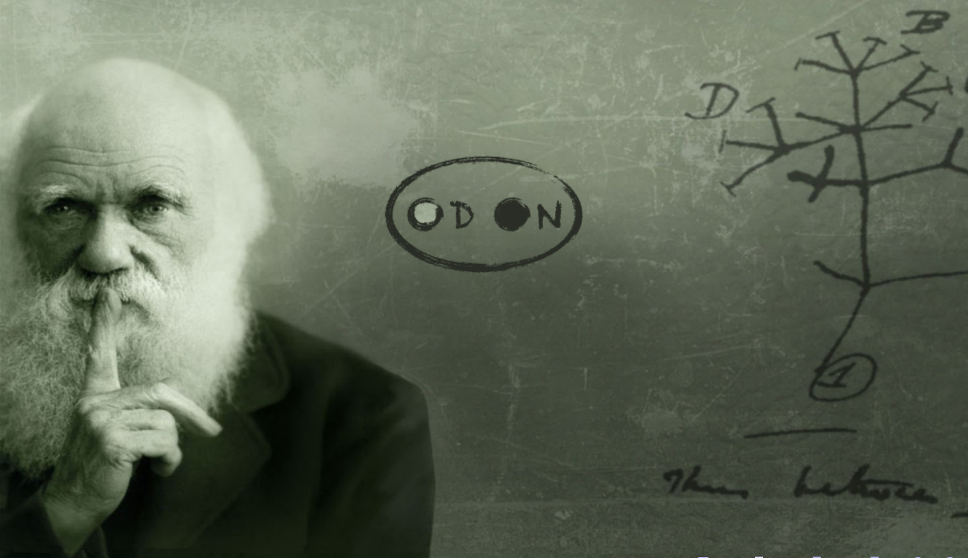
INTRODUCING

DARWINIAN NETWORKS



Introducing Darwinian Networks

Cory J. Butz   Jhonatan S. Oliveira   André E. dos Santos

Department of Computer Science
University of Regina
Regina, S4S 0A2, Canada

www.darwiniannetworks.com

28th FLAIRS Conference
Hollywood, Florida, USA
May 18th, 2015

Outline of the Presentation

1. Motivation

2. Darwinian Networks

    2.1 Inference
    - Variable Elimination
    - Arc-Reversal
    - Lazy Propagation

    2.2 Modeling (Testing Independencies)
    - m-Separation
    - d-Separation

3. Advantages

4. Conclusions

1. Motivation

Understanding *Bayesian network* (BN) inference is not easy.

We sought a *purely graphical approach* for BN inference.

The representation took on a biological feel *(Darwinian Networks)*.

We then observed that Darwinian Networks could represent, simplify, and speed-up the testing of independencies.

And determine good elimination orderings.

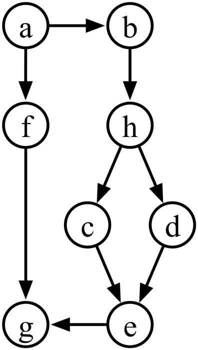
Surprisingly simple, remarkably robust.

Bayesian Networks

A *Bayesian Network* (BN) consists of:

- a *directed acyclic graph* (DAG),
- a matching set of *conditional probability tables* (CPTs).

Example: BN



$$P(U) = P(a) \cdot P(b|a) \cdot P(c|h) \cdot P(d|h) \cdot P(e|c,d) \cdots P(g|e,f)$$

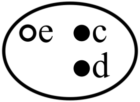
2. Darwinian Networks (DNs)

A CPT $P(X|Y)$ is represented as a *population* $p(X, Y)$.

The variables in the LHS X are *white*.

The variables in the RHS Y are *black*.

$$P(e|c,d) \Longrightarrow$$

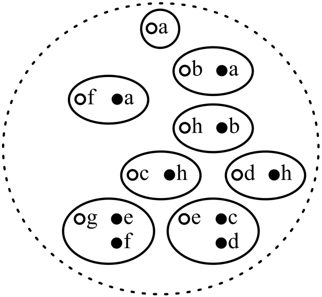


CPT $P(e|c,d)$ is depicted as population $p(e, cd)$.

2. Darwinian Networks

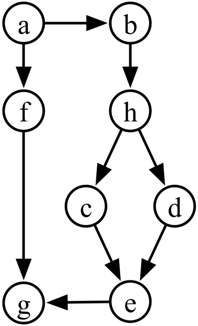
A *Darwinian Network* (DN) is a finite, multiset of populations.

