ELSEVIER

Theoretical Computer Science 187 (1997) 179–202

Theoretical Computer Science

On the Walk

Beatrice Amrhein, Oliver Gloor, Wolfgang Küchlin*

Wilhelm Schickard Institute for Computer Science, University of Tübingen, Germany

Abstract

The Gröbner Walk is a basis conversion method proposed by Collart, Kalkbrener, and Mall. It converts a given Gröbner basis G of a (possibly positive dimensional) polynomial ideal I to a Gröbner basis G' of I with respect to another term order. The target Gröbner basis is approached in several steps (the Walk), each performing a simpler Gröbner basis computation. We address a host of questions associated with this method: alternative ways of presenting the main algorithm, algorithmic variations and refinements, implementation techniques, promising applications, and its practical performance, including a comparison with the FGLM conversion method. Our results show that the Walk has the potential to become a key tool for computing and manipulating ideal bases and solving systems of equations.

1. Introduction

Given any presentation of a polynomial ideal I by a system of polynomials, and given an admissible term ordering $\ll$, Buchberger's famous algorithm [5, 6, 8] computes a canonical representation for I, the *Gröbner Basis* $G(I, \ll)$. *Basis conversion* methods solve the Gröbner basis computation problem $G(I, \ll)$ for the special case that I is already presented by a Gröbner basis with respect to another term ordering $<$.

The main interest in basis conversion today stems from applications in solving systems of polynomial equations. The form and size of a Gröbner basis, and the time for its computation, depend heavily on the term ordering $\ll$. Unfortunately, the lexicographic term orders as well as similar ones that enable the elimination of variables (and hence can be used for polynomial system solving) are "slow" term orders, i.e., they usually lead to particularly long computations.

Basis conversion methods are a promising development to ease this situation. They allow us to compute a lexicographic Gröbner basis via a total degree basis as a stepping-stone, followed by basis conversion. Thus Buchberger's general algorithm is applied only for a "fast" order, and the "slow" order is approached by a more specialized basis

* E-mail: {amrhein, gloor, kuechlin}@informatik.uni-tuebingen.de; http://www-sr.informatik.uni-tuebingen.de.

PII S0304-3975(97)00064-9

conversion. Empirical data show that both parts usually take about the same time, so that speedups of several orders of magnitude can be reaped (cf. Section 5).

A few different basis conversion methods have been suggested by now, among them the so-called FGLM method [12], the Hilbert function approach [14], and the Gröbner Walk by Collart, Kalkbrener, and Mall [9]. The idea of basis conversion can be traced back at least to 1969, when Buchberger [6] sketched a method, similar in spirit to FGLM, for the special case of "constructing the roots of the polynomial ideal" in the zero-dimensional case (cf. Section 6). For further historical pointers and references the reader is referred to [9].

The Gröbner Walk conversion method is particularly interesting because it is inherently independent of the dimension of the ideal. The algorithm takes as input two term orders $<, \ll$, and the (reduced) Gröbner basis $G(I, <)$. It constructs a finite number of term orders $<= \prec_0, \ldots, \prec_m = \ll$ and bases $G_1, \ldots, G_m$, such that G_k is a Gröbner basis of I with respect to $\prec_k$. As G_{k+1} lies in the *neighborhood* of G_k, i.e., the corresponding cones of the Gröbner Fan [21] of I are adjacent, G_{k+1} is computed from G_k with relative ease (cf. Section 2).

The purpose of this paper is to present a first application report of the Gröbner Walk, in order to gauge its potential impact. We report empirical results from system solving, including a comparison with Buchberger's algorithm and a comparison with the FGLM basis conversion method, and timings from the implicitization of Bézier surfaces. Meaningful results on non-trivial applications need a high quality implementation, which in turn must rest on solid theoretical foundations. In particular, it is necessary to cast the abstract algorithm into a concrete form which is practically efficient, and it is necessary to develop know-how about the time consumption of, and implementation techniques for, its constituent parts. We therefore present an implementor's view of the algorithm, and we discuss practically important algorithmic variations and refinements as well as implementation techniques. In practice, these combined efforts have yielded up to 3 orders of magnitude speedup over our first naive implementation.

Our PARSAC-S GRÖBNER WALK is implemented in C within the purely sequential PARSAC-S subset of the PARSAC system framework. PARSAC [19] is a parallel Computer Algebra library which has its origins in the SACLIB package [4], but the SACLIB code is now being phased out. PARSAC-S is a sequential subset, whose code has however the potential to be executed and parallelized within the full PARSAC system. For this paper, all code was executed purely sequentially (no threads of control).

This paper summarizes and extends two earlier papers [2, 3], reflecting important milestones in our installation. Our first implementation [2] was written using SACLIB [4] and some functions of the GRÖBNER package [25]. It already confirmed the key results obtained by the first experimental implementation [9] on top of MATHEMATICA. While for small examples the walk sometimes presented a little overhead, for larger examples a lexicographic Gröbner basis could often be computed one to two orders of magnitude faster with this implementation via a total degree basis followed by a walk.

In [3], we introduced improvements such as *path perturbation*, *interreduction*, *integral weight vectors*, and *special initial Gröbner basis computation*. As a consequence,

we achieved another order of magnitude speedup, and we obtained important insight into the interference of these techniques. We also compared the Walk to FGLM and found it to be generally faster, especially on larger examples.

In this paper, we introduce further algorithmic improvements, and we have phased out the GRÖBNER code in our implementation. Our timings are still taken with the proven SACLIB integer arithmetic, but we expect well over a factor of two speedup from moving to Gnu MP in the immediate future. Still, our empirical results already contain many examples where a lexicographic Gröbner basis could be computed in a few seconds by walking, but failed to terminate within an hour conventionally (cf. Section 5).

Besides the original prototype implementation [9] we are aware of only one other implementation of the Walk to date [23], within the MAGMA system. The empirical results reported in [23] are in general accordance with ours.

The remainder of the paper is organized as follows. In Section 2 we give an alternative presentation of the Gröbner Walk algorithm from an implementor's perspective. In Section 3 we introduce a number of related practically important algorithmic variations. They address the problem of finding the computationally easiest path for the Walk by *path perturbation*. Section 4 presents significant implementation techniques such as *integral weight vectors* and *special initial Gröbner basis computation*. Section 5 presents a table of timings relevant for systems solving, including a comparison with conventional lexicographic Gröbner basis computation. Section 6 gives a short comparison with the FGLM method, including a table of timings. Section 7 outlines the important applications of *implicitization* and *inverse kinematics* mentioned in [11]; for the implicitization problem we present an algorithmic short-cut and first empirical timings. Section 8 describes sources of parallelism in the algorithm. We finish with a conclusion in Section 9.

2. The algorithm

We now present the essential part of Gröbner Walk theory [9] from an algorithmic point of view.

2.1. Weight vectors and orderings

Throughout this paper let $R = K[x_1,\ldots,x_n]$ be a polynomial ring over an arbitrary field K, and let I be an ideal. The ideal generated by a set of polynomials $G \subseteq R$ is denoted by $\langle G\rangle$. For an admissible term ordering $\prec$, $in_\prec(f)$ is the head monomial of a polynomial f, and

$$in_\prec(G) := \{in_\prec(g) \mid g \in G\}.$$

By $G(I,\prec)$ we denote a Gröbner basis of the ideal I with respect to $\prec$. A rational weight vector ω is an element of

$$\{(v_1,\ldots,v_n) \in \mathbb{Q}^n \mid v_i \geqslant 0 \text{ for } i = 1,\ldots,n\}.$$

For a monomial $t = cx_1^{e_1} \cdots x_n^{e_n}$ we define its *ω-degree* by

$$deg_\omega(t) := \sum_{i=1}^{n} e_i v_i.$$

A polynomial is *ω-homogeneous* if all its monomials have the same ω-degree. Every polynomial f can be written as $f = f_1 + \cdots + f_r$ with f_i ω-homogeneous and the f_i sorted by descending ω-degrees. f_1 consists of all monomials with maximum ω-degree and is called the initial form of f, denoted by $in_\omega(f)$. $in_\omega(G)$ is the set $\{in_\omega(g) \mid g \in G\}$. A weight vector ω is *compatible* with a term ordering $\prec$ on G, if for each polynomial $g = m_1 + \cdots + m_s \in G$ ordered in descending order with respect to $\prec$, $deg_\omega(m_1) \geqslant deg_\omega(m_i)$ holds for all $1 < i \leqslant s$. We say that the initial form of g *degenerates* with respect to ω if $deg_\omega(m_1) = deg_\omega(m_2)$.

