Lecture 5: Search 4

http://cs.nju.edu.cn/yuy/course_ai16.ashx
Previously...

Path-based search

Uninformed search
- Depth-first, breadth first, uniform-cost search

Informed search
- Best-first, A* search

Adversarial search
- Alpha-Beta search
Beyond classical search

Bandit search

Tree search: Monte-Carlo Tree Search

General search:
  Gradient decent
  Metaheuristic search
Multiple arms
Each arm has an expected reward, but unknown, with an unknown distribution

Maximize your award in fixed trials
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

problem?
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

problem? waste on suboptimal arms
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

problem? waste on suboptimal arms

Exploitation-only:
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

problem? waste on suboptimal arms

Exploitation-only:
1. try each arm once
2. try the observed best arm $T-K$ times
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

Exploitation-only:
1. try each arm once
2. try the observed best arm $T-K$ times

problem? waste on suboptimal arms
Simplest strategies

Two simplest strategies

Exploration-only:
for $T$ trails and $K$ arms, try each arm $T/K$ times

problem? waste on suboptimal arms

Exploitation-only:
1. try each arm once
2. try the observed best arm $T-K$ times

problem? risk of wrong best arm
Balance the exploration and exploitation:

with \( \varepsilon \) probability, try a random arm
with \( 1 - \varepsilon \) probability, try the best arm

\( \varepsilon \) controls the balance
Balance the exploration and exploitation:
Choose arm with probability

\[ P(k) = \frac{e^{\frac{Q(k)}{\tau}}}{\sum_{i=1}^{K} e^{\frac{Q(i)}{\tau}}} \]  \hspace{1cm} (16.4)

\( \tau \) controls the balance
Balance the exploration and exploitation:
Choose arm with the largest value of

\[ Q(k) + \sqrt{\frac{2 \ln n}{n_k}}, \]

Upper-confidence bound
Gradually grow the search tree:

- **Iterate Tree-Walk**
  - **Building Blocks**
    - Select next action
  - **Add a node**
    - Grow a leaf of the search tree
  - Select next action bis
    - Random phase, roll-out
  - Compute instant reward
    - Evaluate
  - Update information in visited nodes
    - Propagate

- **Returned solution:**
  - Path visited most often
Monte-Carlo Tree Search

Example:

How to select the leave? As bandit

Pic from https://en.wikipedia.org/wiki/Monte_Carlo_tree_search#cite_note-Kocsis-Szepesvari-5
Monte-Carlo Tree Search

```java
public class TreeNode {
    static Random r = new Random();
    static int nActions = 5;
    static double epsilon = 1e-6;

    TreeNode[] children;
    double nVisits, totValue;

    public void expand() {
        children = new TreeNode[nActions];
        for (int i=0; i<nActions; i++) {
            children[i] = new TreeNode();
        }
    }

    public void selectAction() {
        List<TreeNode> visited = new LinkedList<TreeNode>();
        TreeNode cur = this;
        visited.add(this);
        while (!cur.isLeaf()) {
            cur = cur.select();
            visited.add(cur);
        }
        cur.expand();
        TreeNode newNode = cur.select();
        visited.add(newNode);
        double value = rollout(newNode);
        for (TreeNode node : visited) {
            // would need extra logic for n-player game
            node.updateStats(value);
        }
    }

    public void updateStats(double value) {
        nVisits++;
        totValue += value;
    }
}
```

codes from http://mcts.ai/code/java.html
Monte-Carlo Tree Search

codes from http://mcts.ai/code/java.html
Monte-Carlo Tree Search

optimal? Yes, after infinite tries

compare with alpha-beta pruning
no need of heuristic function
Monte-Carlo Tree Search

Improving random rollout

Monte-Carlo-based

1. Until the goban is filled, add a stone (black or white in turn) at a uniformly selected empty position
2. Compute $r = \text{Win}($black$)$
3. The outcome of the tree-walk is $r$

Improvements ?

