

Learning Selection Strategies in Buchberger’s Algorithm

Dylan Peifer
Department of Mathematics
Cornell University
Ithaca, NY 14853

Michael Stillman
Department of Mathematics
Cornell University
Ithaca, NY 14853

Daniel Halpern-Leistner
Department of Mathematics
Cornell University
Ithaca, NY 14853

ABSTRACT

Studying the set of exact solutions of a system of polynomial equations largely depends on a single iterative algorithm, known as Buchberger’s algorithm. Optimized versions of this algorithm are crucial for many computer algebra systems (e.g., Mathematica, Maple, Sage). We introduce a new approach to Buchberger’s algorithm that uses reinforcement learning agents to perform S-pair selection, a key step in the algorithm. We then study how the difficulty of the problem depends on the choices of domain and distribution of polynomials, about which little is known. Finally, we train a policy model using proximal policy optimization (PPO) to learn S-pair selection strategies for random systems of binomial equations. In certain domains, the trained model outperforms state-of-the-art selection heuristics both in number of iterations of the algorithm and total number of polynomial additions performed. These results provide a proof-of-concept that recent developments in machine learning have the potential to improve performance of algorithms in symbolic computation.

KEYWORDS

reinforcement learning, Groebner bases, algebraic geometry

1 INTRODUCTION

Systems of multivariate polynomial equations, such as

$$\begin{cases} 0 = f_1(x, y) = x^3 + y^2 \\ 0 = f_2(x, y) = x^2y - 1 \end{cases} \quad (1)$$

appear in many scientific and engineering fields, as well as many subjects in mathematics. The most fundamental question about such a system of equations is whether there exists an exact solution. If one can express the constant polynomial $h(x, y) = 1$ as a combination

$$h(x, y) = a(x, y)f_1(x, y) + b(x, y)f_2(x, y) \quad (2)$$

for some polynomials a and b , then there can be no solution, because the right hand side vanishes at any solution of the system.

The converse also holds: the set of solutions with x and y in \mathbb{C} is empty if and only if there exists a linear combination (2) for $h = 1$ [24]. Thus the existence of solutions to (1) can be reduced to the larger problem of determining if a polynomial h lies in the *ideal* generated by these polynomials, which is defined to be the set $I = \langle f_1, f_2 \rangle$ of all polynomials of the form (2).

The key to solving this problem is to find a *Gröbner basis* for the system. This is another set of polynomials $\{g_1, \dots, g_k\}$, potentially much larger than the original set, which generate the same ideal $I = \langle f_1, f_2 \rangle = \langle g_1, \dots, g_k \rangle$, but for which one can employ a version of the Euclidean algorithm (discussed below) to determine if $h \in I$.

In fact, computing a Gröbner basis is the necessary first step in algorithms which answer a huge number of questions about

the original system: eliminating variables, parametrizing solutions, studying geometric features of the solution set, etc. This has led to a wide array of scientific applications of Gröbner bases, wherever polynomial systems appear, including: computer vision [15], cryptography [20], biological networks and chemical reaction networks [4], robotics [1], statistics [13, 40], string theory [23], signal and image processing [28], integer programming [10], coding theory, and splines [11].

Buchberger’s algorithm (see e.g. [12]) is the basic iterative algorithm used to find a Gröbner basis. As it can be costly in both time and space, this algorithm is the computational bottleneck in many applications of Gröbner bases. Despite this, all direct algorithms for finding Gröbner bases are variations of Buchberger’s algorithm (e.g. [16, 18, 19, 35]). Highly optimized versions of the algorithm are a key piece of computer algebra systems such as [9, 29–32, 36, 39].

There are several points in Buchberger’s algorithm which depend on choices that do not affect the correctness of the algorithm, but can have a significant impact on performance. In this paper we focus on one such choice, called *pair selection*.

We show that the problem of pair selection fits naturally into the framework of reinforcement learning, and claim that the rapid advancement in applications of deep reinforcement learning over the past decade has the potential to significantly improve the performance of the algorithm.

Our main contributions are the following:

- Initiating the empirical study of Buchberger’s algorithm from the perspective of machine learning.
- Identifying a precise sub-domain of the problem, consisting of systems of binomials, that is directly relevant to applications, captures many of the challenging features of the problem, and can serve as a useful benchmark for future research.
- Training a simple neural network model for pair selection which outperforms state-of-the-art selection strategies by 20% to 40% in this domain, thereby demonstrating significant potential for future research.

1.1 Related work

While we are not aware of any existing work applying machine learning to Buchberger’s algorithm, many authors have applied machine learning to algorithm performance in other domains. Models trained with supervised learning can perform algorithm selection, choose specialized parameters, or quickly predict the good choices produced by computationally expensive strategies [3, 25, 26]. Reinforcement learning provides the additional opportunity to learn entirely new heuristics and strategies inside algorithms. See [8] for an overview of recent ideas and progress.

Algorithm 1 Multivariate Division Algorithm

```

1: Input: a polynomial  $h$  and a set of polynomials  $F = \{f_1, \dots, f_s\}$ 
2: Output: a remainder polynomial  $r = \text{reduce}(h, F)$ 
3:  $r \leftarrow h$ 
4: while  $\text{LT}(f_i) \mid \text{LT}(r)$  for some  $i$  do
5:   choose  $i$  such that  $\text{LT}(f_i) \mid \text{LT}(r)$ 
6:    $r \leftarrow r - \frac{\text{LT}(r)}{\text{LT}(f_i)} f_i$ 
7: end while

```

2 GRÖBNER BASES

Let $R = K[x_1, \dots, x_n]$ be the set of polynomials in variables x_1, \dots, x_n with coefficients in some field K . (K was \mathbb{C} in the introduction, but in our experiments, K will be a large finite field.) Let $F = \{f_1, \dots, f_s\}$ be a set of polynomials in R , and consider $I = \langle f_1, \dots, f_s \rangle$ the ideal generated by F in R .

The definition of a Gröbner basis depends on a choice of *monomial order*, i.e., a well-order relation $>$ on the set of monomials $\{x^a = x_1^{a_1} \cdots x_n^{a_n} \mid a \in \mathbb{Z}_{\geq 0}^n\}$ such that $x^a > x^b$ implies $x^{a+c} > x^{b+c}$ for any exponent vectors a, b, c . Given a polynomial $f = \sum_a \lambda_a x^a$, we define the *leading term* $\text{LT}(f) = \lambda_a x^a$, where x^a is the largest monomial with respect to the ordering $>$ that has $\lambda_a \neq 0$. An important example is the *grevlex* order, where $x^a > x^b$ if the total degree of x^a is greater than that of x^b , or they have the same degree, but the *last* non-zero entry of $a - b$ is *negative*. For example, in the grevlex order, we have $x_1 > x_2 > x_3$, $x_2^3 > x_1 x_2$, and $x_2^2 > x_1 x_3$.

For a given choice of monomial order, and a set of polynomials $F = \{f_1, \dots, f_s\}$ the *multivariate division algorithm* takes any polynomial h and produces a polynomial r , called the remainder, written $r = \text{reduce}(h, F)$, such that $h - r \in \langle f_1, \dots, f_s \rangle$ and $\text{LT}(f_i)$ does not divide $\text{LT}(r)$ for any i . In this case we say that h *reduces to* r . In general the division algorithm is guaranteed to terminate, but the remainder can depend on the choice in line 5 of Algorithm 1.

