15-281 Fall 2023 Lecture Activity

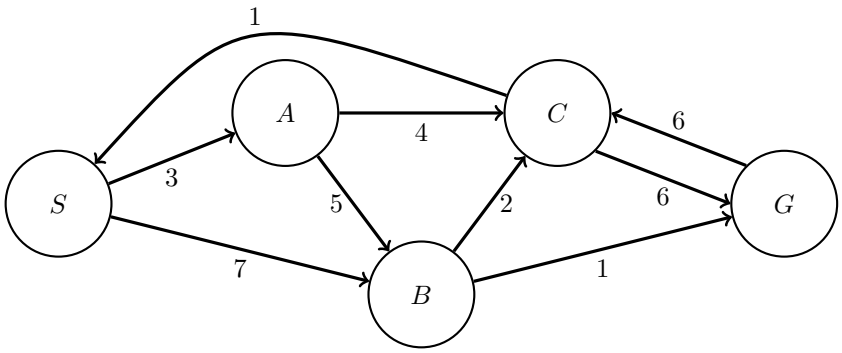
Learning Objectives

- Represent various problems (e.g., search, min/max, rectified linear) as LP/IP
- Explore the connections between different topics

IMPORTANT: Please return the completed activity to the front at the end of class. Write your name and Andrew ID on the top of the first page. You will get 1 point per full page completed (4 total).

Q1. Cost-Based Search as IP

In this problem we are going to explore how to formulate search problems as Integer Programming problems. For the sake of simplicity we will be considering the weighted directed graph below (S is the start, and G is the goal):



We can derive a representation for any path[1] in a graph by considering variables to represent edges in the graph each with domain {0,1}; value 0 if the edge is not in the path and 1 if the edge is in the path. Specifically, define a binary variable $x_{X\to Y}$ if there is an edge $X \to Y$ in the graph.

For example, the path $S \to A \to C \to G$ can also be seen as the set of edges $\{S \to A, A \to C, C \to G\}$. We can then define $x_{S\to A}$ to be the indicator variable for whether the edge $S \to A$ is on the path, $x_{S\to B}$ to be the indicator variable for whether the edge $S \to B$ is on the path, and so on. Using this binary representation, $S \to A \to C \to G$ can be represented as:

$$\big(x_{S\to A}=1 \quad x_{S\to B}=0 \quad x_{A\to B}=0 \quad x_{A\to C}=1 \quad x_{B\to C}=0 \quad x_{B\to G}=0 \quad x_{C\to S}=0 \quad x_{C\to G}=1 \quad x_{G\to C}=0\big)$$

If we fix the order of the indicator variables in the order above, the path can be represented as a 9−tuple: (1, 0, 0, 1, 0, 0, 0, 1, 0)

(a) Answer the following questions below about the representation.

(i) Write the 9−tuple binary representation for the path $S \to A \to B \to C \to G$

Answer:

(ii) Write the 9−tuple binary representation for the path $A \to C \to S \to B$

Answer:

[1] Note that a "path" is different from a "walk"; for our purpose each node can only appear in the path at most once.

(iii) Write the path that corresponds to (0, 0, 1, 0, 1, 0, 0, 0, 0)

Answer:

(b) Note with our binary representation, it is possible to make "paths" that do not start from S and end at G, and some of the 9−tuples are not even valid paths at all. Consider the constraints on valid paths in this representation.

From the binary representation itself, it is clear that we need to constrain the indicator variables to be an integer between 0 and 1 inclusive. To find valid paths from the start to the goal, we need to start by imposing some additional constraints on the 9−tuples.

(i) Write the constraint(s) (in **inequality form**) that a valid path in the specific graph above must start at S. (Hint: What must be true about the values or sums of values of the indicator variables involving S defined above? Remember that constraints must be linear sums of variables.)

Answer:

(ii) Similarly, write the constraint(s) that a valid path must end at G (in inequality form).

Answer:

(iii) Write a 9−tuple (binary representation) that satisfies the constraints in **(i)** and **(ii)** but does **not** represent a valid path from S to G

Answer:

(c) Let us define a non-terminal node as a node that is neither the defined start node (S) nor the goal node (G). Note that the constraints in part **(b)** do **not** guarantee that all other nodes in the path are non-terminal.

(i) We want to ensure that the path only passes through each non-terminal node (e.g., B) at most once. Write the constraint(s) that node B does not appear more than once on the path (inequality form).

Answer:

(ii) Finally, to ensure B is non-terminal, we have to make sure that if there is an edge to B in a path, then there should also be an edge from B to some other node. If there is an edge to B but not from B, then the path has a dead-end at B. Similarly, if there is an edge from B to another node but no edge to B, then B is a starting node but not S. Write down the corresponding constraint(s) about B's non-terminal position (inequality form).

Answer:

(d) We have all the constraints to ensure a valid path (actually we don't, but for now assume we do). Write the objective function for the IP search problem representing the graph above. (Hint: the goal of solving this IP problem is to minimize the objective function that corresponds to the search objective.)