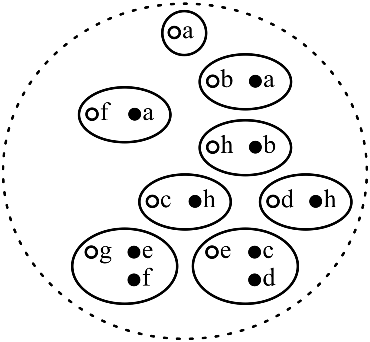
A DN is depicted by a dashed closed curve around its populations.



Populations:
$p(a)$, $p(b, a)$, $p(c, h)$, $p(d, h)$, $p(e, cd)$, $p(f, a)$, $p(h, b)$, $p(g, ef)$

Every BN can be represented as a DN





2.1 Inference

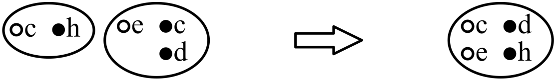
BN inference is called *evolution* in DNs.

We first introduce operations on populations corresponding to:

- multiplication,
- division,
- marginalization.

Multiplication is the Merge of Populations

$$P(c|h) \cdot P(e|c,d) = P(c,e|d,h)$$



○ white	+	● black	=	○ white
● black	+	○ white	=	○ white
● black	+	● black	=	● black
○ white	+	○ white	=	● black

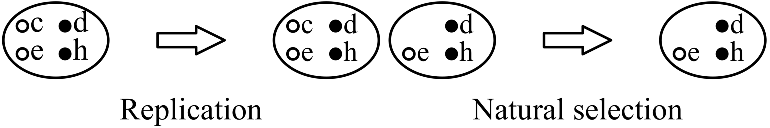
Merge also represents Division

$$P(a, b) \; / \; P(b) \quad = \quad P(a|b)$$

○ white	+	● black	=	○ white
● black	+	○ white	=	○ white
● black	+	● black	=	● black
○ white	+	○ white	=	● black

Marginalization is Replication then Natural Selection

$$\sum_c P(c, e|d, h) \quad = \quad P(e|d, h)$$



DNs can represent BN Inference Algorithms

As DNs can represent multiplication, division, and marginalization, it follows that DNs can represent exact inference algorithms, including:

- Variable Elimination (VE) (Zhang and Poole, 1994),
- Arc-Reversal (AR) (Olmsted, 1983),
- Lazy Propagation (LP) (Madsen and Jensen, 1999).

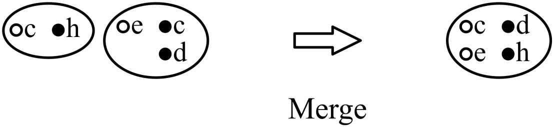
Variable Elimination (VE)

To answer $P(e|b)$, VE computes:

$$\begin{aligned}
P(c, e|d, h) &= P(c|h) \cdot P(e|c, d), && (1)\\
P(e|d, h) &= \sum_c P(c, e|d, h), && (2)\\
P(d, e|h) &= P(d|h) \cdot P(e|d, h), && (3)\\
P(e|h) &= \sum_d P(d, e|h), && (4)\\
P(e, h|b) &= P(h|b) \cdot P(e|h), && (5)\\
P(e|b) &= \sum_h P(e, h|b). && (6)
\end{aligned}$$

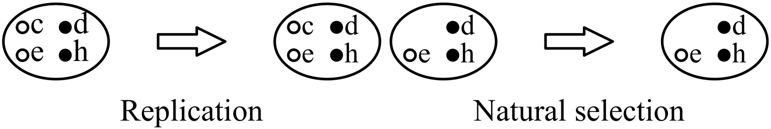
Example: Representing VE as DN Evolution

$$P(c|h) \cdot P(e|c,d) \quad = \quad P(c,e|d,h) \tag{1}$$



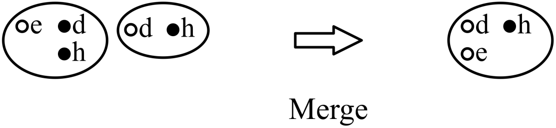
Example: Representing VE as DN Evolution

