PUCKER

Algoritmo de Apoio à decisão ao jogador de Pôquer No Limit Texas Hold'em

Alexandre Marangoni Costa

Marcus Poggi

PUCKER

- Framework
- Simulação
- Algoritmos
- Relatórios
- Test-Driven Development

Histórico

- Projeto Orientado
- Projeto Final I
- Projeto Final II

Arquitetura

- Ruby + Java = JRuby
- Test framework: RSpec
- SBN: Simple Bayesian Networks
- University of Alberta

Pôquer

- No Limit Texas Hold'em
- Mão de 2 cartas
- Flop/Turn/River
- Fold/Check/Raise
- Simplificações

Pôquer

1. Carta alta: vence o jogador com a carta mais alta
2. Par: duas cartas de mesmo número, e outras 3 cartas sem valor
3. Dois pares: dois conjuntos de duas cartas de mesmo número, e outra carta
4. Trinca: três cartas de mesmo número, e outras duas
5. Sequência: cinco cartas de numeração consecutiva.
6. Flush: cinco cartas do mesmo naipe
7. Full House: uma trinca e um par
8. Quadra: quatro cartas de mesma numeração, e um outra
9. Sequência de mesmo naipe
10. Sequência real de mesmo naipe: sequência do 10 ao Ás, do mesmo naipe

Pôquer

Seu jogo (mão + mesa) é forte?

Os outros jogadores estão fortes?

Eu posso ganhar essa rodada, baseado na possível força dos outros jogadores, comparado com a força do meu jogo?

Eu posso ganhar essa rodada apostando forte para todos os outros jogadores desistirem?

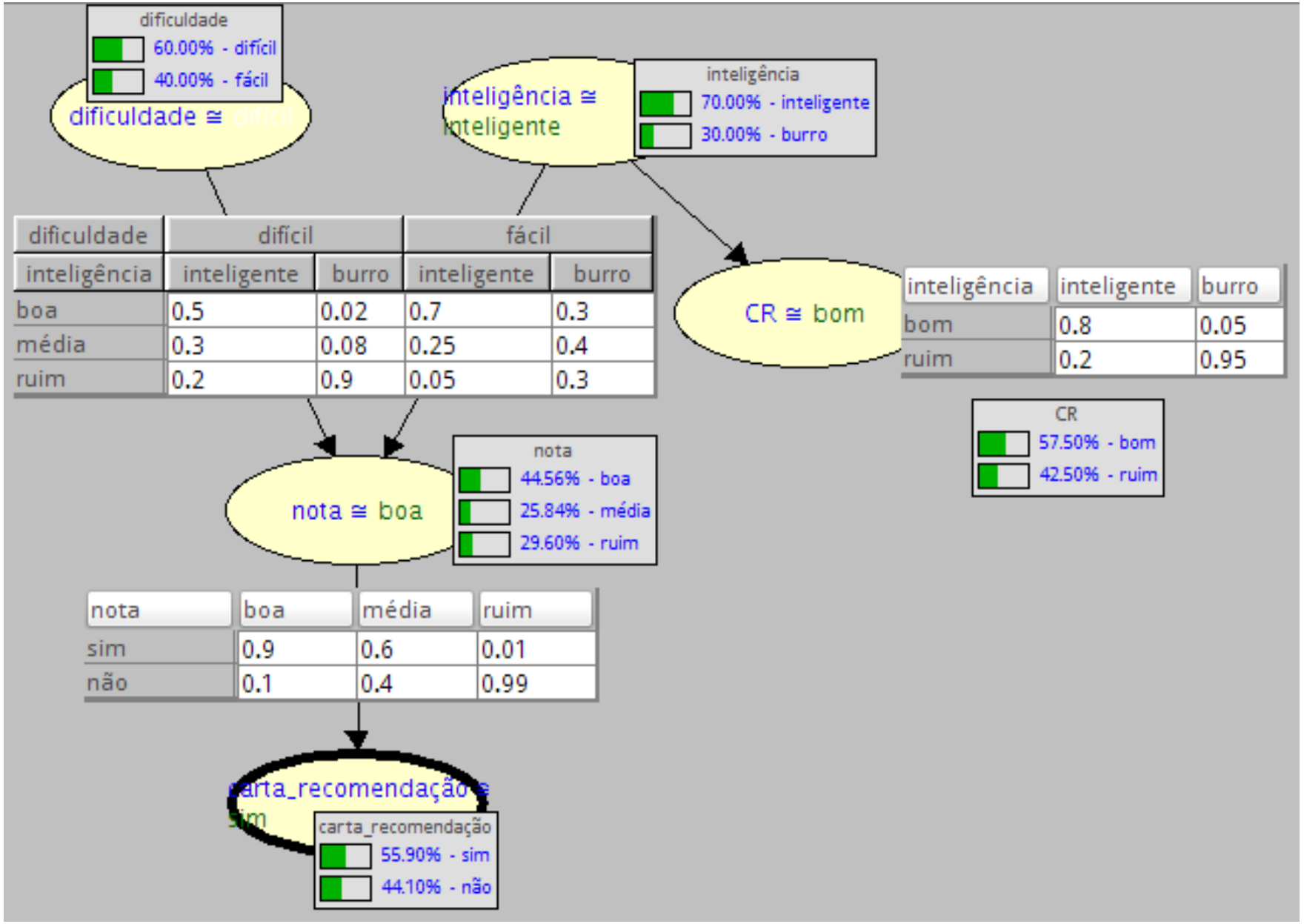
Se a mesa está forte, isso me favorece, mas também favorece os outros jogadores.

