HAL
open science

Worst case analysis on modulo scheduling for specialized processors systems

Abir Benabid, Claire Hanen

To cite this version:

Abir Benabid, Claire Hanen. Worst case analysis on modulo scheduling for specialized processors systems. 10ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision (ROADEF 2009), Feb 2009, Nancy, France. pp.1-12. hal-01298196

HAL Id: hal-01298196

https://hal.science/hal-01298196v1

Submitted on 13 May 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Worst case analysis on modulo scheduling for specialized processors systems

A. Benabid,[1] and C. Hanen[2]

LIP6, 4 place Jussieu, 75252 Paris cedex 05
[1]Abir.Benabid@lip6.fr,[2]Claire.Hanen@lip6.fr

Abstract. The problem of cyclic scheduling for specialized processors systems is presented and a worst case analysis of a heuristic scheduling algorithm is studied. A resource constrained cyclic scheduling problem is characterized by k, the number of types of functional units employed, m_x the maximal number of processors of the same type and δ the maximal precedence delay. The main problem is to cope with both precedence and resource constraints which make the problem $\mathcal{NP}$-complete in general. A guaranteed approach, called decomposed software pipelining, has been proposed by Gasperoni and Schwiegelshohn, followed by the retiming method by Calland, Darte and Robert to solve the problem for parallel processors. We present, in this paper, an extension of this approach to resource-constrained cyclic scheduling problems with precedence delays and we provide an approximation algorithm. Let λ and λ_{opt} be respectively the period given by the algorithm and the optimal period. We establish the bound:

$$\lambda \leq (k + 1 - \frac{1}{m_x\delta})\lambda_{opt} + (1 - \frac{1}{m_x\delta})\delta$$

Mots-Clefs. Cyclic scheduling ; Specialized processors ; Performance bound.

1 Introduction

Cyclic scheduling problems have numerous practical application in production systems [1] as well as in embeded systems [2]. Our research in this field is partically motivated by the advances in hardware technology, but our results still available for mass production systems.

Embedded architectures used for devices such as mobile, automotive and consumer electronics need high performance, low silicon implementation costs, low power consumption and rapid development to ensure minimum time-to-market. Most of todays heigh performance applications uses instruction level parallel processors such as VLIW processors [3].

VLIW architectures are mainly used for media processing in embedded devices, and instruction schedules produced by the compiler is a performance critical optimization that has a direct impact on the overall system cost and energy consumption. High-quality instruction schedules enable to reduce the operating frequency given real-time processing requirements. Most of the parallelism present in these systems is expressed in the form of loops.

In this paper, we consider a loop composed of many unit processing tasks that are to be executed a large number of times. Instruction scheduling for inner loops is also known as software pipelining [4] and can be modelised by a cyclic scheduling problem. Among the

different cyclic scheduling frameworks, modulo scheduling [5] is the most successful in production compilers. The modulo scheduling focuses on finding a periodic schedule with the minimal period λ.

The classical results of modulo scheduling apply to problems that are too limited to be of practical use in instruction scheduling in modern processors as well as in mass production problems. For example, these results assume simple precedence constraints on tasks in a schedule, instead of precedences with delays like those in pipelined processors, and focus on machine models where each operation uses one of m identical processors for its execution.

In order to model the software pipelining problem, [6] proposes an extension of the classic modulo scheduling problem to resource-constrained modulo scheduling problems with precedence delays where the resources are adapted from the renewable resource of the resource-constrained scheduling problem [7].

We define, in this paper, a special case of this problem where the processing times as well as the resource demand is unitary, and we present a guaranteed algorithm for these problems.

1.1 Problem formulation

An instance of a resource-constrained cyclic scheduling problem can be defined by:

- An architecture model defined by $\mathcal{P} = \{P_{(i,j)} \backslash 1 \leq i \leq k, 1 \leq j \leq m_i\}$, where k denotes the number of different types of processors and m_i denotes the number of type i processors. Let $m_x = \max\limits_{1 \leq r \leq k} m_r$.
- Precedence graph $G(V, E)$ where:
  - V is a set of n tasks $V = \{T_i\}_{1 \leq i \leq n}$ with unit processing time. To each task T_i is associated a binary vector $\{b_r^i\}_{1 \leq r \leq k}$ over the resource types, such that T_i uses b_r^i units of resource of type r during its execution.
  - E is a set of edges defining uniform dependence relations denoted by $T_i \xrightarrow{l_{ij}, h_{ij}} T_j$, where the delay l_{ij} and the height h_{ij} are positive integers. l_{ij} and h_{ij} model the fact that the task T_j at iteration q has to be issued at least l_{ij} time units after the start of task i in iteration $q - h_{ij}$. We denote by $\delta = \max\limits_{(T_i, T_j) \in E} l_{ij}$.

