Reinforcement Learning in Buchberger's Algorithm

Dylan Peifer
Cornell University

Summary

1. Buchberger's algorithm is the standard method for computing a Gröbner basis, and highly-tuned and optimized versions are a critical part of many computer algebra systems.
2. The efficiency of Buchberger's algorithm strongly depends on a choice of selection strategy that determines the order in which S-polynomials are processed.
3. By phrasing Buchberger's algorithm as a reinforcement learning problem and applying standard reinforcement learning techniques we can learn new selection strategies that can match or beat the existing state-of-the-art.

Gröbner Bases

Let $R = K[x_1, \dots, x_n]$ be a polynomial ring over some field K and $I = \langle f_1, \dots, f_k \rangle \subseteq R$ be a nonzero ideal generated by polynomials $f_1, \dots, f_k$.

Given a monomial order, a *Gröbner basis* G of a I is a subset $\{g_1, g_2, \dots, g_s\} \subseteq I$ such that any of the following equivalent conditions hold:

(i) $f^G \to 0 \iff f \in I$

(ii) f^G is unique for all $f \in R$

(iii) $\langle \mathrm{LT}(g_1), \mathrm{LT}(g_2), \dots, \mathrm{LT}(g_s) \rangle = \langle \mathrm{LT}(I) \rangle$

where $\mathrm{LT}(g)$ is the lead term of g with respect to the monomial order and $f^G \to r$ is the remainder under polynomial long division of f by the polynomials in G.

Buchberger's Algorithm

Theorem (Buchberger's Criterion): Let $G = \{g_1, g_2, \dots, g_s\}$ generate some ideal I. If $S(g_i, g_j)^G \to 0$ for all pairs g_i, g_j, where

$$S(g_i, g_j) = \frac{\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(g_j))}{\mathrm{LT}(g_i)} g_i - \frac{\mathrm{lcm}(\mathrm{LT}(g_i), \mathrm{LT}(g_j))}{\mathrm{LT}(g_j)} g_j$$

is the *S-polynomial* of g_i and g_j, then G is a Gröbner basis of I.

Algorithm 1 Buchberger's Algorithm

```
input a set of polynomials {f_1, ..., f_k}
output a Gröbner basis G of I = <f_1, ..., f_k>
procedure BUCHBERGER({f_1, ..., f_k})
    G ← {f_1, ..., f_k}                         ▷ the current basis
    P ← {(f_i, f_j) | 1 ⩽ i < j ⩽ k}            ▷ the remaining pairs
    while |P| > 0 do
        (f_i, f_j) ← select(P)
        P ← P \ {(f_i, f_j)}
        r ← S(f_i, f_j)^G
        if r ≠ 0 then
            G ← G ∪ {r}
            P ← P ∪ {(f, r) : f ∈ G}
        end if
    end while
    return G
end procedure
```

Selection Strategies in Buchberger's Algorithm

The implementation of select does not affect correctness of Buchberger's algorithm, but it is critical for efficiency. In general, good selection strategies pick "small" pairs first.

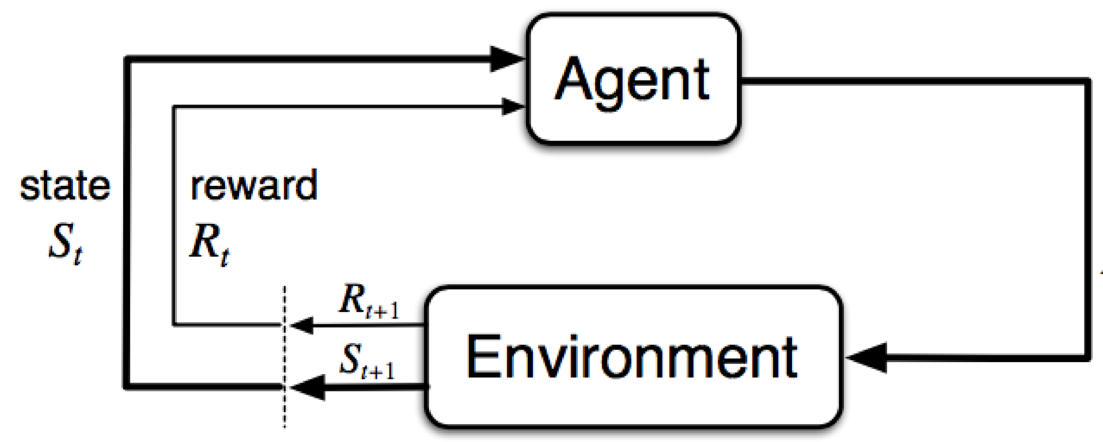
- First: among the pairs with minimal j, pick the pair with smallest i
- Degree: pick the pair with smallest degree of $\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$
- Normal: pick the pair with smallest $\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$ in the monomial order
- Sugar: pick the pair with smallest sugar degree of $\mathrm{lcm}(\mathrm{LT}(f_i), \mathrm{LT}(f_j))$