Bayesian Networks

Wikipedia

"Bayesian Network, also known as probabilistic directed acyclic graphical model, is a **probabilistic graphical model** (a type of statistical model) that represents a set of **random variables** and their **conditional dependencies** via a directed acyclic graph (**DAG**)."

Bayesian Networks

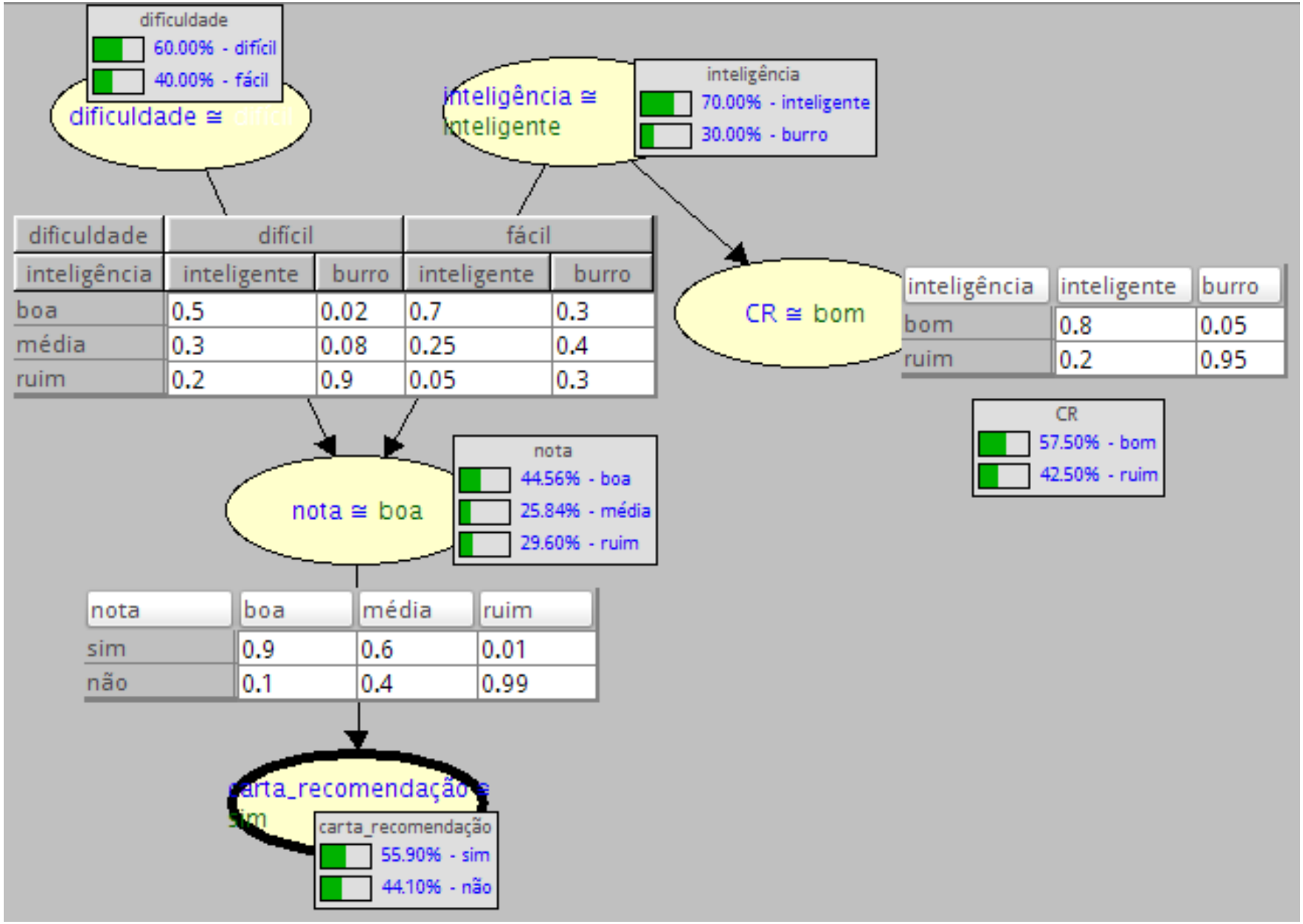


dificuldade	difícil		fácil	
inteligência	inteligente	burro	inteligente	burro
boa	0.5	0.02	0.7	0.3
média	0.3	0.08	0.25	0.4
ruim	0.2	0.9	0.05	0.3