Notice that this model generalize the classical parallel processors statements (in which $k = 1$ -i.e. there is a unique type of processors) as well as typed tasks systems where each binary vector $\{b_r^i\}_{1 \leq r \leq k}$ has only one positive component. Let us illustrate the notions of tasks, iterations, latencies and heights with the following example. We will work on this example throughout the paper.

The loop has $n = 7$ tasks, each one is executed N times. N is a given parameter representing the number of iterations and can be very large. The associated precedence graph $G(V, E)$ is given in Figure 1. Values of l_{ij} and h_{ij} are displayed next to the corresponding arc. The resource request $\{b_1^i, b_2^i\}$ of each task T_i is highlighted next to the corresponding node.

for $1 \leq i \leq N$ **do**
- $(T_1) : t_1(i) = t_3(i)$
- $(T_2) : t_2(i) = t_3(i)$
- $(T_3) : t_3(i) = t_2(i-3)$
- $(T_4) : t_4(i) = t_2(i-1)$
- $(T_5) : t_5(i) = t_4(i-2) + t_2(i-1) + t_7(i)$
- $(T_6) : t_6(i) = t_5(i-1)$
- $(T_7) : t_7(i) = t_6(i-1)$

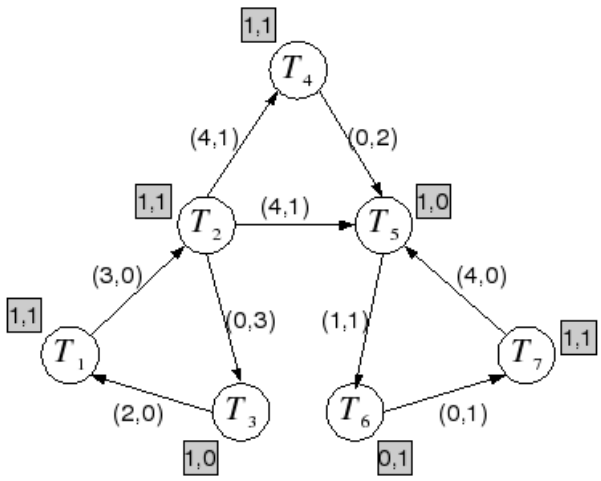

Fig. 1. An example of precedence graph $G(V, E)$.

A resource-constrained cyclic scheduling problem is to find a schedule σ that assigns an issue time $\sigma(T_i, q)$ for each task occurence (T_i, q) such that for all $r \in \{1, \cdots, k\}$ the number of tasks issued on processors of the same type r is at most equal to m_r, and

$$\left(T_i \overset{l_{ij}, h_{ij}}{\longrightarrow} T_j\right) \Rightarrow \sigma(T_i, q) + l_{ij} \leq \sigma(T_j, q + h_{ij}) \quad \forall q \in \mathbb{N}$$

The modulo scheduling focuses on finding a periodic schedule with the minimal period λ such that:

$$\forall i \in \{1, \cdots, n\}, \forall q \in \mathbb{N} : \quad \sigma(T_i, q) = \sigma(T_i, 0) + q \cdot \lambda$$

Periodic schedules are of high interest from a practical point of view, because their representation is compact so that they can be easily implemented in real systems.

1.2 Decomposed software pipelining

Generating an optimal resource constrained cyclic scheduling with minimal period is known to be $\mathcal{NP}$-hard. To overcome this $\mathcal{NP}$-hardness, we used the decomposed software pipelining approach introduced simultaneously by Gasperoni and Schwiegelsohn [9], and by Wang, Eisenbeis, Jourdan and Su [10]. The main idea is to decouple the problem into dependence constraints and resource constraints so as to decompose the problem into two subproblems: a cyclic scheduling problem ignoring resource constraints, and a standard acyclic graph for which efficient techniques are known.

Gasperoni and Schwiegelshohn give an upper bound to the period λ for the problem with m identical processors and precedences without latencies. Let λ_{opt} be the optimal (smallest) period. For unit execution tasks, this bound is given by the following inequality:

$$\lambda \leq (2 - \frac{1}{m})\lambda_{opt}$$

[11] presents a heuristic based on circuit retiming algorithms to generalize the efficiency bound given for Gasperoni-Schwiegelshohn algorithm. The main idea is to use a retiming $\mathcal{R}$ to decide which edges to cut in $G(V, E)$ so as to make it acyclic.

$$\mathcal{R} : V \rightarrow \mathbb{Z}, \quad \forall (i, j) \in E, \mathcal{R}(j) + h_{ij} - \mathcal{R}(i) \geq 0$$

A legal retiming for G of Figure 1 is given in Table 1

Table 1. A retiming $\mathcal{R}$ of G.

Tasks	T_1	T_2	T_3	T_4	T_5	T_6	T_7
$\mathcal{R}$	0	1	0	2	2	0	1

Then, we define the acyclic graph $G^{\mathcal{R}}$ by keeping only the arcs of G for which $\mathcal{R}(j) + h_{ij} - \mathcal{R}(i) = 0$. We add two pseudo-tasks with null processing times and no resource use *Start* and *Stop*:

- *Start* is a predecessor of each task with $l_{Start,i} = 0, \forall i \in E$.
- *Stop* is a successor of each task with $l_{i,Stop} = \max_{j \in Succ(i)} l_{ij}, \forall i \in E$.

Let $\pi^{\mathcal{R}}$ be any (non cyclic) schedule of $G^{\mathcal{R}}$ that fulfills the resource constraints as well as the precedences induced by $G^{\mathcal{R}}$ then, setting $\lambda^{\mathcal{R}} = \pi^{\mathcal{R}}_{Stop}$ and for any task T_i,

$$\sigma^{\mathcal{R}}(T_i, q) = \pi^{\mathcal{R}}_i + (q + \mathcal{R}(T_i))\lambda^{\mathcal{R}}$$

, we get the following result:

Lemma 1. *$\sigma^{\mathcal{R}}$ is a feasible periodic schedule of G with period $\lambda^{\mathcal{R}}$.*

Proof. ■

Now, the idea, previously used by [9] and [11] is to choose a particular retiming and use a guaranteed algorithm to get a schedule $\pi^{\mathcal{R}}$ of $G^{\mathcal{R}}$, and then to extend the guarantee to the induced periodic schedule.

List scheduling algorithms are the most used heuristics for scheduling with precedence and resource constraints. Chou, in [**?**] proves that for specialized processors with precedence delays these algorithm have the following worst case performance:

$$C_{max} \leq (k + 1 - \frac{1}{m_x(\delta)})C_{max}^{opt}$$

We thus propose the following generic algorithm 1 to solve our problem, by using a list algorithm to produce $\pi^{\mathcal{R}}$. Figures 2 and 3 illustrate the algorithm on our example.

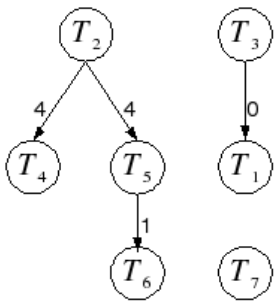

Fig. 2. The acyclic graph given by the retiming $\mathcal{R}$.

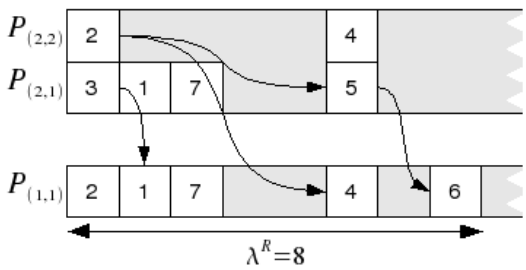

Fig. 3. A pattern generated by a list scheduling of $G^{\mathcal{R}}$: $\lambda^{\mathcal{R}} = 8$.

The acyclic graph provided by the retiming $\mathcal{R}$ is given by Figure 2 and its corresponding list scheduling allocation is presented in Figure 3. The makespan of this pattern is $\lambda^{\mathcal{R}} = 8$ and it gives the period of the modulo scheduling of G. The different steps of this heuristic are illustrated by the following algorithm.

2 Worst case analysis

In this section we analyze the worst case performance of algorithm 1 in general, making use of the proof of Chou [**?**] for list scheduling. Then, we show that using some particular retiming, that can be computed in polynomial time, we can get an overall guarantee for the Extended DSP algorithm.

2.1 Minimal length of pattern

Consider a dependence graph G. An acyclic graph $G^{\mathcal{R}}$ is obtained by a retiming $\mathcal{R}$. Then, we schedule $G^{\mathcal{R}}$ by a list algorithm and generate a pattern $\pi^{\mathcal{R}}$. We note $\phi^{\mathcal{R}}$ the length (sum of the delays) of the longest path in $G^{\mathcal{R}}$. Let λ^{opt} be the optimal period of G.

Algorithm 1: Extended DSP

1. Find a legal retiming $\mathcal{R}$ for G;
2. **for** $(T_i, T_j) \in E$ **do**
   if $\mathcal{R}(T_j) - \mathcal{R}(T_i) + h_{ij} = 0$ **then**
   keep (T_i, T_j) in $G^{\mathcal{R}}$; add nodes Start and Stop;
3. Perform a list scheduling on $G^{\mathcal{R}}$ coping with both precedence and resource constraints. Compute $\pi_i^{\mathcal{R}}$ the start time of task T_i in this schedule and $\lambda^{\mathcal{R}} = C_{max}(G^{\mathcal{R}}) = \pi_{Stop}^{\mathcal{R}}$;
4. Define the cyclic schedule $\sigma^{\mathcal{R}}$ by:
   for $1 \leq q \leq N$ **do**
   for $T_i \in V$ **do**
   $\sigma^{\mathcal{R}}(T_i, q) = \pi_i^{\mathcal{R}} + \lambda^{\mathcal{R}}(q + \mathcal{R}(T_i))$;

We consider two type of bounds obtained from resource and precedence constraints.

Resource bound :

Lemma 2. *For each type i, let u_i and m_i be respectively the number of tasks using resource type i and the number of machines of type i. Then,*

$$\lambda^{opt} \geq \max_{1 \leq i \leq k} \frac{u_i}{m_i}.$$

Proof. The shortest time required to complete the tasks of type i is $\frac{u_i}{m_i}$. Furthermore, the length of the optimal period λ^{opt} is not shorter than the time required to schedule any type of tasks. Hence, $\lambda^{opt} \geq \max_{1 \leq i \leq k} \frac{u_i}{m_i}$. ■

Precedence bounds **on schedule** $\boldsymbol{\pi^{\mathcal{R}}}$:

Let $\pi^{\mathcal{R}}$ be a schedule induced by a list algorithm on $G^{\mathcal{R}}$. In order to reveal the dependencies among tasks, we classify the time slots into three kinds:

1. A full slot t_f is a time slot in which at least all the processors of a certain type are executing tasks.
2. A partial slot t_p is a time slot in which at least one processor of each type is idle.
3. A delay slot t_d is a time slot in which all processors are idle.

We note:

- p: the number of partial slots in $\pi^{\mathcal{R}}$.
- d: the number of delay slots in $\pi^{\mathcal{R}}$.

Lemma 3. *The partial-slots lemma:*
If $\pi^{\mathcal{R}}$ contains p partial slots and d delay slots, then $\phi^{\mathcal{R}} \geq p + d$.

Proof. We prove this lemma by finding a chain $h =< T_{j_1}, \cdots, T_{j_c} >$ in $\pi^{\mathcal{R}}$ such that the length of h is at least equal to $p + d$.

Let $T_{j_c} = Stop$ and assume that we already have a chain $< T_{j_{i+1}}, \cdots, T_{v_c} >$. Consider, if it exists, the predecessor T_{v_i} of $T_{j_{i+1}}$ such that

$$\pi^{\mathcal{R}}_{j_i} + l_{j_i, j_{i+1}}$$

is maximum.

The construction of h leads to the following observation: All the slots before $\pi^{\mathcal{R}}_{j_1}$ or between $\pi^{\mathcal{R}}_{j_i} + l_{j_i, j_{i+1}}$ and $\pi^{\mathcal{R}}_{j_{i+1}}$ (if they exist) are full slots and in which there is no available processor in a type of resource used by $T_{j_{i+1}}$. Otherwise, $T_{j_{i+1}}$ would have been scheduled earlier by the list algorithm.