DEFINITION 1. Given a monomial order, a Gröbner basis G of a nonzero ideal I is a set of generators $\{g_1, g_2, \dots, g_k\}$ of I such that any of the following equivalent conditions hold:

- (i) $\text{reduce}(h, G) = 0 \iff h \in I$
- (ii) $\text{reduce}(h, G)$ is unique for all $h \in R$
- (iii) $\langle \text{LT}(g_1), \text{LT}(g_2), \dots, \text{LT}(g_k) \rangle = \langle \text{LT}(I) \rangle$

where $\langle \text{LT}(I) \rangle = \langle \text{LT}(f) \mid f \in I \rangle$ is the ideal generated by the leading terms of all polynomials in I .

A consequence of (i) is that given a Gröbner basis G for $\langle f_1, \dots, f_s \rangle$, then the system of equations $f_i = 0, i = 1, \dots, s$ has no solutions if and only if $\text{reduce}(1, G) = 0$, that is, if G contains a non-zero constant polynomial. For an introduction to Gröbner bases and their uses, see [12, 34].

2.1 Buchberger's Algorithm

Buchberger's algorithm produces a Gröbner basis for the ideal $I = \langle f_1, \dots, f_s \rangle$ from the initial set $\{f_1, \dots, f_s\}$ by repeatedly producing and reducing combinations of the basis elements.

DEFINITION 2. Let $S(f, g) = \frac{x^Y}{\text{LT}(f)} f - \frac{x^Y}{\text{LT}(g)} g$ where x^Y is the least common multiple of the leading monomials of f and g . This is the S -polynomial of f and g , where s stands for subtraction or syzygy.

THEOREM 1 (BUCHBERGER'S CRITERION). Let $G = \{g_1, g_2, \dots, g_k\}$ generate the ideal I . If $\text{reduce}(S(g_i, g_j), G) = 0$ for all pairs g_i, g_j then G is a Gröbner basis of I .

EXAMPLE 1. Fix the monomial order $>$ to be grevlex. For the generating set $F = \{f_1, f_2\}$ in Equation (1), $r = \text{reduce}(S(f_1, f_2), F) = y^3 + x$. By construction, the set $G = \{f_1, f_2, r\}$ generates the same ideal as F and $\text{reduce}(S(f_1, f_2), G) = 0$, so we have eliminated this pair for the purposes of verifying the criterion at the expense of two new pairs. Luckily, in this example $\text{reduce}(S(f_1, r), G) = 0$ and $\text{reduce}(S(f_2, r), G) = 0$. So G is a Gröbner basis for $\langle f_1, f_2 \rangle$ with respect to the grevlex order.

Generalizing this example, Theorem 1 naturally leads to Algorithm 2, which depends on several implementation choices: select in line 6, reduce in line 8, and update in line 10. The algorithm is guaranteed to terminate regardless of these implementation choices, but all three choices can impact the performance. Most of the improvements to Buchberger's algorithm have come from improved heuristics for the choices involved in these three steps.

The simplest implementation of update is

$$\text{update}(P, G, r) = P \cup \{(f, r) : f \in G\},$$

but most implementations use special rules to eliminate some pairs a priori, so as to minimize the number of S -polynomial reductions performed. In fact, much of the research on improving the performance of Buchberger's algorithm has focused on mathematical methods to eliminate as many pairs as possible. e.g. [16, 19].

We use the pair elimination rules of [21] in all results in this paper. In addition, for the choices involved in reduction, line 5 of Algorithm 1, we always choose the smallest $\text{LT}(f_i)$ which divides r .

2.2 Selection Strategies

In this paper we focus on the implementation of select. The selection strategy is critically important for efficiency, as poor pair selection can add many unnecessary elements to the generating set before finding a Gröbner basis. There has been some research on selection strategies [19, 35], but mostly in the context of signature

Algorithm 2 Buchberger's Algorithm

```

1: Input: a set of polynomials  $\{f_1, \dots, f_s\}$ 
2: Output: a Gröbner basis  $G$  of  $I = \langle f_1, \dots, f_s \rangle$ 
3:  $G \leftarrow \{f_1, \dots, f_s\}$ 
4:  $P \leftarrow \{(f_i, f_j) : 1 \leq i < j \leq s\}$ 
5: while  $|P| > 0$  do
6:    $(f, g) \leftarrow \text{select}(P)$ 
7:    $P \leftarrow P \setminus \{(f, g)\}$ 
8:    $r \leftarrow \text{reduce}(S(f, g), G)$ 
9:   if  $r \neq 0$  then
10:     $P \leftarrow \text{update}(P, G, r)$ 
11:     $G \leftarrow G \cup \{r\}$ 
12:   end if
13: end while

```

Gröbner bases, including Faugere's F_5 algorithm. Other than these, most strategies to date depend on relatively simple human-designed heuristics. We use several well-known examples as benchmarks:

First: Among pairs with minimal j , select the one with minimal i . In other words, treat the pair set as a queue.

Degree: Select the pair with smallest total degree of $\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))$. If needed, break ties with First.

Normal: Select the pair with smallest $\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))$ in the monomial order. If needed, break ties with First. In a degree order ($x^a > x^b$ if the total degree of x^a is greater than that of x^b), this is a refinement of Degree selection.

Sugar: Select the pair with smallest sugar degree, which is the degree $\text{lcm}(\text{LT}(f_i), \text{LT}(f_j))$ would have had if we had homogenized all input polynomials at the beginning. If needed, break ties with Normal. [22]

Random: Select an element of the pair set uniformly at random.

Note that all of these strategies only depend on the leading terms of the polynomials in each pair. One can imagine using more terms of these polynomials in pair selection, but most implementations do not do so.

2.3 Complexity

We will characterize the input to Buchberger's algorithm in terms of the number of variables (n), the maximal degree of a generator (d), and the number of generators (s). How large can a Gröbner basis be? One measure of complexity is the maximum degree $\deg_{\max}(GB(I))$ of an element in the *unique* reduced minimal Gröbner basis for I .

When the coefficient field has characteristic 0, there is an upper bound $\deg_{\max}(GB(I)) \leq (2d)^{2^{n+1}}$ which is double exponential in the number of variables [5]. There do exist ideals which exhibit double exponential behavior [7, 27, 33]: there is a sequence of ideals $\{J_n\}$ where J_n is generated by quadratic homogeneous binomials in $22n - 1$ variables such that for any monomial order

$$2^{2^{n-1}-1} \leq \deg_{\max}(GB(J_n))$$

In the grevlex monomial order, the theoretical upper bounds on the complexity of Buchberger's algorithm are much better if the choice of generators is sufficiently generic. To make this precise, for fixed n, d, s , the space of possible inputs, i.e., the space V of coefficients for each of the s generators, is finite dimensional. There is a subset $X \subset V$ of measure zero¹ such that for any point outside X ,

$$\deg_{\max}(GB(I)) \leq (n+1)(d-1) + 1$$

This implies that the size of $GB(I)$ is \leq the number of monomials of degree $\leq (n+1)(d-1) + 1$, which grows like $O(((n+1)d-1)^n)$.