inteligência	inteligente	burro
bom	0.8	0.05
ruim	0.2	0.95

nota	boa	média	ruim
sim	0.9	0.6	0.01
não	0.1	0.4	0.99

Bom CR: 57.5%

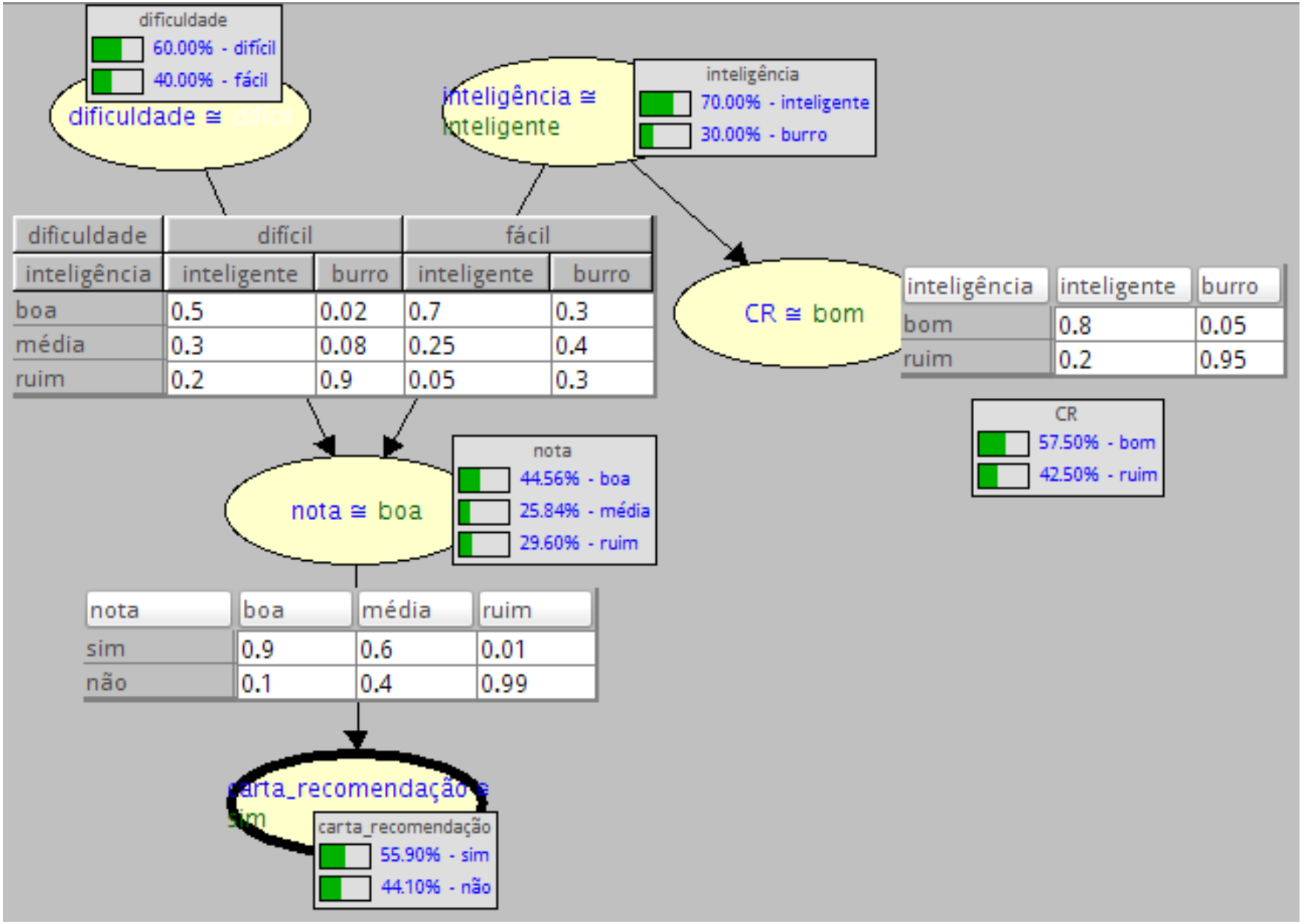


dificuldade	difícil		fácil	
inteligência	inteligente	burro	inteligente	burro
boa	0.5	0.02	0.7	0.3
média	0.3	0.08	0.25	0.4
ruim	0.2	0.9	0.05	0.3

inteligência	inteligente	burro
bom	0.8	0.05
ruim	0.2	0.95

nota	boa	média	ruim
sim	0.9	0.6	0.01
não	0.1	0.4	0.99

Carta de Recomendação: 55.9%



dificuldade	difícil		fácil	
inteligência	inteligente	burro	inteligente	burro
boa	0.5	0.02	0.7	0.3
média	0.3	0.08	0.25	0.4
ruim	0.2	0.9	0.05	0.3