Therefore, all the p partial slots and d delay slots are covered by the intervals $[\pi^{\mathcal{R}}_{v_i}, \pi^{\mathcal{R}}_{j_i} + l_{j_i, j_{i+1}})$, so that the length of h not less than $p + d$. Thus, $\phi^{\mathcal{R}} \geq p + d$.

■

Lemma 4. *The delay-slots lemma:*

If $\pi^{\mathcal{R}}$ contains d delay slots, then $\phi^{\mathcal{R}} \geq d + \lceil \frac{d}{\delta - 1} \rceil$.

Proof. The schedule is computed by a list algorithm, then the number of any consecutive delay slots is not greater than $\delta - 1$. Consider the chain h defined in the previous lemma. All the delay slots are included in $\cup_{1 \leq i \leq c-1} [\pi^{\mathcal{R}}_{j_i} + 1, \pi^{\mathcal{R}}_{j_i} + l_{j_i, j_{i+1}})$, since at time $\pi^{\mathcal{R}}_{j_i}$, T_{j_i} is performed. Now the length of each interval $[\pi^{\mathcal{R}}_{j_i} + 1, \pi^{\mathcal{R}}_{j_i} + l_{j_i, j_{i+1}})$ is less than $\delta - 1$. So that $c(\delta - 1) \geq d$. The length of h is thus not less than d plus the number of the chained tasks c, which is greater than $\lceil \frac{d}{\delta - 1} \rceil$. Thus, $\phi^{\mathcal{R}} \geq d + \lceil \frac{d}{\delta - 1} \rceil$. ■

2.2 Performance bound

Here we define the notations to be used below:

$M = \sum_{1 \leq i \leq k} m_i$.

u_i: the number of tasks using resources of type i.

v_i: the number of tasks using resources of type i which are scheduled in partial slots in $\pi^{\mathcal{R}}$.

$V = \cup_{1 \leq i \leq k} v_i$: the set of tasks which are scheduled in partial slots in $\pi^{\mathcal{R}}$.

Consider the pattern $\pi^{\mathcal{R}}$:

$$M\lambda^{\mathcal{R}} = \text{number of non-idle cycles} + \text{number of idle cycles.}$$

where the second term on the right hand side is composed of the number of idle cycles associated with t_f, those associated with t_d and those associated with t_p.

1. The number of idle cycles per processor associated with t_d is equal to Md.
2. The number of idle cycles per processor associated with t_p is at most equal to $Mp - |V|$.

3. Using Lemma 2, the number of idle cycles associated with t_f is bounded by the following value:

$$\begin{aligned}
&\leq \sum_{1\leq i\leq k}(M-m_i)\frac{u_i-v_i}{m_i}\\
&\leq \sum_{1\leq i\leq k}(M-m_i)\frac{u_i}{m_i}-\sum_{1\leq i\leq k}(M-m_i)\frac{v_i}{m_i}\\
&\leq \sum_{1\leq i\leq k}(M-m_i)\max_{1\leq i\leq k}\frac{u_i}{m_i}-(M-m_x)\frac{|V|}{m_x}\\
&\leq (kM-M)\lambda^{opt}-(M-m_x)\frac{|V|}{m_x}
\end{aligned}$$

Then,

$$\begin{aligned}
M\lambda^{\mathcal{R}} &\leq M\lambda^{opt}+(kM-M)\lambda^{opt}-(M-m_x)\frac{|V|}{m_x}+Mp-|V|+Md\\
&\leq kM\lambda^{opt}-(M-m_x)\frac{|V|}{m_x}+Mp-|V|+Md\\
&\leq kM\lambda^{opt}-M\frac{|V|}{m_x}+Mp+Md
\end{aligned}$$

$|V|$ is the number of tasks scheduled in the partial slots, since each partial slot contains at least one task, $|V|\geq p$. Thus,

$$\begin{aligned}
\lambda^{\sigma} &\leq k\lambda^{opt}-\frac{p}{m_x}+p+d\\
&\leq k\lambda^{opt}+(1-\frac{1}{m_x})(p+d)+\frac{1}{m_x}d\\
&\leq k\lambda^{opt}+(1-\frac{1}{m_x})(p+d)+\frac{1}{m_x}\frac{\delta-1}{\delta}\frac{\delta}{\delta-1}d\\
&\leq k\lambda^{opt}+(1-\frac{1}{m_x})(p+d)+\frac{1}{m_x}\frac{\delta-1}{\delta}(d+\lceil\frac{d}{\delta-1}\rceil)
\end{aligned}$$

From Lemmas 3 and 4, we have $\phi^{\mathcal{R}}\geq p+d$ and $\phi^{\mathcal{R}}\geq d+\lceil\frac{d}{\delta-1}\rceil$. Then,

$$\begin{aligned}
\lambda^{\mathcal{R}} &\leq k\lambda^{opt}+(1-\frac{1}{m_x})\phi^{\mathcal{R}}+\frac{1}{m_x}\frac{\delta-1}{\delta}\phi^{\mathcal{R}}\\
&\leq k\lambda^{opt}+(1-\frac{1}{m_x\delta})\phi^{\mathcal{R}}
\end{aligned}$$

Theorem 1 *Consider a dependence graph G. Let $\mathcal{R}$ be a legal retiming $\mathcal{R}$ on G and $\phi^{\mathcal{R}}$ the length of the longest path in $G^{\mathcal{R}}$. Then,*

$$\frac{\lambda^{\mathcal{R}}}{\lambda^{opt}}\leq k+(1-\frac{1}{m_x\delta})\frac{\phi^{\mathcal{R}}}{\lambda^{opt}}.$$

2.3 Choosing a good retiming

In order to improve the performance bound, it seems important to minimize the ratio between $\phi^{\mathcal{R}}$ and λ^{opt}. So if we have a good lower bound LB of λ^{opt}, using theorem 7 in [12], we can verify the existence of a legal retiming $\mathcal{R}'$ such that $\phi^{\mathcal{R}'}\leq LB\leq\lambda^{opt}$. If such retiming exists, we have a performance guarantee of:

$$\frac{\lambda^{\mathcal{R}'}}{\lambda^{opt}} \leq k + 1 - \frac{1}{m_x \delta}.$$

An another approach consist on minimizing the length of the longest path in the pattern. There are well-known retiming algorithms [12] to minimize $\phi^{\mathcal{R}}$. Let $\mathcal{R}_{opt}$ a retiming for which the length of the longest path in the acyclic graph $G^{\mathcal{R}_{opt}}$ is minimal. We note it ϕ^{opt}. We also denote by σ^{∞} an optimal periodic schedule for unlimited resources (with period λ^{∞}).

Lemma 5.

$$\lambda^{\infty} + \delta - 1 \geq \phi^{opt}.$$

Proof. Consider an optimal cyclic schedule σ_{∞} for unlimited resources. Let us define $r : V \rightarrow [0, \lambda_{\infty} - 1]$ and $\mathcal{R} : V \rightarrow \mathbb{Z}$ such that:

$$\sigma_{\infty}(i, q) = r(i) + \lambda_{\infty}(q + \mathcal{R}(i)),\ \forall i \in V, \forall q \in \mathbb{N}$$

Then, the precedence constraint for each edge $(i, j) \in E$ is:

$$\begin{aligned} \sigma_{\infty}(i, q) + l_{ij} &\leq \sigma_{\infty}(j, q + h_{ij}) \\ r(i) + \lambda_{\infty}(q + \mathcal{R}(i)) + l_{ij} &\leq r(j) + \lambda_{\infty}(q + h_{ij} + \mathcal{R}(j)) \\ r(i) + l_{ij} &\leq r(j) + \lambda_{\infty}(h_{ij} + \mathcal{R}(j) - \mathcal{R}(i)) \end{aligned}$$

[11] proved that $\mathcal{R}$ defines a valid retiming for G. Furthermore, $G^{\mathcal{R}}$ is obtained by keeping the edges of G for which $\mathcal{R}(j) + h_{ij} - \mathcal{R}(i) = 0$. Thus ,

$$r(i) + l_{ij} \leq r(j),\ \forall (i, j) \in E$$

Let $h =< T_{j_1}, \cdots, T_{j_c}, Stop >$ be a chain in $G^{\mathcal{R}}$.

$$r(j_i) + l_{j_i, j_{i+1}} \leq r(j_{i+1}),\ \forall i \in \{1, \cdots, c-1\},$$

