On integer linear programming formulations for the resource-constrained modulo scheduling problem.

Maria Ayala[1,2], Christian Artigues[1,2]

[1] CNRS; LAAS; 7 avenue du Colonel Roche, F-31077 Toulouse, France
[2] Université de Toulouse ; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
{mayala, artigues}@laas.fr

Abstract The resource-constrained modulo scheduling problem (RCMSP) is a general periodic cyclic scheduling problem, abstracted from the problem solved by compilers when optimizing inner loops at instruction level for very long instruction word parallel processors. Since solving the instruction scheduling problem at compilation phase in less time critical than for real time scheduling, integer linear programming (ILP) is a relevant technique for the RCMSP. This paper shows theoretical evidence that the two ILP formulations used by practitioners are equivalent in terms of linear programming (LP) relaxation. Stronger formulations issued from Dantzig-Wolfe decomposition are presented. All formulations are compared experimentally on problem instances generated from real data. In terms of LP relaxation, the experiments corroborates the superiority of the new formulations on problems with non binary resource requirements.

Keywords periodic scheduling, resource-constrained modulo scheduling, integer linear programming formulations, lower bounds, column generation

1 Introduction

A cyclic scheduling problem is specified by a set of generic tasks that are processed an infinity of times and a set of constraints linked to precedence relations between the tasks or to usage of limited resources. The objective is to generally maximize the throughput of scheduling. Cyclic scheduling has many applications like compiler design and parallel computing [7, 13, 15, 16, 20, 30, 29, 31, 9, 24, 21, 25] and production systems [3, 11, 8, 10, 13, 16, 20, 23, 30, 29, 31].

In this paper, we focus on the resource-constrained modulo scheduling problem (RCMSP), a general periodic cyclic scheduling problem, abstracted from the problem solved by compilers when optimizing inner loops at instruction level for VLIW parallel processors.

We briefly describe the context of the RCMSP in terms of parallel computing and very long instruction word (VLIW) architectures. We refer to [13, 34, 29, 16, 17, 15] for more details. Studies in the field of compiling for modern superscalar and VLIW architectures are mainly focused on the instruction scheduling problem. This problem is defined by a set of operations to schedule, a set of dependencies between these operations, and a target processor micro-architecture [13, 15, 34]. An operation is considered as an instance of an instruction in a program text. Instruction scheduling for inner loops is known as software pipelining. Software pipelining is an efficient method of loop

optimization that allows for parallelism of operations related to different loop iterations. Today, commercial compilers use loop pipelining methods based on modulo scheduling algorithms. Modulo scheduling is a software pipelining framework, based only on periodic cyclic schedules with integral period, which renders this model simplest than the classic model of cyclic scheduling [20]. Modulo Scheduling is a technique that links successive iterations of a loop. The goal is to find a valid schedule for a loop (the local schedule) that can be overlapped with itself infinitely. In the modulo scheduling framework, the interval between two local schedules is called initiation interval (or period) and is the main indicator of the schedule quality. The modulo scheduling algorithm must take into account the constraints of the target processor, this is, latencies of operations, resources, and size of the register files. Also, it should consider optimizing goals secondary such as, minimizing the schedule length of a loop iteration, minimizing the register requirements of the resulting modulo schedule. Algorithms based on optimal solvers have been proposed, and are referred to as optimal modulo schedulers. In this study, register constraints and objectives are ignored. In this context, The RCMSP can be informally defined as a periodic scheduling problem consisting in minimizing the period as the main objective, and the length of the local schedule as a secondary objective, while satisfying precedence and resource constraints.

Heuristic solving scheme have been proposed since many years to solve this problem [32, 18, 2, 6]. However, since solving the instruction scheduling problem at compilation phase in less time critical than for real time scheduling, integer linear programming (ILP) is a relevant technique for the RCMSP [13, 16, 17, 15]. Hence, different ILP formulations for the RCMSP have been proposed and used in practice. These formulations can be presented as generalizations of the classical non preemptive time-indexed formulations of Pritsker *et al.* [28] and the tighter variant presented by Christofides *et al.* [10], for the (non periodic) resource constrained project scheduling problem (RCPSP).

However, because of the periodic nature of the problem, the time-indexed formulation of the RCPSP can be extended in two different ways, yielding two categories of ILP formulations. Eichenberger et Davidson [16] proposed a first extension comprising both binary and integer variables. We call this category of formulations the decomposed formulations. Dupont de Dinechin [13, 14] proposed a second extension comprising only binary variables. We call the latter category the direct formulations.

In this paper, three main contributions are proposed. First, theoretical evidence that the two categories of formulations yield ILP that are equivalent in terms of linear programming (LP) relaxation is shown. Second, stronger formulations issued from Dantzig-Wolfe decomposition are presented. Last, all formulations are compared experimentally on problem instances generated from real data issued from the STMicroelectronics ST200 VLIW processor family [13].

Section 2 presents the RCMSP. Section 3 presents a solving and lower bounding framework. Section 4 presents the classical ILP formulations for the RCMSP and proves their equivalence. Section 5 presents the new formulations issued from Dantzig-Wolfe decomposition. Section 6 present a column generation method to compute the LP relaxation of the new formulations. Section 7 presents an experimental comparison of the ILP formulations in terms of LP relaxation and shows the superiority of the new formulations on instances with

tight resource constraints. Section 8 concludes the paper.

2 Problem Statement

Modulo scheduling is defined as a periodic scheduling with an integral period. We consider n generic tasks of unit duration. A schedule assigns a start time $\sigma_i^k \in \mathbb{N}$ to each k^{th} instance of generic task i with $i \in \{1, \ldots, n\}$ and $k \in \mathbb{N}$. A schedule is periodic if there exists an integer $\lambda \in \mathbb{N}^*$ verifying:

$$\sigma_i^k = \sigma_i^0 + k\lambda, \forall i \in \{1, \ldots, n\}, \forall k \in \mathbb{N} \tag{1}$$

In this case, the schedule is periodic with period λ. Given λ, the schedule is fully determined by the start time of the first instance of each task. In the remaining of the paper, we will use σ_i in place of σ_i^0. We now consider two sets of constraints on a schedule: the precedence constraints and the resource constraints.

Let E denote a set of ordered pairs of tasks, defining the precedence constraints. Each precedence constraint $(i, j) \in E$ is characterized by a latency (or length) $\theta_i^j \in \mathbb{N}$, and a distance (or height) $\omega_i^j \in \mathbb{N}$. A schedule satisfies the precedence constraint E if:

$$\sigma_j^{k+\omega_i^j} \geq \sigma_i^k + \theta_i^j \quad \forall (i, j) \in E, \forall k \in \mathbb{N} \tag{2}$$

To understand this constraint, consider that a generic task is a computer instruction inside a loop that must be repeated a large number of times, yielding the cyclic scheduling problem. Each instance k of a task i corresponds to the kth iteration of the loop. The precedence constraint may represent a dependency link between tasks i and j in such a way that an information produced by instruction i at iteration k has to be used as an input of instruction j at iteration ω_i^j iterations later. However, once instruction i is started, the information is available for j only after θ_i^j time steps. Considering the schedule is periodic, we obtain a simplified precedence constraint by inserting (1) into (2) and replacing σ_i^0 by σ_i:

$$\sigma_j \geq \sigma_i + \theta_i^j - \omega_i^j \lambda, \quad \forall (i, j) \in E \tag{3}$$

A set of m resources is considered. Each task $i \in \{1, \ldots, n\}$ requires $b_i^s \in \mathbb{N}$ units of each resource $s \in \{1, \ldots, m\}$. Each resource s has a limited availability $B_s \in \mathbb{N}$. Note that since $\lambda \geq 1$, (1) implies that several instances of the same tasks cannot be scheduled in parallel. Consequently, the set of task instances in process at a time step $t \in \mathbb{N}$ is the set $A(t) = \{i \in \{1, \ldots, n\} | \exists k \in \mathbb{N}, \sigma_i^k = t\}$. The resource constraint can be written as follows:

$$\sum_{i \in A(t)} b_i^s \leq B_s, \quad \forall s \in \{1, \ldots, m\}, \forall t \in \mathbb{N}$$

Consider now a "generic" time step $\tau \in \{0, \ldots, \lambda - 1\}$ and the set $B(\tau) = \{i \in \{1, \ldots, n\} | \exists k \in \mathbb{N}, \sigma_i = \tau + k\lambda\}$ of tasks having their start time σ_i equal to τ modulo λ. Thanks to the schedule periodicity, the resource constraint can be also simplified as it can be shown that there always exists $\mathcal{T} \in \mathbb{N}$ such that for $t \geq \mathcal{T}$, we have $A(t) = B(\tau)$ and for each $t < \mathcal{T}$ we have $A(t) \subset B(\tau)$ where

$t = \tau + k\lambda$. The resource constraints can then be replaced by the following "modulo" resource constraints:

$$\sum_{i \in B(\tau)} b_i^s \leq B_s, \quad \forall s \in \{1, \ldots, m\}, \forall \tau \in \{0, \ldots, \lambda - 1\} \tag{4}$$

The objective to be minimized is first, the period λ and, second, a function of start times of the tasks. If, w_i represents a cost associated with each task i, in general terms, the secondary goal is to minimize the weighted sum of the start times of tasks. To summarize, the RCMSP can be stated as

$$\min Lex(\lambda, \sum_{i=1}^{n} w_i \sigma_i) \tag{5}$$

subject to

$$\sigma_j \geq \sigma_i + \theta_i^j - \omega_i^j \lambda, \quad \forall (i,j) \in E \tag{3}$$

$$\sum_{i \in B(\tau)} b_i^s \leq B_s, \quad \forall s \in \{1, \ldots, m\}, \forall \tau \in \{0, \ldots, \lambda - 1\} \tag{4}$$

$$\sigma_i \in \mathbb{N} \quad \forall i \in \{1, \ldots, n\}$$

$$\lambda \in \mathbb{N} \tag{6}$$

If, we add task $n+1$, such that $\sigma_{n+1} \geq \sigma_i + 1$ with $i = 1, ..., n$, representing the schedule end and suppose $w_i = 0$ for $1 \leq i \leq n$, the objective can be a function of the completion time of the task $n+1$, this is, minimize the makespan, $C_{\max} = \sigma_{n+1}$.

3 Solving and lower bounding framework

Introducing λ as a decision variable yields non-linear mathematical programs. Hence the common framework for solving the RCMSP is based on iteratively solving the problem with a fixed λ [13, 14, 32, 16]. The minimum λ for which a feasible solution is found is the optimal period. This also allows to optimize the secondary objective at each iteration, thus solving the lexicographic optimization problem. In this paper we perform a linear search starting with a lower bound λ^{lb} which can be obtained as follows [14].