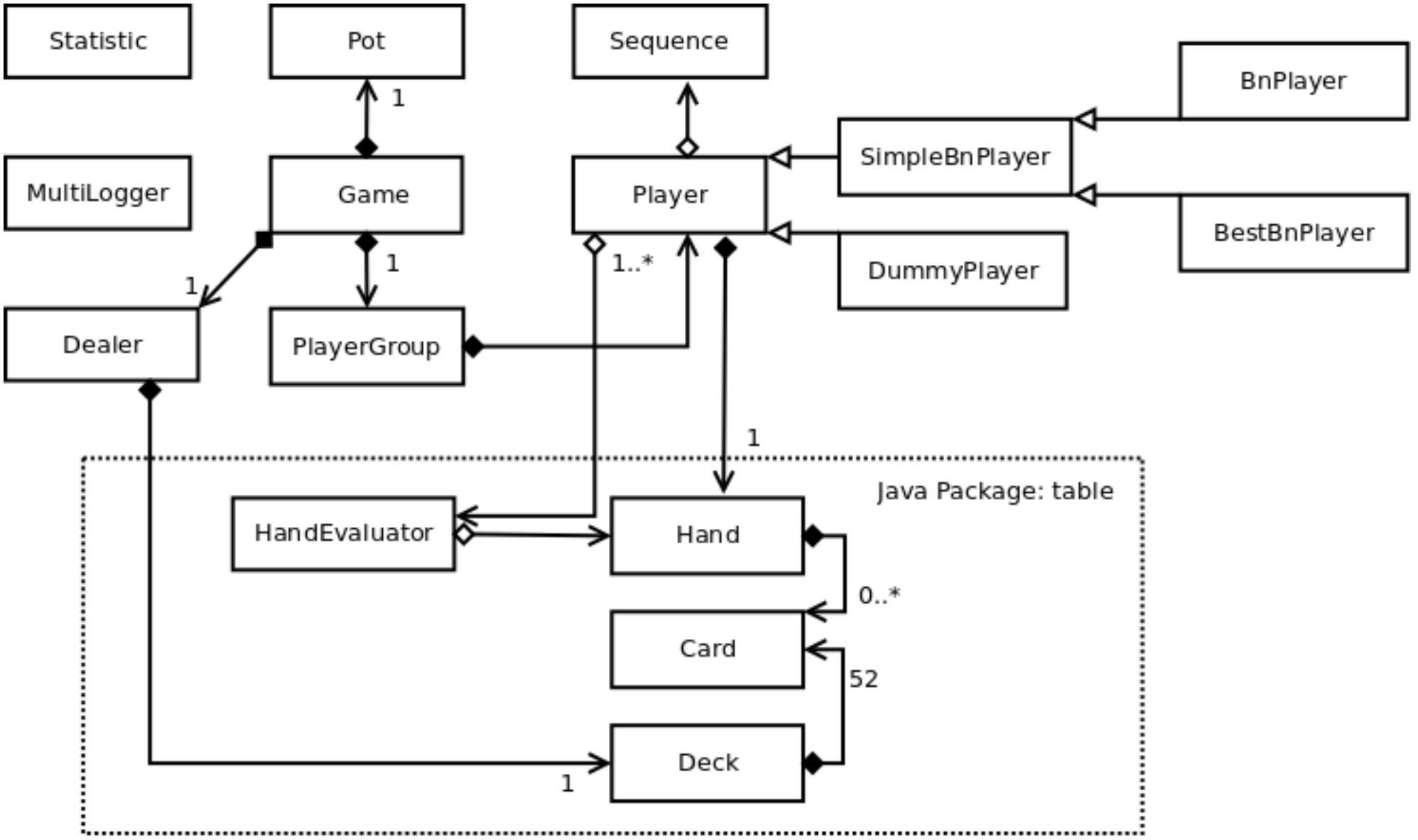
inteligência	inteligente	burro
bom	0.8	0.05
ruim	0.2	0.95

nota	boa	média	ruim
sim	0.9	0.6	0.01
não	0.1	0.4	0.99

Objetivos

1. Simulação: Player
2. Algoritmo Aleatório: Dummy Player
3. BN1: SimpleBnPlayer
4. BN2: BnPlayer
5. BN3: BestBnPlayer

Arquitetura + Objetivos



Arquitetura: Game

```
g = Game.new

g.play

100.times { g.play }
```

Arquitetura: Game

- setup_game
- collect_blinds
- deal_flop
- deal_table_card

```
def setup_game
  table_cards.clear
  dealer.reset
  pot.reset
  prepare_players
end
```

```
def collect_blinds
  players.each do |p|
    amount = p.get_from_stack(BIG_BLIND)
    pot.add_bet(p, amount)
  end
end
```

```
def deal_flop
  3.times do deal_table_card end
end

def deal_table_card
  table_cards << dealer.deal
end
```

Arquitetura: PlayerGroup

- Container
- Delegator

```
players = Pucker::PlayerGroup.new
    players.first
    players.last
    players[2]
    players.size
    players.rotate
```

