8. Universal Generating Function in Analysis and Optimization of Fault-tolerant Software

8.1 Reliability and Performance of Fault-tolerant Software

The NVP approach presumes the execution of n functionally equivalent software versions that receive the same input and send their outputs to a voter, which is aimed at determining the system's output. The voter produces an output if at least k out of n outputs agree. Otherwise, the system fails. As shown in Section 3.2.10, the RBS approach can also be considered as NVP with $k = 1$ when the system performance (task execution time) is considered.

In many cases, the information about the version's reliability and the execution time are available from separate testing and/or reliability prediction models [196]. This information can be incorporated into a fault-tolerant program model in order to obtain an evaluation of its reliability and performance.

8.1.1 Fault-tolerant Software Performance Model

According to the generally accepted model [197], the software system consists of C components. Each component performs a subtask and the sequential execution of the components performs a major task.

It is assumed that n_c functionally equivalent versions are available for each component c. Each version i has an estimated reliability r_{ci} and constant execution time τ_{ci}. Failures of versions for each component are statistically independent, as well as the total failures of the different components.

The software versions in each component c run on parallel hardware units. The total number of units is h_c. The units are independent and identical. The availability of each unit is a_c. The number H_c of units available at the moment determines the amount of available computational resources and, therefore, the number of versions that can be executed simultaneously $L_c(H_c)$. No hardware unit can change its state during the software execution.

The versions of each component c start their execution in accordance with a predetermined ordered list. L_c first versions from the list start their execution simultaneously (at time zero). If the number of terminated versions is less than k_c,

after termination of each version a new version from the list starts its execution immediately. If the number of terminated versions is not less than k_c, after termination of each version the voter compares the outputs. If k_c outputs are identical, the component terminates its execution (terminating all the versions that are still executed), otherwise a new version from the list is executed immediately.

If after termination of n_c versions the number of identical outputs is less than k_c the component and the entire system fail.

In the case of component success, the time of the entire component execution T_c is equal to the termination time of the version that has produced the k_cth correct output (in most cases, the time needed by the voter to make the decision can be neglected). It can be seen that the component execution time is a random variable depending on the reliability and the execution time of the component versions and on the availability of the hardware units. We assume that if the component fails, then its execution time is equal to infinity.

The examples of time diagrams (corresponding to component 1 with $n_1 = 5$, $k_1 = 3$ and component 2 with $n_2 = 3$, $k_2 = 2$) for a given sequence of versions execution (the versions are numbered according to this sequence) and different values of L_c are presented in Figure 8.1.

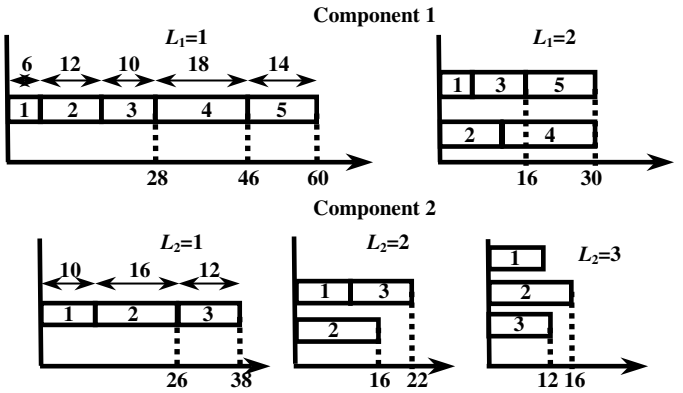


Figure 8.1. Time diagrams for software components with different numbers of versions executed simultaneously

The sum of the random execution times of each component gives the random task execution time for the entire system T. In order to estimate both the system's reliability and its performance, different measures can be used, depending on the application.

In applications where the execution time of each task is of critical importance, the system's acceptability function is defined (according to the performability concept [198, 199]) as $F(T, w) = 1(T<w)$, where w is a maximal allowed system execution time. The system's reliability $R(w) = E(F(T, w))$ in this case is the probability that the correct output is produced in time less than w. The conditional expected system execution time $\tilde{\varepsilon}(w) = E(T \times 1(T < w)) / R(w)$ is considered to be a measure of the system's performance. This index, defined according to Equation

(3.7), determines the system's expected execution time given that the system does not fail.

In applications where the system's average productivity (the number of executed tasks) over a fixed mission time is of interest [200], the system's acceptability function is defined as $F(T) = 1(T<\infty)$, the system's reliability is defined as the probability that it produces correct outputs regardless of the total execution time (this index can be referred to as $R(\infty)$), and the conditional expected system execution time $\widetilde{\varepsilon}(\infty)$ is considered to be a measure of the system's performance.

8.1.1.1 Number of Versions that Can Be Simultaneously Executed

The number of available hardware units in component c can vary from 0 to h_c. Given that all of the units are identical and have availability a_c, one can easily obtain probabilities $Q_c(x) = \Pr\{H_c = x\}$ for $0 \le x \le h_c$:

$$Q_c(x) = \Pr\{H_c = x\} = \binom{h_c}{x} a_c{}^x (1 - a_c)^{h_c - x} \tag{8.1}$$

The number of available hardware units x determines the number of versions that can be executed simultaneously: $l_c(x)$. Therefore:

$$\Pr\{L_c = l_c(x)\} = Q_c(x) \tag{8.2}$$

The pairs $Q_c(x)$, $l_c(x)$ for $1 \le x \le h_c$ determine the p.m.f. of the discrete random value L_c.

8.1.1.2 Version Termination Times

In each component c, a sequence where each version starts its execution is defined by the numbers of versions. This means that each version i starts its execution not earlier than versions 1, …, i–1 and not later than versions i+1, …, n_c. If the number of versions that can run simultaneously is l_c, then we can assume that the software versions run on l_c independent processors. Let α_m be the time when processor m terminates the execution of a version and is ready to run the next version from the list of not executed versions. Having the execution time of each version τ_{ci} $(1 \le i \le n_c)$, one can obtain the termination time $t_{ci}(l_c)$ for each version i using the following simple algorithm:

1. Assign $\alpha_1 = \ldots = \alpha_{l_c} = 0$ (all of the processors are ready to run the software versions at time 0).
2. For $i = 1, \ldots, n_c$ repeat:

   2.1. Find any m $(1 \le m \le l_c)$: $\alpha_m = \min\{\alpha_1, \ldots, \alpha_{l_c}\}$ (m is the number of the earliest processor that is ready to run a new version from the list).