Let $N = \{1, \ldots, n\}$. When the availability of resources is not limited or sufficiently large, the minimal feasible period is given by the critical circuit in the precedence graph $G = (N, E)$. If μ is a circuit in G, the value of μ is defined by $C(\mu) = \frac{\theta(\mu)}{\omega(\mu)}$ where $\theta(\mu) = \sum_{(i,j) \in \mu} \theta_i^j$ and $\omega(\mu) = \sum_{(i,j) \in \mu} \omega_i^j$. A circuit μ is critical if its value $C(\mu)$ is maximum among all the circuits of G. The critical circuit can be found in polynomial time [20]. The lower bound of the period given by the critical circuit is

$$\lambda^{prec} = \max_{\mu \ circuit \ of \ G} C(\mu)$$

Another basic lower bound due to resource limitations can be also obtained.

$$\lambda^{res} = \max_{s=1}^{m} \frac{\sum_{i=1}^{n} b_i^s}{B_s}$$

λ^{res} is the minimum λ such that the renewable resources are not over-subscribed.

The lower bound basic lower bound to initiate the linear such for the minimal λ can be set to $\lambda^{lb} = max(\lambda^{dep}, \lambda^{res})$.

In terms of integer linear programming, let $ILP(\lambda)$ denote an integer linear program minimizing the secondary objective for a fixed λ. The optimal solution of the RCMSP is obtained by finding the smallest λ for which $ILP(\lambda)$ is feasible with $\lambda \geq \lambda^{lb}$ and by solving $ILP(\lambda)$ to optimality for the secondary objective.

This principle can be used to compute stronger lower bounds. If $LP(\lambda)$ denote the LP relaxation of $ILP(\lambda)$ the λ^{lb} lower bound can be improved by finding the smallest $\lambda \geq \lambda^{lb}$ such that $LP(\lambda)$ is feasible. This also provides a conditional lower bound for the secondary objective. The optimal solution of $LP(\lambda)$ is a lower bound for the secondary objective under the condition that the period is equal to λ. In the following section we show that the LP relaxations of the two different classical formulations for the RCMSP are in fact equal.

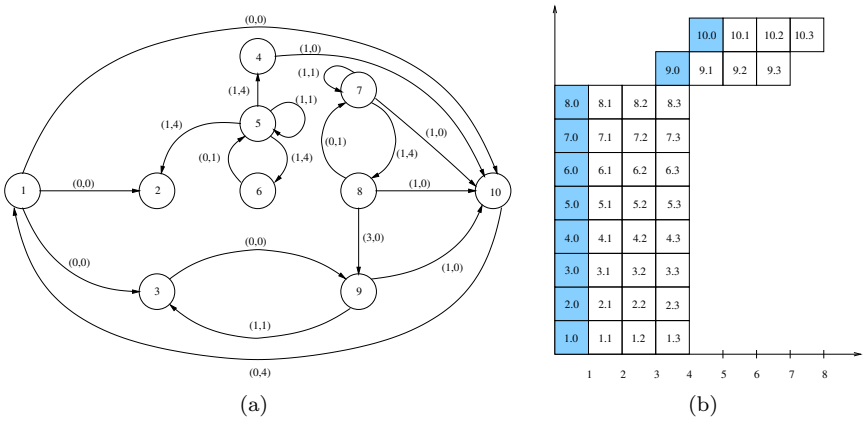

Figure 1: Precedence graph and resource-unconstrained optimal schedule

In Figure 1, a 10-task resource-unconstrained problem instance with $\lambda^{prec} = 1$ and its solution are displayed. Figure 1.a displays the precedence graph and the pair (latency,distance) for each precedence constraint. Figure 1.b displays in a Gantt chart the periodic schedule with $\lambda = 1$ and $C_{\max} = 5$, for the first 3 iterations.

Table 1 now introduces resources constraints. There are 4 resources and a particular situation where each task requirement on each resource is binary.

The optimal solution of the resource-constrained problem is displayed in Figure 2 with $\lambda = 4$ and $C_{\max} = 5$.

Resources	R_1	R_2	R_3	R_4
availability	4	1	1	2
Operations	R_1	R_2	R_3	R_4
1	1	0	0	0
2	1	1	0	0
3	1	1	0	0
4	1	1	0	0
5	1	0	0	0
6	1	1	0	0
7	1	0	0	0
8	1	0	0	0
9	1	0	1	1
10	1	0	0	0

Table 1: Resources availabilities and demands.

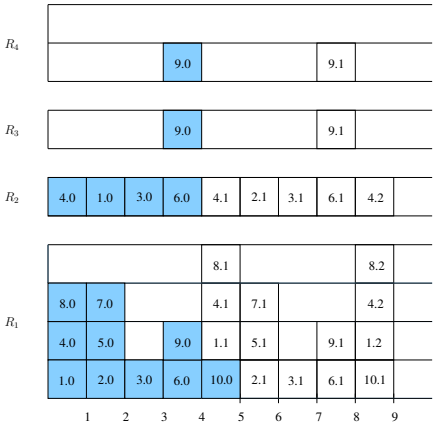

Figure 2: Optimal resource constrained modulo scheduling

4 The classical ILP formulations of the RCMSP

For the non periodic resource-constrained project scheduling (RCPSP), Pritsker *et al.* [28] proposed an ILP formulation based on time-indexed binary variables z_i^t such that $z_i^t = 1$ if and only if the start time of task i is equal to t. According to the periodic nature of the RCMSP, there are two ways for extending this model.

4.1 Direct formulation [13]

Dupont-de-Dinechin [13, 14] proposed a time-indexed formulation based on a direct discretization of start times σ_i. This formulation is based on binary variables x_i^t such that $x_i^t = 1$ if and only if $\sigma_i = t$ and we have $\sigma_i = \sum_{t=0}^{T-1} t x_i^t$, where T is any upper bound of the makespan allowing to achieve the optimum secondary objective for the optimum λ. For ease of notation, we suppose there is an integer K such that $T = K\lambda$ (we can always increase T as needed to obtain this property). This formulation (direct) is expressed as follows:

$$min \sum_{i=1}^{n} w_i (\sum_{t=0}^{T-1} t x_i^t) \tag{7}$$

$$\sum_{t=0}^{T-1} x_i^t = 1 \tag{8}$$

$$\sum_{t=0}^{T-1} t x_i^t + \theta_i^j - \lambda \omega_i^j \leq \sum_{t=0}^{T-1} t x_j^t, \forall (i,j) \in E \tag{9}$$

$$\sum_{i=1}^{n} \sum_{k=0}^{T/\lambda - 1} x_i^{\tau + k\lambda} b_i^s \leq B_s, \forall \tau \in \{0, \ldots, \lambda - 1\}, s \in \{1, \ldots, m\} \tag{10}$$

$$x_i^t \in \{0,1\} \quad \forall i \in \{1, \ldots, N\}, \forall t \in \{0, \ldots, T-1\} \tag{11}$$

Objective (7) and the precedence constraints (9) are obtained directly from (5) and (3), respectively, by replacing σ_i by $\sum_{t=0}^{T-1} t x_i^t$.

Constraints (8) state that each generic task has to be started exactly once in set $\{0, \ldots, T-1\}$. Constraints (10) ensure that the usage of a resource never exceeds its availability. Here, set $B(\tau)$ is the set of tasks such that $x_i^t = 1$ for any t such that $t = \tau + k\lambda$ with $k \in \{0, \ldots, \frac{T}{\lambda} - 1 = K - 1\}$.

Inspired by the results of Christophides *et al* for the RCPSP [10], Dupont-de-Dinechin [13, 14] introduced the following so-called "disaggregated" precedence constraints:

$$\sum_{h=t}^{T-1} x_i^h + \sum_{h=0}^{t+\theta_i^j - \lambda\omega_i^j - 1} x_j^h \leq 1, \forall t \in \{0, \ldots, T-1\}, \forall (i,j) \in E \tag{12}$$

As in the preceding case, replacing constraints (9) by constraints (12) yields a tighter formulation (direct+). The proof can be extended from the results obtained for the RCPSP (see e.g. [33]). Section 4.3 will give more details

4.2 Decomposed formulation [16]

The start time of the generic task i can be decomposed according to the division by λ. We have $\sigma_i = \tau_i + k_i\lambda$ with $\tau_i \in \{0, \ldots, \lambda - 1\}$ and $k_i \in \mathbb{N}$.

Following this decomposition, Eichenberger and Davidson [16] introduce integer variables k_i and binary variables y_i^τ such that, $y_i^\tau = 1$ if only if $\tau_i = \tau$ which yields also $\tau_i = \sum_{\tau=0}^{\lambda-1} \tau y_i^\tau$. The formulation (decomp) is expressed as follows:

$$\min \sum_{i=1}^{n} w_i (\sum_{\tau=0}^{\lambda-1} \tau y_i^\tau + k_i \lambda) \tag{13}$$

$$\sum_{\tau=0}^{\lambda-1} y_i^\tau = 1, \quad \forall i \in \{1, \ldots, n\} \tag{14}$$

$$\sum_{\tau=0}^{\lambda-1} \tau y_i^\tau + k_i\lambda + \theta_i^j - \lambda\omega_i^j \leq \sum_{\tau=0}^{\lambda-1} \tau y_j^\tau + k_j\lambda, \quad \forall (i,j) \in E \tag{15}$$

$$\sum_{i=1}^{n} y_i^\tau b_i^s \leq B_s, \quad \forall s \in \{1, \ldots, m\}, \forall \tau \in \{0, \ldots, \lambda - 1\} \tag{16}$$

$$y_i^\tau \in \{0,1\} \quad \forall i \in \{1,\dots,n\}, \forall \tau \in \{0,\dots,\lambda-1\} \tag{17}$$

$$k_i \in \{0,\dots,K-1\} \quad \forall i \in \{1,\dots,n\} \tag{18}$$

As for the direct formulation, by replacing σ_i by $\sum_{\tau=0}^{\lambda-1} \tau y_i^\tau + k_i\lambda$, objective (13) is exactly the secondary objective in (5) and constraints (15) are the precedence constraints (3). Constraints (14) state that each generic task has to be started exactly once in the period or, equivalently, that the remainder of the division of σ_i by λ lies in $\{0,\dots,\lambda-1\}$. With this decomposition, set $B(\tau)$ is precisely the set of tasks such that $y_i^\tau = 1$, which directly gives resource constraints (16) from original modulo resource constraints (4).

