Learning Fast Monomial Orders for Gröbner Basis Computations

R. Caleb Bunch[1], Alperen A. Ergür[2], Melika Golestani[2], Jessie Tong[3], Malia Walewski[4], and Yunus E. Zeytuncu[5]

[1]Georgia Institute of Technology
[2]The University of Texas at San Antonio
[3]San Diego State University
[4]Emory University
[5]University of Michigan–Dearborn

• except A.E. and Y.Z. all are undergraduate researchers!

Plan for Today

1) RL recap → Half the talk almost.

2) Solving polynomial equations, why bother?

3) Gröbner basis definitions
   - Monomial order
   - Gröbner bases
   - Gröbner fan
   - F4, pair-selection

4) RL problem set-up and experiments

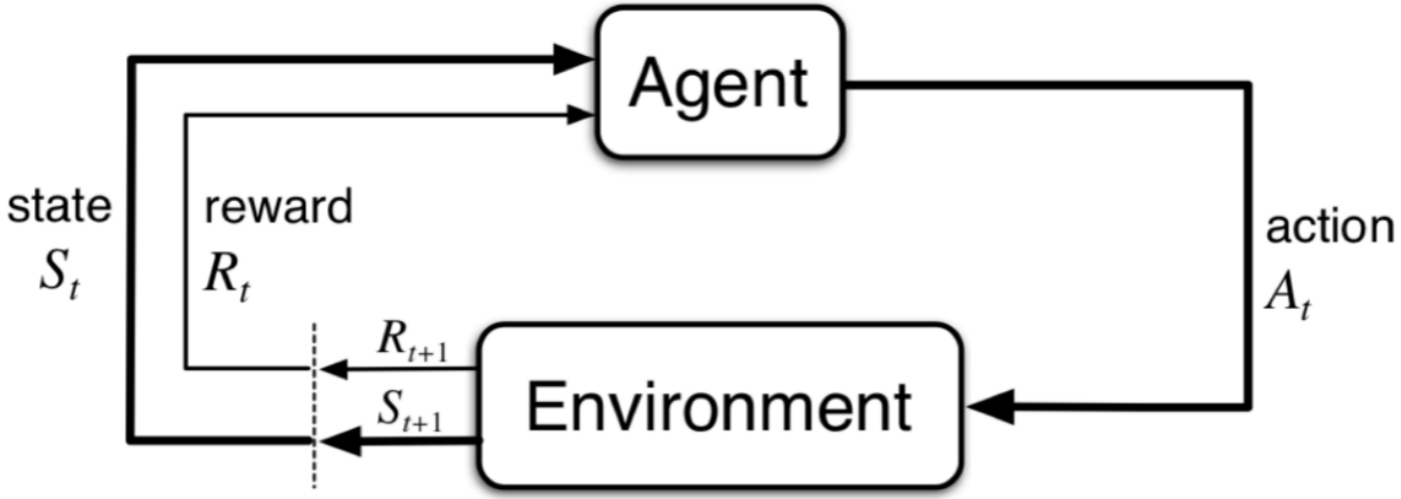
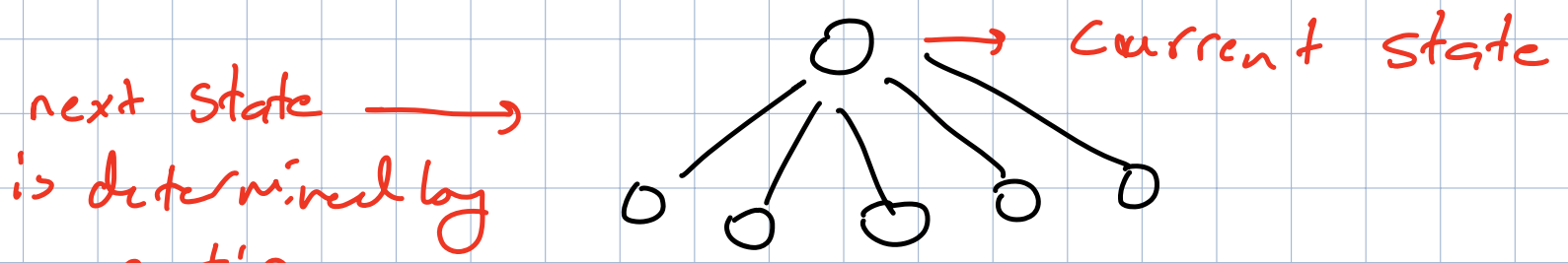


Figure 3.1: The agent–environment interaction in a Markov decision process.

