AN INVESTIGATION OF THE APPLICABILITY
OF INTEGER PROGRAMMING TO THE SCHEDULING
OF COMPUTER SOFTWARE DEVELOPMENT

A thesis submitted in partial satisfaction of the requirements for the degree of Master of Science in
Business Administration
by
Ernest Garth Homer

June, 1972
The thesis of Ernest Garth Homer is approved:

Dr. J. R. Miller

Professor S. J. Lef

Dr. A. Feinberg Committee Chairman

San Fernando Valley State College

June, 1972
CONTENTS

Chapter ..... Page

LIST OF TABLES ............................................. v
LIST OF ILLUSTRATIONS ............................. vi
ABSTRACT ................................................. vii
ONE. PURPOSE OF STUDY ................................. 1
   Introduction ........................................... 1
   Scope of the Study ................................... 2
   Organization of this Paper ......................... 2
TWO. THE IMPORTANCE OF COMPUTER SOFTWARE
      SCHEDULES ........................................... 3
   Introduction ........................................... 3
   Magnitude of Software Costs ....................... 3
   Strategic Impacts of Software Development Schedules .... 4
   Importance of Proper Priority Allocations .......... 5
   Computer Aided Scheduling ........................ 6
THREE. REVIEW OF THE LITERATURE ..................... 7
   Elements of this Review ............................... 7
   Integer Programming ................................ 7
   Conceptual Aids in Integer Problem Solution ........ 8
   Integer Programming Algorithms .................... 11
   Machine Sequencing Models ......................... 17
   Classes of Scheduling Problems .................... 18
   Types of Scheduling Constraints ................... 19
   Forms of Objective Functions for Scheduling ....... 22
   Multiple Dimension Decision Problems ............. 24
   Multicriterion Optimization ........................ 25
   Evaluation by Multiple Judges ...................... 26
   Multiple Objectives in Decision Problems .......... 29
FOUR. PROBLEM FORMULATION AND METHODS OF
       SOLUTION ............................................. 31
   Integer Program Formulation ....................... 31
   Objective Functions ................................ 32
   Constraints on Manpower Availability .............. 32
   Delivery Requirements ............................... 33
   Some Alternate Formulations ....................... 34
   Sensitivity Analysis ................................. 40
   Determination of Feasibility Region ............... 41
   Development of a Priority Assignment ............. 42
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIVE. PROBLEM DATA AND MECHANIZATION OF THE SOLUTION</td>
<td>44</td>
</tr>
<tr>
<td>Background of Software Scheduling Problem</td>
<td>44</td>
</tr>
<tr>
<td>Sources of Data</td>
<td>45</td>
</tr>
<tr>
<td>Manpower Availabilities</td>
<td>63</td>
</tr>
<tr>
<td>SIX. RESULTS AND CONCLUSIONS FROM INTEGER PROGRAMMING SOLUTIONS</td>
<td>67</td>
</tr>
<tr>
<td>Apparent Inconsistencies of Individual Specialists</td>
<td>71</td>
</tr>
<tr>
<td>Sensitivity to Forcing Another Component into the Solution</td>
<td>72</td>
</tr>
<tr>
<td>Conclusions</td>
<td>78</td>
</tr>
<tr>
<td>Selection of Components Using Ranking of Multiple Judgments</td>
<td>78</td>
</tr>
<tr>
<td>SEVEN. EVALUATION AND UTILIZATION</td>
<td>82</td>
</tr>
<tr>
<td>Detailed Cost Data</td>
<td>82</td>
</tr>
<tr>
<td>Conclusions</td>
<td>85</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>87</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-1</td>
<td>List of Software Functions in Work Statement</td>
<td>46</td>
</tr>
<tr>
<td>5-2</td>
<td>Software Functions-Initial Estimate</td>
<td>49</td>
</tr>
<tr>
<td>5-3</td>
<td>Priority Comparison-Two Specialists from Two Companies</td>
<td>50</td>
</tr>
<tr>
<td>5-4</td>
<td>Sample Questionnaire</td>
<td>51</td>
</tr>
<tr>
<td>5-5</td>
<td>Integer Program Selections for Period I</td>
<td>56</td>
</tr>
<tr>
<td>6-1</td>
<td>Solutions Accompanying Initial Cost and No Constraints</td>
<td>69</td>
</tr>
<tr>
<td>6-2</td>
<td>Selections of Eleven and Nine Components using Initial Costs</td>
<td>74</td>
</tr>
<tr>
<td>6-3</td>
<td>Revised Programming Estimates and Final Contract Priority</td>
<td>81</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-1</td>
<td>Flow Chart for Priority Adjustment and Iterative Solution of Subproblems.</td>
<td>37</td>
</tr>
<tr>
<td>4-2</td>
<td>Flow Chart for Integer Program Scheduling by Requirements Adjustment for Each Succeeding Period</td>
<td>39</td>
</tr>
<tr>
<td>5-1</td>
<td>Manpower Chart.</td>
<td>64</td>
</tr>
</tbody>
</table>
ABSTRACT

AN INVESTIGATION OF THE APPLICABILITY OF INTEGER PROGRAMMING TO THE SCHEDULING OF COMPUTER SOFTWARE DEVELOPMENT

by

Ernest Garth Homer

Master of Science in Business Administration

June, 1972

This study investigates the applicability of integer programming as an aid for managers in scheduling software development. The purpose of this aid is to group software components for delivery at different times to meet managerial priorities as closely as possible while respecting resource and other constraints. A minimum number of components is required for delivery at the end of each time period. Cost estimates are required for each component. Constraints on expenses and manpower for each period are also included in the model. The objective is to schedule the maximum number of the most desirable components in each period.

Integer programming was found to be useful in constructing schedules for delivery of software components. Difficulties experienced in excessive computation time were reduced by separating the model into subproblems. A method was developed which permitted
the solution of all the subproblems without resubmitting revised data
after each subproblem solution.

A priority structure for the scheduling decision model was
developed from a survey of experts. Several methods of synthe-
sizing the results of the survey were compared against the priority
structure which actually emerged in a contractual situation. A
simple majority voting system seemed to be most applicable.
CHAPTER ONE

PURPOSE OF STUDY

Introduction

This study investigates the utility of a mathematical program as an aid in scheduling computer software. For reasons to be discussed a particular type of mathematical algorithm, an integer program, was chosen for this purpose. Reasonable accurate software development schedules are needed by most administrators in the computer programming industry. This algorithm should be almost universally adaptable to any computer installation available to software development managers, since it is coded in Fortran and can be used for solving moderate sized problems too complex for hand solution. The limitations as well as the advantages of this algorithm are discussed in terms of application to an actual scheduling problem.

Scope of the Study

The actual scheduling problem involves delivery of twenty-eight software components over three periods. The scheduling of the software components must be based on a priority system, and be performed
within the limits of manpower available for system design, programming and testing. The data for this problem is selected from a real series of contractual decisions made over the last two years between the U.S. Government and a software contractor.

Since in most organizations the priority of tasks to be scheduled is established by several individuals, a consensus of judgment by different specialists with different objectives is compared to the actual contractual schedule. Alternative methods of developing such a consensus are also examined for probable affect on the schedule if these had been used. A survey of a number of specialists familiar with the requirements of the program is used to establish the different priorities used as objective functions.

Organization of this Paper

This study is organized into seven chapters. The second chapter is used to show the importance and need for optimum scheduling in the computer software industry. Chapter Three reviews the literature to establish a background in the areas of integer programming, machine scheduling and methods of synthesizing diverse judgments into a single objective. Chapter Four discusses problem formulation and methods of solution. The Fifth chapter provides insight into the data collection process necessary to formulate the decision problem. An analysis of a number of problem solutions and the sensitivity of the scheduling algorithm to variations in the parameters of the decision problem are found in Chapter Six. Chapter Seven contains an evaluation of the algorithm by several managers of software contracts after an explanation of the precepts and purpose of this method of scheduling software.
CHAPTER TWO

THE IMPORTANCE OF COMPUTER SOFTWARE SCHEDULES

Introduction

Good management of software development is important because of the magnitude of the investment in this facet of every computer operation. An obvious axiom must be that successful management of a development project must contain a reasonable goal or schedule. "Crash" schedules result in inefficiencies and costly errors in programming or system design.

Besides conservation of resources, software schedules can affect corporate strategies in the marketplace. Similar strategic impacts occur in the space or military environment. An unfilled software development schedule may profoundly affect future efforts in all three fields.

Magnitude of Software Costs

Total expenditures for any large computer project are difficult to assess. However, Boehm \(^1\) estimates that NASA's computer software

costs in the 1960's were about one billion dollars. He also estimates that military space software costs in the 1970's will be of the same magnitude.

Some of the commercial investments in the 1960's were equally staggering. T. A. Wise, in Fortune Magazine, reviewed I.B.M.'s investment in the third generation line of computers as follows:

"... Some 2,000 programmers and 'support personnel' are on the job, and the cost of this effort may run over $200 million." Wise also notes that the estimated cost for software in the early planning stage was only $10 million.

Strategic Impacts of Software Development Schedules

The general problem of scheduling a new product is especially applicable to the software industry. Late deliveries of new computers is often blamed on incorrect software development schedules. For example, T. A. Wise notes I.B.M.'s difficulties in the 1960's--"By I.B.M.'s own estimates, the company won't have most of the bugs out of programming the larger systems until the middle of 1967--at least a year behind its expectations." The largest computer manufacturer in the world was unable to estimate its software delivery schedule correctly!

---

224: p. 139.
These late deliveries undoubtedly affected military and space installations as much as commercial users of the computers, since many of the larger systems were sold to governmental users. On a smaller scale, companies that promise their customer's software features with the delivery of a new computer, and then fail to supply the necessary programs with the hardware, often find themselves faced with contract cancellations.

**Importance of Proper Priority Allocations**

Since software is costly and difficult to schedule accurately as discussed above, an important management responsibility lies in assigning the priorities for the computer features during the planning phase of development. Many computer features require software for implementation into the computer complex. In order for these features to be systematically scheduled, a priority structure is necessary.

Many times the decisionmaker, in establishing priorities for software features, finds it necessary to synthesize diverse judgments on the relative importance of the different functions which the computer complex can perform. These judgments are furnished by different specialists. Some are familiar with the technical possibilities offered by the computer complex, others are more in contact with potential users and are able to reflect this element of the decision process.

Similar combinations of opinions on priorities must be reached in scheduling features for computer operations in the space and military environments. This study is directed toward scheduling computer
software for eventual military use. However the programming and testing is performed by commercial software corporations.

**Computer Aided Scheduling**

Since schedules for software development are difficult to formulate, a computer can be used to aid management in this decision process. A particular scheduling algorithm is investigated in this study. The algorithm has been used in numerous other applications, but this investigation concentrates on its applicability to computer software scheduling.
CHAPTER THREE

REVIEW OF THE LITERATURE

Elements of this Review

An investigation into the applicability of integer programming to scheduling must include a review of the definitions and algorithms which make integer programming a viable decision method. This review is found in the first part of this chapter. The second section of this chapter considers the general scheduling problem and the types of equations necessary to model the problem. The final section is devoted to the difficulties of multiple criteria in a decision problem. These difficulties are extended when multiple judges are used to evaluate the criteria. Methods for overcoming these difficulties are discussed.

Integer Programming

Integer Programming is a class of algorithms that are useful in solving mathematical management problems requiring integer solutions. These algorithms reach a solution or indicate the problem as stated has no solutions.
The particular type of integer programming of interest in this thesis requires solution values of either zero or one for the variables. These discrete values are especially desirable for scheduling software components which are considered indivisible into smaller units, and which are delivered only once. Other forms of integer programming algorithms allow the variables in the solution to be expressed as zeros or multiples of one.

