Choosing an Optimal Set of Libraries

Oded Berman
University of Toronto, Toronto

Michal Cutler, Member IEEE
State University of New York, Binghamton

Keywords — Software reliability, Software library, Software cost, Optimization, Heuristics, Branch & bound

Abstract — This paper presents optimization models for selecting a subset of software libraries, viz, collections of programs, residing on floppy disks or compact disks, available on the market. Each library contains a variety of programs whose reliabilities are assumed to be known. The objective is to maximize the reliability of the computer system subject to a budget constraint on the total cost of the libraries selected. The paper includes six models, each of which applies to a different software structure and assumptions. A detailed branch & bound algorithm for solving one of the six models is described; it contains a simple greedy-procedure for generating an initial solution. For solving the rest of the models, see Berman & Cutler (1995).

1. INTRODUCTION

The relationship between reliability & cost is important for software development [8 - 10]. The reliability of the software is often a measure of its quality [6] and the cost of software programs must be considered when the user has limited budget for investing in software.

Ref [2] presented two optimization models for selecting a set of available commercial programs to maximize the system reliability, subject to a budget constraint. In model 1, redundancy of various versions of the programs is not allowed, whereas in model 2 such redundancy is permitted. Ref [3] extended the work in [2] by considering software systems where each program consists of sequence of modules which, upon execution, perform a function. Ref [4] considers the tradeoff between reliability & cost when using N-version programming in the framework of [2, 3].

This paper considers the tradeoff between reliability & cost when the problem is to choose a set of software libraries from many that are available in the market. Each one of the available libraries contains a variety of programs. The user has a list of programs that are required for the computer system, and wishes to ensure that each one of the programs required is included in at least one of the libraries chosen so as to maximize the system reliability, subject to a budget constraint.

This paper presents optimization models to derive the optimal selection of commercial libraries available in the market. Section 2 introduces the models. Section 3 describes the solution method. Section 4 discusses limitations of the models.

Notation

\( n, N \) number, set of programs required by the user
\( N = \{1, 2, \ldots, n\} \)

\( m \) number of commercially available libraries considered

\( L_1, \ldots, L_m \) m partitions (libraries) of \( L \); \( \bigcup_{i=1}^{m} L_i = N \)

\( C_i \) cost of library \( L_i \)

\( R_{ij}, R_j \) reliability of program \( j \) in \( [L_i, \text{the software system}] \)

\( \mathcal{I} \) indicator function: \( \mathcal{I}(\text{True}) = 1, \mathcal{I}(\text{False}) = 0 \)

\( Y_i \) \( Y_i(\text{library } i \text{ is selected}) \)

\( X_{ij} \) \( X_{ij}(\text{library } i \text{ AND program } j \text{ are selected}) \)

\( F_j \) frequency of use of program \( j \)

\( B \) budget available

\( S_k \) set of programs that are required to be used in a sequence

\( F_k \) frequency of use of \( S_k \)

\( v \) number of \( S_k \)

Other, standard notation is given in “Information for Readers & Authors” at the rear of each issue.

2. THE MODELS

2.1 General Models

Three models are developed separately for each of 2 cases (total of 6 models). The 2 cases are:

Case 1. No redundancy. The reliability of a particular program is the largest among selected libraries that contain it.

Case 2. Redundancy. All versions of the selected programs contribute to the reliability.

The 3 models (pertaining to 3 system structures) are:

a. Each program is stand-alone.

b. The set of required programs is used to construct 1 complex user program (system). All programs in the user’s list are executed in sequence (generating 1 stand-alone system).

c. The set of required programs is used to construct several complex user programs (systems): \( S_1, \ldots, S_v \). These complex programs (systems) are executed in sequence.

Thus there are 6 situations (Cases 1 & 2, and Models a,b,c):

C1Ma, C1Mb, C1Mc;

C2Ma, C2Mb, C2Mc.

Assumptions

1. Each library has a known cost.
2. Each program contained in each one of the libraries has a known reliability.

0018-9529/96/$5.00 ©1996 IEEE
3. The collections of library programs are \( s \)-independent.

4. Failure of different versions of the same program are \( s \)-independent.

5. Usage frequency for each program is known, viz., provided by the user.

6. Usage frequency for each required \( S_k \) is known, viz., provided by the user.

7. Each program has 2 states (it either performs well, or fails).

8. Complex systems can be generated directly by combining existing software.

2.2 Case 1: No Redundancy

Only 1 program, \( j \), is used by the system — the one with maximum reliability. The reliability of program \( j \in N \) is:

\[
R_j = \max_{\{i \in L_i, j \in L_j\}} \{R_{ij}\}. \tag{1}
\]

Consider the 3 models (indexed as in section 2.1) for this case 1:

\( \text{C1Ma}, \text{C1Mb}, \text{C1Mc} \).