Arquitetura: Player

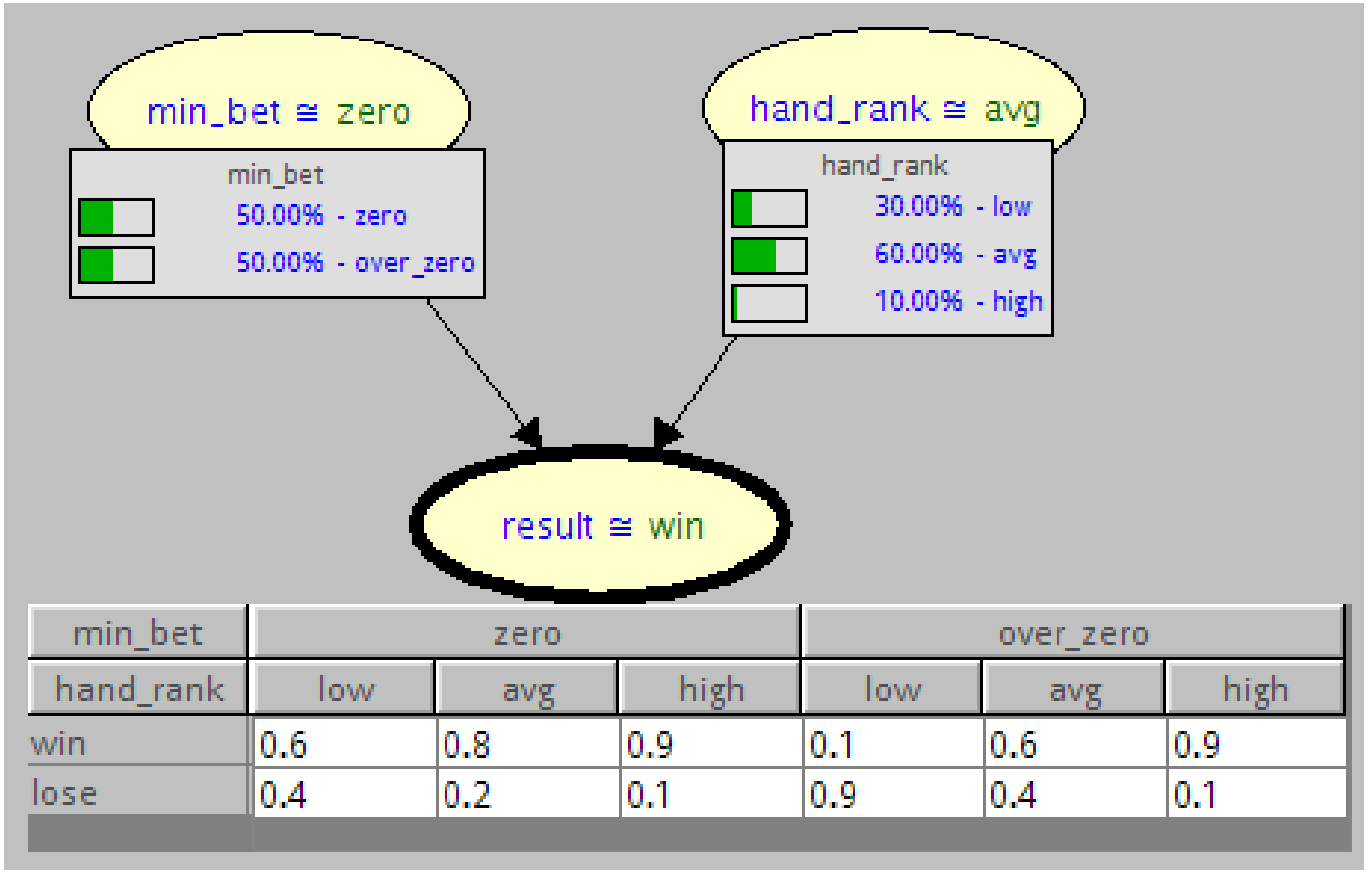
- Super classe de todos os jogadores
- Simplifica a criação de novos perfis de jogador

```
> player = Pucker::Player.new
> player.set_hand(Card.new(1), Card.new(2))
> player.bet(min_bet: 0)
> player.reward(100)
> player.hand_rank(table_cards_array)
```

Arquitetura: relatórios

- MultiLogger + Statistic
- Impressão de logs: arquivo + STDOUT
- Niveis de log: debug, info, fatal
- Número de re-buys
- Número de rodadas como melhor da mesa
- Número de rodadas com stack > 2000