Linear integer programming requires expression of the problem in terms of linear inequalities. The objective function is also linear. However, because of the requirement for an integer solution, linear programming methods can not be used to find optimal solutions.

**Conceptual Aids in Integer Problem Solution**

Geoffrion and Marsten discuss three concepts which can be used individually or collectively in developing Integer Program algorithms. These concepts were identified as separation, relaxation and fathoming.

Separation is the concept of dividing a problem into smaller sub-problems. Thus, in the three period scheduling problem discussed in this thesis, one method for solution might be to divide the problem into three one period problems. The results of the solution for the first period could be used to eliminate candidates for scheduling in the second period. Candidate components already scheduled in the first period...
and second periods would not be considered for the third period. In
the more general case, solutions to subproblems can be compared to
finding the optimum solution for the main problem.

Relaxation is the method of loosening the constraints of the
problem, in effect, yielding a new problem. The restrictions usually
loosened are the integer requirements on the variables so that a
linear program can be used on the relaxed problem.

Assume a problem:

\[
3-1 \quad Z_{\text{min}} = \sum_{j=1}^{M} P_j X_j
\]

Subject to:

\[
B - \sum_{j=1}^{M} A_j X_j = 0.
\]

\[
D - \sum_{j=1}^{M} A_j X_j = 0.
\]

A relaxation of the above problem might be more easily solvable.
Dropping requirements on \( D \), the problem is only constrained by:

\[
3-2 \quad B - \sum_{j=1}^{M} A_j X_j = 0.
\]

If the optimal solution when constrained by only 3-2 is feasible
in 3-1, then it is the optimal solution to 3-1.
Another tactic used is to relax the requirements well within the feasible region, and then raise these requirements gradually until the problem is no longer feasible. This tactic is sometimes regarded as the opposite of a restrictive\(^2\) technique.

Fathoming, unlike the previous two concepts, which were of the "divide and conquer" strategy, is a set of definitions which provides "stop" signals for analysis of the different branches of a problem.\(^3\) These signals provide the clue for discontinuing a search for a solution in a particular branch of the problem. The reason may be that the optimal solution for that branch has been found, or that no feasible solution is available from that branch or that no solution from that branch is better than the current solution.

For instance, in the simple example of a two inequality problem, with three 0-1 variables, \(x_1\), \(x_2\), and \(x_3\):

\[
Z_{\min} = P_1x_1 + P_2x_2 + P_3x_3
\]

Subject to:

\[
A_1x_1 + A_2x_2 + A_3x_3 \leq B_1,
\]


and \[ A_1x_1 + A_2x_2 + A_3x_3 \leq B_2. \]

Let us assume that it is discovered that \( A_1 + A_2 \) is greater than \( B_1 \).
Then the particular branch with \( x_1 = 1, x_2 = 1 \) has been fathomed, since it is pointless to sum all three parts of this constraint. Another branch to be tried would be \( x_2 = 1, x_3 = 1 \).

Integer Programming Algorithms

A number of deterministic algorithms have been introduced which utilize one or more of the above concepts in solving integer problems. Three types of algorithms will be discussed here: (1) Cutting Planes, (2) Enumerative and (3) Group Theoretic.

The cutting plane algorithm was the first one to be introduced for integer programming. The foundation for cutting plane algorithms was laid by Gomory, although Dantzig first showed the feasibility of the approach in a traveling salesman problem. The problem is first solved by relaxing the integer constraints and then successively adding new constraints which tend to force the problem solution into an integer state. The new constraints should not make the solution feasible, even though each new constraint successively removes a portion of the feasible region for the non-integer solution.

---


Geoffrion and Marsten point out that only problems which seem to have natural integrality tendencies and are of moderate size are successfully solved with this algorithm. Gomory proved this technique must eventually converge into an all integer answer.

Basically, the mathematics of this algorithm can be expressed in the following manner. If the solution to the continuous linear problem

$$
\text{Minimum } Z = \sum_{k=1}^{K} \sum_{j=1}^{J} p_{jk} x_{jk}
$$

Subject to

$$\sum_{j=1}^{J} a_{ijk} z_{jk} = c_{ik} \text{ for all } i = 1 \text{ to } M \text{ and } k = 1 \text{ to } K$$

is

$$
Z_{\text{minimum}}^{*} = \sum_{k=1}^{K} \sum_{j=1}^{J} p_{jk} x_{jk}^{*} + p_{jk} y_{jk}^{*}
$$

\[
X_{jk}^{*} = \text{integer solution values},
\]

\[
Y_{jk}^{*} = \text{noninteger solution values}.
\]

Then, the fractional part of particular \( Y_{jk}^{*} \) is equated to the fractional parts of the coefficients of the nonbasic variables \( F_{ijk} \). Thus the

\[12: \text{ p.42}.
\]

\[13: \text{ p.287}.\]
new constraint consists of a new variable $X_{m+1}$, and the above coefficients in the following form:

$$3-6 \quad X_{m+1} - \sum_{j=1}^{J} F_{ijk} X_{jk} = D_{ik}^*$$

where

$$3-7 \quad Y_{jk}^* = 1 + D_{ik}^*$$

$3-7$ is appended to $3-4$.

Then the problem must be solved again. Although the dual simplex algorithm can be used to avoid resolving the entire problem, many solutions of $3-4$ may be required. The algorithm terminates when a solution is found with all integer variables.

**Enumerative Integer Programming**

The elementary procedure for the enumerative solution of an integer program is to follow a tree search according to logical rules. Values are assigned to the different elements in a given branch until either a new solution is found which is better than the best previous solutions, or enough information is provided at some particular node to rule out any further computation on that branch. Then the algorithm backtracks to the branching point and starts the computations for the next branch. When all branches have been tried or evaluated for better solutions, the search is terminated. Balas\(^8\) first organized

the tree search in a logical manner and developed some of the
evaluation techniques to reduce the total computation time.

Glover\textsuperscript{9} developed the concept of the surrogate constraint which
places restrictions on the problem solution. This surrogate con-
straint is derived from a linear combination of constraints which is
stronger than any in the problem. The surrogate constraint is con-
structed by taking a non-negative linear combination of the original
constraints using the values of the dual variables as weights. The
dual variables are found by solving the integer problem with a simplex
algorithm and assuming the variables are continuous.

Geoffrion\textsuperscript{10} showed how the surrogate constraint can be enhanced
by adding to it the constraint that the objective function must have a
better value than that of best previously calculated.

Defining the primal problem as

\[
3-8 \quad Z_{\text{minimum}} = \sum_{j}^{N} C_j x_j \quad \text{subject to:}
\]

\[
\sum_{j}^{N} A_{ij} x_j \geq B_i \quad i=1 \text{ to } M; \quad 0 \leq x_j \leq 1 \text{ and integer.}
\]

\textsuperscript{9}14: Fred Glover, "A Multiphase Dual Algorithm for the Zero-
one Integer Programming Problem", \textit{Operations Research}, Vol. 13,

\textsuperscript{10}10: A. M. Geoffrion, "An Improved Implicit Enumeration
Approach for Integer Programming", \textit{Operations Research}, Vol. 17,
Then the dual of this problem is

$$3-9 \quad U_{\text{max}} = \sum_{i=1}^{M} W_i B_i$$

Subject to:

$$3-10 \quad \sum_{i=1}^{M} W_i A_{ij} \leq C_j \hspace{0.5cm} W_i \geq 0,$$

where

- $W$ is a $1$ by $M$ row vector,
- $B$ is $M$ by $1$ column vector,
- $C$ is $N$ by $1$ row vector, and
- $A$ is $M$ by $N$ matrix.

Then the surrogate constraint is

$$3-10 \quad U(B + AX) + (Z - CX) > 0$$

Geoffrion built the computer code RIP 30C\textsuperscript{11} to determine how much improvement, if any, such strongest surrogate constraints would make in the computation time for Balas algorithm.\textsuperscript{12} The Balas algorithm had appeared slightly earlier. A linear programming subroutine was imbedded so that it could be used to derive the surrogate constraints. The result is a code which alleviates some of the

\textsuperscript{11}10: p. 439.

solution time dependence on the number of variables, reducing it from an exponential to a low order (possibly the fourth) polynomial increase. 13 (The RIP 30C program was used to solve the scheduling problem discussed in subsequent chapters.)

Group Theoretic Integer Programming

Gorry, Shapiro, and Wolsey 14 discuss the necessity for applying relaxation methods not only for reasons of efficiency but in order to implement the group theoretic methods they have studied. The principal relaxation method discussed in that paper is to transform an all integer problem by dividing all the coefficients of an inequality by a rational divisor greater than one, and then separating the integer and noninteger parts. The point is made that all feasible solutions to the original all integer problem are feasible in the transformed or relaxed problem. The proof of this point follows that of the Gomory cut in all integer methods.

Group theoretic algorithms are still in an experimental stage. A heuristic supervisory procedure is being developed to guide the search for an optimal solution. 15 This search must be directed


toward organizing the subproblems and selecting the analytical methods most efficient in their solution. Gorry and Shapiro\textsuperscript{15} show the relationship of the group theoretic algorithms to the overall integer program framework.

Generally, a linear program is applied to the problem first, and then the corrections are applied after the problem has been separated into subproblems. These subproblems are relaxed to a group problem by dropping the non-negativity condition. If an arbitrary correction can not be fathomed, then this correction is replaced by subsequent corrections. These subsequent corrections have the effect of separating the subproblem into problems of lower order.

\textbf{Machine Sequencing Models}

There is a marked similarity between sequencing products through a series of machines (a job shop problem), and the specific problem of scheduling software components among different specialists as discussed in this paper. Wagner\textsuperscript{16} was one of the first to propose a general model for integer programming solution, although no computer code was available for implementation of his model at that time. Other approaches of interest which have been


tested are those of combinatorial analysis of Smith and Dudek\textsuperscript{17}, and the Branch and Bound techniques of Charlton and Death\textsuperscript{18}. An appealing heuristic algorithm has been proposed by Campbell\textsuperscript{19} for job shop scheduling with flexible technological constraints. Papers by Holloway and Nelson\textsuperscript{20} and Charlton and Death\textsuperscript{18} will be discussed here since they encompass most of the considerations required in the general scheduling problem.

Classes of Scheduling Problems

Charlton and Death\textsuperscript{21} define seven specific classes of general machine scheduling problems:

1. J-Job, I-machine scheduling without passing. All J jobs must be processed on each machine. (No availability or due date constraints.) Since no passing is allowed, the sequence of jobs on each machine is identical.


\textsuperscript{21}pp. 698-703.
2. J-job, I-machine problem with passing. Same as 1 except the order need not be maintained.

3. General Machine-Job Problems. The machine which performs each operation is specified, but each job need not be processed on all machines or any common subset of them.

4. Machine Job Problems without Preassignment. The only requirement is to know which machines perform what function. Preassignment is not necessary.

5. Machine Routing Problems. Machines have no capacity restriction. The objective is to minimize total changeover time or the number of machines used.

6. Assembly Line Balancing without Backpassing. No Backpassing is defined as the restriction that if a job has been processed on machine i + 1 (further down the assembly line) it will never be processed on machine i which is closer to the start of the assembly line.

7. Assembly Line Balancing with Backpassing. This model allows provision for rework.

Types of Scheduling Constraints

Charlton and Death\(^22\) list several types of constraints which are characteristic of the general scheduling problem. Because human organizations are more flexible than machines, these constraints are

\(^{22}\text{Charlton and Death, 1990.}\)
not always binding on the software manager in scheduling computer programs.

