Efficient Algorithms for Cyclic Scheduling

Franco Gasperoni
gasperon@cs.nyu.edu
Courant Institute of Mathematical Sciences
251 Mercer Street, New York, NY 10012

Uwe Schwiegelshohn
uwe@watson.ibm.com
IBM T.J. Watson Research Center
P.O. Box 218, Yorktown Heights, NY 10598

Abstract

This work addresses the problem of non-preemptively scheduling a cyclic set of interdependent operations, representing for instance a program loop, when p identical processors are available. For $p = \infty$ we give a simple, efficient, polynomial time algorithm producing optimum results. When $p < \infty$ the problem becomes NP-hard and a slight modification of our algorithm generates provably close to optimum results.

1 Introduction

With advances in hardware technology most of todays high performance computers offer some degree of parallelism. To take advantage of this concurrency, parallel extensions to sequential programming languages have been designed. Such extensions have mostly proved inadequate as they are usually tailored to some underlying parallel architecture and are consequently not portable. On the other hand there is a large amount of sequential applications that the users would like to run on these high performance computers. The high cost of machine specific software development and the lack of highly portable parallel programming languages is a major obstacle in their rewriting. A common approach has been to employ parallelizing compilers which automatically extract the parallelism present in sequential applications [21,3,2]. Most of the concurrency present in these programs is expressed in the form of loops and considerable efforts have been devoted to loop parallelization ([22,17,8,16,20] to name a few).

The problem studied in this paper is optimum and nearly optimum cyclic scheduling. More specifically we propose a simple and efficient scheduling algorithm which produces optimal results when the number of processors is infinite and nearly optimum results otherwise.

The framework employed is that of cyclic scheduling [13,4] and consequently our model encompasses any problem involving the cyclic execution of interdependent activities by a given number of agents. In the sequel the activities and agents are respectively called operations and processors and a full cycle of activities is called an iteration.

In brief a cyclic scheduling model is characterized by a number $p \geq 1$ of available processors and a cyclic task system, that is a doubly weighted directed graph $G = (O, E, \delta, d)$ where the vertex set O is the set of operations to execute, δ is a mapping from O in the positive rationals giving the time an operation needs to complete, the edge set E and the weight function d mapping E into the non-negative integers specify operation dependences. More specifically an edge $e = (op, op') \in E$ with weight $d(e)$ implies that operation op' in iteration i, denoted $op'[i]$, depends on the outcome of operation $op[i - d(e)]$ and cannot be started until its completion. The only restriction imposed on the edge weight function d is that every cycle in G must contain at least one edge with positive weight. Our cyclic scheduling model extends those of [13,4] in that operation $op[i]$ need not depend on $op[i-1]$ and G need not be strongly connected. In the remainder of the paper integer entities will be denoted by roman literals, whereas rational entities will be denoted by greek literals.

Let $[0,\infty]$ denote the set of non-negative integers and $O[0,\infty]$ the set product of O and $[0,\infty]$, that is $O[0,\infty] = \{op[i] \,:\, op \in O \text{ and } i \in [0,\infty]\}$. The overall goal is to construct a rational schedule, that is a mapping σ from $O[0,\infty]$ into the non-negative rationals, which model time instants, such that

1. σ is periodic

$$\exists\, l \geq 1 \quad \exists\, \tau_{ii} \geq 0 \quad \forall\, i \geq 0 \qquad \sigma(op[i]) = \sigma(op[i \bmod l]) + \tau_{ii} \cdot \left\lfloor \frac{i}{l} \right\rfloor$$

2. σ has optimum asymptotic performance $\|\sigma\| = \tau_{ii}/l$

3. σ satisfies G's dependence constraints

$$\forall\, e = (op, op') \in E \quad \forall\, i \geq 0 \qquad \sigma(op[i]) + \delta(op) \leq \sigma(op'[i + d(e)])$$

4. σ can be executed by at most p processors

$$\forall\, \tau \geq 0 \qquad |\{op[i] \in O[0,\infty] \,:\, 0 \leq \tau - \sigma(op[i]) < \delta(op)\}| \leq p$$

A rational schedule which maps $O[0,\infty]$ into the non-negative integers, is said to be integral. A more formal definition of the model will be given in sections 2 and 3.

