Approximation Algorithms Workshop June 13-17, 2011, Princeton

Lecture An exp. lower bound for Zadeh's rule — June 14th, 2011

OLIVER FRIEDMANN *Scribe: TILTE FINO*

1 Overview

In this lecture we learn a deep connection between the simplex algorithm for solving linear programs and the policy iteration algorithm for solving Markov decision processes. Particularly, we see how an exponential lower bound for Zadeh's variant of the simplex algorithm can be obtained by constructing suitable Markov decision processes.

2 Linear Programming and Simplex

A *linear program* consists of a linear objective function that is to be maximized subject to some given linear constraints. The standard algorithm for solving linear programs is the *simplex algorithm* that operates by starting on one of the vertices of the convex polytope of the linear program, moving along improving edges to adjacent vertices until an optimal vertex has been found.

The runtime of the algorithm depends crucially on the so-called *pivoting rule*: the method of choosing adjacent improving vertices in case there is more than one potential improving successor. Many deterministic rules of the literature have been shown to have exponential lower bounds. Along the lines of this lecture, exponential lower bounds have been shown for the most prominent randomized pivoting rules for solving linear programs. In this lecture, we see how a lower bound can be obtained for Zadeh's rule, a history-based pivoting rule.

3 Games and Policy Iteration

A *Markov decision process* is a game that is played between a deterministic and a randomization player on a directed, total graph: every node belongs either to the deterministic or to the randomization player; outgoing edges of the randomization player are labelled with probabilities, all nodes are labelled with some rational reward.

A Markov decision process is played by putting a pebble on a vertex and moving it along the edges. If the pebble lies on a node of the deterministic player, it his turn to choose the successive node, and if the pebble lies on a randomization node, the successive node is determined arbitrarly at random. Every play finally ends in a sink node (unichain condition). The outcome is the total reward obtained by summing over the observed rewards.

The *objective* of the deterministic controller is to maximize the expected total reward by a *positional policy*, which is a selction of outgoing edges of nodes owned by the controller, s.t. exactly one outgoing edge per controller node is chosen.

In order to find the optimal strategy, one applies the *policy iteration algorithm*: starting with an arbitrary policy, one computes the expected total reward w.r.t. this policy, and considers whether there are any *improving switches*. An improving switch is an edge not selected by the current policy that gives a better expected total reward w.r.t. the current policy. If there are no improving switches, an optimal policy has been found. Otherwise, altering the current policy to use the improving switch results in a strictly better policy. Iterating this process allows to find the optimum.

4 Policy Iteration and Simplex

Every Markov decision process gives rise to an associated linear program: optimal policies in the game correspond to optimal solutions of the linear program. But more surprisingly, every policy corresponds to a vertex of the polytope (and vice versa), and every improving switch of the Markov decision process correspodns to a pivoting variable in the simplex algorithm. Hence, every family of Markov decision processes that gives a lower bound for policy iteration can be used to directly obtain a lower bound for the simplex algorithm.

5 Lower Bound for Zadeh's Rule

Lower bound constructions are obtained by building a family of Markov decision processes G_n of polynomial size s.t. G_n implements a binary counter with n bits. The construction are inspired by so-called *parity games* which use only exponentially growing rewards and penalties: To get a higher reward the Markov decision process is willing to sacrifice everything that has been built up so far.

Zadeh's pivoting rule is specified by performing the improving switch that has been applied least often. In some sense, this rule tries to be *fair*. However, it is not completely specified. In case of a tie of least applied switches, it is not clear which improving switch is to take. In the construction presented in the lecture, the tie-breaking rule is left implicit. In other words, the constructor of the lower bound is in charge of giving the tie-breaking.

The most complicated part of the construction is to implement a *fair binary counter*. If we consider switching a bit from 0 to 1 and back to be an improving switch, then it is immediate to see that more significant bits of a binary counter are switched less often than less significant bits. But then Zadeh's rule would ensure that higher bits are switched before they are supposed to be switched.

In order to solve this problem, the construction represents bits of the binary counter in a clever way. First, a bit is represented by a conjunction of two bits s.t. a bit of the counter is set iff both conjunctive bits are set. By this, the *occurrence record* of Zadeh's algorithm can be balanced out to some extent: by setting the first conjunctive bit of every counter bit gadget, then setting the second conjunctive bit of the least unset counter bit (which sets the least unset counter bit), then resetting all other first conjunctive bits, and then setting all other second conjunctive bits, the occurrence record can be kept in balance.

However, counter bits which are set, cannot switch back and forth, and hence, build up an unbalanced occurrence record. After unsetting a counter bit which has been set for a long time, the associated gadget has an extremely low occurrence record, and is therefore set again immediately after, which is not supposed to happen. Therefore, the second trick is introduced: every conjunc-

tive gadget is copied s.t. every counter bit now consists of two *representatives* of the conjunctive structure. The first representative is *active* iff the next counter bit is set, and otherwise the second representative is *active*. Hence, unsetting a bit can only happen by switching the next higher bit, the unsetting bit immediately becomes an inactive representative, and hence, its occurrence record can catch up with the rest while the other representative is active.

By this construction, fair binary counting can be implemented. The bit gadgets with representatives are glued together in a large Markov decision process that gives the complete lower bound construction. A full description is available on the speaker's website.

6 Open Problems

The following open problems are associated with the lecture:

- Strongly polytime algorithm for Markov decision processes and linear programming
- Polytime algorithms for similar two-player games
- Resolving the Hirsch conjecture
- Find other lower bounds for related pivoting rules with unknown status