Q1. What types of problems are most appropriate for deep Q-learning?

(A.) All problems, deep learning has no downsides

(B.) Well-documented problems where Q-learning struggles due to state space size

(C.) Problems with a state space similar to reality

(D.) The same class of problems appropriate for non-deep Q-learning

Q2. Which of the following is a major difference between Q-learning and Value Iteration?

(A.) Q-learning doesn’t require a transition function/forward model

(B.) Q-learning is Value Iteration, but stronger

(C.) Q-learning takes up less memory than Value Iteration.

(D.) Value Iteration is much slower than Q-learning

Q3. For the multi-armed bandit problem, which statement below best summarizes why we can't just use exploitation or exploration?

(A.) Exploration means it will take a long time to find the optimal solution

(B.) Both will minimize our total regret

(C.) Pure exploration doesn't learn, pure exploitation learns too fast

(D.) Pure exploration means we can never do better than chance, pure exploitation can get an agent caught in a local maximum strategy

Q4. Given the list of desired effects to the final policy of a Markov Decision Process (MDP), what MDP element would you alter? Acceptable answers: **state representation, reward function, transition function/forward model, discount factor, or N/A**.

a) Cause the final policy to direct an agent closer or further from world elements

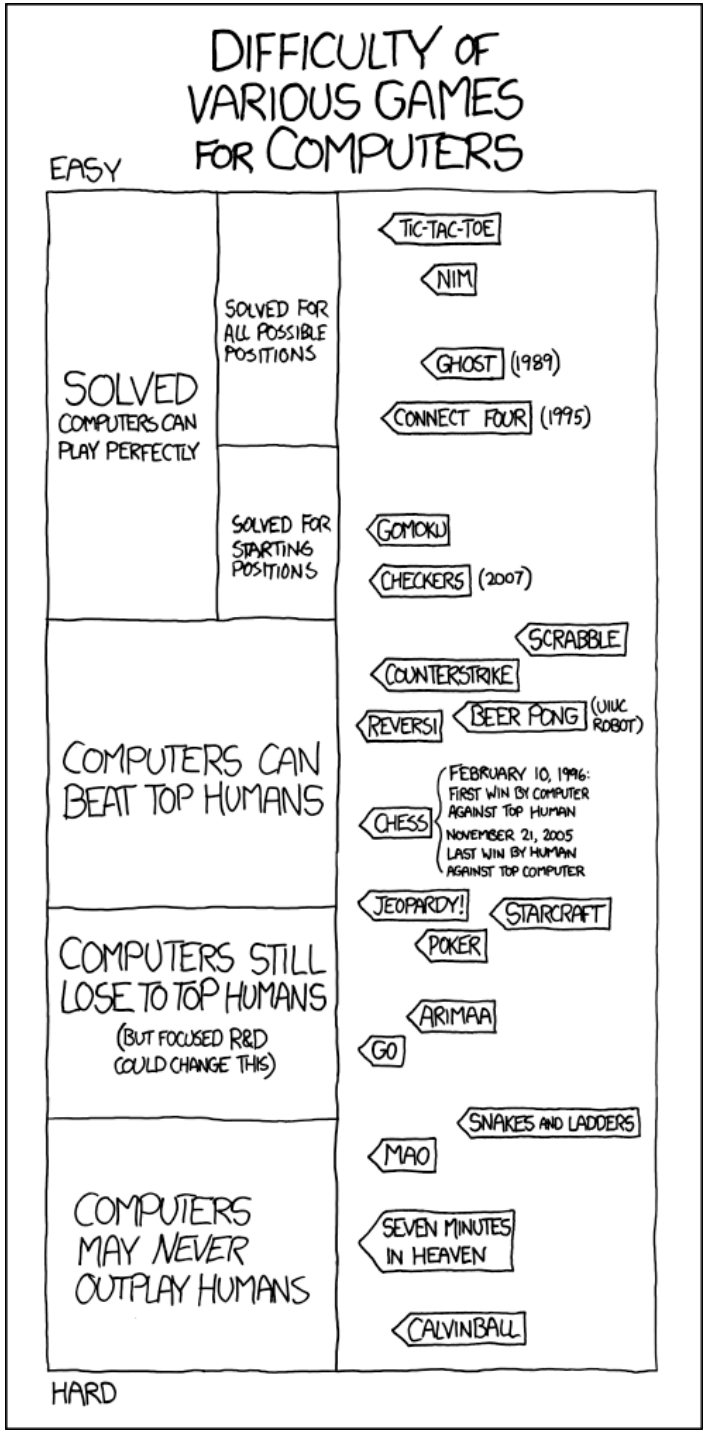
b) Shrink the size of the final policy table

c) Decrease the time for the final policy to converge

Minimax, MCTS

2019-11-20

"It was a **watershed event, but it doesn't have to do with computers becoming intelligent**," said Douglas Hofstadter, a professor of computer science at Indiana University and author of several books about human intelligence, including *Gödel, Escher, Bach*, which won a Pulitzer Prize in 1980, with its witty argument about the **connecting threads of intellect in various fields** of expression. "They're just overtaking humans in certain intellectual activities that we thought required intelligence. My God, **I used to think chess required thought**. Now, I realize it doesn't. It doesn't mean Kasparov isn't a deep thinker, just that **you can bypass deep thinking in playing chess, the way you can fly without flapping your wings**." — Bruce Weber, "Mean Chess-Playing Computer Tears at Meaning of Thought," The New York Times, Feb 19, 1996



https://xkcd.com/1002/

On Chess ~10^{123} (Shannon Number) vs Go ~10^{360}

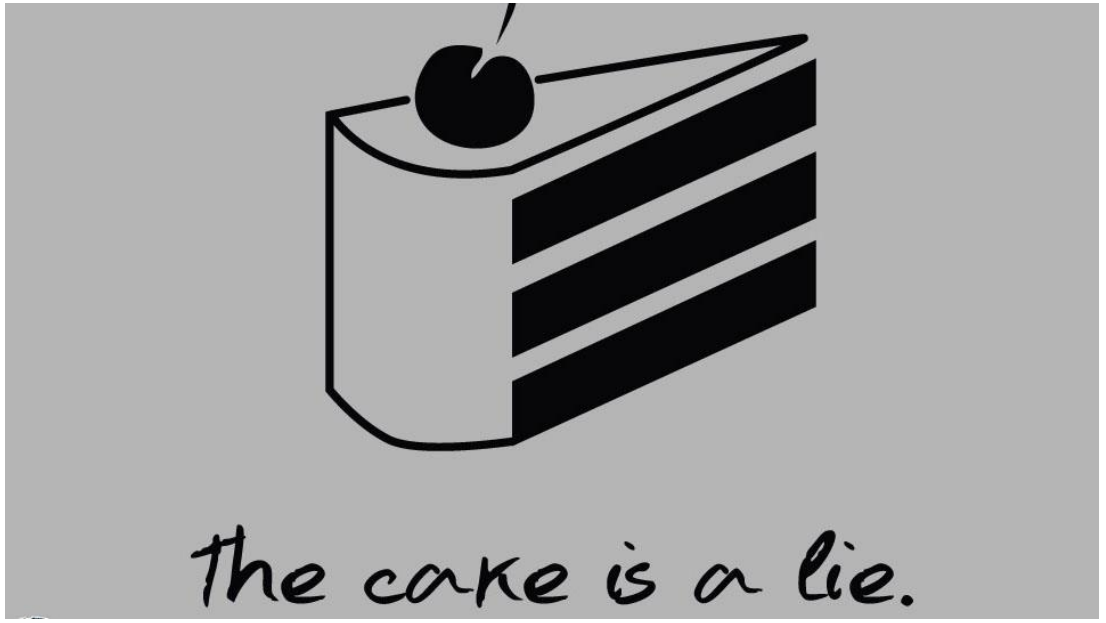
https://en.wikipedia.org/wiki/Shannon_number

- 2017: DeepMind (Google) takes on chess w/ AlphaZero
  - 1997: Deep Blue employed custom VLSI chips to execute the alpha-beta search algorithm in parallel (200m positions per second)
  - AlphaGo wins Go, AlphaZero wins chess, AlphaStar wins starcraft II
  - Combines Deep NN with general reinforcement learning algorithm
- No opening theory, no end-game database – learned all from self play (in 4 hours). In 100 Games
- Stockfish 8 looks at 70m positions per second. AlphaZero 80k per second
  - Stockfish can use up to 512 CPU threads in multiprocessor systems.
  - The maximal size of its transposition table is 128 GB.
  - Stockfish implements an advanced alpha–beta search and uses bitboards.
  - Compared to other engines, it is characterized by its great search depth, due in part to more aggressive pruning, and late move reductions
  - In 100 games: AlphaZero vs Stockfish, 28 wins and 72 draws.

https://en.wikipedia.org/wiki/Game_complexity

Cake-Cutting Dilemma

- Divide a piece of cake between two children
- Each wants the largest piece
- Mother assigns one to be “cutter” the other as “chooser”
- Cutter slices the cake, chooser picks their slice



Chooser's Strategies

Cutter's Strategies

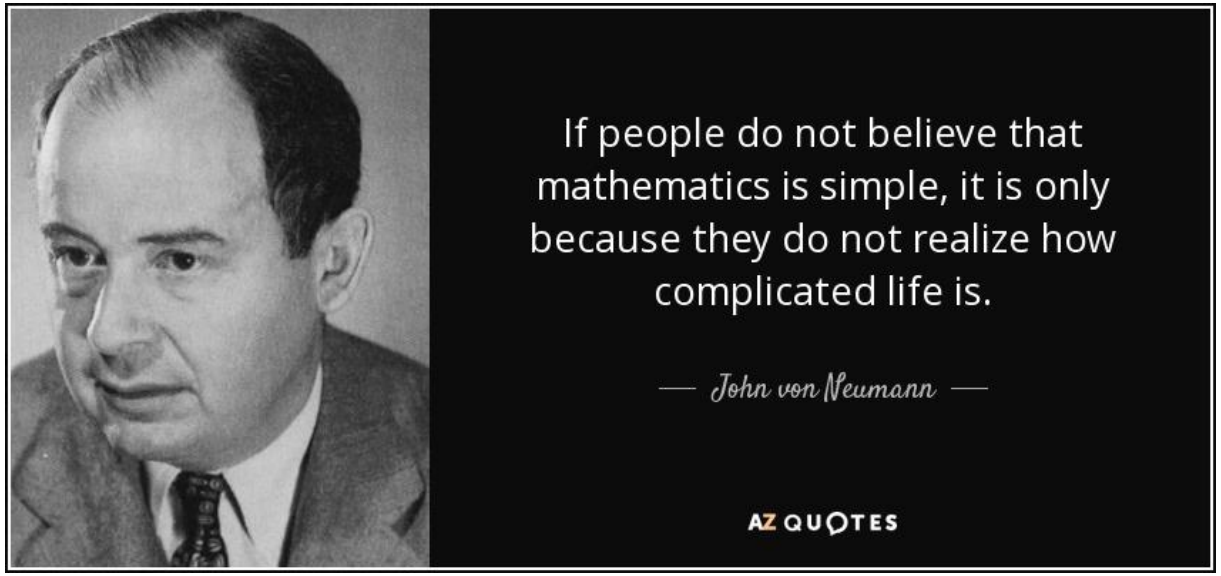
	Choose Bigger Piece	Choose Smaller Piece
Cut as Evenly as Possible	**Chooser gets a slightly bigger piece.**	Chooser gets a slightly smaller piece.
Cut One Piece Bigger	Chooser gets a bigger piece.	Chooser gets a smaller piece.

Zero-Sum Game

- Total amount won at the end of the game is exactly equal to the amount lost.
  - Chess, tic-tac-toe, connect-four, checkers, go, poker, etc.
  - Interests of players are diametrically opposed.
  - What one player loses is gained by the other.
- Cake-Cutting Dilemma is an example
- Study of zero-sum games began the study of game theory, which is a mathematical subject that covers any situation involving several agents attempting to make decisions that maximize their own utility.