A central problem in unveiling periodic schedules with asymptotically optimum performance is determining the maximum duration-to-weight ratio of the cycles in G [18]. More precisely let P be a path in G traversing operations $op_1, \ldots, op_k$ and edges $e_1, \ldots, e_c$, where $c = k$ if P is a cycle and $c = k - 1$ otherwise. If one defines

$$\delta(P) = \sum_{i=1}^{k} \delta(op_i) \quad \text{and} \quad d(P) = \sum_{i=1}^{c} d(e_i)$$

then the maximum duration-to-weight ratio of the cycles in G, which we denote ρ_G, is

$$\rho_G = \max_{\substack{C \text{ cycle} \\ \text{in } G}} \frac{\delta(C)}{d(C)}$$

If G is acyclic, ρ_G is zero by definition. When $\delta(op) = 1$ for every operation op, $1/\rho_G$ is called the minimum cycle mean of G. Efficient algorithms for the minimum cycle mean problem have been given by Karp [15], Ahuja & Orlin [1] and Young, Tarjan & Orlin [23]. Let n and m respectively denote the number of operations and dependence edges in G and $d_{max} = \max_{e \in E} d(e)$. Karp's algorithm runs in $O(n\,m)$ time, Ahuja & Orlin's runs in $O(\sqrt{n}\,m \log(n\,d_{max}))$ and Young, Tarjan & Orlin's runs in $O(\,n\,m + n^2 \log n)$ time in the worst case but experiments on random graphs have suggested that its expected running time is $O(m + n \log n)$. All minimum cycle mean algorithms can be extended to compute ρ_G in the case where operations have arbitrary *integer* durations by adding an additional $O(\log(n\,\delta_{max}\,d_{max}))$ factor in time complexity, where $\delta_{max} = \max_{op \in O} \delta(op)$ [11]. The general case where operation durations are positive rationals can be reduced to the integer case by multiplying every $\delta(op)$ by lcm, the least common multiple of the denominators of the $\delta(op)$. The ρ_G obtained by this transformation must be divided by lcm in order to obtain the actual maximum duration-to-weight ratio of the cycles in G. The algorithm of Young, Tarjan & Orlin can also be used directly in the general case when operations have rational durations. In that instance the worst case running time increases to $O(d_{max}(\,n\,m + n^2 \log n))$. The expected running time should also increase but by a much smaller factor than $O(d_{max})$. Note that to improve the running time of the previous algorithms in the case where $\rho_G \geq \delta_{max}$ one can initially delete all dependence edges e such that $d(e) > n$. This guarantees that $d_{max} \leq n$.

The first problem studied (section 4), is to efficiently generate a periodic schedule with optimum asymptotic performance when there is an unbounded number of processors available, that is $p = \infty$. Previous work by Iwano & Yeh assumes integral schedules and integral operation durations [14]. They give an $O(n\,m\,l_G^2 + \mathcal{T})$ pseudo-polynomial time algorithm where l_G is the denominator of ρ_G in its irreducible form and $\mathcal{T}$ the time to compute ρ_G. Note that l_G can be as big as $n\,d_{max}$. We give a simple $O(\mathcal{T})$ time algorithm for the general case of rational schedules and rational operation durations. When the schedule and operation durations are required to be integral the algorithm generates periodic schedules a factor $\lceil \rho_G \rceil / \rho_G$ away from the asymptotic optimum. A slight modification of the algorithm generates in $O(n\,m\,l_G^2 + \mathcal{T})$ time, asymptotically optimum results.

When the number of processors p is finite the problem of generating an asymptotically optimum periodic schedule becomes NP-hard. Our second contribution, section 5, is to efficiently generate near optimum schedules. Most previous work in this domain has been of empirical nature and performance bounds have solely been validated by benchmarking. Our algorithm is simple, runs in $O(\mathcal{T})$ time and guarantees a maximum factor from optimality of $(2-1/p)+(p-1)/p\cdot\delta_{max}/\|L\|_p$, where $\|L\|_p$ denotes the optimum asymptotic performance for the input cyclic task system L when p processors are available. A better performance bound of $(2-1/p)+(p-1)/p\cdot\delta_{max}/(l\cdot\|L\|_p)$, where l is a user selected parameter, can be obtained at an additional $O(n\,m\,l^2)$ cost in time complexity. If operation durations are integral and the generated schedule is also required to be integral a slight modification of the algorithm guarantees a worst case performance bound of $1+(p-1)/p\cdot((\delta_{max}-1)+\lceil\|L\|_p\rceil)/\|L\|_p$.

2 Task Systems and Admissible Schedules

We first introduce task systems [5,12]. Informally a task system is a collection of several interdependent operations all of which must be executed in order to complete the task.

Definition 1 *A task system T is a triple $T=(O,\delta,\prec)$ where:*

1. *O is the operation set of T, a non necessarily finite set of operations.*
2. *δ is the duration function of T, a function mapping O into the positive rationals.*
3. *$\prec$ is the dependence relation of T, a partial order on O.*

If O is finite then T is said to be acyclic otherwise T is said to be infinite.

The machine model considered comprises p identical processors operating in parallel. There is no preemption: once started, an operation has to be executed without interruption. Given some task system T we formalize the notion of a rational schedule σ for T.

