

Time-Indexed Formulations and a Large Neighborhood Search for the Resource-Constrained Modulo Scheduling Problem

Benoît Dupont de Dinechin

STMicroelectronics STS/CEC, 12, rue Jules Horowitz - BP 217. F-38019 Grenoble,
benoit.dupont-de-dinechin@st.com

The resource-constrained modulo scheduling problem is motivated by the 1-periodic cyclic instruction scheduling problems that are solved by compilers when optimizing inner loops for instruction-level parallel processors. In production compilers, modulo schedules are computed by heuristics, because even the most efficient integer programming formulation of resource-constrained modulo scheduling by Eichenberger and Davidson appears too expensive to solve relevant problems.

We present a new time-indexed integer programming formulation for the resource-constrained modulo scheduling problem and we propose a large neighborhood search heuristic to make it tractable. Based on experimental data from a production compiler, we show that this combination enables to solve near optimally resource-constrained modulo scheduling problems of significant size. We also show that our large neighborhood search benefits to a lesser extent the resource-constrained modulo scheduling integer programming formulation of Eichenberger and Davidson.

Keywords: Cyclic Scheduling, Modulo Scheduling, Instruction Scheduling, Time-Indexed Formulation, Large Neighborhood Search, Adaptive Margins

1 Introduction

Modulo scheduling is the cyclic instruction scheduling framework used by highly optimizing compilers to schedule instructions of innermost program loops [1]. Modulo scheduling is an effective optimization for instruction-level parallel processors [15], especially the Very Long Instruction Word (VLIW) processors that are used for media processing in embedded devices such as set-top boxes, mobile phones, and DVD players. An example of a modern VLIW architecture is the Lx [9], which provides the basis of the successful STMicroelectronics ST200 VLIW processor family.

The modulo scheduling framework is distinguished by its focus on 1-periodic cyclic schedules with integral period, which leads to simplifications compared to the classic formulation of cyclic scheduling [10]. In the modulo scheduling framework, the period is called *initiation interval* and is the main indicator of the schedule quality. A *resource-constrained modulo scheduling problem* (RCMSP) is a modulo scheduling problem where the resource constraints are adapted from the renewable resources of the resource-constrained project scheduling problem [2].

Optimal solutions to the resource-constrained modulo scheduling problem can be obtained by solving the classic integer programming formulations by Eichenberger and Davidson [8]. However, solving such formulation is only tractable for modulo scheduling problems that comprise less than several tenth of operations¹. While developing the ST200 production compiler at STMicroelectronics [6], we found that modulo scheduling heuristics appeared to loose effectiveness beyond such problem sizes, according to the lower bounds on the period obtained by relaxations.

In order to build high-quality modulo schedules for instruction scheduling problems of significant size, our contributions are as follows: we show that for any assumed period λ , the RCMSP appears

¹An operation is an instance of an instruction in a program text. An instruction is a member of the processor instruction set architecture (ISA).

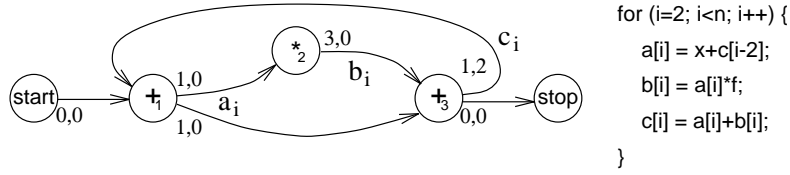


Figure 1: Sample cyclic instruction scheduling problem.

as a resource-constrained project scheduling problem with maximum time lags (RCPSp/max) and modulo resource constraints (Section 2); we present a new time-indexed integer programming formulation for the RCMSP, by adapting the time-indexed integer programming of Pritsker et al. [13] for the RCPSP/max (Section 3); we propose a large neighborhood search (LNS) heuristic for the RCMSP, based on the adaptive reduction of margins and the resolution of the resulting time-indexed integer programming formulations by implicit enumeration (Section 4).