Based on results obtained by Chaudhuri *et al.* [7], Eichenberger and Davidson [16] propose a new precedence constraint, they call "structured" precedence constraint.

$$\sum_{x=\tau}^{\lambda-1} y_i^x + \sum_{x=0}^{(\tau+\theta_i^j-1) \bmod \lambda} y_j^x + k_i - k_j \leq \omega_i^j - \lfloor \frac{\tau+\theta_i^j-1}{\lambda} \rfloor + 1,$$
$$\forall \tau \in \{0,\dots,\lambda-1\}, \forall (i,j) \in E \tag{19}$$

Replacing constraints (15) with constraints (19) yields a tighter formulation (decomp+) (see [16] for the proof).

4.3 Comparison of LP relaxations

Although experimental results have been carried out to compare the merits of the direct and decomposed formulations in terms of integer solutions found with a commercial solver or embedded inside ILP-based heuristics [14, 15], no study has been carried-out yet to compare the quality of their LP relaxations. Theorem 1 below shows that the LP relaxations of formulations (direct) and (decomp) are equal while subsequent Theorem 2 shows that the LP relaxations of tighter formulations (direct+) and (decomp+) are also equal.

Let $T = K\lambda$, with $T, K, \lambda \in \mathbb{N}^*$. Given that $\sigma_i = \tau_i + k_i\lambda$, consider the following constraints, expressing variables y and k of the decomposed formulation as linear functions of variables x of the direct formulation.

$$y_i^\tau = \sum_{k=0}^{K-1} x_i^{\tau+k\lambda} \quad \forall i \in \{1,\dots,n\}, \forall \tau \in \{0,\dots,\lambda-1\} \tag{20}$$

$$k_i = \sum_{k=1}^{K-1} \sum_{t=k\lambda}^{K\lambda-1} x_i^t \quad \forall i \in \{1,\dots,n\} \tag{21}$$

Consider ILP (E-direct) and (E-direct+), obtained by inserting constraints (20) and (21) in (direct) and (direct+), respectively.

Given the following bound constraint for variables x_i^t.

$$0 \leq x_i^t \leq 1 \quad \forall i \in \{1,\dots,n\}, t \in \{0,\dots,T\} \tag{22}$$

we define also ILP (E-decomp) and (E-decomp+), obtained by inserting constraints (20), (21) and (22) in (decomp) and (decomp+), respectively.

With these transformations, we obtain 4 formulations all defined on the (x, y, k) space. If (form) denotes an ILP formulation, let $\tilde{z}^*$(form) denote the optimal objective value of its LP relaxation.

Lemma 1 *Considering formulations (direct), (direct+), (decomp), (decomp+) and their extended versions, we have the following equivalences:*

$$\tilde{z}^*(\textit{direct}) = \tilde{z}^*(\textit{E-direct}),$$

$$\tilde{z}^*(\textit{direct+}) = \tilde{z}^*(\textit{E-direct+}),$$

$$\tilde{z}^*(\textit{decomp}) = \tilde{z}^*(\textit{E-decomp}),$$

$$\tilde{z}^*(\textit{decomp+}) = \tilde{z}^*(\textit{E-decomp+}).$$

Proof For the direct case, we show that inclusion of variables y_i and k_i does not change the optimal LP relaxation are there are no constraints on these variables. For the decomposed case, we show that for any feasible fractional (y, k) solution respecting convexity constraints on variable y, we can always find a feasible fractional x such that $0 \leq x \leq 1$. Detailed proof is given in Appendix I. ∎

Theorem 1 *Let $\tilde{z}^*(\textit{decomp})$ denote the optimal objective value of the decomposed formulation LP relaxation and $\tilde{z}^*(\textit{direct})$ denote the optimal objective value of the direct formulation LP relaxation. We have*

$$\tilde{z}^*(\textit{direct}) = \tilde{z}^*(\textit{decomp})$$

Proof Lemma 4.3 allows us to show this equivalence by comparing the LP relaxation of the extended direct and decomposed formulations. We show that all the constraints of these formulations are equivalent. The proof is detailed in Appendix II. ∎

Last we show the equivalence of the tighter formulations (direct+) and (decomp+).

Theorem 2 *Let $\tilde{z}^*(\textit{decomp+})$ denote the optimal objective value of the improved decomposed formulation LP relaxation and $\tilde{z}^*(\textit{direct+})$ denote the optimal objective value of the improved direct formulation LP relaxation. We have*

$$\tilde{z}^*(\textit{direct+}) = \tilde{z}^*(\textit{decomp+})$$

Proof Lemma 4.3 allows us to show this equivalence by comparing the LP relaxation of the improved direct (direct+) and extended improved decomposed (E-decomp+) formulation. The difference with the previous case lies in the precedence constraints. We show that the optimal solution of the LP relaxation ignoring the resource constraints is 0-1 in both cases. The proof is detailed in Appendix III. ∎

5 Stronger formulations for the RCMSP

5.1 New formulations based on Dantzig-Wolfe decomposition for integer programs

We propose new formulations for the RCMSP based on the Dantzig-Wolfe decomposition [12] applied to integer programs as proposed in [36].

Consider the following bounded integer polyhedron $\mathcal{P}(\tau)$, corresponding to the feasible integer solutions of the resource constraints of the decomposed formulation for a given time $\tau \in \{0, \ldots, \lambda - 1\}$

$$\mathcal{P}(\tau) = \left\{ y^\tau \in \{0,1\}^n \mid \sum_{i=1}^{n} y_i^\tau b_i^s \leq B_s, \forall s \in \{1, \ldots, m\} \right\}$$

there is a one-to-one correspondence between the set of non zero feasible points of $\mathcal{P}(\tau)$ and the set R of sets of tasks $\{i_1, \ldots, i_q\}$ such that $\sum_{p=1}^{q} b_{i_p}^s \leq B_s, \forall s \in \{1, \ldots, m\}$. Each set of R is called a feasible set. The empty set correspond to $y = 0$. Let R^* denote the set of all feasible sets augmented by the empty set. Hence there are at most 2^n elements in R^*.

As $\mathcal{P}(\tau)$ is bounded, let P_0, P_1, ..., $P_{|R|-1}$ denote its elements, each corresponding to a feasible set, P_0 corresponding to the empty set. $\mathcal{P}(\tau)$ can be alternatively represented by the enumeration of its elements as follows:

$$\mathcal{P}(\tau) = \left\{ y^\tau \in \mathbb{R}^n \mid y^\tau = \sum_{l \in R^*} z_l^\tau P_l \text{ and } \sum_{l \in R^*} z_l^\tau = 1 \text{ and } z_l^\tau \in \{0,1\}, l \in R^* \right\}$$

We introduce binary matrix a such that $a_i^l = 1$ if i belongs to the feasible set corresponding to P_l (a_i^l is the ith component of P_l). Now, by replacing y_i^τ by $\sum_{l \in R} a_i^l z_l^\tau$ and using R instead of R^* (we do not consider the empty feasible set), we obtain a new decomposed formulation for the RCMSP (M-decomp).

$$min \sum_{i=1}^{n} w_i (k_i * \lambda + \sum_{\tau=0}^{\lambda-1} \tau \sum_{l \in R} a_i^l z_l^\tau)$$

$$\sum_{l \in R} (\sum_{\tau=0}^{\lambda-1} a_i^l z_l^\tau) = 1 \quad i \in \{1, ..., n\} \tag{23}$$

$$\sum_{l \in R} z_l^\tau \leq 1, \tau \in \{0, \ldots, \lambda - 1\} \tag{24}$$

$$\sum_{\tau=0}^{\lambda-1} \tau (\sum_{l \in R} a_j^l z_l^\tau - \sum_{l \in R} a_i^l z_l^\tau) + (k_j - k_j)\lambda \geq \theta_i^j - \omega_i^j \lambda . (i,j) \in E \tag{25}$$

$$z_i^\tau \in \{0,1\} \quad \forall i \in \{1, \ldots, n\}, \forall \tau \in \{0, \ldots, \lambda - 1\}$$

$$k_i \in \{0, \ldots, K-1\} \quad \forall i \in \{1, \ldots, n\}$$

In this formulation, binary variable z_l^τ equals 1 if and only if feasible subset l is used at time τ. The LP relaxation of (M-decomp) is stronger than that of

(decomp) as resource constraints are replaced by the convex hull of integer set $\mathcal{P}(\tau)$ i.e.

$$conv\left(\mathcal{P}(\tau)\right) = \left\{ y^\tau \in \mathbb{R}^n \mid y^\tau = \sum_{l \in R} z_l P_l \text{ and } \sum_{l \in R} z_l = 1 \text{ and } 0 \leq z_l \leq 1, l \in R \right\}$$

By using the structured precedence constraints (19), we may replace constraints (25) by the following constraints

$$\sum_{x=\tau}^{\lambda-1} \sum_{l \in R} a_i^l z_l^x + \sum_{x=0}^{(\tau+\theta_i^j-1) \mod \lambda} \sum_{l \in R} a_j^l z_l^x + k_i - k_j \leq \omega_i^j - \lfloor \frac{\tau + \theta_i^j - 1}{\lambda} \rfloor + 1,$$
$$\forall \tau \in \{0, \ldots, \lambda - 1\}, \forall (i,j) \in E \quad (26)$$

For the same reason the obtained formulation (M-decomp+) has a LP relaxation stronger than that of (decomp+).

We also obtain new formulations by applying the same decomposition to the direct formulations. We keep however variables x for the precedence constraints and use transformation (20) to make the link between variables z and x. We obtain the following direct formulation for the simple precedence constraints (M-direct):

$$min \sum_{i=1}^{n} w_i \sum_{t=0}^{T} t x_i^t$$

$$\sum_{t=0}^{T} x_i^t = 1, i = 1, ..., n$$

$$\sum_{l \in R} (\sum_{\tau=0}^{\lambda-1} a_i^l z_l^\tau) = 1, i = 1, ...n \quad (27)$$

$$\sum_{l \in R} z_l^\tau \leq 1, \tau \in [0, \lambda - 1] \quad (28)$$

$$\sum_{t=0}^{T} t x_i^t + \theta_i^j - \lambda \omega_i^j \leq \sum_{t=0}^{T} t x_j^t, (i,j) \in E \quad (29)$$

$$\sum_{l \in R} a_i^l z_l^\tau = \sum_{k=0}^{K-1} x_i^{\tau + k\lambda} \quad \forall i \in \{1, \ldots, n\}, \forall \tau \in \{0, \ldots, \lambda - 1\} \quad (30)$$

$$x_i^t \in \{0,1\}, i \in \{1, \ldots, n\}, t \in \{0, \ldots, T-1\}$$

$$z_i^\tau \in \{0,1\}, i \in \{1, \ldots, n\}, \tau \in \{0, \ldots, \lambda - 1\}$$

Last, by using the disaggregated precedence constraints (12) instead of the simple precedence constraints. We obtain formulation (M-direct+). As for the decomposed formulation, LP relaxations (M-direct) and (M-direct+) are stronger than the ones of (direct) and (direct+), respectively. We have to underline that applying the same decomposition for the non-periodic RCPSP to the time-indexed formulations of Pritsker *et al.* [28] gives the Mingozzi *et al* formulation [26]. The latter formulation was however proposed without mentioning the Dantzig-Wolfe decomposition applied to integer programs.

