Towards bandwidth optimal temporal partitioning

Attila Zabos, Alan Burns
Department of Computer Science
University of York, UK

Abstract

In real-time systems the available processing time is partitioned in order to enforce separation of concerns of concurrent application components. This is a necessary step to prevent possible side effects of an erroneous application component on other components in the system. One of the main objectives in temporal partitioning is the selection of appropriate partition parameters (e.g. replenishment period and budget) that minimise the resulting resource bandwidth reservation. Currently there are only heuristic limits on the search space of bandwidth optimal virtual resource (VR) parameters. We provide a lower and upper bound on the period of a periodic VR in order to limit the search space for bandwidth optimal parameters. We also show that the VR parameters derived from the VR period upper bound provide sufficiently good results if the time for VR parameter calculation is limited and an exhaustive search for optimal parameter values is not an option.

1. Introduction and motivation

Due to the increasing processing power of complex embedded real-time systems, there is a trend to integrate real-time application components, that were initially developed and verified independently of each other and also executed on distinct hardware, on a single system. Independently developed and verified application components imply that each of them is associated with a different system function. Hence the diversity of application components on a single integrated platform requires isolation of various application components in order to avoid interference among each other. The requirement for separation of concerns in multi-application systems makes the temporal partitioning of the available processing time essential.

A common approach to implement the required temporal partitioning is by periodic virtual resources (VRs). VRs act as a resource reservation mechanism and manage the processing time budget that is assigned to application components. Characteristic parameters of periodic VRs are their budget and replenishment period. The VR budget is consumed while the associated application component’s tasks are executed and at regular time intervals specified by the VRs replenishment period, this budget is replenished to its initial value.

The selected VR parameter values can be considered as globally or locally optimal [4]. They are globally optimal if the minimum system utilisation, including all the other VRs in the system, is achieved. On the other hand, the parameter values of a single VR, considered in isolation, are locally optimal if the obtained VR utilisation is the smallest.

The main challenge in the selection of locally optimal VR parameters is that the demand requirements of the associated application component are satisfied while the processing time reservation for the application component is kept at a minimum. The VR budget and period parameters leading to this minimal resource reservation (i.e., minimum VR utilisation) are considered in this paper as the bandwidth optimal parameters or simply just optimal parameters.

Various deficiencies of existing approaches inhibit the calculation of locally optimal VR parameters. The origin of these deficiencies are either:

- the application of a linear VR supply model [11, 1] for VR parameter calculation,
- the requirement of additional VR parameter constraints (e.g. the initial delay of the VR supply function [11] or the specification of the VR period [6]) that need to be specified by the application designer, or
- the definition of a search space that might not contain the optimal VR parameters [12, 7].

In contrast to previous approaches we utilise the exact supply bound function of the VR and avoid inaccuracy due to a linear VR model. Furthermore, our analysis requires only the taskset temporal specification, enabling the calculation of optimal VR parameters without the intervention of the application designer.

One method to determine optimal VR parameters for an application component is to provide a VR period for which the smallest possible VR budget can be efficiently determined (similar to the approach introduced for the EDP resource model [6]). This approach implies a brute-force search over an interval of possible VR period values. To our best knowledge there is no definition of an interval that ensures the existence of a VR period from which the optimal VR parameters can be derived.
We approach the problem of optimal VR parameter selection by providing a lower and upper bound on the VR period, thus limiting the search space. It shall be noted that the optimal VR period is not necessarily smaller than the smallest task period or its deadline.

In this paper we examine two level hierarchical real-time systems with a fixed-priority (FP) pre-emptive scheduler at both the operating system and the application component level.

The paper is structured as follows: Section 2 gives an overview of related work in the area of VR parameter selection. The model that is assumed in order to determine the VR parameters is defined in Section 3. The analytical tools that are used in order to derive the optimal VR period upper bound are presented in Section 4. Our contribution, the calculation of an upper and lower bound on the optimal VR period is presented in Section 5 and Section 6, respectively. Section 7 demonstrates with a numerical example how to calculate the VR period bounds using the algorithms and equations that are presented in this paper. The evaluation of the quality of the VR parameters derived from the VR period upper bound and the length of the optimal VR period search interval is provided in Section 8. The conclusion and the summary of our contribution is presented in Section 9.

2. Related work

In the past, various research work [5, 9, 15, 3] provided in-depth analysis of hierarchical real-time systems, leaving the question of efficient VR parameter selection open.

Mok et al. [14] presented the bounded-delay resource partition model that makes use of the linear supply function defined by an availability factor and a partition delay. Since it is a general resource model there may exist many periodic VRs that satisfy a given bounded-delay model specification. Feng and Mok [8] extended the bounded-delay model by an additional constraint that specifies the smallest scheduling quantum of a VR. This constraint shall limit the number of VR context switches and reduce the resulting context switch overhead.

The application of the bounded-delay resource model [14] on the selection of periodic VR parameters was investigated by Shin and Lee [16] but without the option to determine the optimal parameters.

A linear model to represent the VR supply is also used by Lipari and Bini [11]. The VR model is characterised by an initial delay that occurs before the VR supply starts at a uniform rate. This model has the same disadvantage as the bounded-delay model.

Using an Explicit Deadline Periodic model, Easwaran et al. [6] presented an algorithm to determine the optimal VR budget for a given VR period. But without the knowledge of the optimal value or the upper bound of the VR period, this algorithm can generate an infinite set of VR parameters.

Exact schedulability conditions (based on the VR supply function and the demand bound function) in order to determine whether an application component’s taskset is schedulable or not is specified by Lipari and Bini [12] and Easwaran et al. [7]. In order to use the schedulability condition to determine the optimal VR parameters implies a brute force search over the interval of possible VR periods.

As a reasonable upper bound for the search space of a VR period (leading to sufficiently good VR utilisation), the least common multiple of task periods or the task deadlines were suggested by Easwaran et al. [7], and by Lipari and Bini [12], respectively. We will show in an example that for optimal parameters the VR period may be larger than the smallest task period or deadline, and consequently also larger than the least common multiple of task periods.

