1 handout: slides

You think you know when you can learn, are more sure when you can write, even more when you can teach, but certain when you can program.
Combinatorial Optimization
Types of Search Problems

- Shortest-path (M&C, vacuum, tile puzzle)
  - want least-cost path to a goal
  - goal depth unknown
  - given operators and their costs

- Constraint satisfaction (map coloring, $n$-queens)
  - any goal is fine
  - maximum depth = number of variables
  - given explicit constraints on variables

- Combinatorial optimization (TSP, max-CSP)
  - want least-cost goal
  - maximum depth = number of variables
  - every leaf is a solution
A tree representation of alternatives in a small combinatorial problem.
Backtracking

- depth-first search
- child ordering
- lower bounds
- branch-and-bound
**Depth-first Search**

```
DFS (node)
1    If is-leaf(node)
2        Visit(node)
3    else
4        For i from 0 to num-children
5        DFS(child(node, i))
```
Depth-first Search Order
Improved Discrepancy Search

ILDS \((node, \text{allowance}, \text{remaining})\)
1. If is-leaf\((node)\)
2. Visit\((node)\)
3. else
4. If \(\text{allowance} > 0\)
5. ILDS\((\text{child}(node, 1), \text{allowance} - 1, \text{remaining} - 1)\)
6. If \(\text{remaining} > \text{allowance}\)
7. ILDS\((\text{child}(node, 0), \text{allowance}, \text{remaining} - 1)\)

start with ILDS\((\text{root}, \text{iteration}, \text{max-depth})\)
The second pass of ILDS visits all leaves with one discrepancy in their path from the root.
Break

- asst 4
- projects
**Hill-Climbing**

<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>■ Types of Problems</td>
</tr>
<tr>
<td>■ Optimization</td>
</tr>
<tr>
<td>■ Backtracking</td>
</tr>
<tr>
<td>■ Depth-first Search</td>
</tr>
<tr>
<td>■ DFS Order</td>
</tr>
<tr>
<td>■ ILDS</td>
</tr>
<tr>
<td>■ ILDS Order</td>
</tr>
<tr>
<td>■ Break</td>
</tr>
<tr>
<td>■ Hill-Climbing</td>
</tr>
</tbody>
</table>


---

Sol $\leftarrow$ some random solution (probably poor quality).

Do *limit* times

\[
\text{New} \leftarrow \text{random neighbor of Sol}.
\]

If *New* better than *Sol*,
then \(\text{Sol} \leftarrow \text{New}\).
Hill-Climbing

\[ \text{Sol} \leftarrow \text{some random solution (probably poor quality)}. \]

Do \textit{limit} times

\[ \text{New} \leftarrow \text{random neighbor of Sol}. \]

If \text{New} better than \text{Sol},

then \text{Sol} \leftarrow \text{New}.

Elaborations: best neighbor (aka gradient-descent)

restarts

simulated annealing

population (GAs, ‘go with the winners’)
Continuous Optimization
Types of Optimization Problems

- **Discrete**
  - finite (often small) set of values per choice
  - planning, CSPs, combinatorial optimization
  - constraints can be implicit or explicit

- **Continuous**
  - real values
  - constraints can be linear, quadratic, ..., non-linear
  - objective can be linear, quadratic, ..., non-linear

- **Mixed discrete / continuous**
  - ‘hybrid systems’
real variables, linear constraints, linear objective

- cheapest diet that meets nutrition guidelines
  - $y_{\text{vitaminA}} = 2.3x_{\text{broccoli}} + 1.7x_{\text{carrots}}$ . . .
  - $\text{cost} = 4.99x_{\text{broccoli}} + 2.67x_{\text{carrots}}$ . . .
  - minimize $\text{cost}$ subject to $y_{\text{vitaminA}} > 500$ . . .

- max flow through network with capacity constraints
- earliest finish time subject to job durations

polynomial time (ellipsoid, Karmarkar’s), but simplex method is popular
CPLEX, Gurobi, Ipsove
**Beyond Linear Programming**

- **convex programming**: constraints and objective are convex
  polynomial time
- **quadratic programming**: constraints and objective are
  quadratic
  some forms are polynomial time
- **0-1 LP**: 0-1 variables, linear constraints, linear objective
  NP-complete
- **integer linear programming**: integer variables, linear
  constraints and objective
  NP-complete
- **combinatorial optimization**: variables are discrete
Please write down the most pressing question you have about the course material covered so far and put it in the box on your way out.

Thanks!