- Put stones randomly in the neighborhood of a previous stone
- Put stones matching patterns
- Put stones optimizing a value function

Silver et al. 07

Silver et al. 07
General search
Greedy idea in continuous space

Suppose we want to site three airports in Romania:
- 6-D state space defined by \((x_1, y_2), (x_2, y_2), (x_3, y_3)\)
- objective function \(f(x_1, y_2, x_2, y_2, x_3, y_3) = \text{sum of squared distances from each city to nearest airport}\)
Greedy idea in continuous space

discretize and use hill climbing
Greedy idea in continuous space

gradient decent

– 6-D state space defined by \((x_1, y_2), (x_2, y_2), (x_3, y_3)\)
– objective function \(f(x_1, y_2, x_2, y_2, x_3, y_3) = \) sum of squared distances from each city to nearest airport

Gradient methods compute

\[
\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)
\]

to increase/reduce \(f\), e.g., by \(x \leftarrow x + \alpha \nabla f(x)\)

1-order method
Greedy idea in continuous space

gradient decent

- 6-D state space defined by \((x_1, y_2), (x_2, y_2), (x_3, y_3)\)
- objective function \(f(x_1, y_2, x_2, y_2, x_3, y_3) = \text{sum of squared distances from each city to nearest airport}\)

Sometimes can solve for \(\nabla f(x) = 0\) exactly (e.g., with one city). Newton–Raphson (1664, 1690) iterates \(x \leftarrow x - H_f^{-1}(x) \nabla f(x)\) to solve \(\nabla f(x) = 0\), where \(H_{ij} = \partial^2 f / \partial x_i \partial x_j\)

2-order method

Taylor's series:

\[
f(x) = f(a) + (x - a)f'(a) + \frac{(x - a)^2}{2} f''(a) + \cdots = \sum_{i=0}^{\infty} \frac{(x - a)^i}{i!} f^{(i)}(a).
\]
Greedy idea

1st and 2nd order methods may not find global optimal solutions

they work for convex functions
Meta-heuristics

“problem independent
“black-box
“zeroth-order method
...

and usually inspired from nature phenomenon
Simulated annealing

temperature from high to low

when high temperature, form the shape
when low temperature, polish the detail
Simulated annealing

Idea: escape local maxima by allowing some “bad” moves but gradually decrease their size and frequency

function SIMULATED-ANNEALING (problem, schedule) returns a solution state
inputs: problem, a problem
        schedule, a mapping from time to “temperature”
local variables: current, a node
                next, a node
                T, a “temperature” controlling prob. of downward steps

current ← MAKE-NODE(Initial-State[problem])

for t ← 1 to ∞ do
    T ← schedule[t]
    if T = 0 then return current
    next ← a randomly selected successor of current
    ΔE ← VALUE[next] − VALUE[current]
    if ΔE > 0 then current ← next
    else current ← next only with probability \( e^{\frac{\Delta E}{T}} \)

the neighborhood range shrinks with T
the probability of accepting a bad solution decreases with T
Simulated annealing

a demo

Local beam search

Idea: keep \( k \) states instead of 1; choose top \( k \) of all their successors

Not the same as \( k \) searches run in parallel!
Searches that find good states recruit other searches to join them

Problem: quite often, all \( k \) states end up on same local hill

Idea: choose \( k \) successors randomly, biased towards good ones

Observe the close analogy to natural selection!
Genetic algorithm

A simulation of Darwin’s evolutionary theory

Over-reproduction with diversity
Nature selection

Diagram:
- Random initialization
- Parent population
- Reproduction
- Evaluation
- Selection
- Offspring solutions
- Evaluated offspring solutions
Genetic algorithm

Encode a solution as a vector,

1: $Pop \leftarrow n$ randomly drawn solutions from $X$
2: for $t=1,2,\ldots$ do
3: $Pop^m \leftarrow \{\text{mutate}(s) \mid \forall s \in Pop\}$, the mutated solutions
4: $Pop^c \leftarrow \{\text{crossover}(s_1, s_2) \mid \exists s_1, s_2 \in Pop^m\}$, the recombined solutions
5: evaluate every solution in $Pop^c$ by $f(s)(\forall s \in Pop^c)$
6: $Pop^s \leftarrow$ selected solutions from $Pop$ and $Pop^c$
7: $Pop \leftarrow Pop^s$
8: terminate if meets a stopping criterion
9: end for
Genetic algorithms = stochastic local beam search + generate successors from pairs of states

<table>
<thead>
<tr>
<th>Fitness</th>
<th>Selection</th>
<th>Pairs</th>
<th>Cross–Over</th>
<th>Mutation</th>
</tr>
</thead>
<tbody>
<tr>
<td>24748552</td>
<td>24 31%</td>
<td>32752411</td>
<td>32748552</td>
<td>32748[1]52</td>
</tr>
<tr>
<td>32752411</td>
<td>23 29%</td>
<td>24748552</td>
<td>24752411</td>
<td>24752411</td>
</tr>
<tr>
<td>24415124</td>
<td>20 26%</td>
<td>32752411</td>
<td>32752124</td>
<td>32[2]52124</td>
</tr>
<tr>
<td>32543213</td>
<td>11 14%</td>
<td>24415124</td>
<td>24415411</td>
<td>2441541[7]</td>
</tr>
</tbody>
</table>