Following [24], a regular $n \times n$ matrix A over $\mathbb{Q}_+$ determines a term ordering $\prec$ by

$$t \prec r :\Leftrightarrow \bigvee_{i=1}^{n} \left(\bigwedge_{j=1}^{i-1} deg_{A_j}(t) = deg_{A_j}(r) \right) \wedge deg_{A_i}(t) < deg_{A_i}(r).$$

In other words, the rows A_i of matrix A contain weight vectors $\omega_i = A_i$, and $t \prec r$ is determined by comparing the ω_i-degrees of t and r until the first inequality is found. Furthermore, A can be any $m \times n$ matrix of rank n. We denote the ordering $\prec$ determined by A with $\mathcal{O}(A)$, and use $\mathcal{O}(\omega, A)$ for the ordering determined first by ω and then by the weight vectors of A.

Vice versa, given a term order $\prec$, there always exists a matrix A that determines $\prec$ [24].

Example. $\prec_{\text{lex}} = \mathcal{O}(A)$ and $\prec_{\text{tdeg}} = \mathcal{O}(B)$ where

$$A = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} 1 & 1 & \dots & 1 & 1 & 1 \\ 1 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \ddots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \end{pmatrix}.$$

For the walk, we assume that we are given a Gröbner basis $G(I, <)$ with respect to the start order $<$. Our aim is to compute the Gröbner basis of $G(I, \ll)$ with respect to the target order $\ll$. First, we determine the respective order matrices A and B with $\mathcal{O}(A) = <$ and $\mathcal{O}(B) = \ll$. σ and τ are the first rows of A and B and they are compatible (on R) with $<$ and $\ll$, respectively. The path of our walk is along the line segment $\overline{\sigma\tau}$, where we denote the occurring intermediate weight vectors by

$$\sigma =: \omega_1, \dots, \omega_k, \dots, \omega_m := \tau.$$

To obtain a uniform designation, we introduce the following notation:

$$\begin{aligned}
\prec_0 &:= \mathcal{C}(A) = \mathcal{C}(\sigma, A) = \ < \\
\prec_1 &:= \mathcal{C}(\omega_1, B) = \mathcal{C}(\sigma, B) \\
&\vdots \\
\prec_k &:= \mathcal{C}(\omega_k, B) \\
&\vdots \\
\prec_m &:= \mathcal{C}(\omega_m, B) = \mathcal{C}(\tau, B) = \mathcal{C}(B) = \ll
\end{aligned}$$

During the walk, we successively compute the (reduced) Gröbner bases of I with respect to the orderings $\prec_k$, $k = 1, \ldots, m$. These are called the *intermediate* Gröbner bases $G(I, \prec_k)$, which form the stepping-stones for the walk.

In the first step, we keep the first vector σ of the start order matrix and replace the rest by the target order matrix. In the algorithm, this first step is performed in the same way as the subsequent steps, which affects only few initials because the ordering comparison is by the σ-degree first. (In fact, the first step is trivial if we start inside a cone.)

In the subsequent steps, row σ is gradually changed into τ. All steps in the Walk are performed in the following way.

2.2. *One step*

Given the Gröbner basis $G(I, \prec_{k-1})$ and the weight vector ω_k, we take one step from ω_{k-1} to ω_k.[1] We first determine $in_{\omega_k}(G(I, \prec_{k-1}))$ (cf. Step ❷ in Figs. 1 and 2). This is a Gröbner basis of $\langle in_{\omega_k}(I)\rangle$ with respect to $\prec_{k-1}$, as ω_k is compatible with $\prec_{k-1}$ on $G(I, \prec_{k-1})$. By applying Buchberger's algorithm in Step ❹, we compute a reduced Gröbner basis $G(\langle in_{\omega_k}(I)\rangle, \prec_k) = \{m_1, \ldots, m_s\} = M$. This is usually a very short task, as most of the initials in $in_{\omega_k}(G(I, \prec_{k-1}))$ consist of only one monomial. By abuse of language, we sometimes call this basis of initials an *initial Gröbner basis*.

We remark that every S-polynomial of two ω-homogeneous polynomials is ω-homogeneous. Furthermore, every reduction of an ω-homogeneous polynomial by an ω-homogeneous polynomial results in an ω-homogeneous polynomial.

In the lifting Step ❻, we construct the new intermediate Gröbner basis $G(I, \prec_k)$. For that, we reduce each polynomial $m_i \in M = \{m_1, \ldots, m_s\} = G(\langle in_{\omega_k}(I)\rangle, \prec_k)$ by the Gröbner basis $G(\langle in_{\omega_k}(I)\rangle, \prec_{k-1}) = in_{\omega_k}(G(I, \prec_{k-1}))$ and obtain a representation

$$m_i = \sum_{j=1}^{r} h_{ij} in_{\omega_k}(g_j) \tag{1}$$

[1] ω_k is the *first* weight vector on the path $\overline{\omega_{k-1}\tau}$ which causes the degeneration of any other of the initial forms in the reduced Gröbner basis $G(I, \prec_{k-1})$. Obviously, ω_k is compatible with $\prec_{k-1}$ on $G(I, \prec_{k-1})$.

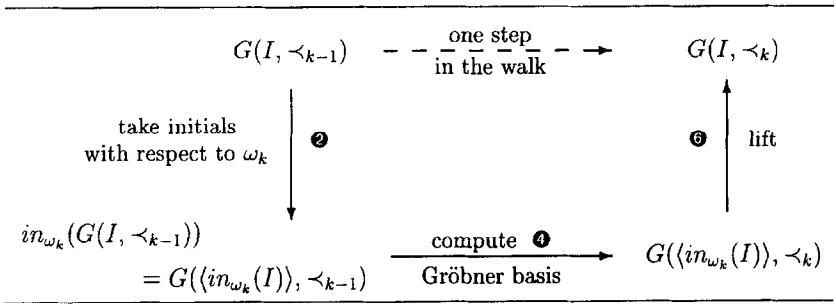


Fig. 1. Step k of the Gröbner Walk.

Gröbner_Walk

Input: Orders $<= \mathcal{O}(A)$ and $\ll= \mathcal{O}(B)$
$G \subset \mathbb{Q}[x_1, \ldots, x_n]$: reduced Gröbner Basis with respect to $<$
starting weight vector σ (first row of matrix A)
target weight vector τ (first row of matrix B)

Output: G: Gröbner Basis with respect to $\ll$

$\prec, \prec^+$: current and next order
ω: current weight vector

❶ $\omega = \sigma$; $\prec = \mathcal{O}(A)$; $\prec^+ = \mathcal{O}(\sigma, B)$;
❷ G_ω = initials(G, ω); // Take initials of G
❸ G_ω^+ = sort(G_ω, $\prec^+$); // Sort initials according to new order
❹ G_ω^+ = init_gb(G_ω^+, $\prec^+$); // Compute Gröbner Basis of initials
❺ G_ω^+ = interreduce(G_ω^+, $\prec^+$); // Interreduce G_ω^+
❻ G = lift(G_ω^+, $\prec^+$, G_ω, G, $\prec$); // Lift G_ω^+ to a full Gröbner B. wrt. $\prec^+$
❼ G = interreduce(G, $\prec^+$); // Interreduce G
❽ if($\omega = \tau$) return(G); // Stop if target order reached
❾ t = determine_border(ω, τ, G); // Determine the next parameter
if(undefined(t)) return(G); // No further conversion needed
ω = (1-t)*ω + t*τ; // Determine the next border
❿ $\prec = \prec^+$; $\prec^+ = \mathcal{O}(\omega, B)$; Goto ❷;