Pair Reductions in Buchberger's Algorithm per Strategy

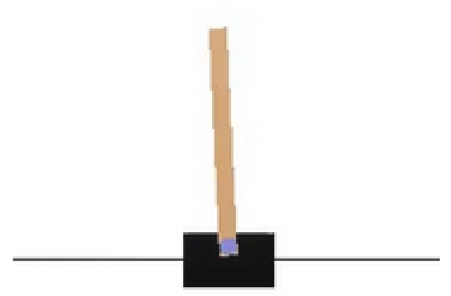
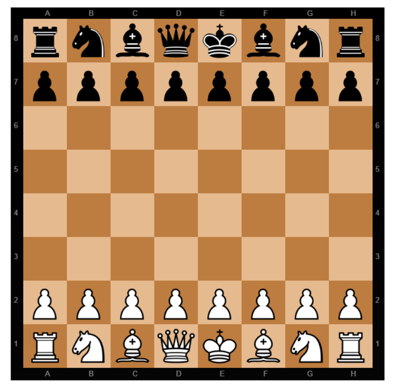
example	First	Normal	Sugar	Random	Last	Strange	Spice
cyclic4	11	11	11	14	21	23	23
reimer3	25	23	25	25	25	28	28
katsura5	28	28	28	44	76	86	86
eco6	67	61	64	97	149	295	295
noon4	71	71	71	100	103	375	375
cyclic6	366	620	343	793	-	-	-
katsura7	164	164	164	285	-	-	-
katsura4-lex	25	46	19	29	44	30	59
eco5-lex	30	22	26	28	91	32	97
cyclic5-lex	104	1602	108	-	-	-	-

Reinforcement Learning

Reinforcement learning problems can be phrased as the interaction of an agent and an environment.



The agent chooses actions and the environment processes actions and gives back the updated state and a reward. The agent wants to maximize its return, which is the amount of reward it gets in the long run.

CartPole	Chess	Buchberger
		$G = \{x^2 - y^3, xy^2 + x, -y^5 - y^3\}$ $P = \{(1,3), (2,3)\}$
S_t = cart/pole position/velocity A_t = push the cart left or right R_t = 1 while pole is upright	S_t = positions of all pieces A_t = a valid move R_t = 1 if you win at t, otherwise 0	S_t = current basis and pair set A_t = a pair from the pair set R_t = -1 until the pair set is empty

Policies and Trajectories

A *policy* π is a function $\pi : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ given by

$$\pi(a|s) = Pr(A_t = a | S_t = s)$$

which maps state-action pairs to the probability of choosing the given action in the given state.

Policies are often viewed as functions that take in a state and return a probability distribution on actions. An agent follows a policy by applying the policy to its current state and sampling from the returned probability distribution to choose the next action.

A *trajectory* or *rollout* τ of a policy π is a series of states, actions, and rewards

$$\tau = (S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \dots, R_T, S_T)$$

obtained by following the policy π one time through the environment, and the *return* of a trajectory is the sum of rewards along the trajectory.

Given an environment, the goal of reinforcement learning is to find a policy π that maximizes

$$\mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=1}^{T} R_t \right]$$

which is the expected return along trajectories τ obtained by following π.

Policy Gradient

Suppose π_θ is a parametrized policy that is differentiable with respect to its parameters θ. Then the expected return

$$J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^{T} R_t \right]$$

has gradient

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(A_t|S_t) \sum_{t'=t+1}^{T} R_{t'} \right].$$

This expectation is a quantity we can sample by interacting with the environment. By starting with any set of parameters θ_1 and updating by gradient ascent steps

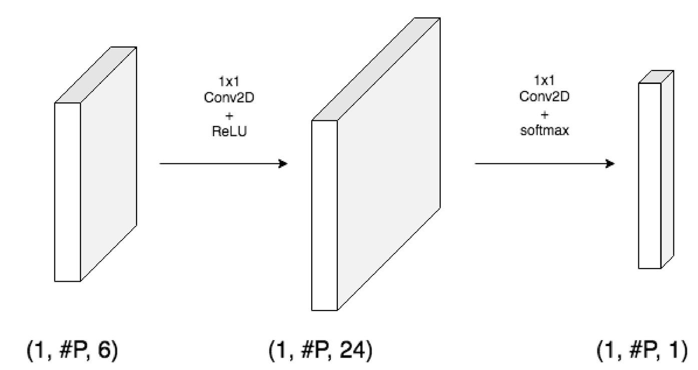
$$\theta_k = \theta_{k-1} + \alpha \cdot \nabla_\theta J(\theta_k)$$

for some small learning rate α, we can incrementally improve the policy. Intuitively, we should increase the probability of taking the action we chose proportional to the future reward we received and the derivative of the log probability of choosing that action again.