Minimax Theory

- Von Neumann discovered that there is an optimal strategy for each player in zero-sum games
  - In two-player zero-sum games, the minimax solution is the same as the Nash equilibrium!
- Optimal strategy is “maximize their minimum potential result”



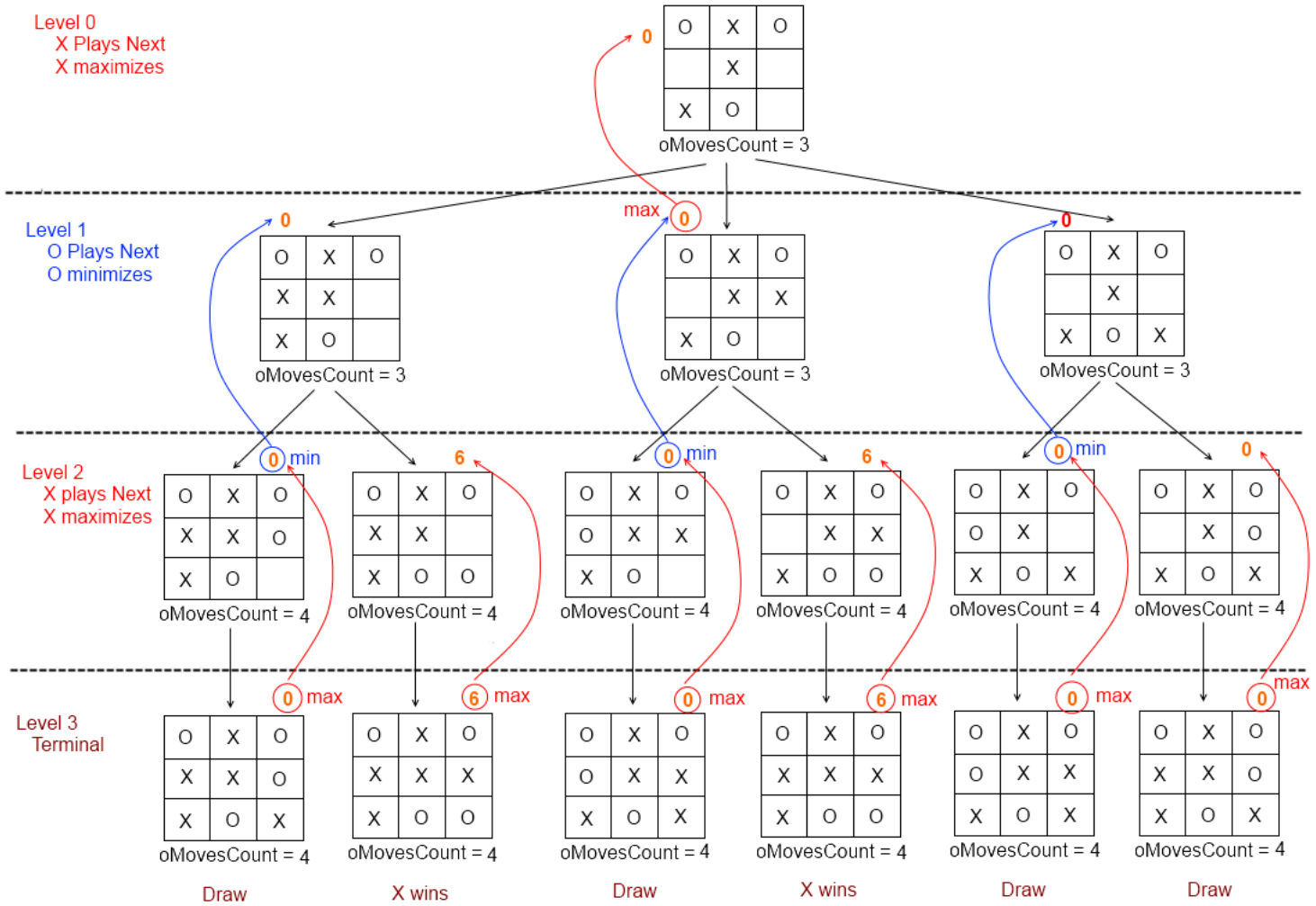
Adversarial Search (in a Game Tree)

- Components of the search problem:
  - Initial state
  - Possible actions
  - Transition model (S,A → S')
  - Players
  - Terminal test (win/loss/tie)
  - Utility function: U(S,P) → real

- BFS, DFS, A* etc attempt to maximize (or minimize) every choice equally
  - Example: find shortest route
- In game search, maximize the payoff of our choices but minimize the payoff of the other player's choices.
  - Treat opponent's positive utility as our negative utility

Minimax: Tic Tac Toe

basically depth-first search but also propagate information about utilities (max or min) upwards



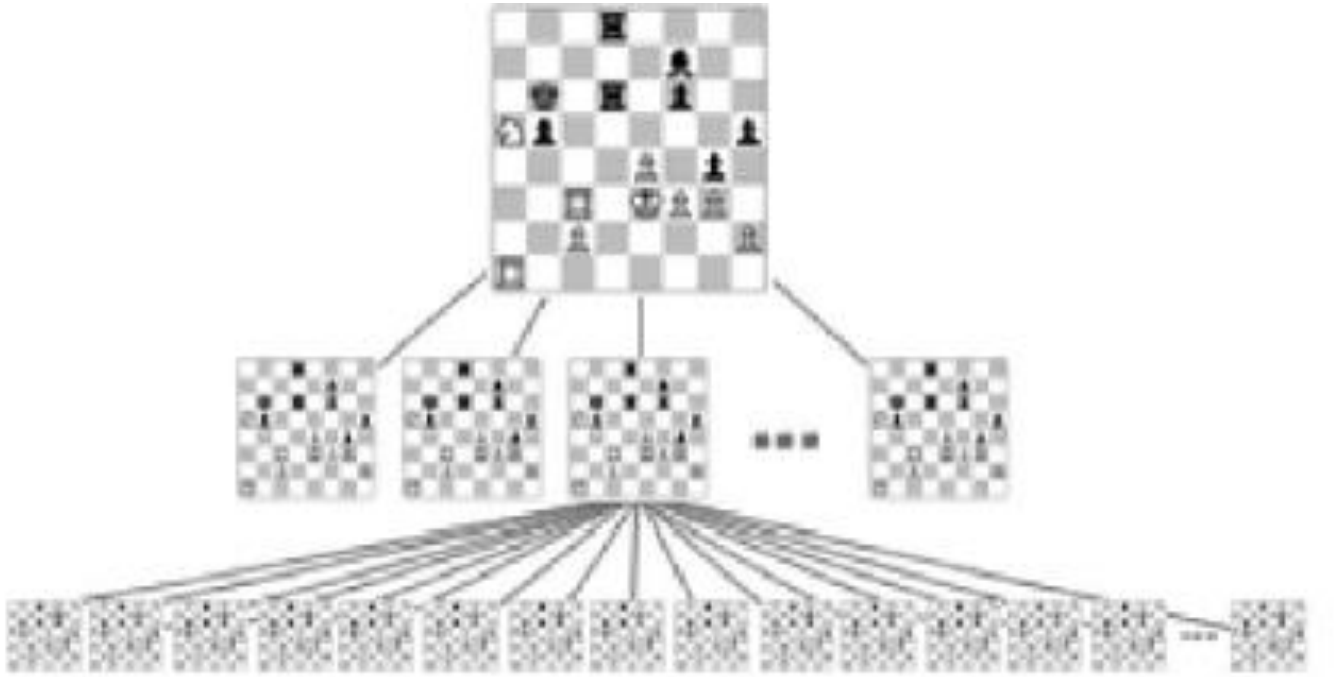
https://mostafa-samir.github.io/Tic-Tac-Toe-AI/

Minimax: Tic Tac Toe

- Minimax strategy works easily by going down the tree of possibilities to a fully filled tic tac toe board
- Only at a final (filled) board state can we determine a winner
- But what if we tried to apply minimax to a game like chess?

Minimax: Chess

- Problem: The range of possible moves is impossible to evaluate given time and memory constraints!



http://stanford.edu/~cpiech/cs221/apps/deepBlue.html

Minimax: Evaluation Function

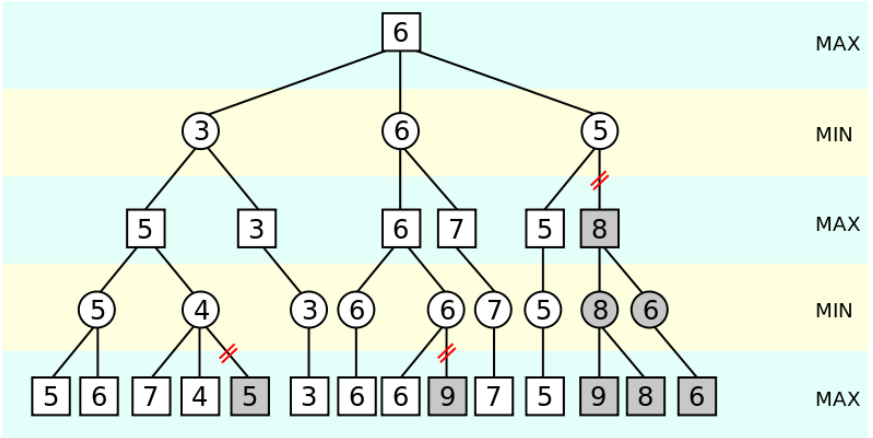
- For games that are impossible to exhaustively evaluate, an evaluation function can be used
  - The evaluation function will take the game state as an input and return a real value score
- Minimax search can then be applied to a desired depth
- At the max depth, the evaluation function scores each leaf in the graph

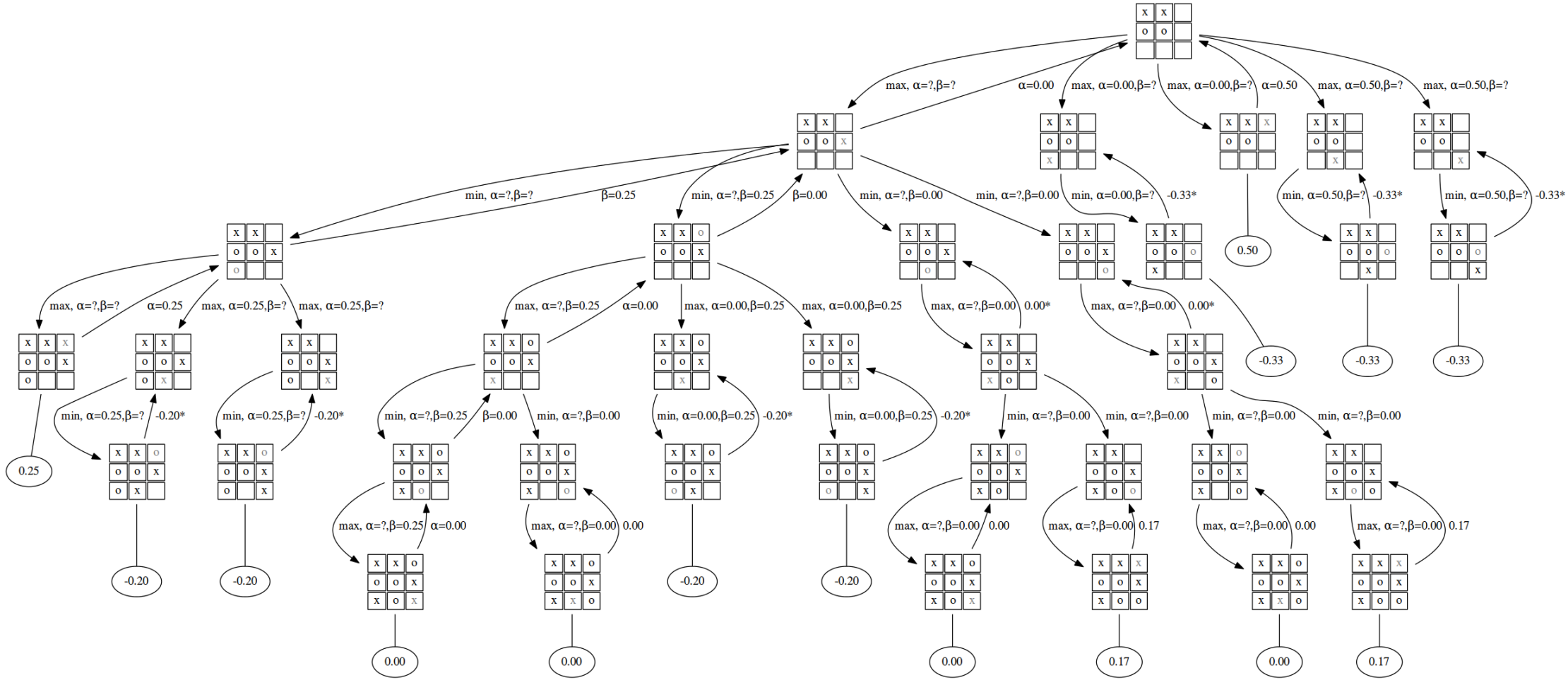
Minimax: Evaluation Function for Chess

- There is no known perfect evaluation function for chess; use a heuristic
- Criteria for scoring might be related to:
  - Opponent's king in checkmate (probably score infinity)
  - Agent has more pieces than opponent
  - Agent has queen, but opponent does not
  - Pieces have relative value (queen worth more than pawn)
  - Certain relative positions of units
  - Etc.