It is expected, but not known, that it is rare for the maximum degree of a Gröbner basis element in the grevlex monomial order to be double exponential in the number of variables. Also, as early as the 1980's, it was realized that for many examples, the grevlex Gröbner basis was often much easier to compute than Gröbner bases for other monomial orders. For these reasons, the grevlex monomial order is a standard choice in Gröbner basis computations.

¹Technically, X is a closed algebraic subset. With coefficients in \mathbb{R} or \mathbb{C} , this is measure zero in the usual sense.

We use grevlex throughout this paper for all of our computational experiments.

3 THE REINFORCEMENT LEARNING PROBLEM

We model Buchberger's algorithm as a Markov Decision Process (MDP), in which an agent interacts with an environment to perform pair selection in line 6 of Algorithm 2.

Each pass through the while loop in line 5 of Algorithm 2 is a time step, in which the agent takes an action and receives a reward. At time step t , the agent's state $s_t = (G_t, P_t)$ consists of the current generating set G_t and the current pair set P_t . The agent must select a pair from the current set, so the set of allowable actions is $\mathcal{A}_t = P_t$. Once the agent selects an action $a_t \in \mathcal{A}_t$, the environment updates by removing the pair from the pair set, reducing the corresponding S -polynomial, and updating the generator and pair set if necessary.

After the environment updates, the agent receives a reward r_t which is -1 times the number of polynomial additions performed in the reduction of pair a_t , including the subtraction that produced the S -polynomial. This is a proxy for computational cost that is implementation independent, and thus useful for benchmarking against other selection heuristics. For simplicity, this proxy does not penalize monomial division tests or pair elimination.

Each trajectory $\tau = (s_0, a_0, r_1, s_1, \dots, r_T)$ is a sequence of steps in Buchberger's algorithm, and ends when the pair set is empty and the algorithm has terminated with a Gröbner basis. The agent's objective is to maximize the expected return $\mathbb{E}[\sum_{t=1}^T \gamma^{t-1} r_t]$, where $0 \leq \gamma \leq 1$ is a discount factor. With $\gamma = 1$, this is equivalent to minimizing the expected number of polynomial additions taken to produce a Gröbner basis.

This problem poses several interesting challenges from a machine learning perspective:

- (1) The size of action set changes with each time step and can be very large.
- (2) There is a high variance in difficulty of problems of the same size.
- (3) The state changes shape with each time step, and the state space is unbounded in several dimensions: number of variables, degree and size of generators, number of generators, and size of coefficients.

3.1 The Domain: Random Binomial Ideals

Formulating Buchberger's algorithm as a RL problem forces one to consider the question of what is a random polynomial. This is a significant departure from the typical way of thinking about the Gröbner basis problem.

We have seen that Buchberger's algorithm performs much better than its worst cast on generic choices of input. On the other hand, many of the ideals that arise in practice are far from generic in this sense. As n, d , and s grow, Gröbner basis computations tend to blow up in several ways simultaneously: (i) the number of polynomials in the generating set grows, (ii) the number of terms in each polynomial grows, and (iii) the size of the coefficients grows (e.g., rational numbers with very large denominators).

There is a standard way to handle (iii). Many questions over a field of characteristic 0 can be converted to an analogous question

over a finite field $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ which gives the same answer with high likelihood, and coefficients in a finite field have bounded size. For all our experiments we will fix the coefficient field as $\mathbb{Z}/32003\mathbb{Z}$.

In order to address (ii), we restrict our training to systems of polynomials with at most two terms. These are known as *binomials*. We will also assume neither term is a constant. If the input to Buchberger’s algorithm is a set of binomials of this form, then all of the new generators added to the set will also have this form. This side-steps the thorny issue of how to represent a polynomial of arbitrary size to a neural network.

Restricting our focus to binomial ideals has several other benefits: We will show that using binomials typically avoids the known “easy” case when the dimension of the ideal, which is defined to be the dimension of the set of solutions of the corresponding system of equations, is zero. We have also seen that some of the worst known examples with double exponential behavior are binomial systems. Finally, binomials capture the qualitative fact that many of the polynomials appearing in applications are sparse. In fact, several applications of Buchberger’s algorithm, such as integer programming, specifically call for binomial ideals [10, 11].

We also remark that a model trained on binomials might be useful in other domains as well. Just as most standard selection strategies only consider the leading monomials of each pair, one could use a model trained on binomials to select pairs based on their leading *binomials*.

We performed experiments with two probability distributions on the set of binomials of degree $\leq d$ in n generators. The first, uniform, selects the degree of each monomial uniformly at random, then selects each uniformly at random among monomials of the chosen degree. The second, weighted, selects both monomials uniformly at random from the set of monomials of degree $\leq d$. The main difference between these two distributions is that uniform tends to produce more binomials of low total degree. Both distributions assign non-zero coefficients uniformly at random.

For the remainder of the paper, we will use the format “ n - d - s (weighted/uniform)” to specify our distribution on s -tuples of binomials of degree $\leq d$ in n variables.

3.2 Statistics

We will briefly discuss the statistical properties of the problem in the domain of binomial ideals to highlight its features and challenges.

Difficulty increases with n : (Table 1) This is consistent with the double exponential behavior in the worst-case analysis.

Degree and Normal outperform First and Sugar: (Table 1) This pattern is consistent across all distributions in the range tested ($n = 3, d \leq 30, s \leq 20$). The fact that Sugar under-performs in an average-case analysis might reflect the fact that it was chosen because it improves performance on known sequences of challenging benchmark ideals in [22].

Very high variance in difficulty: This is also illustrated in Table 1, especially as the number of variables increases. Figure 1 provides a more detailed view of a single distribution, demonstrating the large variance and long right tail that is typical of Gröbner basis calculations. This poses a particular challenge for the training of reinforcement learning models.

Table 1: Number of polynomial additions for different selection strategies on the same samples of 10000 ideals. Distributions are n -5-10 uniform. Table entries show mean [stddev].

| n | FIRST | DEGREE | NORMAL | SUGAR |
|-----|-------------|-------------|-------------|-------------|
| 2 | 36.4 [7.24] | 32.3 [5.71] | 32.0 [5.49] | 32.4 [6.15] |
| 3 | 52.8 [17.9] | 42.2 [13.2] | 42.4 [13.1] | 44.2 [15.1] |
| 4 | 86.3 [40.9] | 63.8 [28.5] | 66.5 [29.8] | 70.0 [32.9] |
| 5 | 151. [85.7] | 109. [58.8] | 117. [64.4] | 120. [68.7] |
| 6 | 280. [174.] | 198. [118.] | 221. [132.] | 223. [143.] |
| 7 | 527. [359.] | 379. [240.] | 435. [277.] | 430. [296.] |
| 8 | 1030 [759.] | 760. [510.] | 887. [588.] | 863. [639.] |

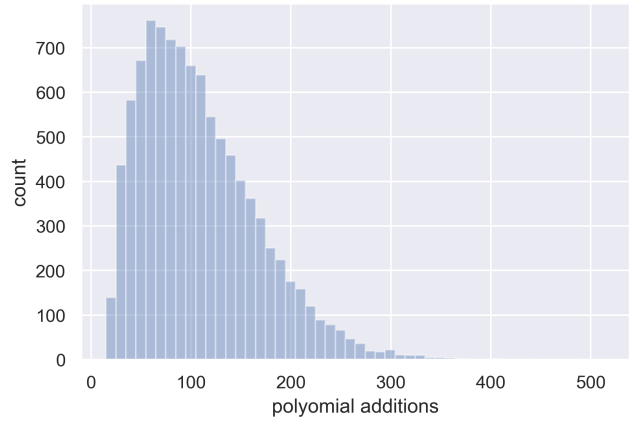


Figure 1: Histogram of polynomial additions in 5-5-10 uniform following Degree selection over 10000 samples.

