MODULO SCHEDULING WITH REGULAR UNWINDING

Benoît Dupont de Dinechin
Benoit.Dupont-de-Dinechin@st.com

Abstract. Modulo scheduling is used by compilers to build 1-periodic cyclic schedules. We present a new framework for modulo scheduling, based on the unwinding of the modulo scheduling problem, and the acyclic scheduling of the resulting unwinded problem under an additional constraint of regularity. Given λ the modulo schedule initiation interval, a regular unwinded schedule is such that two successive instances of any unwinded operation are scheduled at least λ cycles apart. The regular unwinding framework is based on the equivalence between modulo schedules, and regular unwinded schedules of suitable size. Its first application is to solve new modulo scheduling relaxations in pseudo-polynomial time.

Key Words. Cyclic scheduling, modulo scheduling, instruction scheduling.

1 Introduction

1.1 Modulo Scheduling Problems

Modulo scheduling [9, 5, 10] is the cyclic instructions scheduling technique used by compilers for the software pipelining of loops on VLIW processors. In modulo scheduling problems, a set of operations $\{O_i\}_{1\leq i\leq n}$ is repeatedly executed with a period of λ cycles, called the *initiation interval*. Let $\{\sigma_i\}_{1\leq i\leq n}$ denote the modulo schedule dates. Modulo scheduling is constrained as follows [4]:

- Uniform dependence constraints denoted $O_i \xrightarrow{\alpha_i^j,\beta_i^j} O_j$: for each such dependence, a valid modulo schedule satisfies $\sigma_i + \alpha_i^j - \lambda\beta_i^j \leq \sigma_j$. The *latency* α_i^j and the *distance* β_i^j of the dependences are non negative integers. The *carried* dependences are such that $\beta_i^j > 0$. In addition, the dependence graph without the carried dependences is a DAG.
- Modulo resource constraints: each operation O_i requires $\vec{b}_i \geq \vec{0}$ resources for all the time intervals $[\sigma_i + k\lambda, \sigma_i + k\lambda + p_i - 1], k \in \mathbb{N}$, and the total resource use at any time must not exceed $\vec{B}$. The positive integer value p_i is the processing time of operation O_i.

The first difficulty of modulo scheduling problems is that the dependence constraints and the resource constraints are parameterized by the initiation interval λ. To solve modulo scheduling problems, one has to select a given value of λ, then try to build a schedule. If scheduling fails, a new attempt is made at a higher λ value. In the classic modulo scheduling framework [10], the search of a value of λ starts at $\max(\lambda_{rec}, \lambda_{res})$, where:

$$\lambda_{rec} \stackrel{\text{def}}{=} \max_C \frac{\sum_C \alpha_i^j}{\sum_C \beta_i^j} : \quad C \text{ dependence circuit}$$

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

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.

Even when λ is set to a particular value, a modulo scheduling problem is difficult to solve in practice, due to the modulo resource constraints, and because the dependence graph may include circuits (directed cycles). These two features prevent the application of classic machine scheduling theory to modulo scheduling problems.

1.2 Overview of Regular Unwinding

Regular unwinding is our proposed framework to solve modulo scheduling problems. Its principle is the *unwinding* of the modulo scheduling problem, followed by the acyclic scheduling of the resulting unwinded scheduling problem, under an explicit or implicit constraint of *regularity*:

Unwinding is the creation of an acyclic scheduling problem by instancing p iterations of the set of operations $\{O_i\}_{1\leq i\leq n}$ of the modulo scheduling problem, and by instancing the dependences accordingly.

Regularity means that any two operation instances O_i^k, O_i^{k+1} created by the unwinding of any operation O_i of the modulo scheduling problem are scheduled at least λ cycles apart.

Thanks to the regularity condition, we show that either the unwinded schedule becomes stationary with all the O_i^k, O_i^{k+1} operations scheduled exactly λ cycles apart, or the regularized unwinded scheduling problem is infeasible, implying there is no modulo schedule at initiation interval λ. From the stationary part of the unwinded schedule, we extract a 1-periodic cyclic schedule with period λ, that is, a modulo schedule. In cases the regularity needs to be explicitly enforced, the extra constraints introduced are forward dependences with a positive length λ.