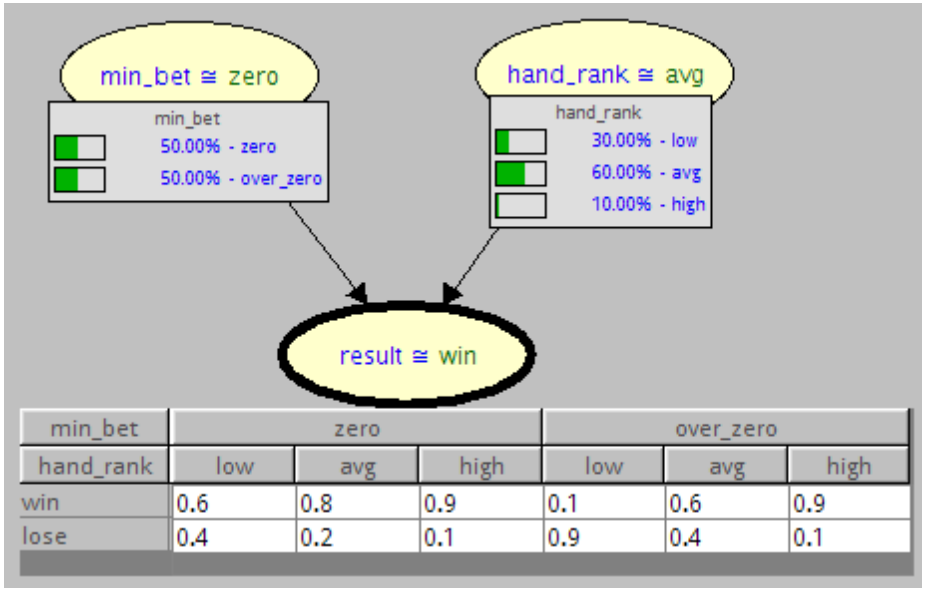
Bayesian Network: SAMIAM



min_bet	zero			over_zero		
hand_rank	low	avg	high	low	avg	high
win	0.6	0.8	0.9	0.1	0.6	0.9
lose	0.4	0.2	0.1	0.9	0.4	0.1

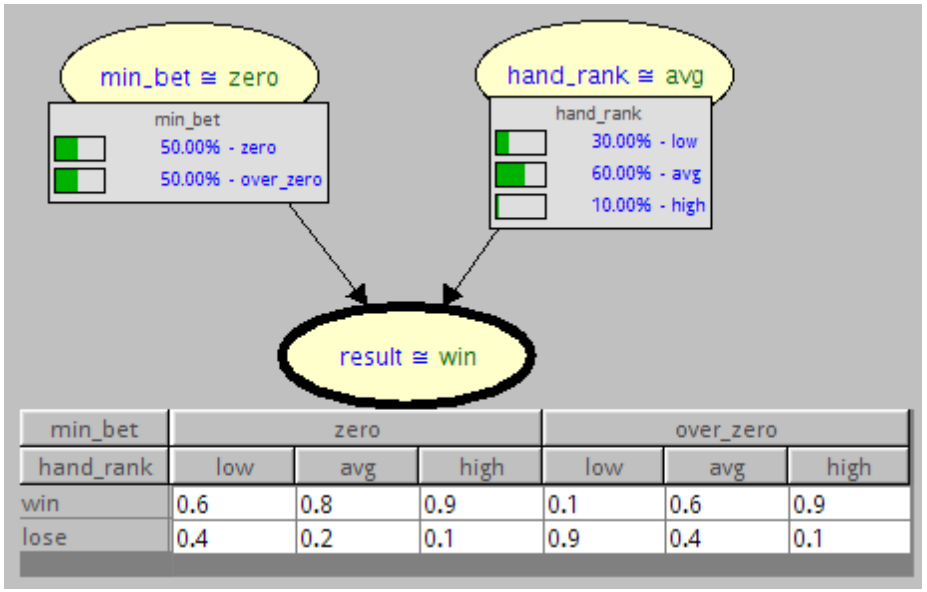
Bayesian Network: RSpec

```
Pucker::SimpleBnPlayer
  #bet
    when has low hand
      when min_bet equals zero
        should check
      when min_bet greater than zero
        should fold
    when has avg hand
      when min_bet equals zero
        should raise
      when min_bet greater than zero
        should check
    when has high hand
      when min_bet equals zero
        should raise
      when min_bet greater than zero
        should raise
```



min_bet	zero			over_zero		
hand_rank	low	avg	high	low	avg	high
win	0.6	0.8	0.9	0.1	0.6	0.9
lose	0.4	0.2	0.1	0.9	0.4	0.1