Dependence on s is subtle: For $n = 3$, there is a spike in difficulty at four generators, followed by a drop/leveling off, and a slow increase after that (Figures 2 and 3). The spike is even more pronounced in $n > 3$ variables, where it occurs instead at $n + 1$ generators. The leveling off is consistent with the hypothesis that the appearance of low-degree generators makes the problem easier. The minimum number of polynomial additions increases as the number of generators increases, but this could be balanced by an increased likelihood of low-degree generators. The fact that uniform is easier than weighted across values of d and s also supports this hypothesis.

Difficulty increases relatively slowly with d : The growth appears to be either linear or slightly sub-linear in d in the range tested (Figures 2 and 3).

Zero dimensional ideals are rare: (Table 2) For $n = 3, d = 20$, the hardest distribution is $s = 4$, in which case .05% of the ideals were zero dimensional. This increased to 21.2% using the uniform distribution and increasing to $s = 10$, which is still relatively rare. This also supports the hypothesis that the appearance of a generator of low degree makes the problem easier.

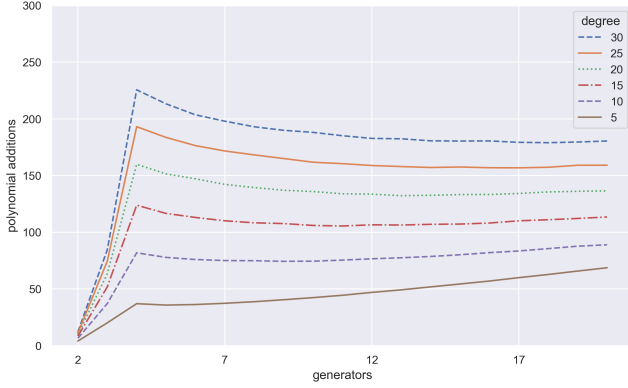


Figure 2: Average number of polynomial additions following Degree selection in $n = 3$ uniform. Each degree and generator point is the mean over 10000 samples.

Table 2: Dimension of the binomial ideals (i.e., the dimension of the solution set of the corresponding system of equations), in a sample of 10000 ($n = 3$, $d = 20$).

| dim | UNIFORM | | WEIGHTED | |
|-----|----------|---------|----------|---------|
| | $s = 10$ | $s = 4$ | $s = 10$ | $s = 4$ |
| 0 | 2121 | 178 | 58 | 5 |
| 1 | 7657 | 6231 | 8146 | 2932 |
| 2 | 223 | 3592 | 1797 | 7064 |

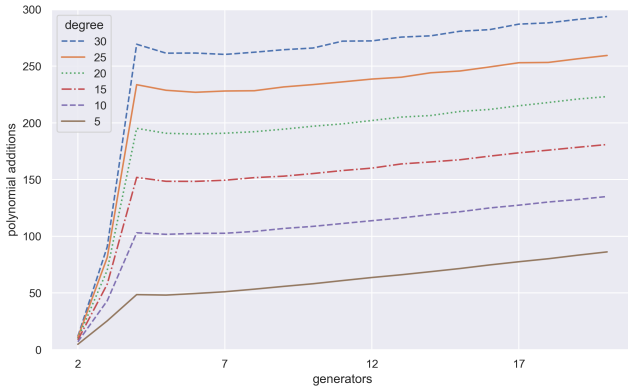


Figure 3: Average number of polynomial additions following Degree selection in 3-weighted. Each degree and generator point is the mean over 10000 samples.

4 EXPERIMENTAL SETUP

We train a neural network model to perform pair selection in Buchberger's algorithm.

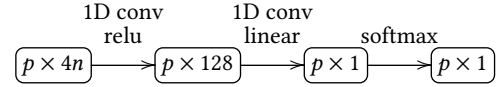
4.1 Network Structure

We represent a state $S_t = (G_t, P_t)$ as a matrix whose rows are obtained by concatenating the exponent vector of each pair. For n variables and p pairs, this results in a matrix of size $p \times 4n$. The environment is now partially observed, as the observation does not include the coefficients.

EXAMPLE 2. Let $n = 3$, and consider the state given by $G = \{xy^6 + 9y^2z^4, z^4 + 13z, xy^3 + 91xy^2\}$, where the terms of each binomial are shown in grevlex order, and $P = \{(1, 2), (1, 3), (2, 3)\}$. Mapping each pair to a row yields

$$\begin{bmatrix} 1 & 6 & 0 & 0 & 2 & 4 & 0 & 0 & 4 & 0 & 0 & 1 \\ 1 & 6 & 0 & 0 & 2 & 4 & 1 & 3 & 0 & 1 & 2 & 0 \\ 0 & 0 & 4 & 0 & 0 & 1 & 1 & 3 & 0 & 1 & 2 & 0 \end{bmatrix}$$

Our agent uses a policy network that maps each row to a single preference score using a series of dense layers. We implement these layers as 1D convolutions with 1×1 kernel in order to compute the preference score for all pairs simultaneously. The agent's policy, which is a probability distribution on the current pair set, is the softmax of these preference scores. In preliminary experiments, network depth did not appear to significantly affect performance, so we settled on the following architecture:



Due to its simplicity, it would be easy to deploy this model in a production implementation of Buchberger's algorithm. The preference scores produced by the network could be used as sort keys for the pair set, which means each pair would only need to be processed once, thereby minimizing the overhead of using the network for pair selection.

4.2 Value Functions

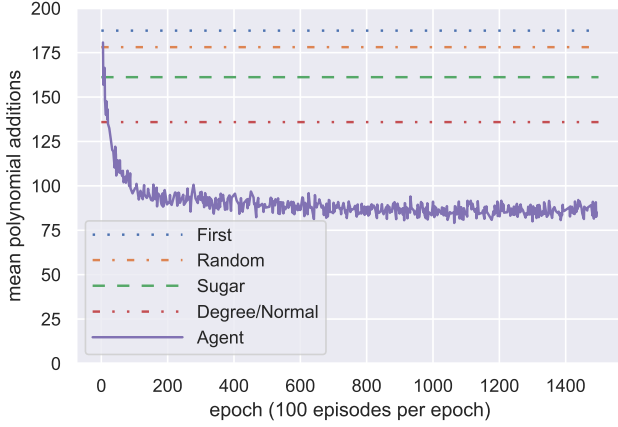
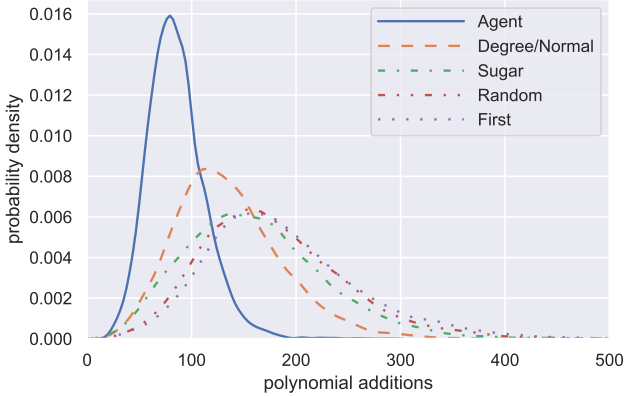
A general challenge for policy gradient algorithms is the large variance in the estimate of expected rewards. This is exacerbated in our context by the large variance in difficulty of computing a Gröbner basis of different ideals from the same distribution. We address this using Generalized Advantage Estimation (GAE) [37], which uses a value function to produce a lower-variance estimator of expected returns while limiting the bias introduced. Our value function $V(G, P)$ is the number of polynomial additions required to complete a full run of Buchberger's algorithm starting with the state (G, P) , using the Degree strategy. This is computationally expensive but significantly improves performance.