Markov Decision Process (MDP)

is a sequential decision making environment



RL setting

- State-space → states agent travels

- Action-space → actions agent can take
  - can be discrete or continuous
  - can be updated after every step or could be fixed

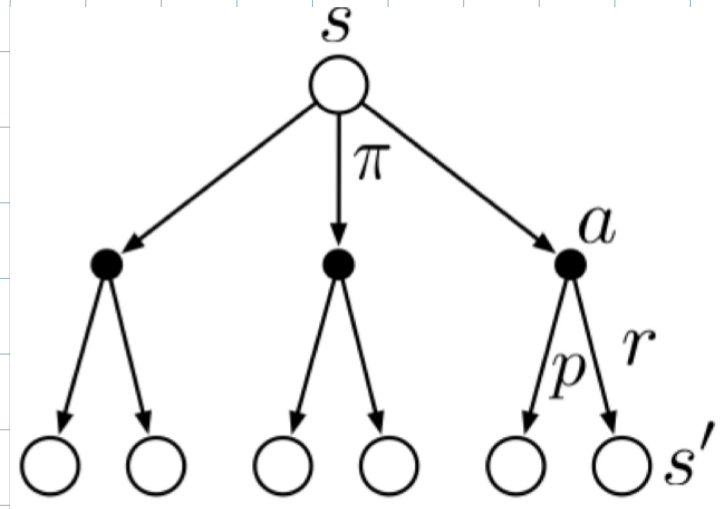
- Reward → the agent aims to maximize cumulative reward

$0 < \gamma < 1$ → discount factor

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

- Policy → agent's strategy to pick the action in a given state

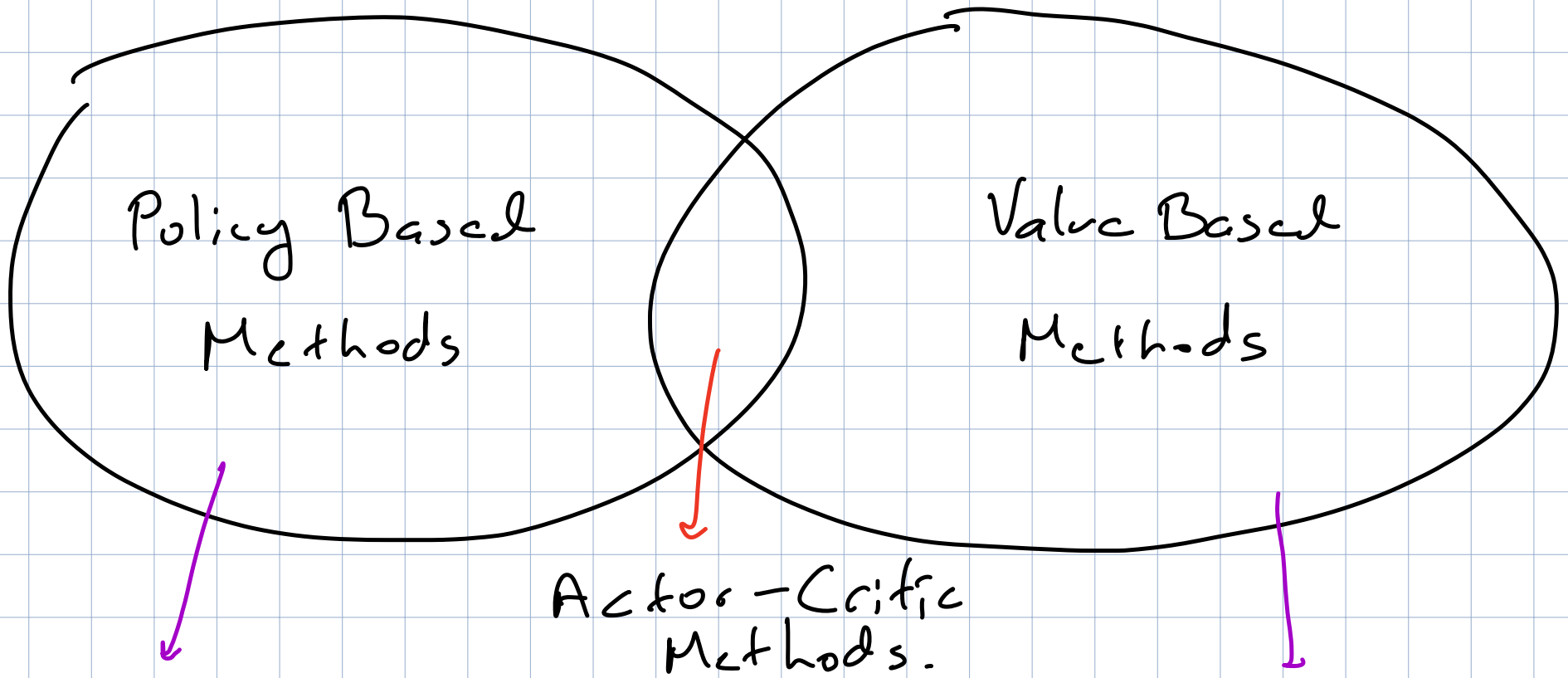
$$\pi : A \rightarrow S \qquad \sum_{a \in A} \pi(a, s) = 1$$