The following conventions will be observed in this discussion:

\[ i = \text{machine or skill group} \]

\[ p\_d = \text{due date for operation } p \]

\[ p\_t = \text{start time of operation } p \]

\[ p = \text{operation} \]

\[ c\_p, p+1, i, i+1 = \text{changeover time when operation } p+1 \text{ on machine } i+1 \text{ follows operation } p \text{ on machine } i.\]

\[ a\_pi = \text{process time or work coefficient for operation } p \text{ on machine } i.\]

**Type 1.** - Operation \( p \) may be placed on not more than one group \( (i) \) and the processing must be continuous.

**Type 2.** - Any group \( (i) \) may perform no more than one operation \( p \) at a time.

**Type 3.** - A technological ordering constraint may exist for any pair of operations \( p \) and \( p+1 \). Thus if operation \( p \) on machine \( i \) must precede operation \( p+1 \) on machine \( i+1 \), then

\[ t\_p + 3-11 \leq t\_p + 1 + a\_pi + c\_p, p+1, i, i+1 \]

where \( t\_p+1 \) is the start time of operation \( p+1 \) and \( t\_p \) is the start time of operation \( p \). \( a\_pi \) as defined above is the process time for operation \( p \) on machine \( i \), and \( c\_p, p+1, i, i+1 \) is the changeover time when machines and
operations are in the sequence indicated. If operations are not assigned to specific machines, then the minimum of the righthand side of the equations must be found.

Type 4. - Some operations \(p, \ldots, p+m\) can not be assigned to machine \(i\) if operations \(p+n, \ldots, p+s\) are assigned to that machine.

Type 5. - Each machine \(i\) is available for a limited time \(t_{p+\phi} - t_p\).
Where \(t_p\) is the start of the first operation, and \(t_{p+\phi}\) is the end of the last operation on machine \(i\). This constraint becomes

\[
3-12 t_{p+\phi} - t_{np} \geq \sum_{n=1}^{N} c_{p, p+n, i, i-1} + a_{mi}.
\]

Type 6. - Certain operations \(p\) must be completed at or before time \(d_p\), the due date. This gives rise to a constraint of the form

\[
3-13 t_p + a_{pi} \leq d_p
\]
for operation \(p\) performed by group \(i\).

Type 7. - Certain pairs of operations (\(g\) and \(h\)) must be performed concurrently. This gives rise to constraints of the form:

\[
3-14 t_g - t_h \geq 0
\]
\[
3-15 t_g - t_h = a_{gi} - a_{h, i+1}
\]
for operation \(g\) performed by group \(i\), and operation \(h\) performed by group \(i+1\).
Software development is similar to the job shop in a number of ways. Each machine can be replaced by a skill group. Designing or testing a software function by a skill group is equivalent to performing machine operations on a product. Not all job shop constraints discussed above are applicable to the software industry, however. In general, constraint types 1 and 2 are not realistic to the software scheduling situation. Constraint type 3 usually holds, i.e., system engineering is required before programming, although the line of demarcation is seldom clear. Constraint type 4 sometimes holds for a period of time, but charters of job functions of groups may change. Constraint types 5, 6 and 7 are realistic situations in the computer software industry.

Forms of Objective Functions for Scheduling

Holloway and Nelson\textsuperscript{23} discuss three forms of objective functions for scheduling. Their article points out that each form is the dual of the classical formulation, but each has certain advantages for different algorithms. The classical objective is to minimize total tardiness or schedule slippage. This objective corresponds to the classical job shop scheduling problem in which early delivery of one job cannot offset late delivery of another job. The objective is to minimize total delay in delivery. Using the notation used for illustrating scheduling constraints, the objective becomes:

\[ \text{Slip}_{\text{Minimum}} = \sum_{i=1}^{I} \sum_{p=1}^{P} [(t_{pi} + a_{pi} + c_{p,p+1,i,i+1} - d_{pi})] \]

\textsuperscript{23}18: p. 4-6.
where I is the last machine on the line, and P is the final operation, lower cases subject to:

1. Satisfying job precedence constraints of type 3.
2. Holding machine capacity fixed similar to constraints types 2 and 5.

The second objective form is concerned with the minimization of additional machine capacity, an expression such as 3-17 would be used

\[
3-17 \quad \text{Capacity}_{\text{Minimum}} = \sum_{i=I+1}^{J} \]

and for additional machines, i=I+1, ..., J. The optimum solution for minimal capacity must be subject to the following types of constraints:

1. Satisfying job precedence constraints of type 3.
2. Satisfying job due dates similar to constraint type 6.

In the third objective function, it is recognized that some operations start before the previous operations are completed. The objective is to minimize these job precedence violations.

\[
3-18 \quad \text{Violation}_{\text{minimum}} = \sum_{i=1}^{I} \sum_{p=1}^{P} \left[ (t_{pi} + a_{pi} + c_{p,p+1,i,i+1}) - t_{p+1} \right],
\]

Subject to the following constraint types:

1. Fixed machine capacity similar to type 2 and 5
2. Job due dates are satisfied similar to type 6.
Multiple Dimension Decision Problems

Generally, decision problems have many dimensions. Realistic problems possess multiple criteria or attributes which can be assigned in varying degrees to each alternative solution. For instance, in the selection of any large scale system, such as a computer complex, a management information system, an airline reservation network, a process control installation or a satellite command and control system, each alternative configuration or system possesses multiple criteria or attributes. The decision maker must somehow weigh these different attributes to decide on the system "nearest his ideal".

In large organizations, the decision process is seldom limited to one person. Consultants outside the organization as well as special skills groups inside the organization perform studies to aid the decisionmakers. The decisionmakers are often a committee or a management team who must somehow digest the conflicting reports from various sources and synthesize these into a single decision. MacCrimmon24 points out the obvious fact that as the size of the problem increases, that is, the number of alternatives and relevant attributes increases, the point is reached where a decisionmaker's processing capacity, and perhaps even the processing ability of his organization, is exceeded.

Three factors which could make a problem multidimensional will be reviewed here: Multicriterion, Multijudges, and Multi-objectives.

Multicriterion Optimization

MacCrimmon attempts to show the relationship of older decision-making methods to multicriterion problems. The principal technique he offered was to attempt to reduce the number of criteria to a size that could be easily comprehended. MacCrimmon concluded only the weakest of these methods (such as dominance and satisficing theory) was suitable for a full dimensionality. Of more interest here, are methods applicable to mathematical programming such as tradeoff curves. Tradeoff curves, MacCrimmon felt, could be used for an intermediate number of criteria, but the complexity and number of terms which had to be calculated made this method unsatisfactory for decisions involving many criteria.

Geoffrion, Dyer, and Feinberg demonstrated the use of trade-offs with multiple criteria. Their approach is interactive between the decisionmaker and a computer. The approach combines utility theory and mathematical programming to perform optimization with multiple criteria when the decisionmaker's utility function is not explicitly known.

25 19: p. 44.

The decisionmaker chooses a feasible starting point. At this point, he estimates tradeoffs between each criterion and a reference criterion. These tradeoffs are used to compute a number of alternate criterion values. The decisionmaker then chooses his most preferred criterion values from those computed. At the new criterion values, the tradeoffs are reestimated and alternative values are computed. This process is continued until the most preferred criterion point for the decisionmaker is reached. 27

Evaluation by Multiple Judges

One procedure for eliciting and refining the judgments of a group of experts is called the Delphi Method. 28 This method consists of collecting these judgments from individuals privately so as to reduce the effect of a socially dominate individual in a committee activity. Individuality is further enhanced by maintaining the anonymity of the participants. After the judgments are collected a statistically derived "group response" is fed back to the individuals. Individuals whose opinions deviate from the median of the group are asked to justify their judgments. Several iterations may be used to discover the best group opinion, since justification for judgments can be reviewed by each member of the group.

Farquhar 28 attempted to improve the estimation process for software costs by use of the Delphi Method. He found that, although

27 11: p. 18, 19.

the estimates of the participants converged with the number of iterations of the estimation process, the convergence was away from the correct answer rather than toward it. He found that the programming and checkout portions of the problem exhibited the largest variances in the initial estimates and converged the most rapidly. Because of the convergence, the Delphi Method seems to be effective in obtaining a consensus of opinion.

It is interesting that Farquhar was using the cost of an actual computer programming effort as the correct answer in evaluating the estimates of the participants. He notes that, in reviewing the history of that computer program development, a particular proposition seemed to be evident. This proposition states that "all available resources will be used." In that particular case, this meant multiply the number of available personnel times the usual life of a development project (two years) yielding the manpower expenditure. Since this seemed to be the basis for his "correct" answer, his conclusions on the effectiveness of the Delphi Method are questionable, inasmuch as actual cost was a function of the number of people available and not the number of people required to complete the task.

Eckenrode compares methods for collecting data on human judgments on the relative value of a series of items. When a system is being developed, and certain criteria are being used in design trade-offs, the relative importance of different criteria must be evaluated.


Expert judgment must be relied upon to make the best selection of design features. Eckenrode's research compares the reliability of six methods for collecting the judgments of experts.

Two of the judgmental situations were similar to the system design discussed in this thesis, in which a group of experts who were thoroughly familiar with the system criteria were asked to evaluate these numerically.

Eckenrode's research was made on the following six methods for collecting judgmental data:

1. Ranking—a technique which requires placing a value on each criterion in a continuous spectrum.
2. Rating—a method which allows the judges to place greater or less importance on criteria by the use of discontinuities in the evaluation spectrum. The numerical evaluation now takes on the dimension of relative weights, as well as tradeoffs with its nearest competitor.
3, 4, 5. Partial and Complete comparison—Pairs of criterion are traded against each other for preference.
6. Successive comparison—a method that is similar to ranking except that the value of the most important criterion was adjusted to be equal to the sum of all the other criteria.

Eckenrode's research was made on the following six methods for collecting judgmental data:

1. Ranking—a technique which requires placing a value on each criterion in a continuous spectrum.
2. Rating—a method which allows the judges to place greater or less importance on criteria by the use of discontinuities in the evaluation spectrum. The numerical evaluation now takes on the dimension of relative weights, as well as tradeoffs with its nearest competitor.
3, 4, 5. Partial and Complete comparison—Pairs of criterion are traded against each other for preference.
6. Successive comparison—a method that is similar to ranking except that the value of the most important criterion was adjusted to be equal to the sum of all the other criteria.

Eckenrode's research was made on the following six methods for collecting judgmental data:

1. Ranking—a technique which requires placing a value on each criterion in a continuous spectrum.
2. Rating—a method which allows the judges to place greater or less importance on criteria by the use of discontinuities in the evaluation spectrum. The numerical evaluation now takes on the dimension of relative weights, as well as tradeoffs with its nearest competitor.
3, 4, 5. Partial and Complete comparison—Pairs of criterion are traded against each other for preference.
6. Successive comparison—a method that is similar to ranking except that the value of the most important criterion was adjusted to be equal to the sum of all the other criteria.
Eckenrode\textsuperscript{32} found each method produced essentially the same ordering of criteria, and that the judges were consistent in their ordering regardless of the method used. However, Eckenrode's tests were limited to a relatively small number of criteria (6) for a very complex system decision situation. This thesis considers the problem of twenty-eight criteria to be chosen for a system.

The ranking method was determined to be far more efficient in the expenditure of time by Eckenrode\textsuperscript{33}. The rating method seemed to be the next most efficient, although some partial comparisons took less time than rating in one of his cases. Successive and complete comparisons took twice as long as the rating method, and at least four times as long as the ranking method.

Multiple Objectives in Decision Problems

The problem of formulating multiple objectives in the application of mathematical programming is discussed by Klahr\textsuperscript{34}. In Klahr's paper, it is pointed out that mathematical programming forces oversimplification in the use of an objective function, since multiple and seemingly conflicting goals are often encountered. Klahr suggests\textsuperscript{35} that a problem with conflicting multiple objectives may be separated into regions of subobjective space. This separation may be difficult.

