Alternatives to Threshold-Based Desire Selection in Bayesian BDI Agents

Bernardo Luz[1] **Felipe Meneguzzi**[2] Rosa Vicari[1]

[1]Federal University of Rio Grande do Sul
bernardo.luz@inf.ufrgs.br
rosa@inf.ufrgs.br

[2]Pontifical Catholic University of Rio Grande do Sul
felipe.meneguzzi@pucrs.br

EMAS 2013

Outline

Motivation

- Traditional BDI agents
  - Beliefs expressed as a closed set of ground literals
  - Logic conditions in desire selection (or plan selection)
  - Do not usually reason about the world under uncertainty

Motivation

- Traditional BDI agents
  - Beliefs expressed as a closed set of ground literals
  - Logic conditions in desire selection (or plan selection)
  - Do not usually reason about the world under uncertainty
- Bayesian BDI agents
  - Handle uncertainty by representing beliefs as a bayesian network
    - Fagundes *et al.*, 2007
    - Kieling and Vicari, 2011
    - Carrera and Iglesias, 2012
  - Need to select desires with uncertainty
    - Threshold criterion
    - Alternatives

From Logic to Probability

- First-order logic approaches insufficient to represent uncertainty about statements
- To address these limitations, probability theory is often used, due to:
  - Cost of representing all possible combinations of truth values
  - Lack of a complete theory of the domain in question
  - Possible inviability of performing all necessary tests to ascertain complete truth for certain statements
- Known and unknown information is represented
- Estimates are possible when there is incomplete information

Bayesian Networks

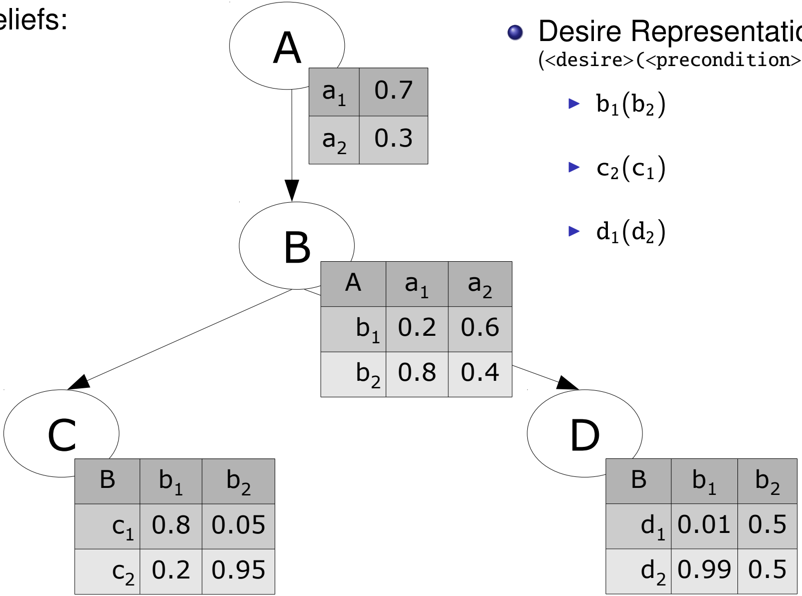
- Bayesian Networks (BN): compact representation of a *joint probability distribution* for conditionally dependent events
- BNs are represented as graphs expressing how parts of the information are conditioned on others
- Efficient algorithms for belief updates given evidence, taking into consideration
  - Conditional and unconditional dependencies
  - Graph connectivity and evidence propagation

Bayesian networks within BDI

- Belief base: entire bayesian network
- Desires have preconditioning beliefs, following the tradition of implemented BDI systems
- Desire satisfaction evaluation
  - Strong desires: probability of the desire itself equal to 1
  - Weak desires: probability of the desire itself equal to or greater than a predefined value
- Desire selection
  - Threshold criterion
  - Precondition evaluation not based on validity, but on confidence

Bayesian networks within BDI: abstract example

- Beliefs:



a_1	0.7
a_2	0.3

A	a_1	a_2
b_1	0.2	0.6
b_2	0.8	0.4

B	b_1	b_2
c_1	0.8	0.05
c_2	0.2	0.95

B	b_1	b_2
d_1	0.01	0.5
d_2	0.99	0.5

- Desire Representation:
  (<desire>(<precondition>))
  - $b_1(b_2)$
  - $c_2(c_1)$
  - $d_1(d_2)$

Bayesian networks within BDI: abstract example

- Beliefs:

A

a1	0.7
a2	0.3

B

A	a1	a2
b1	0.2	0.6
b2	0.8	0.4

C

B	b1	b2
c1	0.8	0.05
c2	0.2	0.95

D

B	b1	b2
d1	0.01	0.5
d2	0.99	0.5

- Desire Representation:
  (<desire>(<precondition>))
  Working scenario:
  - hard evidence that $A = a_1$
    - $\mathtt{b}_1(\mathtt{b}_2)$
      $P(b_2|a_1) = 0.8$
    - $\mathtt{c}_2(\mathtt{c}_1)$
      $P(c_1|a_1) = 0.2$
    - $\mathtt{d}_1(\mathtt{d}_2)$
      $P(d_2|a_1) = 0.598$

Threshold-based selection

```
function ThresholdBasedSelection(threshold,desires)
    for each desire such that desire ∈ desires do
        if desire.preCondition.probability ≥ threshold then
            desires.remove(desire)
            return desire
        end if
    end for
    return null
end function
```

Threshold-based selection

```
function ThresholdBasedSelection(threshold, desires)
    for each desire such that desire ∈ desires do
        if desire.preCondition.probability ≥ threshold then
            desires.remove(desire)
            return desire
        end if
    end for
    return null
end function
```

Example agent scenario

Threshold: 0.75

- $b_1(b_2)$: 0.8
- ~~$c_2(c_1)$: 0.2~~
- ~~$d_1(d_2)$: 0.598~~

- Some desires are never selected
- Given uncertainty, should we ignore low-probability desires? (Conservatism vs pro-activeness)

Alternatives ⇨

Alternative Desire Selection Algorithms

- Probability Ranking
- Biased Lottery
- Multi-Desire Biased Random Selection

Probability Ranking

```
function ProbabilityRankingSelection(desires)
    if desires.length > 0 then
        rankedDesires ← desires ordered by precondition probability
        desire ← rankedDesires.first()
        if desire.preCondition.probability > 0 then
            desires.remove(desire)
            return desire
        end if
    end if
    return null
end function
```

Probability Ranking

```
function PROBABILITYRANKINGSELECTION(desires)
    if desires.length > 0 then
        rankedDesires ← desires ordered by precondition probability
        desire ← rankedDesires.first()
        if desire.preCondition.probability > 0 then
            desires.remove(desire)
            return desire
        end if
    end if
    return null
end function
```

Example agent scenario

Ranking:

1. $b_1(b_2)$: 0.8
2. $d_1(d_2)$: 0.598
3. $c_2(c_1)$: 0.2

- Prioritizes desires according to precondition probability
- Does not account for frequencies or relative proportions among desire precondition probabilities

Biased Lottery

```
function BiasedLottery(desires)
    randomValue ← random number ∈ [0, 1]
    intervals ← GenerateIntervals(desires)
    for i ← 0 to intervals.length do
        if randomValue < intervals[i] then
            desire ← desires[i]
            desires.remove(desire)
            return desire
        end if
    end for
    return null
end function
```

GenerateIntervals generates:

- for each desire, a numeric interval proportional to the precondition probability of other desires
- intervals added to a list in ascending order
- intervals are normalized desire preconditions probability sum > 1

Biased Lottery

Original Probs.

$b_1(b_2)$: 0.8
$c_2(c_1)$: 0.2
$d_1(d_2)$: 0.598

Intervals Generated

Desire	Selection probability	Numeric interval
$b_1(b_2)$	0.5006	$[0.0, 0.5006)$
$c_2(c_1)$	0.1252	$[0.5006, 0.6258)$
$d_1(d_2)$	0.3742	$[0.6258, 1.0]$

- “Lottery” to select **one desire** based on precondition probability
- Explicit inter-desire competition
  (once one desire is randomly selected the others wait)
- Tries to emulate precondition frequency for the “lottery” by normalizing when sum > 1

Multi-Desire Biased Random Selection

```
function MultiDesireBiasedRandomSelection(desires)
    selectedDesires ← {}
    for each desire ∈ desires do
        randomValue ← random number ∈ [0, 1]
        if randomValue ≤ desire.preCondition.probability then
            selectedDesires.add(desire)
            desires.remove(desire)
        end if
    end for
    return selectedDesires
end function
```

Multi-Desire Biased Random Selection

Example agent scenario

Desire	Selection probability	Numeric interval
$b_1(b_2)$	0.8	$[0.0, 0.8]$
$c_2(c_1)$	0.2	$[0.0, 0.2]$
$d_1(d_2)$	0.598	$[0.0, 0.598]$

- Desires considered independently of one another (no competition)
- Multiple desires can be selected at a time
- Ignores conflicts among desires

Conclusions and Future Work

- Ignoring "probabilistically irrelevant" desires not necessarily rational
  - Environment exploration
- Desire conflicts not considered at this point
  - Possible solution: combine Biased Lottery and Multi-Desire Biased Random Selection
    - ★ Biased Lottery for conflicting desires
    - ★ Multi-Desire Biased Random Selection for the rest
- Future Work:
  - Develop selection mechanisms to cope with conflicts
  - Integrate with agent programming languages
  - Implement larger scale experiments

Questions?