The constraints, (2) - (5), are the same for each of the 3 models:

\[
\sum_{\{i \in L_i\}} X_{ij} = 1, \text{ for } j = 1, 2, \ldots, n; \tag{2}
\]

\[
\sum_{\{i \in L_i\}} X_{ij} - |L_i| \cdot Y_i \leq 0, \text{ for } i = 1, \ldots, m; \tag{3}
\]

\[
\sum_{i=1}^{m} C_i \cdot Y_i \leq B; \tag{4}
\]

\[
X_{ij} = 0, 1, \text{ for } Y_i = 0, 1, \text{ for } i = 1, \ldots, m, \text{ and } j \in L_i. \tag{5}
\]

- Constraints (2) guarantee that exactly 1 program, \( j \), is selected.
- Constraints (3) ensure that, if \( Y_i = 0 \) then no program \( j \) for \( j \in L_i \) can be selected; else, at most, all programs in \( L_i \) can be selected.
- Constraint (4) is required so that the budget is not exceeded.

Each model has its own objective function (ObFn) to be maximized, subject to the constraints, (2) - (5).

\[
\text{ObFn}\{\text{C1Ma}\} = \sum_{j=1}^{n} F_j \cdot \Psi_{1j}; \tag{6}
\]

\[
\text{ObFn}\{\text{C1Mb}\} = \prod_{j=1}^{n} \Psi_{1j}; \tag{7}
\]

\[
\text{ObFn}\{\text{C1Mc}\} = \sum_{k=1}^{v} F_k \cdot \prod_{j \in S_k} \Psi_{1j}. \tag{8}
\]

3. SOLVING PROBLEM C1Ma

Problem C1Ma is an integer program (and can be solved as one) while the remaining 5 problems have non-linear objective functions.

All 6 problems can be solved using branch & bound algorithms where the branch is on the \( X_{ij} \). The main ideas of
the algorithms are explained via problem C1M; see [1] for
details on all the problems. The example in the remainder of
this section illustrates the algorithm.

3.1 Statement of Example
There are 5 programs and 4 libraries.

\[ N = \{1, 2, 3, 4, 5\}, \]

\[ L_1 = \{1, 2, 4\}, \]

\[ L_2 = \{2, 3, 5\}, \]

\[ L_3 = \{3, 4, 5\}, \]

\[ L_4 = \{1, 2, 5\}. \]

\[ R_{1,1} = 0.90, R_{1,2} = 0.80, R_{1,4} = 0.70; \]

\[ R_{2,2} = 0.85, R_{2,3} = 0.90, R_{2,5} = 0.95; \]

\[ R_{3,3} = 0.70, R_{3,4} = 0.90, R_{3,5} = 0.80; \]

\[ R_{4,1} = 0.90, R_{4,2} = 0.80, R_{4,5} = 0.85. \]

\[ C_1 = 10, C_2 = 15, C_3 = 20, C_4 = 17. \]

\[ B = 44. \]

\[ F_j = 0.2 \text{ for all } j \in N. \]

3.2 Observations About a Partial Solution

Four observations about a partial solution identified in the
branch & bound process are:

1. If \( (X_i,j = 1) \) in a partial solution, then \( Y_i = 1. \)
2. If \( \left( X_{i,j} = 1 \right) \text{ AND } (Y_i = 1) \) in a partial solution, then
   \( Y_k = 0 \) for all libraries \( k \) such that \( R_{k,j} > R_{j,j}. \)
3. If for a given partial solution, there is:
   a. a program \( t \) which is not part of the partial solution, and
   b. only 1 library \( L_k \) (that is not part of the partial solution) that includes program \( t; \) then set \( Y_k = 1 \) and \( X_{k,t} = 1. \)
4. If for a given partial solution:
   a. \( X_{i,j} = 1, \) and
   b. there is a program \( t, \) not selected yet in \( L_i, \) with
      reliability strictly greater than that of program \( t \) in all libraries
      not selected yet;
   then set \( X_{i,t} = 1. \)