\textsuperscript{32} Eckenrode, pp. 181-182.
\textsuperscript{33} Eckenrode, p. 187.
\textsuperscript{34} Klahr, "Multiple Objectives in Mathematical Programming", \textit{Operations Research}, 1958, Vol. 6, No. 6, p. 849.
\textsuperscript{35} Klahr, p. 851.
A different objective is then defined for each region. Whenever the different objectives are not comparable, an order of preference between regions must be established.

In the problem of scheduling computer software development discussed in this study, many conflicting objectives were encountered by the panel of experts who had to establish a priority relationship between criteria. This conflict was heightened because of differences in the goals of the organizations which they represented. In order to establish a schedule for the software, a decision had to be made on which features to develop in the first period. The relationship of this decision to other methods of synthesizing multiple objectives will be discussed in Chapters Five and Six of this paper.
CHAPTER FOUR

PROBLEM FORMULATION AND METHODS OF SOLUTION

The principal means of solving the software scheduling problem in this study will be by the use of an integer programming algorithm implemented in Fortran called RIP 30C\textsuperscript{1}. However, this algorithm alone is not sufficient to always provide meaningful schedules. Therefore, two alternate auxiliary algorithms are suggested. In addition, some views on sensitivity analysis are presented, as well as a discussion on the development of a priority assignment.

Integer Program Formulation

One advantage in using a discrete integer program for solving a scheduling problem stems from the fact that the optimal solution is expressed in ones or zeros. Thus, a decision-maker can easily visualize the resulting answer in terms of a software component scheduled for delivery or not. For software components and other products which must be delivered in units, this is an important consideration in building a model. Contractually, each software component is considered to be successfully delivered when testing proves the function performs as specified. Fractional delivery of a component under these circumstances is virtually meaningless, but fractional answers

\textsuperscript{1}10:p. 7.
are commonplace in the usual linear programming algorithms. Integer programming avoids this situation.

Objective Functions

The objective function should express management's desire to ship first the software components that are most important to the customer. Assuming that each software component can be assigned a priority, either by the customer or jointly with the software company's management, then an objective function can be written in terms of these priorities and the software components. Let \( Z \) represent the schedule in terms of the most desirable components. Taking the lowest priorities to be the most desirable and should be delivered first, and the optimum \( Z^* \), should be the minimum sum of component priorities possible within the constraints of the decision problem.

\[
4-1 \quad Z^* = Z_{\text{minimum}} = \sum_{k=1}^{K} \sum_{j=1}^{J} P_{jk} X_{jk}
\]

where

\[
X_{jk} = \begin{cases} 
1 & \text{if component } j \text{ is finished in period } k, \\
0 & \text{if otherwise}
\end{cases}
\]

\[
P_{jk} = \text{Priority associated with finished component } j \text{ in period } k.
\]

Constraints on Manpower Availability

In the design, development, and production of computer software, at least three distinct groups of specialists are required. These groups are generally called systems engineering, programming, and testing. A general formula which expresses the limited
availability of manpower must contain a value which represents the
total number of men available in each group i in period k, or \( C_{ik} \). In
addition, the coefficient of manpower from each group associated with
each component must be estimated. This manpower coefficient,
\( A_{ijk} \), where i represents the group, k is the period, and j is the soft­
ware component, need not be the same for each period. Thus, in a
more general case, \( A_{ijk} \) could reflect improved efficiency in suc­
cessive periods based on some learning curve.

Based on these considerations, the formula for constraints can
be expressed as:

\[
4-2 \quad C_{ik} - \sum_{j=1}^{J} A_{ijk} X_{jk} > 0 \quad \text{for groups, } i = 1 \text{ to } M
\]
\[
\quad \text{periods, } k = 1 \text{ to } K
\]
\[
\quad \text{and components } j.
\]

**Delivery Requirements**

Since the customer does not know exactly which components will
be delivered in each period because of the manpower availability con­
straints above, the contract will surely call for some minimum num­
ber of components to be delivered in the early periods. The customer
needs this requirement as an aid in measuring progress toward the
completion of the project, and so at least some very desirable com­
ponents will be delivered in the first and succeeding periods. Let
this minimum number of components in period k be called \( D_k \). Then

\[
4-3 \quad -D_k + \sum_{j=1}^{J} X_{jk} \geq 0 \quad \text{for periods, } k = 1, \ldots, K \text{ and } j \text{ components.}
\]
To force the integer program to schedule a component only once, a number of equations are needed which relate the same components in each period, \( k \). These are simply:

\[
4-4 \sum_{k=1}^{K} X_{jk} = 1, \quad j = 1, \ldots, J.
\]

Some Alternate Formulations

The use of the RIP 30C code pointed up some difficulties common to most integer programming algorithms. The computing time became excessive when attempting to solve the problem consisting of 28 components such as those found in 4-2, which represents 84 variables for three periods. More than 40 inequalities would be required for three periods and three professional groups. When only 26 expressions were introduced to the computer, a CDC 6600, the time exceeded 360 seconds.

One alternative to this dilemma was to separate the problem into three subproblems, one for each period, and solve these iteratively one at a time. For instance, the components selected for delivery in the first period can be removed from the subproblem for period two by raising the priority of the period one components over the components remaining (assuming the lowest priority components are most desirable). The problem is submitted to the computer again, and the process is repeated when that solution is obtained. The savings in time is considerable, since only 28 variables need be present in each subproblem, in this case. The concept of breaking up a problem into subproblems is discussed in Chapter Three.
Another alternative is to remove those variables from the requirements vector which have already been scheduled in the previous period. Since the algorithm selects the minimum sum of these variables times their priorities, these would not be scheduled again.

**Alternative One--Raising Selected Component Priorities**

The solution to the subproblem for period one is

$$
4-5 \quad Z^* = \sum_{j=1}^{R} P_j X_j^* + \sum_{j=R+1}^{J} P_j Y_j^* \quad \text{where } X_j^* = 1 \quad \text{and } Y_j = 0
$$

where the components selected for delivery in period one are now renumbered from 1 to R, and those remaining are numbered R + 1 to J. The priority of the components for the components remaining can be denoted as the first to be considered by the algorithm. This is accomplished by raising the priority numbers of those components already selected for delivery in period one. An easy way to do this is to create a new vector, $E_j$, and add it to $P_j$,

$$
4-6 \quad E_j = G X_j^* \quad \text{for } j = 1 \text{ to } R,
$$

The objective function for period two now becomes

$$
4-7 \quad Z_{\text{minimum}} = \sum_{j=1}^{R} (P_j + E_j) X_j + \sum_{j=R+1}^{J} P_j Y_j.
$$

To be effective in 4-7, then all $(P_j + E_j)$ should be larger than any $P_j$ for components $R + 1$ to J. Otherwise, the same component may be selected in both periods. In fact, although the algorithmic solution
for period two may indicate feasibility, if the period two resources are inadequate, one or more of the components subscripted $j = 1$ through $R$ may be selected again.

Flow Chart for Priority Adjustment

The flow chart, Figure 4-1, demonstrates the interactive adjustments of priorities after each solution for a given period. After setting the initial values of Priorities, $P_j$, Manpower, $C_1$, and Requirements, $D_1$, the integer program is solved using RIP 30C, Block 2. The solution for the first period is printed out, including the components $X_j$ which are scheduled for this period, Block 3. The vector $E_j$ is constructed from $X_j^*$ (equation 4-6) and added to $P_j$, for $j = 1$ to $R$ (equation 4.8) as in Block 4.

The changes in the Constraints, $\Delta C_1$, and any adjustments in the Requirements, $D_1$, are read in the computer per Block 6. These are added to the Constraints and Requirements, Block 5, to form the new integer subproblem for the next period, $k + 1$. Then the algorithm is applied to this subproblem, obtaining the solution for period $k + 1$.

Alternative Two--Removing Components from the Requirements Expression

The second alternative method of restructuring the integer formulation to reduce computer time is similar to the first in that the problem is again divided into subproblems, one for each period. This time, after a solution has been obtained for the first period, the requirement vector is altered so that those components already
FIGURE 4-1
FLOW CHART FOR PRIORITY ADJUSTMENT AND
ITERATIVE SOLUTION OF SCHEDULING SUBPROBLEMS

1. Set Initial Values

2. Solve Integers Program

3. Print $X^*_j = 1$ for

$$Z_{min}^* = \sum_{j=1}^{R} P_j X^*_j$$

4. Set $P_{jk+1} = E_j X^*_j + P_j$
   For all $j = 1$ to $R$

5. Set Constraints + Requirements for period $k + 1$
   $C_{ik+1} = C_{ik} + \Delta C_i$
   $D_{ik+1} = P_{ik} + \Delta D_i$

6. Changes in
   Constraints = $\Delta C_i$
   Requirements = $\Delta D_i$

Solution for Period K
Selection ($X^*_j = 1$) of
Components, Period 1
Components, Period 2
Components, Period 3
scheduled are excluded from succeeding subproblems. This method combines the separation technique with a form of relaxation discussed in Chapter Three.

If the solution to the subproblem for period one is:

$$Z_1^* = \sum_{j=1}^{R} P_j S_j^* + \sum_{j=R+1}^{J} P_j Y_j^*$$

where

- $S_j^* = 1$ scheduled
- $Y_j^* = 0$ not yet scheduled

A new vector $\phi_j = S_j^* + Y_j^*$ will represent those components scheduled and not scheduled in period one. Thus, if this vector is subtracted from the requirements vector, $X_j$, a new expression for requirements will exclude components previously selected. 4.3 will become:

$$-D_2 + \sum_{j=1}^{J} (X_j - \phi_j) = 0$$

where $\phi$ is derived from 4.8

The flow chart for requirements adjustment, Fig. 4-2 shows a feasible approach for automatically adjusting requirements after each period solution. In Block 4, the vector $\phi_j$ is obtained from the values of the solution variables for Period One and subtracted from the variables in the requirements expression. The new value for the number of components to be delivered in the following period ($k+1$) is also inserted here. Any changes in constraints for the following period ($k+1$) are added in Block 5.
FIGURE 4-2
FLOW CHART FOR SCHEDULING BY
REQUIREMENTS ADJUSTMENT FOR
EACH SUCCEEDING PERIOD

1. Set Initial Values
   \( C_i, D_k, P_j \)

2. Solve
   Integer Program

3. Print \( S^*_j = 1 \) for
   \[
   Z^*_{min} = \sum_{j=1}^{R} P_j S^*_j + \sum_{j=R+1}^{J} P_j Y_j
   \]

4. Form \( \phi_j = S^*_j + Y_j \)
   \[-D_{K+1} + \sum_{j=1}^{J} (X_j - \phi_j) \geq 0\]

5. Read \( \Delta C_i \),
   For period \( K+1 \)

6. \( D \) for
   Period \( K+1 \)

7. \( \Delta C_i \)
   Change in
   Constraints
   For Period
   \( K+1 \)
Sensitivity Analysis

The relaxation method discussed in Chapter Three may be applied to this scheduling problem either by loosening the constraints or the requirements. Since the requirements are set above some minimum by contract, however, the important consideration for management is the size of the resources required to make the scheduling problem feasible. Probably the first approach by the software manager is to see if his present staffing is adequate. This is seldom the case when he receives a new contract so he finds that one of the constraints in 4.2 is violated. Call this constraining value $C_y$, for example, then for any combination of variables which might meet the requirements:

$$\text{4-10} \quad C_y - \sum_{j=1}^{J} A_{yj} X_j < 0$$

Presumably the manager will then scan the problem, raise the value of $C_y$ and perhaps other constraining values until the problem is feasible for at least the first period. However, since constraining values must be adjusted in other periods also, this can be tedious and time consuming. And he might overcompensate in order to get a feasible solution by having more resources allocated than are needed, or add resources to a nonconstraining expression.