   2.2. Obtain $t_{ci}(l_c) = \alpha_m + \tau_{ci}$ and assign $\alpha_m = t_{ci}(l_c)$.

Times $t_{ci}(l_c)$, $1 \le i \le n_c$, correspond to intervals between the beginning of component execution and the moment when the versions produce their outputs. Observe that the versions that start execution earlier can terminate later: $j<y$ does not guarantee that $t_{cj}(l_c) \le t_{cy}(l_c)$. In order to obtain the sequence, in which the versions produce their outputs, the termination times should be sorted in increasing order $t_{cm_1}(l_c) \le t_{cm_2}(l_c) \le \ldots \le t_{cm_{n_c}}(l_c)$, which gives the order of versions $m_1, m_2, \ldots, m_{n_c}$, corresponding to times of their termination.

The ordered list $m_1, m_2, \ldots, m_{n_c}$ determines the sequence of version outputs in which they arrive at the voter. Now one can consider the component c as a system in which the n_c versions are executed consecutively according to the order $m_1, m_2, \ldots, m_{n_c}$ and produce their outputs at times $t_{cm_1}(l_c)$, $t_{cm_2}(l_c)$, …, $t_{cm_{n_c}}(l_c)$.

8.1.1.3 The Reliability and Performance of Components and the Entire System

Let r_{cm_i} be the reliability of the version that produces ith output in component c (r_{cm_i} is equal to the probability that this output is correct). Consider the probability that k out of n first versions of component c succeed. This probability can be obtained as

$$\left[\prod_{i=1}^{n}(1-r_{cm_i})\right]\left[\sum_{i_1=1}^{n-k+1}\frac{r_{cm_{i_1}}}{(1-r_{cm_{i_1}})}\sum_{i_2=i_1+1}^{n-k+2}\frac{r_{cm_{i_2}}}{(1-r_{cm_{i_2}})}\ldots\sum_{i_k=i_{k-1}+1}^{n}\frac{r_{cm_{i_k}}}{(1-r_{cm_{i_k}})}\right] \quad (8.3)$$

(according to Equation (2.16) for k-out-of-n systems). The component c produces the correct output directly after the end of the execution of j versions ($j \ge k_c$) if the m_jth version succeeds and exactly k_c-1 out of the first executed $j-1$ versions succeed.

The probability of such event $p_{cj}(l_c)$ is

$$p_{cj}(l_c) = r_{cm_j}\left[\prod_{i=1}^{j-1}(1-r_{cm_i})\right]\left[\sum_{i_1=1}^{j-k_c+1}\frac{r_{cm_{i_1}}}{(1-r_{cm_{i_1}})}\sum_{i_2=i_1+1}^{j-k_c+2}\frac{r_{cm_{i_2}}}{(1-r_{cm_{i_2}})}\right.$$
$$\left.\ldots\sum_{i_{k_c-1}=i_{k_c-2}+1}^{j-1}\frac{r_{cm_{i_{k_c-1}}}}{(1-r_{cm_{i_{k_c-1}}})}\right] \quad (8.4)$$

Observe that $p_{cj}(l_c)$ is the conditional probability that the component execution time is $t_{cm_j}(l_c)$ given l_c versions can be executed simultaneously:

$$p_{cj}(l_c) = \Pr\{T_c = t_{cm_j}(l_c) \mid L_c = l_c\} \quad (8.5)$$

Having the p.m.f. of L_c we can now obtain for $1 \le x \le h_c$

$$\Pr\{T_c = t_{cm_j}(l_c(x))\} = \Pr\{T_c = t_{ck_j}(l_c(x)) \mid L_c = l_c(x)\}\Pr\{L_c = l_c(x)\}$$
$$= p_{cj}(l_c(x))Q_c(x) \tag{8.6}$$

The pairs $t_{cm_j}(l_c(x))$, $p_{cj}(l_c(x))Q_c(x)$, obtained for $1 \le x \le h_c$ and $k_c \le j \le n_c$, determine the p.m.f. of version execution time T_c.

Since the events of successful component execution termination for different j and x are mutually exclusive, we can express the probability of component c success as

$$R_c(\infty) = \Pr\{T_c < \infty\} = \sum_{x=1}^{h_c} [Q_c(x) \sum_{j=k_c}^{n_c} p_{cj}(l_c(x))] \tag{8.7}$$

Since failure of any component constitutes the failure of the entire system, the system's reliability can be expressed as

$$R(\infty) = \prod_{c=1}^{C} R_c(\infty) \tag{8.8}$$

From the p.m.f. of execution times T_c for each component c one can obtain the p.m.f. of the execution time of the entire system, which is equal to the sum of the execution times of components:

$$T = \sum_{c=1}^{C} T_c \tag{8.9}$$

8.1.1.4 Using Universal Generating Function for Evaluating the Execution Time Distribution of Components

In order to obtain the execution time distribution for a component c for a given l_c in the form $p_{cj}(l_c)$, $t_{cm_j}(l_c)$ $(k_c \le j \le n_c)$ one can determine the realizations $t_{cm_j}(l_c)$ of the execution time $T_c(l_c)$ using the algorithm presented in Section 8.1.1.2 and the corresponding probabilities $p_{cj}(l_c)$ using Equation (8.4). However, the probabilities $p_{cj}(l_c)$ can be obtained in a much simpler way using a procedure based on the UGF technique [201].

Let the random binary variable s_{cm_i} be an indicator of the success of version m_i in component c such that $s_{cm_i} = 1$ if the version produces the correct output and $s_{cm_i} = 0$ if it produces the wrong output. The p.m.f. of s_{cm_i} can be represented by the u-function

$$u_{cm_i}(z) = r_{cm_i} z^1 + (1 - r_{cm_i}) z^0 \tag{8.10}$$

It can be easily seen that using the operator $\underset{+}{\otimes}$ we can obtain the *u*-function

$$U_{cj}(z,l_c)=\underset{+}{\otimes}(u_{cm_1}(z),...,u_{cm_j}(z)) \tag{8.11}$$

that represents the p.m.f. of the number of correct outputs in component *c* after the execution of a group of first *j* versions (the order of elements m_1, m_2, …, m_{n_c} and, therefore, $U_{cj}(z, l_c)$ depend on l_c). Indeed, the resulting polynomial relates the probabilities of combinations of correct and wrong outputs (the product of corresponding probabilities) with the number of correct outputs in these combinations (the sum of success indicators). Observe that after collecting the like terms (corresponding to obtaining the overall probability of a different combination with the same number of correct outputs) $U_{cj}(z, l_c)$ takes the form

$$U_{cj}(z,l_c)=\sum_{k=0}^{j}\pi_{jk}z^k \tag{8.12}$$

where π_{jk} is the probability that the group of first *j* versions produces *k* correct outputs.

Note that $U_{cj}(z, l_c)$ can be obtained by using the recurrent expression

$$U_{cj}(z,l_c)=U_{cj-1}(z,l_c)\underset{+}{\otimes}[r_{cm_j}z^1+(1-r_{cm_j})z^0] \tag{8.13}$$

According to its definition, $p_{cj}(l_c)$ is the probability that the group of first *j* versions produces k_c correct outputs and the group of first $j-1$ versions produces $k_c - 1$ correct outputs given that l_c versions can be executed simultaneously. The coefficient π_{jk_c} in polynomial $U_{cj}(z, l_c)$ is equal to the conditional probability that the group of first *j* versions produces k_c correct outputs given that l_c versions can be executed simultaneously.

In order to let the coefficient π_{jk_c} in polynomial $U_{cj}(z, l_c)$ be equal to $p_{cj}(l_c)$, the term with the exponent equal to k_c should be removed from $U_{c\,j-1}(z, l_c)$ before applying Equation (8.13) (excluding the combination in which $j-1$ first versions produce k_c correct outputs while the m_jth version fails).

If after the execution of *j* first versions the number of correct outputs produced is *k* and $k+n_c-j<k_c$, then the required number of correct outputs k_c cannot be obtained even if all the n_c-j subsequent versions produce correct outputs. Therefore, the terms $\pi_{jk}z^k$ with $k<k_c-n_c+j$ can be removed from $U_{cj}(z, l_c)$.

The above considerations lie at the base of the following algorithm for determining all of the probabilities $p_{cj}(l_c)$ ($k_c \le j \le n_c$):

1. For the given l_c, determine the order of version termination m_1, m_2, …, m_{n_c} using the algorithm from Section 8.1.1.2.

2. Determine the *u*-function of each version of component c according to Equation (8.10).

3. Define $U_{c0}(z, l_c) = 1$. For $j = 1, 2, \ldots, n_c$:

3.1 Obtain $U_{cj}(z, l_c)$ using Equation (8.13) and, after collecting like terms, represent it in the form (8.12).

3.2. Remove from $U_{cj}(z, l_c)$ all the terms $\pi_{jk} z^k$ for which $k < k_c - n_c + j$.

3.3. If $j \geq k_c$, assign: $p_{cj}(l_c) = \pi_{jk_c}$ and remove term $\pi_{jk_c} z^{k_c}$ from $U_{cj}(z, l_c)$.

8.1.1.5 Execution Time Distribution for the Entire System

Having the pairs $p_{cj}(l_c(x))$, $t_{cm_j}(l_c(x))$ for each possible realization $l_c(x)$ of L_c ($1 \leq x \leq h_c$) and probabilities $\Pr\{L_c = l_c(x)\} = Q_c(x)$, one can obtain the p.m.f. of random execution times T_c for each component by applying Equation (8.6). If the conditional p.m.f. $p_{cj}(l_c(x))$, $t_{cm_j}(l_c(x))$ are represented by the *u*-function

$$\tilde{u}_c(z, l_c(x)) = \sum_{j=k_c}^{n_c} p_{cj}(l_c(x)) z^{t_{cm_j}(l_c(x))} \tag{8.14}$$

then the *u*-function representing the p.m.f. of the random value T_c takes the form:

$$\tilde{U}_c(z) = \sum_{x=1}^{h_c} Q_c(x) \tilde{u}_c(z, l_c(x)) \tag{8.15}$$

Since the random system execution time T is equal to the sum of the execution times of all of the C components, one can obtain the *u*-function $\tilde{U}(z)$ representing the p.m.f. of T as

$$\tilde{U}(z) = \underset{+}{\otimes}(\tilde{U}_1(z), \ldots, \tilde{U}_C(z)) = \prod_{c=1}^{C} \left(\sum_{x=1}^{h_c} Q_c(x) \tilde{u}_c(z, l_c(x)) \right) \tag{8.16}$$

8.1.1.6 Different Components Executed on the Same Hardware

Now consider the case where all of the software components are consecutively executed on the same hardware consisting of h parallel identical modules with the availability a. The number of available parallel hardware modules H is random with p.m.f. $Q(x) = \Pr\{H = x\}$, $1 \leq x \leq h$, defined in the same way as in Equation (8.1).

When $H = x$, the number of versions that can be executed simultaneously in each component c is $l_c(x)$. The *u*-functions representing the p.m.f. of the corresponding component execution times T_c are $\tilde{u}_c(z, l_c(x))$ defined by Equation (8.14). The *u*-function $\hat{U}(z, x)$ representing the conditional p.m.f. of the system execution time T (given the number of available hardware modules is x) can be obtained for any x ($1 \leq x \leq h$) as

$$\hat{U}(z,x) = \underset{+}{\otimes}(\tilde{u}_1(z,l_1(x)),...,\tilde{u}_C(z,l_C(x))) = \prod_{c=1}^{C} \tilde{u}_c(z,l_c(x)) \tag{8.17}$$

Having the p.m.f. of the random value H we obtain the u-function $\tilde{U}(z)$ representing the p.m.f. of T as:

$$\tilde{U}(z) = \sum_{x=1}^{H} Q(x)\hat{U}(z,x) \tag{8.18}$$

Example 8.1

Consider a system consisting of two components. The first component consists of $h_1 = 2$ hardware units with availability $a_1 = 0.9$ on which $n_1 = 5$ software versions with $k_1 = 3$ are executed. The second component consists of $h_2 = 3$ hardware units with availability $a_2 = 0.8$ on which and $n_2 = 3$ software versions with $k_2 = 2$ are executed. The parameters of versions r_{ci} and τ_{ci} are presented in Table 8.1.

Table 8.1. Parameters of software versions

Componen	$c = 1$					$c = 2$		
Version	1	2	3	4	5	1	2	3
r_{ci}	0.7	0.6	0.8	0.6	0.9	0.8	0.8	0.7
τ_{ci}	6	12	10	18	14	10	16	12
$t_{ci}(1)$	6	18	28	46	60	10	26	38
$t_{ci}(2)$	6	12	16	30	30	10	16	22
$t_{ci}(3)$	-	-	-	-	-	10	16	12

One software version can be executed on each hardware unit: $l_c(h_c) = h_c$.

The terminations times $t_{ci}(l_c)$ obtained for the different possible values of L_1 and L_2 using the algorithm described in Section 8.1.1.2 are also presented in this table. The version execution diagrams for different values of L_1 and L_2 are presented in Figure 8.1.

For the given parameters, the u-functions of the software versions are

$$u_{11}(z) = 0.3z^0+0.7z^1;\ u_{12}(z) = 0.4z^0+0.6z^1;\ u_{13}(z) = 0.2z^0+0.8z^1$$

$$u_{14}(z) = 0.4z^0+0.6z^1;\ u_{15}(z) = 0.1z^0+0.9z^1$$

for component 1 and

$$u_{21}(z) = 0.2z^0+0.8z^1;\ u_{22}(z) = 0.2z^0+0.8z^1;\ u_{23}(z) = 0.3z^0+0.7z^1$$

for component 2.

