CS188 Fall 2011 Section 3: CSPs Part 2 and Game Trees

1 Trains

A train scheduler must decide when trains $A$, $B$ and $C$ should depart. Once a train departs, it moves one space along its track each hour (in discrete jumps) until it arrives at its destination platform. Each train can depart at 1, 2 or 3 pm. The scheduler has two restrictions: All trains must leave at different times, and two trains should not both occupy crossing sections of track after any one hour time step is over. Note that train $A$ is two spaces long. Also note that the collision constraint is enforced only at the conclusion of every hour - time is discrete in this problem.

(a) Describe the constraint satisfaction problem that, when solved, will tell the train scheduler when each train should depart. Let the variables $A$, $B$ and $C$ represent the departure times of the three trains.

1. variables and domains: $A, B, C \in \{1, 2, 3\}$
2. different times constraints: $A \neq B$, $B \neq C$, $A \neq C$
3. intersection constraints: $A + 1 \neq B$, $A + 1 \neq C$, $A + 2 \neq C$, $B \neq C + 1$

These constraints are alternative ways of expressing the intersection constraints: $(B > A + 1 \text{ or } A > B)$, $(C > A + 2 \text{ or } A + 1 > C)$, $(B \neq C + 1)$

(b) Draw the constraint graph for the CSP you defined.
A fully connected graph with $A$, $B$ and $C$. A hypergraph connecting all three nodes with one arc is okay.

(c) After selecting $A = 2$, cross out all values for $B$ and $C$ eliminated by forward checking.

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Forward checking eliminates all values from $B$ and $C$ that are not compatible with $A = 2$. Forward checking does not eliminate $B = 1$, even though it conflicts with $C = 1$. That’s the job of arc consistency.
(d) Cross out all values eliminated by arc consistency before assigning any variables.

\[
\begin{array}{ccc}
A & B & C \\
1 & 2 & 3 \\
\end{array}
\]

- C = 3 is eliminated because it is not compatible with A ∈ \{1, 2, 3\}.
- A = 1 is eliminated because it is not compatible with C ∈ \{1, 2\}.
- B = 3 is eliminated because it is not compatible with A ∈ \{2, 3\}.
- B = 2 is eliminated because it is not compatible with C ∈ \{1, 2\}.
- C = 1 is eliminated because it is not compatible with B ∈ \{1\}.
- A = 2 is eliminated because it is not compatible with C ∈ \{2\}.

(e) After selecting A = 2, cross out all values for B and C eliminated by arc consistency.

\[
\begin{array}{ccc}
A & B & C \\
2 & 1 & 2 \\
\end{array}
\]

Any answer with the domain of either B or C fully crossed out is acceptable.

(f) Describe the execution of backtracking search using forward checking and the minimum remaining values (MRV) and least constraining values (LCV) heuristics. Specifically, in what order are the variables assigned and what values do they take? Start by assigning variable A. You may not need to fill all the lines below:

1. variable A is assigned value 3.  
   \text{3 is the least constraining value.}

2. variable B is assigned value 1.  
   \text{B and C both have two values remaining and degree 1 (they constrain exactly one other unassigned variable). B = 1 and C = 2 are least constraining.}

3. variable C is assigned value 2.  
   \text{Forward checking leaves only one option.}

\text{Note: Lines (2) and (3) may be switched.}
2 Minimax Search

In this problem, we will explore adversarial search.

Consider the zero-sum game tree shown below. Trapezoids that point up, such as at the root, represent choices for the player seeking to maximize; trapezoids that point down represent choices for the minimizer. Outcome values for the maximizing player are listed for each leaf node. It is your move, and you seek to maximize the expected value of the game.

(a) Assuming both opponents act optimally, carry out the minimax search algorithm. Write the value of each node inside the corresponding trapzoid. What move should you make now? How much is the game worth to you?

The game is worth 5. We should make the move that takes us left down to the node containing 5.

(b) Now reconsider the same game tree, but use α-β pruning (the tree is printed on the next page). Expand successors from left to right. In the brackets [ , ], record the [α, β] pair that is passed down that edge (through a call to MIN-VALUE or MAX-VALUE). In the parentheses ( ), record the value (v) that is passed up the edge (the value returned by MIN-VALUE or MAX-VALUE). Circle all leaf nodes that are visited. Put an ‘X’ through edges that are pruned off. How much is the game worth according to α-β pruning?

α-β pruning finds the same solution. The game is still worth 5 to the maximizer.