2 The Resource-Constrained Modulo Scheduling Problem

2.1 Resource-Constrained Cyclic Scheduling Problems

A *basic cyclic scheduling problem* [10] considers a set of generic operations $\{O_i\}_{1 \leq i \leq n}$ to be executed repeatedly, thus defining a set of operation instances $\{O_i^k\}_{1 \leq i \leq n, k > 0}$, $k \in \mathbf{N}$. We call *iteration* k the set of operation instances $\{O_i^k\}_{1 \leq i \leq n}$. For any $i \in [1, n]$ and $k > 0 \in \mathbf{N}$, let σ_i^k denote the schedule date of operation instance O_i^k . Basic cyclic scheduling problems are constrained by *uniform dependences* denoted $O_i \xrightarrow{\theta_i^j, \omega_i^j} O_j$, where the *latency* θ_i^j and the *distance* ω_i^j are non-negative integers:

$$O_i \xrightarrow{\theta_i^j, \omega_i^j} O_j \implies \sigma_i^k + \theta_i^j \leq \sigma_j^{k+\omega_i^j} \quad \forall k > 0$$

In this work, we are interested in *resource-constrained cyclic scheduling problems*, whose resource constraints are adapted from the *resource-constrained project scheduling problems* (RCPSp) [2]. Precisely, we assume a set of *renewable resources*, also known as *cumulative resources*, whose availabilities are given by an integral vector \vec{B} . Each generic operation O_i requires \vec{b}_i resources for p_i consecutive units of time and this defines the resource requirements of all the operation instances $\{O_i^k\}_{k > 0}$. For the cyclic scheduling problems we consider, the cumulative use of the resources by the operation instances executing at any given time must not exceed \vec{B} .

To illustrate the resource-constrained cyclic scheduling problems that arise from instruction scheduling, consider the inner loop source code and its dependence graph displayed in Figure 1. To simplify the presentation, we did not include the memory access operations. Operation O_3 ($c[i]=a[i]+b[i]$) of iteration i must execute before operation O_1 ($a[i]=x+c[i-2]$) of iteration $i+2$ and this creates the uniform dependence with distance 2 between O_3 and O_1 (arc c_i in Figure 1). The dummy operations here labeled **start** and **stop** are introduced as source and sink of the dependence graph but have no resource requirements.

Assume this code is compiled for a microprocessor whose resources are an adder and a multiplier. The adder and the multiplier may start a new operation every unit of time. However, due to pipelined implementation, the multiplier result is only available after 3 time units. This resource-constrained cyclic scheduling problem is defined by $p_1 = p_2 = p_3 = 1$, $\vec{B} = (1, 1)^t$, $\vec{b}_1 = \vec{b}_3 = (0, 1)^t$, $\vec{b}_2 = (1, 0)^t$. The dependences are $O_1 \xrightarrow{1,0} O_2$, $O_1 \xrightarrow{1,0} O_3$, $O_2 \xrightarrow{3,0} O_3$, $O_3 \xrightarrow{1,2} O_1$.

2.2 Resource-Constrained Modulo Scheduling Problem Statement

A *modulo scheduling problem* is a cyclic scheduling problem where all operations have the same processing period $\lambda \in \mathbf{N}$, also called the *initiation interval*. Compared to cyclic scheduling problems, a main simplification is that modulo scheduling problems only need to consider the set of generic operations $\{O_i\}_{1 \leq i \leq n}$. Precisely, by introducing the *modulo schedule dates* $\{\sigma_i\}_{1 \leq i \leq n}$ such that $\forall i \in [1, n], \forall k > 0 : \sigma_i^k = \sigma_i + (k-1)\lambda$, the uniform dependence constraints become:

$$O_i \xrightarrow{\theta_i^j, \omega_i^j} O_j \implies \sigma_i^k + \theta_i^j \leq \sigma_j^{k+\omega_i^j} \implies \sigma_i + \theta_i^j - \lambda\omega_i^j \leq \sigma_j$$

Let $\{\sigma_i\}_{1 \leq i \leq n}$ denote the modulo schedule dates of a set of generic operations $\{O_i\}_{1 \leq i \leq n}$. A *resource-constrained modulo scheduling problem* (RCMSP) is defined by [6]:

- Uniform dependence constraints: for each such dependence $O_i \xrightarrow{\theta_i^j, \omega_i^j} O_j$, a valid modulo schedule satisfies $\sigma_i + \theta_i^j - \lambda\omega_i^j \leq \sigma_j$. The dependence graph without the dependences whose $\omega_i^j > 0$ is a directed acyclic graph (DAG).
- Modulo resource constraints: each operation O_i requires $\vec{b}_i \geq \vec{0}$ resources for all the time intervals $[\sigma_i + k\lambda, \sigma_i + k\lambda + p_i - 1], k \in \mathbf{Z}$ and the total resource use at any time cannot exceed a given availability \vec{B} . The integer value p_i is the processing time of O_i .