4.3 Training Algorithm

Our agents are trained with proximal policy optimization (PPO) [38] using a custom implementation inspired by [2]. In each epoch we first sample 100 episodes following the current policy. Next, GAE with $\lambda = 0.97$ and $\gamma = 0.99$ is used to compute advantages, which are normalized over the epoch. Finally, we perform at most 80 gradient updates on the clipped surrogate PPO objective with $\epsilon = 0.2$ using Adam optimization with learning rate 0.0001. Early-stopping is performed when the sampled KL-divergence from the previous policy exceeds 0.01.

Table 3: Agent performance versus benchmark strategies in 3 variables and degree 20. Each line is a unique agent trained on the given distribution. Performance is mean[stddev] on 10000 randomly sampled ideals from that distribution.

| s | DISTRIBUTION | FIRST | DEGREE | NORMAL | SUGAR | RANDOM | AGENT | IMPROVEMENT |
|-----|--------------|------------|------------|------------|------------|------------|------------|-------------|
| 10 | UNIFORM | 187.[73.1] | 136.[50.9] | 136.[51.2] | 161.[66.9] | 178.[68.3] | 85.6[27.3] | 37% [46%] |
| 4 | UNIFORM | 210.[101.] | 160.[64.5] | 160.[66.6] | 185.[87.2] | 203.[97.8] | 101.[44.9] | 37% [30%] |
| 10 | WEIGHTED | 352.[117.] | 197.[55.7] | 198.[57.1] | 264.[88.5] | 318.[103.] | 141.[42.8] | 28% [23%] |
| 4 | WEIGHTED | 317.[130.] | 195.[70.0] | 194.[70.0] | 265.[107.] | 303.[122.] | 151.[56.4] | 22% [19%] |

**Figure 4: Training an agent on the 3-20-10 uniform distribution.****Figure 5: Estimated distribution of polynomial additions per ideal in the 3-20-10 uniform distribution for the fully trained agent from Figure 4, compared to benchmark strategies. (10000 samples, computed using kernel density estimation)**

5 EXPERIMENTAL RESULTS

Table 3 shows the final performance of agents which have been trained on several distributions with $n = 3$, $d = 20$. All agents use

22% to 37% fewer polynomial subtractions on average than the best benchmark strategies, and reduce the standard deviation in the number of polynomial additions by 19% to 46%. The improvement on weighted distributions, which tend to produce ideals of higher average difficulty, was not as large as the improvement on uniform distributions. Figure 5 gives a more detailed view of the distribution of polynomial additions per ideal performed by the trained agent. Figure 4 shows the rapid convergence during training.

5.1 Interpretation

We have identified several components of the agent’s strategy: (a) the agent is mimicking Degree, (b) the agent prefers pairs whose S -polynomials are monomials, (c) the agent prefers pairs whose S -polynomials are low degree.

On 10000 sample runs of Buchberger’s algorithm using a trained agent on 3-20-10 uniform, the average probability that the agent selected a pair which could be chosen by Degree was 43.5%. If there was a pair in the list whose S -polynomial was a monomial, the agent picked such a pair 31.7% of the time. The probability that the agent selected a pair whose S -polynomial had minimal degree (among S -polynomials), was 48.3%.

It is notable that (b) and (c) are not standard selection heuristics. When we hard-coded the strategy of selecting a pair with minimal degree S -polynomial, the average number of additions (3-20-10 uniform, 10000 samples) was 120.3, a 12% improvement over the Degree strategy. On the other hand, for the strategy which follows Degree but will first select any S -polynomial which is monomial, the average number of additions was 134.2, a 1.2% improvement over Degree. While neither hard-coded strategy achieves the 37% improvement of the agent over Degree, it is notable that these insights from the model led to understandable strategies that beat our benchmark strategies in this domain.

5.2 Variants of the Model

We found that the model performance decreased when we made any of the following modifications: only allowed the network to see the lead monomials of each pair; removed the value function; or substituted the value function with a naive “pairs left” value function which assigned $V(G, P) = |P|$. See Table 4. However, all of these trained models still outperform the best benchmark strategy, which is Degree.

Another variant of the model we considered was to reward each S -polynomial reduction with -1 , rather than the number of polynomial additions. This would be useful in a context where the multivariate division algorithm was being optimized separately. We used the same methods as above, with the pairs left value function.

A trained agent performed an average of 37.9 S -pair reductions [stddev=14.2] over 10000 samples from 3-20-10 uniform, which is an improvement of 28% over the best benchmark strategy, Sugar, which performed an average of 52.9 S -pair reductions [stddev=21.1].

Table 4: Performance of variants of the model. Entries show mean [stddev] of polynomial additions and performance drop relative to the original model on samples of 10000 ideals from 3-20-10 uniform distribution. Original model is 85.6 [27.3].

| AGENT | ADDITIONS | DROP |
|--------------------------|--------------|-------|
| PAIRSLEFT VALUE FUNCTION | 95.2 [32.7] | 11.2% |
| NO VALUE FUNCTION | 103.2 [35.9] | 20.6% |
| LEAD MONOMIAL ONLY | 90.0[29.4] | 5.4% |

Table 5: Agent performance outside of training distribution. Performance is mean[stddev] on a sample of 10000 random ideals.

| 3-20-10 distribution | | | |
|----------------------|------------|------------|--|
| TRAIN \ TEST | UNIFORM | WEIGHTED | |
| | | | |
| UNIFORM | 85.6[27.3] | 140.[45.7] | |
| WEIGHTED | 89.3[29.0] | 141.[42.8] | |

| 3-20-4 DISTRIBUTION | | | |
|---------------------|------------|------------|--|
| TRAIN \ TEST | UNIFORM | WEIGHTED | |
| | | | |
| UNIFORM | 101.[44.9] | 158.[67.9] | |
| WEIGHTED | 107.[42.6] | 151.[56.4] | |

5.3 Generalization

A major question in machine learning is the ability of a model to generalize outside of its training distribution. The Buchberger problem appears to be well-suited to generalization. Figure 6 shows that a model trained on 3-20-10 uniform has similar performance at nearby values of d and s , as compared to the performance of the best benchmark strategy. Table 5 shows that there is also reasonable generalization between weighted and uniform.

