# Approximate Inference for Logic Programs with Annotated Disjunctions

**Stefano Bragaglia**
DEIS — Università di Bologna
stefano.bragaglia@unibo.it

**Fabrizio Riguzzi**
ENDIF — Università di Ferrara
fabrizio.riguzzi@unife.it

## Sintax & Semantic of LPADs

### Motivations

- Increasing interest in combining logic and probability.
- Many formalisms such as Markov Logic Networks, ProbLog and Logic Programs with Annotated Disjunctions (LPADs) enrich logic with probabilistic information. LPAD is the most interesting one because:
  - It can effectively express cause-effect relationships among events;
  - It can communicate well the possible effects of a single cause;
  - It can represent the contemporary contribution of more causes to the same effect in a very natural way.

### LPAD: An Example

The following is a simple **medical diagnosis** application:

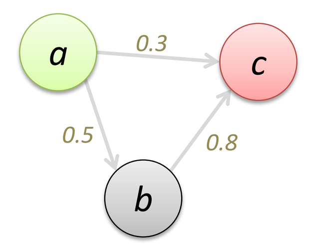
```
C1:  sneezing(X): 0.7 ∨ not_sneezing(X): 0.3 ← flu(X).
C2:  sneezing(X): 0.6 ∨ not_sneezing(X): 0.4 ← hayfever(X).
C3:  flu(andrew).
C4:  flu(david).
C5:  hayfever(david).
C6:  hayfever(robert).
```

It means that flu and hayfever may both be cause of sneezing with the given probability values. The information available let us determine the probability of sneezing in case of:

| | | | |
|---|---|---|---|
| • **flu** | ex.: ?- sneezing(andrew). | attended value: | **0.7** |
| • **hayfever** | ex.: ?- sneezing(robert). | attended value: | **0.6** |
| • **both** | ex.: ?- sneezing(david). | attended value: | **0.88** |

### Syntax of LPAD

- An LPAD is a set of disjunctive clauses.
- Each atom in their head is annotated with a probability value between 0 and 1.




```
C1:  path(Node,Node).
C2:  path(Src,Dst) ← edge(Src,Node), path(Node,Dst).
C3:  edge(a,c): 0.3.
C4:  edge(a,b): 0.5.
C5:  edge(b,c): 0.8.
```

- The atoms of the head represent the mutually exclusive and exhaustive set of effects of the event represented by the body.
- The sum of the probabilities associated to the head atoms must be 1 (if not, another atom with the missing probability value is understood).

The above example depicts a typical **path problem**.

### Semantic of LPAD

- It is based on the concept of **instance** that is a normal logic program obtained by choosing a logical atom from the head of each grounding of every clause of the LPAD.
- The **probability of an instance** is computed by multiplying the probability values of all the atom choosen for that instance.
- The **probability of a query** is given by the sum of the probabilities of each instance where the query is true.

### Probability of the Queries

The above considerations lead to the following results:

#### Medical Diagnosis

**Query:** `?- sneezing(david).`
**Instances where query is true / available instances:** 3/4

`P = 0.7×0.6 + 0.7×0.4 + 0.3×0.6 = 0.88`

#### Path Problem

**Query:** `?- path(a,c).`
**Instances where query is true / available instances:** 5/8

`P = 0.7×0.5×0.8 + 0.3×0.5×0.2 + 0.3×0.5×0.8 + 0.3×0.5×0.2 + 0.3×0.5×0.8 = 0.58`

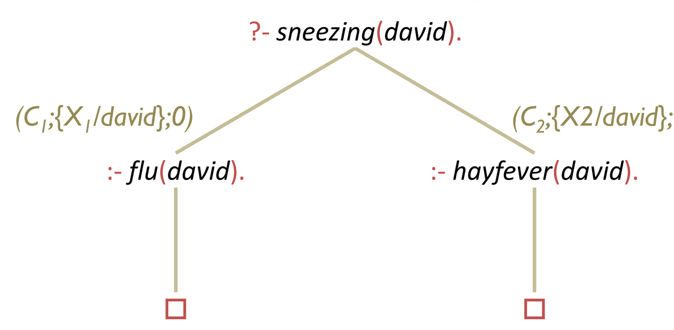
## Inference & Approximation

### Standard Inference

The **cplint** system (http://www.ing.unife.it/software/cplint/) can perform inference on LPADs in two steps by computing:

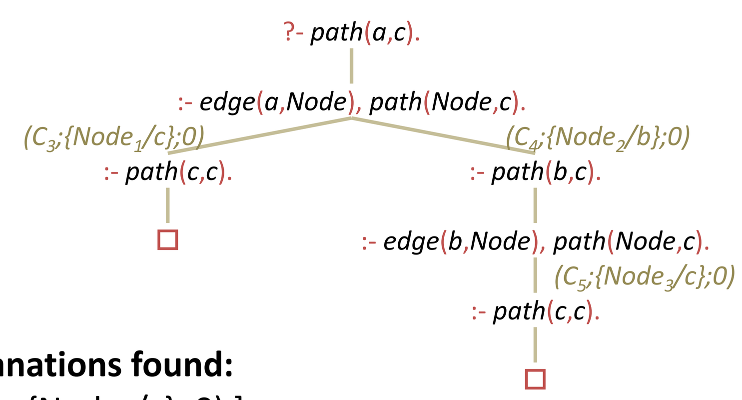
1. the explanations of the query with a meta-interpreter that performs resolution and keeps current set of choices;
2. the probability of the query by making explanations mutually exclusive with a dynamic programming algorithm that traverses their equivalent Binary Decision Diagram (BDD).

#### Explanations of Medical Diagnosis



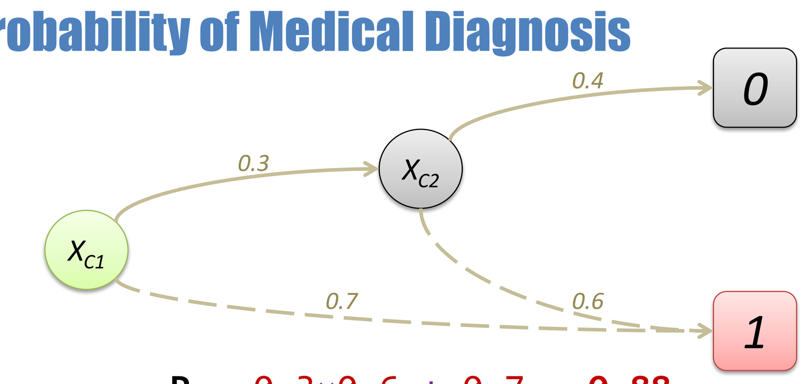

**Explanations found:**
- [ ($C_1$; {$X_1$/david}; 0) ]
- [ ($C_2$; {$X_2$/david}; 0) ]

#### Explanations of Path Problem



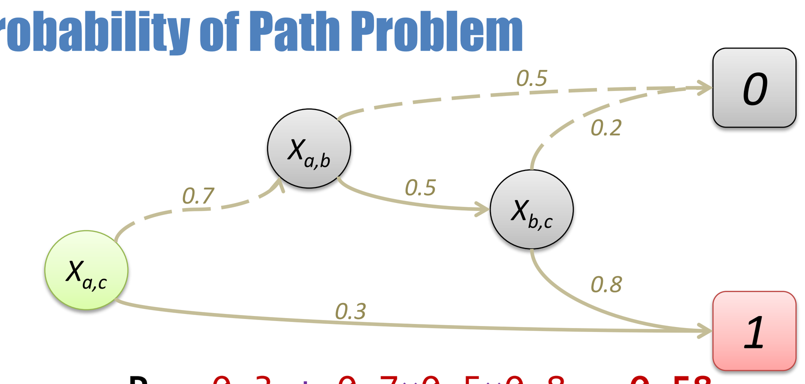

**Explanations found:**
- [ ($C_3$; {$Node_1$/c}; 0) ]
- [ ($C_4$; {$Node_2$/b}; 0), ($C_5$; {$Node_3$/c}; 0) ]

#### Probability of Medical Diagnosis




`P = 0.3×0.6 + 0.7 = 0.88`

#### Probability of Path Problem




`P = 0.3 + 0.7×0.5×0.8 = 0.58`

### Approximated Inference

In some domains, standard inference may be impossible, therefore we considered the following approximate algorithms.

#### Best K Algorithm

This is a **deterministic** algorithm based on a **branch and bound** technique.
- It builds the set of explanations incrementally by keeping only the **k** most probable explanations. (A typical value for k is **64**.)
- The chosen explanations are used to build a BDD and to compute their probability.
- Thus the algorithm provides a lower bound on the probability of the query.

**Note:** considering only a limited number of proofs allows a better control of the overall complexity of inference, which is fundamental in case of evaluation of large problems.