3.3 General Procedure

Notation

\[ J \quad \text{set of all programs considered so far in a partial solution} \]

\[ Y_i = 2 \quad \text{for any library } i \text{ not yet considered in the partial solution} \]

\[ R_{e,j} \quad \text{reliability of the chosen version of program } j \text{ in the partial solution} \]

\[ Y_e = 1 \quad \text{and } X_{e,j} = 1, \text{ for any } j \in J \]

\[ R_{e,j} \quad \text{highest reliability that can be achieved for program } j: \]

\[ R_{e,j} = \max \{ R_{j,j}, Y_e = 0 \} \{ R_{j,j} \}, \text{ for any } j \notin J \]

\[ \text{LB} \quad \text{reliability of the best feasible solution} \]

\[ \text{UP} \quad \text{upper bound for a partial solution for C1M} \]

\[ C_*, C^* \quad [\text{lower, upper}] \text{ bound on cost for the solution cor-} \]

\[ \text{responding to a given partial solution } i, \text{ UP} \]

\[ I^* \quad \text{contains the indexes of libraries with } Y_e = 2 \text{ that include the most reliable versions for programs not} \]

\[ \text{selected yet.} \]

\[ \text{UP} = \sum_{j \in J} F_j \cdot R_{e,j} + \sum_{j \notin J} F_j \cdot R_{e,j}; \]

\[ C^* = \sum_{[i: Y_i = 1]} C_i + \sum_{i \in I^*} C_i; \]

\[ I^* = \left\{ i^*: \text{there exists } j \notin J \text{ with } R_{i,j} \right\} \]

\[ = \max_{[i: Y_i = 1]} \{ R_{i,j} \} \quad \max_{[i: Y_i = 1]} \{ R_{i,j} \} \}

\text{If, from (17), } C^* \leq B, \text{ then the solution corresponding to UP}
\text{is feasible and the current node on the branch & bound tree}
\text{should not be further branched out.}

When \( J = \emptyset, \text{ viz, before any program is selected,} \]

\[ C_* = \sum_{j = 1}^{n} \min_{[i: j \in L_i]} \{ C_i / |L_i| \}; \]

\( i.e., \) for every program \( j, \) take the minimum cost of program \( j \)
\text{if each program were sold separately, and if all the programs}
of a single library had the same cost. Since libraries are sold
as a whole,

\[ \text{'actual cost for program } j' \geq \min_{[i: j \in L_i]} \{ C_i / |L_i| \}. \]

Let the partial solution have \( J \neq \emptyset, \) then the lower bound
on the cost of this partial solution is:

\[ C_* = C_A + C_B. \]

Notation

\[ Q \quad \cup_{Y_i = 1} L_i; \text{ set of all programs in selected libraries} \]

\[ C_A \quad \sum_{[i: Y_i = 1]} C_i; \text{ actual cost of selected libraries} \]

\[ C_B \quad \max(C_{B_1}, C_{B_2}) \]

\[ C_{B_1} \quad \min\{C_i: Y_i = 2, L_i \not\subseteq Q\} \]

\[ C_{B_2} \quad \sum_{j \notin Q} \min_{[i: j \in L_i, Y_i = 2]} \left\{ \frac{C_i}{|L_i \cap N - Q|} \right\}. \]
When \( C^* > B \), the partial solution should not be further branched out since the partial solution cannot be extended to a feasible solution.

An initial feasible solution and LB is not easy to find. Even the minimum cost problem which disregards reliability is NP-complete. However the greedy heuristic for the Set Covering problem [6] can be used to find a low cost solution that satisfies constraints (2), (3), (5). If this solution also satisfies constraint (4) then it can serve as an initial solution and its reliability is LB. Moreover, this solution can be easily improved.

3.4 Greedy Heuristic

This heuristic finds a cheap solution.

Initialize:

\[ N_1 = N, \quad L_1 = \{1, 2, ..., n\}, \quad Y_i = 0 \text{ for } i \in L_1. \]

Iteration \( t \):

Select Library \( i^* \) such that,

\[ \frac{C_{i^*}}{|L_{i^*} \cap N'|} = \min_{i \in L_1} \left\{ \frac{C_i}{|L_i \cap N'|} \right\}. \]

Notation

\( L' \) set of libraries for which \( Y_i = 0 \)
\( N' \) set of programs not included in the partial solution.
\( N'^{t+1} = N' - L'_t \),
\( L'^{t+1} = L' \setminus \{i^*\} \),
\( Y_{i^*} = 1. \)

If \( N'^{t+1} \neq \emptyset \), set \( t \leftarrow t + 1 \) and repeat iteration \( t \).