5.2 Reduction of the number of feasible sets

As previously remarked, there are at most 2^n feasible sets of tasks. If resource constraints are active, this maximum number is never reached, as all subset of tasks whose cumulative demand exceed a resource availability can be a priori discarded. For the related formulation proposed by Mingozzi *et al.* [26] for the non periodic RCPSP, further reduction is obtained by remarking that two tasks linked by a precedence constraint cannot belong to the same feasible set. Due to the cyclic nature of the RCMSP, this exclusion rule does not hold any more and more complex conditions can be derived as follows.

We establish below necessary condition for having $\sigma_i mod \lambda = \sigma_j mod \lambda$ (i.e. $\tau_i = \tau_j$). If there is a precedence constraints from i to j, we have:

$$\sigma_j \geq \sigma_i + \theta_i^j - \lambda\omega_i^j$$

$$\tau_j + k_j\lambda \geq \tau_i + k_i\lambda + \theta_i^j - \lambda\omega_i^j$$

$$\tau_j \geq \tau_i + k_i\lambda - k_j\lambda + \theta_i^j - \lambda\omega_i^j$$

$$\tau_j - \tau_i \geq (k_i - k_j)\lambda + \theta_i^j - \lambda\omega_i^j$$

For having $\tau_i = \tau_i$ we must find k_i and k_j such that

$$k_j \geq k_i + \left\lceil \frac{\theta_i^j}{\lambda} \right\rceil - \omega_i^j \tag{31}$$

and k_i, k_j integer which can be easily obtained independently of other constraints.

If, in addition, if we have also a precedence constraint from j to i we have:

$$\tau_i - \tau_j \geq (k_j - k_i)\lambda + \theta_j^i - \lambda\omega_j^i$$

If $\tau_i = \tau_i$, it comes

$$k_i \geq k_j + \left\lceil \frac{\theta_j^i}{\lambda} \right\rceil - \omega_j^i \tag{32}$$

Finding k_i and k_j such that (31), (32) and k_i, k_j integer is possible if and only if

$$\left\lceil \frac{\theta_i^j}{\lambda} \right\rceil - \omega_i^j + \left\lceil \frac{\theta_j^i}{\lambda} \right\rceil - \omega_j^i \leq 0 \tag{33}$$

On the other hand, if we admit that the problem is precedence feasible, we have

$$\theta_i^j - \lambda\omega_i^j + \theta_j^i - \lambda\omega_j^i \leq 0 \tag{34}$$

A simple counterexample shows that in general (34) $\not\Rightarrow$ (33). Suppose that $\theta_i^j = 1$, $\omega_i^j = 0$, $\theta_j^i = 1$, $\omega_j^i = 1$ and $\lambda = 2$. (34) is verified since $\theta_i^j - \lambda\omega_i^j + \theta_j^i - \lambda\omega_j^i = 1 - 0 + 1 - 2 = 0$ and a feasible solution is given for instance by $\sigma_i = 0$ and $\sigma_j = 1$ but there is no solution satisfying (31) and (32) since $\left\lceil \frac{\theta_i^j}{\lambda} \right\rceil - \omega_i^j + \left\lceil \frac{\theta_j^i}{\lambda} \right\rceil - \omega_j^i = \left\lceil \frac{1}{2} \right\rceil - 0 + \left\lceil \frac{1}{2} \right\rceil - 1 > 0$.

Hence a necessary condition for having $\tau_i = \tau_j$ is given by (33) and if the condition is not verified i and j cannot belong to the same feasible subset. This

rule can be applied to reduce a priori the number of feasible sets of tasks to be considered in set R. It actually can be extended, following the same principle, to any circuit in the precedence graph. Let (i_1, i_2), (i_2, i_3), ..., $(i_q, i_{q+1} = i_1)$ denote such a circuit.

Theorem 3 *Subset of tasks* $\{i_1, \ldots, i_q\}$ *cannot be included in a feasible set if* $\sum_{s=1}^{q} \left\lceil \frac{\theta_{i_s}^{i_s+1}}{\lambda} \right\rceil - \omega_{i_s}^{j_s+1} > 0$

Proof The theorem follows from a trivial generalization of the 2 tasks case. ∎

6 Column generation

6.1 Principle

In formulations (M-direct), (M-direct+), (M-decomp) and (M-decomp+), the number of variables can be huge ($\lambda \times \mid R_l \mid$). For this reason we propose a column generation scheme to solve their LP relaxations. The column generation method is a decomposition technique for solving a structured linear program with a lot of columns. Generally, the column generation in mainly based upon decomposing the original linear program into a master problem and a subproblem. the master problem contains a first subset of columns (it is henceforth called the restricted master problem) and the subproblem, which is a separation problem for the dual linear program, is solved to identify whether the master problem should be enlarged with additional columns or not. The column generation alternates between the master problem and the subproblem, until the former contains all the columns that are necessary for reaching an optimal solution of the original linear program.

In formulation (M-decomp), let ρ_i denote the dual variable associated with constraints (23), let β_τ denote the dual variable associated with constraints (24) and let δ_{ij} denote the dual variable associated with constraints (25).

The dual constraints related to the variable z_l^τ for primal formulation (M-decomp) are

$$\sum_{i=1}^{n} a_i^l \sigma_i - \beta_\tau + \sum_{(i,j)\in E} \delta_{ij} \tau (a_j^l - a_i^l) \leq \sum_{i=1}^{n} w_i \tau a_i^l, \forall l \in R, \tau \in \{0, \ldots, \lambda - 1\}$$

which gives

$$\sum_{i=1}^{n} w_1(i, l, \tau) a_i^l \leq \beta_\tau, \forall l \in R, \tau \in \{0, \ldots, \lambda - 1\}$$

where $w_1(i, l, \tau) = \rho_i + \tau(\sum_{(j,i)\in E} \delta_{ji} - \sum_{(i,j)\in E} \delta_{ij} + w_i)$.

Formulation (M-decomp+) differs from (M-decomp) through the disaggregated precedence constraints (26). Let δ_{ij}^τ denote the dual variable of constraints (26). We obtain the following dual constraints for variable z_l^τ in the primal formulation (M-decomp+):

$$\sum_{i=1}^{n} w_2(i, l, \tau) a_i^l \leq \beta_\tau, \forall l \in R, \tau \in \{0, \ldots, \lambda - 1\}$$

where $w_2(i,l,\tau) = \rho_i + \sum_{(j,i)\in E} \left((\tau + \theta_j^i - 1) \mod \lambda + 1\right) \delta_{ji}^{\tau} + \sum_{(i,j)\in E}(\lambda - \tau)\delta_{ij}^{\tau} + w_i\tau$.

Formulation (M-direct) differs from (M-decomp) through the fact that the precedence constraints and the objective function do not involve variables z_l^{τ} while there are linking constraints (30) between variables x and z. Let γ_i^{τ} denote the dual variables of constraints (30). We obtain the following dual constraints for variables variable z_l^{τ} of the primal formulation (M-direct):

$$\sum_{i=1}^{n} w_3(i,l,\tau)a_i^l \leq \beta_{\tau}, \forall l \in R, \tau \in \{0,\ldots,\lambda-1\}$$

where $w_3(i,l,\tau) = \rho_i + \gamma_i^{\tau}$.

As the disaggregated precedence constraints does not involve z variables, the dual constraints is identical for formulation (M-direct+).

Depending of the formulation, finding a violated dual constraint, i.e. a n-vector a_i such that $\sum_{i=1}^{n} w_x(i,\tau)a_i^l \leq \beta_{\tau}$ (for $q = 1,2,3$) can be done by solving the following subproblem ($SP_q(\tau)$)

$$max \sum_{i=1}^{n} w_q(i,\tau)a_i \tag{35}$$

$$\sum_{i=1}^{n} a_i b_i^s \leq B_s, \forall s \in \{1,\ldots,m\} \tag{36}$$

$$a_i \in \{0,1\}, \quad \forall i \in \{1,\ldots,n\} \tag{37}$$

The subproblem corresponds to a Multi-dimensional Knapsack Problem. For $\tau = 0,...,\lambda-1$ we have λ subproblems.

Once the restricted master problem is solved, the λ subproblems are solved. As soon as for a given τ, the optimal subproblem value exceeds β_{τ} the corresponding a_i generates a new feasible set and the new variable z_l^{τ} is included in the master problem. The new master problem is solved and the process is iterated until no violated dual constraint can be found for all $\tau \in \{0,\ldots,\lambda-1\}$

Note the rules presented in Section 5.2 can be used as the following valid clique inequalities to reduce the search space of the sub problem.

$$\sum_{i\in U} a_i \leq |U| - 1, \quad \forall U \in \mathcal{U}$$

where $\mathcal{U}$ is the set of set of tasks $U = \{i_1,\ldots,i_q\}$ verifying $\sum_{s=1}^{q} \left\lceil \frac{\theta_{i_s}^{i_s+1}}{\lambda} \right\rceil - \omega_{i_s}^{j_s+1} > 0$ as stated in Theorem 3

6.2 Initial set of columns

The initial restricted master problem must have a feasible LP relaxation to ensure that proper dual information is passed to the pricing problem. An initial restricted master can always be found using a two-phase method as the two-phase method incorporated in simplex algorithms to find an initial basic feasible solution [1]. In order to determine an initial feasible restricted master problem (i.e. an initial feasible set of columns), we add λ artificial variables called C_{τ}, $\tau \in$

$[0, \ldots, \lambda - 1]$ and we allow violation of the convexity constraints (24), replacing them by

$$\sum_{l \in R} z_l^\tau - C_\tau \leq 1, \tau \in [0, \lambda - 1] \tag{38}$$

Furthermore, we change the objective by:

$$min \sum_{\tau=0}^{\lambda-1} C_\tau \tag{39}$$

We initiate the set of columns by feasible sets $\{\{i\} | \sum_{s=1}^{m} b_i^s > 0\}$. We start the column generation process with the modified objective until $\sum_{\tau=0}^{\lambda-1} C_\tau = 0$. Then we turn to phase 2, by using the regular objective and convexity constraints. Note that phase 1 of the column generation scheme may end with a positive objective, which early proves the unfeasibility of λ.

7 Experimental Results

The purpose of the experimental experiments is mainly to assess the quality of the proposed new formulation for the RCMSP in terms of LP relaxation and lower bound computation for the period and for the makespan.