Minimax: Optimizations

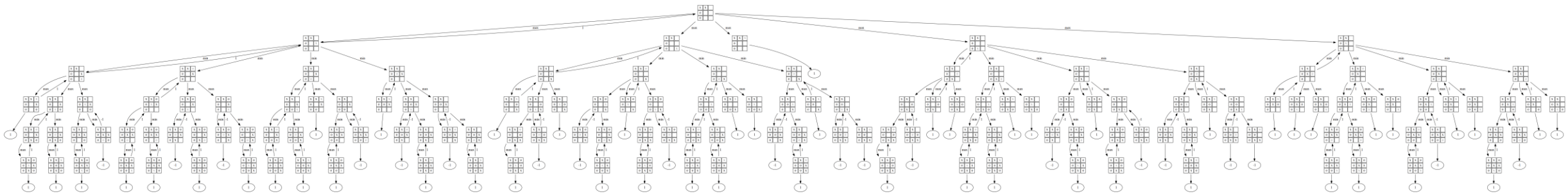
- Identify states that should not be further explored
  - Agent has lost (e.g. chess: checkmate)
  - State recognized to repeat forever
- **Alpha-Beta Pruning**
  - Track the best moves
  - Prune branches that can’t do better
  - Key insight: when performing the "max" step, if a move gives a maximum utility X (by searching its entire subtree), we can avoid any searches of subtrees whose minimum value is less than X.
  - Does not produce different answers than minimax! Simply does not bother checking subtrees that will not change the answer produced by minimax.





http://cse3521.artifice.cc/images/ttt-alphabeta-example.png

http://cse3521.artifice.cc/adversarial-search.html



http://cse3521.artifice.cc/images/ttt-minimax-example-2.png

Minimax: Optimizations

- Iterative-deepening:
  - Go to a reasonable depth first, based on time allowed
  - Pruning may allow for further search
  - Extend search
- For real-time applications, pre-allocated tree structures can speed up
- Consider adding Case-based Reasoning
  - Cases can cache heuristic previously calculated eval func score
  - Cases can reflect history of known matches and opening moves
  - Eval func may score more highly if selected move transforms state to a known case with a desirable outcome

More than two players

- Could extend minimax algorithm for N players…

Chinese Checkers

More than two players

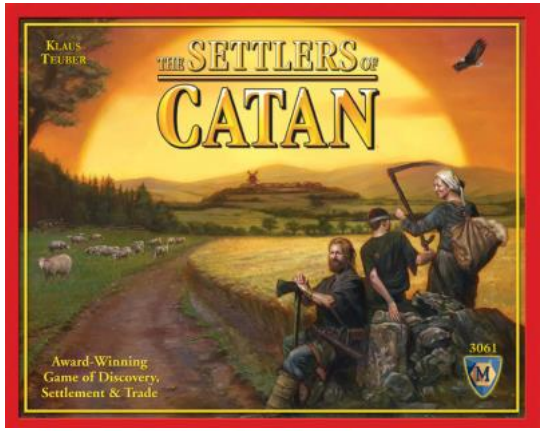
- Could extend minimax algorithm for N players…
- But one's opponents may collaborate to defeat you, forgoing the strategy of maximizing individual gain and instead redefining "winning" as any outcome where you lose
- Human players often conduct *kingmaking* (Player guaranteed to lose but possesses ability to affect final outcome. This impact becomes a new "win" condition for the losing player)
- Human players may gang up on the player with leading score in hopes of later jumping to the front

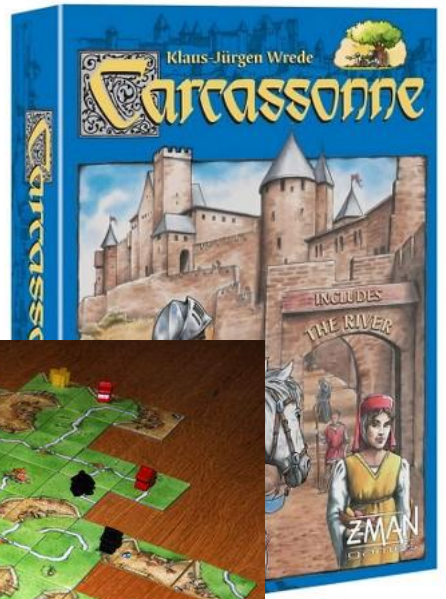
Chinese Checkers

More than two players

- Strategy: Redefine game as you/agent versus everyone else
- Effectively makes the game two-player, but with a more complicated simulation of the second player
- Other than a more complicated step for the virtual 2nd player, apply minimax as usual
- Downside: Can result in overly cautious game decisions
- Observation: Games with the potential for collusion (or kingmaking) may result in non-zero-sum games

What about probabilistic zero sum games?

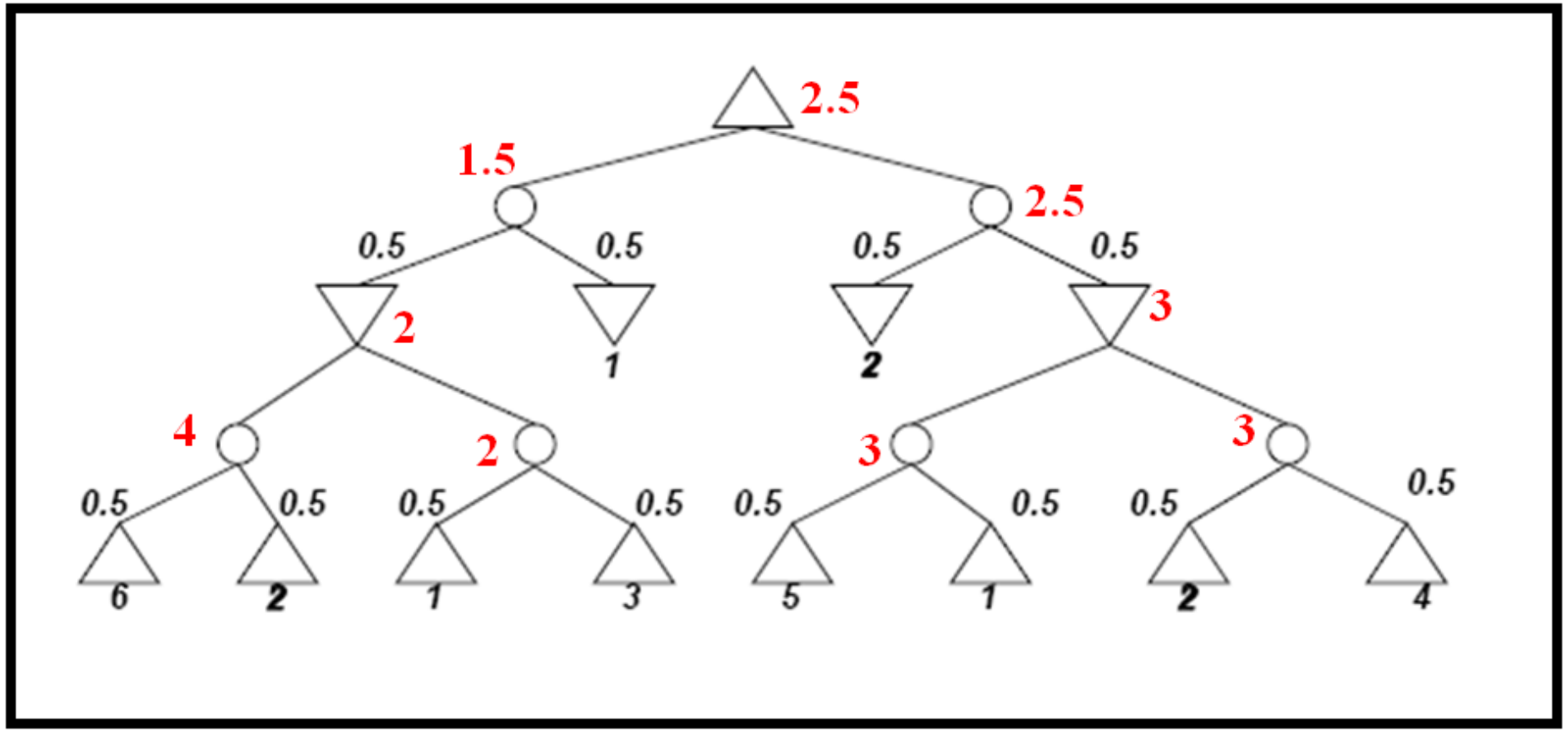




Expectiminimax

- Variant of minimax applicable to zero-sum games that depend on player strategy **as well as chance elements like dice rolls**
- In addition to the *min* and *max* nodes**, *chance* nodes are interleaved between min and max nodes**
- May have a *chance* node between each player, or one *chance* node and then both players go (e.g. *min* and *max*). Actual structure depends on the game rules!
- *Chance* nodes take a weighted average of children, with weight being probability that child is chosen randomly
- Otherwise, *min* and *max* are evaluated the same as normal minimax

Expectiminimax



Monte Carlo Tree Search (MCTS)

- Analysis of most promising moves, expanding the search tree based on **random sampling of the search space**
- Rather than search to a particular incomplete depth, we **analyze many *complete* playouts (simulate a path to game completion)**
- The **best nodes are tracked and are more likely to be *selected* in future** simulation steps
- After some number of steps (or time limit), the node that was most frequency selected is returned

MCTS popular for game of Go

MCTS Tutorial: https://www.youtube.com/watch?v=Fbs4lnGLS8M

MCTS – Four Steps

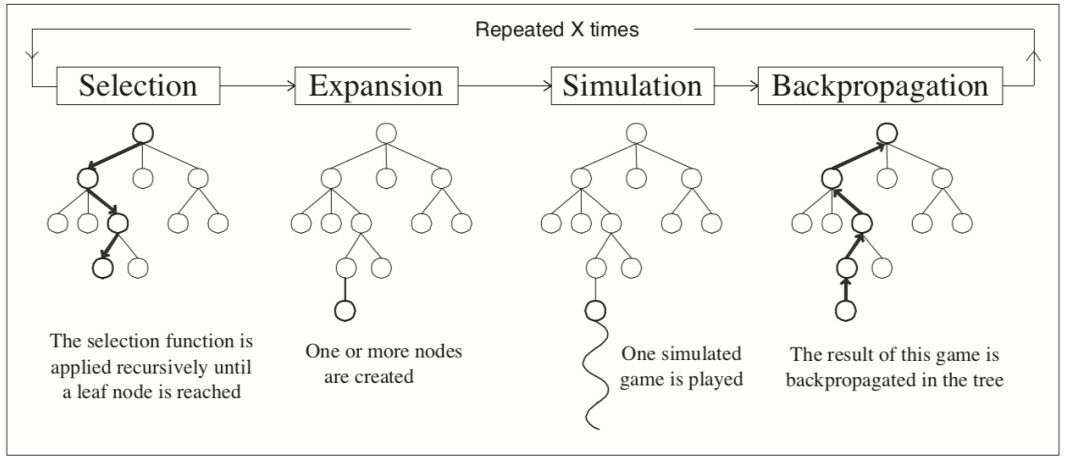
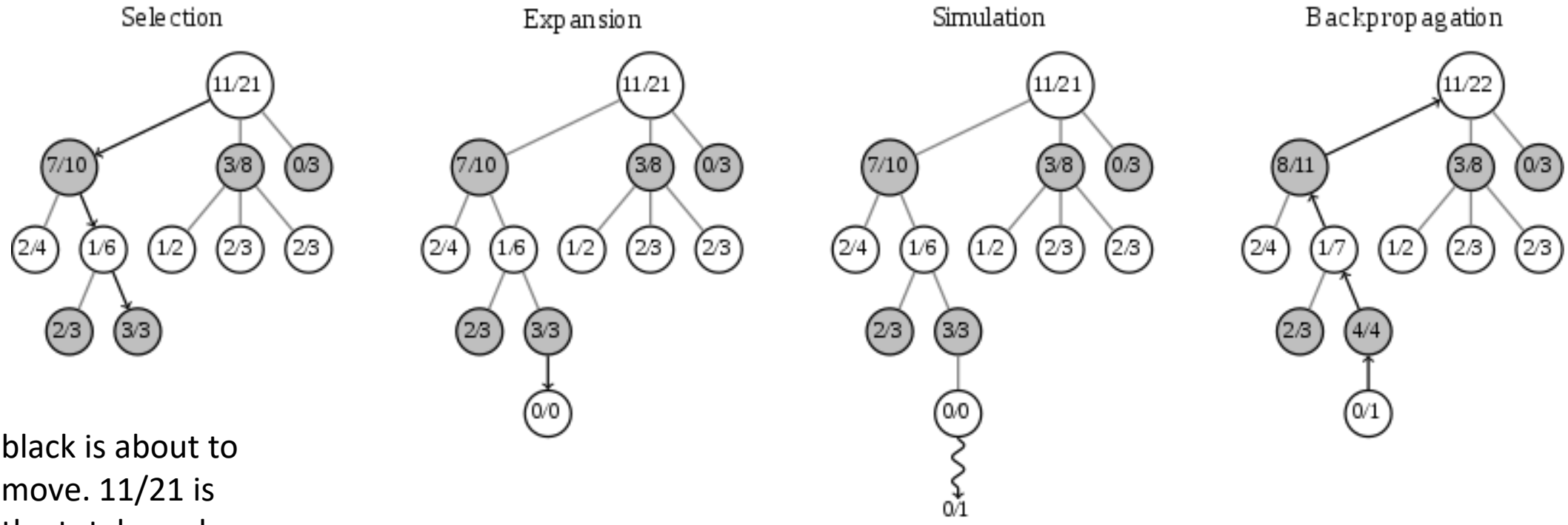


Fig. 1. Outline of a Monte-Carlo Tree Search.