Answer:

(e) Let us take a closer look at the representation. As hinted before, the current constraints **DO NOT** ensure a valid path for graphs in general.

(i) Come up with a directed graph such that there exists a tuple that passes all the previously mentioned constraints but does not represent a valid path on the graph. You should be able to construct such an example with less than 10 edges. The weights of the edges are not important for this question, you may omit them.

Draw your graph, and give the counterexample tuple. As a reminder, the length of your tuple should be the same as the number of edges in your graph. Clearly label the correspondence between the elements in your tuple and the edges on the graph

Answer:

(ii) Despite the existence of these invalid tuples, they will never be returned by the IP above (as defined by the previous constraints and objective function) as long as every edge has a positive cost. Briefly explain why this is the case.

Answer:

(f) Define a new search algorithm to be: formulate the search as an IP problem, then run an IP solver and return the solution as a path. Is this new search algorithm:

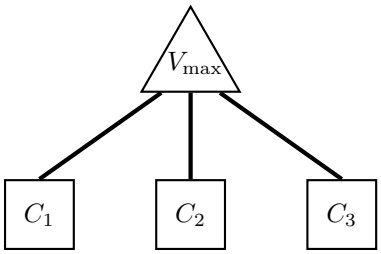
(i) Complete? ◯ Yes ◯ No

(ii) Optimal? ◯ Yes ◯ No

Q2. Adversarial Search as IP (Bonus Question)

In this problem[2] we are going to explore how to do a similar representation for the adversarial search, specifically the **Minimax**. As the Minimax tree consists of different layers of maximizing and minimizing nodes, an intuitive way to get started is to represent each individual node and put them together.

(a) As shown in the figure below, let us start with a maximizing node with three children. For now, assume that all the children are just constant values (not other nodes).



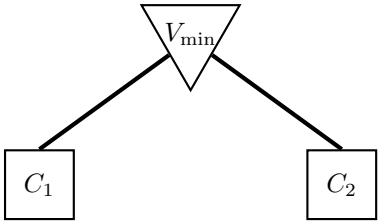
If we define a variable $V_{\max}$ to represent the true value of the maxie node (without any pruning), then we can define the following constraints to represent that the value of the node is greater or equal to all of its children.

$$C_1 - V_{\max} \leq 0 \qquad C_2 - V_{\max} \leq 0 \qquad C_3 - V_{\max} \leq 0$$

These constraints alone do not guarantee that $V_{\max}$ will end up being the maximum value of its children. However, we can enforce that with the help of the objective function. Define an objective function such that with this objective function, the value of $V_{\max}$ will be equal to the maximum value of its children when we solve the IP. (Hint: in the standard form of LP/IP we are *minimizing* the objective function).

Answer:

(b) Now similarly write the constraints and objective function for a minimizing node, where we define a variable $V_{\min}$ to represent the value of the minnie node.



(i) Write the constraints in inequality form (Hint: there should be two constraints).

Answer:

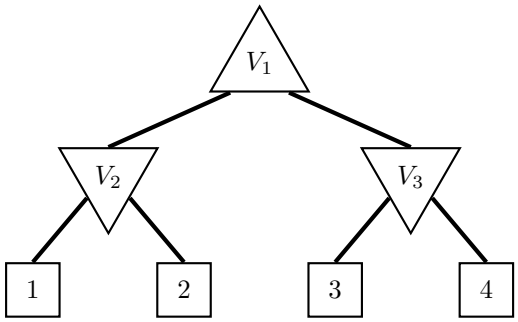
(ii) Write the objective function to be minimized.

Answer:

[2]Although technically still within scope, this is a **VERY HARD** question (part (a-c) should be fine, the rest needs some good intuition). You are not expected to handle questions at this level of difficulty on your own in any part of this course. The background of this question is the thought process of a TA trying to design a problem on Minimax as IP but later realizing it is too hard to be anywhere in the course. It should give you some inspiration on how to approach a problem. For those that are interested, you are encouraged to think about other smarter ways to convert minimax into an IP.

(c) Now that we have two separate IPs for maxie and minnie nodes respectively, we can now put them together to solve the more complex Minimax problem.

Specifically, we start with the following Minimax tree:



(i) First, write the values of the three internal nodes if we run minimax without pruning.

Answer:

(ii) Putting all constraints in part **(a)** and **(b)** together, we have the following constraints (you should complete the omitted constraints when working on the rest of this question).

$$-V_1 + V_2 \leq 0 \qquad -V_1 + V_3 \leq 0 \qquad \text{and the constraints in your part } \textbf{(b)(i)}$$

Since we have two types of objective functions (part **(a)** and **(b)(ii)**), and now we want to optimize both of them, the simplest way to achieve this is to add them together.

Write the objective function for the Minimax tree above using variables V_1, V_2, V_3 (Hint: we are minimizing this objective, and your expression should include all three variables).

Answer:

(iii) The value assignment in part **(c)(i)** passes all the constraints. Write the value of the objective function (defined in part **(c)(ii)**) given this value assignment.