The primary objective of resource-constrained modulo scheduling problems is to minimize the period λ . The secondary objective is usually to minimize the iteration makespan. In contexts where the number of processor registers is a significant constraint, secondary objectives such as minimizing the cumulative register lifetimes [5] or the maximum register pressure [7] are considered. This is not the case for the ST200 VLIW processors, so we shall focus on makespan minimization.

2.3 Solving Resource-Constrained Modulo Scheduling Problems

Most modulo scheduling problems cannot be solved by classic machine scheduling techniques, as the modulo resource constraints introduce operation resource requirements of unbounded extent. Also, the uniform dependence graph may include circuits (directed cycles), unlike machine scheduling precedence graphs that are acyclic. Even without the circuits, some modulo dependence latencies $\theta_i^j - \lambda\omega_i^j$ may also be negative. Thus, the resource-constrained modulo scheduling problem appears as a Resource-Constrained Project Scheduling Problem with maximum time-lags (RCPSP/max) and modulo resource requirements.

In practice, building modulo schedules with heuristics is much easier than building RCPSP/max schedules. This is because the maximum time-lags of RCMSPs always include a term that is a negative factor of the period λ . Such constraints can always be made redundant by increasing enough the period λ . A similar observation holds for the modulo resource constraints.

In the classic modulo scheduling framework [14, 11, 16], a dichotomy search for the minimum λ that yields a feasible modulo schedule is performed, starting from $\lambda_{\min} \stackrel{\text{def}}{=} \max(\lambda_{\text{rec}}, \lambda_{\text{res}})$ with:

$$\lambda_{\text{rec}} \stackrel{\text{def}}{=} \max_C \left[\frac{\sum_C \theta_i^j}{\sum_C \omega_i^j} \right] : C \text{ dependence circuit}$$

$$\lambda_{\text{res}} \stackrel{\text{def}}{=} \max_{1 \leq r \leq R} \left\lceil \frac{\sum_{i=1}^n p_i b_i^r}{B^r} \right\rceil : R = \dim(\vec{B})$$

That is, λ_{rec} is the minimum λ such that there are no positive length circuits in the dependence graph and λ_{res} is the minimum λ such that the renewable resources \vec{B} are not over-subscribed.

3 Time-Indexed Integer Programming Formulations

3.1 The Non-Preemptive Time-Indexed RCPSP Formulation

Due to its ability to describe the RCPSP with general temporal constraints and other extensions, the *non-preemptive time-indexed formulation* by Pritsker, Watters and Wolfe in 1969 [13] provides the basis for many RCPSP solution strategies. In this formulation, T denotes the time horizon and $\{0, 1\}$ variables $\{x_i^t\}$ are introduced such that $x_i^t \stackrel{\text{def}}{=} 1$ if $\sigma_i = t$, else $x_i^t \stackrel{\text{def}}{=} 0$. In particular $\sigma_i = \sum_{t=1}^{T-1} t x_i^t$. The following formulation minimizes the makespan C_{max} by minimizing the completion time of a dummy operation O_{n+1} that is dependent on all the other operations:

$$\sum_{t=1}^{T-1} t x_{n+1}^t : \quad \text{minimize} \quad (1)$$

$$\sum_{t=0}^{T-1} x_i^t = 1 \quad \forall i \in [1, n+1] \quad (2)$$

$$\sum_{s=t}^{T-1} x_i^s + \sum_{s=0}^{t+\theta_i^j-1} x_j^s \leq 1 \quad \forall t \in [0, T-1], \forall (i, j) \in E_{dep} \quad (3)$$

$$\sum_{i=1}^n \sum_{s=t-p_i+1}^t x_i^s \vec{b}_i \leq \vec{B} \quad \forall t \in [0, T-1] \quad (4)$$

$$x_i^t \in \{0, 1\} \quad \forall i \in [1, n+1], \forall t \in [0, T-1] \quad (5)$$

The equations (2) ensure that any operation is scheduled once. The inequalities (3) describe the dependence constraints, as proposed by Christofides et al. [3]. Given (2), the inequalities (3) for any dependence $(i, j) \in E_{dep}$ are equivalent to: $\sum_{s=t}^{T-1} x_i^s \leq \sum_{s=t+\theta_i^j}^{T-1} x_j^s \forall t \in [0, T-1]$ and this implies $\sigma_i \leq \sigma_j - \theta_i^j$. Finally, the inequalities (4) enforce the cumulative resource constraints for $p_i \geq 1$. The extensions of the RCPSP with time-dependent resource availabilities $\vec{B}(t)$ and resource requirements $\vec{b}_i(t)$ are described in this formulation by generalizing (4) into (6):

$$\sum_{i=1}^n \sum_{s=0}^t x_i^s \vec{b}_i(t-s) \leq \vec{B}(t) \quad \forall t \in [0, T-1] \quad (6)$$

3.2 The Classic Modulo Scheduling Integer Programming Formulation

The classic integer programming formulation of the RCMSP is from Eichenberger and Davidson [8]. This formulation is based on a λ -decomposition of the modulo schedule dates, that is, $\forall i \in [0, n+1] : \sigma_i = \lambda \phi_i + \tau_i, 0 \leq \tau_i < \lambda$. Given an operation O_i , ϕ_i is its *column number* and τ_i is its *row number*. The formulation of Eichenberger and Davidson introduces the time-indexed variables $\{y_i^\tau\}_{1 \leq i \leq n}^{0 \leq \tau \leq \lambda-1}$ for the row numbers, where $y_i^t \stackrel{\text{def}}{=} 1$ if $\tau_i = t$, otherwise $y_i^t \stackrel{\text{def}}{=} 0$. In particular, $\tau_i = \sum_{\tau=0}^{\lambda-1} \tau y_i^\tau \forall i \in [1, n]$. The column numbers $\{\phi_i\}_{1 \leq i \leq n}$ are directly used in this formulation:

$$\sum_{\tau=0}^{\lambda-1} \tau y_{n+1}^\tau + \lambda \phi_{n+1} : \quad \text{minimize} \quad (7)$$

$$\sum_{\tau=0}^{\lambda-1} y_i^\tau = 1 \quad \forall i \in [1, n] \quad (8)$$

$$\sum_{s=\tau}^{\lambda-1} y_i^s + \sum_{s=0}^{(\tau+\theta_i^j-1) \bmod \lambda} y_j^s + \phi_i - \phi_j \leq \omega_i^j - \lfloor \frac{\tau+\theta_i^j-1}{\lambda} \rfloor + 1 \quad \forall \tau \in [0, \lambda-1], \forall (i, j) \in E_{dep} \quad (9)$$

$$\sum_{i=1}^n \sum_{r=0}^{p_i-1} y_i^{(\tau-r) \bmod \lambda} \vec{b}_i \leq \vec{B} \quad \forall \tau \in [0, \lambda-1] \quad (10)$$

$$y_i^\tau \in \{0, 1\} \quad \forall i \in [1, n], \forall \tau \in [0, \lambda-1] \quad (11)$$

$$\phi_i \in \mathbf{N} \quad \forall i \in [1, n] \quad (12)$$

In this formulation, λ is assumed constant. Like in classic modulo scheduling, it is solved as the inner step of a dichotomy search for the minimum value of λ that allows a feasible modulo schedule.

3.3 A New Time-Indexed Formulation for Modulo Scheduling

We propose a new time-indexed formulation for RCMSp, based on the time-indexed formulation of RCPSP/max of Pritsker et al. [13]. First consider the modulo resource constraints. Each operation O_i requires \vec{b}_i resources for the dates in $[\sigma_i + k\lambda, \sigma_i + k\lambda + p_i - 1], \forall k \in \mathbf{Z}$ (§ 2.2). The resource requirement function $\vec{b}_i(t)$ of O_i is written $\sum_{k \in \mathbf{Z}} (t \in [k\lambda, k\lambda + p_i - 1]) \vec{b}_i$, so (6) become:

$$\begin{aligned} & \sum_{i=1}^n \sum_{k \in \mathbf{Z}} \sum_{s=0}^t x_i^s (t-s \in [k\lambda, k\lambda + p_i - 1]) \vec{b}_i \leq \vec{B} \quad \forall t \in [0, T-1] \\ \Rightarrow & \sum_{i=1}^n \sum_{k \in \mathbf{Z}} \sum_{s=0}^t x_i^s (s \in [t+k\lambda - p_i + 1, t+k\lambda]) \vec{b}_i \leq \vec{B} \quad \forall t \in [0, T-1] \\ \Rightarrow & \sum_{i=1}^n \sum_{k \in \mathbf{Z}} \sum_{s=t+k\lambda-p_i+1}^{t+k\lambda} x_i^s \vec{b}_i \leq \vec{B} \quad \forall t \in [0, T-1] \\ \Rightarrow & \sum_{i=1}^n \sum_{k=0}^{\lfloor \frac{T-1}{\lambda} \rfloor} \sum_{s=t+k\lambda-p_i+1}^{t+k\lambda} x_i^s \vec{b}_i \leq \vec{B} \quad \forall t \in [0, \lambda-1] \end{aligned}$$

To complete this new RCMSp formulation, we minimize the schedule date of operation O_{n+1} and we adapt the inequalities (3) to the modulo dependence latencies of $\theta_i^j - \lambda\omega_i^j$. This yields:

$$\sum_{t=1}^{T-1} t x_{n+1}^t : \quad \text{minimize} \quad (13)$$

$$\sum_{t=0}^{T-1} x_i^t = 1 \quad \forall i \in [1, n+1] \quad (14)$$

$$\sum_{s=t}^{T-1} x_i^s + \sum_{s=0}^{t+\theta_i^j - \lambda\omega_i^j - 1} x_j^s \leq 1 \quad \forall t \in [0, T-1], \forall (i, j) \in E_{dep} \quad (15)$$

$$\sum_{i=1}^n \sum_{k=0}^{\lfloor \frac{T-1}{\lambda} \rfloor} \sum_{s=t+k\lambda-p_i+1}^{t+k\lambda} x_i^s \vec{b}_i \leq \vec{B} \quad \forall t \in [0, \lambda-1] \quad (16)$$

$$x_i^t \in \{0, 1\} \quad \forall i \in [1, n+1], \forall t \in [0, T-1] \quad (17)$$

4 A Large Neighborhood Search Heuristic

4.1 Variables and Constraints in Time-Indexed Formulations

The time-indexed formulations for the RCMSP (and the RCPSP/max) involve variables and constraints in numbers that are directly related to any assumed earliest $\{e_i\}_{1 \leq i \leq n}$ and latest $\{l_i\}_{1 \leq i \leq n}$ schedule dates. Indeed, the number of variables is $\sum_{i=1}^n l_i - e_i + 1$. Most of the constraints are the dependence inequalities (15), and $e \stackrel{\text{def}}{=} |E_{dep}|$ non transitively redundant dependences appear to generate eT inequalities with $T \stackrel{\text{def}}{=} \max_{1 \leq i \leq n} l_i + p_i$. However, the first sum of (15) is 0 if $t > l_j$. Likewise, the second sum of (15) is 0 if $t + \theta_i^j - \lambda \omega_i^j - 1 < e_j$. So (15) is actually equivalent to:

$$\sum_{s=t}^{T-1} x_i^s + \sum_{s=0}^{t+\theta_i^j-\lambda\omega_i^j-1} x_j^s \leq 1 \quad \forall t \in [e_j - \theta_i^j + \lambda\omega_i^j + 1, l_i], \forall (i, j) \in E_{dep} \quad (18)$$

So (18) is redundant whenever $e_j - \theta_i^j + \lambda\omega_i^j \geq l_i$ ($e_j - \theta_i^j \geq l_i$ in case of the RCPSP/max).

The time-indexed formulation for the RCMSP (and the RCPSP/max) is therefore likely to become more tractable after reducing the possible schedule date ranges $\sigma_i \in [e_i, l_i]$. The basic technique is to initialize $e_i = r_i$ and $l_i = d_i - p_i$, with $\{r_i\}_{1 \leq i \leq n}$ and $\{d_i\}_{1 \leq i \leq n}$ being the release dates and the due dates, then propagate the dependence constraints using a label-correcting algorithm. More elaborate techniques have been proposed for the RCPSP [4]. Ultimately, all these margins reduction techniques rely on some effective upper bounding of the due dates $\{d_i\}_{1 \leq i \leq n}$.