Like the classic modulo scheduling framework [9, 5, 10], the regular unwinding framework requires that an initiation interval λ be assumed before scheduling, a difficulty addressed by performing a dichotomy search over the feasible values of λ. The strength of regular unwinding however, is that for any given value of λ, existing machine scheduling techniques apply to the unwinded scheduling problem, yielding in particular the new modulo scheduling results of this paper.

Although solving cyclic instruction scheduling problems by combining unwinding and acyclic scheduling has been proposed earlier [2, 1], these approaches build P-periodic cyclic schedules [4], and require that the span of any iteration be bounded by extra constraints in order to ensure convergence. These extra constraints are dependences with negative length, which make the unwinded scheduling problem harder to solve.

The organization of the paper is as follows. In section 2, we provide the needed background about parallel machine scheduling, in particular extensions to the $\alpha|\beta|\gamma$ scheduling problem denotation [3] for modulo scheduling, and a survey of the acyclic scheduling algorithm of Leung, Palem, and Pnueli [6]. In section 3, we describe the regular unwinding framework, then apply it to solve two modulo scheduling problems relaxations:

$P|r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$, based on the properties of the Graham List Scheduling Algorithm (GLSA) on unwinded scheduling problems.

$P|circuit(\alpha_i^j, \beta_i^j); r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$, by applying the algorithm of Leung, Palem, and Pnueli [6] to unwinded scheduling problems.

2 Scheduling Background

2.1 Parallel Machine Scheduling

In parallel machine scheduling problems, an operation set $\{O_i\}_{1\leq i\leq n}$ is processed on m identical processors. To be processed, each operation O_i requires the exclusive use of one processor during p_i time units, starting at its *schedule date* σ_i.

Parallel machine scheduling problems may also involve *release dates* r_i, and *due dates* d_i. In such cases, the schedule date σ_i of operation O_i is constrained by $\sigma_i \geq r_i$, and there is a penalty whenever $C_i > d_i$, with C_i the *completion date* of O_i defined as $C_i \stackrel{\text{def}}{=} \sigma_i + p_i$. For problems where $C_i \leq d_i$ is mandatory, the d_i are called *deadlines*.

The *dependence constraints* if any are specified by a partial order between the operations. A dependence between two operations O_i and O_j requires O_i to complete before O_j starts, that is, $\sigma_i + p_i \leq \sigma_j$. In case of *time-lags* l_i^j, the dependence constraint becomes $\sigma_i + p_i + l_i^j \leq \sigma_j$, and we call $p_i + l_i^j$ the *length* of the dependence.

Machine scheduling problems are denoted by a triplet notation $\alpha|\beta|\gamma$ [3], where α describes the processing environment, β specifies the operation properties, and γ defines the optimality criterion. For the parallel machine scheduling problems, the common values of α, β, γ are:

α : 1 for a single processor, P for parallel processors, Pm for the given m parallel processors.

β : r_i for release dates, d_i for deadlines (if $\gamma = \bullet$) or due dates, $p_i = 1$ for Unit Execution Time (UET) operations.

γ : $\bullet$ for the problem feasibility, C_{max} or L_{max} for the minimization of these objectives.

The *makespan* is $C_{max} \stackrel{\text{def}}{=} \max_i C_i$, and the *maximum lateness* is $L_{max} \stackrel{\text{def}}{=} \max_i L_i : L_i \stackrel{\text{def}}{=} C_i - d_i$.

The meaning of the additional β fields is:

$prec(l_i^j)$ Dependences with time-lags l_i^j.

$prec(l_i^j = l)$ All the l_i^j have the same value l.

$inTree$ The dependence graph is an in-tree.

$outTree$ The dependence graph is an out-tree.

$intOrder(mono\ l_i^j)$ The dependence graph is a monotone interval order.