The order of version termination in the first component is 1, 2, 3, 4, 5 for both $L_1 = 1$ and $L_1 = 2$ (see Table 8.1).

According to the algorithm presented in Section 8.1.1.4, determine the u-functions for the groups of versions and corresponding probabilities $p_{1j}(1)$ and $p_{1j}(2)$.

$$U_{10}(z, 1) = U_{10}(z, 2) = 1;\ U_{11}(z, 1) = U_{11}(z, 2) = u_{11}(z) = 0.3z^0+0.7z^1$$

$$U_{12}(z, 1) = U_{12}(z, 2) = u_{11}(z) \underset{+}{\otimes} u_{12}(z) = (0.3z^0+0.7z^1)(0.4z^0+0.6z^1)$$

$$= 0.12z^0+0.46z^1+0.42z^2$$

$$U_{13}(z, 1) = U_{13}(z, 2) = U_{12}(z, 1) \underset{+}{\otimes} u_{13}(z) = (0.12z^0+0.46z^1+0.42z^2)$$

$$\times(0.2z^0+0.8z^1) = 0.024z^0+0.188z^1+0.452z^2+0.336z^3$$

Remove the term $0.024z^0$ from $U_{13}(z, 1)$ according to step 3.2 of the algorithm. Remove the term $0.336z^3$ from $U_{13}(z, 1)$ and $U_{13}(z, 2)$ and obtain $p_{13}(1) = p_{13}(2) = 0.336$ according to step 3.3 of the algorithm:

$$U_{14}(z, 1) = U_{14}(z, 2) = U_{13}(z, 1) \underset{+}{\otimes} u_{14}(z) = (0.188z^1+0.452z^2)$$

$$\times(0.4z^0+0.6z^1) = 0.0752z^1+0.2936\ z^2+0.2712z^3$$

Remove the term $0.0752z^1$ from $U_{14}(z, 1)$ according to step 3.2 of the algorithm. Remove the term $0.2712z^3$ from $U_{14}(z, 1)$ and $U_{14}(z, 2)$ and obtain $p_{14}(1) = p_{14}(2) = 0.2712$ according to step 3.3 of the algorithm:

$$U_{15}(z, 1) = U_{15}(z, 2) = U_{14}(z, 1) \underset{+}{\otimes} u_{15}(z)$$

$$= 0.2936z^2\ (0.1z^0+0.9z^1) = 0.02936z^2+0.11z^2+0.26424z^3$$

Finally, obtain $p_{15}(1) = p_{15}(2) = 0.26424$.

Having the probabilities $p_{1j}(1)$, $p_{1j}(2)$ and the corresponding termination times $t_{1j}(1)$, $t_{1j}(2)$, define the u-functions representing the component execution time distributions

$$\tilde{u}_1(z, 1) = 0.336z^{28}+0.271z^{46}+0.264z^{60}$$

$$\tilde{u}_1(z, 2) = 0.336z^{16}+0.271z^{30}+0.264z^{30} = 0.336z^{16}+0.535z^{30}$$

The p.m.f. of L_1 is

$$Q_1(1) = \Pr\{L_1 = 1\} = \Pr\{h_1 = 1\} = 2a_1(1-a_1) = 0.18$$

$$Q_1(2) = \Pr\{L_1 = 2\} = \Pr\{h_1 = 2\} = a_1^2 = 0.81$$

According to Equation (8.15), obtain

$$\widetilde{U}_1(z) = 0.18\,\widetilde{u}_1(z, 1)+0.81\,\widetilde{u}_1(z, 2)$$

$$= 0.06z^{28}+0.049z^{46}+0.048z^{60}+0.272z^{16}+0.434z^{30}$$

The order of version termination in the second component is 1, 2, 3 for both $L_2 = 1$ and $L_2 = 2$ and 1, 3, 2 for $L_2 = 3$ (see Table 8.1).

According to the algorithm presented in Section 8.1.1.4, determine the u-functions for the groups of versions and corresponding probabilities $p_{2j}(1)$, $p_{2j}(2)$ and $p_{2j}(3)$. For $L_2 = 1$ and $L_2 = 2$:

$$U_{20}(z, 1) = U_{20}(z, 2) = 1;\ U_{21}(z, 1) = U_{21}(z, 2) = u_{21}(z) = 0.2z^0+0.8z^1$$

$$U_{22}(z, 1) = U_{22}(z, 2) = u_{21}(z) \underset{+}{\otimes} u_{22}(z)$$

$$= (0.2z^0+0.8z^1)^2 = 0.04z^0+0.32z^1+0.64z^2$$

Remove the term $0.04z^0$ from $U_{22}(z, 1)$ according to step 3.2 of the algorithm. Remove the term $0.64z^2$ from $U_{22}(z, 1)$ and $U_{22}(z, 2)$ and obtain $p_{22}(1) = p_{22}(2) = 0.64$ according to step 3.3 of the algorithm:

$$U_{23}(z, 1) = U_{23}(z, 2) = U_{22}(z, 1) \underset{+}{\otimes} u_{23}(z) = 0.32z^1\ (0.3z^0+0.7z^1)$$

$$= 0.096z^1+0.224z^2$$

Obtain $p_{23}(1) = p_{23}(2) = 0.224$.

Having the probabilities $p_{2j}(1)$, $p_{2j}(2)$ and the corresponding termination times $t_{2j}(1)$, $t_{2j}(2)$, define the u-functions representing the component execution time distributions

$$\widetilde{u}_2(z, 1) = 0.64z^{26}+0.224z^{38},\ \widetilde{u}_2(z, 2) = 0.64z^{16}+0.224z^{22}$$

For $L_2 = 3$:

$$U_{20}(z, 3) = 1;\ U_{21}(z, 3) = u_{21}(z) = 0.2z^0+0.8z^1;\ U_{22}(z, 3) = u_{21}(z) \underset{+}{\otimes} u_{23}(z)$$

$$= (0.2z^0+0.8z^1)(0.3z^0+0.7z^1) = 0.06z^0+0.38z^1+0.56z^2$$

Remove the term $0.06z^0$ from $U_{22}(z, 1)$ according to step 3.2 of the algorithm. Remove the term $0.56z^2$ from $U_{22}(z)$ and obtain $p_{22}(3) = 0.56$ according to step 3.3 of the algorithm:

$$U_{23}(z, 3) = U_{22}(z, 3) \underset{+}{\otimes} u_{22}(z) = 0.38z^1(0.2z^0+0.8z^1)$$

$$= 0.076z^1+0.304z^2$$

From $U_{23}(z, 3)$ obtain $p_{23}(3) = 0.304$.

The u-function representing the corresponding component execution time distribution takes the form

$$\tilde{u}_2(z, 3)=0.56z^{12}+0.304z^{16}$$

The p.m.f. of L_2 is

$$Q_2(1) = \Pr\{L_2 = 1\} = \Pr\{h_2 = 1\} = 3a_2(1-a_2)^2 = 0.096$$

$$Q_2(2) = \Pr\{L_2=2\} = \Pr\{h_2 = 2\} = 3a_2^2(1-a_2) = 0.384$$

$$Q_2(3) = \Pr\{L_2 = 3\} = \Pr\{h_2 = 3\} = a_2^3 = 0.512$$

According to Equation (8.15), obtain

$$\tilde{U}_2(z) = 0.096\,\tilde{u}_2(z, 1)+0.384\,\tilde{u}_2(z, 2)+0.512\,\tilde{u}_2(z, 3)$$

$$= 0.287z^{12}+0.401z^{16}+0.086z^{22}+0.0614z^{26}+0.0215z^{38}$$

The u-function representing the execution time distribution for the entire system takes the form

$$\tilde{U}(z) = \tilde{U}_1(z) \underset{+}{\otimes} \tilde{U}_2(z) = (0.272z^{16}+0.434z^{30}+0.06z^{28}+0.049z^{46}$$

$$+0.048z^{60})(0.287z^{12}+0.401z^{16}+0.086z^{22}+0.0614z^{26}+0.0215z^{38})$$

$$= 0.078z^{28}+0.109z^{32}+0.023z^{38}+0.017z^{40}+0.141z^{42}+0.024z^{44}+0.174z^{46}$$

$$+0.005z^{50}+0.037z^{52}+0.007z^{54}+0.027z^{56}+0.014z^{58}+0.020z^{62}+0.001z^{66}$$

$$+0.012z^{68}+0.020z^{72}+0.019z^{76}+0.004z^{82}+0.001z^{84}+0.003z^{86}+0.001z^{98}$$

Having this u-function, one can obtain the system reliability and conditional expected execution time for different time constraints w:

$$R(\infty) = 0.739;\quad \tilde{\varepsilon}(\infty) = \frac{1}{0.739}(32.94)=44.559$$

For $w = 50$:

$$R(50) = 0.078+0.109+0.023+0.017+0.141+0.024+0.174 = 0.567$$

$$\widetilde{\varepsilon}(50) = \frac{1}{0.567}(0.078\times28+0.109\times32+0.023\times38+0.017\times40$$
$$+ 0.141\times42+0.024\times44+0.174\times46) = 22.24/0.567 = 39.24$$

Example 8.2

Consider a system consisting of five components. The number of parallel hardware units, the availability of these units, and the parameters n_c and k_c of the fault-tolerant programs for each component are presented in Table 8.2. Components 1, 3 and 5 are of the NVP type and components 2 and 4 are of the RBS type ($k_2 = k_4 = 1$). The reliability and execution times of the software versions are presented in Table 8.3.

Table 8.2. Parameters of system components

Component	1	2	3	4	5
h_c	3	4	2	3	2
a_c	0.80	0.95	0.90	0.85	0.95
n_c	5	2	3	3	5
k_c	3	1	2	1	3
$l_c(1)$	2	0	1	1	3
$l_c(2)$	4	1	3	2	5
$l_c(3)$	5	1	-	3	-
$l_c(4)$	-	2	-	-	-

Table 8.3. Parameters of software versions

	Version	1	2	3	4	5
$c = 1$	r_{1i}	0.86	0.77	0.98	0.93	0.91
	τ_{1i}	10	12	25	22	15
$c = 2$	r_{2i}	0.85	0.92	-	-	-
	τ_{2i}	30	45	-	-	-
$c = 3$	r_{3i}	0.87	0.94	0.98	-	-
	τ_{3i}	6	6	10	-	-
$c = 4$	r_{4i}	0.95	0.85	0.85	-	-
	τ_{4i}	25	15	20	-	-
$c = 5$	r_{5i}	0.68	0.79	0.90	0.90	0.94
	τ_{5i}	10	10	15	20	25

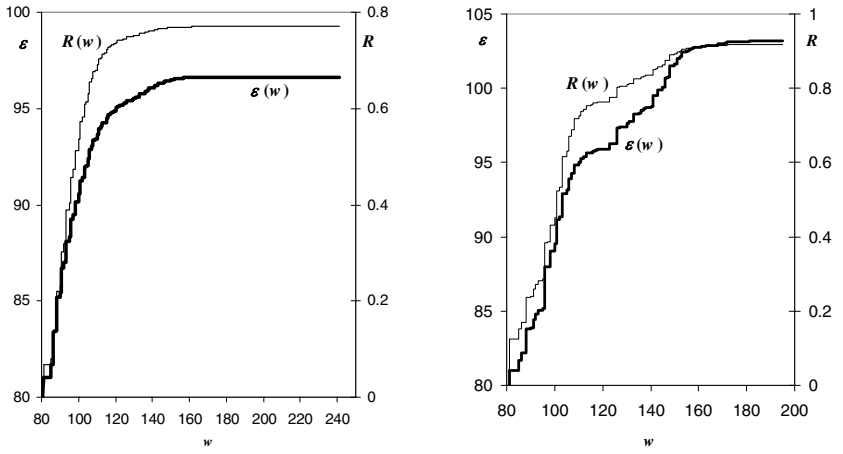
The entire system's execution time varies in the range of $81 \le T \le 241$. The system's reliability and conditional expected execution time obtained by the algorithm described in Sections 8.1.1.1-8.1.1.5 are $R(\infty) = 0.77$ and $\widetilde{\varepsilon}(\infty) = 96.64$ respectively. The functions $R(w)$ and $\widetilde{\varepsilon}(w)$ are presented in Figure 8.2A.

Now consider the same fault-tolerant software system running on a single hardware block consisting of three parallel units with availability $a = 0.9$. The functions $l_c(x)$ for each software component are presented in Table 8.4. Note that the system can operate only when $H \ge 2$, since a single hardware unit has not enough resources for execution of the second software component.

Table 8.4. Number of versions executed simultaneously

Component	1	2	3	4	5
$l_c(1)$	2	0	3	1	1
$l_c(2)$	4	1	3	2	2
$l_c(3)$	5	1	3	3	3

The entire system execution time varies in the range $81 \le T \le 195$. The system reliability and expected execution time (without respect to execution time constraints) are respectively $R(\infty) = 0.917$ and $\tilde{\varepsilon}(\infty) = 103.18$. The functions $R(w)$ and $\tilde{\varepsilon}(w)$ are presented in Figure 8.2B.



A **B**

Figure 8.2. $R(w)$ and $\tilde{\varepsilon}(w)$ functions for a fault-tolerant system running on different hardware components (A) and on a single hardware component (B)

8.2 Optimal Version Sequencing in Fault-tolerant Programs

In programs consisting of versions with different parameters, the sequence of the version execution affects the distribution of the system's task execution time. The influence of the sequence of the version's execution on this distribution is demonstrated in Figure 8.3.

