Polynomial Algorithms for Optimization Problems

Kei-ichi Shiraishi, Hiroshi Kai, Tomokatsu Saito[*] and Matu-Tarow Noda
Department of Computer Science, Ehime University
[*] Department of Mathematics, Sophia University

Abstract

Optimization problems discussed here are expressed as

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \left\{ \begin{array}{ll} g_j(\boldsymbol{x}) \leq 0 & (j = 1, 2, \ldots, m) \\ h_k(\boldsymbol{x}) = 0 & (k = 1, 2, \ldots, l). \end{array} \right. \end{array}$$

The problem of maximizing a function is similarly formulated. Such problems may be solved by floating point computation in usual mathematical programming, but there are some difficulties as shown below.

1. It is difficult to obtain the global minimum, whereas it is easy to obtain the local minima.
2. The computational cost for obtaining the local minima largely depends on the initial value.

Thus, in this paper, we take an algebraic approach to solving the problem. The (objective) function and constraints are changed by a set of polynomial equations by adding *slack* variables. The set of polynomial equations is algebraically solved by a Gröbner basis and reduced to a univariate polynomial (algebraic) equation. In the reduction process, a term–ordering of variables become important for the Gröbner basis computation. Results by two types of the ordering, the lexicographic (Lex) ordering and the reverse lexicographic (DegRevLex and abbreviate simply DRL) ordering, are discussed. Both ordering gives finally not ill–conditioned algebraic equation. The resulting equation is solved by the numerical Durand–Kerner method and compared with solutions by algebraic Sturm's theorem.

1 Introduction

Finding the global minimum is one of the central research subjects of mathematical programming problems such as

$$\begin{array}{ll} \text{minimize} & f(\boldsymbol{x}) \\ \text{subject to} & \left\{ \begin{array}{ll} g_j(\boldsymbol{x}) \leq 0 & (j = 1, 2, \ldots, m) \\ h_k(\boldsymbol{x}) = 0 & (k = 1, 2, \ldots, l). \end{array} \right. \end{array} \tag{1}$$

The problem of maximizing a function is similarly formulated. We call both of them simply an *optimization* problem. If the objective function f and constraints g_j, h_k are linear, the problem is called a linear programming problem and solved by the simplex method. This method is satisfactory for almost all cases. If the objective function and/or constraints are not linear (non–linear programming problems), the problem is usually solved by Newton's iterative method and its extensions.

However, several difficulties occur shown as follows:

1. It is difficult to obtain the global minimum (optimal solution) , whereas it is easy to obtain the local minima.

2. The computational cost for obtaining the local minima largely depends on the initial value.

3. It is impossible for usual computational methods to give the global minimum which doesn't satisfy the Kuhn–Tucker condition.

4. In some cases, results are instable by numeric errors of floating point arithmetic.

Thus, in this paper, we take an algebraic approach to solving the problem. The objective function and constraints are changed by a set of polynomial equations by adding *slack* variables. The set of polynomial equations is algebraically solved by a Gröbner basis[6] and Sturm's theorem[7]. In the Gröbner basis computation, we consider two types of term–ordering of variables, i.e. the lexicographic (Lex) ordering and the reverse lexicographic (DegRevLex and abbreviate simply DRL [1]) ordering, and compare the effectiveness of them. Both ordering gives finally not ill–conditioned algebraic equation. Thus, the resulting algebraic equation is solved by algebraic Sturm's theorem and are compared with the numeric Durand–Kerner(DK) method. The DK–method is a floating point computation. If an input algebraic equation is erroneous, results by the DK–method may be unreliable. However, in this case, the input equation is obtained by the Gröbner basis computation and further not ill–conditioned. Thus, we can expect our numerical computations to give accurate and reliable results. We show that the results of our method are free from the above difficulties by using several examples. Though it is hard to find the optimal solution, the global minimum, by usual method, our method gives satisfactory results for them. Both the linear and a kind of non–linear programming problems are solved by the same algebraic method. A similar method has been proposed by Weispfenning and Xue [5] and applied to an integer programming problem with linear, quadratic, fractional linear functions and linear constraints. The method is based on the quantifier elimination method [4]. Most of the discussions in [4] and [5] are devoted to complexity bounds of the method. Another approach to the same problem has been shown by Wu Tian-jiao[8]. The problem is converted to a set of polynomial equations by using Wu's method[9]. However, coefficients of polynomials in examples shown of [8] are restricted to integers.