$$\sum_c P(c, e|d, h) \quad = \quad P(e|d, h) \tag{2}$$



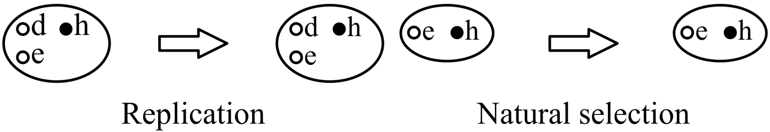
Example: Representing VE as DN Evolution

$$P(e|d,h) \cdot P(d|h) \quad = \quad P(d,e|h) \tag{3}$$



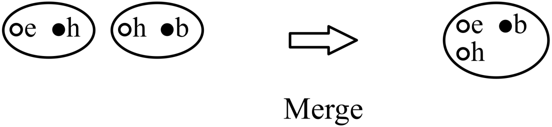
Example: Representing VE as DN Evolution

$$\sum_{d} P(d, e|h) \quad = \quad P(e|h) \qquad (4)$$



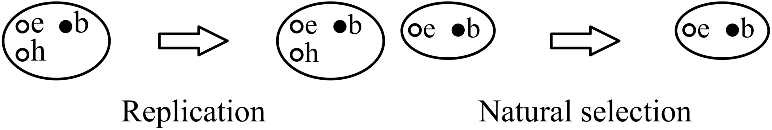
Example: Representing VE as DN Evolution

$$P(e|h) \cdot P(h|b) \quad = \quad P(e, h|b) \tag{5}$$



Example: Representing VE as DN Evolution

$$\sum_h P(e, h|b) = P(e|b) \quad (6)$$



Query $P(e|b)$ is represented as population $p(e, b)$.

DN Advantage

Koller and Friedman (2009) introduce readers to BN inference using VE.

There is a one-to-one correspondence between VE's mathematical equations and the DN illustrations.

Hence, one can learn VE without involving a single equation.

DNs can represent Arc-Reversal (AR)

AR eliminates a variable v_i by reversing the arc (v_i, v_j) between v_i and each child v_j of v_i.

$$\begin{aligned} P(v_i, v_j | P_i P_j) &= P(v_i|P_i) \cdot P(v_j|P_j), \\ P(v_j|P_i P_j) &= \sum_{v_i} P(v_i, v_j|P_i P_j), \\ P(v_i|P_i P_j v_j) &= \frac{P(v_i, v_j|P_i P_j)}{P(v_j|P_i P_j)}. \end{aligned}$$

AR only involves multiplication, division, and marginalization.

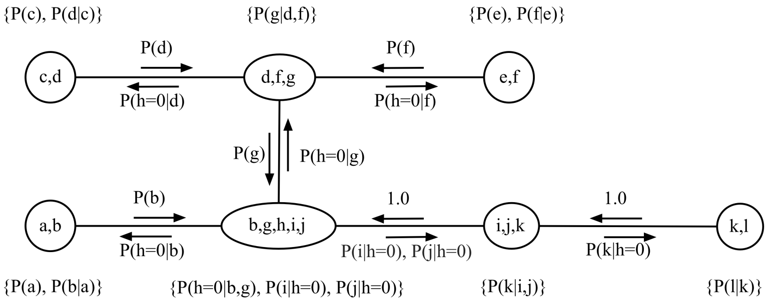
Lazy Propagation (LP)

BN variables are clustered into nodes, organized as a join tree.

Each BN CPT is assigned to a join tree node.

Messages are propagated systematically.

LP only involves multiplication, division, and marginalization.

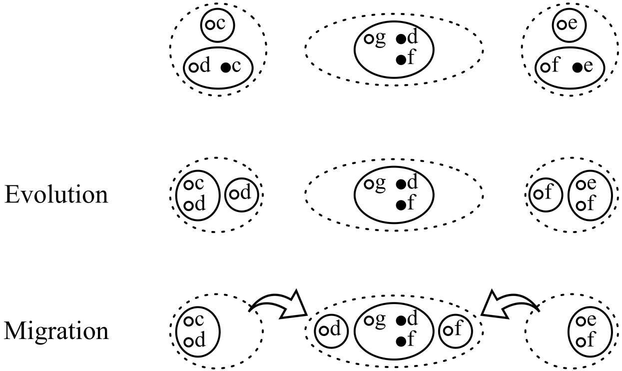


DNs can represent LP

A join tree is represented as a set of DNs.

Each join tree node is represented as one DN.

A propagated message from a join tree node to another is viewed as a population *migrating* from one DN to another.



