Optimal Reliability Allocation

Yashwant K. Malaiya
malaiya@cs.colostate.edu

Department of Computer Science
Colorado State University
Reliability Allocation Problem

- Allocation the reliability values to subsystems
  - to minimize the total cost
  - while achieving the reliability target.

- Widely applicable
  - Software systems
  - Electrical systems
  - Mechanical systems

- Implementation choices
  - Discrete
  - Continuous
Reliability Allocation in Software

- A software system consists of many functional modules
  - Some reused, probably with lower defect densities
  - Some are new, with higher defect densities
  - Some are invoked more often

- To increase reliability
  - Additional testing
  - Replicated using n-version programming?

- What is the best strategy?
Optimal Reliability Allocation

- **System composed of subsystems:**
  - Subsystem cost a function of reliability
  - System reliability depends on subsystems
  - Failure rate as a reliability measure

- **Commons systems:** series and parallel

- **Software system reliability**
  - Fractional execution time
  - Lagrange multiplier: closed form optimal solution
  - Parameter dependence: size, defect density

- **Apportionment & general approach**
Problem Formulation

- System S has subsystems $S_{si}$, $i = 1, \ldots, n$.
- Each subsystem $SS_{i}$ has a specific functionality.
- Several choices with same functionality, but differently reliability levels.

$$C_i = f_i(R_i)$$

- Minimize system cost

$$C_s = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(R_i)$$

- Subject to

  target system reliability $R_{ST}$

  $$\leq$$ achieved reliability $R_s$$
Cost minimization problem

Minimize \[ C_s = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(R_i) \]

Subject to \[ R_{ST} \leq R_s \]

For a series system \[ R_s = \prod_{i=1}^{n} R_i \]

thus \[ R_{ST} \leq \prod_{i=1}^{n} R_i \]
Subsystem implementation choices

- Subsystem can be made more reliable by extending a continuous attribute
  - diameter of a column in building
  - time spent for software testing.
- Different vendors implementations of SSi at different costs.
- Multiple copies of SSi to achieve higher reliability.
  - double wheels of a truck
- Number of copies is constrained between one and a practical number because of implementation issues.
The Cost function

Cost function $f_i$ should satisfy these three conditions:

- $f_i$ is a positive function
- $f_i$ is non-decreasing, thus higher reliability will come at a higher cost.
- $f_i$ increases at a higher rate for higher values of $R_i$

Mettas A, Reliability allocation and optimization for complex systems. Pro Ann Reliability and Maintainability Symp, January 2000, 216-221
In terms of failure rate

- Taking log of both sides, and since $R_i(t) = e^{-\lambda_i t}$

$$\ln(R_{ST}) \leq \sum_{i=1}^{n} \ln(R_i) \quad \lambda_{ST} \geq \sum_{i=1}^{n} \lambda_i$$

- Stating cost as a function of failure rate

$$C_S = \sum_{i=1}^{n} C_i = \sum_{i=1}^{n} f_i(\lambda_i)$$
In terms of failure rate: SRGM

- exponential software reliability growth model

\[ \lambda_i(d) = \lambda_{0i} \exp(-\beta_i d) \]

- \( \lambda_{0i} \) depends on initial defect density
- \( \beta_i \) depends inversely on program size

- Restating it as Cost function

\[ d(\lambda_i) = \frac{1}{\beta_i} \ln\left( \frac{\lambda_{0i}}{\lambda_i} \right) \]

Assumes constant development cost, thus neglected
Series and Parallel Systems: linearization

- Constraint Linearization simplifies the calculations.
- Series system
  \[ \ln( R_{ST} ) \leq \sum_{i=1}^{n} \ln( R_i ) \]
- Parallel system: log of \textit{unreliabilities}

\[ R_{ST} \leq 1 - \prod_{i=1}^{n} (1 - R_i) \quad \ln(1 - R_{ST}) \geq \sum_{i=1}^{n} (\ln (1 - R_i)) \]

- Elegbede: If cost function satisfies 3 properties given above, the cost is optimal if all parallel components have the same cost.
Reliability Allocation for Software Systems

- A block $i$ is under execution for a fraction $x_i$ of the time where $\Sigma x_i = 1$

- Reliability allocation problem

Minimize $C = \sum_{i=1}^{n} \frac{1}{\beta_i} \ln \left( \frac{\lambda_{0i}}{\lambda_i} \right)$

Subject to $\lambda_{ST} \geq \sum_{i=1}^{n} x_i \lambda_i$
Solution using Lagrange multiplier

- solutions for the optimal failure rates

\[ \lambda_1 = \frac{x_1}{\sum_{i=1}^{n} \beta_i} \]
\[ \lambda_2 = \frac{\beta_1 x_1}{\beta_2 x_2} \lambda_1 \quad \wedge \quad \lambda_n = \frac{\beta_1 x_1}{\beta_n x_n} \lambda_1 \]

- optimal values of test times \( d_1 \) and \( d_i \), \( i \neq 1 \)

\[ d_1 = \frac{1}{\beta_1} \ln \left( \frac{\lambda_{10} x_1 \sum_{i=1}^{n} \frac{\beta_1}{\beta_i}}{\lambda_{ST}} \right) \]
\[ d_i = \frac{1}{\beta_i} \ln \left( \frac{\lambda_{i0} \beta_i x_i}{\lambda_1 \beta_1 x_1} \right) \]
Observations: Software reliability allocation

- A reused subsystem have a higher reliability because of past testing causing $\lambda_i \geq \lambda_{i0}$ and hence negative $d_i$.
  - Solution: apply allocation problem only to modules with positive $d_i$.