As introduced by Papadimitriou & Yannakakis [8], an *interval-order* is defined by an incomparability graph that is chordal. An interval-order is also the complement of an interval graph [8].

A *monotone interval-order* graph is an interval-order graph (V, E) with a weight function w on the arcs such that, given any $(v_i, v_j), (v_i, v_k) \in E$: $w(v_i, v_j) \leq w(v_i, v_k)$ whenever the predecessors of v_j are included in the predecessors of v_k.

We extend the $\alpha|\beta|\gamma$ notation to cyclic scheduling by introducing the additional β fields:

$prec(\alpha_i^j, \beta_i^j)$ Uniform dependences, implying cyclic scheduling.

$circuit(\alpha_i^j, \beta_i^j)$ The dependence graph is a single circuit of uniform dependences.

$\pi_i = \lambda_i$ The processing period of operation O_i is λ_i, implying cyclic scheduling.

A modulo scheduling problem at initiation interval λ is thus implied by $\pi_i = \lambda$ in the β field, because it requires that all the operations have the same processing period λ.

The Graham List Scheduling Algorithm (GLSA) is a fundamental algorithm of machine scheduling, where scheduling is performed by scanning the time slots in increasing order. For each time slot, if a processor is idle, it schedules the highest priority operation available at this time. An operation is available if the current time slot is not earlier than its release date, and all its predecessors have completed their execution early enough to satisfy the entering dependences of this operation.

Blazewicz showed in 1977 that the GLSA optimally solves $P|r_i; d_i; p_i = 1|\bullet$ and $P|r_i; p_i = 1|L_{max}$, when using the earliest d_i dates first as a priority (also known as Jackson's rule).

2.2 The Scheduling Algorithm of Leung, Palem, and Pnueli

Leung, Palem, and Pnueli proposed in 2001 the following algorithm for scheduling UET operations on a parallel machine [6]:

1. Compute the latest possible date d_i', called "modified deadline", any operation O_i can be scheduled at in any feasible schedule.
2. Schedule with the GLSA, using the earliest d_i' first as priorities, and check that the resulting schedule does not miss any deadline.
3. In case of minimizing the maximum completion time C_{max}, or the maximum lateness L_{max}, binary search to find the minimum scheduling horizon such that the scheduling problem is feasible.

The steps 1 and 2 of this algorithm solve the following problems in $O(n^2 \log n\alpha(n) + ne)$ time:

Problem
$1\|prec(l_i^j \in \{0,1\})\|\bullet$
$1\|prec(l_i^j \in \{0,1\}); r_i; d_i; p_i = 1\|\bullet$
$P2\|prec(l_i^j \in \{-1,0\}); r_i; d_i; p_i = 1\|\bullet$
$P\|intOrder(mono\ l_i^j); r_i; d_i; p_i = 1\|\bullet$
$P\|inTree(l_i^j = l); d_i; p_i = 1\|\bullet$
$P\|outTree(l_i^j = l); r_i; p_i = 1\|\bullet$

In order to compute the modified deadlines in step 1, Leung et al. apply a technique of *backward scheduling* [7], where a series of relaxations are optimally solved by the GLSA in order to find, for each operation, its latest schedule date such that the relaxations are feasible. Precisely, the implementation of backward scheduling is as follows:

- Iterate in backward topological order on the scheduling problem operation set, to build a series of scheduling sub-problems $S_i \stackrel{\text{def}}{=} \{O_i, succ_i \cup indep_i, r_j', d_j'\}$. Here $succ_i$ is the set of successors of O_i, and $indep_i$ is the set of operations that are independent from O_i.
- For each scheduling sub-problem S_i, binary search for the latest schedule date p of O_i such that the constrained sub-problem $(r_i' = p) \wedge S_i$ is feasible. If there is such p, define the modified deadline of O_i as $d_i' \stackrel{\text{def}}{=} p + 1$. Else the original scheduling problem is infeasible.
- To find if a constrained sub-problem $(r_i' = p) \wedge S_i$ is feasible, convert the transitive dependence lengths from O_i to all the other O_j of S_i into release dates, then forget the dependences. This yields a relaxation, which is the simpler scheduling problem $P|r_i; d_i; p_i = 1|\bullet$.
- Optimally solve this $P|r_i; d_i; p_i = 1|\bullet$ relaxation using the GLSA with the earliest d_i first priority (Jackson's rule). This gives the feasibility status of the relaxation.

