AD-A046 377 WISCONSIN UNIV MADISON MATHEMATICS RESEARCH CENTER F/G 12/1
SOLVING A SYSTEM OF LINEAR INEQUALITIES FOR GENERATING TEST DAT--ETC(U)
AUG 77 B EINARSSON DAAG29-75-C-0024
UNCLASSIFIED MRC-TSR-1780 NL

1 OF 1
ADA
046377

END
DATE
FILMED
12-77
DDC

9 MRC Technical Summary Report #1780

6 # SOLVING A SYSTEM OF LINEAR INEQUALITIES FOR GENERATING TEST DATA.

10 Bo Einarsson

11 B.S.

Mathematics Research Center
University of Wisconsin-Madison
610 Walnut Street
Madison, Wisconsin 53706

11 August 1977

12 15p.

(Received July 8, 1977)

14 MRC-TSR-1780

15 DAAG29-75-C-0024

Sponsored by

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park
North Carolina 27709

1473

221 200

LB

UNIVERSITY OF WISCONSIN - MADISON
MATHEMATICS RESEARCH CENTER

SOLVING A SYSTEM OF LINEAR INEQUALITIES FOR GENERATING TEST DATA

Bo Einarsson[*]

Technical Summary Report #1780
August 1977

ABSTRACT

A simple inequality solver, based on the simplex method, for determining safe input data corresponding to execution of a certain path of a computer program, is presented. Special consideration is given the influence from rounding errors on the solution.

AMS(MOS) Subject Classification: 65K05

Key Words: simplex method, linear programming, inequality solver, rounding, Data Flow Analysis, software validation, test data generation.

Work Unit Number 8. (Computer Science)

[*] National Defense Research Institute, Box 98, S-147 00 Tumba, Sweden

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

SIGNIFICANCE AND EXPLANATION

Data Flow Analysis can be used to find some of the errors in a computer program, especially those where the value of a not yet defined variable is used, or contrary a variable is assigned a value but not used subsequently. This happens easily if the variable is misspelled.

The Data Flow Analysis gives as output a set of dubious paths of the investigated program, which have to be checked for executability. This can be done by solving a system of inequalities. This report discusses how to obtain a reliable solution of this system in the linear case, when rounding effects are taken into account. The method is based on the simplex algorithm from linear programming, and returns a solution in the middle of the feasible region. The general nonlinear case is much more difficult to handle.

SOLVING A SYSTEM OF LINEAR INEQUALITIES FOR GENERATING TEST DATA

by

Bo Einarsson*

1. Introduction

At the testing or validation of computer programs it is often of interest to obtain the information whether a certain program path is executable or not, and if the path is executable to obtain a set of input data that causes execution of that path.

A system to generate test data, and also to symbolically execute programs, has been given by Clarke (1976). The linear inequality solver she used can give test data that are not certain to produce the required path due to rounding errors. It is the purpose of this note to present a simple variant of that inequality solver, whose reliability is greater.

One application of great interest is to see if a dubious path obtained from a Data Flow Analysis program is executable, see Fosdick and Osterweil (1976), and also Einarsson (1977b).

2. Derivation of the System of Inequalities

The path followed at execution of a computer program will, if the program has only well-defined values of all variables and constants, be completely determined by the input data. The branching will occur at conditional statements of different types, where certain relations between variables are tested. To get input data for a specific path,

*National Defense Research Institute, Box 98, S-147 00 Tumba, Sweden

Sponsored by the United States Army under Contract No. DAAG29-75-C-0024.

that path is followed, and at every branching point the corresponding relation is taken as a constraint on the input variables. The unconditional statements between the branching points are of course also of great importance, since they in many cases give relations between the variables.