Backup diagram for v_π

π assigns value to states

$$q(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k} \mid \begin{matrix} S_t = s \\ A_t = a \end{matrix} \right]$$



optimize the policy without learning the value function

Iterate over value function space, turn to a policy at the end.

Policy Based optimization

- θ parameter of policy n-net
- Assume a probability measure on state-space
- Average return of policy π_θ is

$$J(\theta) = \mathop{\mathbb{E}}_{s} [\text{cumulative reward of } s \mid \pi_\theta]$$

- Goal :

$$\arg\max_{\theta} J(\theta)$$

Policy Gradient Theorem

- A trajectory $\tau : s_0, a_0, R_0, s_1, a_1, R_1, \cdots, R_T$

$\pi_\theta(s, a)$ → probability of taking action a at state s w.r.t π_θ

$p(s, a, s', r)$ → when action a is taken the probability of going to s' and reward r

$$P(\tau) = \prod_{i=0}^{T} \pi_\theta(a_i) \cdot p(s_i, a_i, s_{i+1}, r_i)$$

Goal: Find θ so that expected reward is maximized

Policy Gradient Theorem

$$\nabla_\theta \log P(\tau) = \frac{\nabla_\theta P(\tau)}{P(\tau)} = \nabla_\theta \sum_{i=1}^{T} \log \pi_\theta(a_i)$$

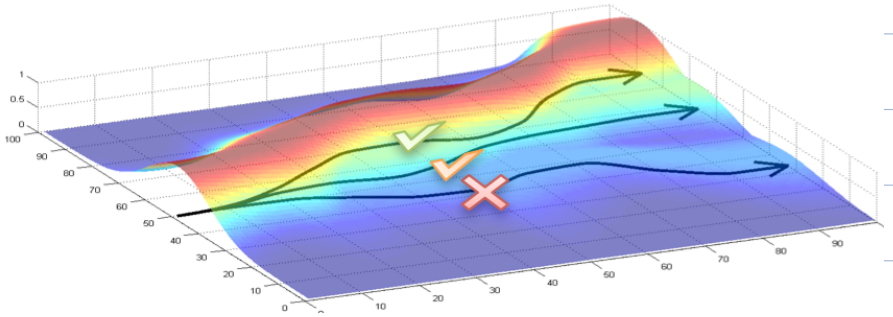
$R(\tau) :=$ discounted reward of path τ

$$\nabla_\theta J(\theta) \propto \sum_{\tau} \nabla_\theta \log P(\tau) \cdot P(\tau) \cdot R(\tau)$$