The main theoretical contributions of this work by Leung, Palem, and Pnueli, are the proofs that the feasible schedules computed this way are in fact optimal for all the cases listed above, and the unification of many earlier modified deadlines techniques under a single framework [6].

3 Regular Unwinding

3.1 Regular Unwinding of the Modulo Scheduling Problem

Given a modulo scheduling problem with the operation set $\{O_i\}_{1 \leq i \leq n}$, dependences $\{O_i \stackrel{\alpha_i^j, \beta_i^j}{\longrightarrow} O_j\}_{(i,j) \in D}$, release dates $\{r_i\}_{1 \leq i \leq n}$, and deadlines $\{d_i\}_{1 \leq i \leq n}$, its *p-unwinded scheduling problem* is:

Operation Set $\{O_i^k\}_{1\le i\le n}^{1\le k\le p}$ with schedule dates $\{\sigma_i^k\}_{1\le i\le n}^{1\le k\le p}$

Dependence Constraints $\sigma_i^k + \alpha_i^j \le \sigma_j^{k+\beta_i^j} \quad \forall (i,j) \in E, \forall k \in [1, p-\beta_i^j]$

Resource Requirements $\vec{b}_i^k \stackrel{\text{def}}{=} \vec{b}_i \wedge p_i^k \stackrel{\text{def}}{=} p_i \quad \forall i \in [1,n], \forall k \in [1,p]$

Release Dates $r_i^k \stackrel{\text{def}}{=} r_i + (k-1)\lambda \quad \forall i \in [1,n], \forall k \in [1,p]$

Deadlines $d_i^k \stackrel{\text{def}}{=} d_i + (k-1)\lambda \quad \forall i \in [1,n], \forall k \in [1,p]$

We denote $\bar{\beta} \stackrel{\text{def}}{=} \max_{(i,j)\in E} \beta_i^j$, and $\bar{\omega} \stackrel{\text{def}}{=} \max_i(\lceil \frac{\max_i d_i - \min_j r_j}{\lambda} \rceil, \bar{\beta})$.

Property 1 *The p-unwinded scheduling problem of a modulo scheduling problem is acyclic with non-negative dependence lengths for any $p > 0$.*

Proof: We defined the modulo scheduling problem such that the dependence graph without the carried dependences is acyclic. Carried dependences yield unwinded dependences between O_i^k and O_j^l with $l = k + \beta_i^j > k$. Therefore the unwinded dependence graph has a topological sort. □

Definition 1 *A q-stationary p-unwinded schedule is a solution to a p-unwinded scheduling problem such that:* $\forall i \in [1,n], \forall k \in [q, p-1] : \sigma_i^k + \lambda = \sigma_i^{k+1}$

A q-stationary p-unwinded schedule satisfies: $\forall t \in [1, p-q] \forall i \in [1,n] : \sigma_i^q + t\lambda = \sigma_i^{q+t}$.

Claim 1 $\forall p > 0$: *a modulo schedule can be transformed into a 1-stationary p-unwinded schedule.*

Proof: Given the feasible modulo schedule $\{\sigma_i\}_{1\le i\le n}$ at initiation interval λ, the 1-stationary p-unwinded schedule defined by $\{\sigma_i^k \stackrel{\text{def}}{=} \sigma_i + k\lambda\}_{1\le i\le n}^{1\le k\le p}$ is also feasible. □

Definition 2 *An interesting unwinded schedule is a q-stationary p-unwinded schedule such that:* $\{\sigma_i\}_{1\le i\le n} \stackrel{def}{=} \{\sigma_i^q\}_{1\le i\le n}$ *is a modulo schedule.*

Theorem 1 *Given a λ-feasible modulo scheduling problem, any q-stationary p-unwinded schedule with $p - q \ge \bar{\omega}$ is an interesting unwinded schedule.*

Proof: We show that the dates $\{\sigma_i \stackrel{\text{def}}{=} \sigma_i^q\}_{1\le i\le n}$ define a modulo schedule at initiation interval λ.