Fig. 2. The Gröbner Walk algorithm.

with $g_j \in G(I, \prec_{k-1})$. Because of the above remark, all polynomials m_i, $i = 1, \ldots, s$, are ω_k-homogeneous. Therefore, the polynomials h_{ij} in the representation of m_i are ω_k-homogeneous as well. Moreover, as m_i and each summand $h_{ij} in_{\omega_k}(g_j)$ have the same ω_k-degree, all $d_{ij} := h_{ij} g_j - h_{ij} in_{\omega_k}(g_j)$ contain only monomials of a smaller ω_k-degree than that of m_i. Therefore, we obtain

$$m_i = \sum_{j=1}^{r} h_{ij} in_{\omega_k}(g_j) = \sum_{j=1}^{r} h_{ij} g_j - \sum_{j=1}^{r} d_{ij} = in_{\omega_k}\left(\sum_{j=1}^{r} h_{ij} g_j\right). \tag{2}$$

In the lifting step, we replace all occurrences of polynomials $in_{\omega_k}(g_j)$ in the representation of m_i (Eq. (1)) by the full polynomials g_j and obtain the set of polynomials

$F := \{f_1, \ldots, f_s\}$ with

$$f_i := \sum_{j=1}^{r} h_{ij} g_j. \tag{3}$$

It remains to be shown that F is a Gröbner basis of I with respect to $\prec_k$, or $\langle in_{\prec_k}(I)\rangle = \langle in_{\prec_k}(F)\rangle$. As (by definition) ω_k is compatible with $\prec_k$ on R, it follows that

$$\langle in_{\prec_k}(I)\rangle = \langle in_{\prec_k}(in_{\omega_k}(I))\rangle \subseteq \langle in_{\prec_k}\langle in_{\omega_k}(I)\rangle\rangle$$

which is equal to $\langle in_{\prec_k}(M)\rangle$, as M is a Gröbner basis of $\langle in_{\omega_k}(I)\rangle$ with respect to $\prec_k$. Replacing the polynomials m_i by their representation (1) and applying Eq. (2), we obtain

$$\left\langle in_{\prec_k}\left\{\sum_{j=1}^{r} h_{ij} in_{\omega_k}(g_j) \,\middle|\, 1 \leqslant i \leqslant s\right\}\right\rangle = \left\langle in_{\prec_k}\left\{in_{\omega_k}\sum_{j=1}^{r} h_{ij} g_j \,\middle|\, 1 \leqslant i \leqslant s\right\}\right\rangle.$$

Since ω_k is compatible with $\prec_k$ (on R), this is equal to

$$\left\langle in_{\prec_k}\left\{\sum_{j=1}^{r} h_{ij} g_j \,\middle|\, 1 \leqslant i \leqslant s\right\}\right\rangle = \langle in_{\prec_k}(F)\rangle$$

and we finally obtained $\langle in_{\prec_k}(I)\rangle \subseteq \langle in_{\prec_k}(F)\rangle$. As $F \subseteq I$, F is a Gröbner basis of I with respect to $\prec_k$.

Since M is a minimal Gröbner basis of $\langle in_{\omega_k}(I)\rangle$ (M is even reduced), F is a minimal Gröbner basis as well. By performing an interreduction in Step ❼, we obtain a reduced Gröbner basis of I.

To finish this step of the walk, it remains to determine the next weight vector in Step ❾, that is, the point on the path where some (other) initial forms of the reduced Gröbner basis $G(I, \prec_k)$ degenerate. To detect a change in the initial forms, we determine the first weight vector $\omega(t) := \omega_k + t(\tau - \omega_k)$, $0 < t \leqslant 1$, on the directed path $\overline{\omega_k \tau}$ with the following property.

$$\begin{aligned} t \;=\; \min(\{s \mid & deg_{\omega(s)}(p_1) = deg_{\omega(s)}(p_i), deg_{\omega(0)}(p_1) \neq deg_{\omega(0)}(p_i), \\ & g = p_1 + \cdots + p_n, g \in G(I, \prec_k)\} \cap [0, 1]). \end{aligned}$$

This can be done by the calculation of one scalar product in $\mathbb{Q}^n$ and one rational quotient per monomial. Thus, if t is defined, it is a positive rational number. Furthermore, if there is no t with $0 < t \leqslant 1$, then $in_{\prec_k}(G) = in_{\ll}(G)$ and we already determined the final Gröbner basis; if $t = 1$ is the minimum, then the next conversion is the last step of the walk.

In this way, we successively compute the Gröbner bases and weight vectors

$$G(I, <) = G(I, \prec_0), \omega_1, G(I, \prec_1), \ldots, \omega_m, G(I, \prec_m) = G(I, \ll).$$

2.3. Termination of the algorithm

For the termination of the algorithm we need some further definitions [21, 9, 3]. Let $G \subseteq R$ be a reduced Gröbner basis of I with respect to a term ordering $\prec$. As we mentioned before, we can always find an order matrix A with $\prec = \mathcal{O}(A)$. Furthermore, we can choose a weight vector $\omega = A_1$ such that none of the initial forms of $g_i \in G$ degenerate. Then, there is even a neighborhood $C \subseteq \mathbb{Q}^n$ of ω such that for all $\omega' \in C$, ω' is compatible with $\prec$ on G as well. So, the leading monomials of $G(I, \prec)$ remain the same in the whole neighborhood C. Therefore, $G(I, \prec) = G(I, \mathcal{O}(\omega', A))$ for all ω' in C.

By $C_\prec(I)$ we determine the set of weight vectors ω which lead to the same reduced Gröbner basis as $\prec$. More precisely, for a term order $\prec$, we denote by $C_\prec(I)$ the topological closure of

$$\{\omega \mid \omega \text{ a weight vector}, \langle in_\prec(I)\rangle = \langle in_\omega(I)\rangle\}.$$

This is a convex cone in $\mathbb{Q}^n$ with nonempty interior. The set

$$F(I) = \{C_\prec(I) \mid \prec \text{ a term order}\}$$

is called the *Gröbner Fan* of I [21].

On the walk, we only have to take a step when we cross into a new cone, that is, when one (or more) of the initials in the Gröbner basis of I degenerate with respect to the upcoming weight vector $\omega(t)$. By a result from [21], the Gröbner fan of a polynomial ideal has finite cardinality, and hence the number of steps is finite.

3. Algorithmic variations: path perturbation

Experience shows that among the many paths from σ to τ (more precisely: their respective cones) in a Gröbner Fan, some may be computationally much faster, often by one to two orders of magnitude. Path finding is therefore an important issue in practice. Path Perturbation is a common principle for finding computationally efficient paths which we explore in a number of variations.

Whenever the path leaves a cone of the Gröbner Fan, some of the head monomials of the Gröbner basis with respect to the weight vector ω become initial forms (true *poly*nomials). Adjacent cones meet in faces, i.e., surfaces or edges in three dimensional Fans. At points of such intersections of several cones, either several monomials in a polynomial have the same maximal weight and become the initial form, or several polynomials have initial forms containing more than one monomial. Hence, the initial forms become larger. Especially in a complicated fan, meeting-points where several cones adjoin are frequent. Moreover, if the walk moves along the intersection of two or even more cones (i.e., along a surface in three-dimensional fans), there are monomials which keep the same maximal weight, and therefore remain in the initial form of a polynomial on this line. Anyway, both cases cause the initial forms to be unnecessarily heavy during the Walk.