Empirical Version:

$$\frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau_i)\, R(\tau_i)$$

- Gradient tries to:
  - Increase probability of paths with positive R
  - Decrease probability of paths with negative R



Baseline Trick

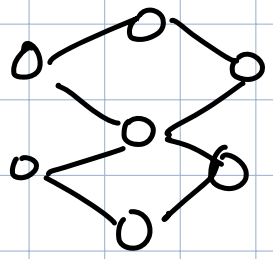
$$\overline{J(\theta)} = \sum \pi_\theta p(\tau) (R(\tau) - b)$$

b: anything that does not depend on θ.

$$\nabla \overline{J(\theta)} = \nabla J(\theta)$$

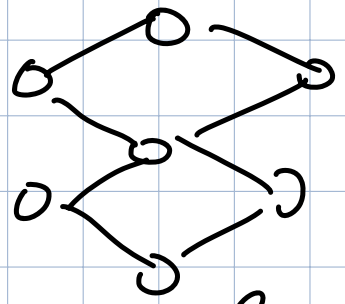
- a good baseline reduces the noise in training

Actor-Critic Style



Actor

Policy
brain



Valuefunction
estimator

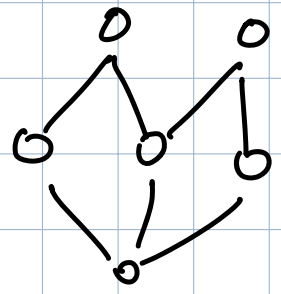
→ Critic
of the action
suggested by
policy $\pi_{\theta}(s_t)$

Prioritized - Replay - Buffer Trick

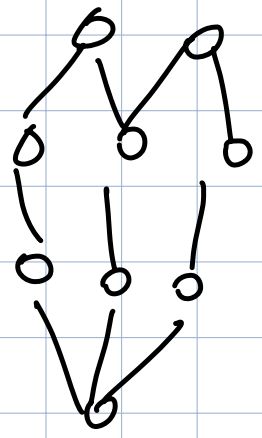
- a data structure to create "artificial experiences" for the agent.

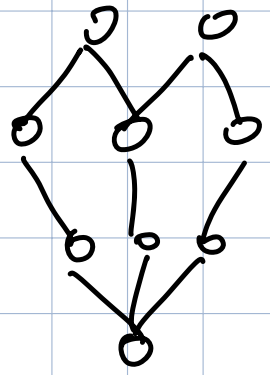
- DDPG: Policy gradients works for deterministic policies in continuous action space as well.

- TD3: DDPG is very noisy. Here is how we stabilize it



actor n.net





two critics

- at every step we update the critics with the reward collected, but delay the update of actor.

Algorithm 1 TD3

Initialize critic networks Q_{θ_1}, Q_{θ_2}, and actor network π_ϕ with random parameters θ_1, θ_2, ϕ
Initialize target networks $\theta_1' \leftarrow \theta_1$, $\theta_2' \leftarrow \theta_2$, $\phi' \leftarrow \phi$
Initialize replay buffer $\mathcal{B}$
for $t = 1$ **to** T **do**
  Select action with exploration noise $a \sim \pi_\phi(s) + \epsilon$, $\epsilon \sim \mathcal{N}(0, \sigma)$ and observe reward r and new state s'
  Store transition tuple (s, a, r, s') in $\mathcal{B}$

  Sample mini-batch of N transitions (s, a, r, s') from $\mathcal{B}$
  $\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$
  $y \leftarrow r + \gamma \min_{i=1,2} Q_{\theta_i'}(s', \tilde{a})$
  Update critics $\theta_i \leftarrow \text{argmin}_{\theta_i} N^{-1} \sum (y - Q_{\theta_i}(s, a))^2$
  if t mod d **then**
    Update ϕ by the deterministic policy gradient:
    $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
    Update target networks:
    $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$
    $\phi' \leftarrow \tau\phi + (1 - \tau)\phi'$
  end if
end for

Solving Polynomials

$$\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in \mathbb{Z}^n \qquad x^\alpha := x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_n^{\alpha_n}$$

