Computers & Operations Research 51 (2014) 227–236

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Converging to periodic schedules for cyclic scheduling problems with resources and deadlines

Benoît Dupont de Dinechin [a], Alix Munier Kordon [b,*]

[a] Kalray, 445 rue Lavoisier, 38330 Montbonnot Saint Martin, France
[b] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, France

ARTICLE INFO

Available online 13 March 2014

Keywords:
Cyclic scheduling
Throughput maximization
Resource constrained project scheduling problem
Software pipelining

ABSTRACT

Cyclic scheduling has been widely studied because of the importance of applications in manufacturing systems and in computer science. For this class of problems, a finite set of tasks with precedence relations and resource constraints must be executed repetitively while maximizing the throughput. Many applications also require that execution schedules be periodic *i.e.* the execution of each task is repeated with a fixed global period w.

The present paper develops a new method to build periodic schedules with cumulative resource constraints, periodic release dates and deadlines. The main idea is to fix the period w, to unwind the cyclic scheduling problem for some number of iterations, and to add precedence relations so that the minimum time lag between two successive executions of any task equals w. Then, using any usual (not cyclic) scheduling algorithm to compute task starting times for the unwound problem, we prove that either the method converges to a periodic schedule of period w or it fails to compute a schedule. A non-polynomial upper bound on the number of iterations to unwind in order to guarantee that cyclic precedence relations and resource constraints are fulfilled is also provided. This method is successfully applied to a real-life problem, namely the software pipelining of inner loops on an embedded VLIW processor core by using a Graham list scheduling algorithm.

1. Introduction

A cyclic scheduling problem is usually defined by a set of tasks that has to be repeated infinitely. This class of problems has been widely studied in the last few years because of the importance of practical applications in different fields (see the surveys [19,24]). For manufacturing systems, they may be found in mass production (Crama [9]; Amstrong et al. [2]; Chen et al. [5]; Kim et al. [20]). Tasks usually represent operations of the production process of a manufactured object for which a large number of copies are to be produced. In computer science, cyclic scheduling appears with software pipelining (Gasperoni and Schwiegelshohn [15]; Allan et al. [1]; Rau [28]), that is, compiler instruction scheduling of the inner program loops on instruction-level parallel or pipelined processor cores. Cyclic scheduling problems were also considered in the context of real-time systems (Cucu and Sorel [10]; Šůcha et al. [30]).

Despite originating from different application domains, two common assumptions appear in most previous works. First, relationships between a finite set of generic tasks are modeled with a bi-valued oriented graph $G=(T,A,\ell,h)$ defined as follows: T is a set of generic tasks, each of them with a fixed duration $p_i \geq 0$. Each arc $e=(i,j)\in A$ is associated with a pair of values $(\ell_{ij},h_{ij})\in\mathbb{Z}^2$ and defines the infinite set of precedence constraints $\forall k > \max\{0,-h_{ij}\}$, $t(i,k)+\ell_{ij}\leq t(j,k+h_{ij})$ where $t(i,k)$ (*resp.*$t(j,k+h_{ij})$) is the starting time of the kth (*resp.*$k+h_{ij}$th) execution of generic task i (*resp.*j). Resource constraints (number of machines, parallel or dedicated processors, bandwidth, etc…) are usually fixed. The objective is to find a feasible schedule with the maximum throughput.

The second common assumption is that solutions are constrained to periodic schedules, that is, there exists a period $w\in\mathbb{Q}^+-\{0\}$ such that, for every pair $(i,k)\in T\times\mathbb{N}-\{0\}$, $t(i,k)=t(i,1)+(k-1)w$. Even if this limitation may lead to sub-optimal solutions (since in the presence of resources, periodic schedules are not dominant [19]), it has obvious implementation advantages; thus, most authors dealing with a practical application have limited their study to this simple class of schedules. In this paper, we limit our study to periodic schedules with the objective of minimizing the period, which is equivalent to maximizing the throughput.

