Playing Dice with Soar

Soar Workshop

June 2011

John E. Laird

Nate Derbinsky, Miller Tinkerhess

University of Michigan

Supporting Software

- Freedice.net [Nate Derbinsky]
  - Supports correspondence dice games for humans and bots through web interface
  - Used for original development
- Dice World [Miller Tinkerhess]
  - Java based dice game
  - Many times faster than Freedice
  - Used for generating results
- Both give same information available to all players
- QnA [Nate Derbinsky]
  - Allows easy attachment of external software device where there is an immediate response.
  - Similar to I/O but with single access for input & output
- Dice Probability Calculator [Jon Voigt & Nate Derbinsky]
  - Computes simple probabilities: how likely is it for there to be 6 dice of the same face if there are a total of 20 dice.

Dice Game Rules

- Roll dice in cup
  - Everyone starts with five dice
- Actions
  - Bid: number of dice of a given rank: six twos
  - Bid and reroll: push out a subset of dice and reroll
  - Exact: exactly the number of dice bid in play
  - Challenge: < number of dice bid in play
  - Pass: all dice are the same (must have >= 2 dice).
- Lose die if lose a challenge
- Last one with remaining dice wins

Key Strategic Considerations

1. If you raise too high, you might be challenged.
2. If you don't raise high enough, it will come around again for another turn.
3. Can only lose a die if you challenged or are challenged.
4. Pushing dice increases information for other players and reduces your flexibility.

Research Issues in Dice Game

- Multi-faceted uncertainty
  - What dice are under other cups?
  - What did that bid indicate?
  - What will the next player do after my bid?
  - How combine different sources of uncertainty?
- Non-trivial cognitive challenge for humans
  - Actions take seconds to minutes
- Human performance correlated with experience
  - A fair amount to learn beyond the rules
  - Potential for opponent modeling
  - Potential for learning

Decision making under multi-faceted uncertainty in Soar?

- Agent 1: Expected Values
  - Compute expected value for the bid based on known and unknown dice.
    - If there are 6 unknown dice, and I have 2 2's, then there are most likely 6/6 = 1 + 2 = 3 2's.
    - Compare bids to that expected value and classify as certain, safe, risky, very risky.
    - Similar to what we think humans do.
- Agent 2: Probability Values
  - Compute probability of bids being successful.
    - Uses external software calculator.
- Additional heuristics contribute to final bid:
  - Don't pass or challenge if have another good bid.
  - Don't bid and push if bid alone is a good bid.
  - …

Overall Approach

1. Propose operators for *all* legal actions
   – Raises, raises with pushes, exact, pass, challenge
2. Tie impasse arises between operators
3. Evaluate all of the operators in selection space

   Create preferences based on evaluations

   1. symbolic evaluations → symbolic preferences
   2. probability-based evaluations → numeric preferences
4. Decision procedure picks the best operator.