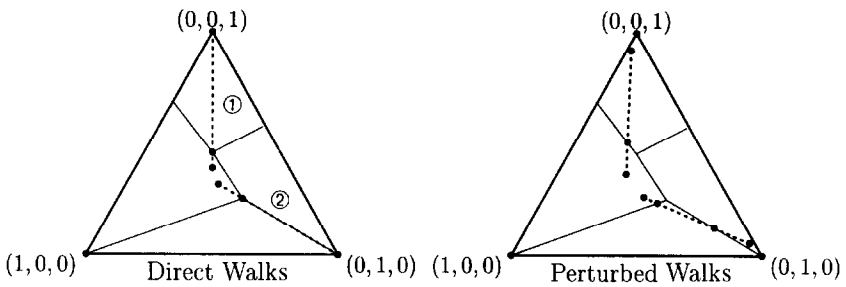


Fig. 3. A slice of a Gröbner Fan.

3.1. Global path perturbation

We can avoid walking through meeting-points or along arbitrary faces of cones if we slightly perturb the starting point (the starting weight vector σ) and the end-point (the target weight vector τ) of the Walk, making sure we stay in their cones. Then, the path passes through a sequence of maximally adjacent cones. The initial forms are shorter and the individual tasks of converting their Gröbner bases from $\prec_{k-1}$ to $\prec_k$ likewise become much smaller. However, we may have to compute more Gröbner bases since we may have to walk through more cones on the perturbed path.

Fig. 3 shows a slice of a sample fan of an ideal in three variables (x, y, z) as intersection with the plane $x + y + z = 1$. Path segment ① goes through a common edge of three cones (a point in the slice), path segment ② runs along a surface of two cones (a line in the slice).

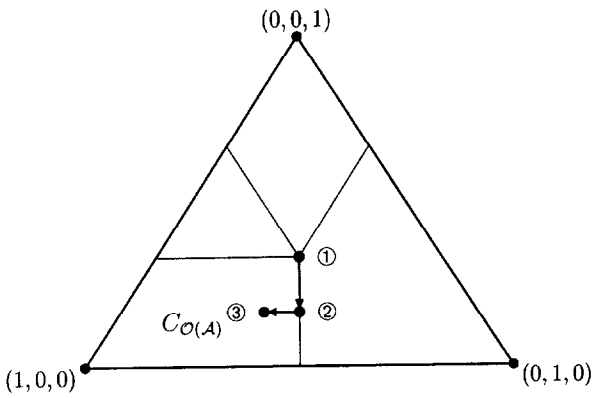
3.2. Computation of perturbed weight vectors

As already pointed out, with each ordering $\prec$ we can easily associate a regular $n \times n$-matrix A over $\mathbb{Q}_+$ that determines this ordering. Obviously, we can even choose A over $\mathbb{N}$ by multiplying each row with the least common multiple of the denominators of this row. The cone $C_\prec$ of a reduced Gröbner basis G with respect to $\prec$ contains the first vector A_1 of A. Now, if we perturb this vector slightly by εA_2, we remain in $C_\prec$, provided that ε is sufficiently small. This is assured if $1/\varepsilon > tdeg(p) * \max(A_2)$ for each p in G, where

$$tdeg(p) := \max\left\{ \sum_{i=1}^{n} e_i \,\middle|\, x_1^{e_1} \cdots x_n^{e_n} \text{ a term of } p \right\}.$$

However, if A_1 is at a vertex of the fan's slice (i.e., an edge and hence a one-dimensional face of the cone), $A_1 + \varepsilon A_2$ is in general at an edge in the slice (i.e., a two-dimensional face of the cone).

To reach a face of dimension at least three, we can perturb by the third vector as well, resulting in $A_1 + \varepsilon A_2 + \varepsilon^2 A_3$. Provided $1/\varepsilon > tdeg(p) * (\max(A_2) + \max(A_3))$ for each p in G, we move to a location that is contained in a three-dimensional open set of the cone.



$$A = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

Fig. 4. First, second, and third degree perturbation.

For n variables, we can extend this procedure up to any $k \leqslant n$ and obtain the perturbed vector

$$A_1 + \varepsilon A_2 + \varepsilon^2 A_3 + \cdots + \varepsilon^{k-1} A_k$$

of *degree* k (see Fig. 4). It is in a face of dimension at least k of the cone. For $k = 1$ we obtain the unperturbed weight vector. If $k = n$, we obtain a *maximally perturbed* weight vector that lies *within* the cone (and hence belongs to only one cone).

Given a reduced Gröbner basis with respect to a start ordering, an ε satisfying the conditions for the perturbation of the start vector can easily be determined. However, to obtain a perturbed target vector, we have to guess such an ε and check the validity at the end of the Walk. Note that the algorithm remains correct even if the ε is invalid. In this case we only did not reach the target ordering yet and have to walk further (closer to the unperturbed target).

The perturbation degrees of the start and target vector are a priori independent and can be combined in any way. However, if we perturb the starting vector by degree k, then during the Walk we usually[2] omit vectors on faces of dimensions $k' < k - 1$, as long as the chosen ε is valid. As the last step can cross a meeting point of several cones, it is usually useful to perturb the target vector, too.

The sequence of cones on the path is determined by the (larger of the) perturbation degrees of the start and target vectors. We will speak of a *direct path/walk* to indicate no perturbation and of a *maximally perturbed path/walk* to indicate maximally perturbed start and target vectors.

Experience shows that in general a small perturbation degree leads to fewer steps with larger initial forms, whereas a large perturbation degree leads to more steps with

[2] It may happen that the degree is lowered if the path direction coincides with one of the perturbation directions or a linear combination thereof.

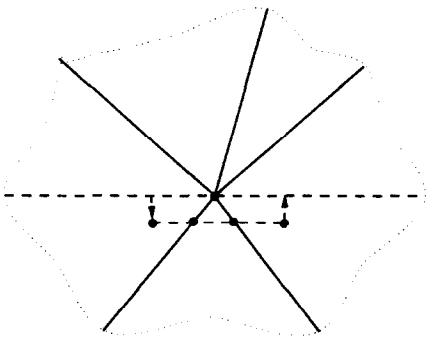


Fig. 5. The Evasive Walk.

smaller initial forms. The advantage of the maximally perturbed path is that the initial Gröbner basis computations take very little time. The main disadvantages are that very large weight vectors may occur (especially if there are many variables) and that the walk may consist of many more steps.

To overcome this dilemma, we need to perturb the path only slightly, so that only the weight vectors causing many large initials are avoided. One solution is the perturbation of start and target vectors by small degrees larger than one. Another solution is to use perturbation only where necessary.

3.3. *Local path perturbation: the Evasive Walk*

With global perturbation we decide statically, before the start of the walk, on the perturbation degree for the entire walk. However, often only very few vectors belong to many cones, and we would like to perturb the path only near these vectors and not as a whole. That is, we need a perturbation that is adaptive to the local situation.

Whenever a path approaches a border of a cone (weight vector ω), we may first measure its complexity by counting the number and size of the initial forms at ω. Depending on this number, we may either proceed on the path, or we may choose a path evading this border by a local perturbation (cf. Fig. 5). To walk on the bypass, we perturb ω once with the order of the actual cone (this is the local start vector) and once with the order of the next cone (this is the local target vector). We thus replace one giant step by several smaller ones.

This method, called the *Evasive Walk*, has not been implemented yet.

3.4. *The Fractal Walk*

As mentioned in [9], the initial Gröbner basis computation can be performed by any basis conversion algorithm, as the initials $in_\omega(G)$ form a Gröbner basis. The idea of the *Fractal Walk* is to apply a walk recursively to this basis conversion.

Taking initials with respect to ω once again does not shorten the polynomials in $in_\omega(G)$ any further. The idea is now to perturb ω into ω' in the hope that $in_{\omega'}(in_\omega(G))$

contains shorter polynomials than $in_\omega(G)$. In other words, we want to walk on a perturbed path, starting with Gröbner basis $in_{\omega'}(in_\omega(G))$, and finishing with a result which is equal to the result of the initial Gröbner basis computation we wish to avoid. In contrast to the Evasive Walk, the path is on a "lower" level (in a "tunnel") because it runs in $\langle in_\omega(G)\rangle$ rather than in $I=\langle G\rangle$. As the stepping-stones during a fractal walk are ideal bases of $\langle in_\omega(G)\rangle$, we avoid the lifting to bases of the full ideal $I=\langle G\rangle$. Hence the interreductions after lifting should become substantially simpler.