For the experimentations on the ILP formulations for the RCMSP, we used Cplex V.11 and an Intel(R) Core(TM)2 Duo CPU E4400 @ 2.00GHz 1.99 GHz RAM. We consider two set of instances.

The first set corresponds to a set of 36 instruction scheduling instances for the ST200 processor supplied by STmicroelectronics [13], where the smallest instance has 10 tasks and 42 dependence constraints and the biggest one has 214 tasks and 1063 constraints.

In these industrial instances, there are task classes and each task belong to a task class, defining its resource demands. In Table 2 we displayed the resources availabilities and the resource requirements of each operation class. The resources are: Issue for the instruction issue width; MEM for the memory access unit; and CTL for the control unit. An artificial resource LANE0 is also introduced to satisfy some encoding constraints. There are in our example, 10 classes of operations. ALU, MUL, MEM and CTL correspond respectively to the arithmetic, multiply, memory and control operations. The classes ALUX, MULX and MEMX represent the operations that require an extended immediate operand. LDH, MULL, ADD, CMPNE and BRF belong respectively to classes MEM, MUL, ALU and CTL.

Note the resources consumption is mainly binary, except for the "ISSUE" resource. To obtain a more constrained (in terms of resource constraints) set of RCMSP instances, the second set corresponds to the same instance set as the first set but, in this case, we have replaced the resources consumption by random consumption chosen between 1 and 10, while each resource availability has been set to 10. Table 3 (Appendix IV), provide for each instance, the number of tasks #task, the number of dependence constraints #prec, the precedence-based lower bound λ^{prec}, the resource-based lower bound for the industrial instances λ^{res} (ind) and the resource-based lower bound for the modified instances λ^{res}

Resource	ISSUE	MEM	CTL	ODD	EVEN	LANE0
Availability	4	1	1	2	2	2
ALL	4	0	0	0	0	0
ALU	1	0	0	0	0	0
ALUX	2	0	0	1	1	0
CTL	1	0	1	1	1	0
ODD	1	0	0	1	0	0
ODDX	2	0	0	1	1	0
MEM	1	1	0	0	0	0
MEMX	2	1	0	1	1	0
PSW	1	1	1	1	1	0
EVEN	1	0	0	0	1	0

Table 2: Resources availabilities and operation requirements for ST200 instances

(mod). One observes the resource constraints are much tighter for the modified instances.

We now evaluate the performance of the standard (direct+) and (decomp+) formulations on the industrial and modified instances. We first provide the results of these formulations in terms of lower bounds for the period and the makespan. The lower bound for the period is set to the smallest λ for which the LP is feasible. The lower bound for the makespan is the optimal LP objective value. As we established the equivalence between (direct+) and (decomp+), Tables 4 and 5 (Appendix IV) compare only the CPU time to obtain the lower bound, on the industrial and modified instances, respectively. The analysis of the results shows that the decomposed formulation is always faster. This was expected since it has much less variables and constraints. However, although high CPU time requirements are obtained on some instances, the lower bound is always equal to the trivial lower bound $\max(\lambda^{prec}, \lambda^{res})$ for both the industrial and modified instances.

Since the LP-based lower bounds for the period do not improve on the trivial ones, it has to be determined if this is due to the fact that the trivial bound is actually feasible or to the weakness of the LP relaxations. To this purpose, we search for the integer optimal solution of the (direct+) and (decomp+) formulations. Starting with the lower bound on the period, the branch-and-bound of the ILP solver is used, incrementing the period until, first, a feasible solution is found which yields the optimal period. Solving the last ILP to optimality then provides the optimal makespan. It appears in Table 6 (Appendix IV) that almost all industrial instances can be solved to optimality, both in terms of minimal period and minimal makespan. Again the (decomp+) formulation is faster than the (direct+) formulation. Furthermore, for the industrial instances, the minimal period is always equal to the trivial lower bound. It follows that the industrial instances cannot be used to evaluate the quality of the (direct+) and (decomp+) LP relaxations in terms of lower bounds for the period. This not the case for the makespan. Actually, as the makespan cannot be lower than the period, (direct+) and (decomp+) LP relaxations improve upon this lower bound as shown in Table 6 (Appendix IV). We underline this has not been the case for formulations (direct) and (decomp) with the weaker precedence constraints (these results are not displayed), which shows the interest of (direct+) and (decomp+) LP relaxation for lower bpunds of the makespan.

Table 7 (Appendix IV) shows the optimal of best known integer solutions obtained by (direct+) and (decomp+) on the modified instances. It appears

that the modified instances are harder to solve as less optimal solutions are found. There are 6 instances whose optimal solution is not found after three weeks of time run. We have two instances for which, after three weeks of time run, a feasible but not optimal solution is found (gap values 1.75% and 8%). However, for the optimally solved instances, the minimal period is larger than the trivial lower bound. We can thus conclude that, unfortunately, the standard LP relaxations of the RCMSP formulations are useless for computing lower bounds for the period (again this is not the case for the makespan).

Table 8 (Appendix IV) recalls the results of the standard formulations and the known optimal solutions for the modified instances. The results of the best new formulations (M-decomp+) in terms of lower bounds of the priod and makespan are provided. The trivial lower bound of the period is significantly improved as the proposed column generation-base lower bound reaches the optimal period for 82% of the instances. The CPU times are significantly increased compared to the standard ILP. However there is still room for improvement as, to demonstrate the intrinsic quality of the proposed formulations, we used a basic implementation of column generation and we did not use the rules proposed in Section 5.2.

8 Summary and Conclusions

We have proposed new formulations for the resource-constrained modulo scheduling problem, based on the integer Dantzig-Wolfe decomposition of existing time-indexed formulations. We have shown that the LP relaxation of the two previously proposed standard formulations are equal. Furthermore, the carried out computational experiments on industrial and modified instance problems show that these formulations are not able to improve upon trivial precedence and resource lower bounds of the period. On the contrary, the proposed formulations yield significantly improved and often near optimal lower bounds, when solved with column generation techniques.

Further work will consist in accelerating the column generation method, for example y improving subproblem solving thorugh the rules proposed in Section Section 5.2. This would allow an integration inside a branch-and-price or heuristic scheme.

Acknowledgement

The authors warmly thank Benoit Dupont De Dinechin for his great help to get the ST200 instances and for fruitful discussions on ILP formulations.

References

[1] Barnhart C., Johnson E., Nemhauser G., Savelsbergh M. and Vance P. *Branch and Price: Column Generation for Solving Huge Integer Programs.* Operations Research, vol 46, pp. 316-329. 1996.

[2] Blachot F. , Dupont de Dinechin B. and Huard G. *SCAN: A Heuristic for Near-Optimal Software Pipelining*. In Euro-Par 2006 Parallel Processing, Lecture Notes in Computer Science, 4128, p. 289–298, 2006.

[3] Boussemart F, Cavory G, and Lecoutre C. *Solving the cyclic job shop scheduling problem with linear precedence constraints using CP techniques*. In IEEE International Conference on Systems, Man and Cybernetics, volume 7, 6-9. 2002.

[4] Bozic S.M. *Digital and Kalman Filtering*. Edward Arnold, London (1979).

[5] Brucker P. and Drexl A. *Resource Constrained Project Scheduling: notation, classification, models and methods*, European Journal of Operational Research 112 (1999), pp 3-14 .

[6] Calland P.-Y., Darte A., Robert Y., *Circuit retiming applied to decomposed software pipelining*, IEEE Transactions on Parallel and Distributed Systems 9(1), pp. 24–35, 1998.

[7] Chaudhuri S., Walker R. A. and Mitchell J. E., *Analyzing and exploiting the structure of the constraints in the ILP approach to the scheduling problem*, IEEE Transactions on Very Large Scale Integration Systems, vol. 2, num. 4, p. 456–471, 1994.

[8] Chretienne P. *Transient and limiting behavior of timed event graph*. RAIRO-TSI. Vol 4:127-192. 1985.

[9] Chretienne P.*On Graham's bound for cyclic scheduling, Parallel Computing*. Volume 26, Issue 9, July 2000, Pages 1163-1174.

[10] Christofides N., Alvarez-Valds R. and Tamarit J. *Project scheduling with resource constraints: a branch and bound approach.* European Journal of Operational Research, vol. 29, num. 3, pages 262–273. 1987.

[11] Cohen G, Moller P, Quadrat J-P and Viot M. *A linear system theoretic view of discrete event processes and its use for performance evaluating in manufacturing*. IEEE Transaction. on Automatic Control 30, 210-220. 1985.

[12] Dantzig G. B., Wolfe P. *Decomposition principle for linear programs*, Operations Research, 8, 101–111.

[13] Dupont de Dinechin B, *From machine scheduling to VLIW instruction scheduling*, ST Journal of Research. Vol 1(2), 2004.

[14] Dupont de Dinechin B. *Time-Indexed Formulations and a Large Neighborhood Search for the Resource-Constrained Modulo Scheduling Problem*. In P. Baptiste, G. Kendall, A. Munier-Kordon, and F. Sourd, editors, 3rd Multidisciplinary International Scheduling conference: Theory and Applications, pp. 144-151, 2007.

[15] Dupont de Dinechin Benoit, Artigues Christian, Azem Sadia, *Resource-Constrained Modulo Scheduling*, in Resource-Constrained Project Scheduling. Models, Algorithms, Extensions and Applications. Artigues Christian, Demassey Sophie and Nron Emmanuel. 2008.

[16] Eichenberger A, and Davidson E.S, *Efficient formulation for optimal modulo schedulers*, SIGPLAN - PLDI'97. 1997.

[17] Eichenberger A., Davidson E. S. and Abraham S. G. *Minimum Register Requirements for a Modulo Schedule.* Proccedings of the 27th Annual International Symposium on Microarchitecture, pp. 75–84, 1994.

[18] Gasperoni F. and Schwiegelshohn U., *Generating Close to Optimum Loop Schedules on Parallel Processors*, Parallel Processing Letters 4(4), pp. 391–403, 1994

[19] Hall N.G, Lee T.E and Posner M.E. *The Complexity of Cyclic Shop Scheduling Problems.* Journal of scheduling 5(4), 307–327. 2002.

[20] Hanen C. and Munier A. *Cyclic Scheduling on parallel Processors: on overview.* In P. Chretienne, E.G. Coffman, J.K. Lenstra, and Z.Liu, eds., Scheduling Theory and its Applications, chapter 4, 194-226. New York: John Wiley and Sons.

[21] Hanen C, Munier Kordon A. *Periodic schedules for linear precedence constraints.* Discrete Applied Mathematics 157(2): 280–291 (2009)

[22] Hoffman A.J., Kruskal J.B. *Integral Boundary Points of Convex Polyhedra*, in Kuhn, H.W.; Tucker, A.W., Linear Inequalities and Related Systems, Annals of Mathematics Studies, 38, Princeton (NJ): Princeton University Press, pp. 223246, 1956.

[23] Kubale M. and Nadolski A. *Chromatic scheduling in a cyclic open shop. European Journal of Operational Research.* 164(3) pp 585–591. 2005.

[24] Munier A. *The Basic Cyclic Scheduling Problem with Linear Precedence Constraints.* Discrete Applied Mathematics 64(3): 219–238 (1996).

[25] Munier A. *The complexity of a cyclic scheduling problem with identical machines and precedence constraints.* European Journal of Operational Research, Volume 91, Issue 3, 21 June 1996, Pages 471–480.

[26] Mingozzi A. and Maniezzo V. *An Exact Algorithm For The Resource Constrained Project Scheduling Problem Based on a New Mathematical Formulation.* Management Science 44 (1998) 714–729.

[27] Papadimitriu Christos. Steiglitz Kenneth. *Combinatorial optimization. Algorithms and Complexity.* Prentice-Hall, INC. 1982.

[28] Pritsker A., Watters L. and Wolfe P. *Multi-project scheduling with limited resources: a zero−one programming approach.* Management Science, vol. 16, pp 93–108. 1969.

[29] Rajagopalan S and Sharad Malik. *A Retargetable VLIW Compiler Framework for DSPs.* CRC Press, 2002.

[30] Ramchandani C. *Analysis of asynchronous system by timed Petri nets.* PhD thesis. Universität Osnabrück, Fachbereich Mathematk/Informatik. 2006.

[31] Ramamoorthy C.V. and HO G.S. *Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets*. IEEE Transactions on Software engineering 6, 440–449, 1980.

[32] Rau B.R. *Iterative modulo scheduling: an algorithm for software pipelining loops*, in Proceedings of the 27th annual international symposium on Microarchitecture, San Jose, California, United States , pp. 63–74, 1994.

[33] Sankaran J., Bricker D. and Juang S. *A strong fractional cutting-plane algorithm for resource-constrained project scheduling*, International Journal of Industrial Engineering 6 pp. 99–111. 1999.

[34] Smelyanskiy M., Mahlke S. and Davidson E.S. *Probabilistic Predicate - Aware Modulo Scheduling.* Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization. Palo Alto, California pp 151, ISBN:0-7695-2102-9, 2004.

[35] Tiente Hsu, Ouajdi Korbaa, Rémy Dupas and Gilles Goncalves. *Cyclic scheduling for F.M.S.: Modelling and evolutionary solving approach*. European Journal of Operational Research, 191(2), p 464–484, (2008).

[36] François Vanderbeck. *On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algoritm*. Operations research, 48(1), p. 111–128, (2000).

[37] Zdenek Hanzalek. *Cyclic Scheduling of Tasks with Unit Processing Time on Dedicated Sets of Parallel Identical Processors*. Department of Control Engineering, Faculty of Electrical Engineering Czech Technical University in Prague.

Appendix I - Proofs of Lemma 4.3

Proof of Lemma 4.3 Adding variables y, k and constraints (20) and (21) to formulations (direct) and (direct+) does not change the relaxed solution space of the formulations in terms of x variables (as there are no other constraints on k and y). Consequently, we have $\tilde{z}^*$(direct) = $\tilde{z}^*$(E-direct) and $\tilde{z}^*$(direct+) = $\tilde{z}^*$(E-direct+).

For formulations (decomp) and (decomp+), extended formulations (E-decomp) and (E-decomp+) are obtained by adding variables x, constraints (20) and (21) together with bound constraints $0 \leq x_i^t \leq 1$ (22). We have to show that this transformation does not alter in both cases the relaxed optimal solution. For this, we can remark that for any y and k such that y verifies also (14), the system of equalities (20), (21), always has a solution with $0 \leq x \leq 1$.

For any $\tau \in \{0, \ldots, \lambda - 1\}$, variable x_i^τ appears only one constraint (20), the one defining y_i^τ, so we can rewrite constraints (20) as follows:

$$x_i^\tau = \sum_{k=1}^{K-1} x_i^{\tau+k\lambda} - y_i^\tau \quad \forall i \in \{1, \ldots, n\} \forall \tau \in \{0, \lambda - 1\}$$

Rewriting $x_i^\tau \geq 0$ using this expression yields:

$$\sum_{k=1}^{K-1} x_i^{\tau+k\lambda} \geq y_i^\tau, \quad \forall i \in \{1, \ldots, n\} \forall \tau \in \{0, \lambda - 1\} \tag{40}$$

Finding a feasible x amounts to find $0 \leq x_i^t \leq 1$, $t \geq \lambda$ verifying (40) and (21). A feasible x can be obtained by the following procedure. If k_i is integer, set $x_i^t = 0$ for $t \in \{0, \ldots, k_i\lambda - 1\} \cup \{(k_i + 1)\lambda, \ldots, T - 1\}$ and set $x_i^{\tau + k_i\lambda} = y_i^\tau$, for $\tau \in \{0, \ldots, \lambda - 1\}$. This verifies constraints (40) and also (21) since

$$k_i \sum_{\tau=0}^{\lambda-1} x_i^{\tau+k_i\lambda} = k_i \sum_{\tau=0}^{\lambda-1} y_i^\tau = k_i$$

as y verifies (14).

To illustrate how x is obtained in this case, suppose that $\lambda = 3$ and $K = 4$ (yielding $T = 12$). Furthermore, let $y_{i0} = 0.2$, $y_{i1} = 0.3$, $y_{i2} = 0.5$ and $k_i = 2$. A non negative x verifying (20) and (21) is simply obtained by setting $x_{it} = 0$ for $t = 0, 1, 2, 3, 4, 5, 9, 10, 11$ and $x_{i6} = 0.2$, $x_{i7} = 0.3$, $x_{i8} = 0.5$.

If k_i is not integer (we have $k_i < K - 1$), use the above-procedure to find a feasible $\tilde{x}$ for $\tilde{k}_i = \lceil k_i \rceil$. Then we have $\sum_{\tau=0}^{\lambda-1} \tilde{x}_i^{\tau+\lceil k_i \rceil\lambda} = \lceil k_i \rceil$. Using (20), we have $y_i^\tau = \tilde{x}_i^\tau + \tilde{x}_i^{\tau+\lceil k_i \rceil\lambda}$ with $\tilde{x}_i^\tau = 0$ for all $i \in \{0, \ldots, n\}, \tau \in \{0, \ldots, \lambda - 1\}$. While keeping the equalities (20), decrease each $\tilde{x}_i^{\tau+\lceil k_i \rceil\lambda}$ by $\frac{(\lceil k_i \rceil - k_i)}{\lceil k_i \rceil} \tilde{y}_i^\tau$ and increase each $\tilde{x}_i^\tau$ by the same amount, yielding x. With this operation we strictly decrease $\lceil k_i \rceil \sum_{\tau=0}^{\lambda-1} \tilde{x}_i^{\tau+\lceil k_i \rceil\lambda}$ by $\sum_{\tau=0}^{\lambda-1} (\lceil k_i \rceil - k_i) y_i^\tau = (\lceil k_i \rceil - k_i)$. So it comes $\lceil k_i \rceil \sum_{\tau=0}^{\lambda-1} x_i^{\tau+\lceil k_i \rceil\lambda} = k_i$.

To understand this transformation, modify the preceding example so that $k_i = 2.4$. We first obtain $\tilde{x} \in [0, 1]$ verifying (20) and (21) for $\tilde{k}_i = 3$ by setting $\tilde{x}_{it} = 0$ for $t = 0, 1, 2, 3, 4, 5, 6, 7, 8$ and $\tilde{x}_{i9} = 0.2$, $\tilde{x}_{i10} = 0.3$, $\tilde{x}_{i11} = 0.5$. Now we decrease $\tilde{k}_i$ back to 2.4 by setting $x_{i0} = \frac{(3-2.4)}{3} 0.2 = 0.04$, $x_{i1} = \frac{(3-2.4)}{3} 0.3 = 0.06$, $x_{i2} = \frac{(3-2.4)}{3} 0.5 = 0.1$, $x_{i9} = 0.2 - 0.04 = 0.16$, $x_{i10} = 0.3 - 0.06 = 0.24$, $x_{i11} = 0.5 - 0.1 = 0.4$.

It follows that for any feasible (y, k) solution verifying convexity constraints on y, a solution x such that $0 \leq x \leq 1$ can be found. So it comes $\tilde{z}^*(\text{decomp}) = \tilde{z}^*(\text{E-decomp})$ and $\tilde{z}^*(\text{decomp+}) = \tilde{z}^*(\text{E-decomp+})$. ∎

Appendix II - Proofs of Theorem 1

Proof of Theorem 1 According to Lemma 4.3, we have $\tilde{z}^*(\text{decomp}) = \tilde{z}^*(\text{E-decomp})$ and $\tilde{z}^*(\text{direct}) = \tilde{z}^*(\text{E-direct})$. The extended versions of both formulation can be compared to derive the desired result. To that purpose we show that each feasible solution of (E-decomp) is feasible for (E-direct) and vice versa. This is precisely shown by the following.

- Equivalence of assignment constraints (14) and (8):
  Inserting (20) in the left-hand side of (14) we obtain :

$$\sum_{\tau=0}^{\lambda-1} \tilde{y}_i^\tau = \sum_{\tau=0}^{\lambda-1} \sum_{k=0}^{K-1} \tilde{x}_i^{\tau+k\lambda} = \sum_{t=0}^{T-1} \tilde{x}_i^t \tag{41}$$

  which is precisely the left-hand side of the assignment constraint of the direct formulation (8). It follows that any solution y satisfying (14) yields a solution x satisfying (8), and vice-versa.

- Equivalence of precedence constraints (15) and (9):
  Inserting (20) and (21) into $\sum_{\tau=0}^{\lambda-1} \tau y_i^\tau + k_i\lambda$ we obtain

$$\begin{aligned}
\sum_{\tau=0}^{\lambda-1} \tau\tilde{y}_i^\tau + \tilde{k}_i\lambda &= \sum_{\tau=0}^{\lambda-1}\tau\sum_{k=0}^{K-1}\tilde{x}_i^{\tau+k\lambda} + \sum_{k=1}^{K-1}\sum_{t=k\lambda}^{K\lambda-1}\lambda\tilde{x}_i^t \\
&= \sum_{k=0}^{K-1}\sum_{\tau=0}^{\lambda-1}\tau\tilde{x}_i^{\tau+k\lambda} + \sum_{k=1}^{K-1} k\sum_{t=k\lambda}^{(k+1)\lambda-1}\lambda\tilde{x}_i^t \\
&= \sum_{k=0}^{K-1}\sum_{\tau=0}^{\lambda-1}\tau\tilde{x}_i^{\tau+k\lambda} + \sum_{k=0}^{K-1}\sum_{\tau=0}^{\lambda-1} k\lambda\tilde{x}_i^{\tau+k\lambda} \\
&= \sum_{t=0}^{T-1} t x_i^t \qquad (42)
\end{aligned}$$

  Again, from this equivalence, any solution x verifying precedence constraints (9) yields a solution y satisfying precedence constraints (15), and reciprocally.