2.2 Modeling - Testing Independencies in BNs

m-Separation (Lauritzen et al., 1990; Zhang and Poole, 1994) tests $I(X, Y, Z)$ in an undirected graph with four steps:

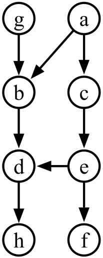
(i) construct the sub-DAG onto $XYZ \cup An(XYZ)$;

(ii) construct the *moralization* by adding an undirected edge between each pair of parents of a common child and then dropping directionality;

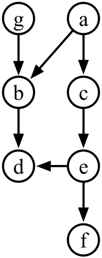
(iii) delete Y and its incident edges;

(iv) if there exists a path from X to Z, then $I(X, Y, Z)$ does not hold; otherwise, $I(X, Y, Z)$ holds.

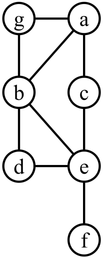
Example: m-Separation $I(a, d, f)$



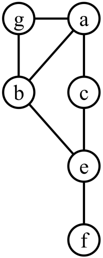
DAG



sub-DAG



moralization



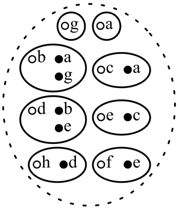
deletion
and
check for path

DNs can represent m-Separation

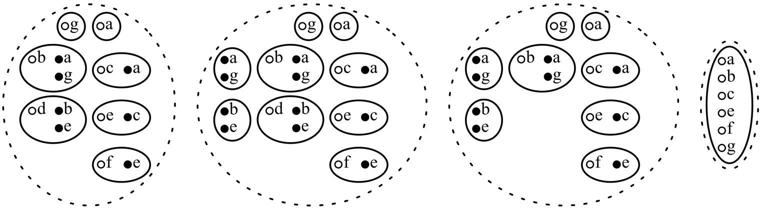
Testing independencies in BNs is represented as testing *adaptation* in DNs.

We observe how populations *adapt* to the removal of other populations.

Adaptation: Testing Independence $I(a, d, f)$ in DNs



(i) (ii) (iii) (iv)



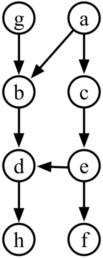
sub-DAG moralization deletion check for path

DNs Simplify m-Separation

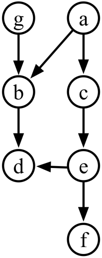
Moralization of m-separation can be excessive.

Adding edge (b, e) is necessary, since d will be deleted.

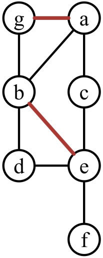
Adding edge (a, g) is unnecessary, since b will not be deleted.



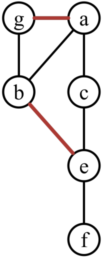
DAG



sub-DAG



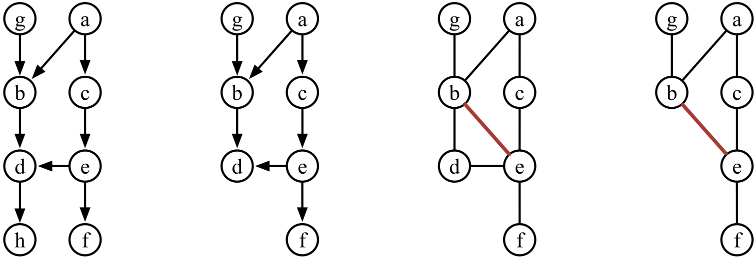
moralization



deletion
and
check for path

DN Contribution (Rationalization)

When testing $I(X, Y, Z)$, add an undirected edge between variables with a common child only when the child is in Y.