By summing up these $c - 1$ inequalities, we have

$$r(j_1) + \sum_{i=1}^{c-1} l_{j_i, j_{i+1}} = r(j_c)$$

Thus,

$$r(j_1) \sum_{i=1}^{c-1} l_{j_i, j_{i+1}} \leq r(j_c)$$

$$\sum_{i=1}^{c-1} l_{j_i, j_{i+1}} \leq r(j_c)$$

This inequality is true for any chain of $G^{\mathcal{R}}$ in particular for the longest path in $G^{\mathcal{R}}$. Hence,

$$\underbrace{\sum_{i=1}^{c-1} l_{j_i, j_{i+1}} + \max_{k \in Succ(T_{j_c})} l_{j_c k}}_{\phi^{\mathcal{R}}} - \max_{k \in Succ(T_{j_c})} l_{j_c k} \leq r(j_c)$$

$$\phi^{\mathcal{R}} - \delta \qquad \leq \lambda_{\infty} - 1$$

Hence, $\phi^{\mathcal{R}} - \delta \leq \lambda_\infty - 1$ and since $\phi^{opt} \leq \phi^{\mathcal{R}}$, we have the desired result.
■

Finally, using theorem 1 applied to $\mathcal{R}_{opt}$, we have

$$\begin{aligned}
\lambda^{\mathcal{R}_{opt}} &\leq k\lambda^{opt} + (1 - \frac{1}{m_x\delta})\phi^{opt} \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x\delta})(\lambda^\infty + \delta - 1) \\
&\leq k\lambda^{opt} + (1 - \frac{1}{m_x\delta})(\lambda^{opt} + \delta - 1) \\
&\leq (k + 1 - \frac{1}{m_x\delta})\lambda^{opt} + (1 - \frac{1}{m_x\delta})(\delta - 1)
\end{aligned}$$

Theorem 2

$$\lambda^{\mathcal{R}_{opt}} \leq (k + 1 - \frac{1}{m_x(\delta+1)})\lambda^{opt} + (1 - \frac{1}{m_x(\delta+1)})(\delta - 1).$$

3 Conclusion

References

1. J. M. Proth and X. Xie. *Modlisation, analyse et optimisation des systmes fonctionnement cyclique.* Masson, 1995.
2. Claire Hanen and Alix Munier. Cyclic scheduling on parallel processors: An overview. In Philippe Chrétienne, Edward G. Coffman, Jan Karel Lenstra, and Zhen Liu, editors, *Scheduling theory and its applications.* J. Wiley and sons, 1994.
3. *Proceedings of the 26th Annual International Symposium on Microarchitecture, Austin, Texas, USA, November 1993.* ACM/IEEE, 1993.
4. Vicki H. Allan, Reese B. Jones, Randall M. Lee, and Stephen J. Allan. Software pipelining. *ACM Comput. Surv.*, 27(3):367–432, 1995.
5. B. Ramakrishna Rau. Iterative modulo scheduling: an algorithm for software pipelining loops. In *MICRO 27: Proceedings of the 27th annual international symposium on Microarchitecture*, pages 63–74, New York, NY, USA, 1994. ACM.
6. C. Artigues and B.D. De Dinechin. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications.* C. Artigues, S. Demassey, E. Nron, 2008.
7. Peter Brucker, Andreas Drexl, Rolf Mohring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(1):3–41, January 1999.
8. Hong-Chich Chou and Chung-Ping Chung. Upper bound analysis of scheduling arbitrary delay instruction on typed pipelined processors. *Int. Journal of High Speed Computing*, 4(4):301–312, 1992.
9. Franco Gasperoni and Uwe Schwiegelshohn. Generating close to optimum loop schedules on parallel processors. *Parallel Processing Letters*, 4:391–403, 1994.
10. Jian Wang, Christine Eisenbeis, Martin Jourdan, and Bogong Su. Decomposed software pipelining: a new perspective and a new approach. *Int. J. Parallel Program.*, 22(3):351–373, 1994.
11. Pierre-Yves Calland, Alain Darte, and Yves Robert. Circuit retiming applied to decomposed software pipelining. *IEEE Trans. Parallel Distrib. Syst.*, 9(1):24–35, 1998.
12. C. E. Leiserson and J. B. Saxe. Retiming synchronous circuitry. *NASA STI/Recon Technical Report N*, 89:17797–+, 1988.