Bayesian Network: SAMIAM + SBN

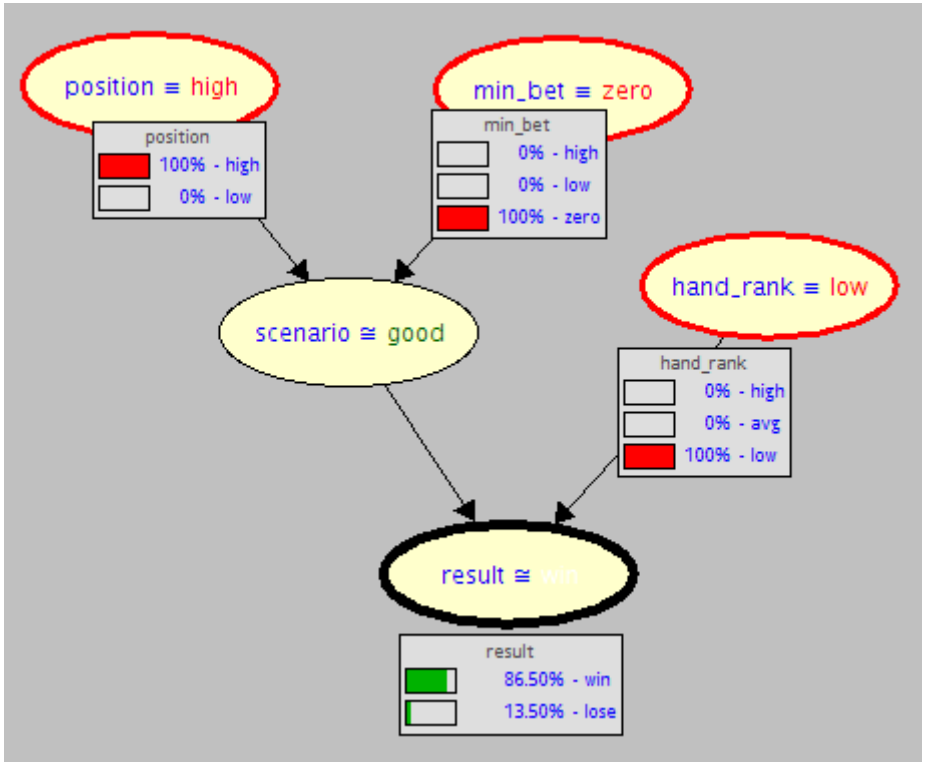


min_bet	zero			over_zero		
hand_rank	low	avg	high	low	avg	high
win	0.6	0.8	0.9	0.1	0.6	0.9
lose	0.4	0.2	0.1	0.9	0.4	0.1

```
net = Sbn::Net.new("min_bet_hand_rank")
min_bet = Sbn::Variable.new(net, :min_bet, [0.5, 0.5], [:zero, :over_zero])
hand_rank = Sbn::Variable.new(net, :hand_rank, [0.3, 0.6, 0.1], [:low, :avg, :high])
result = Sbn::Variable.new(net, :result, [0.6,0.4,0.8,0.2,0.9,0.1,0.1,0.9,0.6,0.4,0.9,0.1])
min_bet.add_child(result)
hand_rank.add_child(result)
```

Bayesian Network: BestBn

Pucker::BestBnPlayer
#bet
when has low hand
when scenario is bad
should fold
when scenario is medium
should fold
when scenario is good
should raise
when has avg hand
when scenario is bad
should fold
when scenario is medium
should fold
when scenario is good
should raise
when has high hand
when scenario is bad
should check
when scenario is medium
should raise



BestBnPlayer

- Estatísticas

Pucker: criando um jogador

SAMIAM

RSpec

SBN

NewPlayer < SimpleBnPlayer

Sobreescrever métodos:

#build_evidence

#build_bayesian_network

Problemas e Comentários

- Complexidade da simulação: collect_bets
- Bayesian Networks: infinitas possibilidades
- Sujestão: algum tipo de inteligência artificial que possa otimizar os parâmetros da rede. Exemplo: perceptron, genetic algorithm

Demonstração Prática

- RSpec: documentação
- SAMIAM
- IRB