Learning and Planning in TankSoar

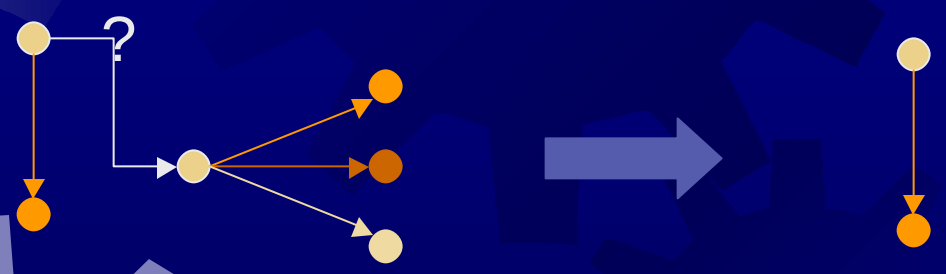
John E. Laird
University of Michigan

Learn to play a complex game

- Task specification & rules of the game
  - What actions can be performed in the world?
  - What are the “physics” of the world?
  - How do you win the game?
- Tactics
  - What action(s) should be performed now?
- Compare learning to hand-coding
  - How much effort?
    - Task-independent vs.Task-dependent knowledge
  - How good is the behavior?

Strategy for Learning Tactics

- Use planning to pick best move
  - Internal look-ahead planning in Soar
    - *Apply* operators via internal simulation
    - *Evaluate* and compare results to pick operator
    - **<u>NO OPPONENT MODELING!!!!</u>**



- Cache results and eliminate planning
  - Chunking in Soar

Knowledge to plan and learn

- Task-independent planning
- Task support knowledge
- Internal simulation of actions
- Evaluation of states

What are the operators?

- Exclusive major actions
  - rotate left/right
  - move forward/back/left/right
  - none
- Inclusive independent minor actions
  - Shields on/off
  - Fire missile yes/no
  - Radar-power: 0-13
- Major action changes minor action results
  - Turning changes what you shoot at
  - Moving changes whether you will get hit
- Total operators =
  - 7 major actions * 18 minor = 126 (vs. 392)

Simulating Operators

- Two stages:
  - In parallel, independent changes to world model
    - Move, rotate, shields on/off, radar power, missile in air
  - Sequentially update impact on energy, health, …
- Fire Missile (3 rules)
  - Decreases number of missiles
  - Put missile in air toward enemy
- Move (16 rules)
  - Change square (indirectly changes radar)
  - Detect collision
  - Update sensors (open squares, incoming)

Evaluation Function

- Must compute components sequentially
  - No parallel addition in Soar!
- Components of Evaluation Function:
  - Live/Die (incoming)
  - Energy (shields, radar, recharger)
  - Health (recharger)
  - Wall collisions (Ouch!)
  - Missiles (fire missiles, pickup missiles)
  - Hit enemy
  - Clear in front
  - Up-to-date knowledge of map (radar, move)
    - Computes value of seeing squares – Argh!

```
==>state no-change
    generate-operator [turn right]
    extend-operator
        generate-operator [shields on]
        evaluate-operator
            [turn right, shields on]
                evaluate-dimension [radar-explore]
                    test-square
                    test-square
                    …
                evaluate-dimension [shields]
                …
                finish-evaluation [45]
        generate-operator [radar 13]
        …
        finish-extend [pick best extensions]
    evaluate-operator
        [turn right, radar 5, shields off]
            evaluate-dimension […]
            …
    generate-operator
```

Example Chunks

If evaluating an operator that movies forward and

turns on the radar to 13 and

the facing east and

there are 10 open positions to the east that haven't been viewed recently and then an obstacle

Then

the exploration value is 250

Chunking Challenges

- Shouldn't be specific to current health/energy
  - Preclassifies health/energy into buckets
  - Avoids blowing away subgoals
- Must learn when operator should terminate
  - Associate parameters with each actions
- Overgeneralization
  - See 10 things I hate about Soar

Knowledge to plan and learn

- Task-independent planning
  - 62
- TankSoar support knowledge
  - 69 of 170 from handcoded version
- Internal simulation of actions
  - 63
- Evaluation of states
  - 61
- Total 255
  - 62 + 193 vs. 170 hand coded

Observations

- Hand-coded system
  - Only 1-2 rules to power up radar during a turn
  - No chained tactics except:
    - Will return to charger when low on health/energy
      - Uses internal features that would be hard to learn
    - Some tactics implicitly have some opponent model
- Learning system
  - Must learn a lot of chunks!
    - Most are pretty good
  - Does a good job in using radar
  - Should require less work for changes to game
  - Can’t learn from experience with enemy
    - NO OPPONENT MODEL

Nuggets and Coal

- Nuggets
  - Identify knowledge required to learn from planning
  - Shows how hard learning is
  - Almost works – chunking all the time
  - Writing this talk gave me ideas to simplify system
- Coal
  - Comparison to hand-coded incomplete
  - Engineering evaluation function is hard
    - Planning always finds holes in evaluation
  - Getting chunking to generalize correctly is hard
  - Shouldn't be this hard!
    - Some of the hardest Soar code I've ever written
    - Up there with mapping in Quakebot