Answer:

(iv) Unfortunately, this naive method of converting minimax to IP does not work. To illustrate this, find another set of value assignments to V_1, V_2, and V_3 such that:

- V_1 is assigned to a different value
- The value assignment passes all the constraints in part **(c)(ii)**
- The objective function returns the same value for this value assignment as in part **(c)(iii)**

Write the new assignment in the form of "V_1 = some constant V_2 = some constant V_3 = some constant"

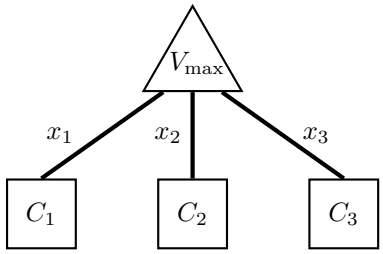
Answer:

Because this new assignment has the same objective value as the correct one in part **(c)(i)**, it is possible for the IP to return it as the final solution, showing that the current definition of the IP cannot be used for solving minimax.

You can try to play around with editing constraints and/or the objective function. However, this variable representation (defining one variable for each node to represent their values) is not expressive enough to solve the minimax problem.

(d) To fix this problem, we should revisit the representation for the maximizing node. Previously we used the constraints to specify that its value is no less than its children, and used the objective function to represent that its value is the actual maximum value among its children. Now we should try doing both things with **constraints only**.

Revisiting the cost-based search as an IP problem, one way to accomplish this is to define binary variables to represent each action, such that we can enforce that the value of the parent is the same as one of its children. So as shown in the figure below, in addition to defining $V_{\max}$ to represent the value of the maximizing node, we also define three binary variables x_1, x_2, x_3 to represent whether the corresponding action is chosen (in this case assume C_1, C_2, and C_3 are constants).



We keep the following constraints the same to enforce V_{max} must be greater than or equal to its children:

$$C_1 - V_{\max} \leq 0 \qquad C_2 - V_{\max} \leq 0 \qquad C_3 - V_{\max} \leq 0$$

But we also need extra constraints to enforce that V_{max} must take the value of one of its children.

(i) Write the constraint(s) to enforce **exactly one** action is taken at the maximizing node

Answer:

(ii) Write the constraint(s) to enforce the value of $V_{\max}$ is the value of the chosen child

Answer:

(iii) Explain in 1-2 sentences why this formulation does not work if the children are values of other nodes (variables) instead of constants (so instead of constants C_1, C_2, C_3 we used variables V_1, V_2, V_3).

Answer:

(e) Now it's time to think about a sub-problem, which is how to represent $x \cdot V$ (where both x and V are variables) using *linear* constraints.

This would not be possible in general, but lucky for us, there are additional constraints (listed below) that we can utilize in the minimax problem so that we *can* present the multiplication using linear constraints.

- x is binary
- V is bounded (i.e., there is a maximum possible value and a minimum possible value for it)

(i) Note that V represents the value of a node in the minimax tree. Briefly explain why it is bounded.

Answer:

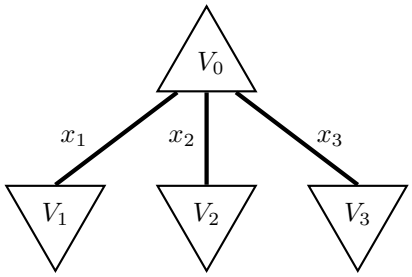
(ii) Hence, we can just assume the value of *all* nodes are bounded in $[0, 1]$ for the rest of this problem. Now let us introduce a new variable V' and enforce $V' = x \cdot V$ using linear constraints. First, we observe that the following are true

$$V' \leq x \qquad \text{and} \qquad V' \leq V$$

Write the objective function on V' that enforces $V' = x \cdot V$ given the constraints above. (Hint: this should be very similar to your answer in part **(a)**

Answer:

(f) Returning from the detour, now we have everything we need to represent a maximizing node with other nodes as children. In the following graph, V_0, V_1, V_2, V_3 are the variables representing the values of each node, and x_1, x_2, x_3 are the binary variables representing whether each action is the chosen option for the node.



(i) Write **all** the constraints relevant to the V_0 node (you should reuse your constraints from parts **(d)** and **(e)**). You may introduce extra variables (e.g., V' in part **(e)(ii)**).

Answer:

(ii) Write the objective function for this maximizing node.

Answer:

(iii) Identify which constraint(s) you would change if all the nodes are flipped (that is, the root is now a minimizing node while the children are maximizing), and explain what you would change them to. Also, explain whether you need to change the objective function, and if so, what it should be changed to.

Answer:

(g) Finally, it is time to put everything together again. As we did in part **(c)**, we can use the following steps to combine the individual IPs of each node:

- Stack all the constraints
- Sum all the objective functions to be the new objective function

However, in part **(c)**, we have seen that adding up all the objective functions did not work. Explain, on a high level, why the updated representation, especially the objective function, works when combining different nodes together but not the original naive representation.

Answer:

(h) Briefly explain why the integer constraints on the action variables are necessary. Describe what might happen if we lift those integer constraints.

Answer: