Solutions Exam MPRI

2019

Let G be a (two-player deterministic) mean-payoff game. We let W denote the largest weight in absolute value (we assume that all the weights are integers). The mean-payoff objective is that the infimum limit of the average of the weights seen along a path is non-negative ($\geqslant 0$).

Question 1: Prove that if Eve has a winning strategy, then she has a strategy that ensures that at all times, the total sum remains always larger than or equal to $-nW$. Is the value nW optimal?

Solution: We consider a positional strategy σ for Eve. If we restrict G to the moves prescribed by σ we have a graph such that all paths satisfy mean-payoff, which equivalently says that there are no cycles with negative total cost. This gives an upper bound of nW. A long path with only $-W$ followed by a self-loop with 0 yields an almost matching lower bound.

Question 2: We construct a deterministic automaton A. The alphabet is the set of weights of G. The set of states is $[-nW, nW]$ plus an extra sink rejecting state $\bot$ and a sink accepting state $\top$. The automaton starts from the state 0 and stores the total sum of the weights, restricted to $[-nW, nW]$. The automaton rejects (goes to $\bot$) if the total sum goes below $-nW$. If the total sum goes above nW, it stays in $\top$. *Important*! **All** states except for $\bot$ are accepting.

Construct a safety game over the synchronised product of G and A such that Eve has a winning strategy in the mean-payoff game G if and only if Eve has a winning strategy in the safety game $G \times A$.

Solution: The difficulty is not to forget to prove both implications!

The left to right direction is almost proved in Question 1, this answer was accepted. To be slightly picky, we need to explain that the choice of accepting if the total sum goes above nW is OK, since a priori it could be the case that the total sum goes very high, and then very low. This does not happen, and can be shown by contradiction reasoning on the graph restricted to the moves prescribed by a positional winning strategy for Eve in the mean-payoff game G.

For the right to left direction, we observe that if the total sum is bounded from below, then the mean-payoff objective is satisfied. Once again the careful student may have trouble with what would happen if the (actual) total sum goes very high, and then very low. To prove this formally, we reason by contradiction on the graph restricted to the moves prescribed by a positional winning strategy for Eve in the safety game $G \times A$.

Question 3: Construct an algorithm for solving mean-payoff games based on the construction above. Analyse its (time) complexity.

Solution: Solving safety games can be done in $O(m)$ where m is the number of edges. The algorithm for solving mean-payoff games is: construct $G \times A$, and solve it as a safety game. We note that the safety game $G \times A$ has $m \cdot nW$ edges: each edge in G induces nW edges in $G \times A$ (because A is deterministic!). Hence the complexity is $O(mnW)$.

Worse complexity can be accepted, for instance a more naive upper bound on the number of edges is to multiply the number of edges of G by the number of edges of A.

Question 4: Construct an algorithm for solving CoBüchi games based on the construction above. Analyse its (time) complexity.

Solution: We define CoBüchi edges to have weight -1 and the others have weight 0. To see that the two games are equivalent, use positional determinacy for Eve both ways. This yields the (optimal) complexity for solving CoBüchi games of $O(nm)$.

Question 5: Let k be the number of different weights in the mean-payoff game G. In the similar way as above, construct an algorithm for solving mean-payoff games whose complexity is $O(mn^k)$.

Solution: Since there are at most n^k different sums of at most n weights, the automaton A constructed above has size n^k, which yields the complexity $O(mn^k)$.

Question 6: In the constructions above, can we avoid constructing explicitly the automaton A to have a better space complexity for solving mean-payoff games?

Solution: The algorithm can be run on the fly. We store at each point a function from vertices to $[-nW, nW]$. The space complexity is $O(n(log(n) + log(W)))$.