4.2 A Large Neighborhood Search Heuristic for Modulo Scheduling

In case of the RCMSP, the primary objective is the period minimization. Heuristics build modulo schedules at some period λ that is often greater than the lower bound $\lambda_{\min} \stackrel{\text{def}}{=} \max(\lambda_{\text{rec}}, \lambda_{\text{res}})$. When this happens, the feasibility of modulo scheduling at period $\lambda - 1$ is open. Moreover, it is not known how to bound the makespan at period $\lambda - 1$ given a makespan at period λ . This means that effective upper bounding of the due dates in RCMSP instances is not available either.

To compensate for the lack of upper bounding, we propose a large neighborhood search (LNS) for the RCMSP, based on adaptive margins reduction and implicit enumeration of the resulting time-indexed integer programs (using a MIP solver). The LNS [17] is a meta-heuristic where a large number of solutions in the neighborhood of an incumbent solution are searched by means of branch and bound, constraint programming, or integer programming. The large neighborhood is obtained by fixing some variables of the incumbent solution while releasing others.

In the setting of time-indexed formulations, we consider as the neighborhood of an incumbent solution $\{\sigma_i\}_{1 \leq i \leq n}$ some schedule date ranges or *margins* $\{e_i, l_i : \sigma_i \in [e_i, l_i]\}_{1 \leq i \leq n}$, which are made consistent under dependence constraint propagation. For each $x_i^{\sigma_i} = 1$, we fix variables $x_i^{r_i} \dots x_i^{e_i-1}$, $x_i^{l_i+1} \dots x_i^{d_i-p_i}$ to zero and release variables $x_i^{e_i} \dots x_i^{l_i}$. The fixed variables and the dependence constraints (18) found redundant given the margins are removed from the integer program.

We adapt the generic LNS algorithm of [12] by using margins to define the neighborhoods and by starting from a heuristic modulo schedule. The period λ is kept constant while this algorithm searches increasingly wider margins under a time budget in order to minimize the makespan. The key change is to replace the diversification operator of [12] by a decrement of the period to $\lambda - 1$ while keeping the schedule dates. This yields a pseudo-solution that may no longer be feasible due to the period change. Computational experience shows that a new solution at period $\lambda - 1$ can often be found in the neighborhood of this pseudo-solution, whenever the problem instance is feasible at period $\lambda - 1$. In case a solution at period $\lambda - 1$ is found, go back to minimize the makespan.

4.3 Experimental Results from the ST200 Production Compiler

We implemented the two time-indexed formulations of RCMSP described in Section 3 in the production compiler for the STMicroelectronics ST200 VLIW processor family [6], along with the proposed LNS heuristic, then used the CPLEX 9.0 Callable Library to solve the resulting MIPs.

The table below reports experimental data for the largest loops that could not be solved with these formulations, assuming a timeout of 300s and a time horizon of $4\lambda_{\min}$. Column $\#O, \#D$ gives the number of operations and of non-redundant dependences. Column *Heuristic* λ, M displays the period and makespan computed by the ST200 production compiler insertion scheduling heuristic [6]. The column groups *Formulation 300*, *Formulation 30*, *Eichenberger 300*, correspond to the proposed LNS using our formulation and Eichenberger and Davidson formulation for timeout values of 300s, 30s, 300s. In each group, column $\#V, \#C$ gives the number of variables and constraints of the integer program sent to CPLEX 9.0. In all these cases, the *Formulation* LNS reached the λ_{\min} .

Loop	#O,#D	Heuristic	Formulation 300		Formulation 30		Eichenberger 300	
		λ, M	λ, M	#V,#C	λ, M	#V,#C	λ, M	#V,#C
q-plsf.5.0_215	231,313	81,97	75,77	1114,1236	75,78	1873,2042	*,*	18942,25989
q-plsf.5.0_227	121,163	42,92	39,46	982,1228	39,46	1378,1685	39,47	4840,6673
q-plsf.5.0_201	124,168	42,92	40,47	1086,1340	40,65	1197,1421	41,50	5208,7217
q-plsf.5.2_11	233,317	82,100	75,78	1113,1216	75,79	1897,2045	*,*	19339,26637
subbands.0_196	130,234	44,65	35,49	718,906	35,48	1008,1248	*,*	5850,10778
transfo.IMDCT_L	232,370	71,109	58,58	1133,1075	58,58	1985,1961	70,74	16472,26482

5 Summary and Conclusions

The resource-constrained modulo scheduling problem (RCMSP) is a resource-constrained cyclic scheduling problem whose solutions must be 1-periodic of integral period λ . The primary minimization objective of the RCMSP is the period and the secondary objective is the makespan. Given any period λ , the RCMSP appears as a resource-constrained project scheduling project with maximum times lags (RCPSP/max) and so-called modulo resource constraints.