The selection of periodic VR parameters based on the task demand function was analysed by Almeida and Pedreiras [1]. They examined the demand at each priority level and determined relevant demand points that were used as input for the VR parameter calculation algorithm. The information about the taskset demand at the time instant of task deadlines was used in order to determine the required processing time of the corresponding taskset. This is a pessimistic assumption since there might be a lower taskset demand at an earlier time instant than at the task deadline. Furthermore, their approach also used a linear VR supply function to determine the final VR budget and period.

The selection of optimal VR parameters was also investigated by Davis and Burns [4]. For a set of periodic VRs, the parameter selection was examined if two out of the three VR parameters (i.e. budget, period or priority) were given.

In this work we utilise the idea of demand function based VR parameter selection [1] and the observation that the task demand increases only at task release times [6].

3. System model

In this paper we investigate uniprocessor real-time systems. It is also assumed that these systems use a two-level hierarchical fixed-priority pre-emptive scheduling. At the operating system level, the global scheduler determines the processor allocation to multitasking application components (in the following only referred to as application component). Within a scheduled application component the local scheduler then determines which of the application component’s ready tasks shall execute.

In order to enforce temporal partitioning among the application components in the system, the periodic VR model is used as a resource reservation mechanism. Each application component $A_i$ is associated with a virtual resource (VR) $S_i$ that manages the predefined budget for this application component and the corresponding taskset. That means there is a one to one mapping between application components and VRs.
In this paper we are concerned with periodic VRs, where each VR $S_i$ is defined by its budget $\Theta_i$ and replenishment period $\Pi_i$. The budget $\Theta_i$ is decreased with the same rate as time progress, when ready tasks of the corresponding application component $A_i$ execute. If the VR’s budget is exhausted, the VR is suspended until its budget is replenished to its full capacity. The time instants of budget replenishment are determined by the VR’s replenishment period $\Pi_i$. The behaviour of the periodic VR is identical to a classical periodic server with the exception that the server’s budget is not exhausted instantaneously if there are no tasks in the ready queue waiting for execution. In the case that there are no tasks ready to execute, the periodic VR’s budget is continuously consumed by an idle task until it is exhausted [3].

Within a time interval between two VR budget replenishment events, the maximum time without processing time supply is defined as $\delta = \Pi - \Theta$.

The supply bound function $s(t)$ determines for a periodic VR and a time instant $t$ the cumulative processing time available to execute the corresponding application component’s tasks. The supply bound function for periodic VRs is a piecewise defined function and it can be expressed as in Equation 1 [1].

$$s(t) = \begin{cases} 0 & : t \leq \Delta \\ t - (\Delta + m\delta) & : \Delta + m\Pi \leq t < \Delta + m\Pi + \Theta \\ (m + 1)\Theta & : \Delta + m\Pi + \Theta \leq t \leq \Delta + (m + 1)\Pi \\ \end{cases}$$  

(1)

$m = \left\lfloor \frac{(t - \Delta)}{\Pi} \right\rfloor$

$\Delta = 2 \cdot \delta = 2 \cdot (\Pi - \Theta)$

Within an application component, each task $\tau_j$ is determined by the tuple $(C_j, D_j, T_j, P_j)$, with $C_j$ denoting the task’s worst-case execution time, $D_j$ its deadline and $T_j$ its period. The task’s deadline may be less than or equal to its period (i.e. $D_j \leq T_j$). Additionally, each task has a unique priority $P_j$ within its associated application component. The priority $P_j$ is determined according to the deadline-monotonic priority assignment policy.

Furthermore, the task release time is considered to be independent of the VR’s replenishment period. This notion of task releases has an impact on the analysis and it will be further described in Section 5.

4. Analysis

In this section we introduce the tools that will be utilised in Section 5 to determine the optimal VR period upper bound.

4.1. Demand points

In order to simplify the analysis of the task requirements and the VR parameter calculation, the demand requirements for each priority level $i$ (expressed by the demand bound function [10]) are mapped to a representative demand point $DP_i$.

A demand point $DP_i$ for priority level $i$ is a pair $(q_i, t_i)$, containing the information about the priority level demand $q_i$ at a certain time instant $t_i$.

The demand bound function for a priority level $i$ is defined in Equation 2 with $hep(i)$ denoting priority levels equal or higher than $i$.

$$dbf_i(t) = \sum_{j \in hep(i)} \left[ \frac{t}{T_j} \right] \cdot C_j$$  

(2)

The analysis of Equation 2 shows that the demand bound function has local minima at time instants that are multiples of task periods. This was also observed by Easwaran et al. [6]. We exploit this fact and determine a specific demand point $DP_i$ for each priority level $i$. This enables us to represent the demand requirements of an entire taskset by a set of demand points.

Algorithm 1 is used to determine the demand points that represent the minimum demand requirements of a given taskset.

**In** : Priority ordered taskset

**Out** : Set of demand points

1. foreach priority level $i$ do

2. At each priority level $i$ determine a set of $(q_m, t_m)$ pairs. Each $(q_m, t_m)$ pair represents a demand point $DP_m$ with the cumulative demand $q_m$ of the tasks $\tau_j : j \in hep(i)$ at task release times $t_m \in \mathcal{T}_i = \{x \mid x = n \cdot T_j \leq D_i\}$;

3. For priority level $i$ let $DP_i$ denote $(q_i, t_i) = (q_m, t_m) : \min_m (q_m/t_m)$ with the smallest ratio $(q_m/t_m)$. If multiple $(q_m, t_m)$ pairs satisfy this condition, then choose the one with the largest $t_m$;

4. Remove redundant demand points;

**Algorithm 1**: Demand point calculation

In summary, the algorithm determines for each priority level $i$ a set of time instants $T_i$ when a task with priority $j \in hep(i)$ is released. At the examined priority level $i$, for each time instant $t_m \in T_i$ the corresponding demand $q_m$ is calculated. From the set of $(q_m, t_m)$ pairs, the one with the lowest ratio is selected and denoted as the representative demand point $DP_i$ for priority level $i$. Before the algorithm terminates, redundant elements are removed from the set of demand points. A demand point $(q_r, t_r)$ is considered to be a redundant element if there is another element $(q_i, t_i)$ such that $t_r = t_i$ and $q_r < q_i$. That means, for a specific time instant $t$ only the demand point with the largest demand is retained. The removal of redundant points from the set of demand points (last step in Algorithm 1) also implies that every time instant $t_i$ is associated to only one demand requirement $q_i$.

A demand point $DP_i$ can also be considered as a constraint on the slowest processor speed that is required in...
order to satisfy the taskset demand at priority level $i$.

It should be noted that line 2 of Algorithm 1 may require a lot of iterations if in a multitasking system at least one of the tasks has a very short or very long period in contrast to the other tasks. However, it is expected that for most embedded real-time systems the range of task periods will be relatively small (e.g. in avionic [13] or automotive system, from a few milliseconds to a second). Thus for these systems, line 2 of Algorithm 1 generates at most 1000 demand points and constrains the number of iterations of Algorithm 1 within reasonable limits.

To demonstrate Algorithm 1 on an example, we examine the taskset defined in Table 1 and assume that task deadlines are equal to their periods.

**Table 1. Taskset example**

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
<th>$C_t$</th>
<th>$T_v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1 (high)</td>
<td>400</td>
<td>1300</td>
</tr>
<tr>
<td>B</td>
<td>2 (medium)</td>
<td>800</td>
<td>4600</td>
</tr>
<tr>
<td>C</td>
<td>3 (low)</td>
<td>1000</td>
<td>6800</td>
</tr>
</tbody>
</table>

In Figure 1 the demand function of each priority level is rendered along with its representative demand point. The line of the slowest processor speed that supplies just sufficient capacity to satisfy the corresponding priority level demand, is rendered as well.

**Figure 1. Demand points example**

Using the aforementioned taskset (defined in Table 1) as an example and applying Algorithm 1 on it, provides the demand point $DP_1 = (400, 1300)$, $DP_2 = (2000, 3900)$ and $DP_3 = (4600, 6500)$ for priority level 1, 2 and 3, respectively. The slowest processor speed that is required to satisfy the taskset demand at each priority level is depicted in Table 2. The slower processor speed can be also considered as the percentage of the initial processing speed.

In this example, the slowest feasible processor speed is determined by the demand point $DP_3$ at priority level 3, since it represents the maximal expected taskset demand per time unit. In this case, the processor speed could be slowed down approximately to 70.77% of its initial processing speed, without rendering the taskset unschedulable.

### 4.2. Optimal VR utilisation

Plotting the minimum VR utilisation as a function of the VR period, reveals a sawtooth shape graph (see Figure 2). The minimum VR utilisation for a VR period (further referred to as the optimal VR utilisation) is determined by calculating the smallest VR budget such that the associated taskset is still schedulable.

The problem of efficient optimal VR parameter calculation due to the sawtooth shape of the optimal VR utilisation graph was also observed by Davis and Burns [3].

Figure 2 shows the minimum VR utilisation as a function of the VR period. For one line on the graph, the VR utilisation values were determined without considering VR context switch overhead, and for the second line on the graph a context switch overhead of 100 ticks was specified.

For the VR with 100 ticks context switch overhead, Figure 2 shows that the rightmost trough on the minimal VR utilisation graph also represents the optimal VR parameters. However, this is not always the case as we will see in the following sections.

**Figure 2. Minimum VR utilisation**

Furthermore, the examination of the minimum VR utilisation graphs indicates that there exists a VR period beyond which the function of the VR utilisation is strict monotonically increasing. Therefore, the optimal VR budget and period parameters must lie at or below this value that we denote as the VR period upper bound.

### 5. VR period upper bound

In this section we use the tools presented in Section 4 to derive an equation for the VR period upper bound.

For the calculation of the VR parameters, we assume a discrete time space. Hence in further equations, the VR period is mapped to an integer value by applying the
floor function. For the same reason, the VR budget is mapped to an integer by applying the ceiling function. Generally, these discretisation steps are not limited to integer values but could be carried out to an arbitrary precision.

5.1. VR context switch excluded

First, we consider the scenario in which there is no VR context switch overhead and task release times are not synchronised with the replenishment period of the VR.

Since the task release times are not synchronised to the VR’s replenishment period, the worst-case task execution delay occurs under the following circumstances.

The VR budget is consumed by lower priority tasks starting right after the VR’s replenishment period. In the following VR period, the budget supply experiences the maximum possible delay due to higher priority VRs. Therefore, after a task is released, its execution can be delayed by $\Delta = 2 \cdot \delta = 2 \cdot (\Pi - \Theta)$ in the worst-case situation (see Figure 3).

![Figure 3. Maximum delay of VR supply](image)

Figure 3. Maximum delay of VR supply

Figure 2 indicates that there is a VR period $\Pi^*$ such that for all other VR period values $\Pi > \Pi^*$ the VR utilisation $U(\Pi)$ monotonically increases.

In the following, we provide the necessary statements that can be used to define the algorithm for the optimal VR period upper bound calculation.

**Lemma 1.** (Monotonic increase of the VR utilisation) Considering a single demand point $DP_i = (q_i, t_i)$, the optimal virtual resource utilisation $U(\Pi)$ (satisfying the demand specified by $DP_i$) monotonically increases with the VR period $\Pi \geq \left\lfloor \frac{t_i + q_i}{2} \right\rfloor$.

**Proof.** With an optimal VR utilisation, the VR is supplying only as much processing time as dictated by the demand point $DP_i$. Hence, the VR supply stops right at the demand point $DP_i$, implying $s(t_i) = q_i$ (see Figure 4).

Additionally, the examination of the second case of the piecewise defined VR supply bound function and the corresponding graph (see Figure 4) shows that a strict monotonic increase of the VR utilisation occurs if the demand point $DP_i$ experiences a single VR replenishment period.

![Figure 4. Single VR replenishment](image)

Figure 4. Single VR replenishment

Consequently a single VR replenishment period up to $t_i$ implies that the VR budget $\Theta$ is equal to the demand requirement $q_i$. Furthermore, considering a single replenishment period, the supply bound function can be simplified to $s(t) = t - \Delta$ and the corresponding VR period can be derived as:

$$s(t_i) = q_i$$
$$t_i - \Delta = q_i$$
$$t_i - 2 \cdot (\Pi - \Theta) = q_i$$
$$t_i - 2 \cdot (\Pi - q_i) = q_i$$

$$\Pi = \left\lfloor \frac{t_i + q_i}{2} \right\rfloor$$

Given a single VR replenishment up to $t_i$, we show that if the VR period $\Pi$ is increased by an arbitrary positive value $x$, i.e. $\Pi = \frac{t_i + q_i}{2} + x$, the VR budget needs to be increased by the same value $x$ in order to preserve $s(t_i) = q_i$.

$$s(t_i) = q_i$$
$$t_i - \Delta = t_i - 2 \cdot (\Pi - \Theta) = q_i$$
$$\Theta = \frac{q_i - t_i}{2} + \Pi$$
$$\Theta = \frac{q_i - t_i}{2} + \frac{t_i + q_i}{2} + x$$

$$\Theta = q_i + x$$

It can be concluded that for demand point $DP_i = (q_i, t_i)$ and a VR period $\Pi \geq \frac{t_i + q_i}{2}$, the VR’s initial delay $\Delta$ remains constant if the VR budget and period are increased by the same value $x$.

$$\Pi = \frac{t_i + q_i}{2}$$
$$\Theta = q_i$$

$$\Delta = (\Pi + x) - (\Theta + x)$$
$$\Delta = \Pi - \Theta$$

$$\Delta = \frac{t_i + q_i}{2} - q_i = \frac{t_i - q_i}{2}$$
To show that the VR utilisation $U(\Pi)$ is a monotonically increasing function for $\Pi \geq \frac{t_i + q_i}{2}$, we define $U = \frac{\Theta}{\Pi}$. Considering that $\Pi - \Theta = \delta$ is constant, the VR utilisation can be rewritten as $U(\Pi) = \frac{\Pi - \delta}{\Pi}$. In order to prove that $U(\Pi)$ is monotonically increasing, we have to show that $\Pi_1 < \Pi_2 \Rightarrow U(\Pi_1) < U(\Pi_2)$.

$$\Pi_1 < \Pi_2 \Rightarrow \frac{\delta}{\Pi_1} < \frac{\delta}{\Pi_2} \Rightarrow 1 - \frac{\delta}{\Pi_1} < 1 - \frac{\delta}{\Pi_2} \Rightarrow \frac{\Pi_1}{\Pi_2 - \delta} < \frac{\Pi_1}{\Pi_2}$$

Thus, $U(\Pi_1) < U(\Pi_2)$.

Furthermore, it can be observed that for optimal periodic VRs the demand point with the smallest difference $(t_i - q_i)$ has a significant influence on the VR period upper bound calculation. It defines the maximal VR initial delay $2 \cdot (\Pi - \Theta)$ (see Figure 5). A supporting argument is presented in Lemma 2.

**Lemma 2.** (Maximal VR initial delay) The demand point $DP_s = (q_s, t_s)$ determines the maximal initial delay of a VR.

**Proof.** The equations to determine the VR parameters under the assumption of a single replenishment period were derived in Lemma 1. Lemma 1 also implies that the VR supply experiences the maximum possible delay and the supply is equal to the demanded processing time, $s(t_s) = q_s$.

Thus, among all the demand points determined by Algorithm 1, $DP_s = (q_s, t_s) = \min_i(t_i - q_i)$ specifies the VR’s maximal initial delay of $t_s - q_s$.

For a longer than $t_s - q_s$ VR initial delay the VR could not supply sufficient processing time in order to satisfy the requirement of the demand point $DP_s$. Therefore the demand point $DP_s = (q_s, t_s) = \min_i(t_i - q_i)$ determines the maximal initial delay of a VR.

We are interested in the largest possible VR period that might enable us to determine the optimal VR parameters. Theorem 1 provides the essential means for an algorithm to determine a bandwidth optimal VR period upper bound.

**Theorem 1.** (Optimal VR period upper bound) The optimal VR period upper bound is $\Pi^u = \Pi_s + x$ and the corresponding minimal VR budget $\Theta^u = \Theta_s + x$ with:

$$DP_s = (q_s, t_s) = (q_i, t_i): \min_i(t_i - q_i)$$

$$\Pi_s = \frac{t_s + q_s}{2}$$

$$\Theta_s = q_s$$

and $x$ being a positive integer number such that for all demand points $DP_i$ the condition $s(t_i) \geq q_i$ holds.

**Proof.** Lemma 2 provided $(q_s, t_s) = \min_i(t_i - q_i)$ as the smallest possible VR initial delay. Lemma 1 showed that for $\Pi \geq \frac{t_s + q_s}{2}$ the VR utilisation $U(\Pi)$ monotonically increases with the VR period $\Pi$, and implies that:

$$\Pi = \Pi_s + x \geq \Pi_s = \left[ \frac{t_s + q_s}{2} \right]$$

The increment $x$ of the initial VR parameters $(\Theta_s, \Pi_s)$ can be determined by calculating the required increase for the VR budget in order to meet all task demand requirements.

It shall be noted that in further steps, references to all demand points imply the exclusion of $DP_i$ from the set of demand points.

Now we take a counter intuitive step and first calculate the minimal VR budget $\Theta^u$ (see Equation 3) for the VR period $\Pi^u$ that is yet to be determined. $\Theta^u$ is the smallest VR budget that is required to satisfy the condition $\forall i: s(t_i) \geq q_i$. The number of VR replenishment periods up to the demand point $DP_i$ is represented by $h_i$.

$$\Theta^u = \max_i \left( \frac{q_i}{h_i} \right)$$

(3)

The time interval taken into account to determine $h_i$ is $[(\Pi - \Theta), t_i]$, i.e. excluding the inactive VR time that originates from the VR period before the critical instant.

In order to derive $h_i$, a geometric mapping of the demand points onto the X-axis was carried out (see Equation 4 and Figure 5). After the mapping, the demand point $(q_i, t_i)$ becomes the point $(t_i - q_i)$ on the X-axis. The end points of the VR supply, also mapped onto the X-axis, reveal that the distance between these points is $\Pi - \Theta$.

$$DP_i = \begin{bmatrix} q_i \\ t_i \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} q_i \\ t_i \\ 1 \end{bmatrix} = \begin{bmatrix} t_i - q_i \\ 0 \\ 1 \end{bmatrix}$$

(4)
After the transformation of all demand points $DP_1$ to $DP_1'$, the number of complete VR replenishment periods $h_i$ up to the time instant $t_i$ can be calculated as specified in Equation 5.

$$h_i = \left\lfloor \frac{(t_i - q_i) - (\Pi_s - \Theta_s)}{\Pi_s - \Theta_s} \right\rfloor \tag{5}$$

In the presence of VR context switch overhead the approach with the mapping of points to the X-axis simplifies the calculation of the number of VR supplies.

Knowing the VR budget $\Theta_u$, the budget increase $\Theta_d$ over the VR’s initial value $\Theta_s$ can be easily determined. The VR budget increase $\Theta_d$ entails an increase of the initial VR period $\Pi_s$ by the same value (see Lemma 1) leading to the bandwidth optimal VR period upper bound. As shown in Equation 6, without considering the VR context switch overhead in the calculation of the bandwidth optimal VR period, the increase of the VR budget and period, $I_d$, is determined only by the required VR budget increase $\Theta_d$. In Section 5.2, considering VR context switch overhead, the calculation of the VR budget and period increase $I_d$ will be extended.

$$\Theta_d = \Theta_u - \Theta_s$$

$$I_d = \Theta_d + I_d$$ \tag{6}

In summary, given $\Pi_u$ and $\Theta_u$ we can conclude that the VR utilisation $U(\Pi)$ monotonically increases for $\Pi \geq \Pi_u$.

In the following we show that by keeping the VR budget constant and decreasing the VR period, for $\Pi \leq \Pi_u$, the VR utilisation $U(\Pi)$ again monotonically increases.

With $\Theta_u = \Theta_s + I_d$ denoting the smallest possible VR budget for $\Pi_u = \Pi_s + I_d$ implies that $\Theta_u$ cannot be reduced by any arbitrary small positive integer value $\epsilon < I_d$, otherwise the solution for the optimal VR parameters would be $\Theta_u = \Theta_s + I_d - \epsilon$ and $\Pi_u = \Pi_s + I_d - \epsilon$.

But a reduction of the optimal VR period $\Pi_u$ is possible without violating the demand requirements, since $(\Pi_u - \epsilon)$ has two effects on the VR supply:

1. the initial delay $\Delta = 2 \cdot ((\Pi_u - \epsilon) - \Theta_u)$ decreases and,
2. the replenishment periods are shortened, thus the VR still supplying the budget $\Theta_u$ just in shorter periods.

Hence, additionally to Lemma 1 the VR utilisation $U(\Pi)$ increases also with $\Pi : (\Pi_u - \epsilon) < \Pi < \Pi_u$. Thus, proving that $\Pi_u$ denotes the largest period contributing to a local optimum on the optimal VR utilisation graph $U(\Pi)$.

5.2. VR context switch included

In this section we extend the analysis provided in Section 5.1 by the context switch overhead, also denoted as virtual resource context switch overhead, that occurs during the scheduling of application components.

Without loss of generality, we assume that a context switch overhead $C_0$ occurs in each replenishment period at the beginning of the VR supply.

First, the equations that calculate the initial VR parameter values need to be extended in order to consider the time consumed by the context switch (see Equations 8 and 9).

$$DP_s = (q_i, t_s) = (q_i, t_i) : \min_i (t_i - q_i - C_0) \tag{7}$$

$$\Pi_s = \left\lfloor \frac{t_s + q_s + C_0}{2} \right\rfloor \tag{8}$$

$$\Theta_s = q_s + C_0 \tag{9}$$

Since we account for one context switch overhead during each replenishment period the number of context switches up to the time instant $t_i$ is equal to the number of VR replenishment periods $h_i$. In order to determine the number of VR processing time supplies for the demand point $DP_1 = (q_i, t_i)$, we assume at $t_i$ a demand of $q_i + h_i \cdot C_0$. The mapped value of the demand point with VR context switch overhead included is equal to:

$$DP'_i = \begin{bmatrix} q'_i \\ t'_i \end{bmatrix} = \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} q_i + h_i \cdot C_0 \\ t_i \\ 1 \end{bmatrix}$$

Using the demand points $DP'_i$ with the incorporated context switch information, Equation 5 can be easily modified to account for the VR context switch overhead (see Equation 10).

$$h_i = \left\lfloor \frac{(t_i - q_i - h_i \cdot C_0) - (\Pi_s - \Theta_s)}{\Pi_s - \Theta_s} \right\rfloor$$

$$h_i \leq \left\lfloor \frac{(t_i - q_i - h_i \cdot C_0) - (\Pi_s - \Theta_s)}{\Pi_s - \Theta_s} \right\rfloor$$

$$h_i + h_i \cdot C_0 \leq t_i - q_i - 1$$

$$h_i \cdot (\Pi_s - \Theta_s + C_0) \leq t_i - q_i - (\Pi_s - \Theta_s)$$

$$h_i \leq \frac{t_i - q_i - (\Pi_s - \Theta_s)}{\Pi_s - \Theta_s + C_0}$$

$$h_i = \left\lfloor \frac{t_i - q_i - (\Pi_s - \Theta_s)}{\Pi_s - \Theta_s + C_0} \right\rfloor \tag{10}$$

After calculating the number of VR budget supplies $h_i$ up to the time instant $t_i$, the VR budget and period increase $I_d$ needs to be determined.

One possible value for $I_d$ is the required increase of the initial VR parameters ($\Theta_s, \Pi_s$) by $\Theta_d$. It can be derived analogous to Equation 3. In order to determine $\Theta_d$ under consideration of the VR context switch overhead (see Equation 11), the demand at each demand point is
defined as \( q_i + h_i \cdot C_0 \).

\[
\Theta_d = \max_i \left( \left( \frac{q_i + h_i \cdot C_0}{h_i} \right) - \Theta_s \right) = \max_i \left( \left( \frac{q_i}{h_i} \right) + C_0 - \Theta_s \right)
\]

(11)

But \( \Theta_d \) might provide a smaller than required increment for the initial VR parameters, leading to a shorter VR period and a higher number of VR replenishment periods \( h_i \), resulting in a higher than required demand at \( t_i \). Figure 6 illustrates this scenario for the demand point \( DP_i \) including the expected cumulative VR context switch overhead for the predetermined number of replenishment periods \( h_i \) (represented by a diamond), and the VR supply function using the initial VR parameters \((\Theta_s, \Pi_s)\).

Figure 6a shows how the two possible VR parameter increments can be determined in order to satisfy the demand requirement of the demand point \( DP_i \).

Figure 6b depicts the VR supply function if the VR parameters are increased by \( \Theta_d \). The resulting VR period is too small and therefore the demand at \( DP_i \) is affected by an additional VR replenishment period. Consequently, the demand at \( DP_i \) is higher than assumed for \( h_i \) number of replenishment periods due to the overhead of an additional VR context switch. The demand point with the higher demand is represented by the small triangle in Figure 6b.

Therefore, additionally to \( \Theta_d \), a second possible value for \( I_d \) is determined.

In summary, the accumulated VR context switch overhead at the time instant \( t_i \) of the demand point \( DP_i \) and the resulting required demand \((q_i + h_i \cdot C_0)\) is affected by the number of VR replenishment periods \( h_i \). \( h_i \) on the other hand is determined by the VR period \( \Pi_i \). Since the required demand \((q_i + h_i \cdot C_0)\) is implicitly influenced by the VR period \( \Pi_i \), it must be ensured for demand point \( DP_i \) that after the initial VR parameters \((\Theta_s, \Pi_s)\) are increased, the number of VR replenishment periods \( h_i \) is equal to the value determined by Equation 10. Figure 6c illustrates the VR supply function after the initial VR parameters \((\Theta_s, \Pi_s)\) were increased by \( \Pi_d \), ensuring that the demand point experiences \( h_i \) number of replenishment periods and the demand requirement \( q_i \) at the time instant \( t_i \) is satisfied by the VR supply.

To derive the condition for the aforementioned VR period constraint, it is assumed that the \( h_i \)-th VR replenishment period ends right after the time instant \( t_i \) (see first line in Equation 12). Using this condition, a VR period constraint \( \Pi_i \) can be determined for the demand point \( DP_i \) (see Equation 12). That means, for the demand point \( DP_i \), the value of the VR period has to be at least as large as \( \Pi_i \).

\[
\Delta + h_i \cdot \Pi_i > t_i \\
\Pi_i > \frac{t_i - \Delta}{h_i} \\
\Pi_i > \frac{t_i - 2 \cdot (\Pi_s - \Theta_s)}{h_i} \\
\Pi_i = \left\lfloor \frac{t_i - 2 \cdot (\Pi_s - \Theta_s)}{h_i} \right\rfloor + 1
\]

(12)

Given a VR period constraint \( \Pi_i \) for every demand point \( DP_i \), the second possible increment \( \Pi_d \) can be determined as specified in Equation 13.

\[
\Pi_d = \max_i (\Pi_i) - \Pi_s \\
\Pi_d = \max_i \left( \left( \frac{t_i - 2 \cdot (\Pi_s - \Theta_s)}{h_i} \right) \right) - \Pi_s
\]

(13)
Finally, the increase $I_d$ of the initial VR parameters $(\Theta_s, \Pi_s)$ is the maximum of either $\Theta_d$ or $\Pi_d$ (see Equation 14).

$$I_d = \max(0, \Theta_d, \Pi_d)$$  (14)

The optimal VR period upper bound $\Pi^u$ and the corresponding budget $\Theta^u$, with VR context switch considered in the analysis, can be determined by the Equations 15.

$$\Pi^u = \Pi_s + I_d$$
$$\Theta^u = \Theta_s + I_d$$  (15)

5.3. Algorithm

This section presents a simple 6-step algorithm to determine the bandwidth optimal VR period upper bound for a given set of demand points. The analysis and equations presented in the previous sections are used to build up Algorithm 2.

| In : | Set of demand points and $C_0$ |
| Out : | VR period upper bound $\Pi^u$ and budget $\Theta^u$ |
| 1 | Determine the demand point leading to the longest feasible VR initial delay (see Equation 7); |
| 2 | Determine the initial VR parameters $(\Theta_s, \Pi_s)$ (see Equation 8 and 9); |
| 3 | For each demand point $DP_i$, calculate the number of VR supplies $h_i$ (see Equation 10); |
| 4 | Calculate $\Theta_d$ and $\Pi_d$ (see Equation 11 and 13); |
| 5 | Determine the initial VR parameter increase $I_d$ (see Equation 14); |
| 6 | Calculate VR period upper bound $\Pi^u = \Pi_s + I_d$ and the corresponding VR budget $\Theta^u = \Theta_s + I_d$; |

Algorithm 2: Virtual resource period upper bound

6. VR period lower bound

This section provides the definition for a VR period lower bound in the presence of VR context switch overhead. We use the VR period upper bound, determined in Section 5.2, in order to calculate a lower bound on the VR period.

According to the optimality definition in this paper, the VR parameters leading to the lowest VR utilisation are considered as the optimal VR parameters. As we showed in Theorem 1, the VR utilisation monotonically increases for $\Pi \geq \Pi_u$, hence the period of the optimal VR (with the lowest utilisation) must be less than or equal $\Pi_u$. Consequently, the optimal VR utilisation $U_o$ has to be less than or equal the utilisation $U_u$ resulting from the VR period upper bound $\Pi_u$ and the corresponding minimal VR budget $\Theta_u$.

Furthermore, the maximum utilisation of the application component $U_a$ is determined by the demand point with the highest demand $q_i$ per time interval $t_i$ (see Equation 16). Consequently, the optimal VR utilisation $U_o$ cannot be less than the application component’s utilisation $U_a$. In contrast to the VR utilisation $U_a$, the application component’s utilisation $U_a$ does not include the VR context switch overhead.

$$U_o = \max_i \left( \frac{q_i}{t_i} \right)$$  (16)

Considering the bounds of the optimal VR utilisation, it can be concluded that $U_o \leq U_o \leq U_a$.

Given the VR's and application component’s utilisation, $U_u$ and $U_a$, the maximum processor utilisation of the VR context switch overhead can be determined as $\frac{C_0}{\Pi_f} = U_u - U_a$. Considering the expected maximum VR context switch execution time $C_0$, a lower bound $\Pi_f$ on the optimal VR period can be derived as shown in Equation 17.

$$\frac{C_0}{\Pi_f} = U_u - U_a$$
$$\Pi_f = \max \left( 1, \frac{C_0}{U_u - U_a} \right)$$  (17)

7. Example

This section demonstrates the application of Algorithm 2 and Equation 17 in order to derive the optimal VR period lower and upper bound from a given taskset specification.

As an example, the taskset defined in Table 1 is used for the illustration. The presented values are expressed in time ticks. Additionally, for the VR context switch overhead $C_0$, we assume a maximal execution time of 100 ticks.

Given the taskset specification (i.e. the worst-case execution time, period and deadline of tasks), for each priority level a representative demand point is determined. The time instants where the demand bound function $dbf(t)$ is examined, is presented in Table 3. For each priority level, the row with the time instant and the taskset defining the representative demand point is highlighted.

Table 3. Demand points

<table>
<thead>
<tr>
<th>Priority level</th>
<th>Task release times (t)</th>
<th>Demand $dbf(t)$</th>
<th>Demand time ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (high)</td>
<td>1300 400</td>
<td>0.3077</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1300 1200</td>
<td>0.9231</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2600 1600</td>
<td>0.6154</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3900 2000</td>
<td>0.5128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 2400</td>
<td>0.5217</td>
<td></td>
</tr>
<tr>
<td>2 (medium)</td>
<td>1300 2200</td>
<td>1.6923</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2600 2600</td>
<td>1.0000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3900 3000</td>
<td>0.7692</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4600 3400</td>
<td>0.7391</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5200 4200</td>
<td>0.8077</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6500 4600</td>
<td>0.7077</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6800 5000</td>
<td>0.7353</td>
<td></td>
</tr>
</tbody>
</table>

After analysing the taskset demand at each priority
level, the set of demand points contains the following elements: \( DP_1 = (400, 1300) \); \( DP_2 = (2000, 3900) \); \( DP_3 = (4600, 6500) \). This data serves as input into Algorithm 2.

As indicated in line 1 of Algorithm 2 the demand point \( DP_i = (q_i, t_i) \) leading to the longest feasible VR initial delay \( \Delta \) is determined. Among the three given demand points, \( DP_1 = (400, 1300) \) satisfies the condition \( \min (t_i - q_i) \). Therefore, \( DP_s = DP_1 = (q_s, t_s) = (400, 1300) \).

The following step of Algorithm 2 (see line 2) calculates the initial VR parameter values using Equation 8 and 9. The calculated initial VR period is \( \Pi_s = 900 \) and the budget is \( \Theta_s = 500 \).

Proceeding with the initial VR parameters \( (\Theta_s, \Pi_s) \), the number of VR supplies \( h_i \) is determined for each demand point (see Equation 10). The resulting \( h_i \) values for the corresponding demand points \( DP_i \) are as follows: \( h_1 = 1, h_2 = 3 \) and \( h_3 = 3 \).

With the available data, the two possible candidate values for the increase of the initial VR parameters are calculated (line 4 of Algorithm 2). Using Equation 11, \( \Theta_{d} = \max (0, 267, 1134) = 1134 \), and using Equation 13, \( \Pi_{d} = \max (400, 134, 1000) = 1000 \) is calculated.

The required increase \( I_d \) of the initial VR parameters is determined as \( \max (\Theta_d, \Pi_d) = \max (1134, 1000) = 1134 \) (line 5 of Algorithm 2).

Finally in line 6 of Algorithm 2 the optimal VR period upper bound is determined as \( \Pi^u = \Pi_s + I_d = 2034 \), and the corresponding VR budget is \( \Theta^u = \Theta_s + I_d = 1634 \).

The next step is to determine the VR period lower bound \( \Pi_l \) using Equation 16 and Equation 17.

Using Equation 16, the application component’s maximum utilisation \( U_u = 4600/6500 = 0.7077 \) is determined by the demand point \( DP_3 = (4600, 6500) \).

The maximum VR utilisation \( U_u \) is determined by the previously calculated VR period upper bound \( \Pi^u \) and the corresponding budget \( \Theta^u \), i.e. \( U_u = 1634/2034 = 0.8033 \).

Considering for this example a VR context switch execution time of 100 ticks, the VR period lower bound \( \Pi_l \) can be calculated by Equation 17 as \( \Pi_l = \max \left( 1, \frac{100}{0.8033-0.7077} \right) = 1046.46 \).

In summary, for the taskset defined in Table 1, the VR parameter pair \( (\Theta, \Pi) \), based on the optimal VR period upper bound value, is equal to \( (1634, 2034) \). Figure 2 shows that these parameters are also the optimal values for the taskset defined in Table 1 and for an assumed VR context switch execution time of 100 ticks.

Although, in the presented example the VR period upper bound and the corresponding VR budget represent the optimal VR parameters, this may not always be the case. In the case of a more exhaustive search for the optimal VR parameters, the optimal VR period is bound by the interval \( [\Pi_l, \Pi_u] = [1046, 1634] \).

Additionally, it can be observed that, as already indicated in Section 2, the optimal VR period is not necessarily smaller than the shortest task deadline or period.

8. Evaluation

Given a set of demand points, the optimal VR period upper bound and the corresponding minimal budget can be efficiently determined by Algorithm 2.

In this section we investigate the quality of the optimal VR upper bound parameters in contrast to the optimal parameters that were determined by a more exhaustive brute-force approach.

The brute-force approach calculated the minimal VR budget for VR period values in the interval bounded by the longest task period of the examined taskset and the VR context switch execution time. Due to the discrete time space, the step size between two consecutive VR period values was one. For example, if the longest task period was equal to 6800 ticks the VR period context switch execution time was 100 ticks, then the interval for VR period values was \([100, 6800] \). Based on this interval, the minimum VR budget was calculated for 6701 different VR period values. Finally, among the generated VR parameter pairs, the one resulting in the lowest processor utilisation was considered as the optimal VR parameter.

The evaluation focuses on the utilisation overhead that emerges from the use of VR parameters derived for the VR period upper bound in contrast to the use of optimal parameters. This utilisation overhead, or simply the utilisation difference between the optimal VR and the VR determined for the period upper bound, is rendered in box-and-whisker diagrams (see example in Figure 7). These diagrams show the minimum, lower quartile (25%), median, upper quartile (75%) and maximum value of the utilisation difference for scenarios described in the following paragraph.

**Figure 7. Box-and-whisker bar illustration**

The results of the empirical evaluation were obtained by generating 10000 random tasksets for each permutation of the three test-case variables:

- taskset utilisation (also referred to as Initial Target Utilisation (ITU) set to 10%, 40% and 70% of a full speed processor utilisation).
- number of tasks per taskset equal to 5, 10, 20 and 30
- context switch overhead equal to 1, 10, 20, ..., 90, 100 ticks

The combination of the aforementioned test-cases
For each taskset, the task parameters were created by applying the generation of uniformly distributed utilisation values, proposed by Bini and Buttazzo [2]. For each utilisation value, a random task period in the interval $[1, 10000]$ is generated. Given the task utilisation and period, the calculation of its assumed worst-case execution time is straight forward.

The evaluation results are depicted in Figure 8, Figure 9 and Figure 10. They show the utilisation overhead of the VR resulting from the VR period upper bound in contrast to the optimal VR, for tasksets with 10%, 40% and 70% initial target utilisation (ITU).

The results show that as the number of tasks in the application component and as the assumed context switch time increases, the utilisation overhead decreases between the optimal VR and the non-optimal VR determined by the efficient method presented in this paper.

Additionally, to express the processor utilisation used by the VR context switch, the assumed absolute value for the VR context switch execution time is compared with the optimal VR period. The resulting processor utilisation denotes the percentage of the processing time that is required for the management of the VR. Although the VR context switch time is inevitable, from the application component’s point of view this percentage of processing time can be considered as wasted amount of processor utilisation.

Since the optimal VR period depends on the values of the test-case variables and the temporal specification of the application component’s tasks, it is impossible to calculate and associate a single processor utilisation value to an absolute VR context switch execution time value. Therefore, the minimum, lower quartile (25%), median, upper quartile (75%) and maximum value of the optimal VR period is determined for the 10000 tasksets of each test-case configuration. In order to determine the processor utilisation of the VR context switch overhead, the absolute value of the VR context switch execution time is divided by each of the five aforementioned optimal VR period values. The results are rendered in box-and-whisker diagrams.

In Figure 11, Figure 12 and Figure 13 the range of the VR context switch overhead is presented for the corresponding absolute value of the VR context switch execution time. The diagrams show that as the ITU of the application component’s taskset and the VR context switch execution time increases, the smaller is the the processor utilisation consumed by the VR context switch overhead and generally the distribution of these values.

It can be further concluded that with increasing ITU the optimal VR period increases as well, hence reducing the impact of the VR context switch overhead on the processor utilisation.

In the case that the VR parameters based on the VR period upper bound do not provide sufficiently good values, an exhaustive search over the interval $[Π_l, Π_u]$ can be carried out. Similar to the box-and-whisker diagrams, Figure 14, Figure 15 and Figure 16 depict the minimum, average (represented by a point) and maximum interval length of $[Π_l, Π_u]$ observed among the test-cases that were described at the beginning of this section. The figures show that the application component’s utilisation (ITU) has the largest impact on the VR period interval. The interval in which the optimal VR period can be found increases with the application component’s ITU.

The VR context switch overhead and the number of tasks in the application component have just a small effect on the interval $[Π_l, Π_u]$. As the average values indicates, there is only a slight decrease in the VR period interval length as the VR context switch overhead and the number of tasks increase.

Figure 14, Figure 15 and Figure 16 also indicate that the interval containing the optimal VR period is significantly reduced for application components with small ITU and it increases with increasing ITU values.

For the sake of clarity, the following example shall demonstrates the information contained in the figures presented in this evaluation section.

We examine the test-case configuration for tasksets with 20 tasks, 40% ITU, and a VR context switch execution time of 60 ticks. Figure 9c shows for the 10000 tasksets that the utilisation of the VR determined by the Algorithm 2 was approximately 0%-18% higher than the utilisation of the optimal VR (i.e. smallest possible VR utilisation) with the median around 3%. Figure 12c shows that for the same test-case configuration, the VR context switch overhead consumes approximately 3%-10% of the processor time.

The VR period interval containing the optimal VR period may range from approximately 200 up to 6200 values as rendered in Figure 9. In the worst-case situation the interval length might be large, though in average the interval length for the aforementioned test-case configuration contains only approximately 1200 values.

In summary it can be stated that the presented algorithm to determine the VR period upper bound and the corresponding budget for a given taskset, provides in the majority of cases an efficient approach to calculate feasible VR parameters that can be used as means for temporal partitioning.

9. Conclusion

In this paper we showed that a lower and upper bound on the VR period can be determined efficiently, in order to narrow down the search space for bandwidth optimal VR parameters. On the other hand, if an exhaustive search for the optimal VR parameters is not feasible, the VR parameters based on the VR period upper bound still provide sufficiently good results. Our contribution can be summarised as:

- we gave a fresh view on task demand functions and their applicability to the parameter selection of vir-
tual resources,

• for a given taskset, we provided the necessary analysis in order to determine a lower and upper bound on the VR period, thus limiting the search space for optimal VR parameters,

• the evaluation showed that the VR parameters derived from the optimal VR period upper bound might not be the optimal VR parameters, but they provide sufficiently good parameters if a brute-force search is not an option even for the limited interval of VR periods.

In future we will examine the effect of non-preemptable jobs (e.g. packet transmission over networks) on the parameter selection of virtual resources.

Currently we are working on a method to derive the bandwidth optimal parameters (i.e. both budget and period) of a periodic virtual resource only from the specification of a given taskset.

Acknowledgements

This work was funded in part by the EU FRESCOR project (contract number FP6/2005/IST/5-034026).

We also thank Dr. Robert I. Davis for his comments on a previous draft version of this paper.

References


Figure 8. VR utilisation difference between optimal and upper bound parameters for 10% ITU

Figure 9. VR utilisation difference between optimal and upper bound parameters for 40% ITU
Figure 10. VR utilisation difference between optimal and upper bound parameters for 70% ITU

Figure 11. Processor utilisation consumed by the VR context switch at 10% ITU
Figure 12. Processor utilisation consumed by the VR context switch at 40% ITU

Figure 13. Processor utilisation consumed by the VR context switch at 70% ITU
Figure 14. The optimal VR period search interval at 10% ITU

Figure 15. The optimal VR period search interval at 40% ITU
Figure 16. The optimal VR period search interval at 70% ITU