2 Algebraic approach to mathematical programming problems

The mathematical programming (optimization) problem, (1) is solved by using numerical Newton's method or its extensions, such as the steepest descent method, the penalty method and others. If we give an initial value, these methods give a local minimum rapidly without any trouble. Generally, we must consider problems which have more than one local minima. Thus, to find the global minimum, we must compute the same problem with different initial values repeatedly. It becomes very difficult to find the global minimum by the above numeric methods.

Here, we consider the problem (1) whose objective function $f(\boldsymbol{x})$ and constraints $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ are expressed as polynomials. Thus, these are real valued functions which are defined in $\boldsymbol{x} \in \mathbf{R}^n$ and satisfy $f(\boldsymbol{x}), g_j(\boldsymbol{x})$, $h_k(\boldsymbol{x}) \in \mathbf{Q}[\boldsymbol{x}]$ for $j = 1, \ldots, m, k = 1, \ldots, l$. Constraints

$$\{\boldsymbol{x} \in \mathbf{R}^n \mid g_j(\boldsymbol{x}) \le 0 (j = 1, \ldots, m), h_k(\boldsymbol{x}) = 0 (k = 1, \ldots, l)\} \tag{2}$$

are decomposed into two parts, an interior domain D and a boundary C as

$$\begin{aligned} \mathrm{D} &= \{\boldsymbol{x} \in \mathbf{R}^n \mid g_j(\boldsymbol{x}) < 0 (j = 1, \ldots, m), h_k(\boldsymbol{x}) = 0 (k = 1, \ldots, l)\} && (3) \\ \mathrm{C} &= \mathrm{C}_1 \cup \mathrm{C}_2 \cup \cdots \cup \mathrm{C}_m, && (4) \end{aligned}$$

where C_i is represented as

$$\{\boldsymbol{x} \in \mathbf{R}^n \mid g_i(\boldsymbol{x}) = 0, g_j(\boldsymbol{x}) \le 0, h_k(\boldsymbol{x}) = 0\},$$

where $j = 1, \ldots, i-1, i+1, \ldots, m$ and $k = 1, \ldots, l$.

Our algebraic method consists of three steps as follows:

1. Find all the local minima in D.
2. Find all the local minima in C.
3. Compare values of all local minima and find the global minimum, i.e., optimal solution.

In the last step, results of all local minima we evaluated and compared by high precision accurate computations. Here, for the simplicity, we use double precision number for comparisons. The algebraic method is applied to the first and the second steps. We show detailed descriptions of these steps and methods of how to solve an algebraic equation below.

The case : the optimal solution is in D

The problem (1) is rewritten as

$$\begin{array}{rl} \text{objective function} & f(\boldsymbol{x}) \longrightarrow \text{minimize (optimize)} \\ \text{constraints} & \left\{ \begin{array}{ll} g_j(\boldsymbol{x}) + \lambda_j = 0 & (j = 1, 2, \ldots, m) \\ h_k(\boldsymbol{x}) = 0 & (k = 1, 2, \ldots, l), \end{array} \right. \end{array} \tag{5}$$

by adding slack variables $\lambda_i \geq 0, i = 1, 2, \ldots, m$. Thus, the necessary condition of the optimal solution , i.e., the local minima, is

$$\left\{ \begin{array}{ll} \frac{\partial f(\boldsymbol{x})}{\partial x_i} & = 0, \quad (i = 1, 2, \ldots, n) \\ g_j(\boldsymbol{x}) + \lambda_j & = 0, \quad (j = 1, 2, \ldots, m) \\ h_k(\boldsymbol{x}) & = 0, \quad (k = 1, 2, \ldots, l). \end{array} \right. \tag{6}$$

For the simplicity, we assume that the ideal generated from a set of polynomials that are left hand side of (6) is 0-dimensional. The Gröbner basis for a set of polynomial equations, (6), is computed. Here, we consider two kinds of *ordering* used in the Gröbner basis computation. The first is the lexicographic ordering of variables. Then (6) is reduced to a univariate algebraic equation. Since the resulting equation has no multiple root factor, it can be solved accurately by the algebraic Sturm's theorem or by the numeric DK–method. The other ordering is the reverse–lexicographic ordering. The basis by this ordering is obtained quickly but it doesn't, in general, contain a univariate polynomial. The basis is called the *shape basis.* Thus in this case, a minimal (characteristic) polynomial [3] should be computed from the shape basis. The minimal polynomial is also a univariate polynomial and has no multiple root factors.

The case : the optimal solution is in C

The procedure is similar to the above case except for the number of variables. Since we consider the solution on the boundary, we put some of all the variables, $x_1, x_2, \cdots, x_n, \lambda_1, \lambda_2, \cdots, \lambda_m$ as zero. Thus, the Gröbner basis should be computed

$$\binom{n+m}{1} + \binom{n+m}{2} + \cdots + \binom{n+m}{n}$$

times repeatedly.

How to solve an algebraic equation

As mentioned above, we can obtain two kinds of univariate algebraic equations by the reduction of the problem, (6). The two univariate equations are not the same. The degree of the equation obtained by the Gröbner basis computation with the lexicographic ordering is generally higher than the minimal polynomial. Thus, we solve the minimal polynomial obtained by the Gröbner basis computation with the reverse–lexicographic ordering. The minimal polynomial is solved by Sturm's theorem and by the DK–method. If the algebraic equation contains multiple root factors, i.e., ill–conditioned, the numerical method gives unreliable results. But the minimal polynomial has no multiple root factors, therefore the numeric method gives satisfactory results for almost all the problems.

3 Applications of the algebraic method

We show several simple examples of optimization problems. They show how our algebraic method gives correct results for the problems. Properties of the examples are as follows:

1. A problem whose global minimum is placed *behind* the local minima. If we use numeric methods, the results may show only the local minima. Thus, it is hard to find the global minimum in this case.
2. A minimization problem shown in [2].
3. A problem which doesn't satisfy the Kuhn–Tucker condition.

In the following tables, we show computation times for

- Obtaining Gröbner basis with term–ordering, the lexicographic ordering (Lex) and the reverse lexicographic ordering (DRL)

- Solving resulting univariate algebraic equation by using the Sturm's theorem symbolically (Sturm).
- Solving resulting univariate algebraic equation by using the DK–method numerically (DK).
- Computing a minimal polynomial for DRL case (minP).

Further, a total time by each computation is added in the tables. The total time contains computation times needed above symbolic and/or numeric computations and other times, combine solutions, and others. In tables 1–3, results are computed by the computer algebra system Risa/Asir on Pentium 120MHz computer with 38.6MB RAM memory and shown in sec.

Example 1

$$\begin{array}{rl} \text{objective function} & f = x_1(x_1-1)(x_1-2)(x_1-3.1) \\ & \quad +x_2(x_2-1)(x_2-2)(x_2-3.1) \longrightarrow \text{minimize} \\ \text{constraints} & \left\{ \begin{array}{ll} g_1 = x_1 - 3 & \leq 0 \\ g_2 = x_2 - 3 & \leq 0 \\ x_1, x_2 & \geq 0 \end{array} \right. \end{array} \tag{7}$$

We consider one of our method, DRL+minP+DK. By the method, the problem is solved as follows:

The case : the optimal solution is in $\mathbf{D}$

1. Introduce slack variables $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ to constraints as

$$\left\{ \begin{array}{ll} x_1 - 3 + \lambda_1 & = 0 \\ x_2 - 3 + \lambda_2 & = 0 \end{array} \right.$$

2. The necessary conditions of the optimal solution are

$$\left\{ \begin{array}{lll} \frac{\partial f}{x_1} & = 4x_1^3 - \frac{183}{10}x_1^2 + \frac{113}{5}x_1 - \frac{31}{5} & = 0 \\ \frac{\partial f}{x_2} & = 4x_2^3 - \frac{183}{10}x_2^2 + \frac{113}{5}x_2 - \frac{31}{5} & = 0 \\ \text{constraint} & x_1 + \lambda_1 - 3 & = 0 \\ \text{constraint} & x_2 + \lambda_2 - 3 & = 0 \end{array} \right. \tag{8}$$

3. Obtain Gröbner basis of (8) with DRL term ordering

$$\left\{ \begin{array}{l} x_2 + \lambda_2 - 3, -40\lambda_2^3 + 177\lambda_2^2 - 208\lambda_2 + 49, \\ x_1 + \lambda_1 - 3, -40\lambda_1^3 + 177\lambda_1^2 - 208\lambda_1 + 49 \end{array} \right.$$

4. Compute minimum polynomials from the Gröbner basis as

$$40x_1^3 - 183x_1^2 + 226x_1 - 62, 40x_2^3 - 183x_2^2 + 226x_2 - 62, \cdots$$

5. Obtain zeros of the minimum polynomial by the DK–method as

$$x_1 = 2.6869, 1.5047, 0.38338$$

6. Substitute the result of x_1 to another Gröbner basis and obtain all solutions. Among the solutions, a solution which satisfy conditions $x_1, x_2, \lambda_1, \lambda_2 \geq 0$ and gives the minimum value of the objective function f is obtained as

$$\left\{ \begin{array}{rl} (x_1, x_2, \lambda_1, \lambda_2) & = (2.6869, 2.6869, 0.31309, 0.31309) \\ f & = -2.5723 \end{array} \right. \tag{9}$$

The case : the optimal solution is in C

1. For $x_1 = 0$, substitute $x_1 = 0$ to the problem and repeat the above computation. A candidate of the optimal solution is

$$(x_1, x_2, \lambda_1, \lambda_2 \text{and } f) = (0, 2.6869, 3, 0.31309, \text{and} - 1.2861) \tag{10}$$

2. For $x_2 = 0$, we obtain another candidate as

$$(x_1, x_2, \lambda_1, \lambda_2 \text{and } f) = (2.6869, 0, 0.31309, 3, \text{and} - 1.2861) \tag{11}$$

3. By the same computations, we obtain

$$\text{for } \lambda_1 = 0 \to f = -1.8861, \quad \text{and for } \lambda_2 = 0 \to f = -1.8861 \tag{12}$$

4. Then put two variables to be zero, and compute the same procedures. Results are

$$\left\{ \begin{array}{ll} x_1 = x_2 = 0 \to f = 0, & x_1 = \lambda_2 = 0 \to f = -0.6 \\ x_2 = \lambda_1 = 0 \to f = -0.6, & \lambda_1 = \lambda_2 = 0 \to f = -1.2 \end{array} \right. \tag{13}$$

Compare all candidates and find the optimal solution

1. Obtain the minimum value of f from all candidates (9) $\sim$ (13). In the case, the candidate (9) is the minimum and the optimal solution.

2. Computation times by several methods are shown in the following table.

	Gröbner	Sturm	DK	minP	total
Lex+Sturm	0.035860	0.21346	——	——	0.38974
Lex+DK	0.033726	——	0.032343	——	0.20666
DRL+minP+Sturm	0.034419	0.27785	——	0.41743	1.0516
DRL+minP+DK	0.037278	——	0.023801	0.42321	0.79224

Table.1 Computation time(sec.) of the example 1.

Example 2

$$\begin{array}{rl} \text{objective function} & f = (10(x_2 - x_1^2))^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ & \quad +10(x_2 + x_4 - 2)^2 + \frac{1}{10}(x_2 - x_4)^2 \longrightarrow \text{minimize} \\ \text{constraints} & x_1, x_2, x_3, x_4 \geq 0 \end{array} \tag{14}$$

Four types computations give the following results.

	Gröbner	Sturm	DK	minP	total
Lex+Sturm	0.44686	0.50500	——	——	1.1862
Lex+DK	0.45207	——	0.06280	——	0.7463
DRL+minP+Sturm	0.11677	0.78600	——	1.0815	2.4352
DRL+minP+DK	0.11669	——	0.10503	1.0687	1.7086

Table.2 Computation time(sec.) of the example 2.

The numeric computation of DK–method gives results quickly than the algebraic computation based on the Sturm theorem. In the method "DRL+minP+DK", the computation time depends strongly on the computation time for obtaining the minimal polynomial. Thus, if a procedure for the minimal polynomial will be improved, the method "DRL+minP+DK" will be a good method. The optimal solution, $f = 0$ for $(x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$, is obtained accurately.

Example 3

$$\begin{array}{rl} \text{objective function} & f = (x_1 - 2)^2 + x_2^2 \longrightarrow \text{minimize} \\ \text{constraints} & \begin{cases} g_1 = -(1 - x_1)^3 + x_2 & \leq 0 \\ x_1, x_2 & \geq 0 \end{cases} \end{array} \tag{15}$$

The problem doesn't satisfy the Kuhn–Tucker condition. Thus, it is hard to obtain the optimal solution by usual method. But, our method gives the correct result as $f = 1$ for $(x_1, x_2) = (1, 0)$.

	Gröbner	Sturm	DK	minP	total
Lex+Sturm	0.034947	0.025071	——	——	0.11928
Lex+DK	0.030616	——	0.0089112	——	0.10072
DRL+minP+Sturm	0.021885	0.031961	——	0.16523	0.29938
DRL+minP+DK	0.021285	——	0.011445	0.15938	0.26932

Table.3 Computation time(sec.) of the example 3.

4 Conclusions

Here, we propose an algebraic method to the mathematical programming problem and show the efficiency of the method. We compared several algebraic methods from viewpoints of the computational time and the memory requirement of the computations. Finally, we show the best method which is based mainly on algebraic computation. In the last step of the method, the resulting univariate algebraic equation is solved numerically by the Durand–Kerner method. The method is shown briefly as follows:

1. Change the problem to a set of polynomial equations by adding slack variables.
2. Compute the Gröbner basis of the set of polynomials with the reverse lexicographic order.
3. Compute the minimal polynomial algebraically.
4. Solve the resulting univariate algebraic equation by the numerical DK–method.

Through several examples, we show that the method can be applied to optimization problems and also to certain kinds geometrical theorem proving problems. Optimization problems which are hard to solve by the numeric methods such as Newton's method, can be solved satisfactory by our method.

The following problems remain for our future studies.

1. Obtain theoretical complexity bounds of the method.
2. Apply the method to the geometrical theorem proving problems.
3. Apply the method to large sized optimization problem and examine the effectiveness of the method.

4. How to apply the method to optimization problems whose objective function and constraints are not polynomials?

The last problem is especially important for our future advance. The problem may be reduced to find a good interpolation method of discrete data to polynomials. Further, our algebraic computation method can be applied to problems of geometrical theorem proving.

References

[1] P. Conti and C. Traverso, Buchberger algorithm and integer programming, in "Applied Algebra, Algebraic Algorithms and Error–Correcting Codes" *Eds. H.F. Mattson et.al.*, pp.130 – 139, Springer Verlag, 1991.

[2] J.J. Moré, B.S. Garbow and K.E. Hillstrom, Testing Unconstrained Optimization Software, ACM Trans.Math.Softw., Vol.7, pp.17–41, 1981.

[3] M. Noro and K. Yokoyama, Factoring Polynomials over Algebraic Extension Fields, Proc. Risa Consortium *Eds. M.T. Noda, et.al.*, pp.11–34, 1998.

[4] V. Weispfenning, Simulation and Optimization by Quantifier Elimination, J.Symb.Comp., Vol.24, pp.189–208, 1997.

[5] V. Weispfenning and R. Xue, Parametric Mixed Integer Programming by Elimination, Technical Report MIP–9503, Universität Passau, 1995.

[6] K. O. Geddes, S. R. Czapor and G. Labahn, Algorithms for Computer Algebra, Kluwer Academic Publishers, 1992.

[7] J. H. Davenport, Y. Siret and E. Tournier, Computer Algebra (Systems and algebraic Computation) 1st ed., Academic Press, 1988.

[8] Wu Tian-jiao, Some Test Problems on Applications of Wu's Method in Nonlinear Programming Problems, MM-Res.Preprints, No.6, pp.144–155, 1991.

[9] Wu Wen-tsun, A Mechanization Method of Equations-solving and Theorem-proving, Advances in Computing Research, Vol.6, pp.103–138, 1992.