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

(c) Willem Vaessen, 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>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Introduction</td>
<td>1</td>
</tr>
<tr>
<td>2. Mathematical Preliminaries</td>
<td>7</td>
</tr>
<tr>
<td>3. The Covering Relaxation Approach</td>
<td>12</td>
</tr>
<tr>
<td>4. Evaluation of the Basic Covering Relaxation Algorithm</td>
<td>19</td>
</tr>
<tr>
<td>5. Modifications of the Covering Relaxation Approach</td>
<td>21</td>
</tr>
<tr>
<td>6. Greedy-like Approximate Procedures</td>
<td>32</td>
</tr>
<tr>
<td>6.1 Greedy-like procedures for the SC Problem</td>
<td>33</td>
</tr>
<tr>
<td>6.2 Greedy-like procedures for the MC Problem</td>
<td>37</td>
</tr>
<tr>
<td>7. Evaluation of the Accelerated Covering Relaxation Algorithm</td>
<td>39</td>
</tr>
<tr>
<td>7.1 Aspects of the Implementation</td>
<td>40</td>
</tr>
<tr>
<td>7.2 Problem Generator</td>
<td>45</td>
</tr>
<tr>
<td>7.3 Computational Results</td>
<td>47</td>
</tr>
<tr>
<td>8. Bibliography</td>
<td>61</td>
</tr>
<tr>
<td>9. Appendix: Three Benchmark Problems</td>
<td>64</td>
</tr>
<tr>
<td>Table</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
</tr>
<tr>
<td>Table I</td>
<td>53</td>
</tr>
<tr>
<td>Table II</td>
<td>54</td>
</tr>
<tr>
<td>Table III</td>
<td>55</td>
</tr>
<tr>
<td>Table IV</td>
<td>56</td>
</tr>
<tr>
<td>Table V</td>
<td>57</td>
</tr>
<tr>
<td>Table VI</td>
<td>58</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 1.  CPU Time Use 59
Figure 2.  Method I vs Method VA 60
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:

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

subject to

$$F_i(x) \leq b_i \quad i=1, \ldots, m$$

$$x_h \in \{0,1\} \quad h=1, \ldots, n$$

where for each $h$, $i$ and $j$

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

$$a_{ij} \geq 0; \quad c_h > 0 \text{ and}$$

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

$N_{ij}$ is the set of variables in $j^{th}$ term of the $i^{th}$ 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 $T_{ij} = \bigoplus_{k \in N_{ij}} x_k$. 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:

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

**Definition 1** A subset \( U \) of \( N = \{1, 2, \ldots, n\} \), 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:

\[
(3) \quad \sum_{k=1}^{p} a_k \bigvee_{j \in N_k} x_j \leq b, \text{ where } a_k > 0, \ k=1, \ldots ,p \text{ and } N_k \subseteq N
\]

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

\[
(4) \quad \sum_{k=1}^{p} a_k z_k \leq b \text{ and } z_k = \bigvee_{j \in N_k} x_j \text{ for } k=1, \ldots ,p.
\]

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

\[
\bigvee_{k \in U} z_k = 0 \text{ for all } U \in C,
\]

equivalently, if and only if

\[
\bigvee_{k \in U} \bigvee_{j \in N_k} x_j = 0 \text{ for all } U \in C.
\]

This condition can also be rewritten as a set of linear covering
inequalities in the complementing variables.

Theorem 2 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.

Definition 3 A covering constraint (5) is prime 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).

Definition 3a The size of $S(U)$ is the cardinality of the covering constraint, i.e. The number of variables occurring in the covering constraint.