Based on the RCPSP/max integer programming formulation of Pritsker et al. [13] and the strong dependence equations of Christofides et al. [3], we present a new time-indexed integer programming formulation for the RCMSP. This formulation differs from the classic time-indexed integer programming formulation of the RCMSP by Eichenberger and Davidson [8].

Both formulations of the RCMSP are impractical to solve for problems that comprise over several tenths of operations, so we propose a large neighborhood search (LNS) heuristic based on solving those integer programming formulations by implicit enumeration after adapting the operation margins. Experiments show this LNS heuristic is quite effective to find a solution at period $\lambda - 1$ given an incumbent solution at period λ , even for RCMSP instances that comprise hundreds of operations. To our knowledge, this is the first application of LNS to cyclic scheduling.

References

- [1] V.H. Allan, R.B. Jones, R.M. Lee, S.J. Allan (1995), Software Pipelining, *ACM Computing Surveys* **27**(3), 367 – 432.
- [2] P. Brucker, A. Drexl, R. Möhring, K. Neumann, E. Pesch (1999), Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods, *European Journal of Operational Research* **112**.

- [3] N. Christofides, R. Alvarez-Valdés, J. M. Tamarit (1987), Project Scheduling with Resource Constraints: A Branch and Bound Approach, *European Journal of Operational Research* **29**.
- [4] S. Demassez, C. Artigues, P. Michelon (2005), Constraint-Propagation-Based Cutting-Planes: An Application to the Resource-Constrained Project Scheduling Problem, *INFORMS Journal on Computing* **17**(1).
- [5] B. Dupont de Dinechin (1994), An Introduction to Simplex Scheduling, *1994 International Conference on Parallel Architecture and Compiler Techniques – PACT’94*.
- [6] B. Dupont de Dinechin (2004), From Machine Scheduling to VLIW Instruction Scheduling, *ST Journal of Research* **1**(2), <http://www.st.com/stonline/press/magazine/stjournal/vol10102/>.
- [7] A. E. Eichenberger, E. S. Davidson, S. G. Abraham (1995), Optimum Modulo Schedules for Minimum Register Requirements, *International Conference on Supercomputing – ICS*.
- [8] A. E. Eichenberger, E. S. Davidson (1997), Efficient Formulation for Optimal Modulo Schedulers, *SIGPLAN Conference on Programming Language Design and Implementation – PLDI’97*.
- [9] P. Faraboschi, G. Brown, J. A. Fisher, G. Desoli, F. Homewood (2000), Lx: a Technology Platform for Customizable VLIW Embedded Processing, *27th Annual International Symposium on Computer Architecture – ISCA’00*.
- [10] C. Hanen, A. Munier (1995), A Study of the Cyclic Scheduling Problem on Parallel Processors, *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science* **57**.
- [11] M. Lam (1988), Software Pipelining: an Effective Scheduling Technique for VLIW Machines, *SIGPLAN Conference on Programming Language design and Implementation – PLDI’88*.
- [12] M. Palpant, C. Artigues, P. Michelon (2004), LSSPER: Solving the Resource-Constrained Project Scheduling Problem with Large Neighborhood Search, *Annals of Operations Research* **131**.
- [13] A. A. B. Pritsker, L. J. Watters, P. M. Wolfe (1969), Multi-Project Scheduling with Limited Resources: A Zero-One Programming Approach, *Management Science* **16**.
- [14] B. R. Rau, C. D. Glaeser (1981), Some Scheduling Techniques and an Easily Schedulable Horizontal Architecture for High Performance Scientific Computing, *14th Annual Workshop on Microprogramming – MICRO-14*.
- [15] B. R. Rau, J. A. Fisher (1993), Instruction-Level Parallel Processing: History, Overview, and Perspective, *Journal of Supercomputing* **7**(1-2), 9 – 50.
- [16] B. R. Rau (1996), Iterative Modulo Scheduling, *The International Journal of Parallel Processing* **24**(1).
- [17] P. Shaw (1988), Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proc. of 4th International Conference Principles and Practice of Constraint Programming – CP98*.