This could be viewed at first as an algorithmic optimization of the Evasive Walk. However, the method can be applied recursively with increasing levels of perturbation. In each step, the higher the level of perturbation, the shorter the initials of the Gröbner bases. Obviously, we obtain the same number of steps as we would obtain with a normal walk on a maximally perturbed path. However, the interreductions become much smaller on each level. As a price, we have the additional overhead of taking initials and lifting the initial Gröbner basis on each level.

The Fractal Walk has not been implemented yet. It exhibits similarities with the *Gröbner Stripping Algorithm* [10] which deserve further investigation.

4. Implementation techniques

We have observed performance increases by one to two orders of magnitude as a result of refining our implementation. There are two sets of problems: first, implementation techniques for the walk proper, and, second, implementation techniques for the special class of Gröbner basis computations occurring during the walk. For the latter, we shall see that the distribution of computation time in Buchberger's algorithm is significantly different on initial forms, and we have the new situation that hundreds of Gröbner basis computations may be needed in quick succession.

Besides the prototype implementation of the Walk [9], we are only aware of one other report on an implementation [23].

4.1. Integral polynomials

Most of our implementation is independent of the particular choice of the coefficient field K, but all our examples are chosen over $R=\mathbb{Q}[x_1,\ldots,x_n]$, hence $K=\mathbb{Q}$.

Since polynomial arithmetic with coefficients in $\mathbb{Z}$ is much faster than with coefficients in $\mathbb{Q}$, we do not store the polynomials in monic form, but convert them to integral polynomials. It is common knowledge that this leads to significant speedups, typically of a factor around 5, for Buchberger's algorithm. Then, the reduction of a polynomial's tail becomes more complex, as we may have to multiply all coefficients with a cofactor in order to be able to perform a reduction step.

During the walk, it does not much affect the computation of the Gröbner bases of the initials as most of the initials are monomials and hence their coefficients can be nor-

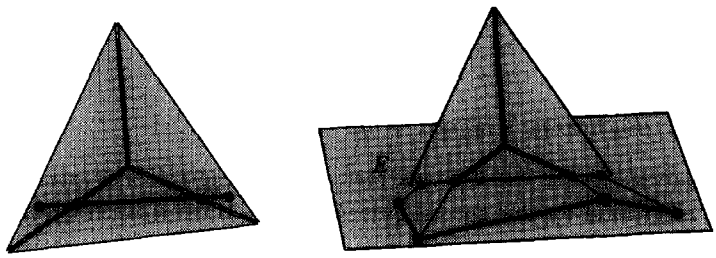


Fig. 6. The Zigzag Walk.

malized. However, using integral polynomials is rather important for the interreduction of the intermediate Gröbner bases.

4.2. Weight vectors

Given the ordering $\mathcal{O}(\omega, A)$, every comparison of two monomials involves costly rational arithmetic with the rational weight vector ω. In particular, the higher the perturbation degree has been chosen, the longer the representation of the rational numbers in the weight vectors may become. Therefore, it is important to make the comparisons as efficient as possible.

4.2.1. Integral weight vectors: the Zigzag Walk

Neither the algorithm nor the implementation requires the walk to stay on the hyperplane $\sum x_i = 1$.[3] As only the direction of the weight vectors is needed, we may scale them to integral vectors. Then all comparisons become much cheaper, mainly because their computation does not consume heap space any more.

The first step for this modification of the implementation is to choose an integral starting vector and an integral target vector. Moreover, all intermediate vectors can be chosen integral as well. From the geometric point of view, the walk is then on a zigzag course in the plane E through starting point, target point, and the zero of the fan (see Fig. 6). Its projection onto the plane $\sum x_i = 1$ gives the original walk, which also lies on plane E.

4.2.2. Storing the weights

As reported in [23], it is worth to store the weights of the monomials. However, all computations with initials involve only ω-homogenous polynomials and therefore we simply have to omit the computations of the weights for the comparisons between monomials. (Only for sorting the set of initials in Steps ❸, ❹, and ❺, the weight of the leading monomials have to be computed.) Thus, we compute and store the weights during the lifting step and use the weights for the interreduction, after which

[3] In fact, the Gröbner fan can be regarded as an object in projective space.

we delete the weights again (as afterwards the weight vector and hence the weights change).

4.3. Specializing Buchberger's algorithm for initial forms

A closer analysis of the initial forms that occur in the Gröbner basis computation suggested an adaptation for this special case. We need to tailor Buchberger's algorithm to this unusual situation, because we wish to perform a possibly long sequence of such computations.

In a conventional implementation of Buchberger's algorithm, the reduction of S-polynomials is the most time consuming part. Therefore, it is rather unimportant how efficiently the pairs are created, and it is profitable to apply very sophisticated deletion criteria even if relatively few pairs are removed.

In our case, in particular in the perturbed walk, most of the initial forms are monomials. In fact, in our examples typically only one or two initials out of 60 or more polynomials are not monomials. In this situation, the following easy observation assumes some practical importance.

Monomial criterion: *The critical pair of two monomials is unnecessary, because its S-polynomial is equal to zero.*

Moreover, initial forms are by their very nature relatively short, and hence the corresponding S-polynomials are relatively short and their reduction is relatively cheap. Therefore, the other parts of the algorithm, and in particular the creation of pairs, become much more important. Hence, all key decisions in configuring Buchberger's algorithm have to be re-evaluated.

4.3.1. Creating pairs and applying criteria

In order to implement the monomial criterion, we maintain two lists of polynomials, one with p monomials only and one with q true polynomials. This allows us to avoid on the order of p^2 operations when creating the pairs.

Buchberger's first criterion (the product criterion) can also be applied very fast, because it uses only local data of the two parent polynomials.

For the second criterion (the chain criterion) [7], however, one has to search the whole basis for a suitable third polynomial. We found that in our average case it is cheaper to reduce an S-polynomial of initial forms than to search the whole basis. Therefore, we omit the chain criterion because it takes too much time.

4.3.2. Selection strategies

Experience shows that Gröbner bases of two maximally adjacent cones (as we deal with in the perturbed walk) are very similar. The computation of the initial Gröbner basis usually adds only one or two new polynomials to the basis. As most of the S-polynomials reduce to zero in one step, it is not worth spending time on sophisticated selection strategies.

4.3.3. Reduction

Whenever we can apply a reducer that consists of one monomial only, the reduction step amounts to a removal of the respective monomial in the polynomial that is reduced. Therefore, it is worth applying reductions by monomials first.

5. Empirical results

We observed the run times displayed in Table 1 on examples of the PoSSo Library [22]. All times are in seconds on one 90 MHz HyperSPARC[4] processor of a SUN-10 under Solaris 2.4/2.5. Our implementation uses some software modules of the PARSAC framework [1, 19], but it is completely sequential (no threads of control). Timings of some examples with different combinations of our algorithmic variations and implementation techniques can be found in [3]. Reference timings are for version 4 of the GB system [13]. LEXICO and SUGAR refer to the respective options of GB; speedup is computed as SUGAR/(Grevlex + Walk).

5.1. Where does the time go?

A closer investigation of the run-times yields a few valuable guidelines for setting up the Walk. More information on this question can be found in [3].