Definition 3b The cardinality of a covering constraint (5) is minimal 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.

\[ 8x_2 x_3 + 5x_2 x_4 + 3x_1 x_3 x_5 + 2x_2 x_6 \leq 9 \]

\[
\begin{align*}
&z_1 \quad z_2 \quad z_3 \quad z_4 \\
&z_1 z_2 = 0 \quad \iff \quad x_2 x_3 x_4 = 0 \quad \iff \quad y_2 + y_3 + y_4 \geq 1 \\
&z_1 z_3 = 0 \quad \iff \quad x_1 x_2 x_3 x_5 = 0 \quad \iff \quad y_1 + y_2 + y_3 + y_5 \geq 1 \\
&z_1 z_4 = 0 \quad \iff \quad x_2 x_3 x_6 = 0 \quad \iff \quad y_2 + y_3 + y_6 \geq 1 \\
&z_2 z_3 z_4 = 0 \quad \iff \quad x_1 x_2 x_3 x_4 x_5 x_6 = 0 \quad \iff \quad y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \geq 1
\end{align*}
\]

\[ x_i = 1 - y_i \text{ for } i=1,2,3,4,5,6. \]

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

\[ \sum_{j=1}^{n} x_j \leq T(n/2), \text{ where } T(r) \text{ denotes the integer part of } r, \]
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 (CR₀) 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 $CR_0$ 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:

\[ \frac{P}{k=1} a_k P_k \leq b, \] where \( P_k \) 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 \( P_k \) such that \( a_{k-1} \geq a_k \) for \( k = 2, \ldots, p \);
prime-cover := \emptyset
lhs := 0;
k := 1;
repeat
    lhs := lhs + a_k;
    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 guarantee 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

\[ \text{MAX } 7x_1 + 5x_2 + 4x_3 + 3x_4 + 3x_5 + 2x_6 \]

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 described 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:

\[
\begin{align*}
\text{MIN} & \quad 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6 \\
\text{s.t.} & \quad y_1 + y_2 + y_3 + y_4 \geq 1 \\
& \quad y_1 + y_4 + y_5 \geq 1 \\
& \quad y_1 + y_2 + y_3 + y_5 \geq 1 \\
& \quad y_2 + y_3 + y_4 + y_5 \geq 1 \\
& \quad y_i \in \{0,1\} \text{ for } i=1,2,3,4,5,6.
\end{align*}
\]

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 constraints:

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:

\[
\begin{align*}
\text{MIN} & \quad 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6 \\
\text{s.t.} & \quad y_1 + y_2 + y_3 + y_4 \geq 1 \\
& \quad y_1 + y_4 + y_5 \geq 1 \\
& \quad y_1 + y_2 + y_3 + y_5 \geq 1 \\
& \quad y_2 + y_3 + y_4 + y_5 \geq 1 \\
& \quad y_2 + y_3 + y_6 \geq 1 \\
& \quad y_i \in \{0,1\} \text{ for } i=1,2,3,4,5,6.
\end{align*}
\]
Its solution is $y_1=y_2=y_4=y_6=0$, and $y_3=y_5=1$. 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:

MIN $7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6$

s.t.

\[
\begin{align*}
Y_1 + Y_2 + Y_3 + Y_4 & \geq 1 \\
Y_1 + Y_4 + Y_5 & \geq 1 \\
Y_1 + Y_2 + Y_3 + Y_5 & \geq 1 \\
Y_2 + Y_3 + Y_4 + Y_5 & \geq 1 \\
Y_2 + Y_3 & \geq 1 \\
Y_1 + Y_5 + Y_6 & \geq 1
\end{align*}
\]

$y_i \in \{0,1\}$ 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 interrupted 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].

Modification 2. 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 := ø;
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 one 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.

**Modification 3.** 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 $R=\{i|\forall j \in N_i, x_j(c)=1\}$. $R$ 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:

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

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:
Minimize \(|\bigcup_{i \in S} N_i|\) \(S \subseteq R\) \(i \in S\)

(MC) 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**

1. **find** an approximate solution for the PP problem;
2. **generate** \( CR_0 \) using the approximate solution;
3. **repeat**
   1. solve the current CR problem approximately;
   2. attempt to generate additional cuts to eliminate the approximate solution;
   3. **if** the approximate solution is feasible to the original PP problem **then**
      1. solve the current CR problem to optimality;
      2. attempt to generate additional cuts to eliminate the optimal solution;
      3. **if** no cuts were added **then**
         1. the optimal solution to the CR problem is feasible to the original PP problem;
      **end**
4. **until** the current optimal solution is feasible;
5. **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:

\[
\begin{align*}
\text{MAX} \quad & 7x_1 + 5x_2 + 4x_3 + 3x_4 + 3x_5 + 2x_6 \\
\text{s.t.} \quad & \\
1) \quad & 8x_1x_2x_3 + 6x_2x_4 + 5x_3 + 4x_3x_6 \leq 13 \\
2) \quad & 7x_2x_3x_5 + 2x_1x_6 \leq 6 \\
3) \quad & 7x_4x_5 + 4x_1x_4 + 4x_2x_5x_6 \leq 8 \\
4) \quad & 5x_2x_3 + 5x_1x_5 + 3x_6 + 2x_4 \leq 7 \\
5) \quad & 3x_2x_4 + 2x_3x_4 + 2x_4x_5 + 2x_5x_6 + x_1x_6 \leq 5 \\
x_i \in \{0,1\} \text{ for } i=1,2,3,4,5,6.
\end{align*}
\]
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) \( x_1 x_2 x_3 x_4 = 0 \)

4) \( x_2 x_3 x_6 = 0 \)

5) \( x_1 x_2 x_3 x_4 x_6 = 0 \)

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

from 3) \( x_1 x_4 x_5 = 0 \)

4) \( x_1 x_5 x_6 = 0 \)

5) \( x_2 x_4 x_5 x_6 = 0 \)

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.

\[
\begin{align*}
y_1 + y_2 + y_3 + y_4 & \geq 1 \\
y_2 + y_3 + y_6 & \geq 1 \\
y_1 + y_4 + y_5 & \geq 1 \\
y_1 + y_5 + y_6 & \geq 1 \\
y_2 + y_4 + y_5 + y_6 & \geq 1 \\
y_{i} \in \{0,1\} \text{ for } i=1,2,3,4,5,6.
\end{align*}
\]

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) \( x_1 x_2 x_3 = 0 \)

4) \( x_1 x_2 x_5 = 0 \)
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.

$$\begin{align*}
y_2 + y_3 + y_4 & \geq 1 \\
y_2 + y_3 + y_6 & \geq 1 \\
y_1 + y_4 + y_5 & \geq 1 \\
y_1 + y_5 + y_6 & \geq 1 \\
y_2 + y_4 + y_5 + y_6 & \geq 1 \\
y_i & \in \{0,1\} \text{ for } i=1,2,3,4,5,6.
\end{align*}$$

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 a set of covering constraints to form the initial CR problem.
Iteration 1. Solve the initial CR problem:

\[
\text{MIN } 7y_1 + 5y_2 + 4y_3 + 3y_4 + 3y_5 + 2y_6 \\
\text{s.t.} \\
\begin{align*}
y_1 + y_2 + y_3 & \geq 1 \\
y_1 + y_4 + y_5 & \geq 1 \\
y_2 + y_3 + y_6 & \geq 1 \\
y_2 + y_3 + y_4 + y_5 & \geq 1 \\
y_i & \in \{0, 1\} \text{ for } i=1, 2, 3, 4, 5, 6.
\end{align*}
\]

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 \( x_4 = x_6 = 1 \). \( x^* = (1, 1, 1, 1, 0, 1) \) 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 \((1,1,\ldots,1)\), 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 \((0,0,\ldots,0)\) 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 occurring 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

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

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

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

Define

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

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

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

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

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

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

$$T(y) = \{j \in F | \exists i \text{ (} a_{ij} = 1 \land \forall k \neq i \text{ (} 1 < R_j(f) \leq R_i(f))\})\}, \text{ i.e. } T(y) \text{ is the set of free variables occurring in the tightest unsatisfied covering constraints at } y.$$  

Further let

$$M = \{1, 2, \ldots, m\}; \text{ and let } e \text{ and } e^R \text{ designate suitably small positive constants. Subsequently, } h_i(y), H(y), R_i(f) \text{ and } T(y) \text{ will be denoted by } h_i, H, R_i \text{ and } T \text{ respectively.}$$
SC PRIMAL GREEDY-LIKE PROCEDURE

initialize H, R and T;

\( y := 1; \ f := 1; \)

repeat

\begin{align*}
\text{if slow-option then } & S := F \\
\text{else } & S := T;
\end{align*}