For any uniform dependence $O_i \xrightarrow{\alpha_i^j, \beta_i^j} O_j$ of the modulo scheduling problem, we have $\sigma_i^q + \alpha_i^j \le \sigma_j^{q+\beta_i^j} = \sigma_j^q + \lambda\beta_i^j$, since $\beta_i^j \le \bar{\beta} \le \bar{\omega} \le p - q$. Therefore $\sigma_i + \alpha_i^j - \lambda\beta_i^j \le \sigma_j$.

For the resources, let us consider their latest use date by iteration q, and their earliest use date by iteration p. The latest use date is $\max_i(\sigma_i^q + p_i - 1) < \max_i d_i^q$. The earliest use date is $\min_i \sigma_i^p = \min_i \sigma_i^q + (p-q)\lambda \ge \min_i r_i^q + (p-q)\lambda$. The choice of $\bar{\omega}$ ensures $\max_i d_i^q \le \min_i r_i^q + (p-q)\lambda$, so there are no resource conflicts between iterations q and p in the unwinded schedule.

Let us now extend the q-stationary p-unwinded schedule to q-stationary p+1-unwinded with the schedule dates $\{\sigma_i^{p+1} \stackrel{\text{def}}{=} \sigma_i^p + \lambda\}_{1\le i\le n}$. There are no resource conflicts between this iteration $p+1$, and iterations $q \ldots p$, otherwise this contradicts the existence of the q-stationary p-unwinded schedule. By induction, we extend $\forall n \in \mathbb{N}$ the q-stationary p-unwinded schedule to q-stationary p+n-unwinding without resource conflicts.

Thus for any O_i, the resource requirements $\vec{b}_i$ for all the time intervals $[\sigma_i^q + k\lambda, \sigma_i^q + k\lambda + p_i - 1], k \in \mathbb{N}$ are satisfied by taking $\sigma_i \stackrel{\text{def}}{=} \sigma_i^q$. □

3.2 Foundations of the Regular Unwinding Framework

We know from § 3.1 that enough stationarity yields a modulo schedule. Let us show now that enough regularity implies stationarity.

Definition 3 *A q-regular p-unwinded schedule is s-successive-stationary if* $\exists t \in [q, p-s] \forall i \in [1,n] \forall k \in [t, t+s-1] : \sigma_i^k + \lambda = \sigma_i^{k+1}$

In other words, stationarity holds for s successive iterations. When a q-regular p-unwinded schedule is not s-successive-stationary, by regularity we have: $\forall t \in [q, p-s] \exists i \in [1,n] : \sigma_i^t + s\lambda < \sigma_i^{t+s}$.

Theorem 2 *Given a λ-feasible modulo scheduling problem with finite deadlines $\forall s > 0 \exists r > 0 \forall q > 0 \forall p \ge q + r$: any q-regular p-unwinded schedule is s-successive-stationary.*

Proof: By contradiction, assume $\exists s > 0 \forall r > 0 \exists q > 0 \exists p \ge q + r$: a q-regular p-unwinded schedule is not s-successive-stationary. This implies $\forall t \in [q, p-s] \exists i \in [1,n] : \sigma_i^t + s\lambda + 1 \le \sigma_i^{t+s}$.

From the definition of the deadline dates of the unwinded scheduling problem, we have $d_i^t + s\lambda = d_i^{t+s}$, so each inequality becomes $\forall t \in [q, p-s] \exists i \in [1,n] : \sigma_i^t - d_i^t + 1 \le \sigma_i^{t+s} - d_i^{t+s}$.

Summing over $i \in [1,n]$ yields the inequality $\forall t \in [q, p-s] : \sum_i(\sigma_i^t - d_i^t) + 1 \le \sum_i(\sigma_i^{t+s} - d_i^{t+s})$. Then for each $t \in (q, q+s, \ldots q+ms)$, with $m \stackrel{\text{def}}{=} \lfloor \frac{p-q}{s} \rfloor$, we sum this inequality and obtain $\sum_i(\sigma_i^q - d_i^q) + m \le \sum_i(\sigma_i^{q+ms} - d_i^{q+ms})$.

By increasing $m \stackrel{\text{def}}{=} \lfloor \frac{p-q}{s} \rfloor$, we get $0 \le \sum_i(\sigma_i^q - d_i^q) + m$, so $0 \le \sum_i(\sigma_i^{q+ms} - d_i^{q+ms})$ and the q-regular p-unwinded schedule is infeasible.

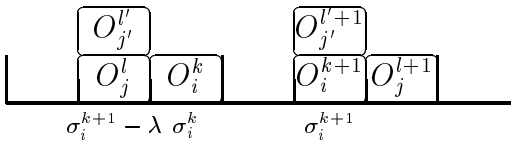

Figure 1: Illustration of the proof of Theorem 3.

In particular, $\sum_i(\sigma_i^q - d_i^q) + m \geq \sum_i(r_i^q - d_i^q) + m = \sum_i(r_i - d_i) + m$, so $m \geq \sum_i(d_i - r_i)$ ensures infeasibility. Finally, a choice of $r \geq s\sum_i(d_i - r_i) + s$ ensures the q-regular p-unwinded schedule is infeasible. This completes the contradiction. □

Corollary 1 *A modulo scheduling problem and its regularized p-unwinded scheduling problem, with $p \geq \bar{\rho} \stackrel{def}{=} \bar{\omega}\sum_i(d_i - r_i) + \bar{\omega} + 1$, are equivalent.*

Proof: Let us first assume a feasible modulo scheduling problem at initiation interval λ. By Claim 1, this yields a 1-stationary p-unwinded schedule for any p, in particular for $p \geq \bar{\rho}$.

Conversely, assume a feasible regularized p-unwinded scheduling problem, with $p \geq \bar{\rho}$. Thanks to regularization, any solution is a 1-regular p-unwinded schedule. Theorem 2 and the definition of $\bar{\rho}$ ensure this 1-regular p-unwinded schedule is $\bar{\omega}$-successive-stationary, so by Theorem 1, it contains an interesting unwinded schedule. The interesting unwinded schedule yields a modulo schedule. □

This provides the foundations of the regular unwinding framework: to solve a modulo scheduling problem at initiation interval λ, build the corresponding regularized p-unwinded scheduling problem, with $p \geq \bar{\rho}$. By Corollary 1, either this regularized p-unwinded scheduling problem is feasible, and we convert its solution to a modulo schedule, or it is not, and there is no feasible modulo schedule at initiation interval λ either.

By comparison, the earlier unwinding techniques proposed for cyclic scheduling with resource constraints [2, 1] neither guarantee optimality, nor bound the unwinding degree that guarantees to converge on a P-periodic schedule.

3.3 Modulo Scheduling Relaxations with Regular Unwinding

Theorem 3 *Running the GLSA on the p-unwinded scheduling problem without dependences, with UET operations, and with a priority function P such that $P(O_i^k) < P(O_j^l) \Rightarrow P(O_i^{k+1}) < P(O_j^{l+1})$, yields a 1-regular p-unwinded schedule.*