A better procedure is to discover the number of components which can be delivered with the resources presently allocated. A computer can do this quite easily, for small problems, by incrementing $D_k$ in equation 4-3 after each solution, and then reapplying the...
integer programming algorithm. This can be continued until the problem becomes infeasible because of a condition similar to 4-10.

The procedure of gradually increasing requirements for any given set of resources (i.e., systems engineering, programming and testing) has another advantage. By acquiring several solutions near the infeasibility region, the components which are subject to change as the constraints become more constrictive can be determined. When resources are ample to meet the requirements, only the most desirable components are selected; however, as the infeasibility boundary is approached, some less desirable components may be chosen instead. The integer programming algorithm makes it possible to schedule these additional components and remain within the constraints.

Thus if the solution to the integer problem when the constraints are not binding is

\[ 4-11 \quad Z^*_{\text{minimum}} = \sum_{j=1}^{S} P_j X_j^* + \sum_{j=S+1}^{L} P_j Y_j \]

for \( X_j^* = 1, \ Y_j = 0 \), and where \( \{P_1, P_2, \cdots, P_S\} \) are in their minimum ordered state. Then as the requirements are increased, and one or more constraint become effective, the solution to 4-11 now includes in the set \( \{P_1, P_2, \cdots, P_S\} \) some priorities which are larger (less desirable) from the set \( \{P_{S+1}, \cdots, P_L\} \).

**Determination of Feasibility Region**

Sensitivity analysis is also the determination of one or more of the planes of the feasibility region. The manager should know which
resource is overallocated, and how much it could be reduced to remain feasible. This can be compared to measuring the distance from the constraint sum for the last feasible solution, and the plane of the feasibility region. Thus \( R \) is this distance, or the amount a resource is overallocated, or surplus,

\[
4-12 \quad R_i = C_i - \sum_{j=1}^{I} A_{ij} X_j^* \quad \text{for some group } i \text{ when } X_j^* = 1 \text{ in the optimum solution.}
\]

**Development of a Priority Assignment**

Many software contracts with the government include provision for the phased delivery of software components. Sometimes commercial software contracts also include a schedule of deliveries of components to different locations. A provision for incremental delivery allows some early computer capability, which is valuable for integration of the computer complex with other parts of the system. Since large computer installations are very costly, it is important to utilize these facilities as soon as possible after they are installed. This is normally long before the complete set of uniquely designed software is available.

The customer's goals may not coincide with the software producer, and for this reason, a negotiation session is often valuable. The producer wants to get the complete job finished as soon as possible, and not spend part of his time in testing and installing a subset of the total package. The customer realizes that partial checkout on site
is more valuable than in the producer's facility, since integration problems can be identified early and corrected.

These considerations were present in the software contract negotiation and execution under study in this thesis. Whether any numerical priority ranking was used initially in identifying the components to be delivered in the first phase is unknown. However, ranking of priorities was in evidence when the contract was renegotiated two years later, and has been a factor in subsequent decisions.

MacCrimmon\(^2\) used a weapon system example to demonstrate a numerical evaluation of a weapon system with multiple attributes. Before such a numerical assignment could be made, a relationship between desirable attributes and system components must be made. Some of the attributes of a command and control system for a satellite were identified by various working groups before the work statement was negotiated. It is assumed these attributes were considered by the negotiating committee when the contractual delivery schedule was actually formulated.

\(^{21}: p. 10.\)
CHAPTER FIVE

PROBLEM DATA AND MECHANIZATION OF THE SOLUTION

Background of Software Scheduling Problem

The data for this problem has been acquired while participating in the sequence of events surrounding the negotiations on a large software contract. The contract was awarded as a letter contract initially, which was to be made more definite subsequently with a statement-of-work. In order to develop a statement-of-work, a group of specialists from several companies was directed to meet for several months to develop a list of command and control requirements. One of the companies represented had been responsible for the design and development of the satellite, so these specialists could define requirements in terms of what was necessary to command and control the satellite. Other specialists were familiar with the design and limitations of the computer complex and ground equipment necessary to communicate with the satellite. This ground equipment required a transmitter for sending commands, and receiving equipment for accepting the telemetry signals from the satellite and converting them to useful information for the station operators.

This committee of specialists produced a two volume document which then had to be reduced to a few words which described the different software functions which would meet these requirements.
The following table lists most of the functions which went into the command and control work statement, although some in this list were implied rather than specifically named. The work statement defined an Initial Version software package and a Final Version software package.

Sources of Data

The data for the integer programming model which is being explored in this thesis was gathered on the scene of the company producing the software. Some of the data was collected by observation, confirmed by interviews with members of the various participating groups, and from interviews and reports of interviews with company management furnished by other interested parties.

Four types of data are required for the linear equations and objective functions which comprise the software scheduling model:

1. Priorities of the software functions or components
2. Requirements for the delivery of these components
3. Estimated costs of the components for the three functional groups.
4. Manpower availabilities

As touched upon in the section on the background of the project under study, the original priorities were set down by the Statement of
<table>
<thead>
<tr>
<th>Software Function Number</th>
<th>Function Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GDS Controls and Status Processing</td>
</tr>
<tr>
<td>2</td>
<td>Satellite Commanding</td>
</tr>
<tr>
<td>3</td>
<td>2260 Display (C&amp;C) TOR, Printout</td>
</tr>
<tr>
<td>4</td>
<td>Log Tape Processing</td>
</tr>
<tr>
<td>7</td>
<td>GCN Processing (Hi Speed)</td>
</tr>
<tr>
<td>8</td>
<td>Analog Processing</td>
</tr>
<tr>
<td>9</td>
<td>Discrete Processing</td>
</tr>
<tr>
<td>10</td>
<td>Digital Processing</td>
</tr>
<tr>
<td>11</td>
<td>Momentum Estimation</td>
</tr>
<tr>
<td>12</td>
<td>Other Offline Processing (Orbit Maneuver, Ipad Adjustment)</td>
</tr>
</tbody>
</table>

(continued)
### TABLE 5-3

**PRIORITY COMPARISON -**

**TWO SPECIALISTS FROM TWO COMPANIES**

<table>
<thead>
<tr>
<th>Software Function No.</th>
<th>Spacecraft Contractor</th>
<th>Software Contractor</th>
<th>Final Contract Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRW</td>
<td>RM</td>
<td>WD</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1-6</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1-6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>7-16</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>7-16</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>7-16</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>19-28</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>19-28</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1-6</td>
<td>2</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>1-6</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>1-6</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>15</td>
<td>7-16</td>
<td>4</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>19-28</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>7-16</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>7-16</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>19-28</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>1-6</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>19-28</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>11</td>
<td>7-16</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>15</td>
<td>19-28</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>19-28</td>
<td>3</td>
</tr>
<tr>
<td>21</td>
<td>10</td>
<td>19-28</td>
<td>3</td>
</tr>
<tr>
<td>22</td>
<td>9</td>
<td>28</td>
<td>3</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>28</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>12</td>
<td>19-28</td>
<td>3</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>7-16</td>
<td>3</td>
</tr>
<tr>
<td>26</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>27</td>
<td>15</td>
<td>19-28</td>
<td>4</td>
</tr>
<tr>
<td>28</td>
<td>7</td>
<td>7-16</td>
<td>4</td>
</tr>
</tbody>
</table>

**Notes:** EH is reversed and normalized (graphically) to a maximum of 27, minimum of 1.
<table>
<thead>
<tr>
<th>List of Software Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority 1 thru 27</td>
</tr>
<tr>
<td>1 most Important</td>
</tr>
<tr>
<td>- GDS Controls and Status Data Processing</td>
</tr>
<tr>
<td>- Satellite Commanding (Single)</td>
</tr>
<tr>
<td>- 2260 Display (C&amp;C)</td>
</tr>
<tr>
<td>- Log Tape Processing (Logging)</td>
</tr>
<tr>
<td>- Command Summary (For SCF)</td>
</tr>
<tr>
<td>- Satellite Commanding History (Print and Log Commands Sent)</td>
</tr>
<tr>
<td>- GCN Processing (Hi Speed)</td>
</tr>
<tr>
<td>- Analog Processing</td>
</tr>
<tr>
<td>- Discrete Processing</td>
</tr>
<tr>
<td>- Digital Processing</td>
</tr>
<tr>
<td>- Momentum Estimation</td>
</tr>
<tr>
<td>- Other Offline Processing (Orbit Maneuver, Ipad Adjustment)</td>
</tr>
<tr>
<td>- Data Base Refresh Automatically</td>
</tr>
<tr>
<td>- Telemetry Quality Logic (All Data Logged) (2 Analogs cause alarm etc)</td>
</tr>
<tr>
<td>- Block Commanding</td>
</tr>
<tr>
<td>- Derived Parameter Processing (Including limit test)</td>
</tr>
<tr>
<td>- Alarm Processing Improvement (GDS Cue, clear out of order)</td>
</tr>
<tr>
<td>- GDS Control Improvements (By Mneumonic), (Limits in Engineering Units)</td>
</tr>
<tr>
<td>- 2250 Displays Status</td>
</tr>
<tr>
<td>- Command Constraints</td>
</tr>
<tr>
<td>- Automatic Retry</td>
</tr>
<tr>
<td>- Subsystem Verification</td>
</tr>
<tr>
<td>- Extraneous Configuration Alarms (On - Off)</td>
</tr>
<tr>
<td>- Time Critical Command Cue</td>
</tr>
<tr>
<td>- Log Tape Processing (Engineering Units) (Selection of Interval, parameter, Point on Change)</td>
</tr>
<tr>
<td>- Program Function Keyboard and Light Pen for Commanding</td>
</tr>
<tr>
<td>- Full Sample of Telemetry (Variations in Time Granularity)</td>
</tr>
</tbody>
</table>
The only introduction to the questionnaire was an oral statement that the specialists should envision this list of software functions in terms of a new system, and then set a priority on each one. They were told that they could use either column according to the instructions at the top. Several of the participants filled out both columns. Each of the specialists was interviewed in his office individually.

Qualification of Experts Surveyed

Some of the experts surveyed were the same ones who were influential in the decision of the negotiating committee of executives that established the statement of work. The statement of work for the contract established the requirements for software components which had to be delivered in the first and second periods. The delivery dates of these two versions of the software were also defined. However, the experts did not recommend any particular delivery dates, as these were based on other system schedules such as the availability of a partial and a complete command and control capability. These capability dates were set by some higher level strategy group in the Pentagon.

Five experts in the Software Company were surveyed. Two of these experts were very influential in the definition stage of the contract when proposals for the contents of the first and second period deliveries were presented to the negotiating committee. These two have the initials of WDD and EH. One of these was the section manager.
of the Systems Engineering Department. He was one of the earliest engineers to study the various attributes which were desirable in the command and control, and helped interpret these into software features or components. Previously, he had considerable experience in other defense companies, and understood the different subsystems in the satellite and how they worked. This enabled him to quickly visualize the requirements for monitoring the satellite by means of the command and control software. Through WDD’s leadership, the system engineering group rose to its point of maximum effectiveness.

The other influential expert, EH, was head of the software programming group. Although his understanding of the satellite was not comparable to WDD, he did understand what impact implementing the many proposed software features would have on his computer resources. EH was probably the most influential in attempting to keep the total number of components required for delivery down to a minimum during the first period. By the time the survey was filled out, EH had spent several months on site actually commanding the satellite, so that some of his opinions may have been modified compared to his feelings when the original schedule was established. Therefore, of all the experts, his survey is the least likely to resemble the priority structure he would have advocated at the beginning of the contract.