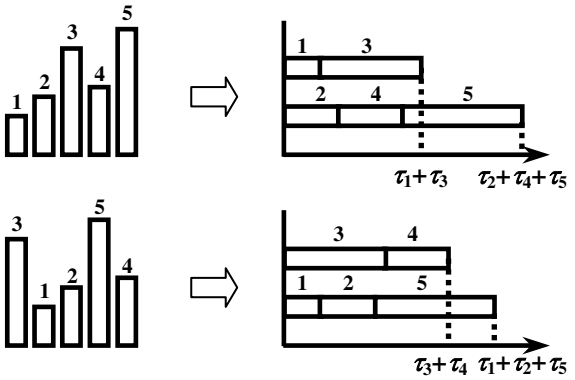


Figure 8.3. Execution of the different sequences of versions in a component with $n_c = 5, k_c = 4, L_c = 2$

While the sequence in which the versions in each component start their execution does not affect the $R(\infty)$ index, it can have a considerable influence on both $R(w)$ and $\widetilde{\varepsilon}(w)$ when the task execution time is constrained. Therefore, two different optimization problems can be formulated in which the sequences maximizing the system's reliability $R(w)$ or minimizing its conditional expected execution time $\widetilde{\varepsilon}(w)$ are to be determined.

To apply the GA, one has to represent the sequences of versions in each component in the form of strings. These sequences can be represented by C substrings corresponding to different components. Each substring c should be of length n_c and contain a permutation of integer numbers ranging from 1 to n_c.

The solution encoding scheme with different substrings is complicated and requires the development of sophisticated procedures for string generation, crossover, and mutation that preserve the feasibility of solutions. In order to simplify the GA procedures, an encoding method was developed in which the single permutation defines the sequences of the versions in each one of the C components.

The solution encoding string consists of $n = \sum_{c=1}^{C} n_c$ different integer numbers ranging from 1 to n. Each number j belonging to the interval

$$\sum_{c=1}^{m-1} n_c + 1 \le j \le \sum_{c=1}^{m} n_c \tag{8.19}$$

corresponds to version $j - \sum_{c=1}^{m-1} n_c$ of component m. The relative order in which the numbers corresponding to the versions of the same component appear in the string determines the sequence of their execution.

Example 8.3

Consider a system consisting of three components with $n_1 = 3$, $n_2 = 5$ and $n_3 = 4$. The solution encoding strings should consist of 3+5+4 = 12 integers. Numbers 1, 2, 3 correspond to component 1, numbers 4, 5, 6, 7, 8 correspond to component 2 (they are marked in bold), numbers 9, 10, 11, 12 correspond to component 3 (they are marked in italic). In the solution string $\boldsymbol{a}$ = (**4**, *9*, **7**, 1, *12*, 2, 3, *10*, **6**, **8**, **5**, *11*) the numbers corresponding to different components appear in the following relative order:

for component 1: 1, 2, 3
for component 2: **4**, **7**, **6**, **8**, **5**
for component 3: *9*, *12*, *10*, *11*

This corresponds to the following sequences of versions execution:

in component 1: 1, 2, 3
in component 2: 1, 4, 3, 5, 2
in component 3: 1, 4, 2, 3

The solution decoding procedure determines the sequence of versions in each component according to string $\boldsymbol{a}$. Then it calculates the version termination times using the procedure presented in Section 8.1.1.2 and the probabilities p_{cj} using the algorithm presented in Section 8.1.1.3. After obtaining the components' execution time distribution, the time distribution of the entire system is calculated as described in Sections 8.1.1.5 and 8.1.1.6. Finally, the indices $R(w)$ and $\widetilde{\varepsilon}(w)$ are obtained from the system's execution time distribution. The solution fitness can be defined as $R(\boldsymbol{a},w)$ or $M-\widetilde{\varepsilon}(\boldsymbol{a},w)$, depending on the optimization problem formulation.

Example 8.4

Consider a fault-tolerant software system consisting of five components and running on a single reliable unit [202]. The parameters of the components are presented in Table 8.5. This table contains the values of n_c and k_c for each component c and the reliability and execution time for each version.

First consider the system in which $L_1 = 1$, $L_2 = 2$, $L_3 = 4$, $L_4 = 1$, $L_5 = 3$. The overall system reliability that does not depend on version sequencing is $R(\infty) = 0.92$. The solutions with minimal conditional expected execution time $\widetilde{\varepsilon}(\infty)$ and with maximal system reliability $R(w)$ for $w = 300$ are presented in Table 8.6. The table contains minimal and maximal possible system execution times for each solution, values of indices $\widetilde{\varepsilon}(\infty)$ and $R(w)$, and the corresponding version sequences.

It can be seen that the minimal possible system execution time $T_{\min}$ (achieved when in each component the first k_c versions succeed) can be obtained when the versions in each component are ordered according to the increased execution time. Such a solution is also presented in Table 8.6. Observe that the solution that

provides the smallest minimal execution time is not optimal, neither in terms of $\widetilde{\varepsilon}(\infty)$ nor in terms of $R(w)$.

Table 8.5. Parameters of system components for the numerical example

No. of component	n_c	k_c		Versions				
				1	2	3	4	5
1	4	1	τ	17	20	32	75	
			r	0.71	0.85	0.89	0.98	-
2	3	2	τ	28	55	58	-	-
			r	0.85	0.85	0.93	-	-
3	5	3	τ	17	20	38	41	63
			r	0.80	0.80	0.86	0.98	0.98
4	3	2	τ	17	20	32	-	-
			r	0.75	0.93	0.97	-	-
5	3	1	τ	30	54	70	-	-
			r	0.70	0.80	0.89	-	-

The poor solution corresponding to the greatest possible $\widetilde{\varepsilon}(\infty)$ is presented in Table 8.6 for comparison. The system's reliabilities $R(w)$ as functions of the maximal allowable execution time w are presented in Figure 8.4A for all of the solutions obtained (numbered according to Table 8.6).

Table 8.6. Parameters of solutions obtained for the fault-tolerant system with $L_1 = 1$, $L_2 = 2$, $L_3 = 4$, $L_4 = 1$, $L_5 = 3$

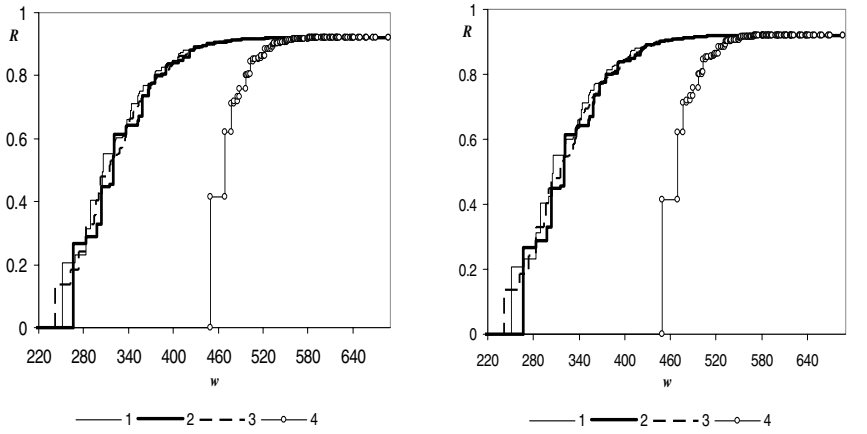
No.	Problem formulation	Sequence of versions	$T_{\min}$	$T_{\max}$	$\widetilde{\varepsilon}(\infty)$	$R(300)$
1	$\widetilde{\varepsilon}(\infty)\rightarrow$min	2134|132|54321|213|132	183	429	211.91	0.914
2	$R(300)\rightarrow$max	2314|312|43521|321|123	198	429	220.22	0.915
3	Increasing τ	1234|123|12345|123|123	177	449	213.84	0.909
4	$\widetilde{\varepsilon}(\infty)\rightarrow$max	4312|213|52134|132|231	247	432	277.67	0.776

Now consider the same system with $L_1 = L_2 = L_3 = L_4 = L_5 = 1$, which corresponds to consecutive execution of versions one at a time. The maximal possible time of system execution in this case does not depend on the versions sequence and is equal to the sum of execution times of all of the versions. The overall system reliability that does not depend on either version sequencing or on L_c is still $R(\infty) = 0.92$. The same four types of solution that were obtained in the previous example are presented in Table 8.7 (the maximal allowable execution time in this case is $w = 430$).

Table 8.7. Parameters of solutions obtained for the fault-tolerant system with $L_1 = L_2 = L_3 = L_4 = L_5 = 1$

No.	Problem formulation	Sequence of versions	$T_{\min}$	$T_{\max}$	$\widetilde{\varepsilon}(\infty)$	$R(430)$
1	$\widetilde{\varepsilon}(\infty)\rightarrow$min	2134|3122|12435|213|123	251	687	313.02	0.886
2	$R(430)\rightarrow$max	2314|132|41235|231|123	266	687	321.33	0.892
3	increasing τ	1234|123|12345|123|123	242	687	316.19	0.886
4	$\widetilde{\varepsilon}(\infty)\rightarrow$max	4312|321|53421|312|321	449	687	469.62	0

Observe that the minimal possible system execution time for the solution with the greatest possible $\tilde{\varepsilon}(\infty)$ is greater than w. Therefore, the corresponding $R(w) = 0$. The system reliabilities $R(w)$ as functions of maximal allowable execution time w are presented in Figure 8.4B for all of the solutions obtained.



A **B**

Figure 8.4. $R(w)$ functions for solutions obtained for the system with $L_1 = 1$, $L_2 = 2$, $L_3 = 4$, $L_4 = 1$, $L_5 = 3$ (A) and for the system with $L_1 = L_2 = L_3 = L_4 = L_5 = 1$ (B)

8.3 Optimal Structure of Fault-tolerant Software Systems

When a fault-tolerant software system is designed, one has to choose software versions for each component and find the sequence of their execution in order to achieve the system's greatest reliability subject to cost constraints. The versions are chosen from a list of the available products. Each version is characterized by its reliability, execution time, and cost. The total cost of the system's software is defined according to the cost of its versions. The cost for each version can be the purchase cost if the versions are commercial and the off-the-shelf cost, or it can be an estimate based upon the version's size, complexity, and performance.

Assume that B_c functionally equivalent versions are available for each component c and that the number k_c of the versions that should agree in each component is predetermined. The choice of the versions and the sequence of their execution in each component determine the system's entire reliability and performance.

The permutation $\boldsymbol{x}^*_c$ of B_c different integer numbers ranging from 1 to B_c determines the order of the version that can be used in component c. Let $y_{cb} = 1$ if the version b is chosen to be included in component c and $y_{cb} = 0$ otherwise. The binary vector $\boldsymbol{y}_c = \{y_{c1},...,y_{cB_c}\}$ determines the subset of versions chosen for

component c. Having the vectors $\boldsymbol{x}^*_c$ and $\boldsymbol{y}_c$ one can determine the execution order $\boldsymbol{x}_c$ of the versions chosen by removing from $\boldsymbol{x}^*_c$ any number b for which $y_{cb} = 0$. The total number of versions in component c (equal to the length of vector $\boldsymbol{y}_c$ after removing the unchosen versions) is determined as

$$n_c = \sum_{b=1}^{B_c} y_{cb} \tag{8.20}$$

The system structure optimization problem can now be formulated as find vectors $\boldsymbol{x}_c$ for $1 \le c \le C$ that maximize $R(w)$ subject to cost constraint

$$\Omega = \sum_{c=1}^{C} \sum_{b \in \boldsymbol{x}_c} \omega_{cb} \le \Omega^* \tag{8.21}$$

where ω_{cb} is the cost of version b used in component c, Ω is the entire system cost and Ω^* is the maximal allowable system cost. Note that the length of vectors $\boldsymbol{x}_c$ can vary depending on the number of versions chosen.

In order to encode the variable-length vectors $\boldsymbol{x}_c$ in the GA using the constant-length integer strings one can use (B_c+1)-length strings containing permutations of numbers $1,\ldots, B_c, B_c+1$. The numbers that appear before B_c+1 determine the vector $\boldsymbol{x}_c$. For example, for $B_c = 5$ the permutations (2, 3, 6, 5, 1, 4) and (3, 1, 5, 4, 2, 6) correspond to $\boldsymbol{x}_c = (2, 3)$ and $\boldsymbol{x}_c = (3, 1, 5, 4, 2)$ respectively. Any possible vector $\boldsymbol{x}_c$ can be represented by the corresponding integer substring containing the permutation of B_c+1 numbers. By combining C substrings corresponding to different components one obtains the integer string $\boldsymbol{a}$, that encodes the entire system structure.

As in the version sequencing problem (Section 8.2), the encoding method is used in which the single permutation defines the sequences of the versions chosen in each of the C components. The solution encoding string is a permutation of $n = \sum_{c=1}^{C}(B_c+1)$ integer numbers ranging from 1 to n. Each number j belonging to the interval $\sum_{c=1}^{m-1}(B_c+1)+1 \le j \le \sum_{c=1}^{m}(B_c+1)$ corresponds to version $j - \sum_{c=1}^{m-1}(B_c+1)$ of component m. The relative order in which the numbers corresponding to the versions of the same component appear in the string determines the structure of this component.