Proof: Assume that the unwinded schedule is not regular. Let O_i^k be the earliest scheduled operation such that $\sigma_i^k + \lambda > \sigma_i^{k+1}$. At date $\sigma_i^{k+1} - \lambda$, operation O_i^k is available, as $r_i^k = r_i^{k+1} - \lambda \leq \sigma_i^{k+1} - \lambda$. Since operation O_i^k was not scheduled at $\sigma_i^{k+1} - \lambda < \sigma_i^k$ even though it was available, and because of the UET hypothesis, there must exist operations $O_j^l, O_{j'}^{l'}, \ldots$ that are all scheduled at date $\sigma_i^{k+1} - \lambda$, meaning they have a higher priority than O_i^k. At date σ_i^{k+1}, the operations $O_j^{l+1}, O_{j'}^{l'+1}, \ldots$ are also available, because $\sigma_j^l = \sigma_i^{k+1} - \lambda \Rightarrow r_j^l \leq \sigma_i^{k+1} - \lambda \Rightarrow r_j^{l+1} \leq \sigma_i^{k+1}$. These operations cannot be scheduled earlier, that is, $\sigma_j^{l+1} < \sigma_i^{k+1}$, because $\sigma_j^{l+1} \geq \sigma_j^l + \lambda$ by regularity, and $\sigma_j^l = \sigma_i^{k+1} - \lambda$. Thus, operation O_i^{k+1} is scheduled before at least one of the operations $O_j^{l+1}, O_{j'}^{l'+1}, \ldots$, meaning it has a higher priority. This contradicts the hypothesis on the priority function P. □

Corollary 2 *The modulo scheduling problem $P|r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$ can be solved in pseudo-polynomial time.*

Proof: Build the corresponding p-unwinded scheduling problem, with $p \geq \bar{\rho}$. This yields a $P|r_i; d_i; p_i = 1|\bullet$ problem that is optimally solved by the GLSA using the earliest d_i first priority. The priority $P(O_i^k) \stackrel{\text{def}}{=} d_i^k \stackrel{\text{def}}{=} d_i + (k-1)\lambda$ of an unwinded scheduling problem satisfies the conditions of Theorem 3, so the resulting schedule if feasible is 1-regular. The value of p ensures this schedule contains an interesting schedule. □

Theorem 4 *Unwinding a modulo scheduling problem $P|circuit(\alpha_i^j, \beta_i^j); r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$ and adding the regularizing dependences creates a $P|intOrder(mono\ l_i^j); r_i; d_i; p_i = 1|\bullet$ problem.*

Proof: Unwinding a modulo scheduling problem $P|circuit(\alpha_i^j, \beta_i^j); r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$ without adding the regularizing dependences yields an unwinded dependence graph that is a chain. The transitive closure of a chain is an interval-order graph, as we can associate to the k-th vertex in the chain the closed interval $[k, k]$.

For each arc of the transitive closure leaving a vertex v_i, now assign to it the same weight (length) as the arc from v_i to its direct successor along the chain. The result is a monotone interval-order graph, as for any two of arcs (v_i, v_j) and (v_i, v_k) their weights w are the same. This dependence graph only enforces the chain dependences.

Now add each regularizing dependence arc in turn to the above graph. Each such arc is parallel to an existing arc (v_i, v_k), and if not redundant, this has the effect of increasing $w(v_i, v_k)$ to λ. Let us consider any successor v_j of v_i. If v_j is between v_i and v_k in the chain, then $w(v_i, v_j) \leq w(v_i, v_k)$

still holds. If v_j is after v_k in the chain, $w(v_i, v_j) \geq w(v_i, v_k)$ may no longer hold. In such case increasing $w(v_i, v_j)$ to $w(v_i, v_k)$ restores the monotone interval order property, without further constraining the problem because $w(v_i, v_j) = w(v_i, v_k) \leq w(v_i, v_k) + w(v_k, v_j)$. □

Corollary 3 *The modulo scheduling problem* $P|circuit(\alpha_i^j, \beta_i^j); r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$ *can be solved in pseudo-polynomial time.*

Proof: Build the corresponding p-unwinded scheduling problem, with $p \geq \bar{\rho}$, and add the regularizing dependences. This yields a $P|intOrder(mono\ l_i^j); r_i; d_i; p_i = 1|\bullet$ problem that is optimally solved by the algorithm of Leung, Palem, and Pnueli [6]. □

