Assignment 2

Artificial Intelligence

October 28, 2019

Written Component Due Date: October 23, 2019 at 11:59pm
Programming Component Due Date: November 1, 2019 at 11:59pm

Written Component: Please submit a PDF containing your solutions called LastName-FirstName-A2.pdf, where you replace LastName and FirstName with your last and first names. You may write out your answers on paper, take pictures of your work, compile them into a PDF and submit in that manner if you prefer not to type out the answers.

1. **Question 1**
The following is known as the KNAPSACK problem. You have a knapsack and a collection of valuable objects. Each object O has a weight O.W and a value O.V. There is a maximum M on the total weight that you can carry. You have a target value T. The question is can you choose objects to "steal" so that their total value is at least T and their total weight is at most M?

For instance, consider that T = 20 and M = 10 and you have the following objects (same example we went over in class):

Object	**A**	**B**	**C**	**D**	**E**
Value	10	8	7	6	4
Weight	8	4	3	3	1

The only solution here is { B, C, D } with a total value of 21 and a total weight of 10. Note that the solution includes neither the most valuable item, A, nor the item with the greatest value per weight, E.

We want to use a hill climbing algorithm with the following state space and evaluation function.

- A state is any set of objects
- A valid action is either to add an object to the set or to delete an item from the set
- For any set S, let Weight(S) and Value(S) refer the total weight and total value of S respectively.
- Evaluation Function: Error(S) = max(Weight(S) - M, 0) + max(T - Value(S), 0)

An global optimum set, S' would then be one where the Error(S') = 0.

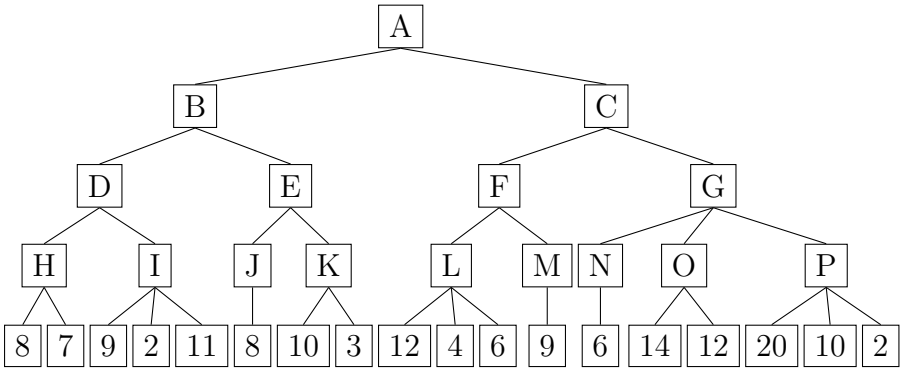
For example, if S = { A, B }, then Error(S) = max(12 - 10, 0) + max(20 - 18, 0) = 2 + 2 = 4.
If S = {A} then Error(S) = max(8 - 10, 0) + max(20 - 10, 0) = 0 + 10 = 10.

(a) Suppose at some iteration of hill climbing the current state is { A, E }. What is the best neighbor of the state { A, E }? What happens on the next iteration after moving to the best neighbor?

(b) Consider the general case of the KNAPSACK problem where there are N objects. What is the size of the state space? What is the maximal number of neighbors of any state.

(c) Assume that the values, the weights, M, and T are all reasonable-sized integers. You can obviously assume that all the weights are less than M (otherwise the object is useless), and that all values are less than T (if an object has a weight less than M and a value greater than T, then your problem is solved). Give an upper bound on the maximum number of iterations of the hill climbing algorithm that are possible as a function of N, M, and T. Incorporating the answer to (B), give an upper bound on the overall running time of the algorithm.

2. **Problem 2**: Minimax Alpha-Beta Pruning
   Consider the tree-like graph. Suppose that the top level is a MAX node (the A node is MAX node). Use Minimax algorithm with alpha-beta pruning to determine what action A should take and the associated utility. That is which direction should the player move to from A (B or C) and why? Also indicate which nodes of the trees will be pruned using alpha-beta pruning.



3. **Problem 3**: Convert the following sentences in propositional logic to conjunctive normal form.

   - $A \iff (B \vee C)$
   - $(A \wedge C) \Rightarrow G$
   - $B \Rightarrow (C \wedge D)$
   - $C \Rightarrow E$
   - $E \Rightarrow \neg G$
   - $(E \wedge G) \Rightarrow A$

4. **Problem 4**: Davis-Putnam Trace the action of the Davis-Putnam algorithm applied to the clauses in problem 3 (you have already converted them to CNF). Assume that when the algorithm encounters a branch point, it selects the first atom in alphabetical order, and that it tries "TRUE" before "FALSE".

Programming Component

Programming Component: Please submit either a Python or Java file that contains your program that fulfills the requirements detailed below. You should also include in your submission a README file with quick overview of how to run your program and anything the grader should be aware of. Other programming languages may be allowed with permission of the instructor.

- **Assignment**
  Write a program that implements the Davis-Putnam algorithm.

  The input to the Davis-Putnam procedure has the following form: An atom is denoted by a natural number: 1, 2, 3, ... For example, you can think 1 as referring to the literal P and -1 as referring to the literal $\neq$P. A clause is a line of text containing the integers of the corresponding literals. After all the clauses have been given, the next line is the single value 0; anything further in the file is ignored in the execution of the procedure and reproduced at the end of the output file. (This allows for passing extra information in case the Davis-Putnam procedure is a part of a larger pipeline of procedures to accomplish a task).

  The output from the Davis-Putnam procedure has the following form: First, a list of pairs of atoms (a natural number) and a truth value (either T or F). Second a line containing the single value 0. Third, the remaining lines after the 0 line in the input file reproduced exactly.

  Example Input: Given the input:
  1 2 3
  -2 3
  -3
  0
  This is a simple example with 3 clauses and 3 atoms.
  Reproduce this line.

  Example Output:
  1 T
  2 F
  3 F
  0
  This is a simple example with 3 clauses and 3 atoms.
  Reproduce this line.

  This corresponds to the clauses
  - $P \vee Q \vee R$
  - $\neg Q \vee R$
  - $\neg R$

If the clauses have no solution, then the Davis-Putnam procedure outputs a single containing a 0, followed by remaining lines in the input file.

You may assume that your program will not be tested with more than 1000 atoms and more than 10,000 clauses. Recall, we went over in class how to generate sample input for your program and the success rate of Davis-Putnam given number of clauses relative to a fixed number of atoms.

- **Grading:**
  9 points for correctly running code and 1 point for well-written code for a total of 10 points on the programming component. The written component and programming component are weighed equally for this assignment.

See Canvas for actual sample input file(s).