Example 8.5

Consider, for example, a system consisting of three components with $B_1 = 3$, $B_2 = 5$ and $B_3 = 4$. The solution encoding strings consist of (3+1)+(5+1)+(4+1) = 15 integers. Numbers from 1 to 4 correspond to component 1, numbers from 5 to 10 correspond to component 2, and numbers from 11 to 15 correspond to component

3. In the solution string $\boldsymbol{a}$ = (5, 9, 7, 1, 12, 2, 15, 3, 10, 6, 14, 8, 4, 13, 5, 11), the numbers corresponding to different components appear in the following relative order:

for component 1: 1, 2, 3, **4**
for component 2: 5, 9, 7, **10**, 8
for component 3: 12, **15**, 14, 13, 11

Only the numbers located before the largest number in each substring (marked in bold) represent the component's structure. This gives the following substrings:

for component 1: 1, 2, 3; for component 2: 5, 9, 7; for component 3: 12

These substrings correspond to the following sequences of versions execution:

in component 1: 1, 2, 3; in component 2: 1, 5, 3; in component 3: 2

The solution decoding procedure determines the sequence of versions in each component. Then it determines the system reliability $R(w)$ as described in Section 8.1 and calculates the system cost using Equation (8.21). The solution fitness is evaluated as $R(w)-\pi \max(\Omega-\Omega^*, 0)$, where π is a penalty coefficient.

Example 8.6

Consider a fault-tolerant software system consisting of four components running on fully available hardware [203]. The parameters of the versions that can be used in these components are presented in Table 8.8. This table contains the values of k_c and L_c for each component and the cost, reliability, and execution time for each version.

Table 8.8. Parameters of fault-tolerant system components and versions

No. of component	k_c	L_c		Versions							
				1	2	3	4	5	6	7	8
1	1	1	τ	17	10	20	32	30	75	-	-
			r	0.71	0.85	0.85	0.89	0.95	0.98	-	-
			c	5	15	7	8	12	6	-	-
2	2	2	τ	28	55	35	55	58	-	-	-
			r	0.82	0.82	0.88	0.90	0.93	-	-	-
			c	11	8	18	10	16	-	-	-
3	3	4	τ	17	20	38	38	48	50	41	63
			r	0.80	0.80	0.86	0.90	0.90	0.94	0.98	0.98
			c	4	3	4	6	5	4	9	6
4	2	1	τ	17	10	20	32	-	-	-	-
			r	0.75	0.85	0.93	0.97	-	-	-	-
			c	12	16	17	17	-	-	-	-
5	1	3	τ	30	54	40	65	70	-	-	-
			r	0.70	0.80	0.80	0.80	0.89	-	-	-
			c	5	9	11	7	12	-	-	-

Twosets of solutions were obtained for the maximal allowable system operation times w = 250 and w = 300. For each value of w, four different solutions were

obtained for different cost constraints. These solutions are presented in Tables 8.9 and 8.10. The tables contain the system cost and reliability for each solution, the expected conditional execution time, minimal and maximal possible system execution times, and the corresponding sequences of the versions chosen.

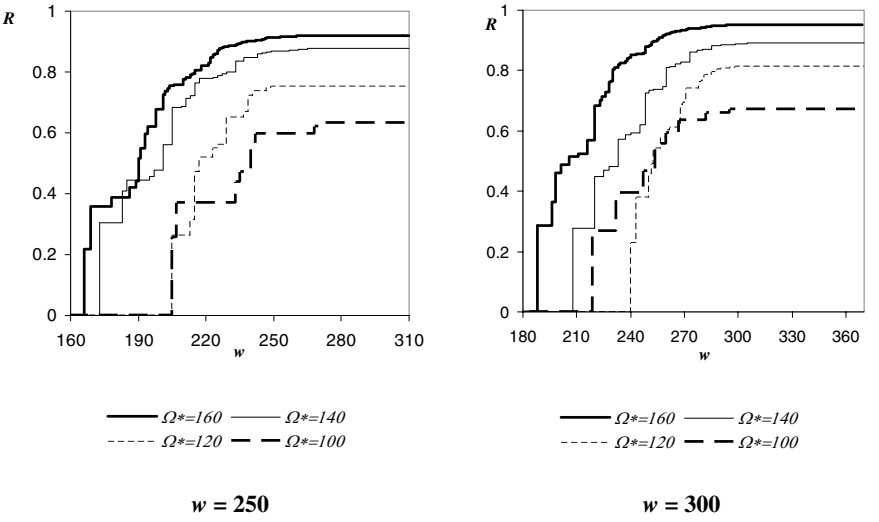
The functions $R(w)$ for the solutions obtained are presented in Figure 8.5.

Table 8.9. Parameters of solutions obtained for $w = 250$

Ω^*	Sequence of versions	Ω	T_{min}	T_{max}	$\tilde{\varepsilon}(\infty)$	$R(250)$
160	231|541|37162|324|214	159	166	307	188.34	0.913
140	34|241|64231|234|123	140	173	301	194.43	0.868
120	5|431|31562|43|21	119	205	249	217.07	0.752
100	3|241|4562|43|41	100	205	270	220.52	0.598

Table 8.10. Parameters of solutions obtained for $w = 300$

Ω^*	Sequence of versions	Ω	T_{min}	T_{max}	$\tilde{\varepsilon}(\infty)$	$R(300)$
160	341|4521|85632|324|41	160	188	369	210.82	0.951
140	53|541|28361|431|51	140	208	335	231.02	0.889
120	6|241|61372|241|31	120	240	307	252.87	0.813
100	4|142|2386|43|41	100	219	295	238.05	0.672



$w = 250$ **$w = 300$**

Figure 8.5. $R(w)$ functions for the solutions obtained

Observe that the greater the reliability level achieved, the greater the cost of further reliability improvement. The cost-reliability curves are presented in Figure 8.6. Each point on these curves corresponds to the best solution obtained by the GA.

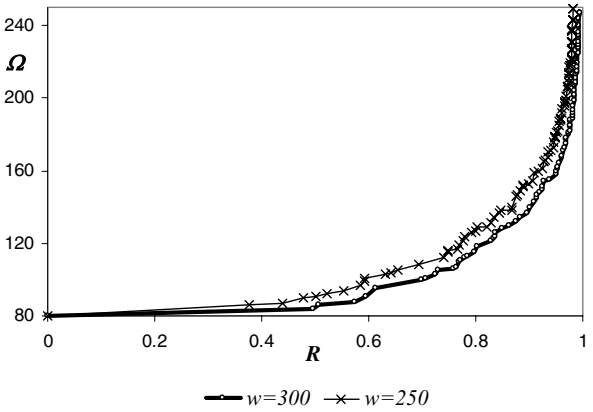


Figure 8.6. Reliability-cost curves for the best obtained solutions