- ***Selection***: start from root *R* and select successive child nodes down to a leaf node *L*
  - Selection criteria is important. Want to explore most promising leads to confirm quality. If a move turns out bad, then we will switch to another
- ***Expansion***: unless *L* ends the game with a win/loss for either player, create one (or more) child nodes and choose node *C* from one of them.
- ***Simulation***: play a random playout from node *C*. This step is sometimes also called playout or rollout.
- ***Backpropagation***: use the result of the playout to update information in the nodes on the path from *C* to *R*

MCTS Example



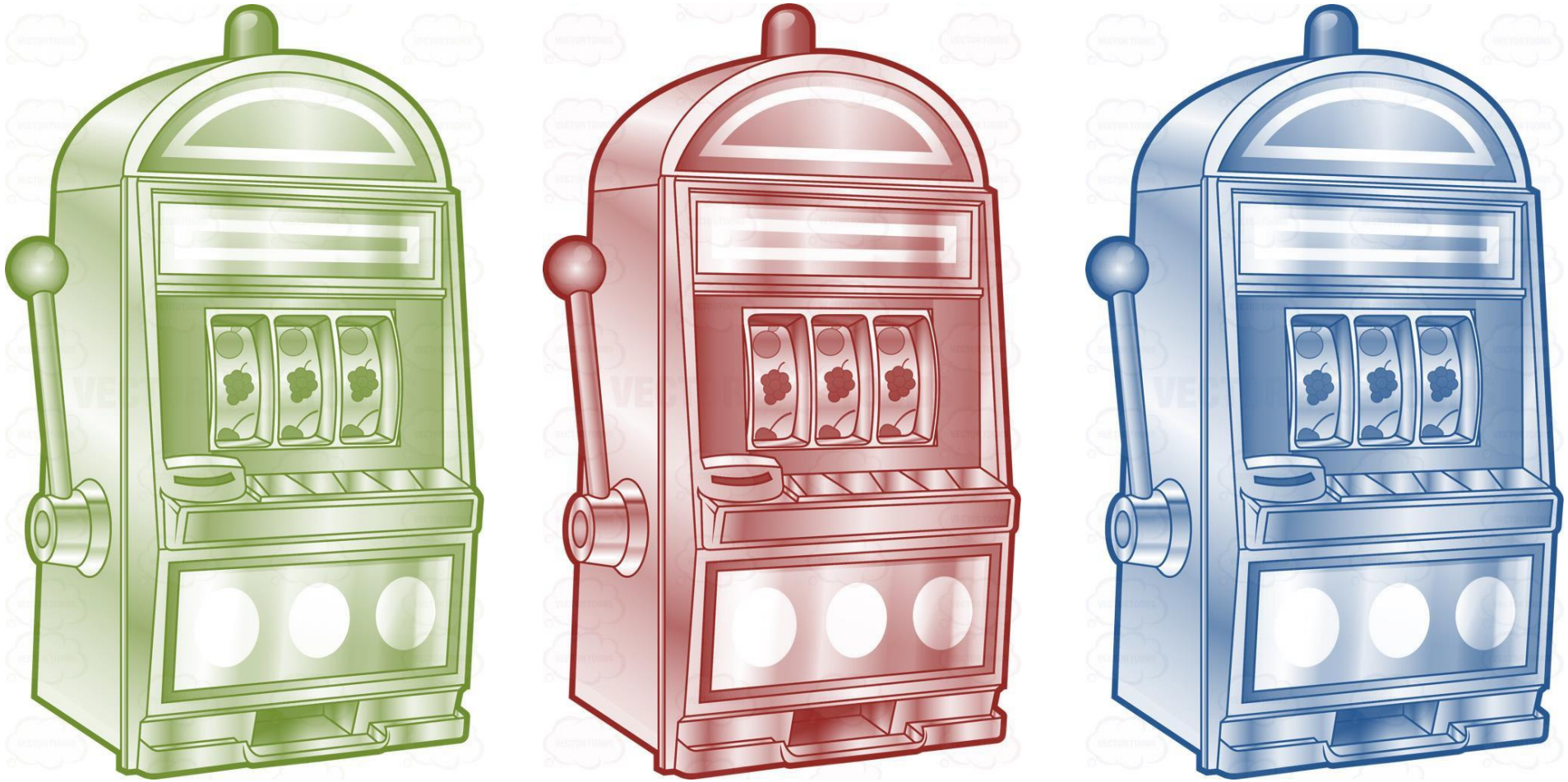
black is about to move. 11/21 is the total number of white wins in the playouts so far from this position

See also https://towardsdatascience.com/monte-carlo-tree-search-158a917a8baa

MCTS: Improvement

- How can we improve MCTS?
- We desire to continue to refine estimates of the quality of moves, but we also want to explore new walks through the tree in hopes of identifying promising moves, or moves that the opponent is likely to select
- How can be do both?

Multi-armed Bandit Problem



“Through repeated action selections you are to maximize your winnings by concentrating your actions on the best levers.” – Sutton & Barto

Upper Confidence bounds applied to Trees (UCT)

- Variation of MCTS on the *Selection* step
- Seeks to **balance exploration and exploitation** (e.g. avoid getting stuck on local maxima)

UCT – Augmentation of MCTS *Selection*

Choose in each node of the game tree the move for which the expression has the highest value

$$\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$$

Exploitation/ Greedy Term

Decaying Exploration Term

- w_i - number of wins for the node considered after the i^{th} move
- n_i - number of simulations for the node considered after the i^{th} move
- N_i - total number of simulations after the i^{th} move
- c - exploration parameter (theoretically equal to √2; in practice usually chosen empirically)

MCTS Disadvantages

- **Converges very slowly towards minimax** equivalency
- **Incomplete coverage of search space** means that a single losing branch may not be detected. However, this may be obvious to a human expert
  - UCT, and variants, can help with exploration and discovery of possibilities

MCTS Advantages

- **Does (eventually) converge to minimax**
- Minimizes search space (with advantages over minimax+alpha-beta pruning)
- **Does not need an evaluation function** other than recognizing final board win/lose condition state (Incomplete minimax does need eval func)
- Easily applicable to probabilistic zero sum games

On Monte Carlo Tree Search and Reinforcement Learning

Abstract: Fuelled by successes in Computer Go, **Monte Carlo tree search (MCTS) has achieved widespread adoption within the games community**. Its links to traditional reinforcement learning (RL) methods have been outlined in the past; however, the **use of RL techniques within tree search has not been thoroughly studied yet**. In this paper we re-examine in depth this close relation between the two fields; our goal is to improve the cross-awareness between the two communities**. We show that a straightforward adaptation of RL semantics within tree search can lead to a wealth of new algorithms, for which the traditional MCTS is only one of the variants**. We confirm that planning methods inspired by RL in conjunction with online search **demonstrate encouraging results on several classic board games and in arcade video game competitions, where our algorithm recently ranked first**. Our study promotes **a unified view of learning, planning, and search**.

Vodopivec, T., Samothrakis, S., & Ster, B. (2017). On monte carlo tree search and reinforcement learning. *Journal of Artificial Intelligence Research*, *60*, 881-936.

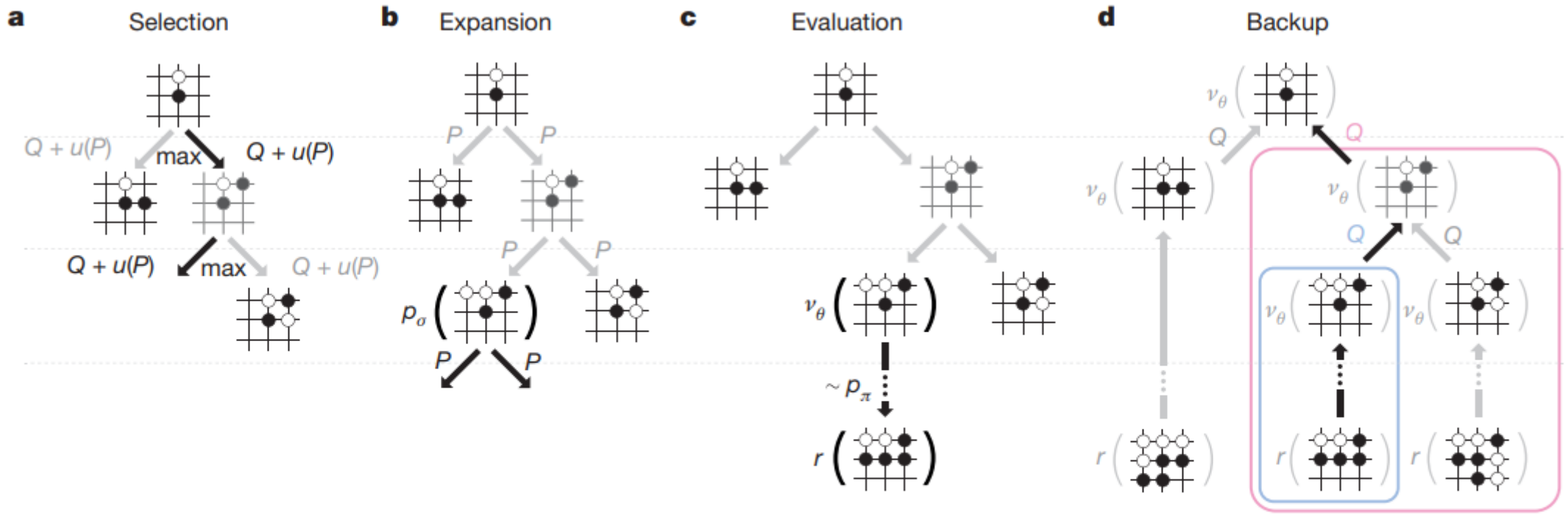


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

“The optimal value function can be computed recursively by minimax (or equivalently negamax) search. Most games are too large for exhaustive minimax tree search; instead, **the game is truncated by using an approximate value function** $v(s) \approx v^*(s)$ in place of terminal rewards. **Depth-first minimax search with alpha–beta pruning has achieved superhuman performance** in chess , checkers and othello, but it has not been effective in Go.”

“Reinforcement learning can learn to approximate the optimal value function directly from games of self-play. …

An alternative approach to minimax search is Monte Carlo tree search (MCTS). … In the limit, **both approximations become exact and MCTS (for example, with UCT) converges to the optimal value function**”

https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

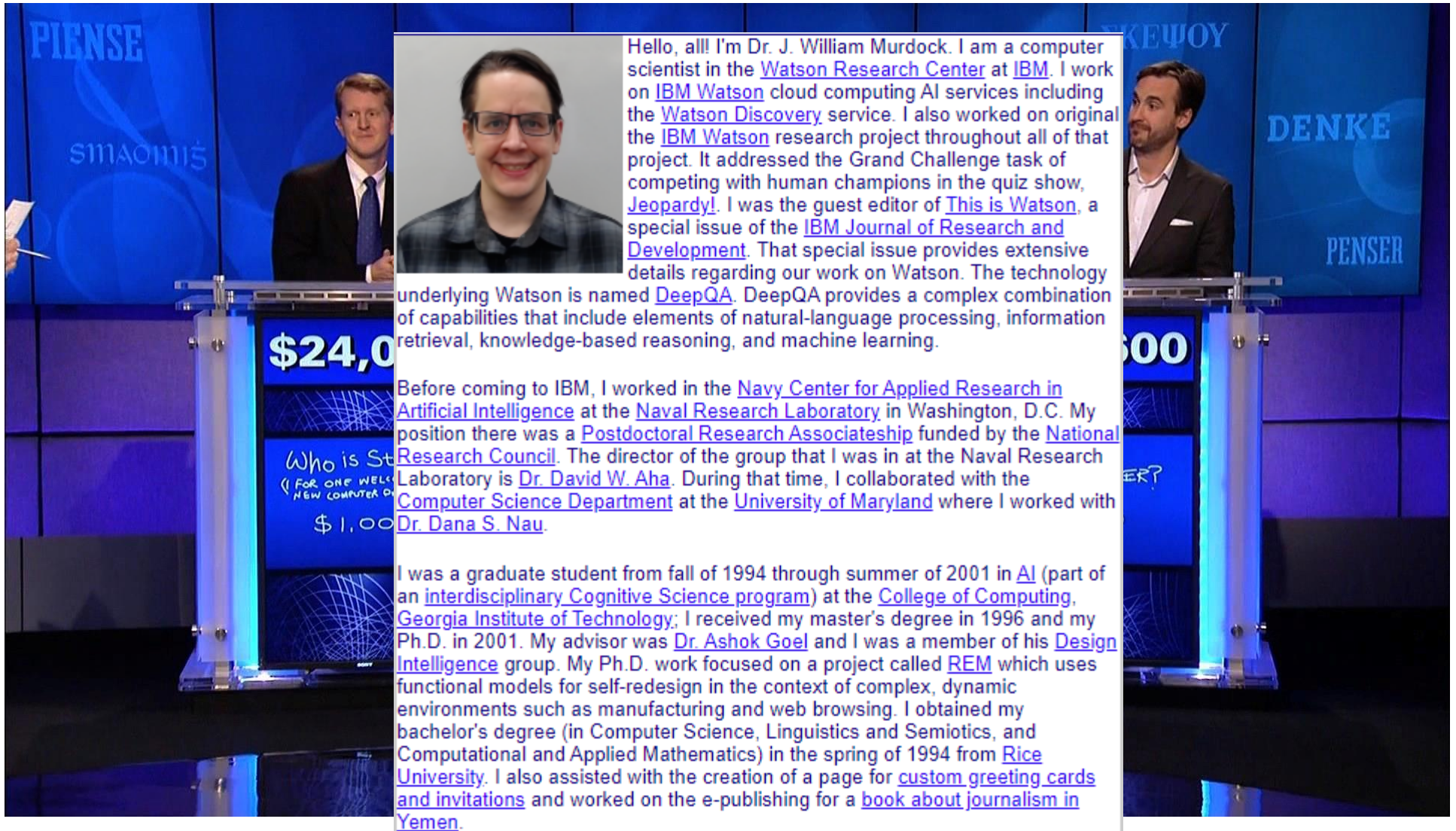
According to (Smith 2016), CBR has led **to more practical applications than any other AI family of techniques with the exception of expert systems and machine learning**. IBM's Watson system (Ferruci et al. 2010) is a famous example of the power of memory-based reasoning.
-- Ashok K. Goel and Belen Diaz-Agudo, AAAI-17. "What's Hot in Case-Based Reasoning"
https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/viewFile/15041/14020

Capturing and reusing exp: Case-Based Reasoning

Videos: CBR in games

- Many Games (Tetris, Soccer, RTS, Poker, …)
  - http://youtu.be/-EPb-zxbEhw
- Xdomain (AAAI 2010 best video)
  - http://www.youtube.com/watch?v=fwYfkCu4mFI
- Imitation in soccer (AAAI 2008 winner)
  - http://www.youtube.com/watch?v=zNjyXLWVSWI
- Football (Casey's Quest)
- The Killer Groove | The Shadow AI of Killer Instinct (@~9:00)
  - https://www.youtube.com/watch?v=Etj5ykJugwU

IBM Watson



Hello, all! I'm Dr. J. William Murdock. I am a computer scientist in the Watson Research Center at IBM. I work on IBM Watson cloud computing AI services including the Watson Discovery service. I also worked on original the IBM Watson research project throughout all of that project. It addressed the Grand Challenge task of competing with human champions in the quiz show, Jeopardy!. I was the guest editor of This is Watson, a special issue of the IBM Journal of Research and Development. That special issue provides extensive details regarding our work on Watson. The technology underlying Watson is named DeepQA. DeepQA provides a complex combination of capabilities that include elements of natural-language processing, information retrieval, knowledge-based reasoning, and machine learning.