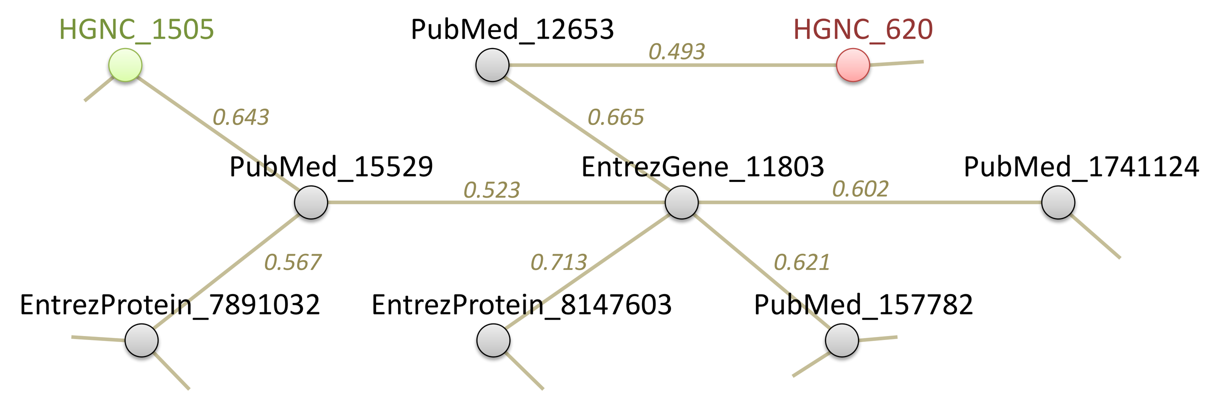
#### Monte Carlo Algorithm

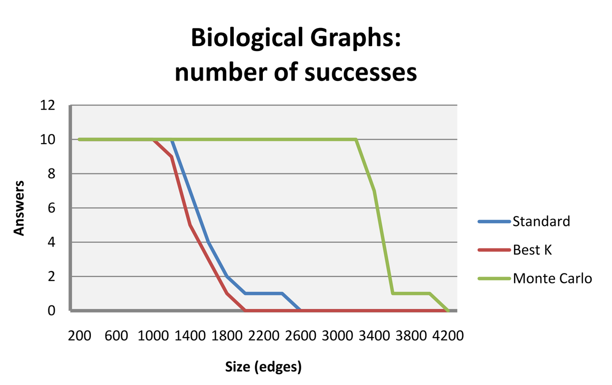
This is a **stochastic** algorithm based on a **Monte Carlo** approach.
- Instances of the problem are sampled repeatedly from the LPAD. The sampling is efficient because it only considers the clauses that are relevant for the query.
- The algorithm also uses a meta-interpreter that stochastically chooses head atoms of resolving clauses.
- The fraction of sampled instances where the query is true gives the probability of the query.

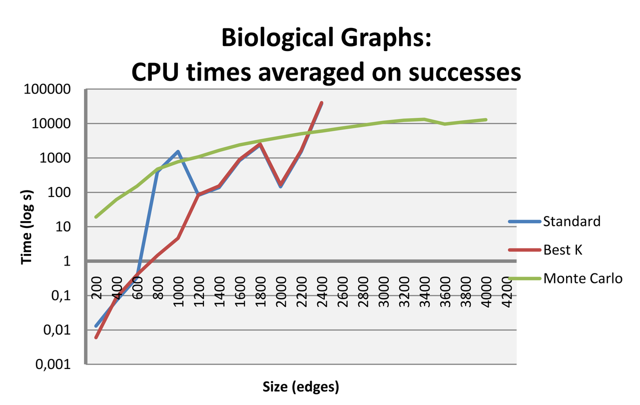
**Note:** this method does not require BDDs to accomplish the computation of the probability of the query, which may be too demanding in the evaluation of large problems.