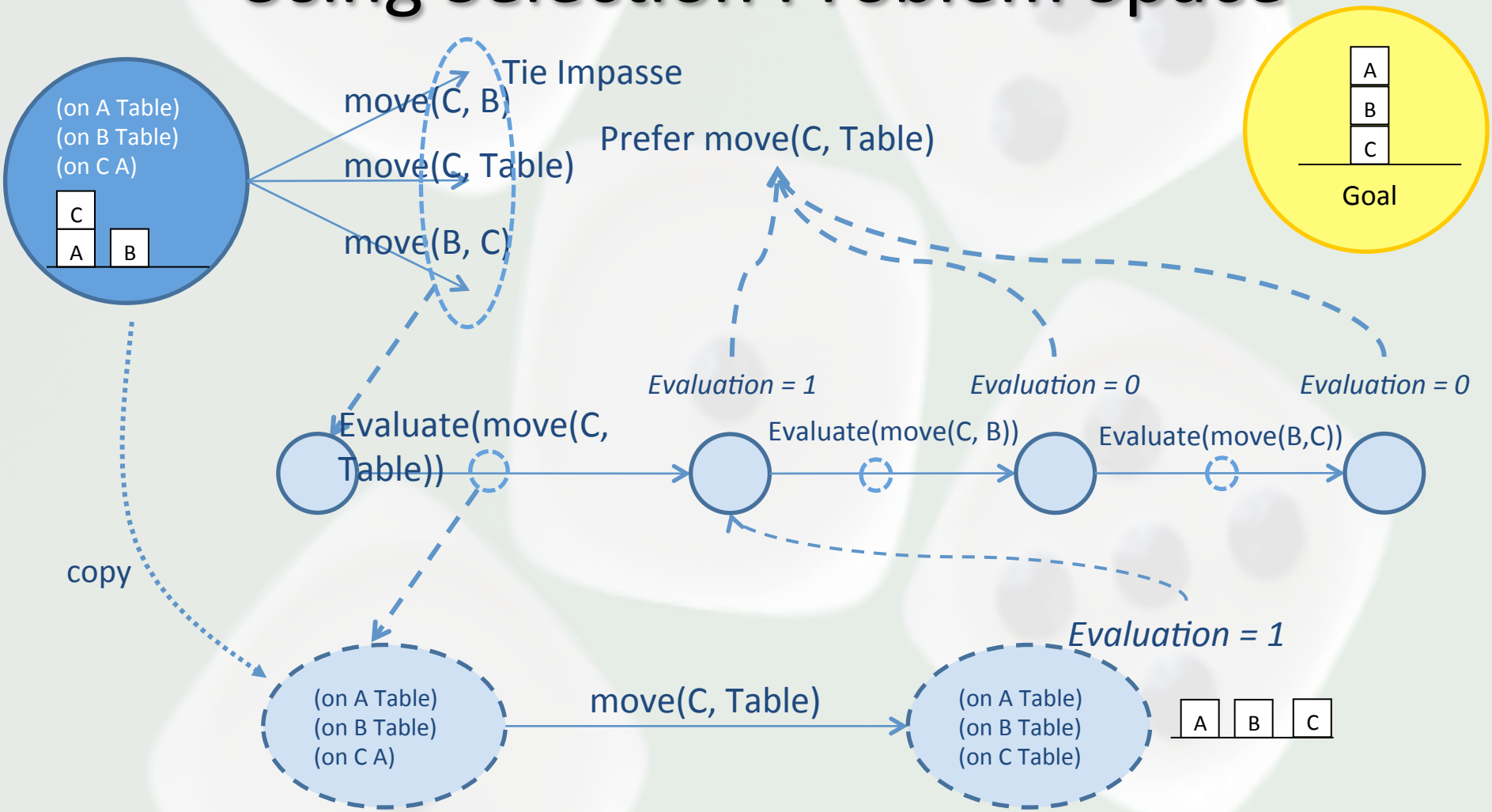
Dice Problem Space Operators

- **Raise bid**
  - Next legal bid given last bid for each die rank (1-6)
  - Special processing to determine first bid and lowest reasonable bid
- **Raise, push, and roll**
  - Next legal bid given last bid for each die rank where there is a possible push
  - Only consider pushing all relevant dice (no Valerie Bids)
- **Challenge**
  - Previous bid, or two back if previous is a pass
- **Exact** (if available)
- **Pass**

A few more operators for initialization and counting up available dice on each turn.

- Total dice showing for each rank
- Total dice under other players' cups
- Totals for my dice under my cup

One-step Look-ahead Using Selection Problem Space



Replace Look-Ahead Evaluation with Expected-Value Calculation (Agent 1)

Bid: 6[4]

Challenge

Pass

Tie Impasse

Challenge > Bid

Challenge > Pass

Bid > Pass

Conversion of evaluations to preferences is done by a bunch of rules with many special cases.

Evaluation = risky

Evaluation = safe

Evaluation = very risky

Evaluate(bid: 6[4])

Evaluate(challenge)

Evaluate(pass)

[Exposed 4's] + [in my cup 4's] + [unknown]/6 – [# of dice bid: 6] = Bid difference

compute-bid-difference

-1

compute-bid-likelihood

Evaluation = risky

Possible Evaluations: lose, very risky, risky, safe, certain

Approach #1
Operators for Evaluation

- Simple Bid (Raises)
  - Compute-bid-difference [-N to +M]
  - Evaluate-bid-likelihood [lose, very risky, risky, safe, certain]
- Bid, Push, and Reroll
  - Compute-bid-push-difference
  - Evaluate-bid-likelihood
- Challenge
  - Compute-challenge-bid-difference
  - Evaluate-challenge-bid-likelihood
  - Compute-challenge-pass-likelihood
- Exact
  - Compute-exact-difference
  - Evaluate-bid-likelihood
- Pass
  - Compute-pass-likelihood

Replace Look-Ahead Evaluation with Probability Calculation (Agent 2)

Bid: 6[4]

Challenge

Pass

Tie Impasse

Challenge = .9
Bid: 6[4] = .3
Pass = .08

Conversion of evaluations to preferences is done by a bunch of rules with many special cases.

Evaluation = .3

Evaluate(bid: 6[4])

Evaluation = .9

Evaluate(challenge)

Evaluate(pass)

Evaluation = very risky .08

Evaluation = .3

compute-bid-probability [using QnA]

.3

Approach #2
Evaluation Operators

- Simple Bids (Raises)
  - Compute-bid-probability
- Bid, Push, and Reroll
  - Compute-bid-push-probability
- Challenge
  - Compute-challenge-probability
  - Compute-challenge-pass-likelihood
- Exact
  - Compute-exact-probability
- Pass
  - Compute-pass-likelihood

Observations

- Plays a good game!
  - Doesn't make stupid bids and is a bit unpredictable
  - Tends to be conservative on bids
  - Tends to be aggressive on challenges
  - Has beaten human "experts"
- Bluffs when it doesn't have a good bid.
  - But doesn't explicitly decide to bluff
- Does not always take the safest bid
  - Sometimes from randomness
  - Sometimes because of selection knowledge
    - Don't take certain pass when have another safe bid

Model Previous Player

- Agent tends to challenge too often.
  - A player usually makes a specific bid for a reason!
- Approach:
  - Add selection space operator that computes likely dice under previous player's cup based on most recent bid.
    - Selected only if no certain bid using known dice.
  - Analysis of previous player's bid
    - Iterate through possible number of dice bid starting with 1 until
      1. find a number that would make the bid reasonable or
      2. reach a number that is very unlikely
  - Use result to recalculate expected value/probabilities of possible bids.

Results for 1000 two player games

	No Model – Probability	No Model – Expectation
Model – Probability	761/239	703/297
Model – Expectation	681/319	607/393

	Expectation – No Model	Expectation – Model
Probability – No Model	451/549	319/681
Probability – Model	703/297	571/429

- Player against self is ~480/520
- Without model, expectation-based is better.
- With model, probability-based is better.
- Model is more important for the probability-based agent.

Three & Four Player Games with Models

Expectation	414
Expectation	289
Probability	297

Expectation	384
Probability	284
Probability	332

Probability	346
Probability	331
Probability	323

Expectation	246
Probability	231
Expectation	267
Probability	256

Expectation	381
Expectation	230
Probability	146
Probability	243

Future Work: More Opponent Modeling

- Extend evaluation calculations
  - Worst case analysis for next player for my best bids
    - "No matter what he has, he won't challenge that bid."
    - Allows agent to bluff more.
  - Possibly biased by next player's last bid.
- Episodic memory:
  - History of how next player responded to similar bids.
  - History of what player had when made similar bid.
- Model other players to have better estimates of hidden dice.

Future Work: Chunking and RL

- Use chunking to learn (lots of) RL rules.
  - Hard to write these by hand – too many cases
  - Chunking works with the current agents to learn rules that test features relevant to created preference.
  - Numeric preferences are initial values for RL.
- Use RL to "tune" rules learned by chunking.
  - Need lots of experience.
- Interesting proposal for the source of RL rules.
- Initial results are not promising…

Future Work: More Baselines and Experiments

- Pure probability and expectation-based without heuristics.
- RL agents with different value functions.
  - Fair number of state-action pairs:
  - Number of possible prior bids (>200) * possible configurations of dice (10^6) * number of next bids (~15) = 3 Billion
- Human players (from the web?).
- Create (large) set of cases that can be used for direct comparison:
  - Find states where different agents make different bids.

Nuggets and Coal

- Nuggets:
  - Two different approaches for reasoning with probabilities in Soar that fits in “naturally”.
  - Example of using opponent model.
  - Leads to a competent dice player.
- Coal
  - Don’t understand strengths and weaknesses.
  - Has promise for generating and using RL rules but haven’t achieved it.