Theorem 5 *Given a regularized p-unwinded UET problem with* $p \geq s \sum_i (d_i - r_i) + s + 1$*, the modified deadlines computed by the algorithm of Leung, Palem, and Pnueli [6] are s-successive stationary.*

Proof: The modified deadlines $d_i'^k$ computed by the backward scheduling process verify: $d_i'^k + \lambda \leq d_i'^{k+1}$, and $r_i^k + 1 \leq d_i'^k \leq d_i^k$. The dates $\{\sigma_i'^k \stackrel{\text{def}}{=} d_i'^k - 1\}$ thus define a pseudo 1-regular p-unwinded schedule, and the proof of Theorem 2 applies. □

This result is useful when using the algorithm of Leung, Palem, and Pnueli [6], to build the p-unwinded schedules: further backward scheduling is not necessary as soon as the modified deadlines become stationary for s iterations. Moreover, using $d_i'^k \stackrel{\text{def}}{=} d_i'^1 + (k-1)\lambda$ improves the convergence rate of the p-unwinded schedules.

Conclusions

We propose a new framework for modulo scheduling, whose principles are to unwind the modulo scheduling problem, and to apply acyclic scheduling techniques in a way that ensures the regularity of the resulting unwinded schedule.

Our main result is the equivalence between any modulo scheduling problem, and its regularized p-unwinded problem, for $p \geq \bar{\rho}$ of pseudo-polynomial size: either the regularized p-unwinded schedule becomes stationary for enough iterations to yield a modulo schedule, or there is no modulo schedule.

Given this equivalence, the application of existing parallel machine scheduling theory to regularized unwinded scheduling problems yields new facts and techniques for modulo scheduling. In particular, we show that the modulo scheduling problems $P|r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$, and $P|circuit(\alpha_i^j, \beta_i^j); r_i; d_i; p_i = 1; \omega_i = \lambda|\bullet$, can be solved in pseudo-polynomial time.

References

[1] A. Aiken, A. Nicolau, S. Novack: *Resource-Constrained Software Pipelining* IEEE Transactions on Parallel and Distributed Systems, 6, 12, 1996.

[2] F. Bodin, F. Charot: *Loop Optimization for Horizontal Microcoded Machines* Proceedings of the 1990 International Conference on Supercomputing, Amsterdam 1990.

[3] G. Finke, V. Gordon, J.-M. Proth: *Scheduling with Due Dates (Annotated Bibliography of Complexity and Algorithms)* Les cahiers du laboratoire Leibniz, Jan. 2002. `http://www-leibniz.imag.fr/LEIBNIZ/LesCahiers/2002/Cahier42/`

[4] B. Dupont de Dinechin: *From Machine Scheduling to VLIW Instruction Scheduling* Technical Report A/352/CRI, November 2003, `http://www.cri.ensmp.fr/classement/2003.html`.

[5] M. Lam: *Software Pipelining: an Effective Scheduling Technique for VLIW Machines* SIGPLAN Conference on Programming Language design and Implementation – PLDI'88, June 1988.

[6] A. Leung, K. V. Palem, A. Pnueli: *Scheduling Time-Constrained Instructions on Pipelined Processors* ACM TOPLAS Vol. 23, No. 1, Jan. 2001.

[7] K. V. Palem, B. Simons: *Scheduling Time-critical Instructions on RISC Machines* ACM Symposium on Principles of Programming Languages – POPL'90, 1990.

[8] C. Papadimitriou, M. Yannakakis: *Scheduling Interval-Ordered Tasks* SIAM Journal of Computing, 8,1979.

[9] B. R. Rau , C. D. Glaeser: *Some Scheduling Techniques and an Easily Schedulable Horizontal Architecture for High Performance Scientific Computing* 14th Annual Microprogramming Workshop on Microprogramming – MICRO-14, Dec. 1981.

[10] B. R. Rau: *Iterative Modulo Scheduling* The International Journal of Parallel Processing, 24, 1, Feb 1996.