In almost all cases, the computation of the initial Gröbner bases and the interreduction of the intermediate Gröbner bases account for more than 90% of the time. On a direct walk, most of the time is spent in initial Gröbner bases computations and little in interreductions. In a perturbed walk, most of the time is spent in interreductions and little in initial Gröbner bases computations. If the polynomials of the intermediate

Table 1
Timings for system solving

	GB			PARSAC-S		Speedup
	LEXICO	SUGAR	GREVLEX	Grevlex	Walk	
arnborg7	186 s	112 s	0.23 s	0.32 s	0.4 s	156
caprasse	5288 s	3174 s	0.23 s	0.18 s	1.0 s	2645
cassou	> 1 h	> 1 h	59.0 s	28.0 s	0.6 s	> 126
cohn-2	> 1 h	> 1 h	12.6 s	16.9 s	46.5 s	> 57
fateman	466 s	280 s	0.7 s	0.6 s	73.0 s	4
hieta-1	> 1 h	> 1 h	147.0 s	46.0 s	3.9 s	> 72
jhd6	> 1 h	> 1 h	18.5 s	43.0 s	33.4 s	> 47
jhd6.inf	> 1 h	> 1 h	14.3 s	10.0 s	0.2 s	> 353
katsura5	> 1 h	> 1 h	13.4 s	5.1 s	4.8 s	> 360
vermeer	> 1 h	> 1 h	1.2 s	0.1 s	0.2 s	> 12 000

[4] Roughly equivalent to a 50 MHz SuperSPARC.

basis consist of several thousands of monomials, it happens that the lifting step needs more than 10% of the time; in this case, the interreduction accounts for more than 80% of the time. Overall, perturbed walks have many more steps but nevertheless are substantially faster than direct walks.

Therefore, interreduction is a very significant factor in the run-time, and the question arises what effect its omission would have on the duration of the walk [3]. The general experience is the following: Reduced intermediate bases lead to fewer steps in the walk. The extra (superfluous) steps occur when monomials which are reducible lead to changes in initial forms. On smaller examples it may be worth omitting the interreduction; on larger examples the interreductions tend to save more time than they cost.

6. Comparison with FGLM

For zero-dimensional ideals the Gröbner Walk can be compared with the FGLM basis conversion method as presented in [12]. There are ways to lift this restriction of FGLM in special cases, but there is reasonable doubt as to their practical significance [20].

The fundamental idea seems to have been already sketched by Buchberger [6] for the special case of constructing a triangular polynomial system for the representation of the roots of the ideal. We paraphrase from the German original: *We determine successively for all* $[x_1^k]$, $k=0,1,\ldots$, *their representation as a linear combination of the elements of* R/I *(by reduction to normal form in the given Gröbner basis). We also find the first* $k=m_1$ *for which* $[x_1^k]$ *linearly depends on* $[1],[x_1],\ldots,[x_1^{k-1}]$. *Thus we find a polynomial of degree* $m-1$ *with* $p_1([x_1])=0$. *Now we form the residue classes of the power products emerging from the multiplication of* $1,x_1,\ldots,x_1^{m_1-1}$ *by* $x_2,x_2^2,\ldots$ *until we find for some residue class* $[x_1^{i_1}x_2^{i_2}]$ *a dependency on earlier residue classes which can be written as* $p_2([x_1],[x_2])=0$, *and so on.*

The method thus enumerates R/I (which is bounded because the ideal is zero-dimensional) and uses the given Gröbner basis for computation in R/I. If the ideal is not zero-dimensional, some other means for bounding the process must be found. Thus the complexity of the method grows with the dimension of R/I. The conversion to the new Gröbner basis always needs about the same computational effort, regardless whether we start with a basis in its neigborhood or not.[5]

Table 2 of timings gives a rough idea of the relative speeds of the FGLM method and the Gröbner Walk. It is based on our own implementation of FGLM, which may not be tuned to the same degree as our implementation of the Walk, but we attempted a reasonable comparison. For details of the examples see the Appendix and [3].

[5] Of course the computations in R/I still depend on the basis.

Table 2
Timings of Walk vs. FGLM

	Walk	FGLM		Walk	FGLM
Ex1 grev → lex	15.3 s	113.4 s	Ex1 tot → lex	15.1 s	143.0 s
Ex2 grev → lex	119.8 s	523.4 s	Ex2 tot → lex	119.6 s	481.4 s
Ex3 grev → lex	71.6 s	677.9 s	Ex3 tot → lex	71.4 s	912.8 s
s6 grev → lex	15.8 s	55.1 s	s6 tot → lex	5.7 s	55.0 s
s7 grev → lex	164.9 s	739.4 s	s7 tot → lex	64.1 s	769.1 s

7. Applications

7.1. Implicitization in computer graphics

In geometric modeling, the problem of converting parametrically defined varieties into their implicit form is of great importance [15]. The parametric representation of a surface is most suitable for rendering it on an output device. It is however ill suited for the computation of intersections, for which the implicit representation is more amenable. As mentioned in [9], the Gröbner Walk should be an attractive tool for the conversion. In the following, after giving a short outline of the theory involved (cf. [16, 17]), we introduce an algorithmic improvement called *Sudden Death*, and we present a first collection of empirical results.

Given rational functions $p_1/q_1,\ldots,p_n/q_n$ in $t_1,\ldots,t_m$, defining the parametric equations

$$x_1 = p_1(t_1,\ldots,t_m)/q_1(t_1,\ldots,t_m)$$

$$\vdots$$

$$x_n = p_n(t_1,\ldots,t_m)/q_n(t_1,\ldots,t_m)$$

how can we find (polynomial) equations in $x_1,\ldots,x_n$ that define the same variety? The basic idea is to eliminate the variables $t_1,\ldots,t_m$ from the equations above. In general, this can be achieved by computing the elimination ideal

$$\left\langle \left\{ q_1x_1 - p_1,\ldots,q_nx_n - p_n, \left(\prod_{i=1}^{n} q_i\right)z - 1 \right\} \right\rangle \cap K[x_1,\ldots,x_n]$$

in the polynomial ring $K[x_1,\ldots,x_n,t_1,\ldots,t_m,z]$.

If all of the q_i are equal to 1, we can leave out the last polynomial and determine the elimination ideal $I \cap K[x_1,\ldots,x_n]$ where I is the ideal $\langle\{x_1 - p_1(t_1,\ldots,t_m),\ldots, x_n - p_n(t_1,\ldots,t_m)\}\rangle$.

In the case of a polynomial parameterization, the set $\{x_1 - p_1,\ldots,x_n - p_n\}$ already forms a Gröbner basis with respect to every term order $<= \mathcal{O}(\sigma,A)$ with

$$\sigma = (\underbrace{1,\ldots,1}_{n},\underbrace{0,\ldots,0}_{m}).$$

Table 3
Timings for implicitization

Example	Buchberger	Walk	Sudden death
Ex4	>2 h	43.7 s	17.8 s
Ex5	>10 h	433 s	265 s
Ex6	—	2676 s	158 s
Ex7	>1 h	76.7 s	49.9 s
Ex8	—	3639 s	122 s
Ex9	>1 h	153 s	68.3 s

With the Gröbner Walk algorithm, we can then successively transform this set into a Gröbner basis using the target vector τ, where

$$\tau = (\underbrace{0,\ldots,0}_{n},\underbrace{1,\ldots,1}_{m}).$$

We can improve the method by short-cuts if we know the number and shape of polynomials we are looking for. In the usual three-dimensional case, where we look for an implicit representation of a surface, we start with three polynomials $x_1 - p_1(t_1,t_2)$, $x_2 - p_2(t_1,t_2), x_3 - p_3(t_1,t_2)$. Then we can finish our walk as soon as we have found an (irreducible) polynomial which only depends on x_1, x_2, and x_3. Usually, this polynomial occurs quite early during the walk, which means we can stop in *sudden death* long before we reach the last cone of our path.

The timings in Table 3 compare the different methods on examples presenting Bézier surfaces.[6] The second column shows the timings for Buchberger's algorithm; the software is the sequential derivate of our PARSAC installation [1]. The third column shows the timings for the maximally perturbed walk from the starting cone of $(1,1,1,0,0)$ to the target cone of $(0,0,0,1,1)$. The last column shows the timings we obtained when we stopped the walk after finding the (irreducible) polynomial in three variables. With this method, we compute the *first* Gröbner basis on our walk with respect to an elimination ordering. This method cannot be simulated by any other basis conversion algorithm we are aware of, as we only know the target ordering *after* the fact that we found the result.