Definition 2 *Let $T=(O,\delta,\prec)$ be some task system. An admissible p-schedule σ for T is a mapping from O into the non-negative rationals such that:*

1. *No more than p operations are being processed at any given moment:*

$$\forall\,\tau\geq 0\qquad |\{op[i]\in O[0,\infty]\;:\;0\leq\tau-\sigma(op[i])<\delta(op)\}|\leq p$$

2. *No operation can start executing until all operations on which it depends have completed:*

$$\forall\,op,op'\in O\quad op\prec op'\;\Rightarrow\;\sigma(op)+\delta(op)\leq\sigma(op')$$

If the schedule σ maps O into the non-negative integers, σ is said to be integral. When the task system T is acyclic one defines $|\sigma|$, the length of σ, as $|\sigma| = \max_{op \in O}(\sigma(op) + \delta(op))$. If no admissible p-schedule for T has a length smaller than $|\sigma|$, σ is said to be p-optimum for T.

For any finite $p > 1$, the problem of generating a p-optimum schedule for an acyclic task system is NP-complete [19]. If operations are restricted to have same duration a polynomial time optimum algorithm for the case $p = 2$ was presented by Coffman and Graham and an almost linear time algorithm was given by Gabow [6,9]. The problem remains open for any fixed $p \geq 3$ that is, NP-hardness has not been proved or disproved [10]. If however p is considered to be a parameter the problem becomes NP-hard [19].

The algorithms of Coffman & Graham and Gabow build on the list scheduling framework [5]. List scheduling algorithms work as follows. The operations in the acyclic task system are implicitly ordered in a priority list. At any given instant where a processor is free an operation is scheduled by selecting the first operation in the priority list all of whose predecessors, with respect to the dependence relation, have finished executing. Note that for $p = \infty$ any list scheduling algorithm yields optimum schedules. If $p < \infty$ any list scheduling algorithm guarantees a schedule length of at most $(2 - 1/p)$ times the p-optimum [5]. The NP-hardness proof given by Lenstra & Rinnooy Kan [19] implies that, unless P = NP, no polynomial time algorithm can approximate p-optimality for arbitrary p, by less than a factor 4/3.

3 Cyclic Task Systems and Periodic Schedules

The following definition models the behavior of a system which must continuously execute a fixed set of interdependent operations.

Definition 3 *A cyclic task system L is an infinite task system $L = (O[0,\infty], \delta, \prec)$ where:*

1. *$O[0,\infty]$, the operation set of L is the product of O, a finite set of operations, and $[0,\infty]$.*
2. *For all $op \in O$ and $i, j \geq 0$, $\delta(op[i]) = \delta(op[j])$. We will denote such number $\delta(op)$.*
3. *For all $op, op' \in O$ and $i, j \geq 0$, $op[i] \prec op'[j]$ implies $i \leq j$. Furthermore let $d(op, op') = \min\{j - i : op[i] \prec op'[j]\}$, where the minimum of the empty set is equal to ∞ by definition. Then if $d(op, op') \neq \infty$ it is required that for all $i \geq 0$, $op[i] \prec op'[i + d(op, op')]$.*

The cyclic task system $L = (O[0,\infty], \delta, \prec)$ will be portrayed by a doubly weighted directed graph $G = (O, E, \delta, d)$ called L's dependence graph. G's vertex set is O. To each vertex op of G we associate its duration $\delta(op)$. G's edge set E must verify the following two requirements:

1. If $e = (op, op') \in E$ then $d(op, op') < \infty$. The weight of e is set to $d(op, op')$.

2. $\forall\, op, op' \in O \quad d(op, op') < \infty \;\Rightarrow\; \exists$ path P from op to op' s.t. $d(P) = d(op, op')$.

Note that E is not necessarily unique. The edge set

$$E = \{(op, op') : d(op, op') < \infty\}$$

has the biggest cardinality, whereas the edge set

$$E = \{(op, op') : d(op, op') < \infty \text{ and } \nexists\, op'' \quad d(op, op'') + d(op'', op') \leq d(op, op')\}$$

is the edge set with the smallest cardinality. By computing the all pairs shortest paths of G it is easy to transform the original edge set into the one with the least amount of edges. This step can be implemented in $O(n\,m + n^2 \log n)$ time, where n and m respectively denote the cardinality of G's vertex and edge set [7].

For cyclic systems one is interested in generating regular schedules which can be finitely encoded. We introduce the notion of a periodic schedule.

Definition 4 *Let $L = (O[0,\infty], \delta, \prec)$ be some cyclic system, σ an admissible p-schedule for L and $l \geq 1$, $\tau_{ii} \geq 0$ respectively an integer and a rational. The schedule σ is said to be (l, τ_{ii})-periodic for L if and only if*

$$\forall\, op \in O \quad \forall\, i \geq 0 \qquad \sigma(op[i]) = \sigma(op[i \bmod l]) + \tau_{ii} \cdot \left\lfloor \frac{i}{l} \right\rfloor$$

The numbers l and τ_{ii} are respectively called the unfolding and initiation interval of σ. The asymptotic performance of σ, denoted $\|\sigma\|$, is defined as $\|\sigma\| = \tau_{ii}/l$.

