CSC384 – Midterm review
Winter 2018

Tips and Resources
• Let the lecture slides and the posted tutorial materials be your guide for studying. If you’re unclear about something, augment it with the text or other online materials and/or come to an office hour!
• Make sure you understand the material. If there are proofs, work through them so you understand them. Understand the rationale for why things work the way they do.
• Work through some problem sets. Look at the posted sample problems on the test web page as well as problems we went through in class on the board.
• Know and understand the facts: know the complexity of different algorithms and why. Know the axioms of probability and understand how to apply.

General Information
All information is on CSC384 web page Test tab at the top of the page.

Pre-Midterm Help Sessions
• Monday, March 5, 4pm-5pm, SF 3207 (10 King’s College Rd) (TA Chris)
• Tuesday, March 6, 2-3pm, Pratt 378 (6 King’s College Rd) (Sonya)
• Wednesday, March 7, 10-11am, Pratt 378 (6 King’s College Rd) (Sheila)

Wednesday, March 9, 2018
LEC 0101, LEC 2001: 13:00 - 14:00 (i.e., 1:00 - 2:00 PM) Test starts at 13:10. EX 100
LEC 0201, LEC 2201: 15:00 - 16:00 (i.e., 3:00 - 4:00 PM) Test starts at 15:10. EX 200

Examination Centre (EX), 255 McCaul Street, Toronto, M5T 1W7
• 50 minutes in duration
• no aids permitted
• worth 15% of your course grade

Topics
1. Uninformed and Informed Search
2. Game Tree Search
3. Backtracking and CSPs
Uninformed and Informed Search

- Breadth-first search
- Depth-first search
- Depth-limited search
- Uniform Cost Search
- A* search
- IDA* search

---

- Cycle Checking, Path Checking

Search: The Basic Algorithm

```
TreeSearch(Frontier, Successors, Goal?)
    If Frontier is empty return failure
    Curr = select state from Frontier
    If (Goal?(Curr)) return Curr.
    Frontier’ = (Frontier - {Curr}) U Successors(Curr)
    return TreeSearch(Frontier’, Successors, Goal?)
```

REMEMBER: The different forms of search just boil down to the order you put nodes on the frontier/open list.

Uninformed and Informed Search

Metrics:
- Completeness
- Space Complexity
- Time Complexity
- Optimality
From Russell and Norvig

3.4.7 Comparing uninformed search strategies

<table>
<thead>
<tr>
<th>Creation</th>
<th>Reachable</th>
<th>Uniformed</th>
<th>Depth Bound</th>
<th>Time</th>
<th>Heuristic</th>
<th>Optimality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete</td>
<td>$O(b^n)$</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Space</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This is slightly different from what we determined in class/notes (e.g., $O(b^d)$ vs $O(b^{d+1})$). Why? Because they allow for a goal check *before* putting a node on the frontier/open list, whereas we do it when we pop the node off. Think about the implications of moving the goal check with respect to optimality.

McIlraith & Allin, CSC284, University of Toronto, Winter 2018

Heuristic Search Techniques

- Greedy Best First Search, A* Search, IDA*
- Time Complexity?
- Space Complexity?
- Optimal?
- Complete?
- Admissible Heuristics, Monotonic (Consistent) Heuristics
- Optimality & how it interacts with cycle checking.

McIlraith & Allin, CSC284, University of Toronto, Winter 2018

Topics

1. Uninformed and Informed Search
2. Game Tree Search
3. Backtracking and CSPs

Game Tree Search

- Definitions: Two-players, Discrete, Finite, Zero-Sum, Deterministic, Perfect Information
- Components of Two-Player Zero-Sum Game: players, states, terminals, successors, utilities
- Minimax Strategy
- DFS Implementation (Time and Space complexity?)
- Alpha-beta pruning (Empirical savings?)
- Heuristics for games.

McIlraith & Allin, CSC284, University of Toronto, Winter 2018
Game Tree Search

Depth-First Implementation of MiniMax

\[
\text{DFMiniMax}(n, \text{Player}) \quad // \text{return Utility of state } n \text{ given that}
\]
\[
\text{//Player is MIN or MAX}
\]
\[
\text{If } n \text{ is TERMINAL}
\]
\[
\text{Return } V(n) \quad // \text{Return terminal states utility}
\]
\[
\text{//(V is specified as part of game)}
\]
\[
\text{//Apply Player’s moves to get successor states.}
\]
\[
\text{ChildList = } n.\text{Successors(}\text{Player} \text{)}
\]
\[
\text{If Player} = \text{MIN}
\]
\[
\quad \text{return minimum of } \text{DFMiniMax}(c, \text{MAX}) \quad \text{over } c \in \text{ChildList}
\]
\[
\text{Else If Player is MAX}
\]
\[
\quad \text{return maximum of } \text{DFMiniMax}(c, \text{MIN}) \quad \text{over } c \in \text{ChildList}
\]

Topics

1. Uninformed and Informed Search
2. Game Tree Search
3. Backtracking and CSPs

Constraint Satisfaction Problems

- Contrasting CSP and a Search
- Vectors of features as states, each with a domain
- Definitions: Unary, Binary or n-ary Constraints, Scope, Partial State, Domain Wipe Out, Support (for a variable assignment)
- Problem formulations (e.g., binary vs n-ary constraints) and implications of different
- Search augmented with inference (constraint propagation) and trade-offs
Backtracking Search