The two other experts, KOH and DP, played a less dominant role in the policies of the company. KOH was familiar with all of the features contained in the survey, and presented briefings to inside management and design reviews with the outside contractor on the detailed design of many of these features. He had also performed
analysis on the different actions which the users must take in order to use the software to control the satellite.

DP is a section manager of the testing group for the command and control. He did not influence the decisions on the original schedule incorporated in the statement of work. He became very knowledgeable about the command and control software features however, since he personally led the group that designed the procedures for testing this software. He also spent several months on site testing the software and witnessing its use under actual conditions. Thus, the priorities which he set for each software function were based on experience and the needs which became evident while only the first ten features were available for use.

The final expert in the software company to be questioned was from the program office. CEK was especially sensitive to the needs of the customer, and the importance of providing an adequate number of software functions to satisfy the contractor. He also has a thorough technical knowledge of the satellite and what software is necessary for command and control. His opinions are sought after and valued on all questions relating to the scheduling and definition of any contemplated software features.

Two priority samples were obtained from the Satellite Company. One, marked TRW, represents a letter which was solicited by the U. S. Air Force, about six months before the rest of the survey was taken. The Air Force had requested the Satellite Company to furnish
it with priorities for those software functions which were required to command and control the satellite which that company had produced. The Satellite Company was also vitally interested in the adequacy of this software, since there was an incentive on the lifetime of the satellite. The software was the principal means by which the subsystems in the satellite could be monitored, and different components of the satellite were switched off and on as conditions changed in the satellite. It is presumed that this letter, then, represented the consensus of many experts and of several levels of management in the satellite company.

The second expert, RM, was very influential in setting forth the original requirements for the software functions. RM was not in favor of delivering part of the software functions at the end of one period, and the remainder later. In fact, he felt that the total list of functions should have been longer. He added some of his own to the list. And he really would have liked to have all of these software functions immediately. RM was not at all sympathetic with the manpower and time constraints which the software managers faced in developing this software. His knowledge of the satellite was, however, unsurpassed by any of the experts interviewed.

Responses to the Questionnaire

Generally the specialists felt that in spite of the number of components being considered, priorities fell into place rather easily. They stated orally that there are two or three software components which are necessary to make the system function usefully.
### TABLE 5-5

**INTEGER PROGRAM SELECTIONS FOR PERIOD I**

*(Without Cost Restraints)*

**TEN FUNCTIONS**

<table>
<thead>
<tr>
<th>Software Function No.</th>
<th>1-28</th>
<th>1-28</th>
<th>1-28</th>
<th>1-100</th>
<th>1-100</th>
<th>1-28</th>
<th>1-28</th>
<th>1-100</th>
<th>1-100</th>
<th>1-28</th>
<th>1-28</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Simple Majority (5 or more):** 1, 2, 3, 4, 7, 8, 9, 10, 13, 14, 16

**Contract:** 1, 2, 3, 4, 7, 8, 9, 10, 11, 12

**Contract (Cost):** 1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 18
After the first few were selected, it seemed that the judges could rather easily allocate priorities to the rest. Only one (RM) stated that such a selection was very unrealistic. His special concern was with the safety of the satellite, and he wrote his criticism at the top of his questionnaire: "For most of these functions, this is like saying what is priority of heart, lungs, intestines, brains, etc." His priorities were expressed as a range of values for each function. The mechanism which the negotiating committee used to agree on a statement-of-work is difficult to establish. One U. S. Air Force major who was prominent in the discussions refused to fill out the questionnaire, indicating he did not feel qualified to evaluate these functions.

A number of the surveyed experts wrote individual remarks on the sheet. This was partly in response to the fact that the author had some influence on the generation of future command and control requirements, so as a byproduct, some additional software functions were listed which they felt would make the list more complete. This also points up the fact that the questionnaire was taken seriously, and was completed to the best of their ability.

One of the experts filled out the right hand column, which should have been filled out as a rating from one to 100. However, the method which the expert chose to use was actually ranking, since there were no gaps and in fact, only showed four levels. This expert, WDD, may
have used this column because the instructions indicated ties were permitted on this side. Three of the experts filled out both columns. One of the three was inconsistent between the two columns. Two of the experts filled out the right hand column making 100 the most important. One of these two filled out the left hand side with one to six as most important, and the right hand side with 100 as most important.

Rough Comparison of Priorities

The most important consideration for the committee was to identify those functions that had to be delivered in the Initial Version. Looking at the four most influential specialists' priority ratings in Table 5-3, a reasonable division into two parts for the specialists' ratings would be at their half-way point. The contract initially had actually only two priorities, Initial and Final. The number 1's in Final Contract column (Figure 5-9) indicate the Initial Version requirements. TRW should be split just below 7, RM just above 7-16, WD above 2, and EH above 9.2. Some of these divisions may seem arbitrary, but they are supported by the author's knowledge of the functions and of the specialists' area of interest.

The allocation of components for the Initial Version would then consist of the following for at least three of the specialists: 1, 2, 3, 4, 8, 9, 10, 13, 14, 16.

Thus, 7, 11 and 12 would have been excluded from the initial delivery; but 13, 14 and 16 would have been included. These allocations were born out by experience in the field of which the specialists were
certainly aware of when the questionnaires were filled out. It may be that the work statement committee actually reflected the opinions of the specialists at the time of the negotiation.

Ten Most Important Components Based on Majority Opinion

The Rip 30C Integer Program was used to sort the priorities of each judge. This was done by using the priority structure set by each judge (expert or specialist) as the objective function for the Integer Program, and using a demand inequality without any cost constraints. An auxiliary program was designed to increment the demand function by one, and then solve the new integer programming problem for each of the specialist's priority structure.

The results of these solutions when the requirement was set for ten components are as shown in Table 5-5. Nine Priority structures are indicated by initials and whether the questionnaire was answered in terms of one to twenty-eight or one to one hundred. The number of the software function is shown on the left column. The total number of selections for any given software function is indicated in the right column. For example, for software function number one, eight priority structures indicated this component should be included for Period I, (assuming Period I was required to have ten software functions).

Using a simple majority voting technique, the right hand total column can be used to discover the majority opinion on which components should be delivered among the first ten. These are:

1, 2, 3, 4, 7, 8, 9, 10, 13, 14, and 16.
By way of comparison, the negotiating committee had indicated the same list of components except for 13, 14, and 16. The work statement had specified components 11 and 12 instead. These results agree with those from the rough comparison of priorities.

Comparisons Based on Ranking and Rating

Six of the priority structures set by experts seem to be most applicable to the ranking method of summation. There were no significant gaps in their numerical judgmental system, and thus no special weighting could be attributed to their selection beyond ordinary preference indications. The six selected for ranking summations were: DP, RM, CE, KOH, WDD, and TRW.

The criterion values were summed across judges, and then these values were totalled to form a grand sum. The composite weight of each criterion was then derived by dividing individual criteria values by the grand sum. This is the same technique that was used by Eckenrode. The simple priority structure resulting from this summation showed the following ten components would be selected first by these experts:

2, 7, 8, 9, 10, 13, 14, 15, 17, and 23.

Four of the judgments seemed to fit the rating category. These were DP, RM, and CE as found on the right hand side of their sheets, and EH, who filled out one side only. EH and RM had to be reversed. RM indicated on the back of his sheet how he rated the components:

100 indicates basic item needed to have a minimum system

37: p. 184.
95 indicates item needed to have a workable system
90 indicates items needed in an operation system
85 indicates items needed for a Turnover Version (Final) operational system

Because of these notations, it was fairly obvious how to reverse his priority rating. Unfortunately, two of the components were marked with a question mark, so the author judged that these were of less significance than any of the others. EH was reversed on a simple graphical basis, by setting the abscissa from 0.1 to 100, and the ordinate as shown on EH's sheet, 5 to 100. A decreasing straight line related the two coordinates.

The mathematical summation, again followed Eckenrode's methods for rating. The results of these calculations showed disagreement on two components in evaluating the ten most desirable. The ten selected on a rating basis were:

1, 2, 7, 8, 9, 10, 14, 15, 16, and 23.

No. 1 and 16 were chosen over numbers 13 and 17 for the ranking system.

As the time for testing the software approached, and the details of the functions which were to be tested for the initial version, it became evident that the Statement of Work was not being followed explicitly. Some functions were not furnished initially, such as

block commanding. However, ten functions seemed to be about the number the negotiating committee had expected to be delivered.

The delivery requirements for the second period were renegotiated in order to have an early capability on site. Although nine and possible ten components were requested by the Air Force, the contractor only committed to six. Later the contractor agreed to a contract change which included two more components, but then he failed to produce these in time for testing.

In searching for equations to be used in estimating the costs of developing computer programs, V. LaBolle\textsuperscript{5} and others at SDC concluded a meaningful cost factor could be derived from the number of new machine instructions. In the original estimate for the model, Table 5-2, the number of instructions were not available. The number of instructions may not be really applicable for system design and testing functions. However, some of the factors which LaBolle\textsuperscript{6} called out as contributing to increased costs also apply here. These are innovation in system, complexity of overall system, first programming effort on computer, and program developed away from operational location. The original estimate was made by first determining how many people were assigned to specific software functions over a period of time, and then modifying these by other factors such as those mentioned by LaBolle.


\textsuperscript{6} 20: p. 568.
In the revised cost estimates, the systems and test incremental costs were retained, but the programming costs were revised upward after counting the number of instructions for functions 1 through 18 and function 26. The original estimate of the model is contained in Table 5.2, the revised estimates are in Table 6-3.

**Manpower Availabilities**

In addition to the priority structures and incremental costs for each of the components, the availabilities must be included in the problem to attain a solution. Looking at Figure 5-1, some calculations are required to establish the total amount of manmonths available for system engineering, programming and testing.

If the management of the software contractor had actually thought that the original contractual schedule was feasible, then the planned manpower available up to the end of the eight months would have been as follows:

- System Engineering: 49 manmonths
- Programming: 38
- Testing: 19

Would these availabilities have been enough to make the problem feasible? The system engineering would have been inadequate, as well as the testing resources. At least 123 manmonths of system engineering were needed to design the Period I software for ten components, and to get the nine most desirable, 138 manmonths were needed. Similar estimates showed that testing was short. Actually, as
FIGURE 5-1
MANPOWER CHART
mentioned earlier, the programming estimates proved to be too low compared with the actual instructions produced in the software.

The actual results of this contract showed that 20 or 30 more manmonths of programming was required, resulting in a slipped schedule. Testing then had time to revise their procedures and finally test the Period I software.

After arriving on the site with the Period I software, it soon became evident that the systems engineering was inadequate. Several more months were required to make changes in the program. Thus the inadequacies of the supply of systems engineering may not show up until the final product is evaluated, as was the case here.

If Period I had been delivered on time, would Period II have been any better. The manpower graph shows that the following manmonths were available.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System Engineering</td>
<td>167.5</td>
</tr>
<tr>
<td>Programming</td>
<td>175</td>
</tr>
<tr>
<td>Testing</td>
<td>35</td>
</tr>
</tbody>
</table>

The testing was estimated based on the time left after their return from the site.

Completing the original contract on time was still not possible. Even after dropping Function No. 21, Automatic Retry; No. 24, Time Critical Command; and No. 26, Miscellaneous; 230 manmonths would still be required for system engineering.

After return of the programming and testing personnel from the site, the contract was renegotiated (in return for certain other considerations) so that only seven components were required,
Numbers 13 through 18 and 25, plus parts of 4, 5 and 6. Manpower was more than adequate for systems engineering and test requirements, and the renegotiated part of the contract was successfully completed on time.