The determination of a periodic schedule with the maximum throughput for a bi-valued graph (without resource limitations) is solved polynomially: indeed, Ramchandani [27] solved it for $h_{ij}\geq 0$ and $\ell_{ij}=p_i>0$, $\forall e=(i,j)\in A$. Chrétienne in [6] and Cohen et al. in [8] proved that the throughput of the earliest schedule equals the maximum throughput of a periodic schedule. All these results were

* Corresponding author.
E-mail addresses: Benoit.Dinechin@kalray.eu (B. Dupont de Dinechin), Alix.Munier@lip6.fr (A. Munier Kordon).

http://dx.doi.org/10.1016/j.cor.2014.03.004
0305-0548/

extended separately by Lee and Park [22,23] and Munier [25] to potentially negative values of ℓ_{ij} and h_{ij}. Note that Chrétienne in [7] also studied the existence of a cyclic (not necessarily periodic) schedule with deadlines and the structure of the latest schedule for a bi-valued graph with $h_{ij} \geq 0$ and $\ell_{ij} = p_i > 0$, $\forall e = (i,j) \in A$.

The computation of a periodic schedule in the presence of resource limitations (with or without release dates and deadlines) is a difficult problem. The complexity is clearly strongly $\mathcal{NP}-$hard, since it includes the computation of a classical (acyclic) scheduling problem of a given length under resource constraints. Many authors have noticed that it may be modeled with Integer Linear Programming. In the setting of periodic cyclic scheduling of inner loop instructions on a Very Long Instruction Word (VLIW in short) processor core, Govindarajan et al. [16] expressed their problem by using a time-indexed formulation. This formulation was improved by Eichenberger and Davidson [13] in order to solve practical cases with a commercial solver. Dupont de Dinechin developed another formulation [12] inspired by the classic non-preemptive time-indexed formulation of Pritsker et al. [26] for the Resource Constrained Project Scheduling Problem (RCPSP in short) [3].

The major drawback of the time-indexed formulations is that the numbers of variables and equations grow with the period. However, Dupont de Dinechin showed that large neighborhood search techniques were effective for heuristically solving problem instances with hundreds of generic tasks [11]. Other formulations were also developed to model resource limitations more efficiently. Hanen in [18] observed that the problem for dedicated processors may be modeled by Integer Linear Programming with the starting time of the first execution of the generic tasks. This formulation was considered by Brucker and Kampmeyer in [4] to test its efficiency for several particular classes of cyclic scheduling problems. Another formulation was developed and tested by Šůcha and Hanzálek in [29] for typed task systems.

Heuristics based on list scheduling are popular among practitioners to handle resource constraints: when a resource is available, it is allocated to a ready task of highest priority [17]. In particular, Gasperoni and Schwiegelshohn in [15] proposed a simple technique to build periodic schedules in the presence of resource constraints, assuming $\ell_{ij} = p_i > 0$ and $h_{ij} \geq 0$ for any arc $(i,j) \in A$. A periodic schedule $t^{\infty}(i,1)$, $i \in T$ of period w_{∞} is first computed from the bi-valued precedence graph ignoring resource constraints. A (noncyclic) acyclic precedence graph $G^{\star} = (T, A^{\star}, \ell)$ is then built by considering only arcs $e = (i,j) \in A$ of null heights (*i.e.* with $h_{ij} = 0$) such that $t_i^{\infty} + \ell_{ij} \leq t_j^{\infty}$ with $t_i^{\infty} = t^{\infty}(i,1) \bmod w_{\infty}$. A list schedule of $G^{\star}$ with the original resource constraints yields a noncyclic schedule of makespan w. A periodic schedule is then built by repeating the (acyclic) schedule obtained with period w. The performance ratio is close to 2 for identical machines.