Before coming to IBM, I worked in the Navy Center for Applied Research in Artificial Intelligence at the Naval Research Laboratory in Washington, D.C. My position there was a Postdoctoral Research Associateship funded by the National Research Council. The director of the group that I was in at the Naval Research Laboratory is Dr. David W. Aha. During that time, I collaborated with the Computer Science Department at the University of Maryland where I worked with Dr. Dana S. Nau.

I was a graduate student from fall of 1994 through summer of 2001 in AI (part of an interdisciplinary Cognitive Science program) at the College of Computing, Georgia Institute of Technology; I received my master's degree in 1996 and my Ph.D. in 2001. My advisor was Dr. Ashok Goel and I was a member of his Design Intelligence group. My Ph.D. work focused on a project called REM which uses functional models for self-redesign in the context of complex, dynamic environments such as manufacturing and web browsing. I obtained my bachelor's degree (in Computer Science, Linguistics and Semiotics, and Computational and Applied Mathematics) in the spring of 1994 from Rice University. I also assisted with the creation of a page for custom greeting cards and invitations and worked on the e-publishing for a book about journalism in Yemen.

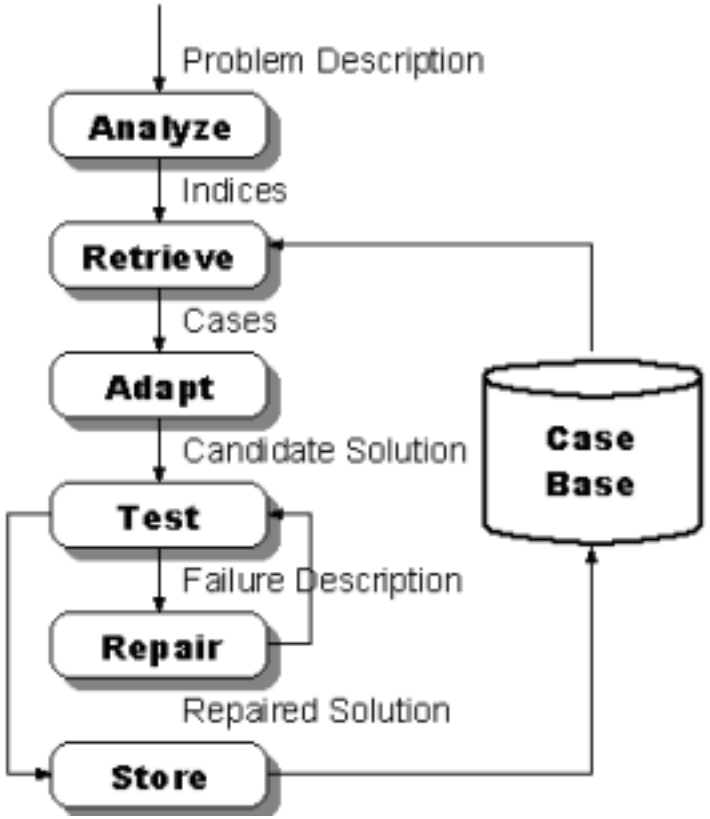
Sources

- **Many(!) slides from Dr. Hector Munoz-Avila**
- cbrwiki.fdi.ucm.es/
- www.iiia.csic.es/People/enric/AICom.html
- www.cse.lehigh.edu/~munoz/CSE335/
- www.aic.nrl.navy.mil/~aha/slides/
- www.csi.ucd.ie/users/barry-smyth
- www.csi.ucd.ie/users/lorraine-mcginty

Alternative

Case-based reasoning

1. Analyze: Identify the current situation
2. Retrieve: The closest case(s) in our case-base
3. Adapt: Change the case(s) to match the current situation
4. Test: Ensure that these cases match our
5. Repair: Make any changes from test (test again if so)
6. Store: Store this new solution in case base



Comparison to other learning techniques

- Machine learning techniques we have seen so far develop a model based on training data before being tested.
- In CBR, the case base constitutes one part of the model
- CBR uses information in the problem to adapt its knowledge base in real time

The Hot Potato

Requires a good chunk of domain knowledge authoring

- State representation
- Distance function
- **Adaptation**
- Revision

Requires a lot of training data or to be be tested iteratively in ways to encourage general learning

Learning can be slow if it gets stuck in a test/repair loop

What the heck is a case?

- Any data structure
- Typical answer: Marvin Minsky Frame
- Frame:
  - Variable slots
  - Values that can go in those slots
- This should sound familiar (CSP)

Overview of Case-Based Reasoning

CBR is […] reasoning by remembering. (Leake 1996)

A case-based reasoner solves new problems by adapting solutions that were used to solve old problems. (Riesbeck & Schank 1989)

CBR is both […] the ways people uses cases to solve problems, and the ways we can make machines use them. (Kolodner 1993)

Bergmann, Survey of CBR, 2000

CBR in one slide

- CBR is a **methodology**
  - to model human reasoning and thinking
  - for building intelligent computer systems
- Basic idea
  - Store known past experiences (cases) in memory (case-base)
  - Given a new problem…
    - Retrieve most similar experience (similarity assessment)
    - Reuse it for the new problem (adaptation)
    - Revise it based on efficacy (feedback)
    - Retain for future use (learning)

CBR: Definition

A problem-solving methodology where solutions to similar, previous problems are reused to solve new problems.

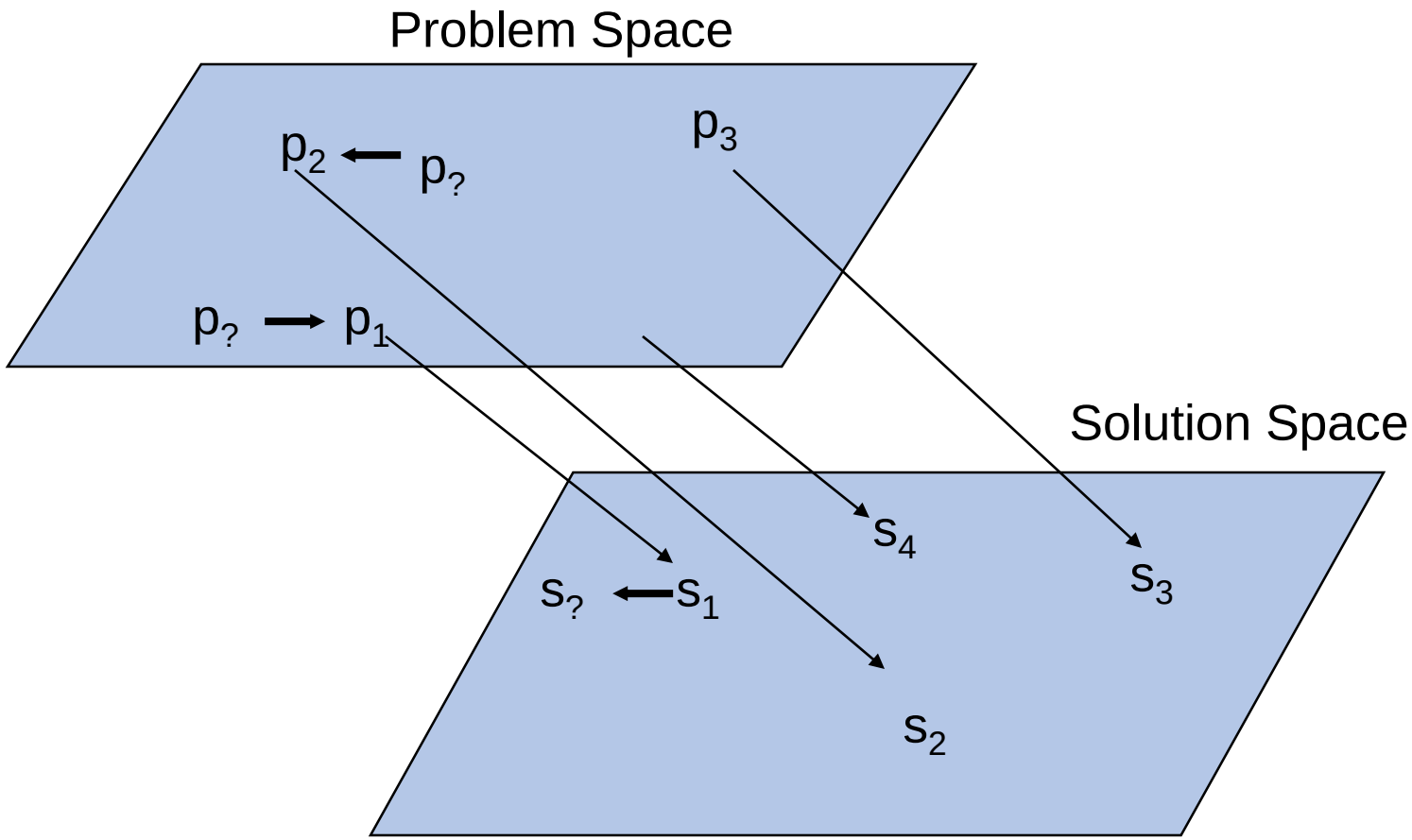
Notes:

- Intuitive
- AI focus (e.g., search, knowledge representation, inference)
- Case = <problem, solution>
- Lazy, incremental, sustained approach to learning