\text{for } j \in S \text{ compute } p_j = c_j/(\sum_{i=1}^{m} a_{ij}/((R_i-1)h_i+e^r))

k := \text{MAX } p_i, \text{ in case of a tie } \text{MIN } c_i;

\text{if } p_k \geq e \text{ then}

\text{begin}
\begin{align*}
& y_k := 0; \ f_k := 0; \\
& \text{for } i:=1 \text{ to } n \text{ if } R_i = 1 \text{ then}
\end{align*}

\text{begin}
\begin{align*}
& j := 0; \\
& \text{repeat } j := j + 1 \\
& \text{until } a_{ij}f_j = 1; \\
& f_j := 0 \\
& \text{update } H;
\end{align*}

\text{end}

\text{update } R \text{ and } T;

\text{end}

\text{until } p_k < e \text{ or } F=\emptyset \text{ or } H=M;

\text{stop.}

"R_i-1" \text{ 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 \in S compute \( p_j = c_j / (\sum_{i=1}^{m} a_{ij} / (R_i h_i + e^r)) \)
    \( k := \text{MIN } p_i, \text{ in case of a tie } \text{MIN } c_i; \)
    if \( p_k \geq e \) 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 $j$th term,

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

as before define

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

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

$V(y)$ represents the set of free variables in the current partial covering constraint.
MC PRIMAL GREEDY-LIKE PROCEDURE

\[ V := \text{Union } N_j \text{ for } \forall j \in \mathbb{R}; \]
\[ y := 1; f := 1; \]
repeat
\[ \text{for } j \in F \text{ if } a_j > \sum_{i \in F} a_i y_i - b \text{ then } f_j = 0; \]
\[ \text{if } F \neq \emptyset \text{ then} \]
\[ \text{begin} \]
\[ \text{for } j \in F \text{ compute } p_j := (|V| - |V/N_j|)/a_j; \]
\[ k := \text{MAX } p_j, \text{ in case of a tie } \text{MAX } a_j; \]
\[ y_k := 0; f_k := 0; V := V/N_k; \]
\[ \text{end} \]
\[ \text{until } F = \emptyset \]
stop.

MC DUAL GREEDY-LIKE PROCEDURE

\[ V := \emptyset; y := 0; f := 1; \]
repeat
\[ \text{for } j \in F \text{ compute } p_j := (|\text{Union}(V,N_j)| - |V|)/a_j; \]
\[ k := \text{MIN } p_j, \text{ in case of a tie } \text{MAX } a_j; \]
\[ y_k := 1; f_k := 0; \]
\[ V := \text{Union}(V,N_k); \]
\[ \text{until } F = \emptyset \]
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.

1.1.3 ELCOLS finds the variables whose values can be determined in advance and eliminates them from the 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.
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 \( \text{Max}_S \) solutions have been found. \( \text{Max}_S \) is a parameter of SOLVE-MC. \( \text{Max}_S = 1 \) 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);
c₀ := 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 := φ;
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 := \sum_{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:

<table>
<thead>
<tr>
<th>N</th>
<th>M</th>
<th>P</th>
<th>E</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>30</td>
<td>(2,5, ...</td>
<td>17,20)</td>
<td>(1,5,9)</td>
</tr>
<tr>
<td>40</td>
<td>40</td>
<td>(2,5, ...</td>
<td>17,20)</td>
<td>(1,5,9)</td>
</tr>
</tbody>
</table>

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

The modifications that are incorporated in the accelerated 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.

Performance indicators

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 \( \frac{(\text{best approximate value} - \text{optimal value})}{100\%} \) 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 Maxs=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) $\text{Max}_g = 5$, 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 $\text{Max}_g = \infty$, 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: \( \alpha = 0.5, E = 5. \)

<table>
<thead>
<tr>
<th>m x n</th>
<th>( p )</th>
<th>METHOD I</th>
<th></th>
<th>METHOD II</th>
<th></th>
<th>METHOD III</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CPU</td>
<td>OPT.</td>
<td>HEUR.</td>
<td>COVERS</td>
<td>CPU</td>
<td>OPT.</td>
</tr>
<tr>
<td>30 x 30</td>
<td>2</td>
<td>0.1</td>
<td>1.0</td>
<td>-</td>
<td>11.6</td>
<td>0.1</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.3</td>
<td>2.3</td>
<td>-</td>
<td>18.7</td>
<td>0.3</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>1.1</td>
<td>4.5</td>
<td>-</td>
<td>27.9</td>
<td>0.9</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>2.0</td>
<td>5.7</td>
<td>-</td>
<td>37.8</td>
<td>1.2</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>2.6</td>
<td>6.4</td>
<td>-</td>
<td>50.2</td>
<td>1.8</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>8.1</td>
<td>10.5</td>
<td>-</td>
<td>78.9</td>
<td>5.5</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>10.0</td>
<td>10.9</td>
<td>-</td>
<td>76.7</td>
<td>6.2</td>
<td>3.2</td>
</tr>
<tr>
<td>40 x 40</td>
<td>2</td>
<td>0.2</td>
<td>1.0</td>
<td>-</td>
<td>16.8</td>
<td>0.3</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>3.8</td>
<td>3.2</td>
<td>-</td>
<td>25.4</td>
<td>3.2</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>11.8</td>
<td>5.0</td>
<td>-</td>
<td>40.2</td>
<td>6.8</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>21.3</td>
<td>5.9</td>
<td>-</td>
<td>50.4</td>
<td>9.7</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>44.2</td>
<td>8.5</td>
<td>-</td>
<td>70.7</td>
<td>27.0</td>
<td>3.2</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>71.6</td>
<td>10.3</td>
<td>-</td>
<td>100.6</td>
<td>38.1</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>84.7</td>
<td>11.5</td>
<td>-</td>
<td>99.1</td>
<td>45.3</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Table II:  alpha = 0.5, E = 5.

| mnx | P | CPU | OPT. | HEUR. | COVERS | DENSITY | 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 |
|      | 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 |
|      | 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 |
|      | 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 |
|      | 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 |
|      | 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 |
|      | 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 |
| 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 |
|      | 5 | 2.0 | 1.0 | 2.2 | 23.1 | 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 |
|      | 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 |
|      | 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 |
|      | 17 | 38.0 | 3.5 | 11.0 | 99.8 | 0.24 | 26.1 | 2.7 | 8.3 | 95.0 | 0.23 | 28.2 | 2.7 | 8.3 | 95.0 | 0.23 |
|      | 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 |

CPU: Central Processing Unit time; OPT: Optimal solution; HEUR: Heuristic solution; COVERS: Number of covers; DENSITY: Density of covers.
Table III: alpha = 0.5, E = 5.

<table>
<thead>
<tr>
<th>m xn</th>
<th>P</th>
<th>METHOD V</th>
<th></th>
<th>METHOD VA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CPU</td>
<td>OPT.</td>
<td>HEUR.</td>
<td>COVERS</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.1</td>
<td>1.0</td>
<td>1.0</td>
<td>10.4</td>
</tr>
<tr>
<td>30x30</td>
<td>5</td>
<td>0.3</td>
<td>1.1</td>
<td>1.8</td>
<td>18.0</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>0.7</td>
<td>1.3</td>
<td>3.6</td>
<td>25.2</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.8</td>
<td>1.1</td>
<td>4.6</td>
<td>36.9</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>1.1</td>
<td>1.5</td>
<td>4.1</td>
<td>42.5</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>3.6</td>
<td>2.6</td>
<td>7.9</td>
<td>62.0</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>3.8</td>
<td>2.8</td>
<td>7.2</td>
<td>63.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>0.3</td>
<td>1.1</td>
<td>1.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>1.9</td>
<td>1.1</td>
<td>2.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>4.5</td>
<td>1.3</td>
<td>4.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
<td>5.8</td>
<td>1.3</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>14</td>
<td>11.6</td>
<td>2.8</td>
<td>6.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17</td>
<td>30.2</td>
<td>3.6</td>
<td>8.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>32.2</td>
<td>3.8</td>
<td>9.4</td>
</tr>
</tbody>
</table>
Table IV: $\alpha = 0.5$.

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

* 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.

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

+ 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.

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


9. Appendix: Three Benchmark PP Problems

This appendix provides complete descriptions of three positive 0-1 polynomial programming problems. It 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.

<table>
<thead>
<tr>
<th>Problem Parameters</th>
<th>N</th>
<th>M</th>
<th>P</th>
<th>E</th>
<th>alpha</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>40</td>
<td>40</td>
<td>8</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>II</td>
<td>40</td>
<td>40</td>
<td>14</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>III</td>
<td>40</td>
<td>40</td>
<td>20</td>
<td>5</td>
<td>0.5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Performance</th>
<th>CPU</th>
<th>CR-CPU</th>
<th>OPT</th>
<th>HEUR</th>
<th>COVERS</th>
<th>DENSITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>2.5</td>
<td>2.2</td>
<td>1</td>
<td>3</td>
<td>39</td>
<td>0.15</td>
</tr>
<tr>
<td>II</td>
<td>17.4</td>
<td>16.4</td>
<td>1</td>
<td>8</td>
<td>79</td>
<td>0.19</td>
</tr>
<tr>
<td>III</td>
<td>35.7</td>
<td>34.2</td>
<td>1</td>
<td>9</td>
<td>127</td>
<td>0.25</td>
</tr>
</tbody>
</table>

An example will be used to demonstrate the notation employed for specifying the three benchmark problems.
Consider the following PP and CR problems:

Max \( 2x_1 + 5x_2 + 5x_3 + 6x_4 + 10x_5 \)

st

\( 3x_1 + 2x_2x_3 \leq 4 \)
\( 2x_5 + 2x_3x_4 \leq 2 \)
\( 11x_1x_2 + 5x_4 + 2x_5 \leq 13, \)

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:

<table>
<thead>
<tr>
<th>Objective Function Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 5 5 6 10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polynomial Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>3(1) 2(2 3) \leq 4</td>
</tr>
<tr>
<td>2(5) 2(3 4) \leq 2</td>
</tr>
<tr>
<td>11(1 2) 5(4) 2(5) \leq 13</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SC Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>11100</td>
</tr>
<tr>
<td>00111</td>
</tr>
<tr>
<td>11010</td>
</tr>
<tr>
<td>11001</td>
</tr>
</tbody>
</table>
PROBLEM I

Objective Function Coefficients

\[
\begin{align*}
8 & 15 & 25 & 33 & 35 & 43 & 45 & 53 & 57 & 59 & 65 & 69 & 77 & 80 & 82 & 92 & 100 & 109 & 110 & 119 \\
120 & 121 & 121 & 124 & 124 & 130 & 139 & 145 & 155 & 161 & 168 & 174 & 184 & 192 \\
200 & 210 & 220 & 226 & 227
\end{align*}
\]