Let us look at a short segment of a simple code. Although this example is written in Fortran, the main ideas apply to most languages, see Einarsson (1977b). The letters A, B, C, and D represent real variables.

```
IF (A .GT. B) GOTO 100
IF (C .GT. D) GOTO 200
IF (A .GT. C) GOTO 300
IF (A .GT. D) GOTO 400
```

Transfer of execution is made to
statement number 100 if $A > B$;
statement number 200 if $A \leq B$ and $C > D$;
statement number 300 if $A \leq B$, $C \leq D$, and $A > C$;
statement number 400 if $A \leq B$, $C \leq D$, $A \leq C$, and $A > D$;
and no transfer is made if $A \leq B$, $C \leq D$, $A \leq C$, and $A \leq D$.

In this case we see that transfer cannot be made to statement number 400, since the two conditions $A > D$ and $A \leq C \leq D$ contradict each other. The conditions for no transfer may be reduced to
$A \leq B$ and $A \leq C \leq D$.

It is path defining inequalities like those above we wish to study, and we want to obtain input data for a certain path, or information that it is not executable. A path is especially interesting if Data Flow Analysis has shown it to be dubious, for example that execution of that path requires the value of a variable which has not been defined, or calculates a value which is not used subsequently, see Fosdick and Osterweil (1976).

In almost all cases the relations in the conditional statements are linear. If nonlinearities occur, it is possible to introduce artificial variables into the set of input variables. The same applies to nonlinearities in the non-conditional statements.

Example IF (X .GT. SIN(Y)) GOTO 10

is replaced by

```
        Z = artificial variable
        IF (X .GT. Z) GOTO 10
```

In this case Z is of course restricted to $-1. \leq Z \leq 1.$ and the relation between Y and Z has to be considered at the final part of the solution.

3. Desired Properties of the Solution

If the system is linear, it may be normalized to a linear programming problem, with an arbitrary objective function. It is well known (see for example the book by Dantzig) that the solution of such a maximizing (minimizing) problem is at one of the corners of the convex hypervolume, defined by the hyperplanes represented by the constraints. However, every corner point is satisfying several of the constraints as equalities. Some of these constraints have probably been treated by the symbolic manipulation system, creating the systems of inequalities, in such a way that the round-off properties of these relations have been changed. It might therefore happen that the input values obtained from the inequality solver will not, due to round-off, give the desired path at execution of the program. This effect can be interpreted as if some of the inequalities are slightly unsatisfied. The probability for this to occur is drastically diminished if the linear programming problem solver returns an <u>interior</u> solution instead of the usual one at a boundary point.

Methods for obtaining such interior solutions have been investigated and were reported in Einarsson (1977a). In that paper eight different methods were discussed, but none of them looked very promising. The main disadvantage with all the suggested methods is that they require a lot of extra programming, and in most cases also more computing, except for the recommended one.

Professor M. J. D. Powell, University of Cambridge, has advised me on a method that seems to be both simple and powerful. It is described in detail in the next section, which also includes relevant parts from my earlier report.

4. Solving the System of Linear Inequalities

We restrict here to *linear* constraints. With classical linear programming methods the solution is returned without any theoretical problems. However, linear programming requires an objective function, which is of no primary interest to us. We know that with any *linear* objective function the method will return an extreme point (on the *boundary* of the allowed region). This might mean that the solution returned by the inequality solver (even if it is *forced* to satisfy the set of linear constraints being solved) might still not satisfy the original path constraints of the program under investigation because of rounding effects. This is especially dangerous with real variables being checked for conditions to avoid "binary" conditions such as divide by zero, square root of a negative number, or the arcsine of a quantity exceeding unity. It seems natural to assume that the highest probability of failure is at the extreme points of the region, followed by the boundary, and the lowest probability of failure is in the middle of the region.

The linear constraints problem can be written in normalized form as (assuming $x_i \geq 0$)

(1) $\sum_{i=1}^{N} a_{ji} x_i \leq b_j$ $\quad j = 1,\ldots,M$

N = number of variables
M = number of constraints

and it can be replaced by

(2) $\sum_{i=1}^{N} a_{ji} x_i + s_j = b_j$ $\quad j = 1,\ldots,M$

$s_j \geq 0$ $\quad j = 1,\ldots,M$

with M equations, M new additional constraints $s_j \geq 0$, and N + M unknowns x_i and s_j.

Note 1: Solving the M equations in N unknowns x_i and M unknowns $s_j \geq 0$ is the problem, but in addition we wish to maximize each s_j. Since we do not wish a solution on the boundary of the region, we may change the constraint $s_j \geq 0$ into $s_j > 0$.

Note 2: If the system (1) above is normalized by multiplying each equation with $1/(\sum_{i=1}^{N} a_{ji}^2)^{\frac{1}{2}}$, then the variable s_j, when the individual equation is satisfied, will represent the distance to the corresponding constraint, as represented by its hyperplane.

The problem would be in standard linear programming form if we wished to maximize

$$\sum_{j=1}^{M} s_j .$$

This optimization might however very well be satisfied with one or several of $s_j=0$. It would therefore be better to maximize

$$\min_{1 \leq j \leq M} s_j .$$

This can however not be done with standard linear programming methods such as the simplex method, since this objective function is not linear.

The method suggested by Mike Powell turns the above into a standard linear programming problem by choosing all s_j equal, or as fixed constants α_j times s.

The system can therefore be written

$$(5) \quad \sum_{i=1}^{N} a_{ji}x_i + \alpha_j s = b_j \qquad j = 1,\dots,M$$

$$x_i \geq 0 \qquad i = 1,\dots,N$$

$$s \geq 0$$

$$\alpha_j \geq 0 \quad \text{(fixed)} \qquad j = 1,\dots,M$$

Only $\underset{\sim}{x}$ and s are subject to optimization, and the objective function is chosen simply as

s.

The main reason for the constants α_j is to allow pure equalities as constraints. These constants can also replace the normalization mentioned in note 2 above.

This method has only slightly increased the complexity in the original linear programming problem, by adding one additional unknown s, but it will return a solution in the middle of the original convex hypervolume.

The previously recommended method on the other hand did not require the calculation of any optimum solution at all. The method was simply to use Phase 2 of the simplex method (possibly without the use of an objective function) on the original problem (1) to obtain several feasible solutions, until not all of them are on the same hyperplane, and then returning the mean of them as the final solution. Due to the convexity, it has to be inside the hypervolume of points satisfying (1). The basic method could be improved by always looking for a new feasible point to be "as far as possible" from the already obtained points.

The amount of computation is probably about the same for the Powell method and the previously recommended method, but the algorithm is much simpler for the Powell method, which also returns a solution that is optimal in a rather natural sense.

5. Additional Remarks

The determining of input data to obtain a desired path of a general computer program is an in general not solvable problem (cf. the halting problem). In this simplified treatment some additional remarks may be of interest.

It is believed that most conditional statements involve only linear relations, although a common nonlinear case arises from comparing Euclidean distances. Leon Osterweil has suggested a statistical study of existing computer programs in relevant programming languages for determining the amount of nonlinear statements. It will be considerably harder to develop a solver for nonlinear non-equalities that works in most cases, than the present simple linear solver.

In practice, the set of constraints will include both integer and real variables. Therefore methods for solving mixed linear programming problems have to be used. It is probable that rather distinct sets of variables, only interacting with each other and perhaps with just a few outside the set, can be found in many cases. This decomposition (see Dantzig, chapter 23) would also be suitable for statistical study of available software.

Another thing to note is that most paths are probably not executable. It is therefore very important that the solver first checks for feasibility, before too much effort is spent on obtaining the optimized solution (some of the methods in the earlier paper were not testing feasibility).

For many applications it would be useful if new constraints could be added successively, giving the middle of the convex hypervolume of feasible points as the solution as long as any solution exists. I believe that both algorithms would work satisfactorily in such an environment, but if the hyperplane representing the new constraint intersects the old hypervolume, it is not sure that the old solution point is a feasible point. It <u>might</u> here be necessary to start from scratch. The problem of introducing new constraints has probably been discussed in the literature on linear programming.

A major problem with the new method is the identification of equality constraints. If not identified, the equality constraints will force the distance parameter s to zero. Also with the old method problems will occur when equality constraints are present, since all feasible points have to lie in the corresponding hyperplane. For determining solutions not in the same plane the equality constraints have to be removed.

With the new method we get some extra information from the distance s regarding the closeness of the set of possible input data for the considered path, but $s = 0$ might be an indication of an equality constraint where $\alpha \neq 0$, or that the set of input data really is critical. I strongly recommend implementing the new (Powell) method instead of the previously recommended mean value simplex method.

I would like to stress once again that rounding errors from expressions produced by a symbolic manipulation system might be very different from those of the original computer program. It is of course also possible to obtain exact arithmetic in most cases when symbolic methods are being used.

The rounding errors can also cause problems in program validation if some parameters are given with to high accuracy in the program. For example, if the program checks if SQRT(X) is less than SQRTWO = $\sqrt{2}$, it is probably better to use SQRTWO = SQRT(2.0) than SQRTWO = 1.414.... (with as many digits the compiler uses) to initialize the variable, in order to avoid inconsistencies from a possible interval between SQRTWO and SQRT(2.0). - If the square-root can be removed completely, that is of course the ideal solution.

The relaxation method introduced by Agmon (1954) and Motzkin and Schoenberg (1954) is rather similar to my method H in Einarsson (1977a). They prove termination of the process, but do not consider numerical properties. A more recent reference on linear inequalities is Orden (1971).

Acknowledgements

This paper was written while the author was visiting the Mathematics Research Center at the University of Wisconsin in Madison. I am very grateful to its director, Professor Ben Noble, for his kind hospitality during my stay in Madison.

I would also like to thank Mike Powell for suggesting the new method.

REFERENCES

1. Agmon, Shmuel (1954): The Relaxation Method for Linear Inequalities. Canadian Journal of Mathematics, Vol. 6, pp. 382-392.

2. Clarke, Lori A. (1976): A System to Generate Test Data and Symbolically Execute Programs, IEEE Transactions on Software Engineering, Vol. 2, pp. 215-222.

3. Dantzig, George B. (1963): Linear Programming and Extensions, Princeton University Press, Sixth Printing, 1974, 627 pp., $16.

4. Einarsson, Bo (1977a): Progress Report, dated January 4, 1977.

5. Einarsson, Bo (1977b): Data Flow Analysis on Different Programming Languages, Draft report, dated March 10, 1977. Revised May 10, 1977.

6. Fosdick, Lloyd D. and Osterweil, Leon J. (1976): Data Flow Analysis in Software Reliability, Computing Surveys, Vol. 8, pp. 305-330.

7. Motzkin, T. S. and Schoenberg, I. J. (1954): The Relaxation Method for Linear Inequalities. Canadian Journal of Mathematics, Vol. 6, pp. 393-404.

8. Orden, A. (1971): On the Solution of Linear Equation/Inequality Systems. Mathematical Programming, Vol. 1, pp. 137-152.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 1780 ✓	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SOLVING A SYSTEM OF LINEAR INEQUALITIES FOR GENERATING TEST DATA		5. TYPE OF REPORT & PERIOD COVERED Summary Report - no specific reporting period
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Bo Einarsson		8. CONTRACT OR GRANT NUMBER(s) DAAG29-75-C-0024 ✓
9. PERFORMING ORGANIZATION NAME AND ADDRESS Mathematics Research Center, University of Wisconsin, 610 Walnut Street, Madison, Wisconsin 53706		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS Work Unit Number 8 - (Computer Science)
11. CONTROLLING OFFICE NAME AND ADDRESS U. S. Army Research Office, P.O. Box 12211, Research Triangle Park, North Carolina 27709		12. REPORT DATE August 1977
		13. NUMBER OF PAGES 11
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES Permanent Affiliation of the Author

Försvarets Forskningsanstalt (FOA)
National Defense Research Institute S-147 00 Tumba
Department 2/Proving Grounds SWEDEN
Box 98

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simplex method, linear programming, inequality solver, rounding, Data Flow Analysis, software validation, test data generation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A simple inequality solver, based on the simplex method, for determining safe input data corresponding to execution of a certain path of a computer program, is presented. Special consideration is given the influence from rounding errors on the solution.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)