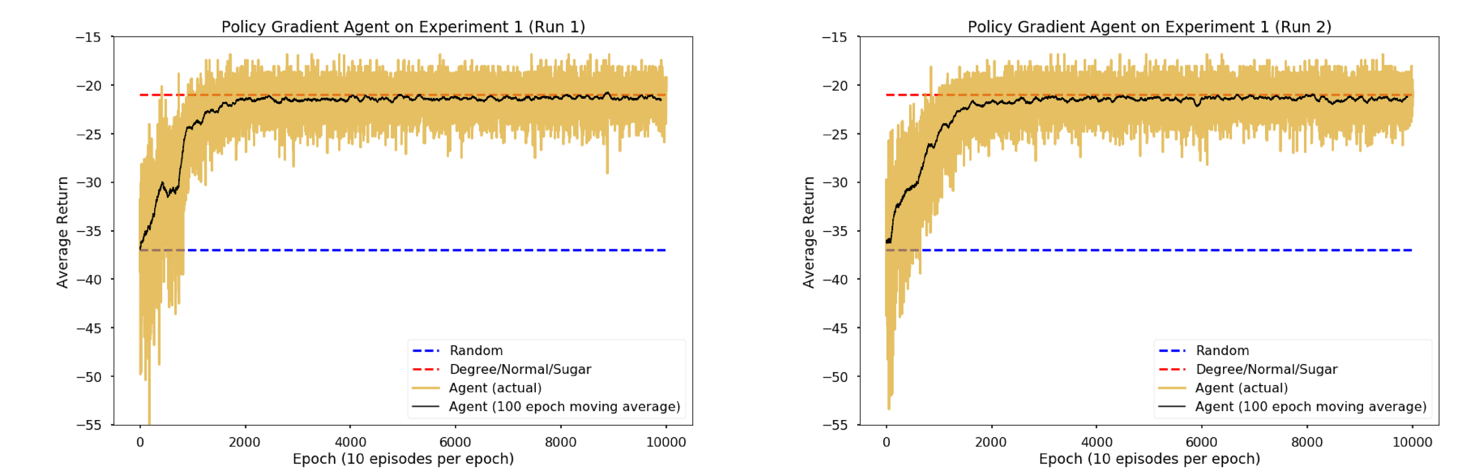
Experiment 1: 5 Homogeneous Binomial Quadrics

Let $R = \mathbb{Z}/32003[x, y, z]$ with grevlex ordering. Consider ideals I generated by 5 random binomial quadrics. Perform Buchberger with no pair elimination.

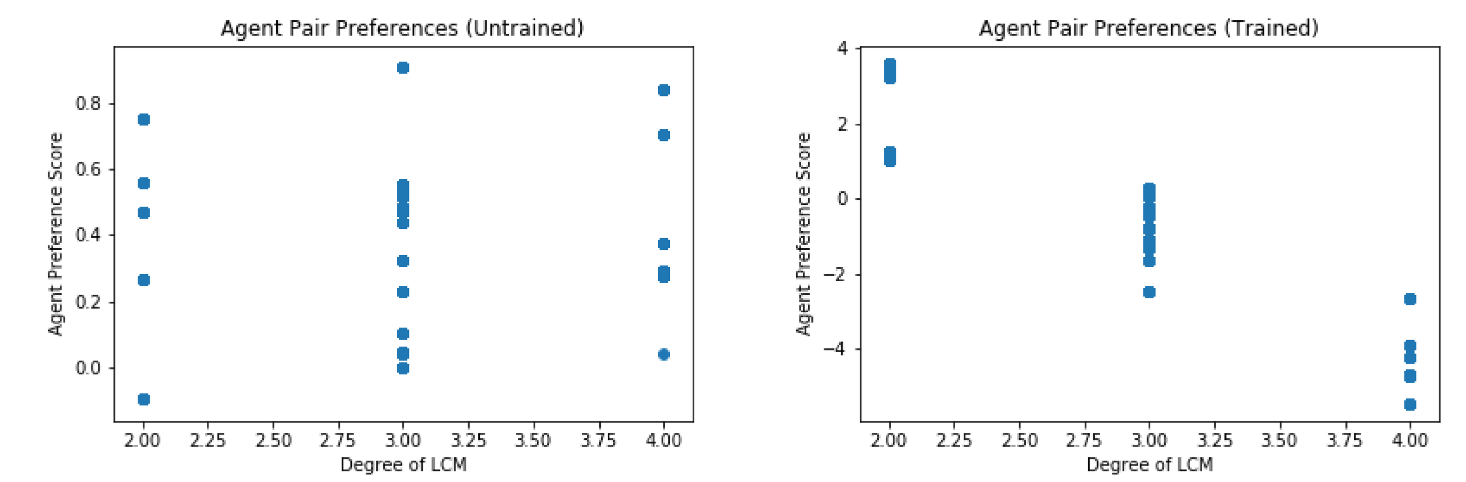
$$G = \{xy + 3y^2, x^2 + 6y^2, xy + 8xz, y^2 - z^2\}$$
$$P = \{(1,2), (1,4), (2,4), (3,4)\}$$
$$\downarrow$$
$$\begin{bmatrix} 1 & 1 & 0 & 2 & 0 & 0 \\ 1 & 1 & 0 & 0 & 2 & 0 \\ 2 & 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 & 2 & 0 \end{bmatrix}$$