Polynomial Constraints

\[
\begin{align*}
19(24) & 19(36) 33 & 15(40) & 10(24) 9 & 1 & 5(6) & 2(25) 17 & \leq & 35 \\
15(38) & 3 & 2 & 15(27) 14 & 8 & 5 & 15(2) & 14(29) 21 & 6 & 10(21) 17 & 16 & 4 \\
8(31) & 25 & 22 & \leq & 38 \\
19(37) & 18 & 17(15) 14 & 10(14) & 9(8) & 8(29) 28 & 27 & 16 & \leq & 31 \\
19(11) & 17(33) 31 & 24 & 22 & 12(29) & 8(20) 8 & 1 & 6(38) 19 & 10 & 8 & 1 \\
3(27) & 25 & 14 & 7 & \leq & 32 \\
1(11) & 9 & 5 & \leq & 0 \\
19(21) 11 & 19(10) 2 & 16(22) 16 & 9 & 12(36) 18 & 13 & 10 & 11(39) 35 & 17 & 7 \\
4(22) 17 & \leq & 40 \\
14(27) 26 & 18 & 10 & 1 & \leq & 7 \\
17(32) 23 & 18 & 13 & 8 & 17(30) 28 & 14(40) 7 & 13(36) 34 & 20 & 8 \\
10(37) 28 & 16 & 15 & 13 & 6(17) 14 & 9 & \leq & 38 \\
16(35) 19 & 6 & 1(11) 8 & 2 & \leq & 8 \\
20(39) 24 & 13 & 17(36) 10 & 6 & 1 & 13(26) 4 & 3 & 11(31) 11(40) 33 & 27 \\
3(33) 14 & 8 & 1(32) 11 & \leq & 38 \\
16(27) 13 & 12 & 11 & 2 & 15(22) 17 & 14 & 2 & 8(37) 10 & 8(39) 26 & 7(36) 28 & 5 \\
5(36) 29 & 27 & \leq & 29 \\
16(28) 27 & 5 & 2 & 15(33) 20 & 18 & 5 & 1 & 14(36) 20 & 15 & \leq & 22 \\
15(37) 26 & 10 & 12(37) 25 & 24 & 21 & 4 & 10(34) 5(5) & 4(34) 31 & 30 & 26 \\
\leq & 23 \\
14(29) 28 & 27 & 14(9) 2 & 14(19) 13 & 10 & 1 & 13(23) 9(29) 8 & 4 \\
6(36) 30 & 28 & 18 & 5 & 6(38) & \leq & 38 \\
17(23) 11(39) 31 & 26 & 21 & \leq & 14 \\
19(31) 16 & 12 & 17(24) 15 & 10 & 3 & 17(7) 16(35) 33 & 21 & 20 & 17 & 12(39) 8 \\
10(19) 17 & 3(21) 17 & 5 & \leq & 47 \\
18(29) 6 & 15(18) 12 & 11 & 7 & 1 & 15(38) 37 & 31 & 7 & 4 & 13(34) 32 & 3 \\
12(26) 22 & 11 & 10(33) 15 & 7(40) 27 & 26 & 16 & 2(29) & \leq & 46
\end{align*}
\]
| 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 0000000010 0010000000
1000000100 1000000001 0000000010 0000000000
0100100000 0000100001 0000001100 0000010000
0100100000 1110000000 0000111000 0000010010
0001000000 1000000001 010000100 0101000000
0000010000 0000000010 0000000000 0000100000
0000010000 0100000000 000110010 0000000000
0000001000 0000000000 0000000000 0001010000
0000000100 0000100000 0000000000 0001000000
0000000001 0000000000 0000010000 0000000000
0000000010 0001100000 0000000000 0000000000
0000000000 0000001000 0000000000 0000000000
0100000000 0000000000 0000100000 0000000000
0000000000 0000000010 0000000000 0000000000
0000000000 0000000000 0000000000 0000000000
0000000100 0010000000 0000000000 0000000000
0000000000 0000000000 0000000000 0000000000
0100000000 0000000000 0000000000 0000000000
1000000000 0000000000 0000000000 0000000000
1100000000 0000000000 0000000000 0000000000
0100000000 1110000000 0000110000 0000000000
0000100000 0011000010 0001101100 1010000000
1100100000 0001000010 0000100000 0010000000
1100000000 0010000010 0001101100 1010000000
1100000000 0010000010 0001101100 1010000000
PROBLEM II

Objective Function Coefficients

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

Polynomial Constraints

\[ 17(35\ 16) \ 16(39\ 22\ 17\ 13\ 7) \ 13(23\ 17) \ 9(25\ 19\ 2) \leq 27 \]
\[ 15(25\ 23\ 19\ 9) \ 13(18\ 8\ 3) \ 8(27\ 1) \ 8(6) \ 5(9) \ 5(24\ 21\ 19\ 3) \]
\[ 4(30\ 29\ 2) \ 2(28\ 23) \ 11(9) \ 2(5) \ 1(37\ 36\ 18\ 8) \leq 31 \]
\[ 19(36\ 29\ 15\ 8\ 1) \ 19(22) \ 19(32\ 31\ 25\ 13\ 7) \ 17(28) \ 12(25\ 10) \]
\[ 11(38\ 17\ 16\ 6) \ 10(37\ 35\ 9\ 2) \ 5(40\ 36\ 21) \ 4(1) \ 4(22) \]
\[ 3(40\ 16\ 6\ 5) \ 3(25) \ 3(39\ 30) \leq 64 \]