BT(\text{Level})
\begin{align*}
&\text{If all variables assigned} \\
&\text{PRINT Value of each Variable} \\
&\text{RETURN or EXIT (RETURN for more solutions)} \\
&\text{(EXIT for only one solution)} \\
&V := \text{PickUnassignedVariable}() \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{TRUE} \\
&\text{for } d := \text{each member of Domain}(V) \hspace{1em} (\text{the domain values of } V) \\
&\text{Value}[V] := d \\
&\text{ConstraintsOK} = \text{TRUE} \\
\quad \text{for each constraint } C \text{ such that} \\
&\hspace{1em} a) \text{ } V \text{ is a variable of } C \text{ and} \\
&\hspace{2em} b) \text{ } \text{all other variables of } C \text{ are assigned:} \\
&\hspace{3em} \text{IF } C \text{ is not satisfied by the set of current assignments:} \\
&\hspace{4em} \text{ConstraintsOK} = \text{FALSE} \\
&\hspace{4em} \text{If ConstraintsOK} = \text{TRUE:} \\
&\hspace{5em} \text{BT(\text{Level}+1)} \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{FALSE} //\text{undo as we have tried all of } V\text{'s values} \\
\text{return} \\
\end{align*}

Constraint Propagation (Inference)

- Forward Checking and GAC
- Variable Ordering Heuristics (MRV)
- Arc Consistency (of a CSP, etc.)

Backtracking Search: The Algorithm BT

BT(\text{Level})
\begin{align*}
&\text{If all variables assigned} \\
&\text{PRINT Value of each Variable} \\
&\text{RETURN or EXIT (RETURN for more solutions)} \\
&\text{(EXIT for only one solution)} \\
&V := \text{PickUnassignedVariable}() \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{TRUE} \\
&\text{for } d := \text{each member of Domain}(V) \hspace{1em} (\text{the domain values of } V) \\
&\text{Value}[V] := d \\
&\text{ConstraintsOK} = \text{TRUE} \\
\quad \text{for each constraint } C \text{ such that} \\
&\hspace{1em} a) \text{ } V \text{ is a variable of } C \text{ and} \\
&\hspace{2em} b) \text{ } \text{all other variables of } C \text{ are assigned:} \\
&\hspace{3em} \text{IF } C \text{ is not satisfied by the set of current assignments:} \\
&\hspace{4em} \text{ConstraintsOK} = \text{FALSE} \\
&\hspace{4em} \text{If ConstraintsOK} = \text{TRUE:} \\
&\hspace{5em} \text{BT(\text{Level}+1)} \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{FALSE} //\text{undo as we have tried all of } V\text{'s values} \\
\text{return} \\
\end{align*}

Forward Checking Algorithm

FC(\text{Level}) /*Forward Checking Algorithm */
\begin{align*}
&\text{If all variables are assigned} \\
&\text{PRINT Value of each Variable} \\
&\text{RETURN or EXIT (RETURN for more solutions)} \\
&\text{(EXIT for only one solution)} \\
&V := \text{PickAnUnassignedVariable}() \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{TRUE} \\
&\text{for } d := \text{each member of } \text{CurDom}(V) \\
&\text{Value}[V] := d \\
&\text{DWOoccurred} := \text{FALSE} \\
&\text{for each constraint } C \text{ over } V \text{ such that} \\
&\hspace{1em} a) \text{ } C \text{ has only one unassigned variable } X \text{ in its scope} \\
&\hspace{2em} \text{IF } \text{FCCheck}(C,X) == \text{DWO} \hspace{1em} /* X domain becomes empty*/ \\
&\hspace{3em} \text{DWOoccurred} := \text{True} \\
&\hspace{3em} \text{break } /* \text{stop checking constraints} */ \\
&\hspace{3em} \text{if(not DWOoccurred) /*all constraints were ok*/} \\
&\hspace{3em} \text{FC(\text{Level}+1)} \\
&\text{RestoreAllValuesPrunedByFCCheck}() \\
\end{align*}
\begin{align*}
\text{Assigned}[V] := \text{FALSE} //\text{undo since we have tried all of } V\text{'s values} \\
\text{return} \\
\end{align*}
**Forward Checking Algorithm**

For a single constraint C:

```plaintext
FCCheck(C, x)
// C is a constraint with all its variables already assigned, except for variable x.
for d := each member of CurDom[x]
    IF making x = d together with previous assignments to variables in scope C
        falsifies C
    THEN remove d from CurDom[x]
    IF CurDom[x] = {} then return DWO (Domain Wipe Out)
ELSE return ok
```

**Enforce GAC (prune all GAC inconsistent values)**

```plaintext
GAC_Enforce()
// GAC-Queue contains all constraints one of whose variables has had its domain reduced. At the root of the search tree, // first we run GAC_Enforce with all constraints on GAC-Queue
while GACQueue not empty
    C = GACQueue.extract()
    for V := each member of scope(C)
        for d := CurDom[V]
            Find an assignment A for all other variables in scope(C) such that C(A ∪ V=d) = True
            if A not found
                CurDom[V] = CurDom[V] – d
                empty GACQueue
                return DWO //return immediately
            else
                push all constraints C' such that V ∈ scope(C') and C' ∈ GACQueue
                on to GACQueue
        return TRUE //while loop exited without DWO
```

**Example: N-Queens GAC search Space**

Recall we kept exploring with FC only consistency stages:
1. V2 = (1, 4), V3 = (2, 4), V4 = (2, 3)
2. V2 = (1, 2) & V3 = 1, 3 & V3 = 1, 4 are inconsistent with V1 = 1;
3. V3 = (2) (V3 = 2) is inconsistent with values in CurDom[V2]
4. V4 = (1) (both values for V4 inconsistent with values in CurDom[V3]
DWO
CONTRAST TO: N-Queens FC search Space

CONTRAST : N-Queens Backtracking Search Space