GAs require states encoded as strings (GPs use programs)

Crossover helps iff substrings are meaningful components
Example

Encode a solution as a vector with length $n$
each element of the vector can be chosen from $\{1,\ldots,V\}$
parameters: mutation probability $p_m$, crossover probability $p_c$

1: $Pop = \text{randomly generate } n \text{ solutions from } \{1,\ldots,V\}^n$
2: for $t=1,2, \ldots$ do
3: \hspace{1em} $Pop^m=\text{emptyset}, \ Pop^c=\text{emptyset}$
4: \hspace{1em} for $i = 1 \text{ to } n$
5: \hspace{2em} let $x$ be the $i$-th solution in $Pop$
6: \hspace{2em} for $j = 1 \text{ to } n$: with probability $p_m$, change $x_j$ by a random value from $\{1,\ldots,V\}$
7: \hspace{2em} add $x$ into $Pop^m$
8: \hspace{1em} end for
9: \hspace{1em} for $i = 1 \text{ to } n$
10: \hspace{2em} let $x$ be the $i$-th solution in $Pop^m$
11: \hspace{2em} let $x'$ be a randomly selected solution from $Pop^m$
12: \hspace{2em} with probability $p_c$, exchange a random part of $x$ with $x'$
13: \hspace{2em} add $x$ into $Pop^c$
14: \hspace{1em} end for
15: \hspace{1em} evaluate solutions in $Pop^c$, select the best $n$ solutions from $Pop$ and $Pop^c$ to $Pop$
16: \hspace{1em} terminal if a good solution is found
17: end for
An evolutionary of virtual life
An evolutionary of virtual life
Properties of meta-heuristics

zeroth order

do not need differentiable functions

convergence

will find an optimal solution if \( P( x^* | x ) > 0 \)
or \( P( x \rightarrow x_1 \rightarrow \ldots \rightarrow x_k \rightarrow x^* ) > 0 \)

a missing link

observation \[\rightarrow\] simulation
Properties of meta-heuristics

zeroth order

do not need differentiable functions

convergence

will find an optimal solution if \( P( x^* | x ) > 0 \)
or \( P( x \rightarrow x_1 \rightarrow ... \rightarrow x_k \rightarrow x^* ) > 0 \)

a missing link
Properties of meta-heuristics

- Genetic Algorithms
- Evolutionary Programming
- Evolutionary Strategies
- Ant Colony Optimization Algorithms
- Particle Swarm Optimization Algorithms
- Artificial Bee Colony Algorithms
- Artificial Immune Systems
- Simulated Annealing
- Bat Algorithm
- Grey Wolf Optimizer
- Fireworks Algorithm
- Brainstorm Algorithm
- Gravitational Search Algorithm
- River Formation Dynamics
- Differential Evolution
- Memetic Algorithms
- Cultural Algorithms
- Tabu Search
- Cultural Algorithms
- Intelligent Water Drops Algorithm

Year:
- 2010
- 2000
- 1990
- 1980
- 1970
- 1960
Example

*hard to apply traditional optimization methods but easy to test a given solution*

**Representation:**

*parameterize*

represented as a vector of parameters

**Fitness:**

*test by simulation/experiment*
Example

Series 700

Series N700

Technological overview of the next generation Shinkansen high-speed train Series N700

M. Ueno¹, S. Usui¹, H. Tanaka¹, A. Watanabe²

¹Central Japan Railway Company, Tokyo, Japan, ²West Japan Railway Company, Osaka, Japan

Abstract

In March 2005, Central Japan Railway Company (JR Central) has completed prototype trial run of the Series N700, the next generation Shinkansen to high-speed train. Various induced pressure waves and other issues related to environmental compatibility such as external noise. To combat this, an aero double-wing-type has been adopted for nose shape (Fig. 3). This nose shape, which boasts the most appropriate aerodynamic performance, has been newly developed for railway rolling stock using the latest analytical technique (i.e. genetic algorithms) used to develop the main wings of airplanes. The shape resembles a bird in flight, suggesting a feeling of high-speed and speed.

On the Tokaido Shinkansen line, Series N700 cars save 19% energy than Series 700 cars, despite a 30% increase in the output of their traction equipment for higher-speed operation (Fig. 4).