7.2. *Inverse kinematics in robotics*

Given the configuration of a robot (i.e., the lengths $l_1,\ldots,l_r$ of the arm segments and the angles $\vartheta_1,\ldots,\vartheta_t$ in the joints), the *Forward Kinematics Problem* is to determine the position (x,y,z) of the hand (effector). In fact, the coordinates x, y, and z can be represented as polynomials in $l_1,\ldots,l_r$ and $\cos(\vartheta_1),\sin(\vartheta_1),\ldots,\cos(\vartheta_t),\sin(\vartheta_t)$. Thus, forward kinematics is essentially the evaluation of polynomials for a particular setting of $l_1,\ldots,l_r$ and $\vartheta_1,\ldots,\vartheta_t$.

[6] Our source for the examples was [18].

The inverse problem is more complex: given the position (x, y, z) of the hand, we want to determine whether it is possible to place the hand of the robot at that point. If it is possible, we wish to find all angles $\vartheta_1, \ldots, \vartheta_t$ and lengths of the prismatic joints $l_1, \ldots, l_{r'}$ that will satisfy this *Inverse Kinematics Problem.*

In algebraic terms, given the polynomial equations

$$x = p_1(l_1, \ldots, l_r, c_1, s_1, \ldots, c_t, s_t)$$

$$y = p_2(l_1, \ldots, l_r, c_1, s_1, \ldots, c_t, s_t)$$

$$z = p_3(l_1, \ldots, l_r, c_1, s_1, \ldots, c_t, s_t)$$

$$c_1^2 = 1 - s_1^2$$

$$\vdots$$

$$c_t^2 = 1 - s_t^2$$

we want to solve for $c_1, \ldots, c_t$, and $l_1, \ldots, l_{r'}$. Obviously, this can be done using Gröbner bases [11]. However, the given equations already form a Gröbner basis with respect to any ordering with

$$x, y, z \gg c_1, \ldots, c_t, \gg s_1, \ldots, s_t, l_1, \ldots, l_{r'}.$$

Hence, we can directly apply basis conversion.

7.3. *Searching for ideal members*

Sometimes we are interested in searching for ideal members with a given property P, rather than computing a Gröbner Basis. We briefly sketch a general framework for searching which is based on the Walk. The previous examples of searching for implicit polynomials and of inverse kinematics polynomials fit into this framework, as does searching for reducible polynomials in a system solving context.

We use the Walk to convert the ideal presentation into a form in which polynomials with some property P are likely, or certain, to occur. Once such an ideal member is found, the Walk may continue, or stop in sudden-death as in implicitization, or even change direction and follow a new target ordering.

Suppose we want to perform an operation F on (some) elements of the ideal I. We know, for example from the asymptotic complexity of F, that it is preferable to run F only on polynomials with property P. We also know that such polynomials are likely, or certain, to occur in $G(I, <)$, for some $<$. We may then use the walk to massage the ideal representation in the right direction, i.e., we *walk down a complexity slope* w.r.t. F, towards $G(I, <)$.

In a system solving context, F may be any operation for splitting the problem, such as factoring or g.c.d. computations (cf. [26]). These operations are much cheaper on

polynomials with few variables; some practically fast algorithms have exponential complexity in the number of variables. Therefore, we may approach some $G(I, <)$ with an elimination ordering $<$ and try F only on polynomials in few variables. More precisely, we may make a run for the (nearest) border of the fan, choosing a first target vector of the form $(1, \ldots, 1, 0, 1 \ldots, 1)$. In the zero-dimensional case, we are certain to obtain a univariate polynomial. But with the Gröbner Walk we can apply this method also in the nonzero dimensional case, as we will always obtain polynomials with fewer variables on which F runs much faster. As soon as F has split some ideal member(s), we may split the entire problem and proceed with several walks, one for each fragment.

8. Parallelization

Since our sequential implementation is already within the PARSAC framework, we have a migration path towards practical parallelism on networks of multiprocessor workstations [19]. A combination with our parallel Gröbner basis algorithm [1] will then yield a parallel equation solver. In a system solving context (cf. Table 1), about half the run-time is now consumed by the Walk.[7] Since Buchberger's algorithm (the first half) can be speeded up substantially by parallelization [1], it is interesting to attack the Walk (the second half) likewise. We will restrict ourselves here to a few fundamental observations and reserve a thorough treatment for the future.

Our parallel speedups for Buchberger's algorithm are due to parallel reductions of S-polynomials, which are rather insignificant in the specialized Buchberger algorithm for initial forms. Therefore we do not expect much gain from attacking the initial Gröbner basis computations. However, we may parallelize the interreductions after lifting, which account for the lion's share of time in large perturbed walks.

Parallel work occurs at a higher granularity when we split a problem (cf. Section 7.3) and the fragments are walked to their destinations concurrently.

Work parallelism such as this generates speedups by performing a given amount of work in parallel, and is therefore limited to at most linear speedups. In contrast, search parallelism may exhibit super-linear speedups.

A typical source of such parallelism is indicated in Section 7.3. We may pursue several searches in parallel, and stop as soon as the first walk reached the target. We may also employ a searching party of several searching walks concurrently to look for, and act upon, desirable ideal members, such as univariate polynomials in $x, y, \ldots$.

In practice, a big challenge for the parallelization will be to find the best configurations among all algorithmic and implementation options for the parallel implementation, and to achieve significant speedups over the best sequential configuration.

[7] We expect the share of the Walk to increase on larger examples.

9. Conclusions

Our installation of the Gröbner Walk has yielded speedups of 2–4 orders of magnitude for many examples of system solving (cf. Section 5). For the Computer Graphics application of implicitization, the Walk allowed the computation of examples which were unreachable by Buchberger's algorithm alone (cf. Section 7).

Among its competitors "FGLM" and "Hilbert driven Buchberger", the Walk stands out as the method which places no additional restrictions whatsoever on the input, be it the dimension of the ideal or the shape of the polynomials in the presentation. Our comparison with FGLM (cf. Section 6) has shown that the Walk is an order of magnitude faster in our implementation and on our examples. A conservative conclusion may be that the Walk is no slower than FGLM, but free of any restrictions.

The theoretical foundation of the Walk has proved to be sufficiently broad to accomodate the variations which an implementation and successful applications require. It is possible to improve the speed of a naive implementation by 1–3 orders of magnitude by the theoretical and practical methods described in Sections 3 and 4 of this paper. For the Computer Graphics application, the special *sudden death* variation yielded an extra speedup of an order of magnitude on larger examples.

We have also outlined sources of parallelism in the Walk algorithm which give hope for further very substantial speedups with super-linear components on existing parallel computers.

Based on these findings we conclude that the Gröbner Walk will become an essential component in applications of Gröbner bases; conversely, it is scarce imaginable that real world applications can afford to not include some kind of basis conversion at least as powerful as the Gröbner Walk.

Acknowledgements

We are above all indebted to Stéphane Collart, Michael Kalkbrener, and Daniel Mall for sharing their work with us at a very early stage, including many helpful discussions and suggestions, such as an independent suggestion for path perturbation. We wish to thank Bruno Buchberger and Wolfgang Windsteiger for making their GRÖBNER package publicly available; it helped us considerably in setting up our first versions of the PARSAC-S WALK. We are grateful to Reinhard Klein for providing us with a choice of interesting examples from Computer Graphics, including the Bézier surfaces. This work is based upon research supported by grant Ku 966/2-1 from Deutsche Forschungsgemeinschaft.