Ex: $\alpha = (2, 3) \qquad x^\alpha = x_1^2 x_2^3$

Support set: $A = \{ a_1, a_2, \dots, a_t \} \subseteq \mathbb{Z}^n$

$$a_i = (a_{i1}, a_{i2}, \dots, a_{in}).$$

$$p_1(x) = \sum_{\alpha \in A} c_\alpha^{(1)} x^\alpha \quad , \quad p_2(x) = \sum_{\alpha \in A} c_\alpha^{(2)} x^\alpha \quad , \dots$$

$$\dots, \quad p_k(x) = \sum_{\alpha \in A} c_\alpha^{(1)} x^\alpha$$

Goal: Find all $u = (u_1, u_2, \ldots, u_m) \in \mathbb{R}^n$ such that

$$Z(p) = \{ u : p_1(u) = p_2(u) = \cdots = p_k(u) = 0. \}$$

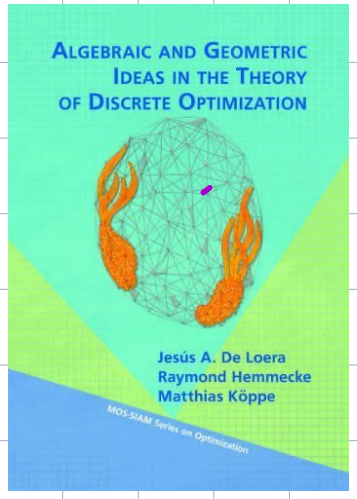
- Note that for any $u \in Z(p)$ and for any h s.t.

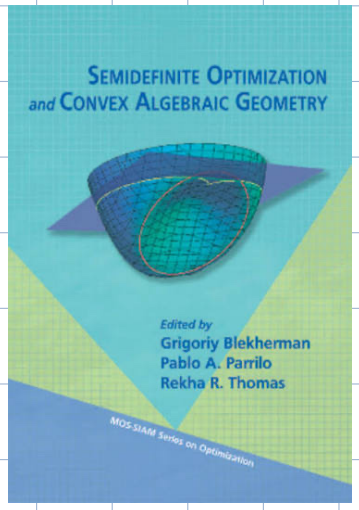
$$h = p_1(x) g_1(x) + p_2(x) g_2(x) + \cdots + p_n(x) g_n(x)$$

we have $h(u) = 0$.

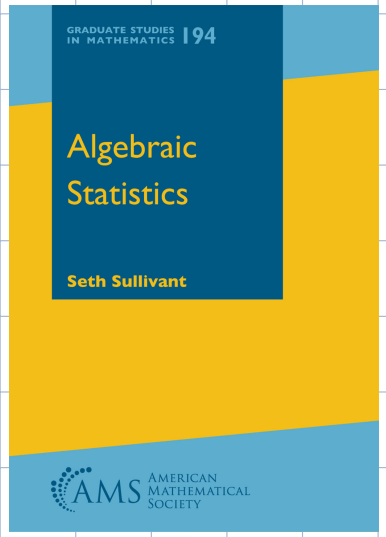
Why bother?