6 CONCLUSION

We have introduced the Buchberger environment, a challenging reinforcement learning problem with important ramifications for the performance of computer algebra software. We have identified binomial ideals as an interesting domain for this problem that is tractable, maintains many of the problem's interesting features, and can serve as a benchmark for future research. We have shown that standard reinforcement learning algorithms with simple models can develop strategies that improve over state-of-the-art in this domain. This suggests that modern developments in machine learning could be used to improve the performance of critical algorithms in symbolic computation.

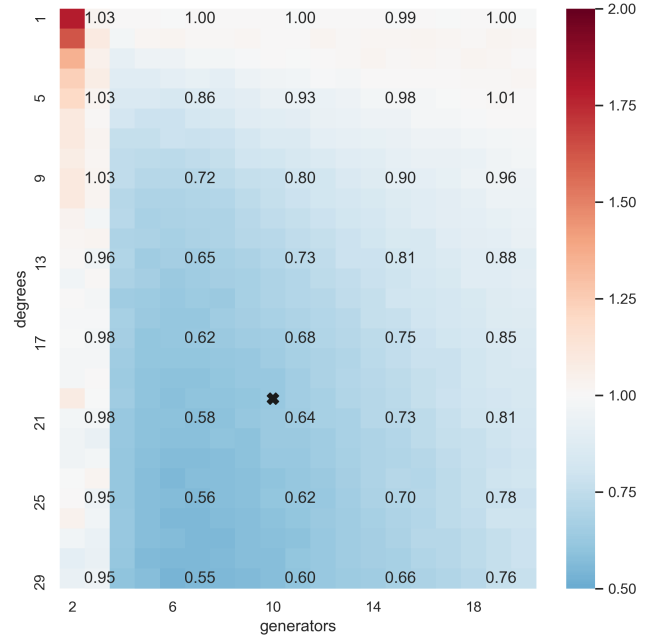


Figure 6: Testing a single agent on 3- d - s uniform distribution for $1 \leq d \leq 30$ and $2 \leq s \leq 20$. Numbers are the ratio of average polynomial additions by the agent over that of the best benchmark strategy. The agent was trained with 3-20-10 uniform, indicated with an "X." The agent was tested on 1000 random ideals in each distribution, and the strategies were tested on 10000.

REFERENCES

- [1] ABLAMOWICZ, R. *Some Applications of Gröbner Bases in Robotics and Engineering*, in Geometric Algebra Computing: in Engineering and Computer Science. Springer London, London, 2010, pp. 495–517.
- [2] ACHIAM, J. Spinning Up in Deep Reinforcement Learning.
- [3] ALVAREZ, A. M., LOUVEAUX, Q., AND WEHENKEL, L. A machine learning-based approximation of strong branching. *INFORMS Journal on Computing* 29, 1 (2017), 185–195.
- [4] ARKUN, Y. Detection of biological switches using the method of Gröbner bases. *BMC Bioinformatic* 20 (2019).
- [5] BAYER, D., AND MUMFORD, D. What can be computed in algebraic geometry? In *Computational algebraic geometry and commutative algebra (Cortona, 1991)*, Sympos. Math., XXXIV. Cambridge Univ. Press, Cambridge, 1993, pp. 1–48.
- [6] BAYER, D., AND STILLMAN, M. A criterion for detecting m -regularity. *Invent. Math.* 87, 1 (1987), 1–11.
- [7] BAYER, D., AND STILLMAN, M. On the complexity of computing syzygies. *J. Symbolic Comput.* 6, 2-3 (1988), 135–147.
- [8] BENGIO, Y., LODI, A., AND PROUVOST, A. Machine learning for combinatorial optimization: a methodological tour d’horizon. *CoRR abs/1811.06128* (2018).
- [9] CoCoA. A system for doing Computations in Commutative Algebra, Abbott, J., Bigatti, A. M., and Robbiano, L., 2019. <http://cocoa.dima.unige.it>.
- [10] CONTI, P., AND TRAVERSO, C. Buchberger algorithm and integer programming. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes* (1991), Springer, pp. 130–139.
- [11] COX, D. A., LITTLE, J., AND O’SHEA, D. *Using algebraic geometry*, second ed., vol. 185 of *Graduate Texts in Mathematics*. Springer, New York, 2005.
- [12] COX, D. A., LITTLE, J., AND O’SHEA, D. *Ideals, varieties, and algorithms*, fourth ed. Undergraduate Texts in Mathematics. Springer, Cham, 2015.
- [13] DIACONIS, P., AND STURMFELS, B. Algebraic algorithms for sampling from conditional distributions. *Ann. Statist.* 26, 1 (1998), 363–397.
- [14] DUBÉ, T. W. The structure of polynomial ideals and Gröbner bases. *SIAM J. Comput.* 19, 4 (1990), 750–775.

- [15] DUFF, T., KOHN, K., LEYKIN, A., AND PAJDLA, T. PLMP - point-line minimal problems in complete multi-view visibility. *CoRR abs/1903.10008* (2019).
- [16] EDER, C., AND FAUGÈRE, J.-C. A survey on signature-based algorithms for computing Gröbner bases. *J. Symbolic Comput.* 80, part 3 (2017), 719–784.
- [17] EISENBUD, D. *Commutative algebra*, vol. 150 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1995. With a view toward algebraic geometry.
- [18] FAUGÈRE, J.-C. A new efficient algorithm for computing Gröbner bases (F_4). *J. Pure Appl. Algebra* 139, 1-3 (1999), 61–88.
- [19] FAUGÈRE, J.-C. A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation* (2002), ACM, New York, pp. 75–83.
- [20] FAUGÈRE, J.-C., SAFEY EL DIN, M., AND SPAENLEHAUER, P.-J. Computing loci of rank defects of linear matrices using Gröbner bases and applications to cryptology. In *ISSAC 2010—Proceedings of the 2010 International Symposium on Symbolic and Algebraic Computation* (2010), ACM, New York, pp. 257–264.
- [21] GEBAUER, R., AND MÖLLER, H. M. On an installation of Buchberger’s algorithm. *J. Symbolic Comput.* 6, 2-3 (1988), 275–286.
- [22] GIOVINI, A., MORA, T., NIESI, G., ROBBIANO, L., AND TRAVERSO, C. “one sugar cube, please” or selection strategies in the Buchberger algorithm. In *Proceedings of the 1991 International Symposium on Symbolic and Algebraic Computation* (1991), ISSAC ’91, ACM, New York, pp. 49–54.
- [23] GRAY, J. A simple introduction to Gröbner basis methods in string phenomenology. *Adv. High Energy Phys.* 2011 (2011), 12.
- [24] HILBERT, D. Ueber die vollen Invariantensysteme. *Math. Ann.* 42, 3 (1893), 313–373.
- [25] HUANG, Z., ENGLAND, M., WILSON, D. J., BRIDGE, J., DAVENPORT, J. H., AND PAULSON, L. C. Using machine learning to improve cylindrical algebraic decomposition. *Mathematics in Computer Science* 13, 4 (2019), 461–488.
- [26] KHALIL, E., LE BODIC, P., SONG, L., NEMHAUSER, G., AND DILKINA, B. Learning to branch in mixed integer programming. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (2016), AAAI Press, pp. 724–731.
- [27] KOH, J. Ideals generated by quadrics exhibiting double exponential degrees. *J. Algebra* 200, 1 (1998), 225–245.
- [28] LIN, Z., XU, L., AND WU, Q. Applications of Gröbner bases to signal and image processing: a survey. *Linear Algebra Appl.* 391 (2004), 169–202.
- [29] MACAULAY2. A software system for research in algebraic geometry, Grayson, D. and Stillman, M., 2019. <http://www.math.uiuc.edu/Macaulay2/>.
- [30] MAGMA. Algebra system, bosma, w. and cannon, j. and playoust, c., 2019. <http://magma.maths.usyd.edu.au>.
- [31] MAPLE. Maplesoft (2019).
- [32] MATHEMATICA. Wolfram, S., 2019.
- [33] MAYR, E. W., AND MEYER, A. R. The complexity of the word problems for commutative semigroups and polynomial ideals. *Adv. in Math.* 46, 3 (1982), 305–329.
- [34] MORA, T. *Solving polynomial equation systems. II*, vol. 99 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 2005.
- [35] ROUNE, B. H., AND STILLMAN, M. Practical Gröbner basis computation. In *ISSAC 2012—Proceedings of the 37th International Symposium on Symbolic and Algebraic Computation* (2012), ACM, New York, pp. 203–210.
- [36] SAGEMATH. *The Sage Mathematics Software System*, 2019.
- [37] SCHULMAN, J., MORITZ, P., LEVINE, S., JORDAN, M. I., AND ABBEEL, P. High-dimensional continuous control using generalized advantage estimation. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings* (2016).
- [38] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal policy optimization algorithms. *CoRR abs/1707.06347* (2017).
- [39] SINGULAR. A computer algebra system for polynomial computations, Decker, W., Greuel, G.M., Pfister, G., and Schönemann, H., 2019. <http://www.singular.uni-kl.de>.
- [40] SULLIVANT, S. *Algebraic statistics*, vol. 194 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2018.