Note that a (l, τ_{ii})-periodic schedule is perfectly determined by the initiation interval τ_{ii} and the time in which operations in the first l iterations start executing.

When there is an unbounded number of processors, that is $p = \infty$, we show in the next section how to efficiently construct a periodic schedule with asymptotically optimum performance. When $p < \infty$ let $\|L\|_p$ denote the p-optimum asymptotic performance for the input cyclic task system L. Because of Lenstra & Rinnooy Kan's result [19] it will be shown in section 5 that no periodic p-schedule σ admissible for L and satisfying

$$\frac{\|\sigma\|}{\|L\|_p} < 4/3$$

can be constructed in polynomial time unless P=NP. We will however provide an efficient algorithm that generates periodic p-schedules at most a factor $(2-1/p)+(p-1)/p \cdot \delta_{max}/\|L\|_p$ away from $\|L\|_p$.

4 Achieving Optimality for Infinite Processor Machines

In this section we examine the problem of generating periodic rational schedules which are asymptotically optimum when the number of processors p is infinite. We provide a very simple and efficient algorithm.

Algorithm 1

Input: A cyclic task system L represented by a dependence graph $G = (O, E, \delta, d)$.

Output: A $(1, \rho_G)$-periodic rational schedule σ for L with optimum asymptotic performance.

Method:

1. Compute ρ_G, the maximum duration-to-weight ratio of the cycles in G, with any of the algorithms mentioned in the introduction. This step takes $O(\mathcal{T})$ time.
2. Associate to each edge $e = (op, op') \in E$ the length $\tau(e) = \delta(op) - \rho_G \cdot d(e)$. Add a vertex s to G and the edges (s, op) for each $op \in O$. Set the length of these edges to 0. Because of the definition of ρ_G no cycle in G has positive length with respect to τ. Thus for each $op \in O$ the longest path from s to op, denoted $\tau(s, op)$, is well defined. These longest paths are produced as side results of Karp or Young, Tarjan & Orlin algorithms. Otherwise they can be computed in $O(n\,m)$ time [7].
3. For all $op \in O$ and $i \geq 0$ set $\sigma(op[i]) = \tau(s, op) + \rho_G \cdot i$.

The overall algorithm requires $O(\mathcal{T})$ time if Karp's or Young, Tarjan & Orlin's algorithms are used to compute ρ_G and $O(n\,m + \mathcal{T})$ time otherwise.

Theorem 1 *The schedule σ generated by algorithm 1 is an admissible $(1,\rho_G)$-periodic schedule for L, the cyclic task system represented by G. Furthermore σ is asymptotically optimum.*

Proof: It is fairly obvious that σ is $(1,\rho_G)$-periodic. Let C be a cycle in G comprising edges $(op_1, op_2), \ldots, (op_c, op_1)$. Then

$$\forall\, i \geq 0 \quad op_1[i] \prec op_2[i + d(op_1, op_2)] \prec \cdots \prec op_1[i + d(C)]$$

thus at most $d(C)$ iterations can be executed every $\delta(C)$ cycles and consequently no schedule can have an asymptotic performance better than ρ_G. It remains to show that σ is admissible for L, the cyclic task system represented by G. This is true if and only if

$$\forall\, e = (op, op') \in E \quad \forall\, i \geq 0 \quad \sigma(op'[i + d(e)]) - \sigma(op[i]) \geq \delta(op)$$

This is certainly true since $\sigma(op'[i + d(e)]) - \sigma(op[i]) = \tau(s, op') + \rho_G \cdot d(e) - \tau(s, op)$ and $\tau(s, op') - \tau(s, op) \geq \underbrace{\delta(op) - \rho_G \cdot d(e)}_{=\tau(e)}$ by virtue of the longest path inequalities. $\Box$

If operation durations are integral and the schedule is also required to be integral, algorithm 1 continues to work if in steps 2 and 3 one replaces ρ_G with $\lceil \rho_G \rceil$ that is one sets $\tau(e) = \delta(op) - \lceil \rho_G \rceil \cdot d(e)$ in step 2 and $\sigma(op[i]) = \tau(s, op) + \lceil \rho_G \rceil \cdot i$ in step 3. The proof that the generated schedule is $(1, \lceil \rho_G \rceil)$-periodic and admissible for G is unchanged and the ratio to the asymptotic optimum performance is clearly $\lceil \rho_G \rceil / \rho_G$. To generate asymptotically optimum integral schedules we introduce the notion of unrolling.