## Experiments & Results

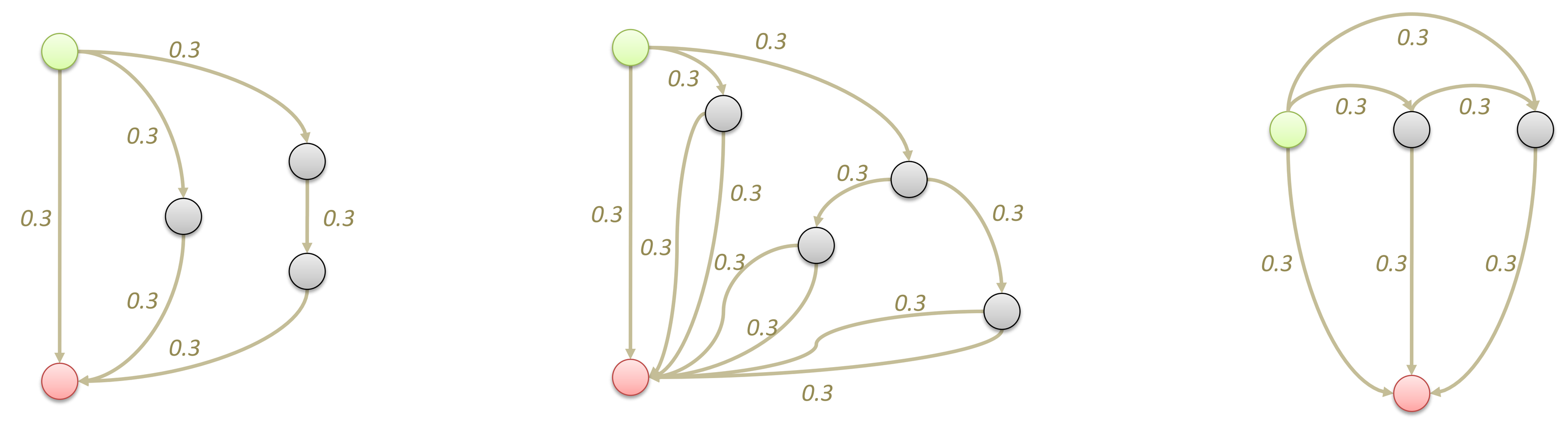
### Biological Graphs

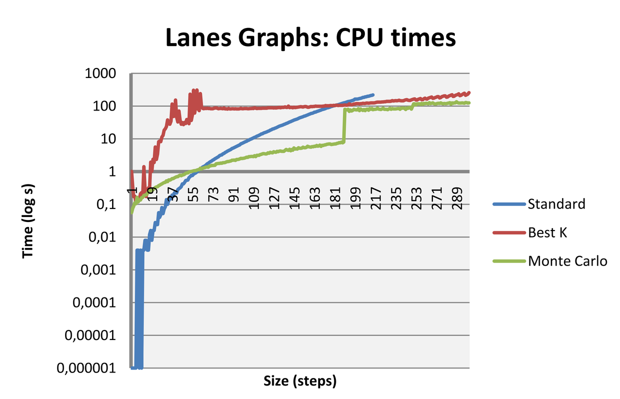


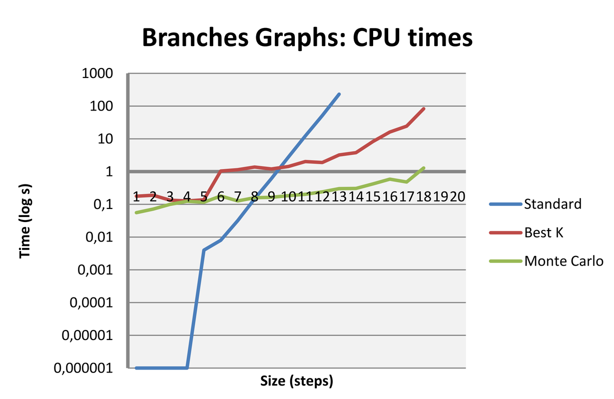



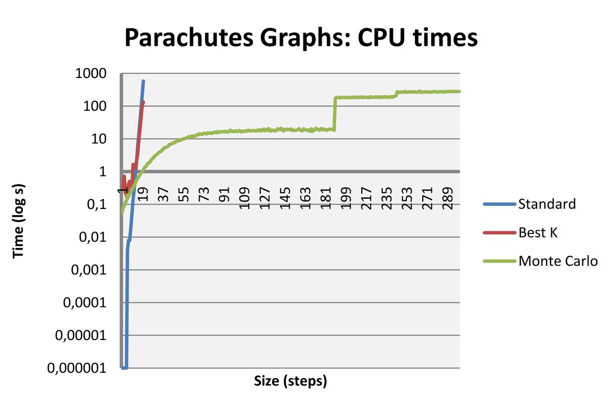




### Artificial Graphs






### Details and Notes

- **Datasets:** the first dataset is a graph that describes a real case where nodes, edges and probabilities respectively represent the biological entities responsible of Alzheimer's disease, the relationships between them and their strengths; we considered 10 samples composed by 50 subgraphs of increasing size each. The other graphs are artificial.
- **Queries:** compute the probability that a path exists between two given nodes of each graph.
- **Tests:** performed on Linux machines equipped with Intel Core 2 Duo E6550 (2333 MHz) and 4 Gb RAM with a 24 hours time limit.