Polynomial Programming Using Groebner Bases *

Yao-Jen Chang and Benjamin W. Wah
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801
{chang,wah}@manip.crhc.uiuc.edu

Abstract

Finding the global optimal solution for a general nonlinear program is a difficult task except for very small problems. In this paper we identify a class of nonlinear programming problems called polynomial programming problems (PP). A polynomial program is an optimization problem with a scalar polynomial objective function and a set of polynomial constraints. By using Groebner Bases, we can determine the global minimum of a polynomial program in a reasonable amount of time and memory.

1 Introduction

In almost every engineering discipline, the need for optimizing a particular application at hand arises constantly. In engineering applications, global solutions mean the most efficient use of resources under practical constraints.

Optimization techniques have been a fruitful domain in engineering research. For linear and quadratic programming problems, efficient algorithms have been developed to guarantee global solutions with convexity. For general nonlinear programs, various optimization procedures have been designed; however, most of these procedures aim at finding local solutions while exploiting differential information (such as gradient) in one form or another.

A general nonlinear program is often too complex for us to find its global solution. In this paper, we propose a class of nonlinear programming problems called polynomial programming problems. A *polynomial program* is a mathematical program with a polynomial scalar objective function constrained by a polynomial system of equations. Polynomial programs specialize to *linear programs* if both the objective and constraint set are linear, and specialize to *quadratic programs* if the objective is a second-degree polynomial and the constraint set is linear. The relation between polynomial programming problems and other mathematical programming problems is shown in Figure 1.

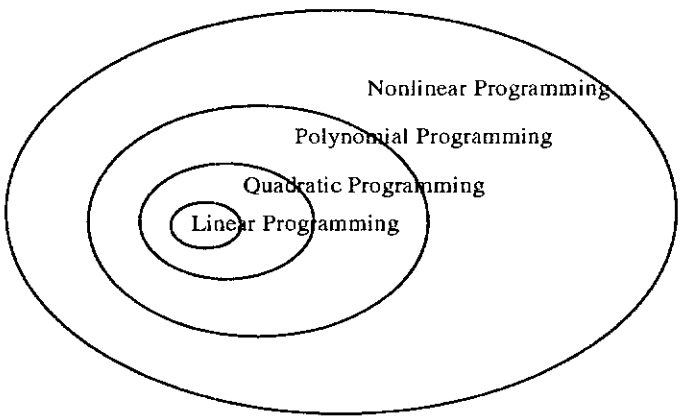

Figure 1: Mathematical Programming Hierarchy

A polynomial program consists of a polynomial objective function and a set of polynomial constraint equations. In general, it is not a convex problem due to the nonlinear terms that may arise in the objective and/or constraints. In this paper, we assume that a polynomial programming (PP) problem is expressed in the following form.

$$\begin{aligned} &minimize && f(x) \\ &subject\ to: && P(x) = 0 \end{aligned}$$

where x is an n-dimensional vector, $f(.)$ is a scalar function, and $P(x)$ is a system of m polynomial equa-

*Research supported by National Science Foundation Grant MIP 92-18715 and Joint Services Electronic Program Contract JSEP N00014-90-J-1270.

Proc. Computer Software and Applications Conference, Nov. 1994.

tions.

In this paper, we propose a systematic procedure for determining global optimal solutions for polynomial programs. Without loss of generality, we assume that minimal solutions are to be found. We first transform a polynomial program into a Lagrangian problem. Based on the notion of Groebner basis, we then reduce the Lagrangian problem to a canonical form by symbolic substitution and simplification. Finally, we use numerical back substitution to locate all extreme solutions in the reduced Lagrangian problem. The idea in this algorithm is to preserve all the extrema by maintaining the algebraic structure of these extrema while performing symbolic reduction. To ensure that a global solution can be obtained, the numerical isolation of global optimal solution is delayed until the mathematical form becomes robust enough to tolerate numerical errors.

2 Modeling Power of Polynomial Programming

Polynomial programs have a wide spectrum of applications. Many examples can be found in [1, 2]. A typical chemical equilibrium system can be modeled by ten to twenty equations in ten to twenty unknowns. In robotics, a six-joint robot can involve a polynomial program with eighteen equations in twelve unknowns. Multi-start optimization methods have been used extensively to obtain better solutions among possibly many local minima.