Definition 5 *Given a dependence graph $G = (O, E, \delta, d)$ and a positive integer l one defines the dependence graph $G^l = (O^l, E^l, \delta, d^l)$, where*

1. $O^l = \{op[i] \,:\, op \in O$ *and* $0 \leq i < l\}$
2. $E^l = \{(op[i], op'[j]) \,:\, e = (op, op') \in E$ *and* $j = (i + d(e)) \bmod l\}$
3. *For* $e = (op[i], op'[j]) \in E^l$, $d^l(e) = \left\lfloor \dfrac{i + d(op, op')}{l} \right\rfloor$

The graph G^l is said to be obtained by unrolling G l times.

The key result concerning G^l is

Theorem 2 *Let G be some dependence graph and l a positive integer than $\rho_{G^l} = l\,\rho_G$.*

Proof: Let C be some cycle in G comprising edges $e_1 = (op_1, op_2)$, $e_2 = (op_2, op_3)$, $\cdots$, $e_c = (op_c, op_1)$. Consider the cycle C^l in G^l going through operations

$$op_1[0],\ op_2[d(e_1) \bmod l],\ op_3[\underbrace{((d(e_1) \bmod l) + d(e_2)) \bmod l}_{=(d(e_1)+d(e_2)) \bmod l}],\ \cdots,\ op_c[(\textstyle\sum_{i=1}^{c-1} d(e_i)) \bmod l],$$

$$op_1[d(C) \bmod l],\ \cdots,\ op_c[(d(C) + \textstyle\sum_{i=1}^{c-1} d(e_i)) \bmod l],\quad \cdots,$$

$$op_1[((l-1)\cdot d(C)) \bmod l],\ \cdots,\ op_c[((l-1)\cdot d(C) + \textstyle\sum_{i=1}^{c-1} d(e_i)) \bmod l], op_1[(l \cdot d(C)) \bmod l]$$

Clearly $\delta(C^l) = l \cdot \delta(C)$, furthermore because for any two integers a and b

$$\left\lfloor \frac{a}{l} \right\rfloor + \left\lfloor \frac{(a \bmod l) + b}{l} \right\rfloor = \left\lfloor \frac{a+b}{l} \right\rfloor$$

we have $d^l(C^l) = \lfloor (l \cdot d(C))/l \rfloor = d(C)$ and therefore $\delta(C^l)/d^l(C^l) = l \cdot \delta(C)/d(C)$ which in turns implies $\rho_{G^l} \geq l \cdot \rho_G$.

Conversely consider a cycle C^l in G^l traversing operations $op_1[i_1]$, $\cdots$, $op_c[i_c]$, $op_1[i_1]$ where

$$1 \leq j \leq c \qquad i_{1+j \bmod c} = (i_j + d(op_j, op_{1+j \bmod c})) \bmod l$$

The corresponding cycle C in G which goes through operations op_1, $\cdots$, op_c, op_1 is such that $\delta(C) = \delta(C^l)$. In addition

$$d^l(C^l) = \sum_{j=1}^{c} \left\lfloor \frac{i_j + d(op_j, op_{1+j \bmod c})}{l} \right\rfloor$$

Because $i_{1+j \bmod c} = (i_j + d(op_j, op_{1+j \bmod c})) \bmod l$, $d(C)$ must be a multiple of l and $d^l(C^l) = d(C)/l$ which in turn implies $l \cdot \rho_G \geq \rho_{G^l}$. $\Box$

It follows that to generate an asymptotically optimum *integral* schedule σ' when operation durations are integral it suffices to apply algorithm 1 to G^{l_G}, rather than G, where l_G is the denominator of ρ_G in its irreducible form. More precisely the schedule σ' is defined as

$$\forall\, op \in O \quad \forall\, i \geq 0 \qquad \sigma'(op[i]) = \tau(s, op[i \bmod l_G]) + \rho_{G^{l_G}} \cdot \left\lfloor \frac{i}{l_G} \right\rfloor$$

Because $\rho_{G^{l_G}}$ is an integer, σ' is an integral schedule. The proof that σ' is $(l_G, \rho_{G^{l_G}})$-periodic and admissible for the input cyclic task L is a straightforward generalization of the proof in theorem 1. The overall time to generate σ' is $O(n\, m\, l_G^2 + \mathcal{T})$.

5 Approximating Optimality for Finite Processor Machines

When the number of available processors p is finite even the problem of generating periodic schedules whose performance is less than a factor $4/3$ from the asymptotic optimum becomes NP-hard. This can be seen as follows. Let T be some acyclic task system. Create the cyclic task system L where each iteration has the same operations and dependences as the task system T. In addition each iteration contains a special serializing operation op_0 which depends on all other operations in the iteration and on which every operation in the next iteration depends. Let $||L||_p$ denote the optimum asymptotic performance of L when p processors are available and assume one could create in polynomial time a periodic schedule σ, admissible for L, such that $||\sigma||/||L||_p < 4/3$. Then by taking the starting time of the operations scheduled before $op_0[0]$ it would be possible to generate, in polynomial time, a schedule for the finite task system T which has a length less than a factor $4/3$ from the p-optimum, and this, as we mentioned at the end of section 2 is possible only if P = NP [19].

