DTProbLog

A Decision-Theoretic Probabilistic Prolog

Guy Van den Broeck
Ingo Thon
Martijn van Otterlo
Luc De Raedt
Motivation

• Many real-world decision problems are relational and probabilistic

  • Wildfire control
  • Who to vaccinate for swine flu
  • Viral marketing
  • ...
Motivation

- Combination of relations, uncertainty and decision theory largely ignored
  
  some exceptions are MLDNs, DTLPs and FOMDPs

- Relations with uncertainty
  
  = Statistical Relational Learning

- ProbLog is a simple probabilistic Prolog

- DTProbLog is a decision-theoretic ProbLog
Outline

- DTProbLog: the Language
- DTProbLog: the Algorithms
  - Exact Solution Algorithm
  - Approximate Solution Algorithms
- Experiments: Viral Marketing
- Related Work & Conclusions
DTProbLog: the Language
Probabilistic Facts
0.3 :: buy_trust(_,__).

Background Knowledge
buy(X) :-
    trusts(X,Y),
    buys(Y),
    buy_trust(X,Y).
Probabilistic Facts
0.3 :: buy_trust(_,_).
0.2 :: buy_marketing(_).

Background Knowledge

```
buys(X) :-
    trusts(X,Y),
    buys(Y),
    buy_trust(X,Y).
```

```
buys(X) :-
    marketed(X),
    buy_marketing(X).
```
Which strategy gives the maximum expected utility?
Probabilistic Facts

Background Knowledge

Decisions

? :: marketed(P) :- person(P).

Utility Facts

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
DTProbLog: the Algorithms
DTProbLog: the Algorithms

- DTProbLog solves **decision** problems in complex **relational** and **uncertain** environments.

- Exact solution algorithm
  - Extends ProbLog's BDD-based inference
  - Efficient datastructures: BDD and ADD

- Approximate algorithms
  - Local search
  - K-best proofs
Example:
Dressing for unpredictable weather

Decision Facts                       Probabilistic Facts
? :: umbrella.                        0.3 :: rainy.
? :: raincoat.                         0.5 :: windy.

Background Knowledge

dry :- rainy, umbrella, not(broken_umbrella).
dry :- rainy, raincoat.
dry :- not(rainy).

broken_umbrella :- umbrella, rainy, windy.

Utility Facts

umbrella => -2.                       dry => 60.
raincoat => -20.                      broken_umbrella => -40.
Exact Solution Algorithm

1. Find all proofs for each utility attribute (Prolog)
2. Binary decision diagrams from the proofs
3. Algebraic decision diagrams for the probability of each attribute
4. Algebraic decision diagrams for the utility of each attribute
5. Algebraic decision diagram for the total utility
1. Find all **proofs** for each utility attribute (Prolog)

2. Binary decision diagrams from the proofs
   - Leaf nodes indicate attribute *true* or *false*
   - Nodes are probabilistic facts or decisions

---

**Leafs indicate attribute true or false**

- **rainy**
  - **true**
  - **false**

- **umbrella**
  - **true**
  - **false**

- **windy**
  - **true**
  - **false**

- **raincoat**
  - **true**
  - **false**

---

**Dry => 60.**

**broken_umbrella => -40.**

**umbrella => -2.**

**raincoat => -20.**
Algebraic decision diagrams for the **probability** of each attribute

- Probabilistic facts are marginalized out
- Nodes are decisions only
- Leaf nodes are probabilities

- **dry** => 60.
- **broken_umbrella** => -40.
- **umbrella** => -2.
- **raincoat** => -20.
Algebraic decision diagrams for the utility of each attribute

- Leaf nodes are expected utilities

\[
\begin{align*}
dry & \Rightarrow 60. \\
\text{broken_umbrella} & \Rightarrow -40. \\
\text{umbrella} & \Rightarrow -2. \\
\text{raincoat} & \Rightarrow -20.
\end{align*}
\]
Algebraic decision diagram for the total utility

\[ \sum \]

+ sound pruning (ADD not built entirely in memory)
Approximate Solution: Local Search

1. Find all proofs for each utility attribute (Prolog)
2. Binary decision diagrams from the proofs
3. Algebraic decision diagrams for the probability of each attribute
4. Algebraic decision diagrams for the utility of each attribute
5. Algebraic decision diagram for the total utility
Approximate Solution: Local Search

1. Find all **proofs** for each utility attribute (Prolog)
2. Binary decision diagrams from the proofs
3. Algebraic decision diagrams for the **probability** of each attribute
4. Algebraic decision diagrams for the **utility** of each attribute
5. Algebraic decision diagram for the **total utility**

3. Greedy hillclimber search
Approximate Solution: K-best Proofs

1. Find all proofs for each utility attribute (Prolog)
2. Binary decision diagrams from the proofs
3. Algebraic decision diagrams for the probability of each attribute
4. Algebraic decision diagrams for the utility of each attribute
5. Algebraic decision diagram for the total utility
Approximate Solution: K-best Proofs

1. Find all *proofs* for each utility attribute (Prolog)

1. Find the *k most likely proofs* for each utility attribute
2. Binary decision diagrams from the proofs

3. Algebraic decision diagrams for the *probability* of each attribute
4. Algebraic decision diagrams for the *utility* of each attribute

5. Algebraic decision diagram for the *total utility*
Experiments: Viral Marketing
Experiments: Viral Marketing

- Synthetic dataset
  Random power law graphs of increasing size

- Real-world Epinions dataset [Domingos02]
(Q1) Does the **exact** solution algorithm perform better than **naively** calculating the utility of all strategies?
(Q2) What is the difference in runtime and solution quality between **exact** and **local** search?
(Q3) What is the difference in runtime and solution quality between for different values of $k$ in $k$-best.

![Graph showing runtime and solution quality comparison between different methods.](graph.png)
(Q4) Do the algorithms scale?

Epinions social network

- 75,000 people
- 500,000 trust relations

YES

- Local search
- 17-best proofs
- Solved in 16 hours
Related Work & Conclusions
## Related Work

<table>
<thead>
<tr>
<th>Representation</th>
<th>Solution</th>
<th>Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relational</td>
<td>Probabilities</td>
<td>Global optimum</td>
</tr>
<tr>
<td>Influence Diagrams</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>MLDNs [Nath]</td>
<td>✔️</td>
<td>❌</td>
</tr>
<tr>
<td>ICL [Poole]</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>DTLPs [Chen]</td>
<td>✔️</td>
<td>✔️</td>
</tr>
<tr>
<td>DTProbLog</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Summary

- DTProbLog, the programming language
  - Probabilistic Prolog
  - Decisions
  - Utilities: rewards or costs attached to goals
- Solution algorithms
  - Exactly
  - Approximately
- Experiments
  - Effective
  - Scale well
Ongoing and Future Work

- Sequential decision problems
  - Easy to represented in DTProbLog
  - Bad fit for solution algorithms
- Solvers
  - Integer linear programming
  - Bounded approximation
  - Monte-Carlo
- Lifting (many BDDs have same structure)
- Learning DTProbLog programs
- Inverse reinforcement learning
Thank You!
Decisions

? :: marketed(P) :- person(P).

Probabilistic Facts

0.3 :: buy_trust(_,_).
0.2 :: buy_marketing(_).

Background Knowledge

buys(X) :-
    trusts(X,Y),
    buys(Y),
    buy_trust(X,Y).

Utility Facts

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).