A EXPERIMENTAL METHODS AND HYPERPARAMETER TUNING

The code used to generate all statistics and results for this paper is available at [\[link available after publication\]](#). Statistics for First, Degree, Normal, Sugar, and Random were generated using Macaulay2 1.14 on Ubuntu 18.04 while agent training was performed on c5n.xlarge instances from Amazon Web Services using the Ubuntu 18.04 Deep Learning AMI.

There were four primary training settings corresponding to the four distributions in Table 3 from Section 5. In each setting we performed three complete runs using the parameters from Table 6. Model weights were saved every 100 epochs, and a single model was selected from the available runs and save points in each setting based on best smoothed training performance.

Table 6: Hyperparameters for primary evaluation runs.

| HYPERPARAMETER | VALUE |
|------------------------------|--------------|
| γ FOR GAE | 0.99 |
| λ FOR GAE | 0.97 |
| ϵ FOR PPO | 0.2 |
| OPTIMIZER | ADAM |
| LEARNING RATE | 0.0001 |
| MAX POLICY UPDATES PER EPOCH | 80 |
| POLICY KL-DIVERGENCE LIMIT | 0.01 |
| HIDDEN LAYERS | [128] |
| VALUE FUNCTION | DEGREE AGENT |
| EPOCHS | 2500 |
| EPISODES PER EPOCH | 100 |
| MAX EPISODE LENGTH | 500 |

Trained models were then evaluated on new sets of ideals to produce Table 3 in Section 5, Table 5 in Section 5.3, and Figure 6 in Section 5.3. The three models from Table 4 in Section 5.2 were selected and evaluated in the same way, but were trained with their respective modifications.

In addition to the main evaluation runs for this paper, we performed a brief hyperparameter search in two stages. Both stages were trained in the 3-20-10-uniform distribution as it gives the fastest training runs.

In the first stage we varied the parameters γ in $\{1.0, 0.99\}$, λ in $\{1.0, 0.97\}$, learning rate in $\{10^{-3}, 10^{-4}, 10^{-5}\}$, and network as a single hidden layer of 128 units or two hidden layers of 64 units. Three runs were performed on each set of parameters, for a total of 72 runs. The pairs left value function was used in this search instead of the degree agent, as it leads to significantly faster training. Learning rates of 10^{-4} showed best performance, though rates of 10^{-5} were still improving at the end of the runs. Changes in γ , λ , and network did not consistently change performance.

In the second stage we varied just the network shape. Single hidden layer networks were tried with 4, 8, ..., 256 units and two hidden layer networks were tried with 4, 8, ..., 256 hidden units in each layer. One run was performed on each network, for a total of 14 runs. Results showed significant improvement in using at least 32 hidden units and no major differences between one and two hidden layers. Small models were also surprisingly effective, with

the model with a single hidden layer of 4 units achieving mean performance of around 100 polynomial additions during training, compared to Degree selection at 136 and our full model at 85.6.

B COMPLEXITY ANALYSIS OF BUCHBERGER'S ALGORITHM

In this section, we consider upper and lower bounds for the complexity of computing a Gröbner basis of an ideal in a polynomial ring, and also describe what happens in the "generic" (random) case, giving more detail than in the paper proper. We first consider the maximum degree of a Gröbner basis element, then describe how that gives bounds for the size of a minimal reduced Gröbner basis.

Let $I = \langle f_1, \dots, f_s \rangle \subset S = k[x_1, \dots, x_n]$, be an ideal, with each polynomial f_i of degree $\leq d$. If $>$ is a monomial order, and $G = GB_{>}(I) = \{g_1, \dots, g_r\}$ is a Gröbner basis of I , G is called a *minimal and reduced* Gröbner basis if the lead coefficient of each g_i is one, and no monomial of g_i is divisible by $LT(g_j)$, for $i \neq j$. Given any Gröbner basis, it is easy to modify G to obtain a minimal and reduced Gröbner basis of I . Given the monomial order, each ideal I has precisely one minimal reduced Gröbner basis with respect to this order. We define $\deg_{\max}(GB_{>}(I)) := \max(\deg g_1, \dots, \deg g_r)$, where $G = GB_{>}(I) = \{g_1, \dots, g_r\}$ is the unique minimal reduced Gröbner basis of I for the order $>$.

Upper bounds

We have the following upper bound for $\deg_{\max}(GB_{>}(I))$.

THEOREM 2 ([14]). *Given I as above, then*

$$\deg_{\max}(GB_{>}(I)) \leq 2\left(\frac{d^2}{2} + d\right)^{2^{n-1}}$$

If I is homogeneous (i.e. each polynomial f_i is homogeneous), we may replace the $n - 1$ by $n - 2$ in this bound.

Such a bound is called double exponential (in the number of variables). This result seems to give an incredibly bad bound, but it is unfortunately fairly tight, which we will discuss next.

Lower bounds

All known double exponential examples (e.g. [7], [27], [34]) are based essentially on the seminal and important construction of [33]. Each is a sequence of ideals J_n where the n -th ideal J_n is in roughly $10n$ or $20n$ variables, generated in degrees $\leq d$, where each f_i is a *pure binomial* (i.e. a difference of two monomials). The following version of [27] is a sequence of ideals generated by *quadratic* pure binomials.