Results for a large number of assumed availabilities are presented in Chapter Six, including some results from using the resources described above. In order to make this problem feasible with the resources available the demand for components delivered must be reduced. This, as noted, was actually done for the second period delivery. The remaining components, still undelivered, will be delivered in subsequent periods by other contractors.
CHAPTER SIX

RESULTS AND CONCLUSIONS FROM INTEGER PROGRAMMING SOLUTIONS

This chapter contains an examination of a large number of solutions furnished by the RIP 30C algorithm. These solutions are used to furnish insight into the sensitivity of the model. In addition, the results from using automatic adjustment of the objective function are presented. Finally, the selection of components resulting from ranking of multiple judgments is discussed.

Since the scheduling depends on a number of different types of parameters, the effect of variations in these parameters is of interest. Because of a survey made of seven specialists who furnish individual priority structures for the software components, solutions using these structures reveal certain characteristics of the integer programming algorithm. These characteristics might not have been obvious by simply examining the priority structure dictated by the contract.

The other parameters to which the solution may be sensitive are variations in cost structures, availabilities and delivery requirements.

Fortunately, the incremental demand procedures mentioned in Chapter Four were available to obtain a wide range of solutions. For instance, solutions were obtained when requirements were varied from one to fifteen components for no cost and revised cost
constraints\textsuperscript{3} and one to eleven for the initial cost constraints.\textsuperscript{4} These and numerous other problem solutions are used in the following analysis.

The algorithm in the RIP 30C program appears to be insensitive to changes in parameters unless the constraints force a particular selection of variables. Sometimes the solution appears to favor a selection of components yielding a sum of the coefficients in one of the cost expressions which is the furthest from the infeasibility boundary, but this is not always the case. The sensitivity of the algorithm would be unquestioned if none of the priorities in the objective function were tied, it would then seek and find the minimum sum of these priorities in every case. However, priorities are seldom so rigidly structured in the case of software components--only two the seven judges in the survey felt this was possible, CE 28 and DP 28.

Some of these effects can be discovered by analyzing Table 6-1. Table 6-1 shows 20 first period solutions including the optimum solution using the contract priority. Ten of the solutions are with no constraints, indicated by \( \emptyset \). These show the ten minimum priorities (most desirable) for the specialists surveyed and the contract priority as determined by the RIP 30C. The enforcement of the constraints uncovered some differences in the components selected, mainly because of system engineering constraints. Programming and test

\textsuperscript{3} Table 6-3.

\textsuperscript{4} Table 5-2.
TABLE 6-1
SOLUTIONS ACCOMPANYING ORIGINAL COSTS AND NO REPAIRS

<table>
<thead>
<tr>
<th>Contract</th>
<th>RM 100's</th>
<th>GE 28's</th>
<th>RM 28</th>
<th>WDD</th>
<th>TRW</th>
<th>KGH</th>
<th>EH</th>
<th>DP 28</th>
<th>DP 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>NC</td>
<td>I</td>
<td>NC</td>
<td>I</td>
<td>NC</td>
<td>I</td>
<td>NC</td>
<td>I</td>
<td>NC</td>
</tr>
<tr>
<td>1.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>2.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>3.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>4.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>5.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>6.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>7.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>8.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>9.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>10.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>11.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>12.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>13.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>14.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>15.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>16.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>17.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>18.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>19.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>20.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>21.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>22.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>23.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>24.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>25.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>26.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>27.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
<tr>
<td>28.</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
<td>φ</td>
<td>X</td>
</tr>
</tbody>
</table>

Difference 2: 1 2 same same 2 1 2 2 1

X = Original Costs Constrained to
350 MM Programming
150 MM Systems Engineering
50 MM Test

Ten Components to be Scheduled

φ = No Constraints
availabilities were more than adequate in all cases. The differences in components selected for the two cases is tabulated in the last row titled "Difference."

This change in the set of components selected is due to the effect of constraints when a higher priority component is selected. However, the new component may have the same priority as the rejected one, but the manpower coefficient assigned to the new variable may have been smaller and necessary to maintain feasibility.

Examining the selections when only one difference resulted from the addition of constraints, analysis shows:

<table>
<thead>
<tr>
<th>Specialists</th>
<th>Component Number</th>
<th>Component Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>CE 28</td>
<td>23</td>
<td>16</td>
</tr>
<tr>
<td>EH</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>Contract</td>
<td>18</td>
<td>4</td>
</tr>
</tbody>
</table>

Some facts may be drawn from this analysis. CE 28 was forced by the new constraints on systems to go to a higher priority component. EH and the Contract had the same sum for an objective as before. The reasons for the new component selection in those latter two cases requires additional explanation.

The EH solution showed that smaller values for the system coefficient were substituted by selecting component 18 instead of 7. However, the solution would have been feasible without the change. One might conclude that the algorithm chose a greater distance from the infeasibility boundary, since the total system engineering cost for this period was reduced by selecting component 18.
The new contract solution, like the EH solution, resulted in no change in the objective but, the selection of component 18 is closer to the infeasibility region than 4 would have been. Therefore, the algorithm does not consistently maintain the greatest distance from the feasibility boundary.

Solution differences of two components are less interesting since either new choice could have stimulated the algorithm to choose a different solution branch. For instance, RM solutions:

<table>
<thead>
<tr>
<th>Specialists</th>
<th>Component Number</th>
<th>Component Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>RM 100</td>
<td>27, 28</td>
<td>11, 16</td>
</tr>
<tr>
<td>RM 28</td>
<td>13, 28</td>
<td>11, 16</td>
</tr>
</tbody>
</table>

The sum of the rejected system engineering costs for RM 100 is 66, the sum of the two new coefficients is 15. Since the sum of the new objective function is larger due to higher priority numbers, it is obvious that this was consistent with the need to meet the constraints. The new solution of RM 28 shows similar characteristics, one of the selected component priorities is higher than the old component previous selected.

**Apparent Inconsistencies of Individual Specialists**

Obviously the algorithm would not ordinarily be used to detect inconsistencies in priority ratings. However, if solutions are based on the two different priority structures of an individual specialist, the results can be misleading. For instance, two judges used both methods of setting priorities, Ranking and Rating, exhibited by RM 28.
and RM 100 for one specialist, and DP 28 and DP 100 for the other. DP did not exhibit any internal inconsistencies that were revealed using RIP 30C as seen in Table 6-1. However, the results for RM would appear to be inconsistent. The rough data for the priority structure shows it to be consistent but there are numerous ties in priority between the components. For example, examination of the sum of the coefficients for system engineering for the two solutions revealed the sums to be the same. The use of the available resources for system engineering for both solutions was 136, with a slack of 14 remaining. In addition, the sum of the objective function for the two solutions was equivalent. Thus with many ties in the objective function and the same cost coefficients for several components, apparent differences in solutions can be misleading.

Sensitivity to Forcing Another Component Into the Solution

Table 6-2 shows the selections offered by the integer program algorithm with original cost and requirements set for eleven and nine components. If the differences between the solutions for any given priority structure differs by more than two components, the rejection of some components might have been caused by constraints. A logical question for a manager might be, "What would be the effect of insisting on the components which were discarded?"
Examination of five priority structures in Table 6-2 that exhibit a difference of more than two components in the referenced solutions are shown below:

<table>
<thead>
<tr>
<th>System Engineering Components</th>
<th>Coefficients</th>
<th>New Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE 100</td>
<td>New</td>
<td>Old</td>
</tr>
<tr>
<td>CE 100</td>
<td>3, 15, 20</td>
<td>16</td>
</tr>
<tr>
<td>CE 28</td>
<td>3, 13, 15</td>
<td>16</td>
</tr>
<tr>
<td>KOH</td>
<td>3, 4, 20</td>
<td>1</td>
</tr>
<tr>
<td>DP 28</td>
<td>7, 10, 23</td>
<td>16</td>
</tr>
<tr>
<td>DP 100</td>
<td>7, 9, 10, 17</td>
<td>16, 18, 18</td>
</tr>
</tbody>
</table>

This compilation shows that the five specialist solutions would have infeasible solutions if the old component was retained, and that the system engineering availability would have to be raised if the discarded component had to be included with the eleven selected. When the revised programming cost coefficients given in Table 6-3 were substituted for the initial costs shown in Table 5-2, then the impact of adding components is quite different. In the following example, the impact of adding components to the revised solution would require changes in both programming and system availabilities:

<table>
<thead>
<tr>
<th>Cost Estimate</th>
<th>First Ten Components Selected</th>
<th>Prog/Systems/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>1, 2, 3, 7, 8, 9, 10, 11, 12, 18</td>
<td>350/150/50</td>
</tr>
<tr>
<td>Revised</td>
<td>1, 2, 3, 7, 8, 9, 10, 11, 12, 13</td>
<td>350/150/50</td>
</tr>
</tbody>
</table>
### TABLE 6-2

**SELECTIONS OF ELEVEN* AND NINE* COMPONENTS USING INITIAL COSTS**

<table>
<thead>
<tr>
<th>CE 100</th>
<th>RM 100</th>
<th>CE 28</th>
<th>RM 28</th>
<th>WDD</th>
<th>TRW</th>
<th>KOH</th>
<th>EH</th>
<th>DP 28</th>
<th>DP 100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>8</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>9</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

**Differential**

| 3X's | 2X's | 3X's | 3X's | 2X's | 2X's | 3X's | 3X's | 4X's |

*X = 11 components required
0 = 9 components required

Initial costs with availabilities set at:

- **Programming**: 350 m.m.
- **System Eng.**: 150 m.m.
- **Testing**: 50 m.m.
Adding any components requires an increase in the total systems manpower requirements for both cost estimates. In the initial case, programming availabilities would not be affected. But for the revised costs, the manpower allocation for programming would also have to be increased if components were added to the delivery schedule. Components 4, 5, 6 and 18 are exceptions to this statement.

Because of the large amount of computer time required for solution of a scheduling problem involving eighty-four variables (which is equivalent to scheduling any of twenty-eight components in any of the three periods), fifty-six variables can be used and the remaining variables not selected are assumed to be scheduled in the remaining period. The number of constraints in a problem also impacts computer time, and 28 inequalities are required to prevent the same component from being selected in both periods.

Fewer expressions can be used by iteratively adjusting the objective function based on the previous solution. Resolving the problem with the new objective function usually provides a solution with individual components properly scheduled. This method was explained in Chapter Four. The success of this method is demonstrated by the following results.

For instance, when only 12 inequalities are specified as cross expressions to prevent double selection, four components were selected in two periods as follows:

<table>
<thead>
<tr>
<th>Period 1</th>
<th>1, 2, 3, 4, 12, 13, 14, 18, 25</th>
<th>400/140/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 2</td>
<td>5, 6, 7, 9, 10, 11, 13, 14, 18, 25</td>
<td>400/100/100</td>
</tr>
</tbody>
</table>
It can be seen that the last four components of both periods are the same for the availabilities indicated and the revised programming cost coefficients. By raising Period 2 priorities on these last four components above Period 1, the selection was as follows:

Period 1
1, 2, 3, 7, 13, 14, 17, 18, 25, 27

Period 2
4, 5, 6, 8, 9, 10, 11, 12, 23, 28

In another problem, with larger programming and systems availabilities, even with seventeen cross product inequalities double indication for 17, 18, and 25 were still obtained as follows:

<table>
<thead>
<tr>
<th>Availabilities</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>1, 2, 3, 4, 7, 9, 10, 17, 18, 25</td>
</tr>
<tr>
<td>Period 2</td>
<td>8, 11, 12, 13, 14, 15, 16, 17, 18, 25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Availabilities</th>
<th>Components</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>1, 2, 3, 4, 7, 9, 10, 17, 18, 25</td>
</tr>
<tr>
<td>Period 2</td>
<td>5, 6, 8, 11, 12, 13, 14, 15, 16, 27</td>
</tr>
</tbody>
</table>

By raising the priorities on second period components selected in Period 1, and removing system constraints in period 2 duplicate components were removed:

<table>
<thead>
<tr>
<th>Components</th>
<th>Prog./System/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>1, 2, 3, 4, 7, 9, 10, 17, 18, 25</td>
</tr>
<tr>
<td>Period 2</td>
<td>5, 6, 8, 11, 12, 13, 14, 15, 16, 27</td>
</tr>
</tbody>
</table>

Similar results are obtained when cross period expression are furnished for the first sixteen, number 18 and number 28 components.