Polynomial programming generalizes linear and quadratic programming and can model engineering applications that are expressed by polynomial equations. For problems with transcendental terms such as *sin*, *log*, and radicals, Taylor series expansion can be used to reformulate the problem into a polynomial program.

¿From a theoretical aspect, PP can be shown to model *Integer Linear Programming problems* (ILP). Consider a binary integer linear program (BILP):

$$\begin{aligned} &\textit{minimize} && c^T x \\ &\textit{subject to}: && Ax = b \end{aligned}$$

where x is a binary number, A is an m-by-n matrix, b is an m vector, and c is an n-vector. The equivalent PP formulation of the BILP is as follows.

$$\begin{aligned} &\textit{minimize} && c^T x \\ &\textit{subject to}: && Ax = b \\ & && Diag[x]\, Diag[x] = Diag[x]\,, \end{aligned}$$

where $Diag[x]$ is the diagonal matrix with the value of x in the leading diagonal, and 0 elsewhere.

As a last example, we show the representation of a 3-SATISFIABILITY (3-SAT) problem by a system of polynomial equations. Although a polynomial system at most amounts to the constraint set in a polynomial program, it is still as difficult as its optimization version with a polynomial objective function. We illustrate the following problem conversion by considering a 3-SAT example in conjunctive normal form (CNF).

$$(\bar{x_1} \vee x_2 \vee x_4) \wedge (x_1 \vee \bar{x_3} \vee x_4)$$

The problem is to either find a satisfying assignment for the above logical expression if it is satisfiable or disprove its satisfiability. There are various ways to transform the problem; an example of an equivalent polynomial system is as follows.

$$\begin{aligned} x_1(x_2 - 1)(x_4 - 1) &= 0 \\ (x_1 - 1)x_3(x_4 - 1) &= 0 \end{aligned}$$

In this transformation, we rewrite each clause in the Boolean expression into a third-degree polynomial equation by the following rules:

- a positive literal l is replaced by $(l - 1)$;
- a negated literal $\bar{l}$ is replaced by l; and
- $\vee$ is replaced by $*$.

The 3-SAT problem is satisfiable if and only if the polynomial system is consistent.

3 Global Optimization of Polynomial Programs

Numerical optimization algorithms existed to date mainly rely on differential information such as gradients, Hessian, or their tailored modifications, to make progress iteratively. Unfortunately, differential information, no matter what form it takes, only provides local geometric information in the vicinity of the current position. Lacking a view of the entire solution space, almost all such algorithms are designed to find only local solutions.

In this paper, we introduce the notion of Groebner basis as a useful method for finding optimal solutions in polynomial programming. To apply Groebner basis theory, we need to consider the Lagrangian problem associated with the original polynomial program. The

Lagrangian for a polynomial program is defined as follows.

$$L(x, \lambda) = f(x) + \lambda^T P(x) ,$$

where λ is the vector of Lagrange multipliers. Differentiating this expression with respect to x and λ, we obtain the corresponding Lagrangian Problem that finds x and λ such that

$$\nabla L(x, \lambda) = 0 .$$

The Lagrangian problem gives the first-order necessary condition for optimality. All the global and local extrema for the original optimization problems are solutions to the Lagrangian problem. Hence, if one can solve the Lagrangian problem and obtain all its solutions, then obtaining the global minimum becomes a more manageable task.

The Lagrangian problem associated with a polynomial program consists of a system of polynomial equations. More often than not, the polynomial system is strongly coupled in the sense that any solution to the entire system cannot start with a single equation followed by back substitution to the others. We need a systematic procedure to simplify the polynomial system so that after simplification we can obtain the roots without much difficulty. We know that Gaussian elimination serves this purpose for a linear system. Buchberger carried this procedure one step further to extend Gaussian elimination to polynomial systems [3].

Groebner Basis. A Groebner basis for a polynomial system is a canonical form that represents the original system. It can be shown that a Groebner basis contains the same information as the original set of polynomials, although it may contain more polynomials than the original set. However, through decoupling procedures, Groebner basis reduces the original problem to a form that allows us to solve the reduced set of equations by back substitution, which is extremely difficult to do in the original problem. The following properties shows the usefulness of Groebner Bases.

Properties of Groebner Basis

1. Every polynomial system has a Groebner basis.
2. Two polynomial systems are equivalent if and only if they have the same Groebner basis.
3. A polynomial system is inconsistent if and only if its Groebner basis contains a non-zero constant.
4. Groebner bases are finite.

These results have been stated more precisely by MacCallum and Wright (1991) [4] and Buchberger (1983) [3]. These results show that Groebner bases always exist, are canonical and finite, and are able to derive inconsistencies if the original set of equations are inconsistent.

The algorithm for constructing Groebner bases is due to Buchberger (1965) [5]. We do not intend to state it in a rigorous and formal fashion. Instead, we give the following pseudo code.

Groebner Basis Construction Algorithm

Given P: a polynomial system consisting of $p_1, p_2, \cdots p_m$, $G = \{P\}$

```
While ( Not Completely Reduced )
    Select p_i, p_j ∈ G
    h_{i,j} = LCM(PT(p_i), PT(p_j))
    s_{i,j} = h_{i,j}/PT(p_i) p_i − h_{i,j}/PT(p_j) p_j
    If (s_{i,j} ≠ 0) Then G = G ∪ {s_{i,j}}
}
```

where $PT(p_i)$ stands for the Principal Term of polynomial equation p_i, and LCM stands for least common multiple.

The Groebner basis G is obtained and the polynomial system completely reduced if and only if $s_{i,j}$ reduces to zero for every pair of polynomials p_i and p_j in G. If any $s_{i,j}$ does not reduce to zero, then it is included in G, and the test is repeated. This is the essence of the algorithm used to construct a Groebner basis. Principal terms are defined with respect to an ordering that is lexicographic or inverse lexicographic. Lazard (1983) [6] has shown that lexicographic orderings lead to better bounds on the degree of the Groebner basis obtained. We see in this algorithm that $s_{i,j}$ is defined such that principal terms p_i and p_j are cancelled in $s_{i,j}$.

The algorithm always terminates and gives the Groebner basis for the original polynomial system. However, its computational complexity is still an open research problem.

Global Optimization of Polynomial Programs. Based on Groebner bases, a global optimization algorithm can be readily constructed as follows.

1. Generate $L(x, \lambda)$.
2. Generate $\nabla L(x, \lambda) = 0$ using symbolic differentiation to obtain the Lagrangian formulation.
3. Apply Buchberger's Algorithm to $\nabla L(x, \lambda) = 0$.

4. Back-substitute by using a numerical root finder.
5. Compute $f(x)$ for each set of roots.
6. Sort the values of $f(x)$ obtained in increasing order.
7. Check the second-order necessary condition until the first minimum is found.

The first three steps need to be carried out by symbolic computation so that the algebraic structure of the optimization problem is maintained in spite of the transformations and reductions made. The resulting Groebner basis is much easier to solve than $\nabla L(x, \lambda) = 0$ because of the minimal degree of coupling achieved. In essence, Groebner basis reduction produces a triangular form of the system in which the last equation can be "trivially" solved, and the remainder of the triangular system, iteratively processed by "back-substitution." Since the optimality of Lagrangian problems is based on first-order differential equations, the resulting solution may contain a mix of minima and maxima, and the second-order differential of the Lagrangian condition has to be checked to ensure that the final extremum obtained is indeed a minimum.

4 Illustrative Examples

In this section, we demonstrate the ability of the optimization algorithm presented in the last section to find global optima for general polynomial programs. The first two problems are adapted from Nering and Tucker (1993) [7] and Barbeau (1989) [8], respectively. The third problem is a famous problem in control theory, namely, the trust-region problem. In the last problem, we show how a general nonlinear program can be approximated by a polynomial program via Taylor series expansion and then solved by our algorithm.

Polynomial Program 1 .

$$\begin{aligned} \textit{minimize} \quad & f(x_1, x_2, x_3) = x_1 {x_2}^2 x_3 \\ \textit{subject to}: \quad & -6 + {x_1}^2 + {x_2}^2 + {x_3}^2 = 0 \\ & x_1 + x_2 - x_3 = 0 \end{aligned}$$

Since there are two equality constraints, two Lagrange multipliers, λ_1 and λ_2, are introduced to formulate the Lagrangian problem. To locate all the extrema, the resulting Lagrangian problem will involve solving a system of polynomial equations, $\nabla L(x_1, x_2, x_3, \lambda_1, \lambda_2) = 0$ or explicitly expressed as

$$\begin{aligned} \lambda_2 + 2\lambda_1 x_1 + {x_2}^2 x_3 &= 0 \\ \lambda_2 + 2\lambda_1 x_2 + 2 x_1 x_2 x_3 &= 0 \\ -\lambda_2 + x_1 {x_2}^2 + 2\lambda_1 x_3 &= 0 \\ -6 + {x_1}^2 + {x_2}^2 + {x_3}^2 &= 0 \\ x_1 + x_2 - x_3 &= 0 \end{aligned}$$