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

\[ 11(31\ 14\ 11\ 2) \ 5(37\ 31\ 19) \ 4(35\ 28\ 21\ 11\ 6) \leq 10 \]

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

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

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

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

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

\[ 20(26\ 7) \ 10(32) \ 4(40\ 34) \ 32\ 15\ 5) \leq 17 \]

\[ 16(16\ 8\ 5) \ 16(34\ 23) \ 15\ 12) \ 14(33) \ 22\ 17\ 12) \ 11(24) \ 10(29\ 15) \]
\[ 10(32\ 29\ 23) \ 7) \ 10(37\ 25) \ 15\ 12) \ 6) \ 10(39\ 38) \ 17(7) \ 9(28\ 17) \ 6) \]
\[ 7(39) \ 4(27\ 26) \ 8) \leq 58 \]

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

\[ 11(38\ 28\ 19) \ 11) \ 7(33) \ 31\ 10\ 2) \ 3(33\ 24\ 22) \ 19) \leq 10 \]
70

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 1111111111
Value of approximation: 3495

Optimal solution: 001010010 1010110110 1111011100 1111111111
Value of optimal: 3495

Final CR Problem Solved to Optimality

1010100000 0100000000 0000010010 0010110010
0100001000 0100000000 0100000000 0001110011
0001000000 0000010000 0000101000 0000000101
0001100000 0100000000 1000010001 0001110010
0011000101 0101011010 1010010000 0101000010
0000010100 0000000000 0000000000 0010001010
0010001000 0100000000 0000000000 0001110011
0010110000 0100001010 0100000000 0001100010
0000010001 0000000000 0000000000 1100000101
0000001000 0000000000 0000000000 0000000000
0000000010 0000000000 0000000000 0001001010
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 | 145 | 149 | 150 | 152 | 152 | 154 | 157 | 165 | 171 | 179 | 182 | 182 | 186 | 196 | 196 |

Polynomial constraints

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

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

\[
13(19 15) 5(36 13 8) \leq 9
\]

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

\[
19(32 12) \leq 9
\]

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

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

\[
20(39 33) 20(38 29 11 3) 19(19 7 6 5 2) 17(39) 16(37) 16(33 17) 16(36 19 11 10 8) 15(27 16) 15(30 18) 14(39 29) 14(33 16 8) 12(24 18 16) 12(27 1) 12(40 29 8) 10(28) 9(18) 9(40 31 23 7) 6(26 10) 5(39 24 2) \leq 128
\]
\[
\begin{align*}
  &\text{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) \leq 55} \\
  &\text{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) \leq 70} \\
  &\text{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) \leq 48} \\
  &\text{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) \leq 98} \\
  &\text{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) \leq 71} \\
  &\text{13(40 30 15 1) \leq 6} \\
  &\text{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) \leq 99} \\
  &\text{11(29 19 11) 10(36 11) 9(28 24 12) \leq 15} \\
  &\text{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) \leq 80} \\
  &\text{12(20 6 3) 8(39 26 25 11) 1(29) \leq 10} \\
  &\text{15(30 25 22 13 10) 5(23 11 9) \leq 10} \\
  &\text{19(19 6) 18(26 24 19 17) 16(27 23 13 6 2) \leq 26} \\
  &\text{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) \leq 58} \\
  &\text{17(32 14 12 11) 10(36 18) 8(36 11) 6(35 15 4) 4(19 12 4 3) \leq 22} \\
  &\text{17(32 11) 16(23) 11(40 22) 10(39 36 11) 8(33 19 11 10 6) 6(33 21 20 11 4) \leq 34} \\
  &\text{16(40 33 26 1) 13(32 22 15) 4(37 8 7) \leq 16} \\
  &\text{20(38 32 12 11 3) 15(39 37 3 1) 12(36 13) 9(23 17) 9(28 4)}
\end{align*}
\]
7(16 7 6 3) 7(3) 5(38 29 19 17 6) 3(40 23) \leq 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) \leq 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) \leq 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) \leq 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)
\leq 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) \leq 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) \leq 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) \leq 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) \leq 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) \leq 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) \leq 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) \leq 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) \leq 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