This is a result of adopting the aerodynamically excellent nose shape, reduced running resistance thanks to the drastically smoothened car body and under-floor equipment, effective

this nose ... has been newly developed ... using the latest analytical technique (i.e. genetic algorithms)

N700 cars save 19% energy ... 30% increase in the output... This is a result of adopting the ... nose shape
Example

NASA ST5 satellite

hard to apply traditional optimization methods
but easy to test a given solution
Example

NASA ST5 satellite

hard to apply traditional optimization methods but easy to test a given solution

Computer-Automated Evolution of an X-Band Antenna for NASA’s Space Technology 5 Mission

Gregory, S. Hornby  
Mail Stop 269-3, University Affiliated Research Center, UC Santa Cruz, Moffett Field, CA, 94035, USA

Jason D. Lohn  
Carnegie Mellon University, Mail Stop 23-11, Moffett Field, CA 94035, USA

Derek S. Linden  
JEM Engineering, 8683 Cherry Lane, Laurel, MD 20707, USA

Since there are two antennas on each spacecraft, and not just one, it is important to measure the overall gain pattern with two antennas mounted on the spacecraft. For this, different combinations of the two evolved antennas and the QHA were tried on the ST5 mock-up and measured in an anechoic chamber. With two QHAs 38% efficiency was achieved, using a QHA with an evolved antenna resulted in 80% efficiency, and using two evolved antennas resulted in 93% efficiency. Here “efficiency” means how much power is being radiated versus how much power is being eaten up in resistance, with greater efficiency resulting in a stronger signal and greater range. Figure 11
Example

NASA ST5 satellite

QHAs (□ □) 38% efficiency evolved antennas resulted in 93% efficiency

Since there are two antennas on each spacecraft, and not just one, it is important to measure the overall gain pattern with two antennas mounted on the spacecraft. For this, different combinations of the two evolved antennas and the QHA were tried on the ST5 mock-up and measured in an anechoic chamber. With two QHAs 38% efficiency was achieved, using a QHA with an evolved antenna resulted in 80% efficiency, and using two evolved antennas resulted in 93% efficiency. Here “efficiency” means how much power is being radiated versus how much power is being eaten up in resistance, with greater efficiency resulting in a stronger signal and greater range. Figure 11
Different Environment Properties
Nondeterministic actions

In the **erratic vacuum world**, the *Suck* action works as follows:

- When applied to a dirty square the action cleans the square and sometimes cleans up dirt in an adjacent square, too.
- When applied to a clean square the action sometimes deposits dirt on the carpet.⁹

![Diagram of the erratic vacuum world with states 1 to 8]

almost all real-world problems are nondeterministic

*how do you solve this problem?*
AND-OR tree search

OR node: different actions (as usual)
AND node: different transitions

- **OR node**: different actions (as usual)
- **AND node**: different transitions

**Figure 4.10** The first two levels of the search tree for the erratic vacuum world. State nodes are OR nodes where some action must be chosen. At the AND nodes, shown as circles, every outcome must be handled, as indicated by the arc linking the outgoing branches. The solution found is shown in bold lines.

*a solution is not a path but a tree*
Depth-first AND-OR tree search

function And-Or-Graph-Search(problem) returns a conditional plan, or failure
    OR-Search(problem.Initial-State, problem, [])

function Or-Search(state, problem, path) returns a conditional plan, or failure
    if problem.Goal-Test(state) then return the empty plan
    if state is on path then return failure
    for each action in problem.ACTIONS(state) do
        plan ← And-Search(Results(state, action), problem, [state | path])
        if plan ≠ failure then return [action | plan]
    return failure

function And-Search(states, problem, path) returns a conditional plan, or failure
    for each s_i in states do
        plan_i ← Or-Search(s_i, problem, path)
        if plan_i = failure then return failure
    return [if s_1 then plan_1 else if s_2 then plan_2 else ... if s_{n-1} then plan_{n-1} else plan_n]
Search with no observations

search in belief (in agent’s mind)

Figure 4.14
The reachable portion of the belief-state space for the deterministic, sensor-less vacuum world. Each shaded box corresponds to a single belief state. At any given point, the agent is in a particular belief state but does not know which physical state it is in. The initial belief state (complete ignorance) is the top center box. Actions are represented by labeled links. Self-loops are omitted for clarity.

The main advantage of the incremental approach is that it is typically able to detect failure quickly—when a belief state is unsolvable, it is usually the case that a small subset of the belief state, consisting of the first few states examined, is also unsolvable. In some cases,