- If $x_i$ is proportional to the subsystem code size, then optimal values of the post-test failure rates $\lambda_1, \ldots, \lambda_n$ are equal.
An Illustration (next)

- Five blocks software blocks \( I = 1 \) to 5.
- Parameters \( \beta \) and \( \lambda_{i0} \) values are based on what we know about the relationship between parameters and software size, defect density.
- \( X_i \) is presumed to be proportional to software size. \( d_i \) is the additional testing time.
- Analysis using Excel Solver obtains the optimal solution: note that final \( \lambda_i \) is same for all blocks.
  - Closed form solution will yield the same result.
  - Equal testing or testing only the block with most defects will not be optimal.
### Ex: Optimal: Software with 5 blocks

<table>
<thead>
<tr>
<th>Block</th>
<th>B₁</th>
<th>B₂</th>
<th>B₃</th>
<th>B₄</th>
<th>B₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size KSLOC</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Ini Defect density</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$\beta_i$</td>
<td>$4.59 \times 10^{-3}$</td>
<td>$2.30 \times 10^{-3}$</td>
<td>$1.53 \times 10^{-3}$</td>
<td>$4.59 \times 10^{-4}$</td>
<td>$2.30 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\lambda_{i0}$</td>
<td>0.046</td>
<td>0.046</td>
<td>0.046</td>
<td>0.069</td>
<td>0.092</td>
</tr>
<tr>
<td>$x_i$</td>
<td>0.028</td>
<td>0.056</td>
<td>0.083</td>
<td>0.278</td>
<td>0.556</td>
</tr>
<tr>
<td>Optimal $\lambda_i$</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>Optimal $d_i$</td>
<td>30.1</td>
<td>60.1</td>
<td>90.2</td>
<td>1184</td>
<td>3620</td>
</tr>
</tbody>
</table>

$\lambda_{ST} \leq 0.04$

- **Optimal when all modules have the same failure rate!**
Ex: Equal testing

If Total test time is equally distributed for all 5 blocks, system will have significantly higher failure rate of 0.055 per unit time

<table>
<thead>
<tr>
<th>Block</th>
<th>B₁</th>
<th>B₂</th>
<th>B₃</th>
<th>B₄</th>
<th>B₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size KSLOC</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Ini Defect density</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$\beta_i$</td>
<td>$4.59 \times 10^{-3}$</td>
<td>$2.30 \times 10^{-3}$</td>
<td>$1.53 \times 10^{-3}$</td>
<td>$4.59 \times 10^{-4}$</td>
<td>$2.30 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\lambda_{i0}$</td>
<td>0.046</td>
<td>0.046</td>
<td>0.046</td>
<td>0.069</td>
<td>0.092</td>
</tr>
<tr>
<td>$x_i$</td>
<td>0.028</td>
<td>0.056</td>
<td>0.083</td>
<td>0.278</td>
<td>0.556</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>0.146</td>
<td>0.003</td>
<td>0.01</td>
<td>0.08</td>
<td>0.15</td>
</tr>
<tr>
<td>Equal $d_i$</td>
<td>1109.4</td>
<td>1109.4</td>
<td>1109.4</td>
<td>1109.4</td>
<td>1109.4</td>
</tr>
</tbody>
</table>

$\lambda_{ST} \leq 0.04$
Ex: Testing only B5

<table>
<thead>
<tr>
<th>Block</th>
<th>B₁</th>
<th>B₂</th>
<th>B₃</th>
<th>B₄</th>
<th>B₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size KSLOC</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Ini Defect density</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
<tr>
<td>$\beta_i$</td>
<td>$4.59 \times 10^{-3}$</td>
<td>$2.30 \times 10^{-3}$</td>
<td>$1.53 \times 10^{-3}$</td>
<td>$4.59 \times 10^{-4}$</td>
<td>$2.30 \times 10^{-4}$</td>
</tr>
<tr>
<td>$\lambda_{i0}$</td>
<td>0.046</td>
<td>0.046</td>
<td>0.046</td>
<td>0.069</td>
<td>0.092</td>
</tr>
<tr>
<td>$x_i$</td>
<td>0.028</td>
<td>0.056</td>
<td>0.083</td>
<td>0.278</td>
<td>0.556</td>
</tr>
<tr>
<td>$\lambda_i$</td>
<td>0.146</td>
<td>0.003</td>
<td>0.01</td>
<td>0.08</td>
<td>0.15</td>
</tr>
<tr>
<td>Equal $d_i$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5547</td>
</tr>
</tbody>
</table>

If Total test time is allowed only for block B5, system will have higher failure rate of **0.043** per unit time
Illustration using excel

- See Excel sheet `relallocationexamples.xls`
- Try changing entries.
Common Apportionment rules

- Equal reliability apportionment:
  - At end they all individually have failure rate equal to target failure rate for the system

- Complexity based apportionment
  - Test time apportioned in proportion to the software size

- Impact based apportionment:
  - A component executed more frequently, or more critical, should be assigned more resources
Reliability Allocation for Complex Systems

An iterative approach

- Design the system using functional subsystems.
- Perform an initial apportionment of cost or reliability attributes based on suitable apportionment rules or preliminary computation.
- Predict system reliability.
- Is reallocation feasible and will enhance the objective function. If so, perform reallocation.
- Repeat until optimality is achieved.
- Does this meets objectives? If not, return to step 1 and revising the design at a higher level.
Conclusions

- Reliability allocation: consider how cost varies with reliability.
- Software testing:
  - cost $\propto \log(1/\text{failure rate})$
  - $\beta_1 \propto \text{size}$
- Reliability allocation in systems with replicated subsystems can encounter correlated failures and thus would need a more careful modeling.