c.1

COVERING RELAXATION METHODS FOR SOLVING THE ZERO-ONE POSITIVE POLYNOMIAL PROGRAMMING PROBLEM

by

© WILLEM VAESSEN

B.Sc., University of British Columbia, 1978

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Department of Computer Science)

We accept this thesis as conforming
to the required standard.

THE UNIVERSITY OF BRITISH COLUMBIA

April 30, 1980

In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my Department or by his representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of Computer Science

The University of British Columbia
2075 Wesbrook Place
Vancouver, Canada
V6T 1W5

Date 29-04-1980

ABSTRACT

Covering relaxation algorithms were first developed by Granot et al for solving positive 0-1 polynomial programming (PP) problems which maximize a linear objective function in 0-1 variables subject to a set of polynomial inequalities containing only positive coefficients ["Covering Relaxation for Positive 0-1 Polynomial Programs", Management Science, Vol. 25, (1979)]. The covering relaxation approach appears to cope successfully with the non-linearity of the PP problem and is able to solve modest size (40 variables and 40 constraints) sparse PP problems. This thesis develops a more sophisticated covering relaxation method which accelerates the performance of this approach, especially when solving PP problems with many terms in a constraint. Both the original covering relaxation algorithm and the newly introduced accelerated algorithm are cutting plane algorithms in which the relaxed problem is the set covering problem and the cutting planes are linear covering constraints. In contrast with other cutting plane methods in integer programming, the accelerated covering relaxation algorithm developed in this thesis does not solve the relaxed problem to optimality after the introduction of the cutting plane constraints. Rather, the augmented relaxed problem is first solved approximately and only if the approximate solution is feasible to the PP problem is the relaxed problem solved to optimality. The promise of this approach stems from the excellent performance of approximate procedures for solving integer programming problems. Indeed, the extensive computational experiments that were performed show that the accelerated algorithm has reduced both the number of set covering problems to be solved and the overall time required to solve a PP problem. The improvements are particularly significant for PP problems with many terms in a constraint.

TABLE OF CONTENTS

1. Introduction 1

2. Mathematical Preliminaries 7

3. The Covering Relaxation Approach 12

4. Evaluation of the Basic Covering Relaxation Algorithm 19

5. Modifications of the Covering Relaxation Approach 21

6. Greedy-like Approximate Procedures 32

   6.1 Greedy-like procedures for the SC Problem 33

   6.2 Greedy-like procedures for the MC Problem 37

7. Evaluation of the Accelerated Covering Relaxation Algorithm 39

   7.1 Aspects of the Implementation 40

   7.2 Problem Generator 45

   7.3 Computational Results 47

8. Bibliography 61

9. Appendix: Three Benchmark Problems 64

LIST OF TABLES

Table I 53

Table II 54

Table III 55

Table IV 56

Table V 57

Table VI 58

LIST OF FIGURES

Figure 1. CPU Time Use 59

Figure 2. Method I vs Method VA 60

ACKNOWLEDGEMENT

I would like to thank professor Paul C. Gilmore for his advice and encouragement; my thesis supervisors, professors Frieda and Daniel Granot, for their guidance and financial support; and Ihor Gowda for taking an interest in my work.

This thesis is dedicated to

Zero and One.

1. Introduction

This thesis considers and develops methods for solving the positive 0-1 polynomial programming problem (PP) which maximizes a linear objective function in 0-1 variables that is constrained by a set of polynomial inequalities containing only positive coefficients. It is of the form:

$$\text{MAX} \sum_{h=1}^{n} c_h x_h$$

subject to

$$\text{(PP)} \qquad \begin{aligned} F_i(x) &\le b_i \quad i=1, \ldots ,m \\ x_h &\in \{0,1\} \quad h=1, \ldots ,n \end{aligned}$$

where for each h, i and j

$$F_i(x) = \sum_{j=1}^{P(i)} a_{ij} \prod_{k \in N_{ij}} x_k;$$

$a_{ij} \ge 0$; $c_h > 0$ and

N_{ij} is a subset of $N = \{1,2, \ldots ,n\}$.

N_{ij} is the set of variables in j^{th} term of the ith constraint. F_i will be referred to as a posynomial since it is a polynomial with only positive coefficients.

General polynomial constraints in 0-1 variables were found to be useful for modeling diverse problems in business and engineering. For example, they are used to incorporate risk into capital budgeting [22], to plan irrigation systems [21], to

perform cluster analysis [24], and to formulate various scheduling problems [17,23]. This thesis, however, focuses on the positive polynomial problem.

Methods of Solution

The PP problem, like many other well known mathematical optimization problems, such as the travelling salesman optimization problem, is at least as hard to solve as any one of the notoriously difficult NP-complete problems. The methods that have been proposed in the literature for solving the PP problem employ at some stage an enumeration technique for searching very large solution spaces.

The two major methods are the linearization and the boolean algebra approaches. Neither approach tackles the non-linearity of the PP problem directly. Both approaches solve the PP problem by reducing it to a less complex linear problem. The linearization approach [8,9,29] transforms the PP problem into an equivalent 0-1 linear programming problem. Each distinct term in a PP problem is replaced by a new variable and additional constraints are introduced to ensure that the values of the term and the new variable coincide. Explicitly, let $T_{ij} = \prod_{k \in N_{ij}} x_k$. Then T_{ij} can be replaced by a new 0-1 variable y_{ij} in every constraint in which T_{ij} occurs if the following two linear constraints, which are written in terms of the original 0-1 variables x_k and the new variables y_{ij}, are satisfied:

$$\sum_{k \in N_{ij}} x_k \leq y_{ij} + |N_{ij}| - 1$$

$$\sum_{k \in N_{ij}} x_k \geq |N_{ij}| y_{ij}.$$

A slightly more efficient transformation that reduces the number of constraints is possible, see [9]. But it was found to be less effective computationally. The number of variables in the transformed 0-1 linear programming problem is potentially very large since the possible number of posynomial terms increases exponentially with the number of variables in the original PP problem. In [26,27] an implicit enumeration algorithm for solving the transformed 0-1 linear programming problem has been developed in which the additional constraints introduced by the linearization process are only considered implicitly. This approach obviously suffers from the same drawback as the standard linearization approach. The computational results reported in [26,27] underscore this conclusion.

Using boolean algebra techniques, Granot and Hammer [14] have shown constructively that every PP problem is equivalent to a linear set covering problem in the complementing variables of the original PP problem. Their attempt to solve PP problems employing this transformation failed because the number of covering constraints in the equivalent linear covering problem is often very large.

This attempt is, nevertheless, of theoretical importance since the covering relaxation approach, developed by Granot et al [10], overcomes its major deficiency. The covering

relaxation approach is a cutting plane type algorithm. It produces an optimal solution to the PP problem by solving each of a sequence of nested linear set covering problems to optimality. Each set covering problem is a relaxation of the original PP problem and usually contains only a small subset of the constraints of the equivalent set covering problem. The computational results reported in [10] indicate that the covering relaxation approach is a viable method for solving modest size (40 variables and 40 constraints) sparse problems. However, the performance of the method deteriorates rapidly as the number of distinct terms in the constraints of a PP problem increases.

Using a similar approach, D. Granot and F. Granot [11] have developed an algorithm for solving the general 0-1 polynomial programming problem.

An Outline

The plan of this thesis is as follows. Chapter 2 reviews the theoretical work of Granot and Hammer [14] that forms the basis of the original covering relaxation approach of Granot et al [10], which will be reviewed in chapter 3. The covering relaxation approach is evaluated in chapter 4. The major contribution of this thesis, to be presented in chapter 5, is an improved covering relaxation algorithm. The three modifications incorporated in the improved algorithm are motivated by the shortcomings of the original algorithm discussed in chapter 4.

The most interesting of these modifications makes novel use of approximate procedures for solving the set covering problem to reduce the number of set covering problems to have to be solved to optimality. The improved algorithm does not solve a set covering problem to optimality at every iteration. Instead, a set covering problem is first solved approximately and will be solved to optimality only if the approximate solution is feasible to the original PP problem. The second modification that is introduced uses an approximate solution for the PP problem to produce a better initial set covering problem. The third modification generates tighter set covering constraints. The computational results to be reported in chapter 7 show that these modifications have significantly improved the performance of the covering relaxation approach by reducing the number, the size and the density of the set covering problems solved to optimality. Chapter 6 develops greedy-like approximate procedures for the set covering problem. It also discusses similar procedures that can be used to produce tight set covering constraints. Chapter 7 discusses the pertinent aspects of the implementation of the covering relaxation approach which has been a helpful and reliable tool for the development approach. The implementation employs a rudimentary implicit enumeration algorithm for solving the set covering problem. The computational results reported in chapter 7 show that if the number of variables and the density of the set covering problems to be solved to optimality is large, then the CPU time required for solving a PP problem to optimality is almost entirely

devoted to solving the set covering problems. The performance measures indicate that if a more efficient algorithm for the set covering problem (see for instance [1,5]) is implemented properly, then the covering relaxation approach will be able to solve much larger PP problems. The appendix contains three pseudo-randomly generated PP problems that were solved by the accelerated covering relaxation algorithm. The set covering problems that were solved to optimality are also included in the appendix. It is hoped that these PP problems will serve as benchmarks for further research.

The new material that is developed in this thesis is collected in ["An Accelerated Algorithm for the Postive 0-1 Polynomial Problem", Faculty of Commerce and Business Administration, University of British Columbia, May 1980] by F. Granot, D. Granot and W. Vaessen.

2. Mathematical Preliminaries

This chapter summarizes the theoretical results and notions of [2,14] that form the basis of the covering relaxation approach to be discussed in chapter 3.

Granot and Hammer [14] used boolean algebra techniques to show that every posynomial inequality can be replaced by an equivalent set of linear covering inequalities. To see how this can be accomplished, consider, first of all the linear inequality of the form:

$$(1) \quad \sum_{j=1}^{n} a_j x_j \leq b, \text{ where } a_j > 0,\ j=1, \ldots ,n.$$

Definition 1 A subset U of N, $N = \{1,2, \ldots ,n\}$, is a cover of (1) if $\sum_{j \in U} a_j > b$.

Definition 2 A cover U is prime if no proper subset of U is a cover.

Definition 2a $|U|$ is the cardinality of the prime cover U.

Definition 2b The cardinality of a prime cover is minimal if it is smaller than or equal to the cardinalities of all other prime covers of (1).

Let C be the set of all prime covers of (1).

Theorem 1 A 0-1 vector x satisfies (1) if and only if

$$\prod_{j \in U} x_j = 0 \text{ for all } U \in C,$$

equivalently, if and only if y, the complement of x, satisfies

(2) $\sum_{j \in U} y_j \geq 1$ for all $U \in C$, where $y=1-x$.

Thus the linear inequality (1) in the 0-1 vector x can be reduced to an equivalent set of linear covering inequalities (2) in the complement of x. This reduction can be applied to every constraint of a positive 0-1 linear problem to obtain an equivalent linear covering problem. A similar result was independently obtained by Balas and Jeroslow [2] using a completely different approach.

The above results were generalized in [14] for positive 0-1 polynomial inequalities. Consider the inequality of the form:

(3) $\sum_{k=1}^{p} a_k \prod_{j \in N_k} x_j \leq b$, where $a_k>0$, $k=1, \ldots ,p$ and $N_k \subset N$

A 0-1 vector x satisfies (3) if and only if there is a 0-1 vector z such that x and z satisfy:

(4) $\sum_{k=1}^{p} a_k z_k \leq b$ and $z_k = \prod_{j \in N_k} x_j$ for $k=1, \ldots ,p$.

Let C be the set of all prime covers of the inequality of (4). Then (3) is satisfied if and only if

$$\prod_{k \in U} z_k = 0 \text{ for all } U \in C,$$

equivalently, if and only if

$$\prod_{k \in U} \prod_{j \in N_k} x_j = 0 \text{ for all } U \in C.$$

This condition can also be rewritten as a set of linear covering

inequalities in the complementing variables.

<u>Theorem 2</u> A 0-1 vector x satisfies (3) if and only if y, the complement of x, satisfies

$$(5) \quad \sum_{j \in S(U)} y_j \geq 1 \text{ for all } U \in C,$$

where y=1-x and S(U) is the union of N_k for all $k \in U$.

Thus, every constraint in a positive 0-1 polynomial programming problem can be reduced to an equivalent set of linear covering constraints to obtain a linear set covering (SC) problem equivalent to the original 0-1 polynomial problem.

To simplify the subsequent discussion of the covering relaxation approach three related properties of covering constraints need to be defined.

<u>Definition 3</u> A covering constraint (5) is <u>prime</u> if there does not exist a subset R of S(U) such that $\sum_{j \in R} y_j \geq 1$ can also be generated from (3).

<u>Definition 3a</u> The size of S(U) is the <u>cardinality</u> of the covering constraint, i.e. The number of variables occuring in the covering constraint.

<u>Definition 3b</u> The cardinality of a covering constraint (5) is <u>minimal</u> if it is smaller than or equal to the cardinalities of all other covering constraints that can be derived from (3).

The reader who is unfamiliar with the above material is urged to examine the following example:

Example 1. Reducing a Posynomial Constraint to an Equivalent Set of Covering Constraints.

$$\underset{z_1}{\underset{\vdots}{8x_2x_3}} + \underset{z_2}{\underset{\vdots}{5x_2x_4}} + \underset{z_3}{\underset{\vdots}{3x_1x_3x_5}} + \underset{z_4}{\underset{\vdots}{2x_2x_6}} \leq 9$$

$$\begin{array}{lcll}
z_1z_2=0 & \Leftrightarrow & x_2x_3x_4=0 & \Leftrightarrow \quad y_2+y_3+y_4 \geq 1 \\
z_1z_3=0 & \Leftrightarrow & x_1x_2x_3x_5=0 & \Leftrightarrow \quad y_1+y_2+y_3+y_5 \geq 1 \\
z_1z_4=0 & \Leftrightarrow & x_2x_3x_6=0 & \Leftrightarrow \quad y_2+y_3+y_6 \geq 1 \\
z_2z_3z_4=0 & \Leftrightarrow & x_1x_2x_3x_4x_5x_6=0 & \Leftrightarrow \quad y_1+y_2+y_3+y_4+y_5+y_6 \geq 1
\end{array}$$

$x_i=1-y_i$ for $i=1,2,3,4,5,6$.

The first three covering constraints are prime. The cardinalities of the first and second constraint are minimal.

Granot and Hammer [14,15] used the above result to propose an algorithm for solving the PP problem that first reduces the PP problem to an equivalent SC problem which is then solved to optimality by a standard method. This algorithm, however, is not attractive computationally because the number of constraints in the equivalent SC problem is often very large, even for fairly small PP problems. For example, a constraint of the form

$\sum_{j=1}^{n} x_j \leq T(n/2)$, where $T(r)$ denotes the integer part of r,

is equivalent to C(n,T(n/2)+1) covering constraints.

The next chapter describes the covering relaxation approach which was developed by Granot et al [10] to overcome the deficiency of employing the transformation of the PP problem to the SC problem. The covering relaxation approach does not generate and solve the entire equivalent SC problem but instead, solves each of a nested sequence of much smaller SC problems, each of which is a relaxation of the PP problem to be solved.

3. The Covering Relaxation Approach

The covering relaxation approach is a cutting plane type algorithm. Cutting plane algorithms were first developed to solve integer linear programming (ILP) problems. See e.g. [7] for an in-depth treatment of this technique.

The basic idea of the cutting plane algorithms for the ILP problems is to solve a sequence of successively tighter linear programming (LP) relaxations of the original ILP problem until the LP optimum satisfies the integrality requirements of the ILP problem. A tighter LP relaxation is obtained by adding one or more constraints to the current LP problem which cut off the current non-integer optimum but which do not remove any integer points from the feasible region of the ILP problem. Such constraints are called cutting planes. The initial relaxation is the LP problem obtained by dropping the integrality requirements on the variables from the ILP problem.

The motivation for the application of a cutting plane technique for solving the PP problem is the often unmanageable number of covering constraints in the equivalent SC problem. Rather than solving the entire equivalent SC problem, the covering relaxation approach solves each of a nested sequence of SC problems that are successively tighter relaxations of the original PP problem. The SC problems are referred to as covering relaxation (CR) problems.

The initial covering relaxation (CR_0) problem consists of a small subset of the covering constraints of the equivalent SC

problem. Each CR problem is obtained by augmenting the constraint set of its predecessor with cutting planes. These cutting planes are covering constraints generated from those terms in the violated posynomial constraints of the original problem that do not vanish at the optimal solution for the current CR problem. They cut-off solutions to the CR problems that are not feasible to the original problem. Clearly, each CR problem is a relaxation of the original PP problem and is tighter than all its predecessors.

The basic version of covering relaxation approach can now be stated in an informal procedural fashion. The discussion of some aspects of the implementation is deferred until chapter 7.

BASIC COVERING RELAXATION ALGORITHM

```
find a starting node;
generate CR0 with the starting node;
repeat
  solve the current CR problem to optimality;
  attempt to generate one or more additional covering
  constraints that cut off the optimal solution to
  the current CR problem;
  if no cuts can be added then
    the optimal solution to the current CR problem is
    also feasible to the original PP problem;
until the current optimal solution is feasible;
stop.
```

The algorithm terminates with an optimal solution for the PP problem in at most 2^n iterations. Termination occurs in finitely many steps because the solution space contains at most 2^n binary vectors and at each iteration the optimal solution to the CR problem is discarded. (The experimental results reported in [10] show that that the average behaviour of the algorithm is significantly better!) The solution on termination is optimal because each CR problem is a relaxation of the PP problem to be solved.

A single covering constraint suffices to cut off the optimal solution to a CR problem. In general, however, more than one posynomial constraint is violated. In addition, more than one covering constraint can be generated from each violated posynomial constraint. This flexibility is exploited in the basic algorithm and to a larger extent in the accelerated covering relaxation algorithm to reduce the number of iterations. See section 7.1 for details.

It is often possible to reduce the dimensions of the CR problem by applying a small number of simplification rules that will eliminate dominated covering constraints and variables whose values can be determined in advance, see for instance [7].

If at most one covering constraint is added at each iteration to form the augmented CR problem, then a cutting plane technique for solving SC problems [7] can be employed to obtain the optimal solution for the augmented problem. This method, however, is unlikely to be effective because the number of iterations will be large.

The starting node used by the basic algorithm is (1,1, ... ,1).

The basic algorithm generates one covering constraint from each violated posynomial constraint. To see which covering constraint is generated, consider the posynomial constraint of the form:

$$\sum_{k=1}^{p} a_k P_k \leq b, \text{ where } P_k \text{ denotes a product of variables.}$$

The basic algorithm generates the prime cover which forms the covering constraint in the following manner:

```
begin
order the terms Pk such that ak-1 ≥ ak for k = 2,...,p;
prime-cover := ∅
lhs := 0;
k := 1;
repeat
    lhs := lhs + ak;
    prime-cover := prime-cover + Union(prime-cover,{k});
    k := k +1;
until lhs > b or k >p
end
```

The ordering of the terms ensures that the cardinality of the prime cover forming the covering constraint is minimal. However, this condition does not quarantee that the covering constraint is prime. Hence, the cardinality of the covering constraint generated in the above manner is not minimal.

Example 2. Solving a positive 0-1 polynomial problem.

Solve

MAX $7x_1 + 5x_2 + 4x_3 + 3x_4 + 3x_5 + 2x_6$

s.t.

1) $8x_1x_2x_3 + 6x_2x_4 + 5x_3 + 4x_3x_6 \leq 13$

2) $7x_2x_3x_5 + 2x_1x_6 \leq 6$

3) $7x_4x_5 + 4x_1x_4 + 4x_2x_5x_6 \leq 8$

4) $5x_2x_3 + 5x_1x_5 + 3x_6 + 2x_4 \leq 7$

5) $3x_2x_4 + 2x_3x_4 + 2x_4x_5 + 2x_5x_6 + x_1x_6 \leq 5$

$x_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Use (1,1, ... ,1) to generate one covering constraint from each violated posynomial constraint in the manner desribed above:

from 1) $x_1x_2x_3x_4=0$

2) $x_1x_2x_3x_5x_6=0$

3) $x_1x_4x_5=0$

4) $x_1x_2x_3x_5=0$

5) $x_2x_3x_4x_5=0$

The fourth constraint dominates the second constraint, which can therefore be dropped. The four remaining constraints are written as covering constraints to form the initial covering relaxation problem.

Iteration 1. Solve the initial CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{lllllllll}
y_1 & + & y_2 & + & y_3 & + & y_4 & & & \geq 1 \\
y_1 & & & & & + & y_4 & + & y_5 & \geq 1 \\
y_1 & + & y_2 & + & y_3 & & & + & y_5 & \geq 1 \\
 & & y_2 & + & y_3 & + & y_4 & + & y_5 & \geq 1
\end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Observe that the value of y_2 can be fixed a priori at 0 because y_3 can satisfy the same constraints at a lower cost.

The solution for the first CR problem is $y_1=y_2=y_3=y_6=0$, and $y_4=y_5=1$. Equivalently, $x_1=x_2=x_3=x_6=1$, and $x_4=x_5=0$. Use $x^*=(1,1,1,0,0,1)$ to generate additional covering constaints:

from 1) $x_1x_2x_3x_6=0$

4) $x_2x_3x_6=0$

Only the latter constraint needs to be included in the augmented problem since it dominates the former.

Iteration 2. Solve the augmented CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{lllllllllll}
y_1 & + & y_2 & + & y_3 & + & y_4 & & & & \geq 1 \\
y_1 & & & & & + & y_4 & + & y_5 & & \geq 1 \\
y_1 & + & y_2 & + & y_3 & & & + & y_5 & & \geq 1 \\
 & & y_2 & + & y_3 & + & y_4 & + & y_5 & & \geq 1 \\
 & & y_2 & + & y_3 & & & & & + \; y_6 & \geq 1
\end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Its solution is $y_1=y_2=y_5=y_6=0$, and $y_3=y_4=1$.

Use $x^*=(1,1,0,0,1,1)$ to generate additional covering constraints:

from 4) $x_1x_5x_6=0$

Iteration 3. Solve the augmented CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{lllllll}
y_1 + y_2 + y_3 + y_4 & & & & & & \geq 1 \\
y_1 & & & + y_4 + y_5 & & & \geq 1 \\
y_1 + y_2 + y_3 & & & + y_5 & & & \geq 1 \\
\quad\;\; y_2 + y_3 + y_4 + y_5 & & & & & & \geq 1 \\
\quad\;\; y_2 + y_3 & & & & + y_6 & & \geq 1 \\
y_1 & & & + y_5 & + y_6 & & \geq 1
\end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Its solution is $y_1=y_2=y_4=y_6=0$, and $y_3=y_5=1$. Equivalently, $x_1=x_2=x_4=x_6=1$, and $x_3=x_5=1$. No covering constraint can be generated with $x^*=(1,1,0,1,0,1)$. In other words, it is feasible to the original PP problem. Since x^* is an optimal solution for a relaxation of the PP problem it is an optimal solution for the PP problem itself.

Solving this problem by first transforming it to a 0-1 linear problem requires 14 extra variables and at least 21 extra constraints. A considerable amount of computation is required to reduce the equivalent SC problem to 9 covering constraints.

This example will be used in chapter 5 to illustrate the effects of the modifications of the basic algorithm.

4. Evaluation of the Basic covering relaxation Algorithm

The experimental results reported in [10] showed the covering relaxation approach to be very promising. Over 200 randomly generated problems with up to 50 variables and 50 constraints were solved using the basic algorithm. The CPU time required to solve each individual problem was often less than 1 second and exceeded 1 minute only for three problems with a relatively large number of terms. (These results were obtained on an IBM 370/168.) The number of covering constraints in the final CR problem usually did not exceed the number of posynomial constraints in the original problem. The number of CR problems solved was typically between 1 and 7.

The success of the covering relaxation approach can largely be attributed to the fact that no additional variables are introduced in the solution process. It is worthwhile to note that the major shortcoming of the procedures that linearize the posynomial constraints stems from the radical increase in the number of variables in the linearized problem to be solved.

However, most of the problems solved in [10] contained on average only 3 terms per constraint. It was shown in [10] that all three performance indicators (the CPU time, the number of iterations, and the number of covering constraints in the final CR problem) deteriorate as the average number of terms in a constraint becomes larger. The CPU time showed the most marked increase. This increase is due, in part, to the much higher density of the CR problems. The computational results in [10]

reveal that the density of the CR problem has a dramatic effect on the CPU time required to solve the PP problem.

The accelerated covering relaxation algorithm, which will be discussed in chapter 5, employs three modifications that are shown to substantially improve the effectiveness of the covering relaxation approach for less sparse PP problems.

5. Modifications of the Covering Relaxation Approach

The accelerated covering relaxation algorithm, which will be presented in this chapter, is a more sophisticated version of the basic covering relaxation algorithm. Three aspects of the basic algorithm have been modified to improve the effectiveness of the covering relaxation approach.

Modification 1. The conceptual difference between the accelerated covering relaxation algorithm and other cutting plane methods in integer programming (IP) is that each relaxed problem is not necessarily solved to optimality. The basic idea of modification 1 is to solve the CR problems approximately as long as the approximate solution at each iteration is not feasible to the original PP problem. Cuts can be generated in the usual manner to eliminate the successive approximate solutions. An CR problem is solved to optimality only when the approximate solution for this problem is feasible to the original problem. The modification is implemented by adding one extra step to the basic covering relaxation algorithm to compute an approximate solution for the current CR problem.

The promise of this modification stems from the excellent performance of the various approximate algorithms for solving IP problems, see for example [3,6,19,25,28]. Four greedy-like approximate procedures for the SC problem are developed in section 6.1. These procedures are fast and the values of the optimal solutions obtained are on average within 2% of

optimality. Observe that if the approximate procedures for solving the SC problem obtain the optimal solution most of the time, then the number of times that a CR problem will need to be solved to optimality will be very close to 1.

It is possible to ensure that only a single CR problem is solved to optimality. Suppose that a slightly modified enumeration technique is used the current best solution is for solving the CR problems. The enumeration of the current CR problem will be interupted whenever a solution better than the current best solution is encountered. If this solution is feasible to the original PP problem then the enumeration continues; otherwise covering constraints are generated to eliminate this solution and the enumeration of the augmented CR problem starts anew. If the enumeration of the current CR problem terminates then the current best solution is feasible and optimal to the original PP problem.

There is evidence that modification 1 can also be applied to the covering relaxation approach for the general 0-1 polynomial programming problem. Moreover, D. Granot and F. Granot are currently investigating the effectiveness of a very similar idea for improving Benders partitioning procedure for solving mixed integer programming problems, see [12].

<u>Modification 2</u>. An approximate solution for the PP problem is employed to obtain an initial CR problem. This CR problem is, in a sense, a closer approximation of the PP problem in the neighborhood of its optimal solution than the initial CR problem

generated by (1,1, ... ,1). Modification 2 is also motivated by the excellent performance of the approximate procedures for solving IP problems. F. Granot and W. Vaessen have developed, implemented and analyzed greedy-like approximate procedures for the PP problem, see F. Granot [12]. The values of the approximate solutions found by these procedures were consistently close to the optimal values and were equal to the optimal values in over 60% of the problems that were solved.

CR_0 is formed by generating covering constraints from a number of different starting nodes that are derived from the approximate solution for the PP problem. Explicitly, let x_{pp} designate an approximate solution for the PP problem. CR_0 is constructed in the following manner:

```
begin
CR := ∅;
Zeros := {v|x_pp(v)=0};
starting-node := x_pp;
for v ∈ Zeros do
    begin
    starting-node(v) := 1;
    generate augmented CR with starting-node;
    starting-node(v) := 0;
    end
CR_0 := CR;
end
```

The approximate solutions produced by the methods described in [12] cannot be improved by changing one or more variables from 0 to 1. This property of x_{pp} ensures that at least on starting-node will be infeasible and hence that CR_0 will not be empty.

It is intuitively clear that if x_{pp} is equal or close to the optimal solution of the PP problem, then the solution of CR_0 formed in the above manner is likely to be closer to the solution of the original PP problem than the solution of CR_0 generated by (1,1, ... ,1). The overall number of iterations and the number of constraints in the final CR problem are, therefore, likely to be smaller. The density of CR_0 generated in the above manner is not necessarily lower.

The values of the approximate solutions for the PP problem and the CR problems can be utilized as lower bounds by the procedure solving the CR problem to optimality.

<u>Modification 3</u>. The quality of the covering constraints is improved by implicitly enumerating all or most of the prime covers that can be generated from the violated posynomial constraints and selecting those prime covers whose associated covering constraints have minimal cardinality. The objective of modification 3, which will be justified shortly, is to generate tighter covering constraints.

Let x(c) designate either an approximate or optimal solution for the current CR problem. Consider the posynomial inequality (3). Let $R=\{i \mid \forall j \in N_i,\ x_j(c)=1\}$. R represents the set

of non-vanishing terms of (3) at x(c). The posynomial inequality from which the terms vanishing at x(c) have been dropped will be of the form:

$$(6) \quad \sum_{i \in R} a_i \prod_{j \in N_i} x_j \leq b.$$

The covering constraints that can be derived from (6) are of the form:

$$\sum_{j \in S} y_j \geq 1, \text{ where } S \text{ is a subset of } N.$$

Let C_1 be such a covering constraint with $S=S_1$. If C_1 is not prime (see def. 3) then there exists a covering constraint C_2 with $S=S_2$ such that S_2 is a proper subset of S_1. Clearly, C_2 is a better choice than C_1 since it dominates the constraint associated with S_2. If, on the other hand, C_1 is prime but not minimal (see def. 3b) then there exists a covering constraint C_2 with $S=S_2$ such that C_2 is prime and $|S_2| < |S_1|$. The choice between C_1 and C_2 is not clear-cut in this case. The properties of covering constraints that are defined in chapter 2 do not suffice to evaluate the relative merits of C_1 and C_2 since they do not take into account the properties and structure of the CR problem to be augmented. For lack of insight, C_2 is chosen because, first of all, it eliminates more 0-1 vectors from the solution space of current CR problem and secondly, the solution of the augmented CR problem is more likely to move closer to the optimal solution since C_1 is a tighter covering constraint.

The previous paragraph motivates the search for covering constraints with minimal cardinality. Formally, the minimal cardinality (MC) problem can be stated as follows:

$$\text{(MC)} \qquad \underset{S \subset R}{\text{Minimize}} \left| \bigcup_{i \in S} N_i \right|$$

subject to

$$\sum_{j \in S} a_j > b$$

If $|R|$ is not much larger than $|N|$, we can employ an implicit enumeration technique for solving the MC problem. If a fixed branching method is used, then the covering constraint produced by the basic algorithm is the first feasible solution encountered in a depth-first search of the solution tree of the MC problem. Note that if (6) is a linear inequality, then the first feasible solution is also the optimal solution.

If, on the other hand, $|R|$ is much larger than $|N|$, then approximate procedures can be employed to solve the MC problem. Greedy-like approximate procedures for the MC problem are developed in section 6.2.

The covering constraints generated by a restricted enumeration of the solution tree of the MC problem or by the greedy-like approximate procedures to be discussed, are not guaranteed to be prime. Explicitly, let

$\sum_{j \in V} y_j \geq 1$ be a covering constraint produced from (6) using an approximate method. Then it is possible that a proper subset V' of V is also a covering constraint for the same posynomial inequality (6). This condition can easily be detected. Let $x_i(v')=1$ if $i \in V'$ and 0 otherwise. Then V is not prime if and only if there exists a proper subset V' of V such that (6) is not violated by $x(v')$.

The accelerated version of the covering relaxation approach can now be stated in an informal procedural fashion.

ACCELERATED COVERING RELAXATION ALGORITHM

```
find an approximate solution for the PP problem;
generate CR0 using the approximate solution;
repeat
  solve the current CR problem approximately;
  attempt to generate additional cuts to
  eliminate the approximate solution;
  if the approximate solution is feasible
      to the original PP problem then
      begin
      solve the current CR problem to optimality;
      attempt to generate additional cuts
      to eliminate the optimal solution;
      if no cuts were added then
        the optimal solution to the CR problem
        is feasible to the original PP problem;
      end
until the current optimal solution is feasible;
stop.
```

The algorithm terminates with an optimal solution for the PP problem in at most 2^n iterations. Termination occurs in finitely many steps because the solution space contains at most

2^n binary vectors and at each iteration either an approximate solution or an optimal solution is discarded. The solution on termination is optimal to the PP problem because it is optimal to the CR problem which is a relaxation of the PP problem.

The accelerated algorithm does not necessarily generate one covering constraint from each violated posynomial constraint. If the cardinalities of the covering constraints that can be generated from one particular posynomial constraint are smaller than those that can be generated from all other posynomial constraints, then the accelerated algorithm uses more than one covering constraint from this posynomial constraint to augment the CR problem. Furthermore, if only a few posynomial constraints are violated then the accelerated algorithm attempts to add more than one covering constraint from those constraints that are violated. See section 7.1 for further details.

Example 3. Example 2 revisited.

In example 2 the following PP problem was solved:

MAX $7x_1 + 5x_2 + 4x_3 + 3x_4 + 3x_5 + 2x_6$

s.t.

1) $8x_1x_2x_3 + 6x_2x_4 + 5x_3 + 4x_3x_6 \leq 13$

2) $7x_2x_3x_5 + 2x_1x_6 \leq 6$

3) $7x_4x_5 + 4x_1x_4 + 4x_2x_5x_6 \leq 8$

4) $5x_2x_3 + 5x_1x_5 + 3x_6 + 2x_4 \leq 7$

5) $3x_2x_4 + 2x_3x_4 + 2x_4x_5 + 2x_5x_6 + x_1x_6 \leq 5$

$x_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

To illustrate modification 2, $x_{pp}=(1,1,0,1,0,1)$ will be used to generate CR_0.

The first starting-node (1,1,1,1,0,1) generates

from 1) $x_1x_2x_3x_4=0$

4) $x_2x_3x_6=0$

5) $x_1x_2x_3x_4x_6=0$

The second starting-node (1,1,0,1,1,1) generates

from 3) $x_1x_4x_5=0$

4) $x_1x_5x_6=0$

5) $x_2x_4x_5x_6=0$

These two sets of constraints are simplified and written as a set of covering constraints to form the initial CR problem.

Iteration 1. Solve the initial CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{lllllll}
y_1 + & y_2 + & y_3 + & y_4 & & & \geq 1 \\
 & y_2 + & y_3 & & & + y_6 & \geq 1 \\
y_1 & & & + y_4 & + y_5 & & \geq 1 \\
y_1 & & & & + y_5 & + y_6 & \geq 1 \\
 & y_2 & & + y_4 & + y_5 & + y_6 & \geq 1
\end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Its solution is $y_1=y_2=y_3=y_5=0$, and $y_4=y_6=1$. Equivalently, $x_1=x_2=x_3=x_5=1$, and $x_4=x_6=0$. Use (1,1,1,0,1,0) to generate additional covering constraints:

from 1) $x_1x_2x_3=0$

4) $x_1x_2x_3x_5=0$

The former constraint dominates the latter constraint which is therefore not included in the augmented problem. The former constraint also eliminates the first constraint in the initial CR problem.

Iteration 2. Solve the augmented CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{rcrcrcrcrcrcl}
 & & y_2 & + & y_3 & + & y_4 & & & & & \geq & 1 \\
 & & y_2 & + & y_3 & & & & & + & y_6 & \geq & 1 \\
y_1 & & & & & + & y_4 & + & y_5 & & & \geq & 1 \\
y_1 & & & & & & & + & y_5 & + & y_6 & \geq & 1 \\
 & & y_2 & & & + & y_4 & + & y_5 & + & y_6 & \geq & 1
\end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Its solution is $y_1=y_2=y_4=y_6=0$, and $y_3=y_5=1$. Equivalently, $x_1=x_2=x_4=x_6=1$, and $x_3=x_5=0$. $x^*=(1,1,0,1,0,1)$ is feasible and optimal to the PP problem.

To illustrate modification 3, (1,1,1,1,1,1) will be used to generate covering constraints with minimal cardinality:

from 1) $x_1x_2x_3=0$

5) $x_1x_2x_3x_5x_6=0$

3) $x_1x_4x_5=0$

4) $x_2x_3x_6=0$

5) $x_2x_3x_4x_5=0$

This set of constraints is simplified and written as as a set of covering constraints to form the initial CR problem.

Iteration 1. Solve the initial CR problem:

$$\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$$

s.t.

$$\begin{array}{lllllll} y_1 + y_2 + y_3 & & & & & \geq 1 \\ y_1 & & + y_4 + y_5 & & & \geq 1 \\ & y_2 + y_3 & & & + y_6 & \geq 1 \\ & y_2 + y_3 & + y_4 + y_5 & & & \geq 1 \end{array}$$

$y_i \in \{0,1\}$ for $i=1,2,3,4,5,6$.

Its solution is $y_1=y_2=y_4=y_6=0$, and $y_3=y_5=1$. Equivalently, $x_1=x_2=x_3=x_5=0$, and $x_4=x_6=1$. $x^*=(1,1,0,1,0,1)$ is feasible and optimal to the PP problem.

This example is perhaps misleading. It is not too difficult to construct an example for which the modifications are not as successful.

6. Greedy-like Approximate Procedures for the SC and MC Problems

This chapter develops the greedy-like procedures that are used by the accelerated algorithm to obtain good solutions for the CR and MC problems at polynomial cost.

The primal greedy approach starts with a feasible solution, usually (1,1, ... ,1), and moves towards a local optimum. Initially all variables are free. At each iteration the "promise" of each free variable is calculated to determine which variable will be dropped from the current solution. The promise of a variable is the expected pay-off of fixing the variable at a particular value. Once a variable is dropped, its value and possibly the value of other variables are fixed. The primal approach iterates until no more variables can be dropped.

The dual greedy approach starts with (0,0, ... ,0) and moves towards a feasible solution. Initially all variables are free. At each iteration the promise of each free variable is calculated to determine which variable will be added to the current solution. Once a variable is added, its value is fixed. The dual approach iterates until all covering constraints are satisfied.

An approximate solution obtained by the dual procedure can often be improved by applying the primal procedure to it. The primal procedure will drop to 0 the variables that are no longer needed to ensure feasibility.

6.1 Greedy-like Procedures for the SC Problem

The approximate procedures for the SC problem, to be described shortly, are variants of the basic primal and dual approaches. Two modifications have been introduced.

The first of these calculates a more accurate promise. The promise of a free variable y_i is usually calculated as the ratio of c_i, the cost-coefficient associated with y_i, and $|C_i|$, where $|C_i|$ is the cardinality of the column associated with y_i. However, if the variability of the cardinalities of the unsatisfied constraints is substantial, then a more accurate promise is calculated by taking into account the cardinalities of the constraints in which y_i occurs. The computational results to be reported in section 7.3 show that the inclusion of this weighting factor in calculating promises improves the effectiveness of the approximate procedures. A similar idea was independently proposed in [20] for the multidimensional knapsack problem.

The second modification restricts the set of free variables for which the promise is calculated. The restricted set consists of the free variables that occur in the tightest constraints. The effect of this modification is that the value of variables occuring in tight constraints are determined at an early stage.

It is assumed that the cardinality of each constraint is strictly greater than 1.

Consider the SC problem

$$\text{MIN} \sum_{j=1}^{n} c_j y_j$$

(SC) subject to

$$\sum_{j=1}^{n} a_{ij} y_j \geq 1 \text{ for } i=1, \ldots ,m$$

$$y_j \in \{0,1\} \text{ for } j \in N,$$

where $a_{ij} \in \{0,1\}$ for $\forall i,j$.

Define

$$f_i = \begin{cases} -\ 1 \text{ if } y_i \text{ is free} \\ -\ 0 \text{ if } y_i \text{ is fixed to either 0 or 1} \end{cases}$$

$F = \{i \mid f_i = 1\}$

$$h_i(y) = \begin{cases} -\ 0 \text{ if } \sum_{j=1}^{n} a_{ij} y_j \geq 1 \\ -\ 1 \text{ otherwise} \end{cases}$$

$H(y) = \{i \mid h_i(y)=0\}$, i.e. $H(y)$ is the set of constraints that are satisfied by a particular 0-1 vector y.

Now, for a 0-1 vector y and a corresponding vector $f=(f_i)$ let

$R_i(f) = \sum_{j=1}^{n} a_{ij} f_j$ and let

$T(y) = \{j \in F \mid \exists i \ (a_{ij}=1 \wedge \forall k \neq i \ (1 < R_j(f) \leq R_i(f)))\}$, i.e. $T(y)$ is the set of free variables occuring in the tightest unsatisfied covering constraints at y. Further let $M \equiv \{1,2, \ldots ,m\}$; and let e and e^r designate suitably small positive constants. Subsequently, $h_i(y)$, $H(y)$, $R_i(f)$ and $T(y)$ will be denoted by h_i, H, R_i and T respectively.

SC PRIMAL GREEDY-LIKE PROCEDURE

```
initialize H, R and T;
y := 1; f := 1;
repeat
  if slow-option then S := F
                  else S := T;
  for j∈S compute p_j = c_j/( Σ_{i=1}^{m} a_ij/((R_i-1)h_i+e^r))
  k := MAX p_i, in case of a tie MIN c_i;
       i∈S
  if p_k ≥ e then
      begin
      y_k := 0; f_k := 0;
      for i:=1 to n if R_i = 1 then
          begin
          j := 0;
          repeat j := j + 1
          until a_ij f_j = 1;
          f_j := 0
          update H;
          end
      update R and T;
      end
until p_k < e or F=∅ or H=M;
stop.
```

"R_i-1" is the weighting factor for the primal approach.

SC DUAL GREEDY-LIKE PROCEDURE

```
initialize T, H and R;
y := 0; f := 1;
repeat
  if slow-option then S := F
                  else S := T;
  for j∈S compute p_j = c_j/( Σ_{i=1}^{m} a_ij/(R_i h_i + e^r))
  k := MIN_{i∈S} p_i, in case of a tie MIN c_i;
  if p_k ≥ e then
      begin
      y_k := 1; f_k := 0;
      update T and H;
      end
until p_k < e or H=M;
stop.
```

"$R_i h_i$" is the weighting factor for the dual approach.

Both procedures run in $O(n^2)$ time. These time requirements are an insignificant overhead in the context of solving a PP problem to optimality.

At every iteration the accelerated algorithm employs both options for the primal and for the dual procedure to find four approximate solutions to the current CR problem. The solution with the smallest objective function value is used to generate the covering constraints that will be added to the current CR problem.

6.2 Greedy-like Procedures for the MC Problem

The two approximate procedures for the MC problem are again a variation of the greedy theme. To simplify the discussion, consider a slightly different formulation of the MC problem:

Minimize $|U(y)|$

subject to

$$\sum_{j \in R} a_j y_j > b$$

$$y_j \in \{0,1\},$$

where

$U(y)$ is the union of all N_j for which $y_j=1$.

N_j is the set of variables in the jth term,

R is the set of non-vanishing terms defined earlier,

as before define

$$f_i = \begin{cases} -\ 1 \text{ if } y_i \text{ is free} \\ -\ 0 \text{ if } y_i \text{ is fixed to either 0 or 1} \end{cases}$$

$$F(y) = \{i \mid f_i=1\}.$$

$V(y)$ represents the set of free variables in the current partial covering constraint.

MC PRIMAL GREEDY-LIKE PROCEDURE

```
V := Union Nj for ∀j∈R;
y := 1; f := 1;
repeat
    for j∈F if aj > Σ_{i∈F} aiyi - b then fj=0;
    if F≠∅ then
        begin
        for j∈F compute pj := (|V|-|V/Nj|)/aj;
        k := MAX_{j∈F} pj, in case of a tie MAX aj;
        yk := 0; fk := 0; V := V/Nk;
        end
until F=∅
stop.
```

MC DUAL GREEDY-LIKE PROCEDURE

```
V := ∅; y := 0; f := 1;
repeat
    for j∈F compute pj := (|Union(V,Nj)|-|V|)/aj;
    k := MIN_{j∈F} pj, in case of a tie MAX aj;
    yk := 1; fk := 0;
    V := Union(V,Nk);
until F=∅
stop.
```

7. Evaluation of the Accelerated Covering Relaxation Algorithm

The development of an algorithm for a mathematical optimization problem is often an iterative process. Once an initial algorithm has been developed a correct and reliable implementation is required to investigate the average behaviour of the algorithm. Analytic techniques for determining average and worst-case behaviour tend to fail because the algorithm is almost always too complex to obtain any valuable results. If the experimental results show the initial algorithm to be promising, then the development process can advance to the next stage in order to assess the effectiveness of strategies and combinations of strategies that can be incorporated to improve the performance of the algorithm. This stage is repeated until either the performance of the algorithm becomes satisfactory or until the algorithm designers abandon the project for lack of effective strategies.

If empirical techniques are used to investigate the properties of the algorithm, then the algorithm's performance will depend on the class of problems on which the algorithm is tested. Average behaviour is, therefore expressed with respect to a particular class of problems. Truly average behaviour is difficult to determine empirically because the concept of a random problem is vague. Ideally, the class of problems on which the algorithm is tested is representative of real world problems.

In view of these observations, careful design and complete

and accurate reporting of computational experiments are particularly important aspects of the development of mathematical programming algorithms and software. Crowder et al [4] have recommended a set of guidelines for designing and presenting computational experiments. Their survey of the professional journals found the standards in this area to be inadequate. This thesis adheres where possible to the recommendations of [4].

7.1 Aspects of the Implementation

The following sections supply the pertinent information about the implementation of the covering relaxation algorithm for the PP problem which will allow the reader to correctly interpret the experimental results that will be discussed in section 7.3.

In keeping with the time-honoured and widespread convention in this field of scientific endeavour, the program was written in FORTRAN[1]. The reader will pardon the author for omitting the usual justifications.

The program consists of more than fifty functions and subroutines. The source code, including comments, is roughly 6000 lines long. During the course of the experiments, over 2000 randomly generated problems were solved. All known bugs have been removed from the program.

---

[1] For portability, ANSI FORTRAN was employed. The program has been tested on IBM and DEC machines.

The Purpose and Calling Hierarchy of the Major Procedures

1. PP-OPT solves a PP problem to optimality.

Inputs: PP problem; approximate solution; value of an approximate solution which can be used as a lower bound.

Output: optimal solution for the PP problem.

1.1 CR-GENERATOR generates an augmented CR problem if the solution for the current CR problem is infeasible to the original PP problem. Covering constraints are selected from a pool of constraints which is formed by repeatedly invoking SOLVE-MC.

Input: approximate or optimal solution for the current CR problem.

Outputs: logical variable which is true if the solution to the current CR problem is feasible to the original problem; augmented CR problem.

1.1.1 SOLVE-MC collects a small set of solutions for the MC problem using an implicit enumeration.

Inputs: posynomial constraint; integer variable indicating the maximum number of solutions that will be examined.

Output: set of the k best covering constraints found in the search of the solution tree. k is a constant.

1.1.2 ADD-CC adds a covering constraint to the CR problem and removes dominated constraints.

Input: covering constraint.

Output: updated CR problem.

1.1.3 ELCOLS finds the variables whose values can be determined in advance and eliminates them from the CR problem.

Input: CR problem.

Output: reduced CR problem.

1.2 SC-APPRO finds an approximate solution for a SC problem using the greedy-like procedures described earlier.

Input: CR problem.

Output: approximate solution for the CR problem.

1.3 SC-OPT finds an optimal solution for a SC problem using an implicit enumeration.

Inputs: CR problem; lower bound; starting node.

Outputs: optimal solution for the CR problem; first feasible solution.

2. PP-APPRO finds an approximate solution for a PP problem using a greedy-like procedure.

Input: PP problem.

Output: approximate solution for the PP problem.

Generating Covering Constraints

Various strategies can be used in CR-GENERATOR for selecting the covering constraints that are to be added to the current CR problem. Firstly, it is possible to set a lower bound on the number of covering constraints that CR-GENERATOR will attempt to add to the current CR problem. If only a few posynomial constraints are violated, then it is often advantageous to add more than one constraint from a violated posynomial constraint. Secondly, it is possible to vary the manner in which the covering constraints are selected from the pool of constraints generated by SOLVE-MC:

1) add at least one covering constraint from each violated posynomial constraint,
2) add the k covering constraints with the smallest cardinalities from the pool, where k is the maximum of the earlier mentioned lower bound and the number of violated posynomial constraints.

Some other possibilities such as choosing constraints at random were found to be ineffective.

SOLVE-MC uses a fixed branching strategy for implicitly enumerating the solution tree of a MC problem in a depth-first manner. SOLVE-MC collects the five best solutions encountered. Observe that if the first solution encountered is an optimal solution, then no worse solution can be collected. The ordering of the variables is such that $a_k \leq a_{k+1}$, where a_k is the

coefficient of the kth term, or alternatively, such that $a_k/|N_k| \leq a_{k+1}/|N_{k+1}|$, where N_k is the set of variables in the kth term.

The search of the solution tree terminates after Max_s solutions have been found. Max_s is a parameter of SOLVE-MC. Max_s=1 in the basic algorithm.

Solving the SC Problem

SC-OPT uses a fixed branching strategy for implicitly enumerating the solution tree of a SC problem in a depth-first manner. The ordering of the variables is determined by their cost-coefficients. No surrogate constraints or subgradient optimization techniques (see for example [1,5,18]) are employed. SC-OPT returns the optimal and first feasible solutions for the SC problem.

A lower bound, supplied by SC-APPRO, curtails the search for the optimal solution. Furthermore, the first feasible solution from the previous CR problem solved to optimality, if there is one, is used as a starting node. No nodes that occur in the solution tree before the first feasible solution can have a feasible completion, since every CR problem is a tighter relaxation than each of its predecessors, and will, therefore, not have to be examined.

The special structure of the covering constraints is not used to exploit the boolean operations available at the machine language level. SC-OPT represents the set of covering constraints by a two-dimensional integer array containing only

0's and 1's. (This array occupies roughly 60% of the memory space required by the data structures of the program.) CPU time and memory space requirements can be reduced by an order of magnitude if all bits in a word are used, see [16]. The techniques described in [16] are, unfortunately, machine dependent and complicate the task of the implementor considerably. Moreover, the use of these kinds of techniques tends to shift the attention away from the algorithmic aspects of the problem and is, therefore, inappropriate given the status of the research reported in this thesis.

7.2 Problem Generation

An essential tool for the analysis of the covering relaxation algorithm is a problem generator which is needed to supply a class of reproducible problems for which the performance of the algorithm can be determined. This section describes the generator of PP problems and its parameters.

PROBLEM PARAMETERS

alpha is a measure of the tightness of a constraint

N is the number of variables

M is the number of constraints

P is the maximum number of terms in a constraint

E is the maximum number of factors in a term

seed initializes the random number generator

Uniform(s,t) returns an integer randomly distributed between s and t.

```
PP PROBLEM GENERATOR ( seed, N, M, P, E, alpha)
integer P, E;
integer N, M, seed;
real    alpha, 0 ≤ alpha ≤ 1;

begin
initialize Uniform(seed);
c_0 := 1;
for j:=1 to N do c_j := c_{j-1} + Uniform(0,10);
for i:=1 to M do
    begin
    p(i) := Uniform(1,P);
    for j:=1 to p(i) do
        begin
        a_ij := Uniform(1,20)
        N_ij := ∅;
        for k:= 1 to Uniform(1,E) do
          N_ij := Union(N_ij,{Uniform(1,n)});
        N_ij is the set of variables in term ij
        end
        b_i := Σ_{j=1}^{n} a_ij;
    end
end.
```

7.3 Computational results

The major objective of this section is to show that the modifications of the original covering relaxation algorithm that were presented in chapter 5 have significantly improved the performance of the covering relaxation approach. These improvements are mainly due to a 70% reduction in the number of SC problems that have to be solved to optimality in order to produce an optimal solution to a PP problem.

Experimental design

To evaluate the modifications, various methods, to be described shortly, were tested for the following class of PP problems:

N	M	P	E	alpha
30	30	(2,5, ... 17,20)	(1,5,9)	0.5
40	40	(2,5, ... 17,20)	(1,5,9)	0.5

The choice of this particular class was determined by a number of different factors:

The modifications that are incorporated in the accelerated covering covering relaxation algorithm were primarily developed to reduce the number of SC problems that have to be solved to optimality. It is necessary, therefore, to evaluate their effectiveness for PP problems for which an optimal solution is produced at the cost of solving many SC problems to optimality.

The extensive experiments performed in [10] show that

1) the average number of terms in a posynomial constraint is the dominant parameter affecting the number of SC problems that have to be solved to optimality; this number appears to increase linearly with P
2) the overall CPU time required to solve a PP problem increases dramatically as N becomes large, (This increase is, of course, due to an exponential increase in the size of the solution spaces of the SC problems to be solved.)
3) PP problems with alpha = 0.5 are the most difficult.

Figure 1. clearly shows that the size of the PP problems that can be solved is bounded by the difficulty of the SC problems. The following conclusions can be drawn:

1) N is a secondary factor in the evaluation of the accelerated algorithm.
2) P, however, is a primary factor.
3) The performance of the covering relaxation algorithm for PP problems with large N will mainly be improved by employing a more sophisticated technique for solving the SC problems.

<u>Performance indicators</u>

The following indicators are used to assess the performance of the algorithm:

CPU is the execution time (seconds) required to solve a PP problem.

The time required to generate the problem and to obtain

an approximate solution is excluded from CPU. A negligible amount of I/O time is included. The measurements were obtained on an AMDAHL 470 V/6-II under MTS using the FORTRAN H optimizing compiler. CPU was found to vary about 1% with the load of the installation.

CR-CPU is the total execution time required for solving the SC problems to optimality.

OPT is the number of SC problems that were solved to optimality in order to solve a PP problem.

HEUR is the number of SC problems solved approximately.

COVERS is the number of covering constraints in the final SC problem solved to optimality.

DENSITY is the density of the final SC problem.

EFFECTIVENESS is the average of ((best approximate value)(optimal value))100% for all sC problems that were solved to optimality.

Methods investigated

To evaluate the various modifications and strategies that were discussed in chapters 5, 6 and 7 the following methods were tested:

METHOD I is the basic covering relaxation algorithm discussed in chapter 3. One covering constraint is generated from each violated posynomial constraint and $Max_S=1$.

METHOD II is the same as METHOD I except that modification 1

discussed in chapter 5 is incorporated.

METHOD III is the same as METHOD II except that modification 2 discussed in chapter 5 is also incorporated.

METHOD IV is the same as METHOD III except that

(1) the terms in a posynomial constraint are ordered such that $a_{k+1}/|N_{k+1}| > a_k/|N_k|$ and

(2) $Max_s=5$, i.e. the enumeration of the MC problems terminates after 5 solutions have been encountered. (See section 7.1)

METHOD IVA is the same as METHOD III except that Max_s=infinity, i.e. the enumeration of the MC problems is not restricted.

METHOD V is the same as METHOD IVA except that the k covering constraints with the smallest cardinalities are selected. k is set equal to the number of violated posynomial constraints. (See section 7.1)

METHOD VA is the same as METHOD V except that k is set to $\max\{k,(0.2M)\}$.

METHOD VAA is the same as METHOD VA except that the "weighting factor" has been dropped from the approximate procedures for the SC problem that were described in section 6.1.

Conclusions

The following conclusions are drawn from the tables and the figures that can be found at the end of this chapter. Each entry in the six tables was obtained by computing the average of

ten samples.

Table I assesses the improvements obtained by (i) solving an SC problem to optimality only when its approximate solution is feasible to the PP problem to be solved (i.e. modification 1) and (ii) using an approximate solution for the PP problem to generate CR_0 (i.e. modification 2). Table I shows that

1) modification 1 reduces the number of SC problems to be solved to optimality by 65% and the overall CPU time by 50%; however, modification 1 increases the number of iterations (as measured by the number of SC problems solved to optimality for METHOD I and approximately for METHOD II) and the number of covering constraints in the final SC problem
2) additional improvements are realized by applying modification 2; explicitly, the number of covering constraints in the final SC problem and the number of SC problems solved approximately are reduced.

Table II assesses the effectiveness of the various strategies for generating and selecting covering constraints. Table II shows that

3) for large values of P, modification 3 reduces the overall CPU time and the number of SC problems solved approximately
4) the performance of the algorithm is not improved by solving the MC problems to optimality (When P is very large this may no longer be true)
5) the density of the SC problems is reduced, as expected, by selecting minimal cardinality covering constraints.

Table III shows that

6) substantial improvements are realized by generating at least .2M minimal covering constraints at each iteration; explicitly, both the number of SC problems solved to optimality and the overall CPU time are reduced.

Table IV shows that

7) PP problems with larger E are easier to solve.

Table V shows that

8) the SC problems become more difficult to solve if P is increased to 30; however, the number of SC problems to be solved to optimality appears to grow linearly with P.

Table VI shows that

9) the "weighting factor" improves the effectiveness of approximate procedures for the SC problem; more significantly, the overall performance of the accelerated covering relaxation algorithm improves even if only a small improvement in the effectiveness of the approximate procedures for the SC problem is obtained.

In summary, all three modifications have improved the performance of the covering relaxation approach. METHOD VA, the best accelerated covering relaxation method, requires roughly 70% less CPU time and solves roughly 70% fewer SC problems to optimality than METHOD I, the basic covering relaxation method. Figure 2. illustrates the extent to which the covering relaxation approach has been improved. Further improvements can be expected if more effective approximate procedures are employed to solve the SC problem, see (9) above.

Table I: alpha = 0.5, E = 5.

mxn	P	METHOD I				METHOD II				METHOD III			
		CPU	OPT.	HEUR.	COVERS	CPU	OPT.	HEUR.	COVERS	CPU	OPT.	HEUR.	COVERS
30x30	2	0.1	1.0	-	11.6	0.1	1.0	1.0	11.6	0.1	1.0	1.0	10.4
	5	0.3	2.3	-	18.7	0.3	1.1	2.6	19.1	0.3	1.1	2.0	17.7
	8	1.1	4.5	-	27.9	0.9	1.6	5.1	28.4	0.7	1.3	3.9	25.0
	11	2.0	5.7	-	37.8	1.2	1.5	6.0	38.4	1.0	1.1	4.7	36.4
	14	2.6	6.4	-	50.2	1.8	1.6	7.0	52.0	1.6	1.7	4.8	43.4
	17	8.1	10.5	-	78.9	5.5	3.1	12.3	78.6	5.3	2.8	9.1	67.9
	20	10.0	10.9	-	76.7	6.2	3.2	12.0	78.7	5.4	2.9	9.5	70.9
40x40	2	0.2	1.0	-	16.8	0.3	1.0	1.0	16.8	0.3	1.1	1.2	14.3
	5	3.8	3.2	-	25.4	3.2	1.1	3.5	25.8	2.0	1.0	2.2	23.1
	8	11.8	5.0	-	40.2	6.8	1.6	6.0	41.9	4.4	1.3	3.8	36.1
	11	21.3	5.9	-	50.4	9.7	1.6	7.4	51.8	6.6	1.4	5.0	46.5
	14	44.2	8.5	-	70.7	27.0	3.2	9.6	76.8	18.7	2.5	8.0	67.6
	17	71.6	10.3	-	100.6	38.1	3.5	13.2	109.9	38.0	3.5	11.0	99.8
	20	84.7	11.5	-	99.1	45.3	4.0	14.6	106.1	41.5	3.8	11.3	93.9

Table II: alpha = 0.5, E = 5.

mxn	P	METHOD III					METHOD IV					METHOD IV A					METHOD V				
		CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY
30x30	2	0.1	1.0	1.0	10.4	0.09	0.1	1.0	1.0	10.4	0.09	0.1	1.0	1.0	10.4	0.09	0.1	1.0	1.0	10.3	0.09
	5	0.3	1.1	2.0	17.7	0.13	0.3	1.1	1.7	17.9	0.13	0.3	1.1	1.7	17.9	0.13	0.3	1.1	1.8	18.0	0.13
	8	0.7	1.3	3.9	25.0	0.17	0.7	1.3	3.6	25.4	0.16	0.7	1.3	3.6	25.4	0.16	0.7	1.3	3.6	25.2	0.16
	11	1.0	1.1	4.7	36.4	0.21	0.8	1.1	4.3	36.7	0.21	0.8	1.1	4.3	36.7	0.21	0.8	1.1	4.6	36.9	0.20
	14	1.6	1.7	4.8	43.4	0.23	1.1	1.4	4.1	45.6	0.23	1.1	1.4	4.1	45.6	0.23	1.1	1.5	4.1	42.5	0.22
	17	5.3	2.8	9.1	67.9	0.29	4.2	2.3	7.2	69.1	0.28	4.2	2.3	7.2	69.1	0.28	3.6	2.6	7.9	62.0	0.26
	20	5.4	2.9	9.5	70.9	0.31	4.4	2.9	6.9	66.9	0.30	4.2	2.7	6.9	66.7	0.30	3.8	2.8	7.2	63.9	0.28
40x40	2	0.3	1.1	1.2	14.3	0.07	0.3	1.1	1.2	14.3	0.07	0.3	1.1	1.2	14.3	0.07	0.3	1.1	1.2	14.3	0.07
	5	2.0	1.0	2.2	23.1	0.10	2.0	1.1	2.1	23.3	0.10	2.0	1.1	2.1	23.3	0.10	1.9	1.1	2.1	23.3	0.10
	8	4.4	1.3	3.8	36.1	0.14	4.6	1.3	3.9	37.8	0.13	4.5	1.3	3.9	37.8	0.13	4.5	1.3	4.0	37.9	0.13
	11	6.6	1.4	5.0	46.5	0.16	6.5	1.4	5.1	45.9	0.16	6.5	1.4	5.1	45.9	0.16	5.8	1.3	4.5	44.4	0.16
	14	18.7	2.5	8.0	67.6	0.19	19.0	2.9	7.7	69.1	0.19	19.0	2.9	7.7	69.1	0.19	11.6	2.8	6.2	61.0	0.18
	17	38.0	3.5	11.0	99.8	0.24	28.1	2.7	8.3	95.0	0.23	28.2	2.7	8.3	95.0	0.23	30.2	3.6	8.7	85.1	0.21
	20	41.5	3.8	11.3	93.9	0.25	34.3	3.6	9.3	93.5	0.24	34.8	3.6	9.4	94.9	0.24	32.2	3.8	9.4	86.8	0.22

Table III: alpha = 0.5, E = 5.

mxn	P	METHOD V					METHOD V A				
		CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY
30x30	2	0.1	1.0	1.0	10.4	0.09	0.1	1.0	1.0	10.4	0.09
	5	0.3	1.1	1.8	18.0	0.13	0.3	1.1	1.7	17.7	0.13
	8	0.7	1.3	3.6	25.2	0.16	0.6	1.1	3.1	26.2	0.16
	11	0.8	1.1	4.6	36.9	0.20	0.8	1.1	3.6	37.6	0.21
	14	1.1	1.5	4.1	42.5	0.22	1.1	1.3	3.6	44.6	0.22
	17	3.6	2.6	7.9	62.0	0.26	2.1	2.4	7.1	66.7	0.27
	20	3.8	2.8	7.2	63.9	0.28	2.6	2.2	6.7	67.5	0.28
40x40	2	0.3	1.1	1.2	14.3	0.07	0.3	1.1	1.2	14.3	0.07
	5	1.9	1.1	2.1	23.3	0.10	2.0	1.1	2.1	23.4	0.10
	8	4.5	1.3	4.0	37.9	0.13	4.8	1.3	3.6	39.4	0.14
	11	5.8	1.3	4.5	44.4	0.16	7.2	1.5	5.1	49.8	0.16
	14	11.6	2.8	6.2	61.0	0.18	13.6	2.4	6.0	70.4	0.18
	17	30.2	3.6	8.7	85.1	0.21	24.0	2.9	8.5	98.1	0.22
	20	32.2	3.8	9.4	86.8	0.22	25.8	3.0	8.9	100.9	0.23

Table IV: alpha = 0.5.

mxn	P	METHOD VA E = 1					METHOD VA E = 5					METHOD VA E = 9				
		CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY	CPU	OPT.	HEUR.	COVERS	DENSITY
30x30	5	0.2	1.1	1.6	11.9	0.07	0.3	1.1	1.7	17.7	0.13	0.2	1.2	1.8	16.8	0.17
	8	0.4	1.2	2.5	24.9	0.09	0.6	1.1	3.1	26.2	0.16	0.3	1.0	2.0	19.6	0.22
	11	0.9	1.3	4.0	48.2	0.11	0.8	1.1	3.6	37.6	0.21	0.4	1.2	3.0	26.3	0.26
	14	5.6	2.9	9.4	88.8	0.13	1.1	1.3	3.6	44.6	0.22	0.5	1.3	3.3	28.6	0.30
	17	8.7	3.6	12.5	111.8	0.14	2.1	2.4	7.1	66.7	0.27	0.7	1.4	3.3	31.2	0.31
	20	22.9	3.8	19.5	174.2	0.16	2.6	2.2	6.7	67.5	0.28	0.8	1.4	4.0	40.8	0.37
40x40	5	0.3	1.0	1.4	17.3	0.05	2.0	1.1	2.1	23.4	0.10	0.5	1.1	1.8	19.8	0.14
	8	1.4	1.2	2.7	36.6	0.06	4.8	1.3	3.6	39.4	0.14	1.1	1.3	2.5	26.7	0.17
	11	6.4	1.3	4.4	63.7	0.08	7.2	1.5	5.1	49.8	0.16	2.3	1.4	3.5	35.6	0.21
	14	24.1	2.0	8.5	119.6	0.10	13.6	2.4	6.0	70.4	0.18	1.7	1.3	4.0	39.1	0.25
	17	52.7*	4.3*	12.8*	164.3*	0.12*	24.0	2.9	8.5	98.1	0.22	3.2	2.3	4.4	46.5	0.27
	20	+	+	+	+	+	25.8	3.0	8.9	100.9	0.23	2.7	2.1	5.5	60.1	0.32

* In two of the ten runs the optimal solution was not obtained after 150 seconds of CPU time. The average is of the other eight runs.

+ No attempt was made to solve these problems.

Table V: alpha = 0.5, E = 5.

mxn	P	METHOD V A				
		CPU	OPT.	HEUR.	COVERS	DENSITY
30x30	2	0.1	1.0	1.0	10.4	0.09
	5	0.3	1.1	1.7	17.7	0.13
	8	0.6	1.1	3.1	26.2	0.16
	11	0.8	1.1	3.6	37.6	0.21
	14	1.1	1.3	3.6	44.6	0.22
	17	2.1	2.4	7.1	66.7	0.27
	20	2.6	2.2	6.7	67.5	0.28
	23	3.1	3.1	8.5	70.7	0.29
	26	7.7	4.6	11.2	95.5	0.32
	30	14.3	4.4	13.1	103.0	0.36
40x40	2	0.3	1.1	1.2	14.3	0.07
	5	2.0	1.1	2.1	23.4	0.10
	8	4.8	1.3	3.6	39.4	0.14
	11	7.2	1.5	5.1	49.8	0.16
	14	13.6	2.4	6.0	70.4	0.18
	17	24.0	2.9	8.5	98.1	0.22
	20	25.8	3.0	8.9	100.9	0.23
	23	31.7	3.7	10.3	108.0	0.25
	26	63.8	5.2	13.2	148.6	0.27 +
	30	64.2	4.8	14.7	142.6	0.31 +

+ In one of the ten runs the optimal solutions was not obtained after 200 seconds of CPU time. The average is of the other nine runs.

Table VI: alpha = 0.5.

mxn	P	Method VA					Method VAA				
		CPU	OPT.	HEUR.	COVERS	EFFECTIVENESS	CPU	OPT.	HEUR.	COVERS	EFFECTIVENESS
30x30	2	0.1	1.0	1.0	10.4	99.9	0.1	1.0	1.0	10.4	99.6
	5	0.3	1.1	1.7	17.7	99.5	0.3	1.1	1.9	18.0	99.2
	8	0.6	1.1	3.1	26.2	99.5	0.7	1.4	3.4	26.8	99.3
	11	0.8	1.1	3.6	37.6	98.5	0.8	1.3	3.7	36.5	98.4
	14	1.1	1.3	3.6	44.6	98.7	1.1	1.6	3.8	46.9	98.3
	17	2.1	2.4	7.1	66.7	98.7	2.2	2.5	7.6	69.0	98.1
	20	2.6	2.2	6.7	67.5	98.5	2.6	2.5	7.2	68.6	97.6
40x40	2	0.3	1.1	1.2	14.3	99.8	0.3	1.1	1.2	14.3	99.6
	5	2.0	1.1	2.1	23.4	99.7	2.1	1.1	2.4	23.8	99.3
	8	4.8	1.3	3.6	39.4	98.7	4.8	1.4	4.1	40.8	98.5
	11	7.2	1.5	5.1	49.8	98.9	9.0	1.7	4.5	48.2	98.4
	14	13.6	2.4	6.0	70.4	98.3	20.6	3.4	7.3	72.9	98.1
	17	24.0	2.9	8.5	98.1	98.7	25.9	3.0	9.3	98.1	98.0
	20	25.8	3.0	8.9	100.9	98.8	33.9	3.4	10.6	110.0	98.4

Figure 1. CPU Time Use

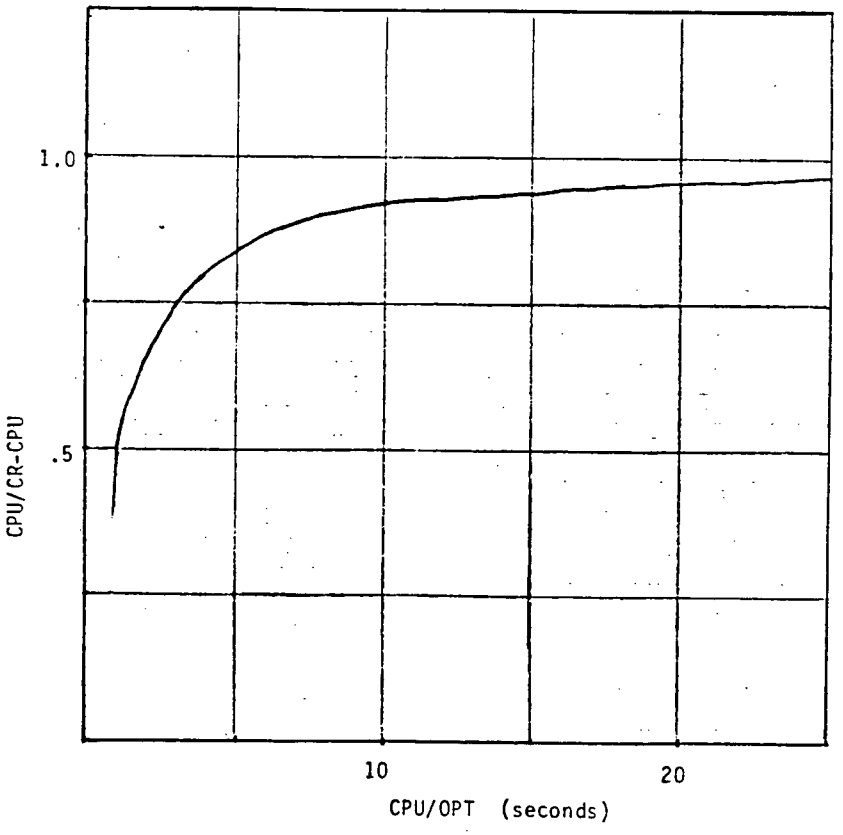
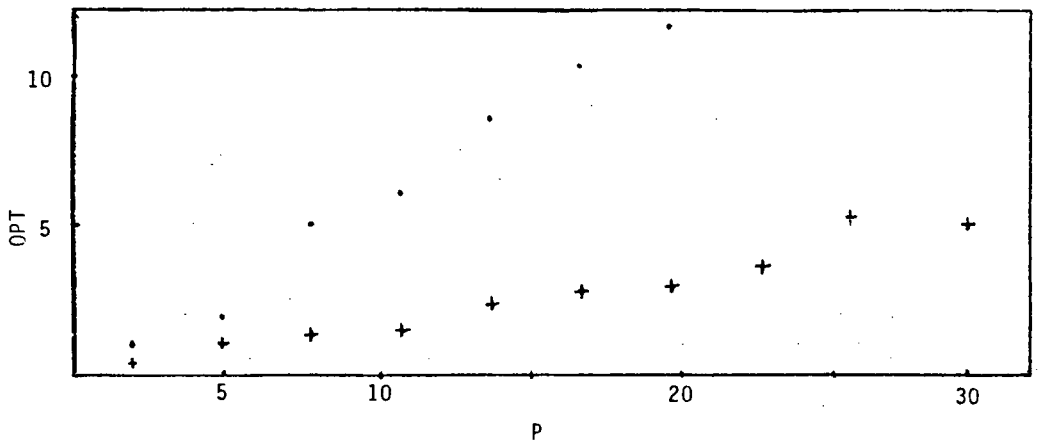


Figure 2. Method I vs Method VA

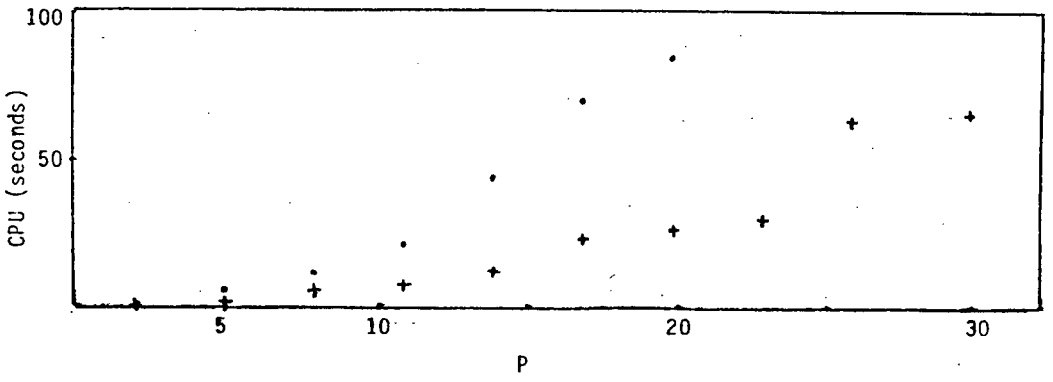
N = 40, M =40, E = 5, alpha = 0.5.



Dots for Method I,

Plusses for Method VA.

Data obtained from tables I and V.



8. Bibliography

1. E. Balas and A. Ho, "Set Covering Algorithms Using Cutting Planes, Heuristics and Subgradient Optimization: A Computational Study", WP No. 6-79-80, GSIA, July 1979.

2. E. Balas and R. Jeroslow, "Canonical Cuts on the Unit Hyperplane", MSR report No. 196(R), Carnegie Mellon University, August-December 1969, revised February 1971.

3. E. Balas and C. H. Martin, "Pivot and Complement - A Heuristic for 0-1 Programming", MSR report No. 414, Carnegie Mellon University, February 1978.

4. H. P. Crowder, R. S. Dembo and J. M. Mulvey, "Reporting Computational Experiments in Mathematical Programming", Mathematical Programming, Vol. 15 (1978), pp. 316-329.

5. J. Etcheberry, "The Set-Covering Problem: A New Implicit Enumeration Algorithm", Operations Research , Vol. 25 (1977), pp. 760-772.

6. B. Faaland and F. S. Hillier, "Interior Path Methods for Heuristic Integer Programming Procedures", Operations Research, Vol. 27 (1979), pp. 1069-1087.

7. R. S. Garfinkel and G. L. Nemhauser, Integer Programming, Wiley 1972.

8. F. Glover and E. Woolsey, "Further Reduction of 0-1 Polynomial Programming Problems to 0-1 Linear Programming Problems", Operations Research, Vol. 21 (1973), pp. 141-161.

9. F. Glover and E. Woolsey, "Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program", Operations Research, Vol. 22 (1974), pp. 180-182.

10. D. Granot, F. Granot and J. Kallberg, "Covering Relaxation for Positive 0-1 Polynomial Programming Problems", Management Science Vol. 25, No. 3 (1979), pp. 264-273.

11. D. Granot and F. Granot, "Generalized Covering Relaxation in 0-1 Programming", WP No. 533, Faculty of Commerce and Business Administration, University of British Columbia, December 1977. (Operations Research, forthcoming)

12. D. Granot and F. Granot, "On Using Heuristics in Benders Partitioning Procedure: I The Linear Case", WP No. 636, Faculty of Commerce and Business Administration, University of British Columbia, August 1979.

13. F. Granot, "Efficient Heuristic Algorithms for Positive 0-1 Polynomial programming problems", WP No. 595, Faculty of Commerce and Business Administration, University of British Columbia, August 1978.

14. F. Granot and P. L. Hammer, "On the Use of Boolean Functions in 0-1 Programming", Methods for Operations Research, XII (1977), pp. 154-184.

15. F. Granot and P. L. Hammer, "On the Role of Generalized Covering Problems", Cahiers du Centre D'Etudes de Recherche Operationelle, Vol. 17 (1975), pp. 275-289.

16. J. S. Graves, "On the Storage and Handling of Binary Data Using FORTRAN with Applications to Integer Programming", Operations Research, Vol. 27 (1979), pp. 534-547.

17. P. L. Hammer and S. Rudeneanu, Boolean Methods In Operations Research and Related Areas, Springer Verlag, 1967.

18. M. Held, P. Wolfe and H. Crowder, "Validation of Subgradient Optimization", Mathematical Programming, Vol. 6 (1974), pp. 68-88.

19. G.A. Kockenberg, B. A. Mccoil and F. P. Wyman, "A Heuristic for General Integer Programming", Decision Sciences, Vol. 5 (1974), pp. 26-44.

20. R. Loulou and E. Michaelides, "New Greedy-like Heuristics for the Multidimensional 0-1 Knapsack Problem", Operations Research, Vol. 27 (1979), pp. 1101-1114.

21. G. Oron, "Optimizing Irrigation System Design", Ph.D. dissertation, Ag. Eng. Dept., Technion Haifa, (1975).

22. D. E. Peterson and D. Laughhunn, "Capital Expenditure Programming and Some Alternative Approaches to Risk", Management Science, Vol. 17 (1971), pp. 320-336.

23. A. Pritsker, L. J. Watters and F. Wolfe, "Multiproject Scheduling With Limited Resources: A Zero-One Programming Approach", Management Science, Vol. 16 (1969), pp. 93-108.

24. M. R. Rao, "Cluster Analysis and Mathematical Programming", Journal of the American Statistical Association, Vol. 66 (1971), pp. 622-626.

25. S. Senjiu and Y. Toyoda, "An Approach to Linear Programming with 0-1 Variables", Management Science, Vol. 15 (1968), pp. B196-207.

26. H. A. Taha, "A Balasian-Based Algorithm for Zero-One polynomial programming", Management Science, Vol. 18 (1969), pp. B398-343.

27. H. A. Taha, "Further Improvements in the Polynomial Zero-One Algorithm", Management Science, Vol. 19 (1972), pp. B196-227.

28. Y. Toyoda, "A Simplified algorithm for Obtaining Approximate Solutions to 0-1 Programming Problems", Management Science, Vol. 21 (1975), pp. 1417-1427.

29. L. J. Watters, "Reduction of Integer Polynomial Problems to Zero-One Linear Programming Problems", Operations Research, Vol. 15 (1967), pp. 1171-1174.

9. Appendix: Three Benchmark PP Problems

This appendix provides complete descriptions of three positive 0-1 polynomial programming problems. Is is hoped that these problems will serve as benchmarks for future research. All three problems were solved by Method VA (see section 7.3). In each case, only a single CR problem had to be solved to optimality. The specifications of these CR problems are also provided in this appendix. It would be valuable to determine how much faster these CR problems can be solved by a sophisticated algorithm for the SC problem.

Problem Parameters

	N	M	P	E	alpha
I	40	40	8	5	0.5
II	40	40	14	5	0.5
III	40	40	20	5	0.5

Performance

	CPU	CR-CPU	OPT	HEUR	COVERS	DENSITY
I	2.5	2.2	1	3	39	0.15
II	17.4	16.4	1	8	79	0.19
III	35.7	34.2	1	9	127	0.25

An example will be used to demonstrate the notation employed for specifing the three benchmark problems.

Consider the following PP and CR problems:

$$\text{Max } 2x_1 + 5x_2 + 5x_3 + 6x_4 + 10x_5$$

st

$$3x_1 + 2x_2x_3 \leq 4$$

$$2x_5 + 2x_3x_4 \leq 2$$

$$11x_1x_2 + 5x_4 + 2x_5 \leq 13,$$

$$\text{Min } 2y_1 + 5y_2 + 5y_3 + 6y_4 + 10x_5$$

st

$$y_1 + y_2 + y_3 \geq 1$$

$$y_3 + y_4 + y_5 \geq 1$$

$$y_1 + y_2 + y_4 \geq 1$$

$$y_1 + y_2 + y_5 \geq 1.$$

These problems will be specified as follows:

<u>Objective Function Coefficients</u>

2 5 5 6 10

<u>Polynomial Constraints</u>

3(1) 2(2 3) ≤ 4

2(5) 2(3 4) ≤ 2

11(1 2) 5(4) 2(5) ≤ 13

<u>SC Problem</u>

11100
00111
11010
11001

PROBLEM I

Objective Function Coefficients

8 15 25 33 35 43 45 53 57 59 65 69 77 80 82 92 100 109 110 119
120 121 121 124 124 124 130 139 145 155 161 168 174 184 192
200 210 220 226 227

Polynomial Constraints

19(24) 19(36 33) 15(40) 10(24 9 1) 5(6) 2(25 17) ≤ 35

15(38 3 2) 15(27 14 8 5) 15(2) 14(29 21 6) 10(21 17 16 4)
8(31 25 22) ≤ 38

19(37 18) 17(15 14) 10(14) 9(8) 8(29 28 27 16) ≤ 31

19(11) 17(33 31 24 22) 12(29) 8(20 8 1) 6(38 19 10 8 1)
3(27 25 14 7) ≤ 32

1(11 9 5) ≤ 0

19(21 11) 19(10 2) 16(22 16 9) 12(36 18 13 10) 11(39 35 17 7)
4(22 17) ≤ 40

14(27 26 18 10 1) ≤ 7

17(32 23 18 13 8) 17(30 28) 14(40 7) 13(36 34 20 8)
10(37 28 16 15 13) 6(17 14 9) ≤ 38

16(35 19 6) 1(11 8 2) ≤ 8

20(39 24 13) 17(36 10 6 1) 13(26 4 3) 11(31) 11(40 33 27)
3(33 14 8) 1(32 11) ≤ 38

16(27 13 12 11 2) 15(22 17 14 2) 8(37 10) 8(39 26) 7(36 28 5)
5(36 29 27) ≤ 29

16(28 27 5 2) 15(33 20 18 5 1) 14(36 20 15) ≤ 22

15(37 26 10) 12(37 25 24 21 4) 10(34) 5(5) 4(34 31 30 26)
≤ 23

14(29 28 27) 14(9 2) 14(19 13 10 1) 13(23) 9(29 8 4)
6(36 30 28 18 5) 6(38) ≤ 38

17(23) 11(39 31 26 21) ≤ 14

19(31 16 12) 17(24 15 10 3) 17(7) 16(35 33 21 20 17) 12(39 8)
10(19 17) 3(21 17 5) ≤ 47

18(29 6) 15(18 12 11 7 1) 15(38 37 31 7 4) 13(34 32 3)
12(26 22 11) 10(33 15) 7(40 27 26 16) 2(29) ≤ 46

17(27 26) 13(30 12 6) 11(9) 9(23 15 7) 1(40 28 22 17 16)
≤ 25

18(23 11) 11(40 19 10 4 2) 9(32 3) 3(35 25 16 2) ≤ 20

11(32 28 23) ≤ 5

19(34 28 20) 18(35 29 27 19 9) 10(32 22 4) 9(11)
4(36 28 24 9 6) 2(23 11) ≤ 31

15(36 9 8) ≤ 7

6(9) ≤ 3

16(24) 16(40 35 23) 14(36 25 16 9) 13(36 14) 12(35 21 14)
12(39 35 10) 6(21 20 1) ≤ 44

18(39 27 16) ≤ 9

8(32 23 15 12) ≤ 4

20(16) 16(35 32 28 26) 9(31 15 7) 3(12) ≤ 24

18(31 25 19) 18(33 22 20 17 15) 17(40 26 16 3) 14(28 24 19 13)
13(33 2 1) 10(33 14 5) 8(27) 1(35) ≤ 49

20(30) 12(34 33 30 19 14) 10(29 20 14 6) 3(33 17) 2(38 20 9)
≤ 23

20(36 34 7) 1(12 11) ≤ 10

20(40 35 16 14 11) 20(28 1) 11(34 1) 9(29 20) 5(39 38 35 27 1)
4(13 8 4) 4(39 36 20) 1(34 27 12 10 2) ≤ 37

20(32 3) 19(36 25 23 18 10) 19(18 17 14 1) ≤ 29

8(26 24 14 8) 3(34 30 29 2) ≤ 5

17(36 33) ≤ 8

17(18 17 3) 16(39 29 14 12) 9(7) ≤ 21

14(36 33 32 24 18) 11(17) ≤ 12

15(40 17) 11(25 23 1) ≤ 13

19(34 25 24) 12(39 34 23 13 7) 10(19) 7(33 15 10) 6(9) 5(27)
1(38 27 16 13 6) ≤ 30

20(38 17) 18(34 26) 15(29) 10(30 27 18 7 1) 8(26 23 9)
4(36 27 23 12 4) 3(15 13 7) ≤ 39

17(27 26 25 14) 12(23 20) 11(33 28 23 20 3) 10(33 16 5)
10(38 36 16 1) ≤ 30

Solutions

Approx. solution: 0010100100 1110100111 1100111111 1101111111
Value of approximation: 3590

Optimal solution: 0010100100 1110100101 1101111111 1101111111
Value of optimal: 3604

Final CR Problem Solved to Optimality

1000000000	0000000001	0000000110	0001000000
1000000100	1000000001	0000000010	0000000000
0100100000	0000100001	0000001100	0000010000
0100100000	1110000000	0000011100	0000010010
0001000000	1000000001	0100000100	0101000000
0000010000	0000000010	0000000000	0000100000
0000010000	0100000000	0000011001	0000000000
0000001000	0000000000	0000000000	0001010000
0000001000	0000100000	0000010100	1100100000
0000000001	0000000000	0000010000	0001001000
0000000100	0001100000	0000000000	0000000000
0000000100	0001000100	0000000000	0000001000
0000000000	0001100100	0000000000	0000001000
0000000000	0000010000	0000001000	0000000010
0000000000	0000010000	0000010100	0100100000
0000000100	0000010100	0000001110	0000001000
0000000100	0010110001	0000000101	0001011000
0000000000	0000001000	0000000000	0000000001
0000000000	0000001000	0000010010	0001000100
0000000000	0000000010	0001101000	0001000000
0000000000	0000000000	0000000000	0010010000
0000001000	0000110000	0000000000	1000000000
0000001100	0100010000	0000000000	1000000010
0000000001	0000100010	0001100000	0011000000
0001100000	0000000000	1001100000	0001001000
0000001000	0010110000	0000000101	0000001001
0111000001	0000010010	0000100000	0100100001
0000001000	0101000000	0000000010	0000000010
1000000001	0000000100	0000011000	0000000000
0000000000	1000000000	0101000000	1010000000
0001000001	0000000000	1001110000	0000001000
0000000000	0010000000	0001001000	1010000011
0011000000	0010000000	0001010000	1000000010
1000000001	0001000001	1001000000	0000100010
1100100000	0000000101	0000001100	0010000000
0100000001	1110000000	0000011000	0000001010
0000100000	0011000010	0001101100	1010000000
1100100000	0001000010	0000101000	1010100000
1100000000	0010000010	0001101100	1010000000

PROBLEM II

Objective Function Coefficients

6 11 20 28 31 38 47 47 52 57 62 65 69 75 76 85 87 97 98 104
104 111 114 117 123 128 133 143 151 157 165 171 176 186 194
203 209 218 218 226

Polynomial Constraints

17(35 16) 16(39 22 17 13 7) 13(23 17) 9(25 19 2) ≤ 27

15(25 23 19 9) 13(18 8 3) 8(27 1) 8(6) 5(9) 5(24 21 19 3)
4(30 29 2) 2(28 23 11 9) 2(5) 1(37 36 18 8) ≤ 31

19(36 29 15 8 1) 19(22) 19(32 31 25 13 7) 17(28) 12(25 10)
11(38 17 16 6) 10(37 35 9 2) 5(40 36 21) 4(1) 4(22)
3(40 16 6 5) 3(25) 3(39 30) ≤ 64

20(29) 20(36 21 15 4) 18(35 28 17) 13(32 9 6) 13(33) 11(38 22)
10(36 16 12 5 3) 9(34 15 3) 9(29) 6(35 28 26 21 10) 3(38)
3(30 25 22 15) 1(32 28 10 2) ≤ 68

11(31 14 11 2) 5(37 31 19) 4(35 28 21 11 6) ≤ 10

20(14 11) 20(20 13 11 5) 18(20 16 13) 17(36 10 7 3)
12(39 34 30 26 19) 11(40 17 15) 9(35 11) 9(40 39 4 3) 8(25)
2(4) 2(38 35 33 26 4) 1(40 11 3 2) ≤ 64

17(28) 15(35 19) 13(16 5) 10(5) 8(38 26 24 23 5) 2(40 36 24)
2(27) ≤ 33

17(24) 14(39 22 10 6) 12(17 2) 11(26 12) 11(31 21 15) 10(18 1)
8(15 10) 8(39 36 33 26 6) 3(32 19 17 11) 2(9 2) 2(8 5)
1(35 28 16 12) ≤ 49

18(34 12) 11(38 32 29 9 4) 11(9) 9(21 20 4 3) 6(29 4)
3(39 27 21 17 8) 2(35 24 23 13 6) 2(32 23 18 14 3)
2(37 31 25 23 7) 2(34 10) 1(39) ≤ 33

20(17 8) 15(29 12 10 7 6) 13(37 29 26 13) 11(23 17 1) 8(19)
7(25 21 19 10) 7(10) 2(40 33 7) ≤ 41

20(26 7) 10(32) 4(40 34 32 15 5) ≤ 17

16(16 8 5) 16(34 23 15 12) 14(33 22 17 12 1) 10(24) 10(29 15)
10(32 29 23 7) 10(37 25 15 12 6) 10(39 38 17 7) 9(28 17 6)
7(39) 4(27 26 8) ≤ 58

17(40) 15(36 33 29 17) 12(38 32 31) 9(21 16 11) 3(40 28 26 17)
3(14 8 7) 2(35 11 7) ≤ 30

11(38 28 19 11) 7(33 31 10 2) 3(33 24 22 19) ≤ 10

18(37 20 13) 17(7) 17(22) 13(37 31 30) 12(5) 11(12 11 6) 9(37)
7(35 19 11) 5(38 28) 4(33 24) 3(37 35 22 14) 3(38 16 15)
3(32 30 23 17 12) ≤ 61

20(19) 18(38 36 34 33 16) 12(25) 4(23 16) 1(39 31 9) ≤ 27

14(30 21 12) 10(25 22 13 5 2) 6(39 36 12 7) 3(40 24 16 10 5)
≤ 16

18(38 4) 16(40 27 25 17) 14(39 24 20 12 3) 12(39 13) ≤ 30

20(39) 17(39 32 28 21) 17(40 37 34) 15(40 39 28 2)
12(33 14 12 7) 10(32 29) 9(26 15) 5(32 29 10 9) 1(24 22)
≤ 53

19(9) 19(39 37 33) 12(8 6) 2(34 30 29 22) 1(34 28 5 2 1)
≤ 26

19(25 13 7) 12(19 18 9) 12(1) 2(30 24 5) ≤ 22

18(38 24) 16(40 15) 10(30 17 9) 6(36 32 19 6) 5(24 14) 5(26)
2(39 32 30 23 18) 1(23 22) ≤ 31

18(36 31 20 19 18) 16(37 24 12) 11(39 33 19 3) 10(37)
9(40 37 19 4 1) 7(38 35 24 20) 6(40) 4(38 30 24 8) 3(13 6)
≤ 42

20(37 29 22 11) 20(13 7 6 4) 20(36 34 33 28 20) 18(20 16)
14(38 27 23 13 10) 11(36 2) 10(28 23 20) 9(32 11 1)
3(32 19 17 7 6) 1(36) ≤ 63

20(9 6) 12(34) 11(28 10 9) 10(34 26 14 1) 7(9) 6(40 14)
4(40 29) 4(24) 1(24 6 3) 1(34 30 22 12) ≤ 38

19(12) 19(36 28 15) 16(12 6 3) 12(35 34) 12(25 19 16 3)
9(40 39 36 8) 8(22 2) ≤ 47

17(20 12) 16(40 24 16 3) 16(12) 16(19) 12(35 34 28 7)
10(40 27 26 11) 9(39) 8(34 6) 8(18 6 5) 2(37 36 30 24)
1(39 35 22 3) 1(39 22 14 8 4) ≤ 58

9(33 21 19) 5(22 12) ≤ 7

20(31 29 28 16 11) 18(24) 17(20) 14(35 25 24 14 10)
13(38 36 12) 12(38 36 29 15) 12(32) 10(28 25 13) 10(29 14 6)
6(30) 2(28) 1(13) 1(6) 1(19 14) ≤ 68

20(30 17 8 4) 19(26 13 8 4) 19(30 13) 18(20) 18(22) 16(18 16)
12(29 27 18 14) 12(32 31 25 13) 10(38 12) 8(38 30 12 4) 7(28)
7(23) 3(40 39 31 19) ≤ 84

19(11) 3(32 31 27 5 4) ≤ 11

20(12) 15(33 13 7) 14(24) 8(39 36 5) 7(35 12) 7(33 30 27 3 1)

7(34 30 26 21 4) 4(35 25 9 6) 1(20) ≤ 41

19(27) 18(22) 17(38 25) 16(25 14) 16(31 30 29) 12(29)
11(28 23 21 5) 11(24) 8(24 5 1) 7(13) 5(33 10) 5(18 13 7)
4(39 27 26 19 15) 1(36 13 10) ≤ 75

18(37 4) 16(28 8 2) 15(29) 14(18 12 1) 10(40 25 24 20 5)
9(29 27) 7(5) 1(29 21 10 7) ≤ 45

19(35 31 25 19 12) 14(35 34 31 21 9) 12(30) 12(36 8 2)
11(33 32 28 27 11) 8(36 34 23 2) 4(34 29) 4(36 32 25) ≤ 42

19(33 6) 14(37 34 27 13) 11(30 11 10) 11(24 19) 10(37)
5(30 18 7 6 3) 5(36 24 12 8 6) 3(19) ≤ 39

18(25 20 18 5) 13(29) 13(36 26 22 1) 11(38 28 22 21) 7(13)
2(32 28) ≤ 32

19(29 26 11) 19(36 16 14 7) 19(32 1) 17(38 37 35 28 27)
13(27 15 3 2) 4(6) 3(38 18 17 14 8) ≤ 47

20(9) 18(20 19) 17(21 13 4 1) 12(30 21) 6(38 11) 2(38 23 12 5)
2(39 23 16 14 4) ≤ 38

18(31 19 8) 16(35 34 27 16 3) 13(31 26) 13(39 32 19)
11(37 18 16) 9(2) 8(31 30 9) 7(38 36 1) 2(35 22 14 2)
1(23 16 15 7) 1(36 32 30) ≤ 49

Solutions

Approx. solution: 001010010 1010110110 1111011100 11111111111
Value of approximation: 3495

Optimal solution: 001010010 1010110110 1111011100 11111111111
Value of optimal: 3495

Final CR Problem Solved to Optimality

1010100000 0100000000 0000001001 0010110010
0100000100 0100000000 0100000000 0001110011
0001000000 0000001000 0000101000 0000000101
0001100000 0100000000 1000010001 0001110010
0011000101 0101001101 1010001000 0101000010
0000010100 0000000000 0000000000 0010001010
0010010100 0100000000 0000000000 0001110011
0010110000 0100000101 0100000000 0001100010
0000010001 0100100000 1100010000 1010010010
0000001000 0000000000 0000010000 0000000000
0000001000 0100000000 1000000001 0000010010
0000101000 0000000000 0100000001 1000001000
0000001100 0001000000 0000000000 1100000101
0000000010 0000000000 0000000000 0010001010

0000000010 0000101000 0110010001 0000000001
0000000111 0100001000 1000001000 0001000010
0000000010 0000101100 0010010001 0100000011
0010000110 0101001100 1010001000 0101000010
0000000000 0011000000 0100101000 0000000100
0000100000 0010000001 0100000001 1000001000
0000000000 0000011000 0010000000 0000100000
0000000000 0000010000 0000100000 0011010100
0000000000 0000000010 0000100000 0000000000
0000000000 0000000010 1000000000 0010000000
0010000000 0100000011 0100000000 0000100010
0010000000 0000000111 0000000000 1010011011
0000000000 0100000001 0001000000 0000100000
0000000000 0000100000 0001000000 0000000101
0000100000 0100000000 0001000000 0000010010
0000000000 0001000000 0101101000 0000000100
0000000000 0100100000 0000000100 0001110000
0000000000 0000000000 1000000100 0101001011
0000000000 0000001000 0000010100 1100000101
0000100000 0001000000 1110101100 0000000100
0000100000 0101001000 0110000101 1100101100
0000000000 0000001000 0000000010 0010010001
0000000000 0001000000 0100101010 0000000100
0000000000 0000100000 0000010010 0101001011
0000000000 0000000010 0000001100 0000100000
1000000000 0010000000 0100010010 0000010000
0101000100 0000000000 0000000110 0000001000
0000000000 0000000000 0101001011 1000000000
0000000000 0000010010 0000000000 0011010100
0000000000 0000000010 0001000100 0000110001
0100000100 0000000010 0000010000 1100000010
0000000000 0010010101 0110000101 0000000000
0100000000 0000000000 0000000110 0101001011
0000000000 0010000001 0100000101 1000001100
0000000100 0000010110 0000010000 1100001010
0001000100 0010000001 0110010101 0000000000
0001000100 0010010101 0100010001 0000000000
1001000000 0000000111 0000000000 1000011001
0000100000 0000010000 0000000100 0000000000
0001100000 0000000000 0000001010 0000001000
0000100000 0010000101 0000100010 0000000000
0010010000 0100100000 0000000100 0000010000
0000000000 0001000000 0000101011 1000000100
0000000000 0000100000 1000010110 0100000010
0100000000 0000000000 1000000110 0100000011
0000010000 0101100001 0000000011 0100010100
0011100000 0100110000 1000000010 0010010000
0000000000 0110100001 0000100111 0100010100
0000010010 0000000000 0000000000 0001000000
0000010110 0000000000 0000000000 0000000000
0011000010 0000000001 1000000010 0100000100
0001000100 0010010101 0010010101 0000000000
0001000100 0011010101 0010011011 0000000000
0001000100 0011010101 0000011111 0000000000

```
0000010000 0010000000 0000001000 0011001000
0000000000 0000000000 0100101011 1000000100
0011000000 0100000001 1000000010 0001000010
0011000001 0100000001 1000000010 0001000000
0001000000 0110000001 0100100001 1100000100
0000100000 0000010000 0011011000 0000010101
1000100001 0010000000 0001001011 1010000000
0110000000 0000010100 0000011001 1101111000
1110000000 0000010100 0000011000 1001111100
0000000001 0010000000 0101101000 0010000100
1000100000 0011000000 0001101000 0000000100
```

PROBLEM III

Objective Function Coefficients

9 16 17 26 36 43 47 55 64 69 76 80 89 97 97 101 102 112 112
118 123 127 137 140 144 145 149 150 152 152 154 157 165 171
179 182 182 186 196 196

Polynomial constraints

15(36) 14(34 30 27 25 20) 14(15) 13(35 18 11 8 7) 7(36 31 21)
4(36 10 7 4) 1(29 15) ≤ 34

20(4 3) 20(34 31 22 20 8) 18(20 18 8) 18(37 36 12 10 6)
18(39 35 31 24) 17(37 34) 17(32 31 19 3 1) 16(27 1)
16(40 14 13) 13(15) 11(39 23 18) 7(34 22 21 9 3) 6(37 16) 4(3)
3(40 10 9) 2(31 22 21 20) ≤ 103

13(19 15) 5(36 13 8) ≤ 9

19(31 30 22 20) 18(35 27 9 4) 6(36 30 9 1) 5(40 36 10 2)
1(40 27 19 17 12) ≤ 24

19(32 12) ≤ 9

18(8) 15(28 27 22 21) 15(38 36 35 3) 14(40 32) 13(35) 10(9 8)
7(25 5) 6(11) 6(25 3) 4(20) 3(38 30 16) ≤ 55

18(11) 17(22 12) 12(39 35 31) 12(34 32 30 8) 11(33 22 19)
10(10 4) 10(15) 10(32 25) 9(29 27) 7(28 15 2) 7(37 23)
6(29 19 12) 5(28) 5(24) 5(32 11 2) 4(12) 4(4) 3(20 8 1)
3(31 13 4 3) 1(37 2) ≤ 79

20(39 33) 20(38 29 11 3) 19(19 7 6 5 2) 17(39) 16(37)
16(33 17) 16(36 19 11 10 8) 15(27 16) 15(30 18) 14(39 29)
14(33 16 8) 12(24 18 16) 12(27 1) 12(40 29 8) 10(28) 9(18)
9(40 31 23 7) 6(26 10) 5(39 24 2) ≤ 128

20(38 22 12 5) 20(13) 15(29 18 11 3 2) 14(20) 11(21 20 18 12)
10(32 24 20 17 6) 6(40 26) 6(26 20 19 15 1) 5(38 36 30 5)
3(27) ≤ 55

20(26 23) 20(12 9 2) 19(39) 17(31 16 12 3) 15(34 26 20 17 4)
11(21 17) 11(24 15) 8(40) 6(27 13) 5(38 34) 4(40 24 21 9 6)
2(36 21 18) 2(32 22 21 10 7) 1(11) ≤ 70

19(24 20) 16(33 29 26 23 20) 15(28 25 17 6 2) 15(38 15)
14(39 33) 11(24) 6(9 1) ≤ 48

19(28 18 17 3 1) 18(9) 18(31 29 28 11) 18(4) 15(36 18 14 5)
14(23 13) 13(25 18 4) 12(25) 12(19 14) 12(31) 9(37 22 17 14)
9(21 20) 7(39 4) 7(38 28 21 15 12) 6(11 8) 6(33 16 12 7 6)
2(39 36 17) ≤ 98

20(36 33) 18(40 17 15) 17(37 34 24 2) 15(39 5) 14(31 9)
11(34 33) 10(39 33 26 15 7) 10(40 4) 8(20) 7(40 24 11)
6(36 31 26 13 8) 4(39 23 2) 1(12) 1(40 7) 1(33 20 13) ≤ 71

13(40 30 15 1) ≤ 6

19(2) 18(21 17 16) 17(23 15 5) 16(32 13) 15(36 9 8 1)
14(32 21 1) 14(34 33 7 3) 13(12 9 2) 12(38 31 27 17 2)
11(30 26 24 15 13) 11(17 14 8) 10(40 16) 9(18 15 13)
6(40 35 23 14 5) 5(39 37 12) 4(17 14 11 1) 3(32 13) 2(36)
≤ 99

11(29 19 11) 10(36 11) 9(28 24 12) ≤ 15

20(36 26 24 6) 19(22) 17(30 26 11 6) 13(31 25 21) 12(35)
12(11) 12(37 33) 11(40 18 16) 10(31 21 10) 9(34 13 9)
7(38 25 8 3) 6(23) 6(39 34 33 28) 3(30 26 14 11 2)
2(39 35 23 12) 2(13) ≤ 80

12(20 6 3) 8(39 26 25 11) 1(29) ≤ 10

15(30 25 22 13 10) 5(23 11 9) ≤ 10

19(19 6) 18(26 24 19 17) 16(27 23 13 6 2) ≤ 26

18(39 37) 15(36 20 16 7) 14(36 27 19 14) 12(38 17 1) 11(40 5)
11(33) 10(35 32 24 5) 10(13 12 11) 9(35 16 14 4)
4(35 32 26 19 9) 3(23 22 20) ≤ 58

17(32 14 12 11) 10(36 18) 8(36 11) 6(35 15 4) 4(19 12 4 3)
≤ 22

17(32 11) 16(23) 11(40 22) 10(39 36 11) 8(33 19 11 10 6)
6(33 21 20 11 4) ≤ 34

16(40 33 26 1) 13(32 22 15) 4(37 8 7) ≤ 16

20(38 32 12 11 3) 15(39 37 3 1) 12(36 13) 9(23 17) 9(28 4)

7(16 7 6 3) 7(3) 5(38 29 19 17 6) 3(40 23) ≤ 43

20(32 19 17 16 14) 19(36 26 24) 18(17 14 13) 16(11 9 5 2)
16(6) 15(40 14 7 3) 15(19) 11(17 11 4 2) 10(28) 9(35 18 1)
9(36 32 18) 6(11) 5(38 36 24 15) 1(38 32 25 7) ≤ 85

19(14 9) 17(8 5 3) 17(29 20) 15(37 35 21 14) 14(27 23 4)
14(34 22 14) 10(39 30 28 19) 10(39 36 35 20) 10(30 26) 9(33)
8(18 15) 5(36 34 2 1) 5(4) 3(22 17 6 5) 2(28 27 21 14) 2(12 4)
1(36) 1(31 21 12 5) ≤ 81

20(20 8) 18(8) 17(37 22 5 3) 17(2) 16(31 23 5) 14(31 30 23)
13(8 2) 12(38 20) 12(35 30 19 13 11) 8(39 36 27 1) 8(25 9 6 5)
4(38 37 25) 3(10) ≤ 81

16(32 28 23 1) 14(38 34 23 11) 12(13) 12(15 8) 10(35 6 5)
7(31 27 12) 3(23 18 15 10 6) 3(30 20 10 1) 2(39 35 2) 1(38 3)
≤ 40

18(15) 17(29 4) 16(31 17 3) 15(4) 11(33 31 14 8) 10(38 28)
9(35 30 26 25 2) 6(39 34 21 2) 6(28) 5(38 5 3 1) 4(26 11)
2(39 37) ≤ 59

18(37 10) 15(25 3) 15(40 33 21 9) 11(31 29 22 16 13)
10(38 35 28 19 3) 4(31 16 15) 2(30 10 9 7) ≤ 37

19(33 24 14) 17(15 14 2) 12(39 33 24 14 4) 12(32 31 19 14 2)
12(9 7 2) 12(1) 11(35 31 14) 6(23 11) 5(22 18 14) 4(24 17)
4(38 37 29 26) 4(29 1) 2(32 24 23 11 10) ≤ 60

19(40 32 28 22 8) 18(34 20 1) 17(37 34 24) 17(33 18 11 4)
14(35 32 31 29 7) 13(30) 10(39 32 26 16 6) 10(32)
7(22 17 16 6 1) 7(29 24 17) 7(18 11 3 1) 4(35 26 24 16 1)
1(36) ≤ 72

20(35 5) 19(37 5) 18(35 19 10 8 3) 18(39 35 26 11 10)
14(37 29 13) 13(37 35 29 2) 11(37 25 7 4) 8(14) 8(40)
7(25 3 2) 7(30) 5(33 32 31 27 14) 5(40 31 16) 2(12)
1(20 19 15) ≤ 78

20(35 13 11) 17(26 3) 17(38 22) 12(17) 12(37 33)
12(27 25 16 15 6) 12(36 33 2) 12(10) 10(37 35 27 25) 6(16 10)
5(36) 4(29 9) 3(37 36 29 18 2) 1(8) 1(18) ≤ 72

20(27 18 16 9) 19(36 15) 17(35 26 15 11) 15(16 5 3) 14(38)
13(36 35 34 31) 13(16) 13(33 14) 10(39 36 13) 10(37 35 16 1)
8(18 3) 4(35 24 18 5) 3(33 31 11) 2(24 12 4) 2(28 19 17)
1(27 20 13 3) ≤ 82

20(16 2) 19(23 1) 19(33 26 20 13 12) 19(40 29 23 10) 18(30)
18(20 17 14) 18(38 37 12 3) 17(39 38 12) 17(33 31 26) 14(9)
13(29 22 19 4) 12(35) 12(22 21 17) 10(39 31 22 12)
10(40 10 4 3 2) 9(32 31 1) 8(35 9) 6(37 20) 6(37 27 20 9)
5(19) ≤ 135

20(35 20 11 4) 6(26 22) ≤ 13

12(39 32 16 6) 9(31 16) ≤ 10

19(22 15 9 8) 19(37 32 26 17 5) 18(11) 17(38 37 16) 16(38)
14(27 1) 13(31 30) 13(34 19) 12(37 32 16 6 2) 10(19) 9(6)
7(40 30 16 9) 5(39 16) 5(9 2) 4(38 17) 3(36 31 13) 2(19 5 3)
1(37 29 19 16) 1(39 37 20 15) ≤ 94

Solutions

Approx. solution: 1000001010 0011000110 1111111111 1111111111
Value of approximation: 3717

Optimal solution: 0000001010 0011011010 1111111111 1111111111
Value of optimal: 3799

Final CR Problem Solved to Optimality

1100001010 0001000010 0001000000 1110100000
1100000010 0001000010 1100010101 0011111010
0010000010 0000000010 1000100100 0010100101
0010000000 0000000000 1100001100 0100110101
1010000010 0011000110 1111001000 1101101011
0001000010 0011000110 0010100000 1000000000
1001000000 0001000100 0101000010 1010100010
0001000010 0001000000 1110011001 0011101000
0001000010 0001000010 1110001101 0011101010
0001000010 0001000010 1110011101 0001101010
0000100000 0011000000 0000001011 1110101001
0000100000 0001000010 0001001000 0110111011
0000010000 0000000000 1111110000 1010111000
0000010010 0010000000 1101110000 1011111000
1000001100 0000000000 0000010000 0010001001
0000000110 0000000000 1100001100 0000100000
0000000110 0000000000 1100001100 0100000001
0000000100 0000000000 1100001100 0100100001
0000001100 0000000000 0101000111 1101101001
1000001100 0000000100 0010001111 1010001011
0000000001 0010000000 0100100001 0000000000
0000000011 0010000000 1110100000 1011101000
0000000000 1000000010 0000000010 0000010000
0000000000 1000000000 0010000000 0100010010
0000000000 1000000000 0110000000 0100000001
1000000000 1010000000 0010000100 0101000100
0000000010 1010000000 1110100000 1011101000
1000000000 1001000100 0111010010 1010101100
0000000000 0100000000 0000000000 0100000000
0000000010 0110000000 1110100100 1011101010
0000000000 0000100010 0000000000 0000000000
1000000000 0000100000 0000000001 0000000001
0000000000 0000100000 1000000000 1000010000

1000000000	0000100000	0100010000	0110000001
0000000000	0010100100	1011011000	0001010111
1000000000	0000010100	0001001011	0010001010
1000000010	0000010010	0000001001	1001001111
1000000010	0001010100	0000001000	1011111100
0000000010	0011010100	0000001000	1011110110
0000000010	0010010100	1110100000	1011101001
1000000000	0000001100	0000001111	0010001010
0000000000	0010001100	1010011000	0001010111
1000000010	0001001010	0000011000	0110111110
1000000010	0000000001	0001000000	0010000010
0000000000	0000000001	0011010010	0010000010
1000000010	0000000001	0100000001	1000010000
0000000000	0000000001	1000101001	1001010000
1000001000	0000000001	0001000011	1101111000
0000000010	0001000001	1100000010	0011111010
0000000010	0001000001	1100010011	0001111010
0000000010	0001000011	1100010111	0001101010
0001000000	0000100000	0000000110	0000000100
0000000000	1100000000	0001000100	0000010000
0011000000	0000101000	0000000010	1000000000
0000000000	1000000000	0110000000	0000010011
0001000000	1000100100	0000000000	0000110000
0010000110	0000000000	0000000000	0000110100
0000001100	1000100100	0000000000	0000110000
0010100010	0001010100	0000001000	0010000100
0010100010	0000110100	0000001000	0000010100
0110100101	0000000000	0110000000	1000001000
0100110110	0000000000	0010100001	1000000000
0100100100	0000000000	0010100001	1000001100
0110100100	0000000000	0110000001	1000001000
0000010000	1000000000	0111010001	0000110000
0100000010	0100100000	0011010000	0000000011
0100000010	1100100000	0011010000	0000000010
0100000010	0100000000	0010010000	0001000111
0110000010	0100010000	0010010000	1000000010
0000000010	1000110100	0000011000	0000110100
0101000000	0001101000	0001000000	1010100010
0101001010	0001101000	0001000000	0010000010
0010001011	0000110000	0000100001	1000001000
0000010000	1000000000	0101010001	0010111000
0100000000	1000101000	0001000000	0011011001
0101000000	0000101000	0001000000	0011011001
0001000110	1000000100	0000100110	1000000010
0000010000	0000010000	0000000000	0100000010
0000010000	0000001010	0001010000	0000000000
0000000110	0000010000	0000000001	0100100101
0001000100	0001000000	0000000110	1010001110
0101000010	0000000000	0001000000	1011011001
0000000000	0000011100	0001001011	0010001010
0000000100	0000011100	0000001011	0010001010
0100010010	0000010010	0000000001	1101001100
0100000001	0000001100	0100000010	0010011100
0100000011	0000001000	0100000010	0010011100

0100000001	0000001000	0100101000	0010111100
0100001010	0001001110	0101000000	1110100000
0100001000	0000001100	0011000111	1010001011
0000001001	0000001100	0010010111	1010001011
0000100000	0110000001	0100001000	0000000100
0010100100	0000000000	0000100000	0000110100
0010000001	0000000010	0000100100	0000101100
0000100000	0110000001	0100010000	0000000101
0000100000	0110000101	1100000000	0000000100
0010000001	0000001000	0100010000	0010011100
0010000001	0000011000	0100010000	0010001100
0010000001	0010010000	0100100010	1000001000
0010100100	0000000000	1100101100	0000100000
0000100100	0000000001	1100101100	0000100000
0100000101	0000000001	0000000000	0000000100
0100000100	0000000001	0000100000	0000001100
0100100100	0000000001	0010000000	1000000000
0000000110	0000000001	0000000000	0100100001
0000100110	0000000000	0000100000	0100100001
0100100000	0011000000	0000000010	0000101001
0000000000	0011011110	0001010100	0100010000
1000100000	0000011010	0000011000	0101001110
1000001000	0001011011	0000001000	0000011110
1000001100	0000000001	0101000110	1101101001
0100000001	0000010000	0100101000	0010111100
0100000000	0000010100	0001001111	0010001010
0000000001	0000010100	1110100000	1010101001
0000000001	0000010100	0001011111	0010001010
0000001000	0000010100	0011001111	1010001011
0100001010	0001000110	0101010010	1110101100
0100001010	0001001010	0001010010	1110101100
0001000010	0000000001	0100001001	1000100000
0001100010	0001000110	0000100000	1000010000
0100100000	0001010000	0000000011	1000101001
0101101000	0001000000	0000100010	0000101001
0000100000	0000011010	0000010001	1101001110
0000100010	0000011010	0000010001	1101001101
0000101000	0001010011	0110001000	0000011011
0001001000	0001010011	0110001000	0000111010
0001101000	0001010001	0001000000	0100111011