Appendix

Example	Variables	Polynomials
Ex1	$x<y<z$	$xy^3+y^4+yz^2-z^3-2xz^3$, $2x^2y+x^3y+2xy^2z$, $2-3x^2y+2x^3y+yz^3$
Ex2	$x<y<z$	$x+3xy^3+y^4+yz^2$, $-x^2z+2y^3z+z^2+2yz^2+3xyz^2$, $3x^3+xy^2+yz^2-2xz^3$
Ex3	$x<y<z$	$x^2+y^4+x^3z+yz-2xz^3$, $x^2y^2+y^3z+z^3+3yz^3$, $y^4-x^2z+2y^2z-2xyz^2$
s6	$x_6<x_5<x_4$ $<x_3<x_2<x_1$	$2x_6x_2+2x_5x_3+x_4^2+x_1^2+x_1$, $2x_6x_3+2x_5x_4+2x_2x_1+x_2$, $2x_6x_4+x_5^2+2x_3x_1+x_3+x_2^2$, $2x_6x_5+2x_4x_1+x_4+2x_3x_2$, $x_6^2+2x_5x_1+x_5+2x_4x_2+x_3^2$, $2x_6x_1+x_6+2x_5x_2+2x_4x_3$
s7	$x_7<x_6<x_5<x_4$ $<x_3<x_2<x_1$	$2x_7x_2+2x_6x_3+2x_5x_4+x_1^2+x_1$, $2x_7x_3+2x_6x_4+x_5^2+2x_2x_1+x_2$, $2x_7x_4+2x_6x_5+2x_3x_1+x_3+x_2^2$, $2x_7x_5+x_6^2+2x_4x_1+x_4+2x_3x_2$, $2x_7x_6+2x_5x_1+x_5+2x_4x_2+x_3^2$, $x_7^2+2x_6x_1+x_6+2x_5x_2+2x_4x_3$, $2x_7x_1+x_7+2x_6x_2+2x_5x_3+x_4^2$

Example	Bézier surfaces in parametric form
Ex4	$x=u+u^2-2v-2u^2v+2uv^2$ $y=-6u+2v+v^2-5v^3+2uv^2-4u^2v^2$ $z=-2+2u^2+6v-3u^2v^2$
Ex5	$x=3-2u+2u^2-2u^3-v+uv+2u^2v^3$ $y=6u+5u^2-u^3+v+uv+v^2$ $z=-2+3u-uv+2uv^2$
Ex6	$x=2u^3v-uv^2+6uv^3$ $y=3u^2+3uv-u^2v+4u^3v+v^2$ $z=-uv+4u^2v-3u^3v+v^2$
Ex7	$x=3-3u-3u^2+4u^3-9v+2u^2v^2$ $y=6u^2-3u^3+3uv-4u^2v+2u^3v^2$ $z=3u^2+3u^3-9uv+2u^3v^2$

Example	Bézier surfaces in parametric form
Ex8	$x = 2 - 6u + 6u^2 + 6v + u^2v - 3u^3v - 2v^2$ $y = 3 - 9u + u^2 - 3u^3 + 3v^2 + 9uv^2$ $z = 9u^2 - 9u^3 - u^2v + 2u^3v + u^2v^2$
Ex9	$x = 2u - 2u^2 + u^3 - 2uv + u^2v$ $y = u - 4u^3 + 3v - 2uv + u^2v - 6v^2 + uv^2 + 3v^3 - uv^3$ $z = 3 - 5u + u^2 - 2v + uv^2$

A Bézier surface of degree $m \times n$ with parameters $b_{i,j} \in \mathbb{Q}^3$, $i = 0, \ldots, m$, $j = 0, \ldots, n$, is of the form [18]

$$f(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} \binom{m}{i} \binom{n}{j} u^i v^j (1-u)^{m-i} (1-v)^{n-j}.$$

The other examples originate from the PoSSo [22] examples list.

References

[1] B. Amrhein, O. Gloor, W. Küchlin, A case study of multi-threaded Gröbner basis completion, in: Y.N. Lakshman (Ed.), ISSAC'96, Zurich, Switzerland, ACM Press, New York, 1996, pp. 95–102.

[2] B. Amrhein, O. Gloor, W. Küchlin, How fast does the Walk run?, in: A. Carrière, L.R. Oudin (Eds.), 5th Rhine Workshop on Computer Algebra, Saint-Louis, France, 1996, ISL, pp. 8.1–8.9.

[3] B. Amrhein, O. Gloor, W. Küchlin, Walking faster, in: J. Calmet, C. Limongelli (Eds.), DISCO'96, Karlsruhe, Germany, Lecture Notes in Computer Science, vol. 1128, Springer, Berlin, 1996, pp. 150–161.

[4] Buchberger, Collins, Encarnación, Hong, Johnson, Krandick, Loos, Mandache, Neubacher, Vielhaber, SACLIB User's Guide, 1993. On-line software documentation.

[5] B. Buchberger, Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal, Ph.D. thesis, Universität Innsbruck, 1965.

[6] B. Buchberger, Ein algorithmisches Kriterium für die Lösbarkeit eines algebraischen Gleichungssystems, Aequationes Math. 4 (3) (1970) 374–383.

[7] B. Buchberger, A criterion for detecting unnecessary reductions in the construction of Gröbner-Bases, in: E.W. Ng (Ed.), EUROSAM'79, Marseille, France, Lecture Notes in Computer Science, vol. 72, Springer, Berlin, 1979, pp. 3–21.

[8] B. Buchberger, Gröbner bases: an algorithmic method in polynomial ideal theory, in: N.K. Bose (Ed.), Recent Trends in Multidimensional Systems Theory, Reidel, Dordrecht, 1985, chap. 6.

[9] S. Collart, M. Kalkbrener, D. Mall, Converting bases with the Gröbner Walk, J. Symbol. Comput. 1997, to print.

[10] S. Collart, D. Mall, The ideal structure of Gröbner base computations, Preprint, 1994.

[11] D. Cox, J. Little, D. O'Shea, Ideals, Varieties, and Algorithms, Undergraduate Texts in Mathematics, Springer, Berlin, 1992.

[12] J. Faugère, P. Gianni, D. Lazard, T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering, J. Symbol. Comput. 16 (1993) 329–344.

[13] J.C. Faugère, Résolution des systèmes d'équations algébriques, Ph.D. thesis, Université Paris 6, 1994.

[14] C. Traverso, Hilbert functions and the Buchberger algorithm, J. Symbolic Comput. 22(4) (1996) 355–376.

[15] C. Hoffmann, Geometric and Solid Modeling: An Introduction, Morgan Kaufmann, Los Altos, CA, 1989.

[16] M. Kalkbrener, Implicitization of rational curves and surfaces, in: S. Sakata (Ed.), AAECC-8, Lecture Notes in Computer Science, vol. 508, Tokyo, Japan, Springer, Berlin, 1990, pp. 249–259.

[17] M. Kalkbrener, Implicitization by Gröbner basis conversion, in: IMACS'95, Albuquerque, NM, 1995. http://math.unm.edu/ACA/1995/Proceedings.
[18] R. Klein, Netzgenerierung impliziter und parametrischer Kurven und Flächen in einem objektorientierten System, Ph.D. Thesis, Eberhard-Karls-Universität, Tübingen, 1995.
[19] W.W. Küchlin, PARSAC-2: Parallel computer algebra on the desk-top, in: J. Fleischer, J. Grabmeier, F. Hehl, W. Küchlin (Eds.), Computer Algebra in Science and Engineering, World Scientific, Singapore, 1995, pp. 24–43.
[20] S. Licciardi, T. Mora, Implicitization of hypersurfaces and curves by the Primbasissatz and basis conversion, in: J. von zur Gathen, M. Giesbrecht (Eds.), ISSAC'94, Oxford, England, ACM Press, New York, 1994, pp. 191–196.
[21] T. Mora, L. Robbiano, The Gröbner Fan of an ideal, J. Symbol. Comput. 6 (1988) 183–208.
[22] PoSSo, Polynomial systems library, ftp: posso.dm.unipi.it.
[23] A. Steel, The Magma Gröbner Walk, 1996, Preprint.
[24] V. Weispfenning, Admissible orders and linear forms, ACM Sigsam Bull. 21 (2) (1987) 16–18.
[25] W. Windsteiger, B. Buchberger, GRÖBNER: A Library for computing Gröbner Bases based on SACLIB, Manual for Version 2.0, 1993.
[26] D.Y.Y. Yun, On algorithms for solving systems of polynomial equations, ACM Sigsam Bull. 27 (1973) 19–25.