DAG

sub-DAG

rationalization

deletion
and
check for path

Modeling - Testing Independencies with d-Separation

Algorithm 3.1 Find nodes reachable from X given Y via active paths in DAG $\mathcal{B}$

```
procedure REACHABLE(X,Y,B)
    ▷ Phase I: insert Y and all ancestors of Y into A
    An(Y) ← ANCESTORS(Y, B)
    A ← An(Y) ∪ Y
    ▷ Phase II: traverse active paths starting from X
    for v ∈ X do                 ▷ (Node,direction) to be visited
        L ← L ∪ {(↑, v)}
    V ← ∅                        ▷ (Node,direction) marked as visited
    R ← ∅                        ▷ Nodes reachable via active path
    while L ≠ ∅ do               ▷ While variables to be checked
        Select (d, v) in L
        L ← L − {(d, v)}
        if (d, v) ∉ V then
            if v ∉ Y then
                R ← R ∪ {v}                ▷ v is reachable
            V ← V ∪ {(d, v)}  ▷ Mark (d, v) as visited
            if d =↑ and v ∉ Y then
                for v_i ∈ Pa(v) do
                    L ← L ∪ {(↑, v_i)}
                for v_i ∈ Ch(v) do
                    L ← L ∪ {(↓, v_i)}
            else if d =↓ then
                if v ∉ Y then
                    for v_i ∈ Ch(v) do
                        L ← L ∪ {(↓, v_i)}
                if v ∈ A then
                    for v_i ∈ Pa(v) do
                        L ← L ∪ {(↑, v_i)}
    return R
```

Geiger et al. (1989) provide a linear time complexity algorithm for implementing d-separation.

Implementation of d-separation given by Koller and Friedman (2009).

DNs Simplify d-Separation

The main idea is to start from X, follow active paths, and see if it reaches Z.

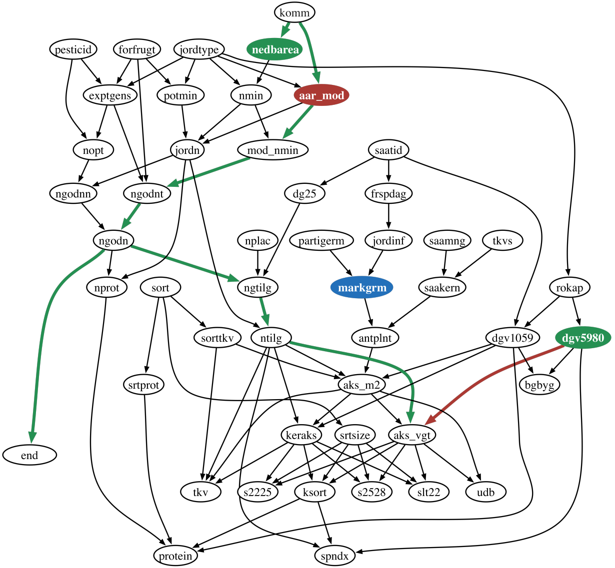
The linear implementation of d-separation considers all active paths until they become blocked.

The DN improvement is the identification of a class of active paths that are doomed to become blocked.

Thus, there is no benefit to exploring these paths.

Example: Testing $I(nedbarea, markgrm, dgv5980)$ in Barley

Any path through *aar_mod* will eventually become blocked.



DN Contribution

Stop traversing active paths that are doomed to become blocked.

BN	$\|N\|$	d-Sep Tests	i-Sep Tests	Test Savings	Time Savings
Insurance	27	58898	36642	38%	-22%
Water	32	41392	23959	42%	-27%
Alarm	37	35224	23078	34%	-11%
Barley	48	78794	56804	28%	-15%
Hailfinder	56	51922	42543	18%	-23%
Pathfinder	135	125932	62820	50%	-79%
Munin1	186	167809	64329	62%	14%
Diabetes	413	827291	681468	18%	11%
Pigs	441	116795	12841	89%	36%
Link	724	336780	75505	78%	38%
Munin4	1038	509299	77314	85%	47%
Munin3	1041	459409	50147	89%	55%

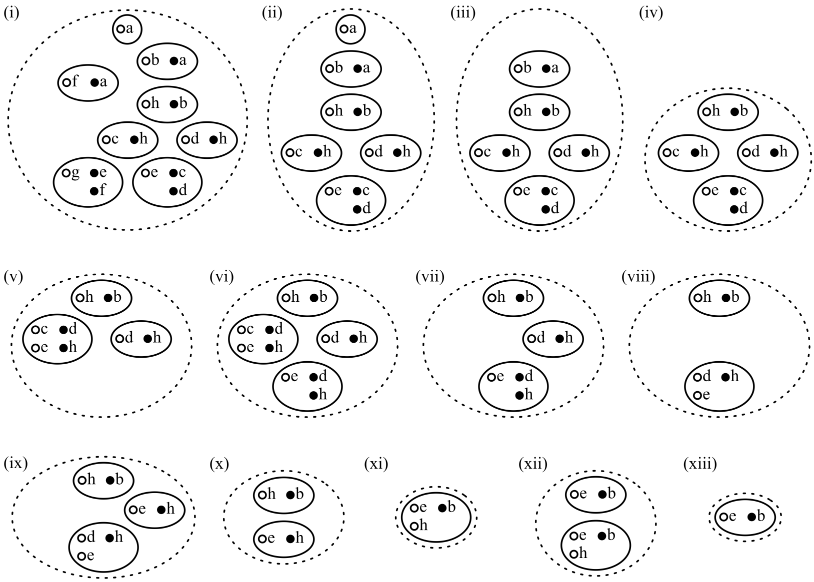
3. Advantages - Surprisingly Simple, Remarkably Robust

