Problem Solving in Computer Science - Week 1 Notes

Scribe on duty: Radu Negoescu

September 17, 2008

1 General Information for Course Attendants

- Course assessment will be done through project work, in groups of 3 to 4 people;
- Main course information hub is the course home page at: http://mtc.epfl.ch/courses/ProblemSolving−2008/;
- Teaching Assistant (TA): Ms. Verena Wolf (http://mtc.epfl.ch/∼vwolf/);
- It is preferable that the decision to continue / discontinue attending the course be taken as soon as possible by the participants;
- Every week a volunteer will take notes for the whole class, turn them into a LaTeX document, and make it available to everybody through the TA.

2 First Project - Probabilistic Graph Games

2.1 Description of the Problem (Informal)

The problem: the input to the problem is a directed graph, with 3 types of nodes. Type 1 nodes called MAX nodes, type 2 nodes called MIN nodes, and type 3 nodes called RAND nodes (from random). For simplicity, each node has out-degree 2, except for two nodes in the graph, one MAX node and one MIN node. These nodes without any descendants are called *sink nodes*. Two players, MAXP and MINP, and a token are also part of the problem.

Figure **??** shows an example of such a graph. Square nodes are MAX nodes, diamond nodes are MIN nodes, round nodes are RAND nodes. Doubled nodes are the sink-MAX and sink-MIN nodes. The graph game supposes the token is present at any given time at a given node in the graph, and can be moved from one node to another according to a set of rules and objectives. MAXP's objective is to take the token to the sink-MAX node, MINP's objective is to take the token to the sink-MIN node, point at which the game ends. The token moving rules are:

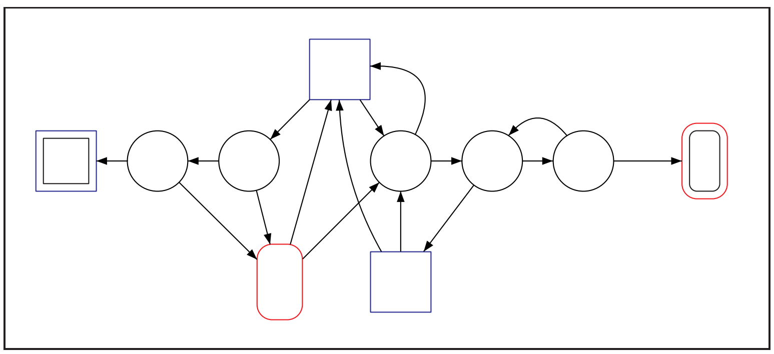

Figure 1: Example of a probabilistic graph. Square nodes are MAX nodes, roundish rectangular nodes are MIN nodes, and round nodes are RAND nodes. Sink-MAX and sink-MIN have doubled edges.

1. the token can be moved to a node if and only if a directed edge links the current node to the target node;
2. if the token is at a MAX node, it will be moved by the MAXP player;
3. if the token is at a MIN node, it will be moved by the MINP player;
4. if the token is at a RAND node, it will be moved with a random probability defined a priori for the RAND node (for our example, uniform probability $\frac{1}{2}$).

The question: what is each player's probability to win, supposing they play optimally?

Additional simplifications:

- the full graph is known in advance by the two players;
- the probability to eventually reach one of the sink nodes is one.

2.2 Known Algorithms

Observation: in order to compute the winning probability for either player, we will need to compute partial at-node min/max winning probabilities. Without loss of generality, we will consider from here on only the winning probability of MAXP.

2.2.1 Value Improvement Algorithm

Let us call the probability of MAXP to win the game if the token is at the current node i the *value of the node*, denoted by x_i. Under this notation, the value of sink-MAX for player MAXP is 1, and the value of sink-MIN for player MAXP is 0. In general, for MAXP we have the following rules for a node i's value with two descendants j and k, regardless of nodes' j and k types:

- if i is a MAX node: $x_i = \max(x_j, x_k)$;
- if i is a MIN node: $x_i = \min(x_j, x_k)$;
- if i is a RAND node: $\frac{x_j + x_k}{2}$.

The algorithm has the following steps:
Step 1 Naively assign node values as follows:

- all MAX nodes value: 1;
- all MIN nodes value: 0;
- all RAND nodes value: $\frac{1}{2}$.

Figure **??** shows this first step with the naive node values.

Step 2: pick a node for which the rules do not hold, update its value, and propagate this update through the entire graph. Repeat this step until all nodes' values satisfy the rules.
Observation: this algorithm is guaranteed to converge to a solution, however it may need exponential iterations (in the size of the graph).

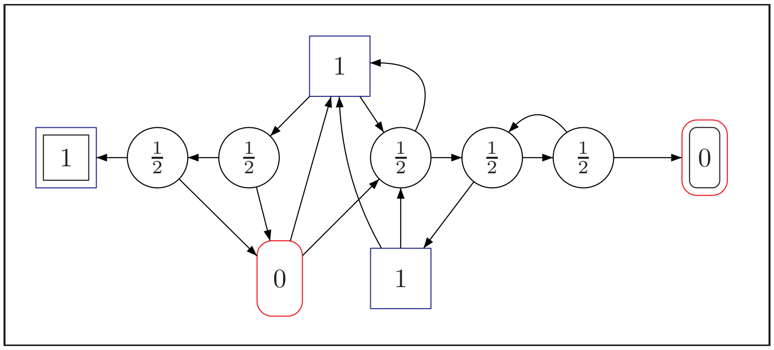

Figure 2: Naively assigning node values for MAXP

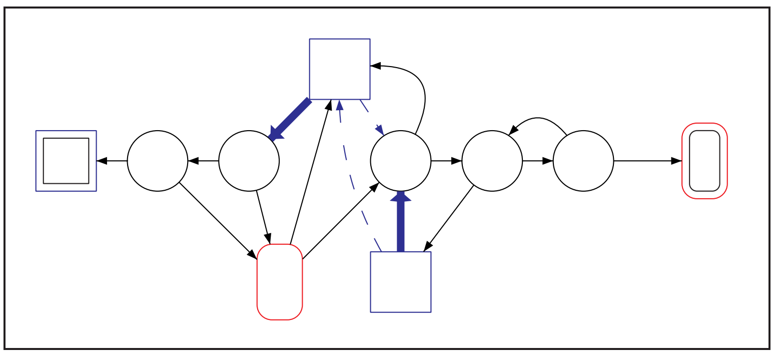

Figure 3: Fixing strategy σ for MAXP and removing unused edges

2.2.2 Strategy Improvement Algorithm

Observation: it does not make sense to choose different edges for a different path history. Hence, we can talk about strategies for each of the players, denoted as σ for MAXP and τ for MINP, that will select a direction (Left or Right) in each node:

- σ: $V_{max} \rightarrow$ {Left, Right};
- τ: $V_{min} \rightarrow$ {Left, Right}.

Once a strategy is fixed, some edges become unused, and can thus be removed from the graph. Figure **??** exemplifies this: in green, the strategy, and in superimposed green dashes the removed edges.

Fixing a strategy for one player (say MAXP) renders the problem into a Markov Decision Process (MDP), with only the MINP being able to make choices. This MDP is solvable by linear programming, with the following constraints:

- for MIN nodes, $x_i \geq min(x_j, x_k)$;

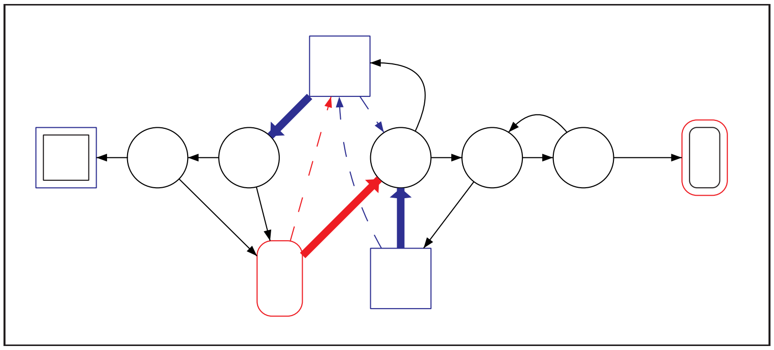

Figure 4: Fixing strategy σ for MAXP and removing unused edges, then determining optimal strategy τ for MINP and removing unused edges

- for RAND nodes, $x_i = \frac{x_j + x_k}{2}$;
- $x_i \geq 0$.

The objective function in this case is $O = min \sum_i x_i$

The algorithm, assuming we are solving for MAXP, has the following steps:
Step 1: arbitrarily guess a strategy σ for MAXP.
Step 2: determine optimal strategy τ for MINP. This is exemplified in Figure **??**.

Once both strategies are fixed, the problem becomes a Markov Chain (MC) with no choices except the RAND nodes. In this MC we can compute node values through linear equations on each node:

- for RAND nodes, $x_i = \frac{x_j + x_k}{2}$;
- for sink-MAX, $x_i = 1$.
- for sink-MIN, $x_i = 0$;
- MIN and MAX nodes (other than sinks) inherit the value of their (now unique) successor.

Step 3: once node values are available, improve if possible the strategy σ taking into account these values, and then continue from **Step 2** until no more changes are available. This step is exemplified in Figure **??**.
Observation This algorithm is not guaranteed to find a solution in polynomial number of iterations.
Observation The decision problem is in $NP \cap coNP$, strategies are polynomial size witnesses.

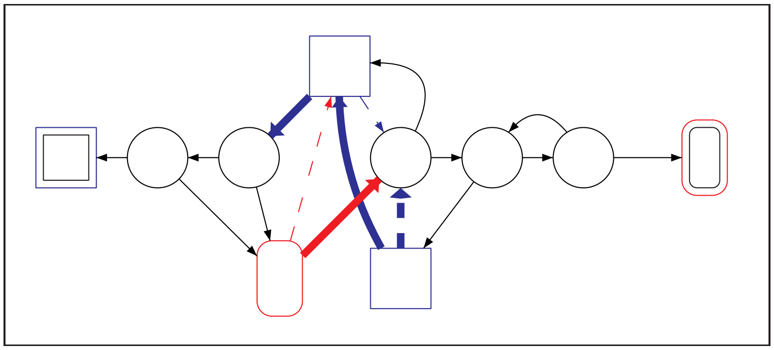

Figure 5: Re-evaluating strategy σ based on node values

2.3 TO DO

- Form groups of 3 to 4 people, and come up with a formal definition of the problem;
- Test on a sample graph the two algorithms described above;
- Make decision on taking or dropping the course.