Evolution of Cooperative problem-solving in an artificial economy
by E. Baum and I. Durdanovic

presented by Quang Duong
Outline

• Reinforcement learning and other learning approaches’ limitations
• Artificial Economy
• Representation Language: S-expression
• Block Worlds
• Rubik’s Cube
• Rush Hour
• Conclusion and further exploration
Reinforcement learning

• Value iteration, policy iteration
• TD learning: neural network, inductive logic programming
• Limitation:
  – state/policy spaces are enormous
  – evaluation function (reward) is hard to learn/present
  – (based on empirical results)
Related work

Holland Classifier System:
- A set of agents –classifiers: communicate with the world and other agents
- Bucket brigade and genetic algorithms

• Is it a fair comparison? search vs. plan
E.g: GraphPlan, Blackbox
Evolution approach

- Artificial economy of modules.
- Learn a program capable of solving a class of problems, instead of an individual problem.
- Higher reward for harder problems
- Revolution:
Artificial Economy

- Hayek: collection of modules/agents
- \( W_{\text{init}} \) set to the largest reward earned
- Auctions: among modules/agents; the winner executes its set of instructions.
- Each agent:
  - Wealth
  - Bid: min of wealth and the world simulation’s returned value
  - Create agents if Wealth > 10 \( \times \) \( W_{\text{init}} \)
- An example…
- Assumptions:
  - only one agent owns everything at a time! Competing for the right to interact with the real world
  - conservation of money
  - voluntary inter-agent transaction and property rights
Representation Language

• S-expression: a symbol or a list structure of S-expressions; typed expression !!!

• Hayek3; capable of solving large but finite problems
Block World

- Max moves: $10n \log_3(k)$ while $k$ is the number of colors
Evolution process

- Start: a hand-coded agent
- New agent: mutation of the parent agent.
- $W_{\text{init}}$: highest earned reward (initially 0)
- Demonstration…
- Additional elements: Numcorrect and R node
Block World

- Difficulty: combinatorial numbers of states
- A class of problems: uniform distribution of size
- Result: 100-block problems solved 100%, 200-block solved 90% with 3 colors
Resulting system

• 1000 agents

• Bid calculation formula: $A \cdot \text{Numcorrect} + B$:
  - $A, B$ (agent type)
  - $\text{Numcorrect}$: environment
    - 3 recognizable types: a few cleaners, many stackers, closers (say Done)

Macro-action.

Example:
  Approximate $\text{Numcorrect}$. $R(a,b)$ node.

\[ \text{For} (\text{And} (\text{EQ}(\text{Look}(0, h), \text{Look}(1, h)), \text{Not} (\text{EQ}(\text{Look}(0, h), \text{Empty}))))). \]
Meta learning

• 2 types: Creator (don’t bid): modify or create – otherwise, Solver.

• Example…

• Result: show 30%-50% better performance after a week of execution.

• Limitations:
  – expensive matching computation
  – shows no better performance than the simple Hayek system
Rubik’s cube model variations

• 2 schemes:
  – Scrambled in one rotation, reward only for completely unscrambling.
  – Completely scrambled, partial reward: (1) number of correct cubes (2) number of correct cubes in a fixed sequence (3) number of correct faces.

• 2 cube models
• 2 languages.
• Limitations: after a few days cease to make progress due to:
  – More difficult to find new beneficial operators without destroying the existing correct part.
  – Increasingly complex operators
    Less structure to be exploited
Rush Hour

- Challenges: possible exponential move solution, representation language
- Expressions are limited to 20,000 nodes
- Note: the use of Creation Agents.
- Learn with subsampled problems (40 different instances)
Rush Hour (cont.)

• The ability of agent to recognize distance from the solution (low bid for unsolvable problem)

• Reward: \[ R = (\text{orderno.})e^{(4.06p-4.06)} \]

• Break down to subgoals

• Genetic algorithms never learned to solve any of the original problems
Conclusion

• Property rights and money conservation
• Learning in complex environment will involve automated programming
• Problem: space of programs is enormous
• Hayek: creates very complex program from small chunk of codes
• Suggestions: trade info, single and universal language
• Comments?
• Applications?
• How related to the course?