Convert the state $S_t = (G, P)$ to a matrix with rows the exponent vectors of the lead terms of each pair. Each step input this matrix to a neural network that learns the policy function.



In each epoch we perform 10 rollouts, compute future rewards for each state on each trajectory, baseline by the size of the current pair set in each state, and normalize these scores before performing the policy gradient step. Total training time was 45 minutes.

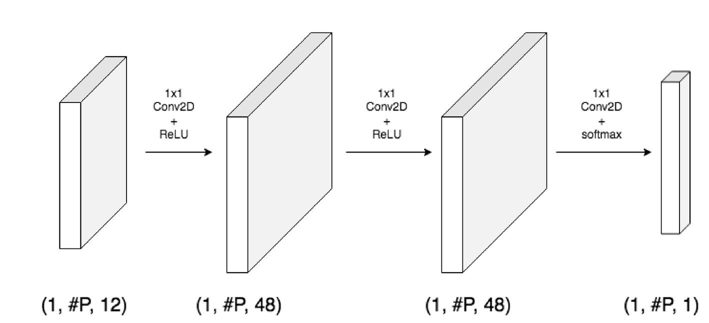


By examining the agent's preferences for picking different pairs we see that it has learned an approximation to degree selection, the best strategy in this case.

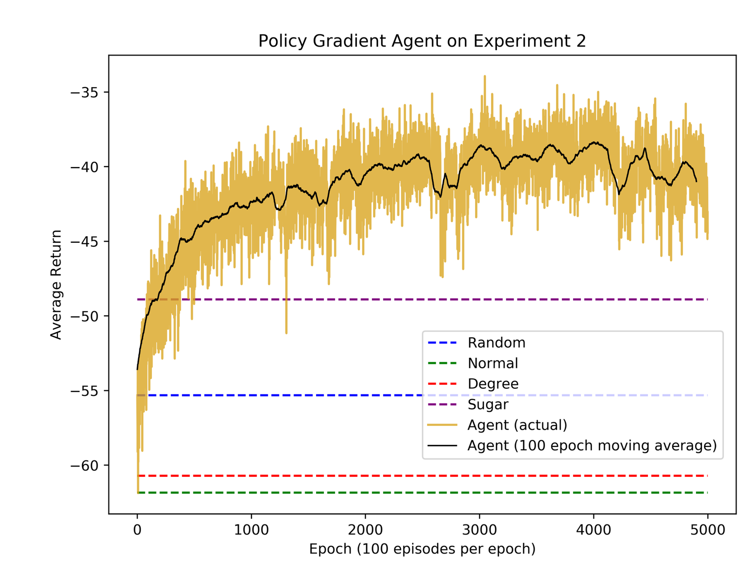
Experiment 2: 10 Nonhomogeneous Binomials of Degree $\leqslant 20$

Let $R = \mathbb{Z}/32003[x, y, z]$ with grevlex ordering. Consider ideals I generated by 10 random binomials of degree $\leqslant 20$. Perform Buchberger with Gebauer-Möller pair elimination.

$$G = \{xy + 3y^2, x^2 + 6y^2, xy + 8xz, y^2 - z^2\}$$
$$P = \{(1,2), (1,4), (2,4), (3,4)\}$$
$$\downarrow$$
$$\begin{bmatrix} 1 & 1 & 0 & 0 & 2 & 0 & 2 & 0 & 0 & 0 & 2 & 0 \\ 1 & 1 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 \\ 2 & 0 & 0 & 0 & 2 & 0 & 0 & 2 & 0 & 0 & 0 & 2 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 2 & 0 & 0 & 0 & 2 \end{bmatrix}$$



Convert the state $S_t = (G, P)$ to a matrix with rows the exponent vectors of both terms of each pair. Each step input this matrix to a neural network that learns the policy function.



After 12 hours of training the agent has learned a policy that averages 20% fewer pair reductions than the best known selection strategies.