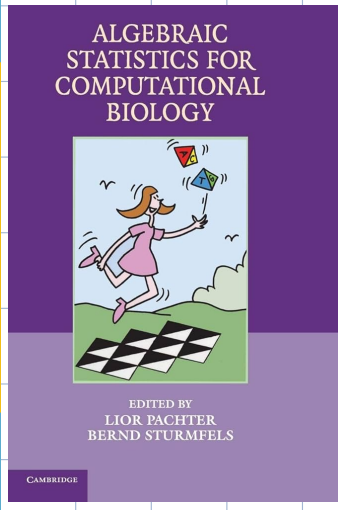
Optimization



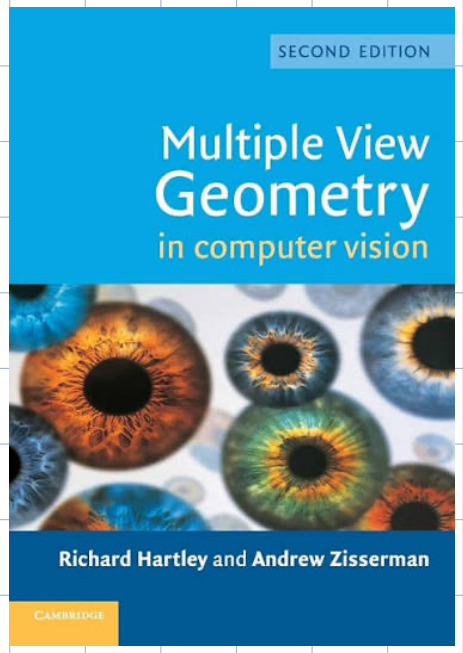


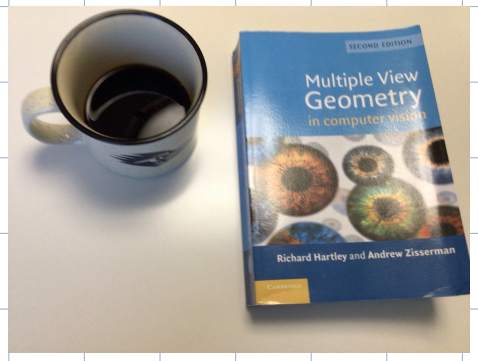
Statistics



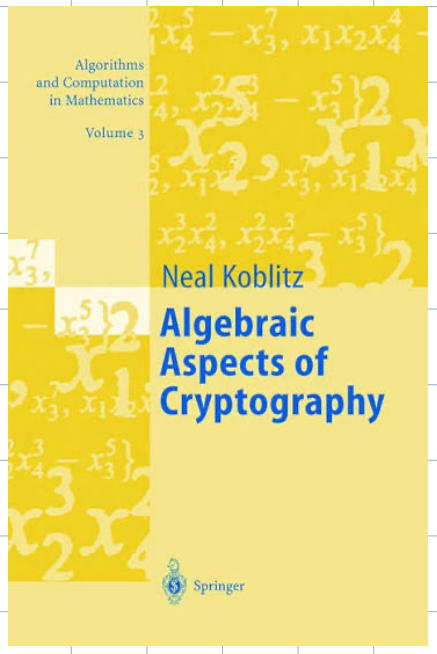


Computer Vision

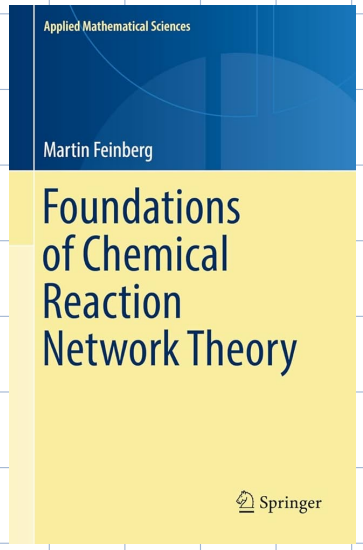




Cryptography



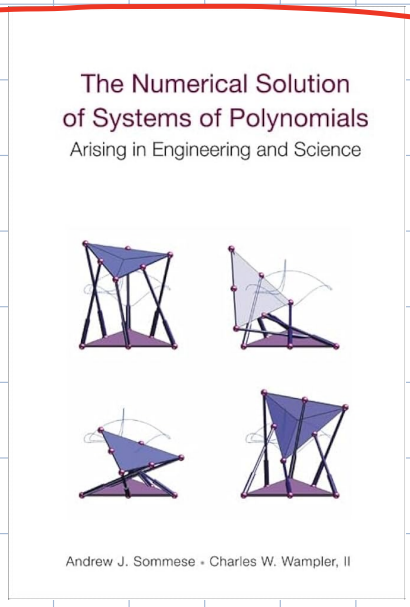
Systems Biology



Kinematics

Particle Physics

Power-Flow networks



Gröbner Basis

- Monomial Order $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$

How to compare α and β?

- the order should be a well ordering

no ties!

- If $\alpha > \beta$ then $\alpha + \gamma > \beta + \gamma$

for any γ.

Dfn: $p(x) = \sum_{\alpha \in A} c_\alpha x^\alpha$

The $\alpha \in A$ such that $\alpha > \beta$ for all $\beta \in A$

is called the leading term. $LT_>(p)$.