- Equivalence of resource constraints (16) and (10):
  Inserting (20) in the left-hand side of (16) we obtain :

$$\sum_{i=1}^{n}\tilde{y}_i^\tau b_i^s = \sum_{i=1}^{n}\sum_{k=0}^{K-1}\tilde{x}_i^{\tau+k\lambda}b_i^s \qquad (43)$$

  which is precisely the left-hand side of the resource constraint of the direct formulation (10).

- Bounds on variables y and k:
  Constraints (20) and (8) ensure that $0 \leq \tilde{y}_i^\tau \leq 1$ for all $i \in \{1,\ldots,n\}$, $\tau \in \{0,\ldots,\lambda-1\}$. Similarly (21) and (8) imply that $0 \leq \tilde{k}_i \leq K-1$, for all $i \in \{1,\ldots,n\}$

- Objective function equivalence:
  From equality (42) established above, it follows that the objective function values in the relaxed (E-direct) formulation and in the relaxed (E-decomp) formulation are the same.

From all this points it comes that (E-decomp) and (E-direct) have equal LP relaxations. According to Lemma 4.3, this holds also for (decomp) and (direct). ∎

Appendix III - Proofs of Theorem 2

Proof of Theorem 2 As for the previous case, we use Lemma 4.3 that establishes, on the one hand, that $\tilde{z}$(direct+) = $\tilde{z}$(E-direct+) and, on the other hand, that $\tilde{z}$(decomp+) = $\tilde{z}$(E-decomp+). Consequently, we compare (direct+) and (E-decomp+).

By using equalities (41) and (43) and replacing in (E-decomp+) y and k by their expression in function of x we obtain two formulations which only differ

by the so-called "disaggregated" precedence constraints and the "structured" precedence constraints (with x variables), respectively.

Ignoring the resource constraints, we first establish that every basic feasible solution of the constraints matrix corresponding to both formulation is $0-1$.

Consider first the improved decomposed formulation. we rewrite the structured precedence constraint (19) by replacing y_i and k_i with their expression in function on x_i^t (20) and (21). We consider first expression $\sum_{s=\tau}^{\lambda-1} y_i^s + k_i$. We have

$$\begin{aligned}\sum_{s=\tau}^{\lambda-1} y_i^s + k_i &= \sum_{k=0}^{K-1}\sum_{s=\tau}^{\lambda-1} x_i^{s+k\lambda} + \sum_{k=1}^{K-1}\sum_{s=k\lambda}^{K\lambda-1} x_i^s \\ &= \sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{(k+1)\lambda-1} x_i^s + \sum_{k=0}^{K-1}\sum_{s=(k+1)\lambda}^{K\lambda-1} x_i^s \\ &= \sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s\end{aligned}$$

If in the structured precedence constraint (19) we use the transformations (20) and (21) and replacing $\sum_{s=\tau}^{\lambda-1} y_i^s + k_i$ by $\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s$ as established above, then the structured precedence constraint (19) can be rewritten as:

$$\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s + \sum_{s=0}^{(\tau+\theta_i^j-1)mod\lambda}\sum_{k=0}^{K-1} x_j^{s+k\lambda} - \sum_{k=1}^{K-1}\sum_{t=k\lambda}^{K\lambda-1} x_j^t \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor + 1$$

$$\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s + \sum_{s=0}^{(\tau+\theta_i^j-1)mod\lambda}\sum_{k=0}^{K-1} x_j^{s+k\lambda} - \sum_{k=0}^{K-1}\sum_{t=k\lambda}^{K\lambda-1} x_j^t + 1 \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor + 1$$

$$\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s + \sum_{s=0}^{(\tau+\theta_i^j-1)mod\lambda}\sum_{k=0}^{K-1} x_j^{s+k\lambda} + \sum_{k=0}^{K-1}\sum_{t=0}^{k\lambda-1} x_j^t - K + 1 \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor + 1$$

$$\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s + \sum_{k=0}^{K-1}\Big(\sum_{s=0}^{(\tau+\theta_i^j-1)mod\lambda} x_j^{s+k\lambda} + \sum_{t=0}^{k\lambda-1} x_j^t\Big) - K + 1 \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor + 1$$

$$\sum_{k=0}^{K-1}\sum_{s=\tau+k\lambda}^{K\lambda-1} x_i^s + \sum_{k=0}^{K-1}\Big(\sum_{s=0}^{k\lambda+(\tau+\theta_i^j-1)mod\lambda} x_j^s\Big) - K + 1 \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor + 1$$

$$-\sum_{k=0}^{K-1}\sum_{s=0}^{s=\tau+k\lambda-1} x_i^s + \sum_{k=0}^{K-1}\sum_{s=0}^{k\lambda+(\tau+\theta_i^j-1)mod\lambda} x_j^s \le \omega_i^j - \lfloor\frac{\tau+\theta_i^j-1}{\lambda}\rfloor$$

Now we make the transformation of the variables x_i^s to variables $z_i^{\tau,k,p}$ as

$$z_i^{\tau,k,p} = \sum_{q=k}^{p} \sum_{s=0}^{\tau+q\lambda} x_i^s$$

Then the structured precedence constraint can be written

$$-z_i^{\tau-1,0,K-1} + z_j^{(\tau+\theta_i^j-1) mod \lambda,0,K-1} \leq \omega_i^j - \lfloor \frac{\tau+\theta_i^j-1}{\lambda} \rfloor \quad \forall (i,j) \in E, \forall \tau \in \{0,\ldots,\lambda-1\} \tag{44}$$

Constraints (14) becomes

$$z_i^{\lambda-1,0,K-1} - z_i^{-1,0,K-1} = 1. \forall i \in \{1,...,n\} \tag{45}$$

Bound constraints on $x_i^{\tau+k\lambda}$ can be written:

$$0 \leq z_i^{\tau,k,k} - z_i^{\tau-1,k,k} \leq 1. \forall i \in \{1,...,n\}, \forall \tau \in \{0,\ldots,\lambda\}, \forall k \in \{0,\ldots,K-1\} \tag{46}$$

The constraints matrix Z corresponding to (44),(45) and (46) has a solution $z_i^{\tau,k,p}$ with integer coordinates if and only if Z is totally unimodular, i.e, each square submatrix of Z has determinant 0, 1 or -1. A sufficient but not necessary condition [27] for the unimodularity of an integer matrix A with $a_{i,j} =\in \{0,-1,1\}$ consists in verifying that no more than two nonzero entries appear in any column, and that the rows of A can be partitioned into two sets I_1 and I_2 such that:

1. If a column has two entries of the same sign, their rows are in different sets.

2. If a column has two entries of different signs, their rows are in the same set.

Consequently constraint matrix Z is totally unimodular.

We consider now the improved direct formulation without the resource constraints. We make another non-singular transformation of the variables x_i^t to variables h_i^t as

$$h_i^t = \sum_{s=0}^{T-1} x_i^s$$

Now constraints (8) become:

$$h_i^t = 1, \forall i \in \{1,\ldots,n\}$$

constraint (12) can be rewritten:

$$h_i^t - h_j^{t+\theta_i^j-\lambda\omega_i^j-1} \geq 0, \forall (i,j) \in E, \forall t \in \{0,\ldots,T-1\}.$$

Bounds on x_i^t are rewritten:

$$0 \leq h_i^t - h_i^{t-1} \leq 1 \quad \forall i \in \{1,\ldots,n\}, \forall t \in \{0,\ldots,T-1\}$$

and

$$h_i^t \geq 0$$

Using the same sufficient conditions as above, we obtain a totally unimodular constraint matrix H.

Both the improved direct and decomposed formulation have $0-1$ solutions in terms of x variables concerning the precedence constraints. Furthermore, they have the same resource constraints. Consequently their LP relaxations are equal. ∎

Appendix IV - Experimental result tables

Table 3: Characteristics of the industrial and modified instances

Instance name	#task	#prec	λ_{prec}	λ^{res} (ind)	λ^{res} (mod)
adpcm-st231.1	86	405	11	21	52
adpcm-st231.2	142	722	38	35	82
gsm-st231.1	30	190	24	12	16
gsm-st231.2	101	462	12	26	59
gsm-st231.5	44	192	4	11	26
gsm-st231.6	30	130	3	7	17
gsm-st231.7	44	192	4	11	28
gsm-st231.8	14	66	1	8	9
gsm-st231.9	34	154	28	8	21
gsm-st231.10	10	42	1	4	6
gsm-st231.11	26	137	20	6	16
gsm-st231.12	15	70	1	8	10
gsm-st231.13	46	210	19	11	27
gsm-st231.14	39	176	5	10	20
gsm-st231.15	15	70	1	8	9
gsm-st231.16	65	323	4	16	38
gsm-st231.17	38	173	8	9	23
gsm-st231.18	214	1063	8	53	120
gsm-st231.19	19	86	2	8	12
gsm-st231.20	23	102	3	6	13
gsm-st231.21	33	154	18	8	20
gsm-st231.22	31	146	18	8	18
gsm-st231.25	60	273	10	16	37
gsm-st231.29	44	192	4	11	28
gsm-st231.30	30	130	3	7	16
gsm-st231.31	44	192	4	11	26
gsm-st231.32	32	138	15	8	21
gsm-st231.33	59	266	4	15	33
gsm-st231.34	10	42	2	4	7
gsm-st231.35	18	80	2	6	12
gsm-st231.36	31	143	2	10	18
gsm-st231.39	26	118	3	8	15
gsm-st231.40	21	103	2	10	12
gsm-st231.41	60	315	2	18	33
gsm-st231.42	23	102	3	6	14
gsm-st231.43	26	115	8	6	15

Table 4: CPU time comparison for LP-relaxation based lower bound (direct+) and (decomp+) on industrial instances

