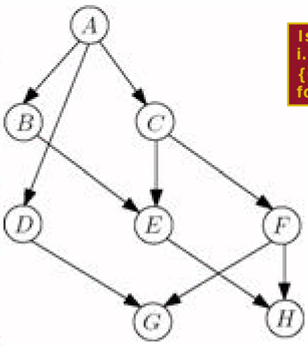
Partial MAP



Given an instantiation to G and H, what is the most likely value of {A,B,C} ?

NP^{PP}-complete
(Park & Darwiche,2004)

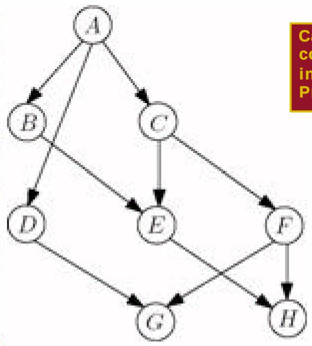
Monotonicity



Is C monotone in {G,H}, i.e. do higher values for {G,H} make higher values for C more likely?

co-NP^{PP}-complete
(van der Gaag et al, 2004)

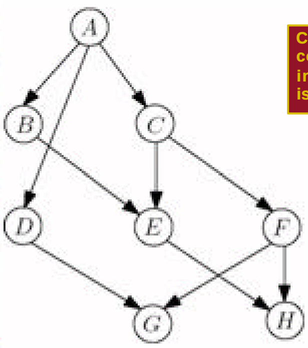
Parameter Tuning



Can we adjust a set X of conditional probabilities in the network such that Pr(C=c) > q?

NP^{PP}-complete
(Kwisthout and Van der Gaag, 2008)

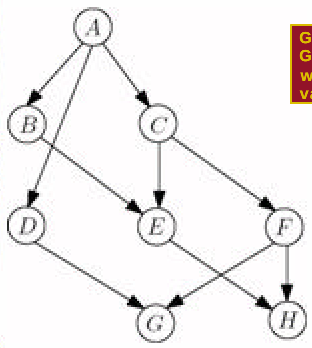
Tunable Monotonicity



Can we adjust a set X of conditional probabilities in the network such that C is monotone in {G, H}

$NP^{NP^{PP}}$-complete
(Kwisthout , unpublished)

Enumerating Partial MAP

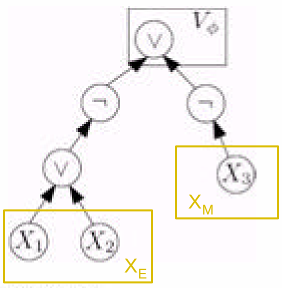


Given an instantiation to G and H and an integer *k*, what is the ***k*th** most likely value of {A,B,C} ?

$P^{PP^{PP}}$-complete
(Kwisthout , 2008)

Generic hardness proof structure

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



Reduction from a class in the Counting Hierarchy (E-Majsat etc)

Quantification over subsets of variables X

e.g. Partial MAP: IS there a variable instantiation to X_E such that $Pr(V_F) > 0.5$?

Marginalize over all instantiations of X_M

Problems on Probabilistic Networks

- These problems all combine selecting, verifying and/or enumeration with stochastic reasoning
- Typically, many interesting problems from various areas have such properties
- E.g. stochastical planning (Littman et al, 1998); Partially Observable MDPs (Goldsmith et al, 1996); stochastic scheduling (van den Akker and Hoogeveen, 2008)
- However, often only NP-hardness is shown, without further examining the exact complexity
- Nevertheless, this is interesting to determine which restricted variants are feasible vs. remain hard, and to make use of approximation strategies for such classes

On finding the exact complexity

- For example, Parameter Tuning:
  - Is NP^{PP}-complete in general
  - Remains NP-complete when inference is easy
  - Remains PP-complete for a bounded number of parameters
  - Thus, polynomial algorithms are unlikely except when **both** constraints are met
- Thus, studying the exact complexity and characteristics of such problems gives us more insight about why some things are hard to compute
- Also, it can help to determine whether efforts should be placed in improving existing algorithms
  - Known Parameter Tuning algorithm is exponential in both the treewidth of the graph and the number of parameters

Overview

- Probabilistic Networks – usage and definitions
- Complexity of Inference
  - The Inference problem
  - Probabilistic Turing Machines and the class PP
  - Inference is in PP (proof)
  - Inference is PP-hard (proof)
- Lower bound on inference running time
- Oracles and the Counting Hierarchy
- Interesting Problems in PNs and their complexity
  - Partial MAP, Monotonicity, Parameter Tuning, Tunable Monotonicity, Enumeration
- **How about other formalisms like games?**

Stochastic games

- Stochastic games (Shapley, 1953) introduce a Random player, next to deterministic players Even and Odd
- In Simple Stochastic Games, the complexity of finding the likely winner of a SSG is in NP $\cap$ co-NP (Condon, 1992)
- Can we formulate a stochastic game, including winning conditions, such that the complexity of finding the likely winner is:

  PP-complete? P^{PP}-complete? NP^{PP}-complete?
- How do the properties of such stochastic games relate to the properties of these classes?

Stochastic games

- (Infinite) games have a strong application in specification and verification of interactive systems
- From the GAMES Programme:
  - specifying a module amounts to formally describing a game
  - synthesizing a module amounts to computing a winning strategy
  - verifying a module against a specification amounts to checking that a strategy is indeed a winning strategy
- What if finding such a strategy turns out to be infeasible in the game?
  - Maybe we can pinpoint its exact complexity in order to show 'where the hardness comes from' and how we can restrict the problem to reduce its complexity

Conclusions, looking back and forth

- Take home-message: finding the exact complexity of a problem (vs ‘NP-hardness’) is relevant to pinpoint which restrictions are needed to arrive at feasible algorithms
- We have discussed the inference problem and its PP-completeness proof and sketched a proof of its lower bound complexity using treewidth preserving reductions
- We have discussed some other problems that combine selecting, verifying, enumeration and stochastic reasoning
- We suggested some further work to ‘export’ the take home-message to other applications like stochastic games with a reference to the GAMES program

Universiteit Utrecht