Courtesy of David W. Aha

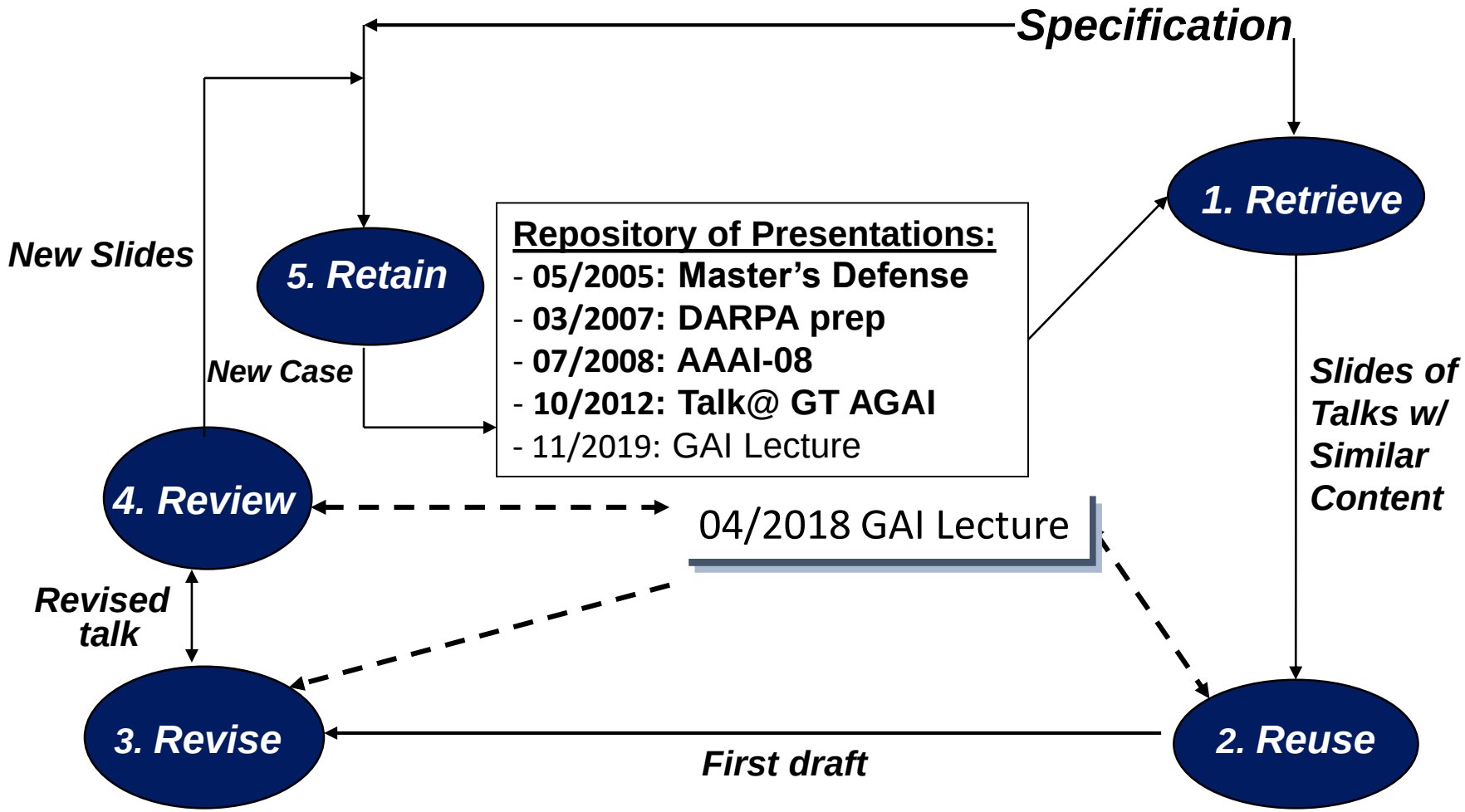
Problem-Solving with CBR

CBR(problem) = solution



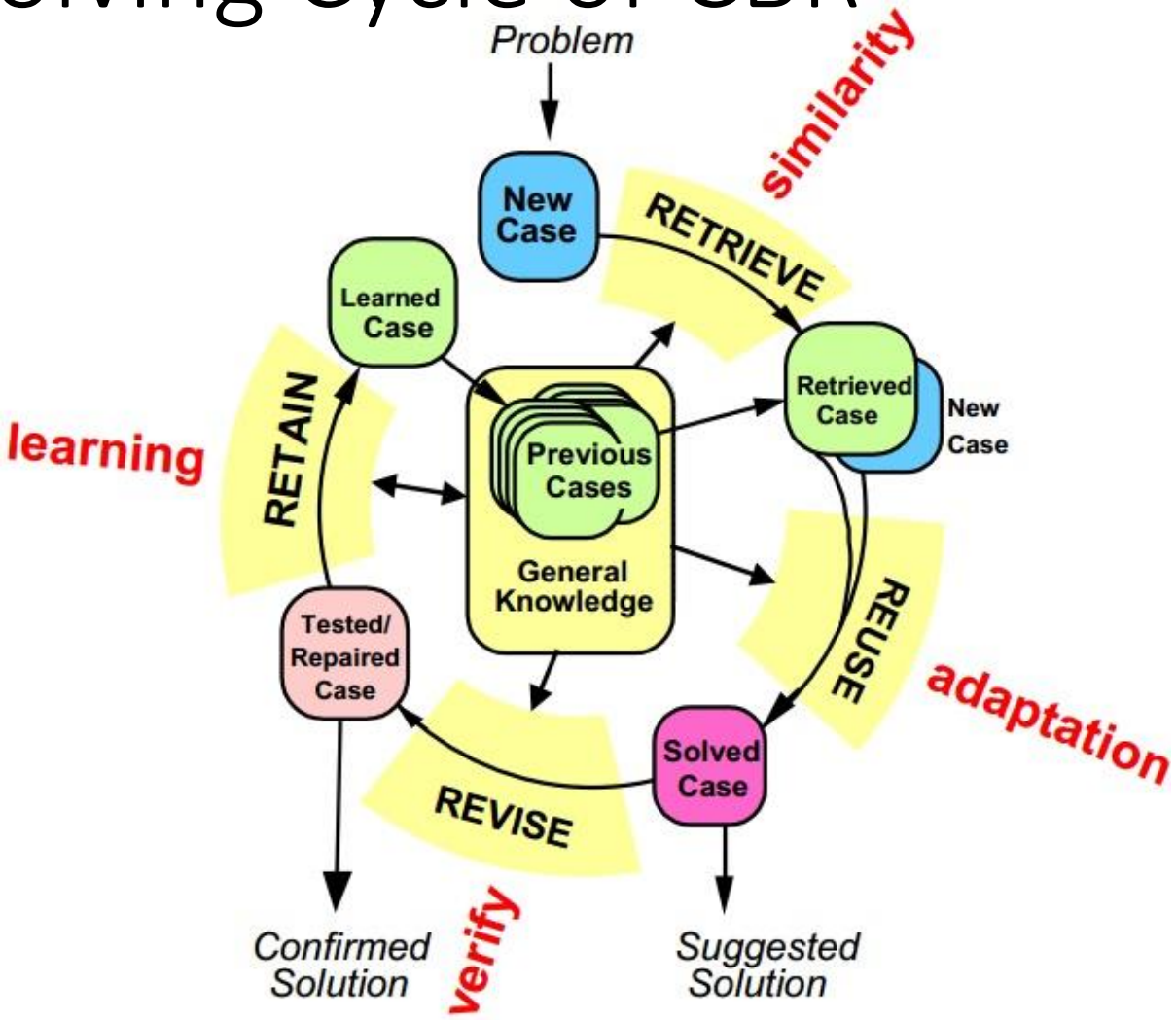
Courtesy of David W. Aha

Example: Slide Creation



Courtesy of David W. Aha

Problem Solving Cycle of CBR



Aamodt & Plaza 1994

Key ideas

- “Similar problems have similar solutions”

- Observations define a new problem
  - Not all feature values must be known
  - A new problem is a case without solution part

- Similarity computation is essential (retrieval)
- Adaptation can be essential (reuse)

CBR: History

1982-1993: Roger Schank's group, initially at Yale
- Modeling cognitive problem solving (Janet Kolodner, 1993)
- New topics: Case adaptation, argument analysis, …

1990: First substantive deployed application (Lockheed)

1991-: Help-desk market niche (Inference/eGain)

1992: Derivational analogy (Veloso, Carbonell); CBP

1993: European emergence (EWCBR'93)

1993-1998: INRECA ESPRIT projects

1995: First international conference (ICCBR'95)
- Knowledge containers (M. Richter)
- First IJCAI Best Paper Award (Smyth & Keane: Competence models)

1997-: Knowledge management / CB maintenance

1999-: e-Commerce

2001-: Recommender Systems

2003-: *Readings in CBR*

2016: International Conference on CBR held at GT

You Have Seen this Before!
(A consumer's Customer Service Experience)

Have you called a customer service support line lately?

It goes something like this (automatic machine):

1. If you want to speak to a sales representative, please press one
2. ….

…

9. If you are experiencing technical difficulties with our wonderful product **Neutronious-L** please press nine

Representing Cases

- Cases contain knowledge about a previous problem solving experiences

- Typically a case contains the following information:
  - **Problem/Situation**
  - **Solution**
  - Adequacy (**utility**)

- Scope of the information:
  - Complete/partial solution
  - Detailed/abstracted solution

- Representation formalism (depends upon domain/task):
  - Attribute-value vector: Case = (V_1, …, V_k, V_{k+1}, …, V_n)
  - Structured representation: Objects, graphs
  - High-order: predicate logic formula, plans

Similarity and Utility in CBR

•The goal of the similarity is to select cases that can be easily adapted to solve a new problem

Similarity = *Prediction* of the utility of the case

•**Utility**: measure of the improvement in efficiency as a result of a body of knowledge

•However:

- ➢ The similarity is an a priori criterion
- ➢ The utility is an a posteriori criterion

• Sample similarity metric: aggregating local similarity metrics, SIM():

- $SIM(V_{1..n}, Y_{1..n}) = \alpha_1 sim_1(V_1, Y_1) + \ldots + \alpha_n sim_n(V_n, Y_n)$
- $sim_i()$ is a local similarity metric, values in [0,1]

Case Retrieval

Problem description:

•**Input**: a collection of cases CB = {C_1, …C_n} and a new problem P

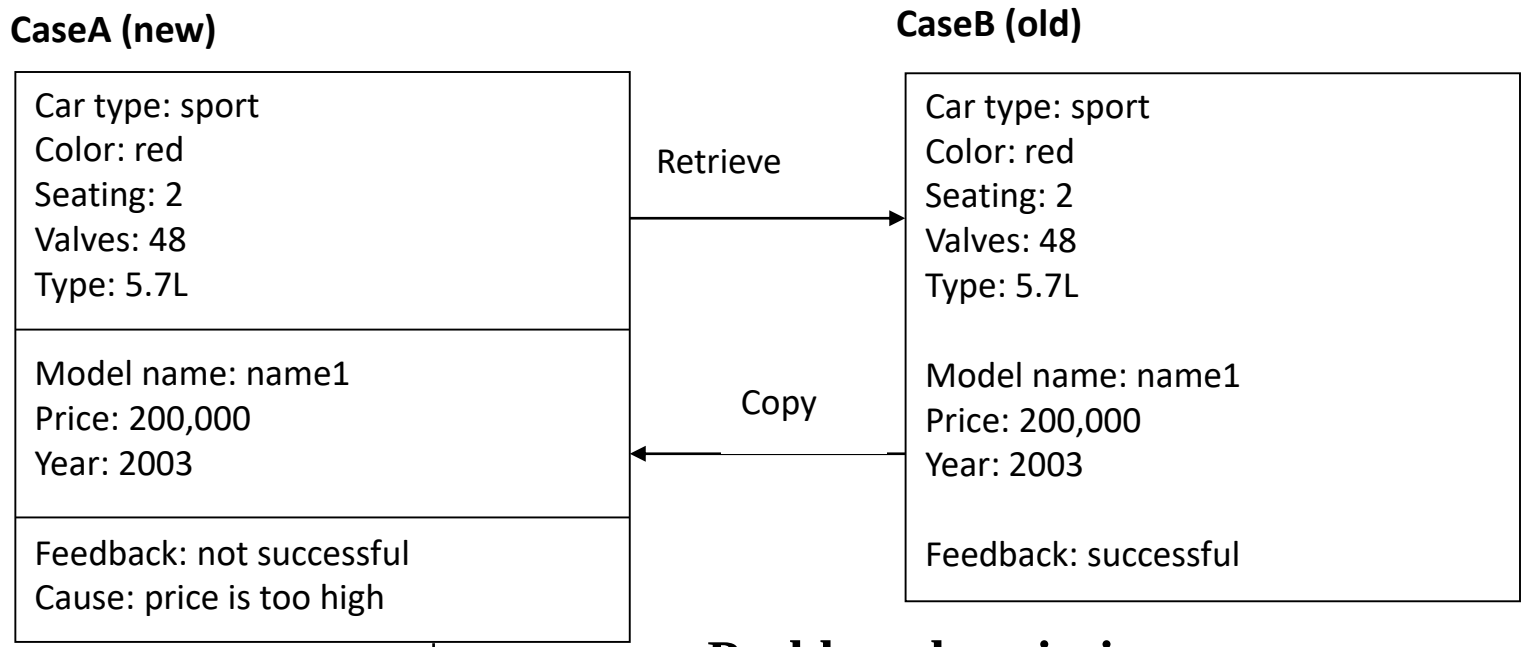
•**Output**:

- The most similar case: A case C_i in CB such that sim(C_i, P) is minimal, *or*
- A collection of *m* most similar cases in CB {C_1,…, C_m}, *or*
- A *sufficiently* similar case: case C_i in CB such that
  sim(C_i, P) > *th*

Solutions:

•Sequential retrieval: O(|CB| × $\log_2$ (k))
•Two-step retrieval: (1) select subset S of cases. (2) Sequential retrieval on S.
•Retrieval with indexed cases

Case Adaptation

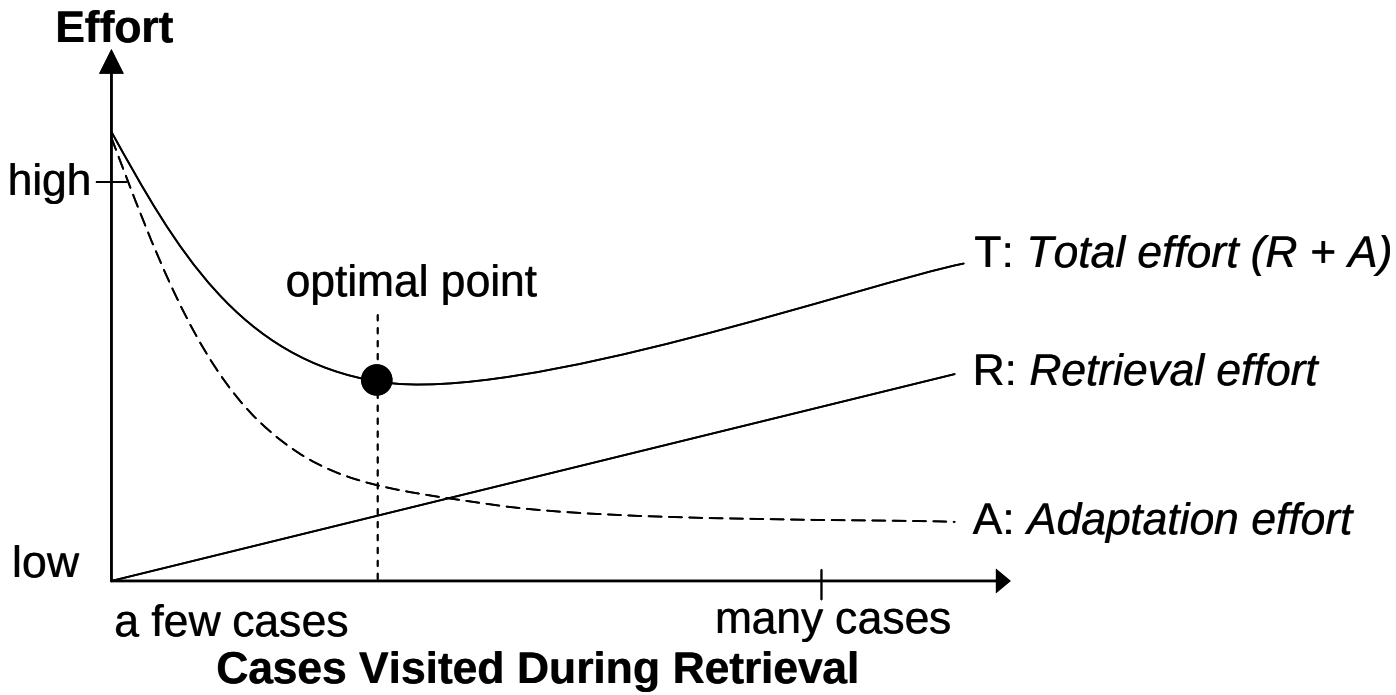


Problem description:

- **Input**: A retrieved case C and a new problem P
- **Output**: A solution for P obtained from C

Considered an open problem

Trade-off between Retrieval and Adaptation Effort

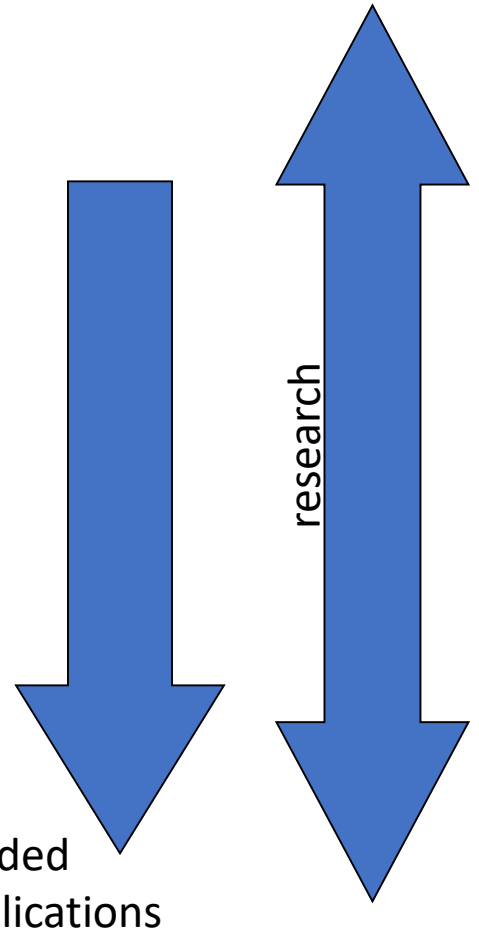


- If little time is spent on retrieval, then the adaptation effort is high
- If too much time is spent on retrieval, then the adaptation effort is low
- There is an optimal intermediate point between these two extremes

Taxonomy of Problem Solving and CBR

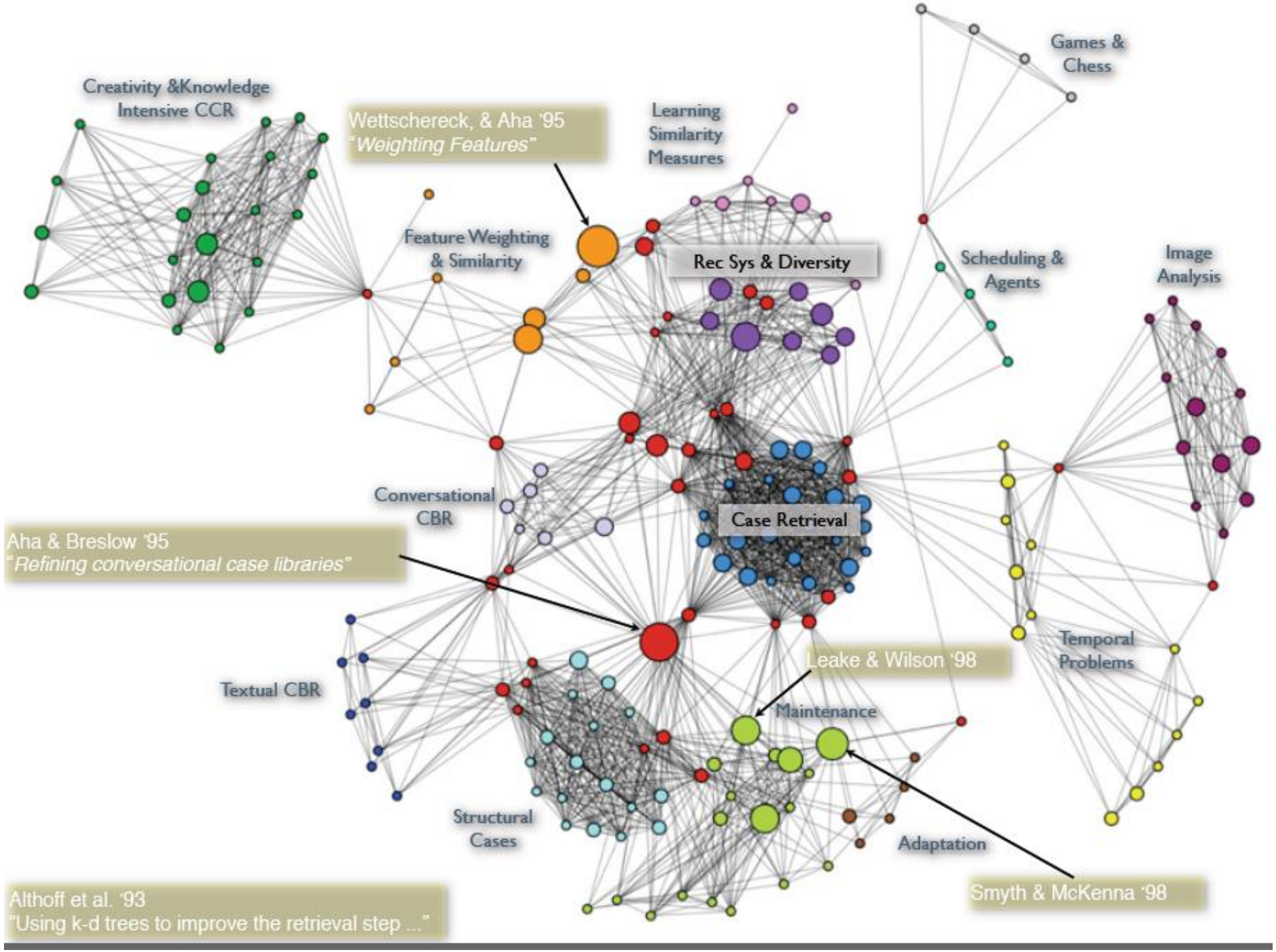
For which of these CBR have been shown to be effective?

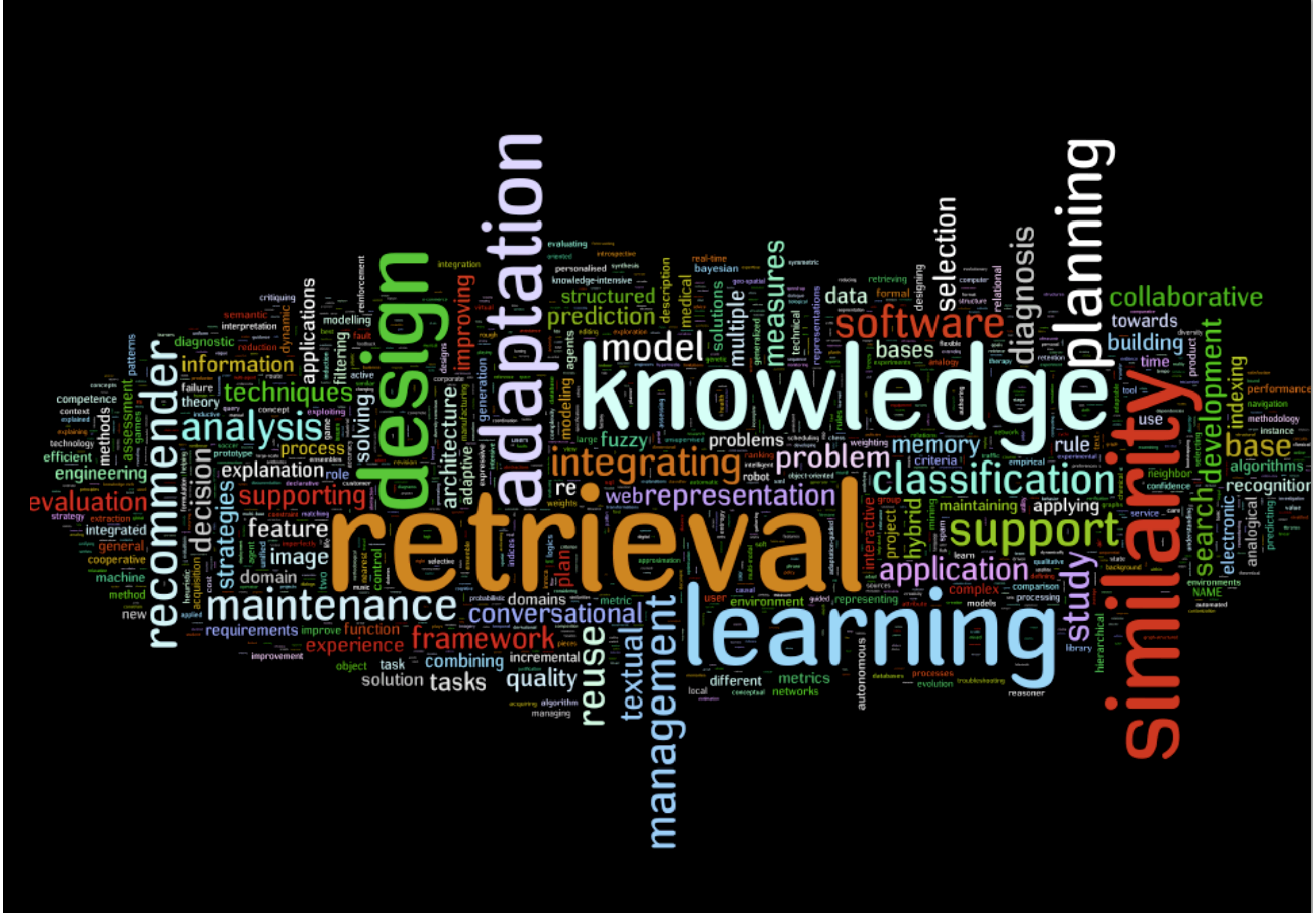
- **Synthesis**:
  - constructing a solution
  - Methods: planning, configuration
- **Analysis**:
  - interpreting a solution
  - Methods: classification, diagnosis



Main Topics of CBR Research ~ 10yr

- Study by Derek Greene, Jill Freyne, Barry Smyth, Pádraig Cunningham
- Social network analysis based on co-citations links
- Sources:
  - Bibliographic data from Springer about ICCBR, ECCBR
  - Citation data from Google scholar
- **An Analysis of Research Themes in the CBR Conference Literature**. ECCBR'08
- Next two slides from http://mlg.ucd.ie/cbr


Games &
Chess
Creativity &Knowledge
Intensive CCR
Wettschereck, & Aha '95
"Weighting Features"
Learning
Similarity
Measures
Feature Weighting
& Similarity
Rec Sys & Diversity
Scheduling &
Agents
Image
Analysis
Conversational
CBR
Case Retrieval
Aha & Breslow '95
"Refining conversational case libraries"
Temporal
Problems
Leake & Wilson '98
Textual CBR
Maintenance
Structural
Cases
Adaptation
Smyth & McKenna '98
Althoff et al. '93
"Using k-d trees to improve the retrieval step ..."



retrieval
knowledge
similarity
learning
adaptation
design
planning
recommender
maintenance
software
classification
support
integrating
representation
application
management
reuse
textual
diagnosis
selection
model
analysis
collaborative
prediction
structured
techniques
information
evaluation
supporting
framework
conversational
problem
memory
measures
decision
strategies
feature
image
domain
explanation
process
experience
combining
quality
tasks
solution
study
development
search
electronic
base
algorithms
recognition
towards
building
bases
data
multiple
fuzzy
problems
web
hybrid
interactive
project
maintaining
applying
criteria
rule
indexing
performance
engineering
efficient
competence
machine
method
new
cooperative
general
integrated
requirements
function
improve
incremental
different
metrics
networks
environment
representing
domains
applications
filtering
solving
architecture
adaptive
improving
generation
agents
modeling
solutions
medical
description
time
use
navigation
methodology
instance
neighbor
analogical
predicting
hierarchical
library
autonomous
reasoner
complex
processing
comparison
models
diagnostic
semantic
critiquing
interpretation
modelling
reinforcement
theory
failure
concept
context
technology
methods
assessment
games
patterns
extraction
strategy
task
object
improvement
re
robot
weighting
empirical
formal
bayesian
personalised
real-time
evaluating
integration
knowledge-intensive

Major Themes in CBR

- Recommender systems and diversity
- Case-Based Maintenance
- Case Retrieval
- Learning similarity measures
- Adaptation
- Image analysis
- Textual & Conversational CBR
- Feature weighting and similarity

http://www.csi.ucd.ie/files/ucd-csi-2009-03.pdf

Some Interrelations between Topics

- Retrieval
  - Information gain
  - Similarity metrics
  - Indexing
- Reuse
  - Rule-based systems
  - Plan Adaptation
- Revise & Review
  - Constraint-satisfaction systems
- Retain
  - Induction of decision trees

Focus Point: Diversity in CBR

Traditional Retrieval Approach

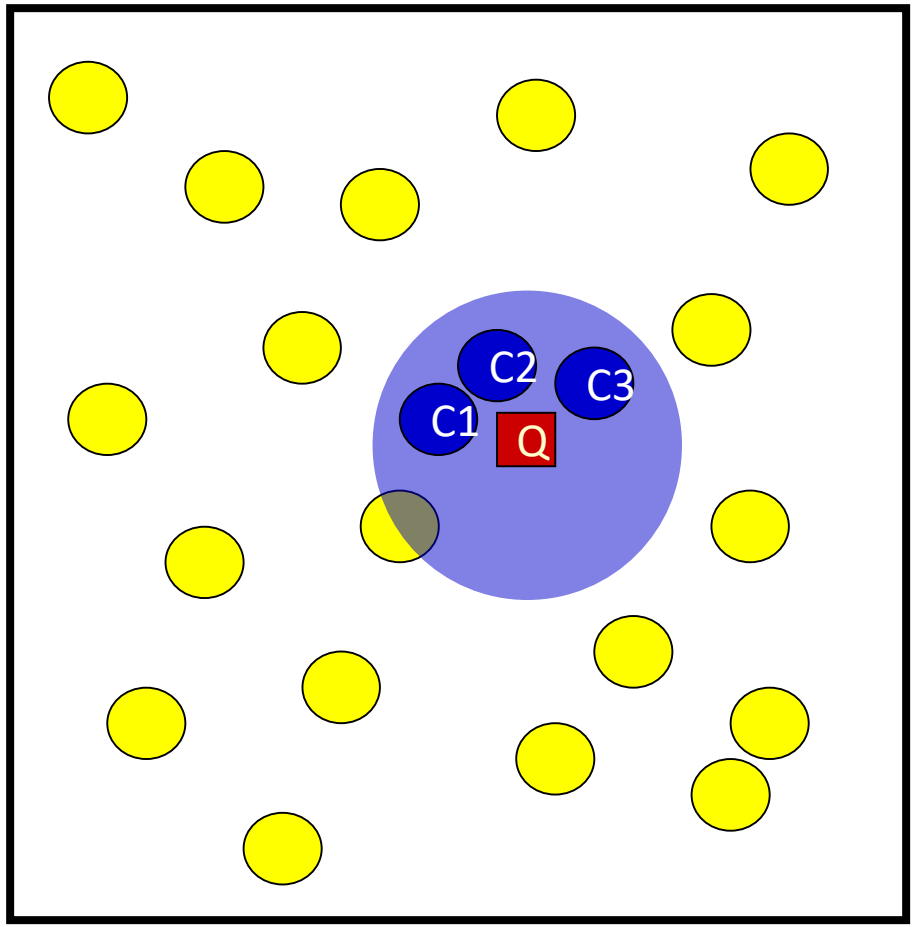
- Similarity-Based Retrieval
  - Select the *k* most similar items to the current query.

Query

Available case

Similar case

- Problem
  - Vague queries.
  - Limited coverage of search space in every cycle of the dialogue.



Lorraine McGinty and Barry Smyth
Department of Computer Science, University College Dublin

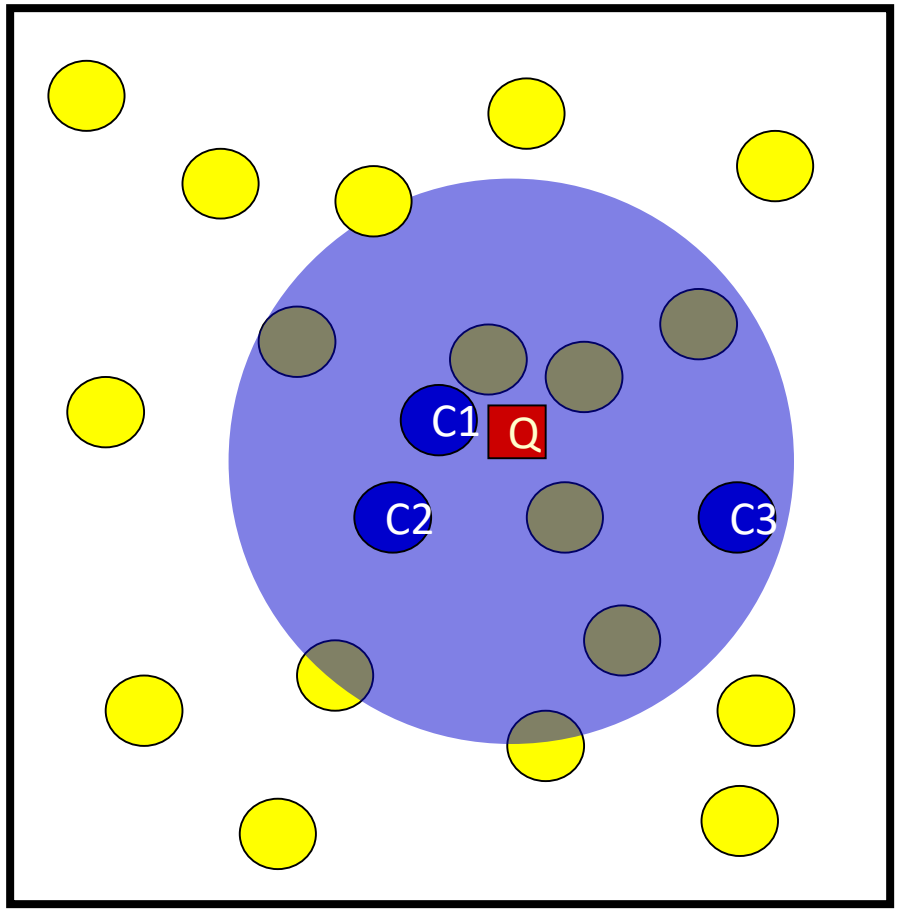
Diversity Enhancement

- Diversity-Enhanced Retrieval
  - Select *k* items such that they are both **similar to the current query but different from each other**.

    Query

    Available case

    Retrieved case
  - Providing a wider choice allows for broader coverage of the product space.
  - Allows many less relevant items to be eliminated.



Lorraine McGinty and Barry Smyth
Department of Computer Science, University College Dublin

Focus Point: Augmenting General Knowledge with Cases

Why Augment General Knowledge With Cases?

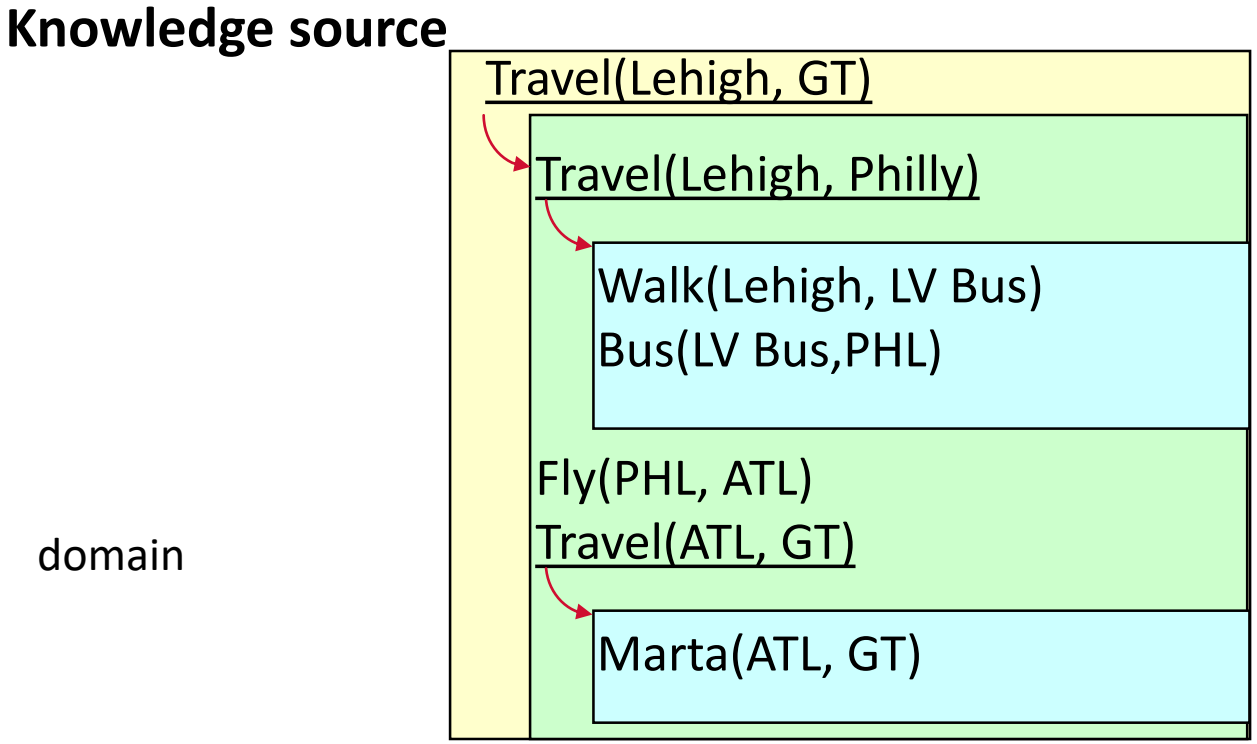
- In many practical applications, **encoding complete domain knowledge is unpractical/infeasible and episodic knowledge is available**

  Example: Some kinds of military operations where two kinds of knowledge are available:

  - **General guidelines and standard operational procedures** which can be encoded as a (partial) general domain knowledge
  - Whole compendium of **actual operations and exercises** which can be captured as cases

Hierarchical Problem Solving

Hierarchical case-based planning techniques combine domain knowledge and episodic knowledge (cases)



Knowledge Sources

General

Methods denote generic task decompositions and *conditions* for selecting those decompositions:

Task: travel(?A,?B)

Decomposition:

travelC(?A, ?Airp1)
travelIC(?Airp1,?Airp2)
travelC(?Airp2, ?B)

Conditions:

in(?A,?City1)
in(?B,?City2)
airport(?Airp1,?City1)
airport(?Airp2,?City2)

Episodic

Cases denote concrete task decompositions:

Task: travelC(Lehigh, PHL)

Decomposition:

take(bus, Lehigh, PHL)

Conditions:

enoughMoney()

CBR: Final Remarks

Advantages of CBR

- Reduces knowledge acquisition effort
- Requires less maintenance effort
- Reuse of solutions improves problem solving performance
- Can make use of existing data
- Improves over time, adapts to changes
- Has enjoyed high user acceptance

Bergmann, Survey of CBR, 2000

Why not cbr?

In fact, this is the crux of the argument: if you have a good scripting language, or even a visual tree editor to capture sequences, **you'll be orders of magnitude more productive (and more reliable) than an expert trying indirectly to get the system to induce specific sequences from examples**. As such, it's fair to claim that CBR isn't particularly well suited to these kinds of problems in game AI.

(2008) http://aigamedev.com/open/editorial/critique-case-based-reasoning/

Past uses at GT for Digital Entertainment

Dialog

- Sanjeet Hajarnis, Christina Leber, Hua Ai, Mark O. Riedl, and Ashwin Ram (2011). **A Case Base Planning Approach for Dialogue Generation in Digital Movie Design**. Proceedings of the 19th International Conference on Case Based Reasoning, London, UK.

Decision making

- Santiago Ontañón and Ashwin Ram (2011) **Case-Based Reasoning and User-Generated AI for Real-Time Strategy Games**. In Pedro Antonio González-Calero and Marco Antonio Gómez-Martín (Editors), Artificial Intelligence for Computer Games, pp. 103-124. Springer-Verlag.

Drama Management

- Manu Sharma and Santiago Ontañón and Manish Mehta and Ashwin Ram (2010) **Drama Management and Player Modeling for Interactive Fiction Games**, in Computational Intelligence Journal, Volume 26 Issue 2, pp. 183-211.

PCG

- Manish Mehta and Santiago Ontañón and Ashwin Ram (2008) **Adaptive Computer Games: Easing the Authorial Burden**. in Steve Rabin (Editor), AI Game Programming Wisdom 4. pp. 617-632

Other applications in games

- Gillespie, K., Karneeb, J., Lee-Urban, S., and Munoz-Avila, H. (2010) **Imitating Inscrutable Enemies: Learning from Stochastic Policy Observation, Retrieval and Reuse**. Proceedings of the 18th International Conference on Case Based Reasoning (ICCBR 2010). AAAI Press.
- Auslander, B., Lee-Urban, S., Hogg, C., and Munoz-Avila, H. (2008**) Recognizing The Enemy: Combining Reinforcement Learning with Strategy Selection using Case-Based Reasoning**. Proceedings of the 9th European Conference on Case-Based Reasoning (ECCBR-08).
- Hogg, C., Lee-Urban, S., Auslander, B., and Munoz-Avila, H. (2008) **Discovering Feature Weights for Feature-Based Indexing of Q-Tables**. Proceedings of the Uncertainty and Knowledge Discovery in CBR Workshop at the 9th European Conference on Case-Based Reasoning (ECCBR-08).

CBR: Takeaway

1. Sometimes natural (e.g., law, diagnosis)

2. Cases simplify knowledge acquisition
   - Easier to obtain than rules
   - Captures/shares people's experiences

3. Good for some types of tasks
   - When perfect models are not available
     - Faulty equipment diagnosis
     - Online sales
     - Legal reasoning
     - Games

4. Commercial applications
   - Help-desk systems (e.g., Inference corp.: +700 clients)

5. Similar problems have similar solutions.
   - Retrieve, Reuse, Revise, Retain

Questions?

- http://cbrwiki.fdi.ucm.es/
- http://aitopics.net/CaseBasedReasoning
- http://www.cse.lehigh.edu/~munoz/CSE335/
- http://mlg.ucd.ie/cbr

- http://gaia.fdi.ucm.es/research/colibri/jcolibri

CBR: Recap

1) What are the 4 processes, each beginning with an "R", commonly used to describe the CBR methodology?

2) The _____ metric is used to find the problem/solution pair in the casebase most related to the new problem, by comparing the relatedness of the features of the new problem to the features of known problems in the casebase.

3) In case-based reasoning, problem solving cannot commence without the ability to compute this, which is a number indicating how related an existing case is to the new problem.

4) A foundational assumption in CBR is that "Similar problems have _____ _____".