Instances	λ	$C_{\max}$	CPU (s) (decomp+)	CPU (s) (direct+)
adpcm-st231.1	21	29.5	498	536
adpcm-st231.2	38	42	5326.37	6012
gsm-st231.1	24	33	0.02	0.02
gsm-st231.2	26	31.5	586	614
gsm-st231.5	11	15.2	0.02	0.02
gsm-st231.6	7	11.2	0.02	0.02
gsm-st231.7	11	15.2	0.02	0.02
gsm-st231.8	8	8	0.02	0.02
gsm-st231.9	28	28	0.02	0.02
gsm-st231.10	4	4	0.00	0.00
gsm-st231.11	20	21	0.02	0.02
gsm-st231.12	8	8	0.00	0.00
gsm-st231.13	19	25	0.02	0.02
gsm-st231.14	10	12.63	0.02	0.02
gsm-st231.15	8	8	0.00	0.00
gsm-st231.16	16	16.97	3625.12	3812.03
gsm-st231.17	9	15.25	0.02	0.02
gsm-st231.18	53	53	7256	8002.03
gsm-st231.19	8	8	0.00	0.00
gsm-st231.20	6	10	0.00	0.00
gsm-st231.21	18	22	15	15
gsm-st231.22	18	21	17	20
gsm-st231.25	16	24.52	789.26	814
gsm-st231.29	11	15.2	7.52	7.84
gsm-st231.30	7	11.2	0.02	0.02
gsm-st231.31	11	15.2	0.02	0.02
gsm-st231.32	15	15	4.25	4.45
gsm-st231.33	15	17.23	42	42
gsm-st231.34	4	5	0.00	0.00
gsm-st231.35	6	8.2	0.00	0.00
gsm-st231.36	10	10	0.02	0.02
gsm-st231.39	8	12.5	0.02	0.02
gsm-st231.40	10	10	0.00	0.00
gsm-st231.41	18	18	12	0.02
gsm-st231.42	6	10	0.02	0.02
gsm-st231.43	8	10.4	0.00	0.00

Table 5: CPU time comparison for LP-relaxation based lower bound (direct+) and (decomp+) on modified instances

Instances	λ	Cmax	CPU (s) (decomp+)	CPU (s) (direct+)
adpcm-st231.1	52	52	1800	1995.03
adpcm-st231.2	82	82	7214	7327
gsm-st231.1	24	32	600	626
gsm-st231.2	59	59	7200	7298.04
gsm-st231.5	26	26	600	600
gsm-st231.6	17	17	600	600
gsm-st231.7	28	28	22	23
gsm-st231.8	9	9	0.02	0.02
gsm-st231.9	28	28	25	30
gsm-st231.10	6	6	0.0001	0.0001
gsm-st231.11	20	21	0.15	0.152
gsm-st231.12	10	10	0.0001	0.0001
gsm-st231.13	27	27	48	52
gsm-st231.14	20	20	18	21
gsm-st231.15	9	9	0.001	0.002
gsm-st231.16	38	38	420	452
gsm-st231.17	24	24	720	795
gsm-st231.18	120	120	600	603
gsm-st231.19	12	12	0.002	0.002
gsm-st231.20	13	13	0.002	0.002
gsm-st231.21	20	22	24	24
gsm-st231.22	18	21	8	8
gsm-st231.25	37	37	300	317
gsm-st231.29	28	28	60	62
gsm-st231.30	16	16	0.25	0.253
gsm-st231.31	26	26	58	60
gsm-st231.32	21	21	3.02	3
gsm-st231.33	33	33	52	51
gsm-st231.34	6	6	0.001	0.001
gsm-st231.35	11	11	0.002	0.002
gsm-st231.36	18	18	8	8
gsm-st231.39	15	15	0.06	0.08
gsm-st231.40	12	12	0.0001	0.0001
gsm-st231.41	34	34	47	49
gsm-st231.42	14	14	4.03	6
gsm-st231.43	15	15	0.026	0.02

Table 6: Optimal integer solutions of the industrial instances

Instances	PLNE(decomp+)			PLNE(direct+)		
	λ	Cmax	CPU_s	λ	Cmax	CPU_s
adpcm-st231.1	21	30	14400	21	30	16235
adpcm-st231.2	40	42	582362	40	42	601000
gsm-st231.1	24	33	0.05	24	33	0.05
gsm-st231.2	26	32	79362	26	32	83991
gsm-st231.5	11	17	0.05	11	17	0.05
gsm-st231.6	7	13	17	7	13	20
gsm-st231.7	11	17	0.05	11	17	0.05
gsm-st231.8	8	8	0.05	8	8	0.05
gsm-st231.9	28	28	0.05	28	28	0.05
gsm-st231.10	4	4	0.05	4	4	0.05
gsm-st231.11	20	21	0.05	20	21	0.05
gsm-st231.12	8	8	0.05	8	8	0.05
gsm-st231.13	19	25	1856	19	25	2023
gsm-st231.14	10	13	301.25	10	13	478
gsm-st231.15	8	8	0.05	8	8	0.05
gsm-st231.16	16	20	7520	16	20	8156
gsm-st231.17	9	16	0.05	9	16	0.05
gsm-st231.18	-	-	-	-	-	-
gsm-st231.19	8	9	0.05	8	9	0.05
gsm-st231.20	6	10	0.05	6	10	0.05
gsm-st231.21	18	22	0.05	18	22	0.05
gsm-st231.22	18	22	0.05	18	22	0.05
gsm-st231.25	16	25	3652	16	25	4001
gsm-st231.29	11	17	12.6	11	17	15
gsm-st231.30	7	13	12	7	13	15
gsm-st231.31	11	17	47	11	17	73
gsm-st231.32	15	15	0.05	15	15	0.05
gsm-st231.33	15	21	2365	15	21	2503
gsm-st231.34	4	5	0.05	4	5	0.05
gsm-st231.35	6	11	0.05	6	11	0.05
gsm-st231.36	10	15	27	10	15	42
gsm-st231.39	8	16	0.05	8	16	0.05
gsm-st231.40	10	10	0.05	10	10	0.05
gsm-st231.41	18	24	2356	18	24	2562
gsm-st231.42	6	10	0.05	6	10	0.05
gsm-st231.43	8	14	0.05	4	14	0.05

Table 7: Optimal of feasible integer solutions of the modified instances

Instances	PLNE(decomp+)			PLNE(direct+)		
	λ	Cmax	CPU_s	λ	Cmax	CPU_s
adpcm-st231.1	-	-	-	-	-	-
adpcm-st231.2	-	-	-	-	-	-
gsm-st231.1	25	42	250	25	42	375
gsm-st231.2	-	-	-	-	-	-
gsm-st231.5	36	46	280	36	46	299.03
gsm-st231.6	27	27	152	27	27	265
gsm-st231.7	41	45	92	41	45	115
gsm-st231.8	12	12	0.27	12	12	0.31
gsm-st231.9	32	35	0.56	32	35	60
gsm-st231.10	8	8	0.10	8	8	0.11
gsm-st231.11	24	24	0.37	24	24	0.39
gsm-st231.12	13	13	12.65	13	13	19
gsm-st231.13	43	48	985.03	43	48	1236
gsm-st231.14	33	45	220	33	45	252
gsm-st231.15	12	12	12.36	12	12	13
gsm-st231.16	-	-	-	-	-	-
gsm-st231.17	33	33	90	33	33	105
gsm-st231.18	-	-	-	-	-	-
gsm-st231.19	15	15	38.23	15	15	43
gsm-st231.20	20	27	123	20	27	137
gsm-st231.21	30	30	42.03	30	30	59
gsm-st231.22	29	29	80.36	29	29	112
gsm-st231.25	56 (Gap=1.75%)	56	604800	-	-	-
gsm-st231.29	42	42	210	42	42	513
gsm-st231.30	25	25	58	25	25	67
gsm-st231.31	39	39	142	39	39	169
gsm-st231.32	30	30	0.25	30	30	1.01
gsm-st231.33	52(Gap=8%)	50	604800	-	-	-
gsm-st231.34	7	7	5.05	7	7	8
gsm-st231.35	14	16	52	14	16	53
gsm-st231.36	24	28	230	24	24	403
gsm-st231.39	21	25	95	21	25	168
gsm-st231.40	17	21	15	17	21	29
gsm-st231.41	-	-	-	-	-	-
gsm-st231.42	18	26	12	18	26	17
gsm-st231.43	20	25	15	20	25	23

Table 8: Lower bounds obtained by column-generation for the modified instances

Instances	(decomp+)			(M-decomp+)			Opt	
	λ	$Cmax$	CPU_s	λ	$Cmax$	CPU_s	λ	$Cmax^*$
adpcm-st231.1	52	52	1800	79	79	1758.38	-	-
adpcm-st231.2	82	82	420	-	-	-	-	-
gsm-st231.1	24	32	600	25	42	32.15	25	42
gsm-st231.2	59	59	7200	93	61	4823	-	-
gsm-st231.5	26	26	600	36	36	385.27	36	46
gsm-st231.6	17	17	600	27	27	17.44	27	27
gsm-st231.7	28	28	22	41	41	17.63	41	45
gsm-st231.8	9	9	0.02	12	12	27.36	12	12
gsm-st231.9	28	28	25	31	32	2.57	32	35
gsm-st231.10	6	6	0.0001	8	8	1.73	8	8
gsm-st231.11	20	21	0.15	24	24	1.2	24	24
gsm-st231.12	10	10	0.0001	13	13	0.08	13	13
gsm-st231.13	27	27	48	42	42	48.29	43	48
gsm-st231.14	20	20	18	33	33	15.7	33	45
gsm-st231.15	9	9	0.001	12	12	16.1	12	12
gsm-st231.16	38	38	420	59	59	469	-	-
gsm-st231.17	24	24	720	33	33	16.01	33	33
gsm-st231.18	120	120	600	-	-	-	-	-
gsm-st231.19	12	12	0.002	15	15	0.3	15	15
gsm-st231.20	13	13	0.002	20	20	0.88	20	27
gsm-st231.21	20	22	24	30	30	10.78	30	30
gsm-st231.22	18	21	8	29	29	1.85	29	29
gsm-st231.25	37	37	300	55	55	175.52	56 (Gap=1.75%)	56
gsm-st231.29	28	28	60	42	42	28.82	42	42
gsm-st231.30	16	16	0.25	25	25	3.48	25	25
gsm-st231.31	26	26	58	39	39	28.4	39	39
gsm-st231.32	21	21	3.02	29	29	627	30	30
gsm-st231.33	33	33	52	51	51	3287.42	52(Gap=8%)	50
gsm-st231.34	6	6	0.001	7	7	1.56	7	7
gsm-st231.35	11	11	0.002	14	14	0.46	14	16
gsm-st231.36	18	18	8	24	24	333.16	24	28
gsm-st231.39	15	15	0.06	20	20	9.17	21	25
gsm-st231.40	12	12	0.0001	16	16	2.56	17	21
gsm-st231.41	34	34	47	49	49	812.74	-	-
gsm-st231.42	14	14	4.03	18	18	13.4	18	26
gsm-st231.43	15	15	0.026	20	20	25.43	20	25