As the previous reduction shows every polynomial time approximation algorithm for the cyclic scheduling problem can be used to approximate optimality in the acyclic case. The best such algorithm for the acyclic case guarantees a factor of $(2 - 1/p)$ from p-optimality [5]. Given a cyclic task system L, the goal of this section will therefore be that of generating a periodic p-schedule σ, admissible for L, such that $||\sigma||/||L||_p$ is, in the worst case, as close as possible to $(2 - 1/p)$.

The algorithm which follows is simple, runs in $O(\mathcal{T})$ time and guarantees a maximum factor from asymptotic optimality of $(2 - 1/p) + (p - 1)/p \cdot \delta_{max}/||L||_p$. A better performance bound of $(2 - 1/p) + (p - 1)/p \cdot \delta_{max}/(l\,||L||_p)$, where l is a user selected parameter, can be obtained with an additional $O(n\,m\,l^2)$ cost in time complexity. If operation durations are integral and the generated schedule is also required to be integral a slight modification of the algorithm guarantees a performance bound of $1+(p-1)/p\cdot((\delta_{max}-1)+\lceil ||L||_p \rceil)/||L||_p$ times the optimum.

The strategy adopted is to transform the input dependence graph G into an acyclic dependence graph G' and invoke a list scheduling algorithm on G' to construct the periodic schedule. The dependence graph G' is obtained by deleting edges from G. The difficult part when cutting edges is to shorten dependence paths as much as possible while preserving admissibility.

Algorithm 2

Input: A cyclic task system L represented by a dependence graph $G = (O, E, \delta, d)$ and a number of processors $p < \infty$.

Output: An admissible periodic rational p-schedule σ for L with unfolding 1 such that $\|\sigma\|/\|L\|_p \leq (2 - 1/p) + (p-1)/p \cdot \delta_{max}/\|L\|_p$.

Method:

1. Do steps 1 and 2 of algorithm 1.
2. For any two rational numbers λ, μ define $(\lambda \bmod \mu) = \lambda - \lfloor \lambda/\mu \rfloor \cdot \mu$. Delete an edge $e = (op, op')$ in G if and only if

$$\tau(s, op') \bmod \rho_G < \tau(s, op) \bmod \rho_G + \delta(op)$$

The resulting graph G' is acyclic (see lemma 1). This step takes $O(m)$ time.

3. Using any list scheduling algorithm generate a p-schedule σ_a admissible for the acyclic task system represented by the dependence graph G'. This step takes $O(m + n \log n)$ time.
4. For all $op \in O$ and $i \geq 0$ set $\sigma(op[i]) = \sigma_a(op) + |\sigma_a| \cdot (i + \lfloor \tau(s, op)/\rho_G \rfloor)$.

The overall algorithm requires $O(m + n \log n + \mathcal{T})$ time if Karp's or Young, Tarjan & Orlin's algorithms are used to compute ρ_G and otherwise $O(n\, m + \mathcal{T})$ time.

The first step in proving the correctness of algorithm 2 is to show that schedule σ_a is well defined, that is G' is acyclic. As a side result we bound the length of the longest path in G'.

Lemma 1 *The graph G' obtained in step 2 of algorithm 2 is acyclic. Furthermore*

$$\max_{\substack{P \text{ path} \\ \text{of } G'}} \delta(P) < \rho_G + \delta_{max}$$

Proof: Let C be a cycle in G comprising edges $(op_1, op_2), \ldots, (op_c, op_1)$. Suppose that no edge is deleted from C. Then

$$\forall\, 1 \leq i \leq c \quad \tau(s, op_i) \bmod \rho_G + \delta(op_i) \leq \tau(s, op_{1+i \bmod c}) \bmod \rho_G$$

Thus $(\tau(s, op_1) \bmod \rho_G) + \delta(C) \leq (\tau(s, op_1) \bmod \rho_G)$, a contradiction.

For the second claim let P be a path of G' comprising edges $(op_1, op_2), \ldots, (op_{c-1}, op_c)$ and suppose that $\rho_G \leq \sum_{i=1}^{c-1} \delta(op_i)$. Then because of the previous inequalities one can write

$$\rho_G \leq \tau(s, op_1) \bmod \rho_G + \sum_{i=1}^{c-1} \delta(op_i) \leq \tau(s, op_c) \bmod \rho_G$$

a contradiction. Thus $\delta(P) < \rho_G + \delta_{max}$. □

The previous lemma shows that algorithm 2 does indeed unambiguously generate a periodic schedule. It remains to prove its admissibility for the input cyclic task system L.