<table>
<thead>
<tr>
<th>Components</th>
<th>Prog./System/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>1, 2, 3, 4, 5, 6, 7, 8, 9, 25</td>
</tr>
<tr>
<td>Period 2</td>
<td>10, 11, 12, 13, 14, 15, 16, 17, 18, 25</td>
</tr>
</tbody>
</table>
Component 25 was scheduled for both periods. But with priority raised on 25 and some changed availabilities:

<table>
<thead>
<tr>
<th>Period 1</th>
<th>1,2,3,4,5,6,7,8,9,25</th>
<th>500/140/100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 2</td>
<td>10,11,12,13,14,15,16,17,18,27</td>
<td>650/200/100</td>
</tr>
</tbody>
</table>

Component number 25 is now scheduled for Period 1, and 27 is now added for Period 2.

These examples demonstrate the utility of this method of solving problems by iterative solution of subproblems involving fewer expressions. It also demonstrates the care necessary to obtain useful solutions. This method can be applied by making the subproblems even smaller by only including enough variables for one period at a time. The priorities of these components selected for Period 1 are adjusted so that these components will be omitted from solutions for other periods.

It was found the computer could make these adjustments and resolve the problem in less time than a two period problem. The results were not quite identical, since the objective function is now applied to fewer components simultaneously than in the two period problem.

For instance, in the revised cost problem, the following results were obtained using no cross expressions to prevent double selection instead of twelve or more such expressions discussed in previous pages:

<table>
<thead>
<tr>
<th>Components</th>
<th>Prog./Systems/Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Period 1</td>
<td>1,2,3,4,5,7,9,10,11,12</td>
</tr>
<tr>
<td>Period 2</td>
<td>6,13,14,17,18,20,23,27,28</td>
</tr>
</tbody>
</table>
It can be seen that there are no double selections in the above list, and therefore this is a successful problem formulation.

**Conclusions**

The following conclusions can be reached by this study.

1. That integer programming can be applied against a multiple period scheduling problem with success. However, some precautions must be taken:
   a. Problems that have too many variables and inequalities may have to be reduced to subproblems for solution.
   b. When the number of cross period inequalities is reduced, duplicate component selection may result.

2. The method of raising the priorities on subproblems can be used to reduce the number of inequalities in a problem without the above difficulty with a reduction in computation time.

3. Integer Programming can be used to provide reliable schedules if the above precautions and aids are used.

**Selection of Components using Ranking of Multiple Judgments**

In Chapter Five, it was noted that six of the specialists' survey sheets were applicable to the ranking method. If the priority structure resulting from Eckenrode's methods are reduced to simple integer numbers, it would appear as follows:

\[57: \text{p. 184.}\]
From this table, it is easy to see the unconstrained selection of ten components for the two periods. The solution for Period 1 with initial costs and constraints on systems engineering is shown below:

<table>
<thead>
<tr>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog./Systems/Test</td>
</tr>
</tbody>
</table>

| Period 1 | 2, 4, 7, 8, 9, 10, 13, 14, 17, 23 | 50 | 150 | 50 |
| Period 2 | 1, 3, 5, 6, 15, 16, 18, 19, 20 | 28 | 104 | 255 | 42 |

If the unconstrained selection would have been used, then the available systems engineering would have been exceeded in the first period. Since function numbers 4 and 10 were not selected in Period 1 because of the constraints, their priority assignments made them prime candidates for the second period selection. These solutions were
obtained using the first alternative method described in Chapter
Four: the method of raising the priorities of the components selected
in period one so that they would be excluded from selection for the
second period.

The comparison of the above solutions with the contractual model
has to be based, first of all, on the differences in priorities. This
comparison follows:

<table>
<thead>
<tr>
<th>Function</th>
<th>1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contract</td>
<td>1 1 1 2 2.5 2.5 1 1 1 1 1 1 2 2 2</td>
</tr>
<tr>
<td>Ranking</td>
<td>15 10 11 14 17 3 5 8 9 21 22 2 4 7</td>
</tr>
<tr>
<td>Function</td>
<td>16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28</td>
</tr>
<tr>
<td>Contract</td>
<td>2 2 2 4 4 10 4 4 5 2 11 3 4</td>
</tr>
<tr>
<td>Ranking</td>
<td>2 6 14 14 12 19 16 6 20 18 24 23 13</td>
</tr>
</tbody>
</table>

There is great disparity between the two priority structures. Hence,
the solutions reflect these differences.

The conclusion must be made that the ranking method did not
approximate the contractual model nearly as well as the majority
vote system described in Chapter Five. The majority vote method
was applied using the combined survey including both the ranking and
rating systems. Six of the respondents used the ranking method but
this seemed to be too small a sample to adequately test the procedure
described by Eckenrode.

\[ ^7: p. 184 \]
<table>
<thead>
<tr>
<th>SOFTWARE COMPONENT NUMBER</th>
<th>REVISED PROGRAMMING COSTS</th>
<th>FINAL CONTRACT PRIORITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58.9</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>87.0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>31.3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>30.0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>21.5</td>
<td>2.5</td>
</tr>
<tr>
<td>6</td>
<td>6.0</td>
<td>2.5</td>
</tr>
<tr>
<td>7</td>
<td>15.9</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>18.7</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1.1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>25.0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>7.5</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>45.7</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>23.8</td>
<td>2</td>
</tr>
<tr>
<td>14</td>
<td>4.3</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>43.0</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>34.8</td>
<td>2</td>
</tr>
<tr>
<td>17</td>
<td>97.3</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>15.4</td>
<td>2</td>
</tr>
<tr>
<td>19</td>
<td>97.3</td>
<td>4</td>
</tr>
<tr>
<td>20</td>
<td>20.0</td>
<td>4</td>
</tr>
<tr>
<td>21</td>
<td>20.0</td>
<td>10</td>
</tr>
<tr>
<td>22</td>
<td>80.0</td>
<td>4</td>
</tr>
<tr>
<td>23</td>
<td>80.0</td>
<td>4</td>
</tr>
<tr>
<td>24</td>
<td>80.0</td>
<td>5</td>
</tr>
<tr>
<td>25</td>
<td>24.2</td>
<td>2</td>
</tr>
<tr>
<td>26</td>
<td>99.0</td>
<td>11</td>
</tr>
<tr>
<td>27</td>
<td>97.3</td>
<td>3</td>
</tr>
<tr>
<td>28</td>
<td>40.0</td>
<td>4</td>
</tr>
</tbody>
</table>
CHAPTER SEVEN

EVALUATION AND UTILIZATION

The interest in managerial aids for software scheduling has been expressed by all managers interviewed by the author. These managers included Air Force Officers (Contracting Groups), Aerospace Managers (Consultants to the Air Force), as well as contractors. Some of the factors which will influence their decision to utilize the scheduling program discussed in this paper or some similar method are discussed below.

Detailed Cost Data

Some of the data needed to formulate the scheduling problem have been requested on requests for quotations or proposals in the past. Data have been recently furnished by contractors on the cost of numerous modifications to software. These quotation sheets included costs of each new software function in some instances. In a few cases, the different professional categories of cost were also itemized such as programming and test.
Sometimes, on request, instruction estimates are furnished for the different software features. These estimates may be furnished by other companies when the specifications are available in sufficient detail.

Priority assignments to software function can be gathered from a number of technical experts from contractors and Air Force Commands. Synthesizing techniques for combining disparate priority tabulations is performed most often in conference with the interested parties. Impasses on a coordinate priority assignments are resolved by the highest ranking member of the committee or the director of the program office.

Recently, priority ratings have been formally requested from using contractors. The responses are seldom completely satisfactory. The tendency is to group most of the functions into a "Mandatory" category, and leave only a few for the "Desirable but not necessary" role.

In summary, some precedent has been set for obtaining costs for each software function, although more information should be required in the cost proposals. Priorities are often established for the different software features, although formal collection of different priority structures is still in the future.

The biggest practical difficulty in gathering enough data for the decision problem is in discovering the amount of resources available to the contractor. If the contractor is using the model, this would not
be difficult. In fact, it would be beneficial to force the managers of the different groups to look at their manpower availabilities and future demands for their services.

To get an honest appraisal from the contractor about his manpower resources for outside use may prove very difficult. A contractor would be very reluctant to indicate any limitation in his resources which would jeopardize his chances of getting the contract in question, or might result in some risk of losing a contract he is currently trying to fulfill. In other instances, however, such as the analysis of one software contractor's availabilities contained in this study, the contractor may use the argument that not enough trained people are available in the short period of time requested in the delivery schedule. When this situation arises, then the integer programming model described herein would provide an excellent means of selecting as many highly desirable priority components as possible.

One of the project managers from the Federal Division of IBM stated that availability of resources was not a problem, since IBM could and has drawn upon their programming and systems labor engineering groups all over the United States when necessary to fulfill a particular contract. Nevertheless, he was intensely interested in this study and the integer programming algorithm since there are practical time and cost limitations to such a transfer of manpower from state to state.

A Contracting Officer from another software company which had just completed a very poor schedule performance on software objected to the concept. He felt that the cost of collecting the data
from the individual managers, and updating it as previous estimates proved erroneous, would far exceed the benefits of such a managerial aid. In this author's opinion, the very fact that the top management of this company did not feel this kind of information was valuable, is the exact reason their management has been inadequate to assure reasonable schedule performance. The programming managers under him did not feel the cost for individual software functions could be estimated, but that the entire contract had to be bid on a lump sum.

One former Associate Group Director of a prominent defense company felt apprehensive about the influence of computer based algorithms on managerial decisions. He was concerned about the prestige given to any solution when generated by a computer, and the tendency of people to put faith in the solution even if the input data is erroneous. This was a very penetrating observation, although any decision model is naturally limited by the accuracy of the raw data and the assumptions used in making a decision. The same observation must hold for scientific or engineering models as well.

**Conclusions**

The use of an integer programming algorithm for generating schedules in the software industry is not apt to find wide acceptance for some time to come. Nevertheless, many companies have and will continue to encourage the development of more optimum management methods. Recently, the RIP 30C algorithm was applied by the Aerospace Corporation to the decision problem of scheduling communication with many different satellites with a ground-based
Satellite Control Network. Other algorithms are also being tried for this special case of fixed resource availabilities and variable demands. This is of course an entirely different application than the scheduling of software development. But as semitechnical applications increase, management will feel the influence of these successes, and be willing to try for themselves.

Two further comments are in order. Integer programming can be used for scheduling many other activities in addition to the development of software components. The successful demonstration of the methodology in this thesis should encourage other potential applications. The process of building and testing the model for scheduling was, in itself, quite informative. Many managers find that the collection and analysis of the data necessary to build a model provides valuable insight into the parameters affecting a decision, even when the solutions furnished by an algorithm are not immediately adopted for the problems at hand.
BIBLIOGRAPHY