Other cyclic scheduling heuristics that build periodic schedules have been proposed for loop software pipelining on VLIW processor cores. In particular, the *modulo scheduling* framework [28] uses a job-based list scheduling (*i.e.* at each step of the algorithm, available tasks are listed and the algorithm chooses a task with a highest priority) extended with backtracking. Assuming a period w, the starting times of the scheduled operations are considered modulus by w for handling the resources (*i.e.* any task i for which its first execution $t(i,1)$ is fixed requires its resources at time $t(i,1) \bmod w$). This heuristic is attempted for increasingly larger values of the period w until it succeeds.

The main contribution of this paper is to present a new method to build periodic schedules with periodic release dates, deadlines and complex resource limitations. A feasible periodic schedule is built using a scheduler (heuristic or optimal) for the associated noncyclic scheduling problem. An arbitrary period w is fixed and the scheduler computes the successive starting times of the generic tasks. In favorable cases, our algorithm converges to a feasible periodic schedule of period w. Otherwise, it must be restarted with a larger value for the period. Resource constraints are a cyclic extension of the RCPSP, where resources are divided into P classes, each of them composed by a fixed number of identical machines. Each generic task $i \in T$ requires a subset of resources during each execution defined by a vector b_i of size P.

The bi-valued graph G is supposed to be strongly connected and to comprise a fictitious task 0 from T that is scheduled periodically with fixed period w. We show that this assumption allows the association to any other task from T of periodic release dates and deadlines, which are characterized by critical paths of G for the period w.

The main idea developed in this paper is to add fictitious precedence relations (i,i), $\forall i \in T$ with $\ell_{ii} = w$ and $h_{ii} = 1$ called regularizing precedence relations and to compute a feasible schedule (using as example a list scheduling algorithm) after unwinding the scheduling problem. The minimum time lag between two consecutive executions of a same generic task is then w. Because of the presence of periodic release dates and deadlines, the number of iterations for which this difference is strictly greater than w is bounded in any feasible schedule; we show then that after a limited number of iterations, for any integer $\Delta > 0$, the difference between Δ consecutive executions of each generic task is exactly w.

Afterward, two lower bounds on Δ are expressed to ensure that a feasible periodic schedule of period w may be built from the unwound schedule. The first bound Δ_1 is the minimum for fulfilling the precedence relations. The other bound Δ_2 is the minimum for satisfying the resource constraints.

A case study coming from an actual industrial problem is lastly presented to illustrate our method. The problem is to find a periodic schedule for inner loop operations on an embedded VLIW processor core of the Lx/ST200 family [14]. Benchmarks are extracted from real-life programs, each of them with several inner loops. The acyclic scheduling heuristic used here is a Graham list scheduling algorithm with a priority proportional to an upper bound of the longest path to a final task. Experimental results are compared to the near-to-optimal modulo scheduling method developed by Dupont de Dinechin et al. in [12] and with lower bounds of the period.

Organization of the paper is as follows. Section 2 presents some additional notations and the computation of release dates and deadlines. An example coming from [29] illustrates our notations. Section 3 is devoted to the convergence proof of any feasible schedule after the regularizing precedence relations are added. Section 4 shows how to build a feasible periodic schedule from the feasible (not necessarily periodic) schedule obtained in the previous section. A minimum number of iterations required to fulfill the cyclic precedence relations and resource constraints is also evaluated. Section 5 presents our case study. Section 6 is our conclusion.

2. Basic Notations

The aim of this section is to present formally the problem tackled in this paper.

2.1. Generic tasks and precedence relations

Let $T = \{0, \ldots, n\}$ be the set of generic tasks, $n \geq 1$. For any integer $\nu > 0$ and task $i \in T$, $\langle i, \nu \rangle$ denotes the νth execution of i, each of them with duration $p_i \geq 0$.

Precedence relations are defined by a bi-valued directed and strongly connected graph $G = (T, A, \ell, h)$: each arc $e = (i,j) \in A$ is associated with a pair of values $(\ell_{ij}, h_{ij}) \in \mathbb{Z}^2$ and corresponds to the