Buchberger's algorithm is then applied to perform as much decoupling as possible. One step of Buchberger's algorithm is carried out here to illustrate the algorithm. Consider the first two equations in Iteration one. Their principal terms are ${x_2}^2 x_3$ and $2 x_1 x_2 x_3$, respectively. $h_{1,2}$, the LCM of this pair of principal terms, is $2 x_1 {x_2}^2 x_3$. Hence, $s_{1,2}$, the new polynomial to be added into the basis, is $2 x_1 (\lambda_2 + 2\lambda_1 x_1 + {x_2}^2 x_3) - x_2 (\lambda_2 + 2\lambda_1 x_2 + 2 x_1 x_2 x_3)$, or $2 x_1 \lambda_2 + 4\lambda_1 {x_1}^2 - x_2 \lambda_2 - 2\lambda_1 {x_2}^2$ after simplification. Through such iterative reductions, the Lagrangian problem can then be simplified to the following set of equations.

$$\begin{aligned} 12 x_2 - 11 {x_2}^3 + 2 {x_2}^5 &= 0 \\ x_2 \left(-6 + 4 {x_2}^2\right) x_3 &= {x_2}^2 \left(-3 + 2 {x_2}^2\right) \\ -(x_2 x_3) + {x_3}^2 &= 3 - {x_2}^2 \\ x_1 &= -x_2 + x_3 \\ 3\lambda_1 &= {x_2}^2 \left(-3 + {x_2}^2\right) \\ 3\lambda_2 &= x_2 \left(-6 + {x_2}^2\right) \end{aligned}$$

We see that these equations bear a triangular form that can readily be solved by back substitution. After a sequence of numerical computations as specified in the global optimization procedure, we obtain six solutions to the Lagrangian problem that evaluate f to $\{-4.0, -4.0, 0, 0, 2.25, 2.25, 2.25, 2.25\}$. The minimum of f is simply -4 with (x_1, x_2, x_3) equal to $(-1,2,1)$ or $(1,-2,-1)$.

Polynomial Program 2 .

$$\begin{aligned} \textit{maximize} \quad & f(x_1, x_2) = x_1 + {x_2}^2 \\ \textit{subject to}: \quad & \frac{-9 x_1}{8} + {x_1}^4 + {x_2}^2 - x_1 {x_2}^3 = 0 \\ & {x_1}^2 - \frac{9 x_2}{8} - {x_1}^3 x_2 + {x_2}^4 = 0 \end{aligned}$$

Applying the global optimization procedure, f can take values from $\{0., 1.25, 1.5, 2.39062\}$, giving 2.39062 as the maximum when $x_1 = x_2 = 1.125$. We have observed that the number of solutions to the Lagrangian problem is ten if complex roots are also taken into account. Since complex roots are not relevant in optimization problems, they are simply ignored.

Polynomial Program 3 . Given an n-by-n symmetric matrix A, an n-vector b, and a scalar α, the following polynomial program

$$\begin{aligned} &\textit{minimize} \quad x^T Ax + b^T x \\ &\textit{subject to}: \quad ||x||_2^2 = \sum_{i=1}^{n} x_i^2 = \alpha^2 \end{aligned}$$

is the Trust-Region Problem studied by Forsythe and Golub (1965) [9], Spjoetvoll (1972) [10], and Lyle and Szularz (1994) [11]. It is equivalent to minimizing a second-degree polynomial equation in the unit sphere. Historically, eigensystems are analyzed to solve this problem numerically. Since this problem can be conveniently parameterized, it serves as a good tool to study the complexity of solving general polynomial programs.

Using the global optimization procedure, we conducted a series of experiments to evaluate the performance of our algorithm for this class of nonlinear programs with larger sizes. The CPU time required to obtain the Groebner bases is collected for randomly generated problems using Mathematica running on a Sun Sparc 10/30 workstation.