Dfn: $\langle p_1, p_2, \ldots, p_k \rangle = \{ h \in R[x] : h(x) = p_1(x) g_1(x) + \cdots + p_k(x) g_k(x) \}$

Dfn: A finite set of polynomials $g_1, g_2, \ldots, g_m$ is a Gröbner basis for $p_1, p_2, \ldots, p_k$ w.r.t $\succ$ if

$$\langle LT(g_1), LT(g_2), \ldots, LT(g_m) \rangle = \langle LT(p_1), \ldots, LT(p_k) \rangle$$

Buchberger's Algorithm

- Fix a monomial order, and a pair selection strategy.
- Start with $p_1, \ldots, p_k$, order, pair-select, reduce until it stops.

A simple way to encode monomial orders

- Pick $v \in \mathbb{R}^n$

  Compute $\langle v, \alpha \rangle$ for all $\alpha \in A$

- If $\langle \alpha, v \rangle > \langle \beta, v \rangle$

  then say $\alpha > \beta$

- Create a tie-breaking rule in case

  $\langle \alpha, v \rangle = \langle \beta, v \rangle$

Note: Scaling v does not change the order.

Infinitely many orders actually create finitely many different Gröbner basis.

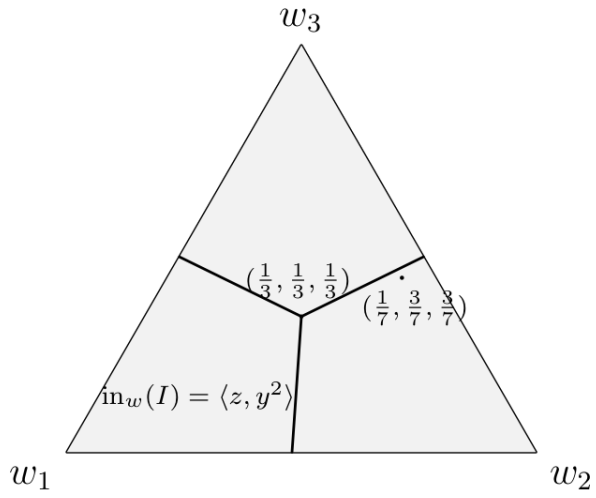


Figure 1: A schematic two-dimensional slice ($w_1 + w_2 + w_3 = 1$, $w \geq 0$) of the Gröbner fan for $I = \langle x + y + z,\ x^3z + x + y^2 \rangle$, illustrating how regions of constant $\mathrm{in}_w(I)$ arise. The diagram is not to scale and is intended to convey the geometric structure of cones rather than exact fan boundaries; see [23].

In practice,

Instead of optimizing for fastest monomial order, community defaulted to a few monomial orders.

Lex, Grad Lex, Reverse Grad Lex

We object!