- In modeling, d-separation can use specialized terminology not referenced in inference such as "open sequential valves" and "closed divergent valves."
- In inference, LP involves specialized terminology not referenced in modeling such as the "running intersection property."
- DNs use the same terminology for inference and modeling.

Representing All Steps of LP in DNs

- LP involves 2 networks.
- LP tests independencies in a BN, yet conducts inference in a JT.
- DNs do both in the same network.

All Steps of VE: BN, query, independencies, computation



Determining Good Elimination Orderings

- The order in which variables are eliminated can have a profound impact on the amount of computation performed.

- DNs can represent four well-known heuristics for determining good elimination orderings in BNs:
  - *min-neighbours* (MN),
  - *min-weight* (MW),
  - *min-fill* (MF),
  - *weighted-min-fill* (WMF).

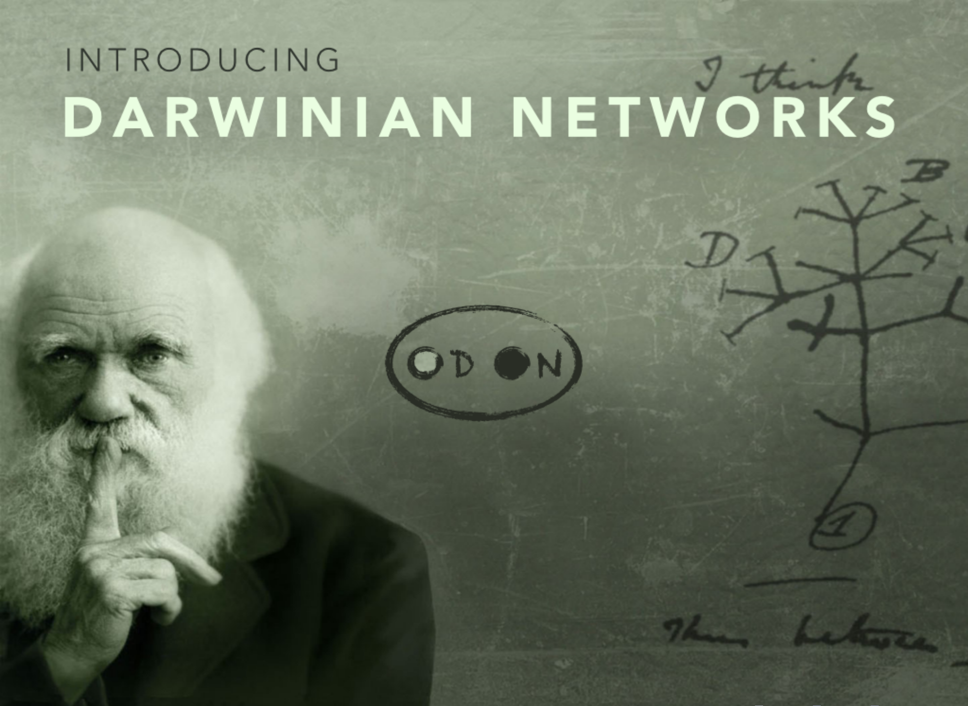
- We introduced a new heuristic, called *potential energy* (PE), based on DNs.

- PE can score more accurately than the above heuristics.

4. Conclusion

- Purely graphical approach to VE, which is often used to introduce BN inference to beginners
- Faster way to test independencies in BNs
- Unify modeling and inference into one network using common terminology
- DNs are like looking at BNs through a microscope

Watch www.darwiniannetworks.com for updates



Kilroy, Jeff. Darwin "I think..." 2009. Kilrizzy's deviantART gallery. deviantART. 2015. Web. 23 Apr. 2015.