Trust Region Problem			
I.D.	size (n)	Groebner reduction	root finding
trp9	9	15.6 s	19.8 s
trp11	11	44.7 s	18.8 s
trp13	13	181.9 s	50.1 s
trp15	15	101.2 s	739.9 s
trp17	17	331.7 s	20533 s

We see the algebraic reduction only takes a limited amount of time and is reasonable when one needs to find the global optimal solution. The actual run time for solving these problems will be longer if we take into account numerical steps following symbolic simplification.

Polynomial Program 4

$$min \;\; 2x_1^2 + 2x_2^2 - cos(x_1 + x_2) - cos(x_1 - x_2) + 2$$

The problem has a global minimum 0 at (x_1, x_2)=$(0, 0)$ and an infinite number of local minima. It would be very difficult to obtain the global optimum using common numerical optimization methods unless the starting point is very close to the origin. Our global optimization method does not rely on good starting points. To apply our algorithm, transcendental terms have to be approximated by their Taylor series expansions so that a polynomial program can be formulated. Expanding the power series to the 7-th order term, we have

$$\begin{aligned} \textit{minimize} \quad & 3\,{x_1}^2 - \frac{{x_1}^4}{12} + \frac{{x_1}^6}{360} + 3\,{x_2}^2 - \frac{{x_1}^2 {x_2}^2}{2} + \\ & \frac{{x_1}^4 {x_2}^2}{24} - \frac{{x_2}^4}{12} + \frac{{x_1}^2 {x_2}^4}{24} + \frac{{x_2}^6}{360} \end{aligned}$$

With little difficulty, we are able to find a good approximate global solution. It takes only 0.98 seconds to obtain the Groebner basis followed by 0.7 seconds in finding the minimum. The approximate solution found is $(x_1, x_2) = (0.088185, 0.088185)$, which gives 0.0469053 for the objective value. In general, postprocessing by numerical-descent methods can be used to obtain the true optimal solution from the approximate global solution. Note that the latter step depends on the accuracy in representing the original nonlinear function by Taylor series expansion. The accuracy required is likely to be problem-dependent.

5 Conclusions

In this paper, we have identified a class of nonlinear constrained optimization problems called polynomial programming problems. This class of problems can be used to model many real-world applications as well as theoretically interesting problems. Based on Lagrange formulations, we propose a deterministic global optimization procedure for finding optimal solutions of a polynomial program. Our algorithm employs both symbolic and numeric computations; the symbolic part concerns the generation of Groebner bases as a canonical form for Lagrange formulations. We find that Groebner Bases can remove as much coupling as possible in the Lagrange formulations, resulting a triangular system of equations that can be solved by numerical back substitution. Finally, we illustrate our algorithm on some interesting example problems.

References

[1] C. A. Floudas and P. M. Pardalos, "A collection of test problems for constrained global optimization algorithms," in *Lecture Notes in Computer Science* (G. Goos and J. Hartmanis, eds.), Springer-Verlag, 1990.

[2] A. Morgan, *Solving Polynomial Systems Using Continuation for Engineering and Scientific Problems.* Prentice-Hall, 1987.

[3] B. Buchberger, "A note on the complexity of construction groebner-bases," in *Lecture Notes in Computer Science*, pp. 137–145, Springer-Verlag, 1983.

[4] M. MacCallum and F. Wright, *Algebraic Computing with Reduce*. Oxford University Press, 1991.

[5] B. Buchberger, *An algorithm for finding a basis for the residue class ring of a zero-dimensional polynomial ideal*. PhD thesis, Univ. of Innsbruck, Austria: Math. Inst., 1965.

[6] D. Lazard, "Groebner bases, gaussian elimination and resolution of systems of algebraic equations," in *Lecture Notes in Computer Science*, pp. 146–156, Springer-Verlag, 1983.

[7] E. D. Nering and A. W. Tucker, *Linear Programs and Related Problems*. Academic Press, 1993.

[8] E. J. Barbeau, *Polynomials*. Springer-Verlag, 1989.

[9] G. E. Forsythe and G. H. Golub, "On the stationary values of a second degree polynomial on the unit sphere," *SIAM J. on Applied Mathematics*, vol. 13, pp. 1050–1068, 1965.

[10] E. Spjoetvoll, "A note on a theorem of forsythe and golub," *SIAM J. on Applied Mathematics*, vol. 23, no. 3, pp. 307–311, 1972.

[11] S. Lyle and M. Szularz, "Local minima of the trust region problem," *Journal of Optimization Theory and Applications*, vol. 80, January 1994.