STOP

Use this heuristic on the example.

\( N_1 = \{1, 2, 3, 4, 5\} \),
\( L_1 = \{1, 2, 3, 4\} \),
\( Y = (0, 0, 0, 0, 0). \)

Iteration 1

\( i^1 = 1 \) (since \( \min[10/3, 15/3, 20/3, 17/3] = 3\frac{1}{3} \)),
\( N_2 = \{3, 5\} \),
\( L_2 = \{2, 3, 4\} \).

Notation

\( q_t \) set of libraries in the solution
\( R(S \cup \{i\}) \) reliability of \( S \cup \{i\} \).

\( \triangleright \) Step 1

\( P = \{i: Y_i = 0 \text{ and } \sum_{j \in S} C_j + C_i \leq B\} \)

Step 2

If \( P \neq \emptyset \), find,

\( R = \max \{R(S \cup \{i\}: i \in P) = R(S \cup \{i^*\}) \} \)

If \( R > R(S) \), then set,
\( S = S \cup \{i^*\}, \quad Y_{i^*} = 1; \)
GoTo step 1. (End of step 2)

STOP

Use this procedure on the example.
BERMAN/CUTLER: CHOOSING AN OPTIMAL SET OF LIBRARIES

Step 1

\[ P = \{4\}, \ (S = \{1,2\}) \]

Step 2

\[ R(S \cup \{4\}) = 0.18 + 0.17 + 0.18 + 0.14 + 0.19 = 0.86 \]

Thus \( S = \{1,2\} \) remains as the initial solution and LB = 0.86.

3.6 Branch & Bound Algorithm

Apply the branch & bound algorithm [7]; the result is the optimal solution:

\[ Y_1 = Y_2 = 1, \ Y_3 = Y_4 = 0; \]

\[ X_{11} = X_{22} = X_{23} = X_{14} = X_{25} = 1. \]

This is also the initial solution.

4. LIMITATIONS

Assumptions 2, 4, 8 limit the usefulness of the models and indicate the need for further research. Their limitations are discussed here.

2. In reality these numbers are not available. Moreover, even when serious attempts are made to evaluate the reliability of systems, the results are not completely accurate. A possible remedy for this problem might be to assume fuzzy reliabilities such as highly-reliable in our models.

4. In reality the failure behavior of different versions is s-correlated because they are subject to the same input [11, 12]. Although s-correlation is an important issue in N-version programming, it is less severe in our models since we demand only that one version must be reliable for a given input.

8. Experience shows that some code must be added to control the execution of the system, and to deal with the compatibility of the programs in the system.

Despite these limitations, we believe that constructing reliable systems that use commercially available libraries under budgetary constraint is important and the models in this paper can be used as building blocks for more pragmatic models. We plan to continue our research effort in this direction, including the analysis of the sensitivity of the results to the assumptions.

REFERENCES


AUTHORS

Dr. Oded Berman; Faculty of Management,; Univ. of Toronto; Toronto, Ontario M5S 3E6 CANADA.

Internet (e-mail): berman@fmgmt.mgmt.utoronto.ca

Oded Berman is a Full Professor and Associate Dean of Programs at the Faculty of Management at the Univ. of Toronto. He received his PhD (1978) in Operations Research from the Massachusetts Institute of Technology. He has been with the Electronic Systems Lab at MIT, the Univ. of Calgary, and the Univ. of Massachusetts at Boston, where he was also the Chair’n of the Dept. of Management Sciences. He has published over 80 articles and has contributed to several books in his field. His main research interests include operations management in the service industry, location theory, network models, and software reliability. He is the area editor of public services and military Operations Research, a member of the editorial board for Computers and Operations Research, and Associate Editor of Management Science and Transportation Science.

Dr. Michal Cutler; Dept. of Computer Science; State University of New York; Binghamton, New York 13902 USA.

Internet (e-mail): cutler@bingsuny.cc.binghamton.edu

Michal Cutler (M’86) received a BA in Applied Mathematics from Tel-Aviv University, and MA & PhD in Computer Science from the Weizmann Inst. of Science, Rehovot. She is an Associate Professor of Computer Science at Binghamton University. Her interests are fault-tolerant computing, information retrieval, and artificial intelligence. She is a member of the IEEE Computer Society.

Manuscript received 1995 November 4

Publisher Item Identifier S 0018-9529(96)05153-6