THEOREM 3 ([27]). *For each $n \geq 1$, there exists an ideal J_n , generated by quadratic homogeneous pure binomials in $22n - 1$ variables such that for any monomial order $>$,*

$$2^{2^{n-1}-1} \leq \deg_{\max}(GB_{>}(J_n))$$

[27] shows that there is a minimal syzygy in degree $2^{2^{n-1}}$. It is well known (see e.g. [34], section 38.1) that this implies that there must be a minimal Gröbner basis element of degree at least half that, giving the stated bound. Thus there are examples of ideals whose Gröbner basis has maximum degree bounded below by roughly $2^{2^{n/2}}$ (where now n is the number of variables). Some of the other

modifications of the Mayr-Meyer construction have slightly higher lower bounds (e.g. $2^{2^{n/10}}$).

Better bounds

Given these very large lower bounds, one might conclude that Gröbner bases cannot be used in practice. However, in many cases, there exist much lower upper bounds for the size of a grevlex Gröbner basis. The key is to relate these degree bounds to the *regularity* of the ideal.

Given a homogeneous ideal $I = \langle f_1, \dots, f_s \rangle \subset S = k[x_1, \dots, x_n]$, with each polynomial of degree $\leq d$, several notions which often appear in complexity bounds and are also useful in algebraic geometry are:

- the dimension $\dim(I)$ of I .
- the depth $\text{depth}(I)$. This is an integer in the range $0 \leq \text{depth}(I) \leq \dim(I)$. In many commutative algebra texts, this is denoted as $\text{depth}(S/I)$, not $\text{depth}(I)$, but in [34], $\text{depth}(I)$ is the notation. This is the length of a maximal S/I -regular sequence in (x_1, \dots, x_n) (see [?]).
- the (Castelnuovo-Mumford) regularity, $\text{reg}(I)$ of the ideal I , see [17] or [34].

The regularity $\text{reg}(I)$ should be considered as a measure of complexity of the ideal.

Generic change of coordinates. Let's consider a homogeneous, linear, change of coordinates $\phi = \phi_A$, where $A \in k^{n \times n}$ is a square n by n matrix over k , with

$$\phi_A(x_i) = \sum_{j=1}^n A_{ij}x_j.$$

Let $\phi_A(I) := \{f(\phi_A(x_1), \dots, \phi_A(x_n)) \mid f \in I\}$ be the ideal under a change of coordinates. Consider the n^2 -dimensional parameter space V (where a point $A = (A_{ij})$ of V corresponds to a homogeneous linear change of coordinates ϕ_A). It turns out that there is a polynomial F in the polynomial ring (with n^2 variables) $k[A_{ij}]$, such that for all points $A \in V$ such that $F(A) \neq 0$, then $LT_{\text{grevlex}}(\phi_A(I))$ is the same ideal. This monomial ideal is called the *generic initial ideal* of I (in grevlex coordinates), and is denoted by $\text{gin}(I)$. Basically, for a random homogeneous linear change of coordinates, one always gets the same size Gröbner basis, with the same lead monomials.

Define $G(I)$ to be the maximum degree of a minimal generator of $\text{gin}(I)$. This is the maximum degree of an element of the unique minimal and reduced Gröbner basis of the ideal $\phi_A(I)$ under almost all change of coordinates ϕ_A (i.e. those for which $F(A) \neq 0$).

The reason this is important is that we have more control over Gröbner bases in generic coordinates. For instance

THEOREM 4 ([6]). *If the base field is infinite, then*

$$\text{reg}(I) = \text{reg}(\text{gin}(I))$$

If the characteristic of k is zero, then

$$G(I) = \text{reg}(I)$$

If the characteristic of k is positive, then

$$\frac{1}{n} \text{reg}(I) \leq G(I) \leq \text{reg}(I)$$

It is known that $\text{reg}(I) \leq \text{reg}(LT_{>}(I))$, for every monomial order $>$. This result states that in fact in generic coordinates, equality is obtained for the grevlex order. The Gröbner basis, after a random change of coordinates, always has maximum degree at most $\text{reg}(I)$.

In particular, if an ideal I has small regularity, as often happens for ideals coming from algebraic geometric problems, then the corresponding Gröbner basis in grevlex order will have much smaller size than the double exponential upper bounds suggest.

THEOREM 5. *If the homogeneous ideal $I = \langle f_1, \dots, f_s \rangle \subset S$ has $\dim(I) = \text{depth}(I)$ (this includes the case when $\dim(I) = 0$, then*

$$\text{reg}(I) \leq (d-1) \min(s, n - \dim(I)) + 1$$

This follows from two basic facts about regularity: First, if $\dim I = 0$, then the regularity of I is the first degree m such that the degree m polynomials in I consist of all degree m polynomials. Second, if I has depth r and y_1, \dots, y_r is a regular sequence of linear forms mod I , then the regularity of I is the regularity of the ideal $\bar{I} := IS/(y_1, \dots, y_r)$. Since the depth and dimension of I are equal, the ideal \bar{I} is of dimension 0, and contains a complete intersection of polynomials each of degree d . This implies by a Hilbert function argument, or by the Koszul complex, that the regularity of \bar{I} is at most $(d-1)(n-r) + 1$ (see [17] for these kinds of arguments).

This implies that $G(I) \leq (d-1) \min(s, n - \dim(I)) + 1 \leq dn$, a dramatic improvement on the double exponential bounds!

Ideals generated by random, or generic, polynomials

What happens for random homogeneous ideals generated by s polynomials each of degree d ? For fixed n, d, s , the space of possible inputs, i.e., the space V of coefficients for each of the s generators, is finite dimensional. There is a subset $X \subset V$, a closed algebraic set (so having measure zero, if the base field is \mathbb{R} or \mathbb{C}), such that for any point outside X , the corresponding ideal I satisfies $\dim(I) = \text{depth}(I)$, and therefore,

$$G(I) \leq (d-1) \min(s, n) + 1.$$

In characteristic zero, equality holds.

If instead of homogeneous ideals, we consider random inhomogeneous ideals, generated by s polynomials each of degree d . The same method holds: the homogenization of these polynomials puts us into the situation in the previous paragraph. Therefore for such inhomogeneous ideals, whose coefficient point is outside of X , then the ideal J generated by the homogenization of the f_i with respect to a new variable satisfies $\dim(J) = \text{depth}(J)$, and therefore,

$$G(I) = G(J) \leq (d-1) \min(s, n+1) + 1.$$

In characteristic zero, equality holds.

Bounds on the size of the reduced minimal Gröbner basis

In the unique reduced minimal Gröbner basis of an ideal I , there can not be two generators with identical lead monomials. It follows that if all generators in this Gröbner basis have degree $\leq D$, then

there are at most

$$\begin{aligned}\#\{\text{monomials of degree} \leq D\} &= \binom{D+n}{n} \\ &= O\left((n+D)^{\min(n,D)}\right)\end{aligned}$$

generators in the Gröbner basis. If one combines this with the upper bound above on the maximum degree, $D = (d-1)\min(s, n+1) + 1$,

one finds the following upper bound on the size of the minimal reduced Gröbner basis of a generic ideal generated by s polynomials of degree $\leq d$ in n variables:

$$\#\{\text{Gröbner basis generators}\} \leq O\left((n+1)^n d^n\right),$$

where the simplification comes from approximating $\min(s, n+1) \leq n+1$, so that our bound is independent of s , and assuming $d \geq 2$.