Theorem 3 *For any cyclic task L and any number of processors p, the output of algorithm 2 is a periodic p-schedule admissible for L.*

Proof: The schedule σ generated by algorithm 2 is clearly periodic. Furthermore no more than p operations are being processed at any given moment because the acyclic schedule σ_a created in step 3 is a p-schedule and σ's initiation interval is $|\sigma_a|$. It remains to show that σ is admissible for L. This is true if and only if

$$\forall\, e = (op, op') \in E \quad \forall\, i \geq 0 \qquad \sigma(op[i]) + \delta(op) \leq \sigma(op'[i + d(e)])$$

Because of the definition of σ the previous inequality can be rewritten as

$$\sigma_a(op) + |\sigma_a| \cdot \left\lfloor \frac{\tau(s, op)}{\rho_G} \right\rfloor + \delta(op) \leq \sigma_a(op') + |\sigma_a| \cdot \left(d(e) + \left\lfloor \frac{\tau(s, op')}{\rho_G} \right\rfloor \right)$$

As in theorem 1 the proof that the above inequality holds relies on the longest paths inequality

$$\tau(s, op) + \delta(op) - \rho_G \cdot d(e) \leq \tau(s, op')$$

There are two cases two consider depending on whether edge $e = (op, op')$ has been deleted by algorithm 2 in step 2. If e has not been deleted then

$$\sigma_a(op) + \delta(op) \leq \sigma_a(op')$$

furthermore by dividing the longest path inequality by ρ_G and taking the floor one derives the following inequality:

$$\left\lfloor \frac{\tau(s, op)}{\rho_G} \right\rfloor \leq d(e) + \left\lfloor \frac{\tau(s, op')}{\rho_G} \right\rfloor$$

and therefore if e has not been deleted the dependence constraint is respected in σ. If on the contrary edge e has been deleted in step 2 then it must be that

$$\tau(s, op') \bmod \rho_G < \tau(s, op) \bmod \rho_G + \delta(op)$$

Thus by rewriting the longest path inequality as:

$$\left\lfloor \frac{\tau(s, op)}{\rho_G} \right\rfloor \cdot \rho_G + \tau(s, op) \bmod \rho_G + \delta(op) - \rho_G \cdot d(e) \leq \left\lfloor \frac{\tau(s, op')}{\rho_G} \right\rfloor \cdot \rho_G + \tau(s, op') \bmod \rho_G$$

one derives

$$\left\lfloor \frac{\tau(s, op)}{\rho_G} \right\rfloor < d(e) + \left\lfloor \frac{\tau(s, op')}{\rho_G} \right\rfloor$$

and therefore

$$\sigma_a(op) + \delta(op) - \sigma_a(op') \leq |\sigma_a| \leq |\sigma_a| \cdot \left(d(e) + \left\lfloor \frac{\tau(s, op')}{\rho_G} \right\rfloor - \left\lfloor \frac{\tau(s, op)}{\rho_G} \right\rfloor \right)$$

which shows that the dependence constraint is as well respected in σ if $e = (op, op')$ has been deleted. $\square$

It remains to bound the performance of σ.

Theorem 4 *For any cyclic task L and any number of processors p, the p-schedule σ generated by algorithm 2 is such that*

$$\frac{\|\sigma\|}{\|L\|_p} < 2 - \frac{1}{p} + \frac{p-1}{p} \cdot \frac{\delta_{max}}{\|L\|_p}$$

Proof: The optimum asymptotic performance $\|L\|_p$ is clearly bounded by the duration of operations in O, the number of available processors p and ρ_G:

$$\frac{1}{p} \cdot \sum_{op \in O} \delta(op) \leq \|L\|_p \qquad \rho_G \leq \|L\|_p$$

Let ϕ denote the overall time in σ_a when no more than $p-1$ processors are busy. Because σ_a is generated by a list scheduling algorithm there must exist a dependence path P in G' such that $\phi \leq \delta(P)$. Furthermore lemma 1 implies

$$\phi < \rho_G + \delta_{max} \leq \delta_{max} + \|L\|_p$$

Thus

$$\begin{aligned} p \cdot |\sigma_a| &\leq \sum_{op \in O} \tau(op) + (p-1) \cdot \phi \\ &< p \cdot \|L\|_p + (p-1) \cdot (\delta_{max} + \|L\|_p) \end{aligned}$$

$$\Rightarrow \quad \frac{|\sigma_a|}{\|L\|_p} < 2 - \frac{1}{p} + \frac{p-1}{p} \cdot \frac{\delta_{max}}{\|L\|_p}$$

□

Note that if $\|L\|_p$ is small and δ_{max} is big, the previous performance bound may be poor. To improve it, it suffices to unroll G until a satisfactory bound can be guaranteed. More precisely if for some $l \geq 1$, algorithm 2 is to operate on G^l rather the G the bound on performance becomes

$$\frac{|\sigma|}{\|L\|_p} < 2 - \frac{1}{p} + \frac{p-1}{p} \cdot \frac{\delta_{max}}{l \cdot \|L\|_p}$$

This improved bound comes at an additional $O(n\, m\, l^2)$ cost in running time complexity.