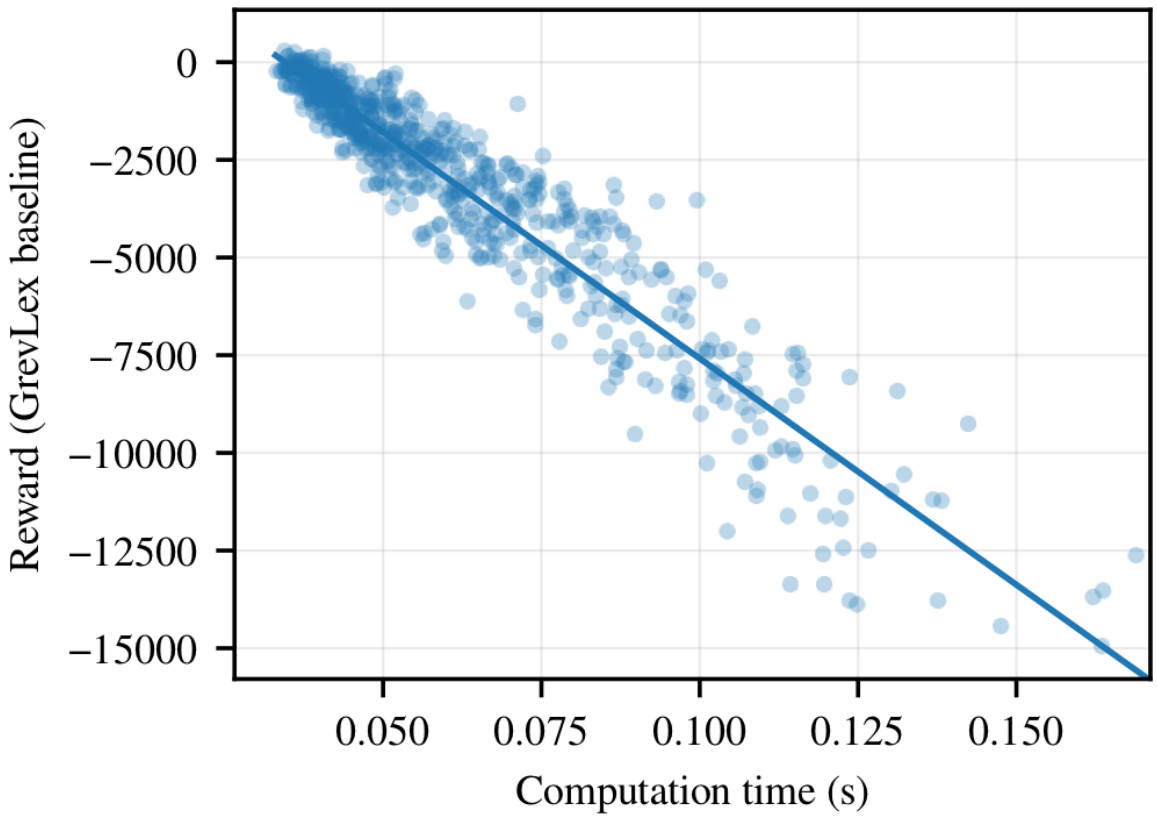
Make monomial order selection a RL-problem.

Computational Setup

- We had to use Julia because Python packages only allow few fixed monomial orders
- We used Groebner.jl. It implements F4 algorithm for Gröbner basis computation
- F4 does several reductions at the same time to speed-up computations.

Our reward function utilizes the Gröbner trace statistics of the F4 algorithm recorded by `Groebner.jl`, specifically: the number of iterations (t), the number of columns in the elimination matrix ($n_M(i)$) for $1 \leq i \leq t$, the number of pairs ($n_P(i)$), and the degree of pairs ($d_P(i)$). The reward function is defined as:

$$-\sum_{i=1}^{t} n_M(i) n_P(i) \ln(d_P(i))$$



State - Space

The `Groebner.jl` package accepts only integer entries in weight vectors for the monomial ordering. However, the theoretical search space is the simplex:

$$\Delta_n := \{x = (x_1, x_2, \ldots, x_n) : x_i \geq 0, \sum x_i = 1\}$$

Therefore, we define the action space as the simplex and define a monomial ordering by scaling the action vector by 10^3 and rounding to the nearest integer.

Action - Space

Steps inside Δ_n that has norm at most δ.

- We tested on four problems from Computer Vision and Systems Biology.

	GrevLex baseline			GrLex baseline		
Polynomial System	**Win/Tie/Loss**	**Improvement**	**Degradation**	**Win/Tie/Loss**	**Improvement**	**Degradation**
Relative Pose	98.4 / 0.0 / 1.6	19.43	−1.27	32.2 / 16.1 / 51.6	7.95	−7.04
Triangulation	27.0 / 66.1 / 7.0	0.86	−0.72	100 / 0 / 0	37.62	–
n-Site Phos. (n=14)	100 / 0 / 0	70.77	–	100 / 0 / 0	70.77	–
Wnt Shuttle	91.6 / 0.0 / 8.4	54.64	−26.40	86.9 / 0.0 / 13.1	55.25	−20.83

Relative Pose
Triangulation
> vision

n-Site Phos
Wnt Shuttle
> Bio

Thank you!

Danke! Gracias! Teşekkürler!

Ευχαριστώ!

متشکرم!

有り難う!