If operation durations are integral and the generated schedule is also required to be integral, algorithm 2 continues to work if, as in section 4, one replaces ρ_G with $\lceil \rho_G \rceil$. Let σ' be the integral schedule generated. The bound on optimum performance becomes

$$\frac{|\sigma'|}{\|L\|_p} \leq 1 + \frac{p-1}{p} \cdot \frac{(\delta_{max} - 1) + \lceil \|L\|_p \rceil}{\|L\|_p}$$

References

[1] R. K. Ahuja and J. B. Orlin, *New Scaling Algorithms for Assignment and Minimum Cycle Mean Problems*, Tech. Rep. Sloan Working Paper 2019-88, M.I.T., 1988.

[2] F. Allen, M. Burke, P. Charles, R. Cytron, and J. Ferrante, *An overview of the PTRAN analysis system for multiprocessing*, Journal of Parallel and Distributed Computing, 5 (1988), pp. 617–640.

[3] R. Allen and K. Kennedy, *Automatic translation of Fortran programs to vector form*, ACM Transactions on Programming Languages and Systems, 9 (1987), pp. 491–542.

[4] P. Chretienne, *The basic cyclic scheduling problem with deadlines*, Discrete Applied Mathematics, 30 (1991), pp. 109–123.

[5] E. G. Coffman, *Computer and Job-shop Scheduling Theory*, John Wiley and Sons, New York, New York, 1976.

[6] E. G. Coffman and R. L. Graham, *Optimal scheduling for two processor systems*, Acta Informatica, 1 (1972), pp. 200–213.

[7] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, MIT Press and Mc Graw Hill, 1990.

[8] R. Cytron, *Doacross: beyond vectorization for multiprocessors*, in Proceedings of the 1985 International Conference on Parallel Processing (Penn State University, Pennsylvania), IEEE and ACM, Silver Spring, Maryland, Aug. 1986, pp. 836–844.

[9] H. N. Gabow, *An almost-linear algorithm for two-processor scheduling*, Journal of the ACM, 29 (1982), pp. 766–780.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability - A Guide to the Theory of NP-Completeness*, Freeman, New York, New York, 1979.

[11] M. Gondran and M. Minoux, *Graphs and Algorithms*, Wiley, 1984.

[12] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey*, vol. 5 of Annals of Discrete Mathematics, North Holland Publishing Company, 1979, pp. 287–326.

[13] N. S. Grigor'yeva, I. S. Latypov, and I. V. Romanovskii, *Cyclic problems of scheduling theory*, Tekhnicheskaya Kibernetika, (1988), pp. 3–11. English translation.

[14] K. Iwano and S. Yeh, *An efficient algorithm for optimal loop parallelization*, in International Symposyum on Algorithms, Springer-Verlag, Aug. 1990, pp. 201–210. Lecture Notes in Computer Science 450.

[15] R. M. Karp, *A Characterization of the Minimum Cycle Mean in a Digraph*, vol. 23 of Discrete Mathematics, North Holland Publishing Company, 1978, pp. 309–311.

[16] M. Lam, *Software pipelining: an effective scheduling technique for VLIW machines*, in Proceedings of the SIGPLAN 1988 Conference on Programming Language Design and Implementation (Atlanta, Georgia), ACM, June 1988, pp. 318–328.

[17] L. Lamport, *The parallel execution of DO loops*, Communications of the ACM, 17 (1974), pp. 83–93.

[18] E. L. Lawler, *Optimal cycles in doubly weighted directed linear graphs*, in Theory of Graphs-International Symposyum, P. Rosenstiehl, ed., Gordon and Breach, Rome 1966, pp. 209–213.

[19] J. K. Lenstra and A. H. G. Rinnooy Kan, *Complexity of scheduling under precedence constraints*, Operations Research, 26 (1978), pp. 22–35.

[20] A. Munshi and B. Simons, *Scheduling loops on processors: algorithms and complexity*, SIAM Journal of Computing, 19 (1990), pp. 728–741.

[21] D. A. Padua and M. J. Wolfe, *Advanced compiler optimizations for supercomputers*, Communications of the ACM, 29 (1986), pp. 1184–1201.

[22] R. Reiter, *Scheduling parallel computations*, Journal of the ACM, 15 (1968), pp. 590–599.

[23] N. E. Young, R. E. Tarjan, and J. B. Orlin, *Faster parametric shortest path and minimum-balance algorithms*, Networks, 21 (1991), pp. 205–221.