Tight Timing Estimation With the Newton-Gregory Formulae*

Robert van Engelen, Kyle Gallivan, and Burt Walsh
Department of Computer Science
and School of Computational Science and Information Technology
Florida State University
Tallahassee, FL 32306-4530
{*engelen,gallivan,walsh*}*@cs.fsu.edu*

Abstract

Parametric worst-case execution time (WCET) bounds are critical in removing restrictions, such as known loop bounds, on algorithms for important applications such as scheduling for real-time embedded systems. Current parametric approaches have difficulties with loop nests that include non-rectangular loops, zero-trip loops, and/or loops with non-unit strides. This paper presents a novel approach to parametric WCET estimation based on numeric/symbolic manipulation of polynomial representations for the timing of rectangular and non-rectangular loop nests including those with zero-trip loops, non-unit strides, and multiple critical paths.

1. Introduction

Worst-case execution time (WCET) estimation has important applications in scheduling and real-time embedded systems. Traditional timing analysis statically determines the maximum execution time of a program, which requires the maximum number of loop iterations to be statically known. Recent work on WCET estimation [7] incorporated work on scheduling [10] to bound the number of iterations of non-rectangular loops to derive WCET estimates. The method requires the bounds of the outer loop to be constant or to be bounded by constants, resulting in a loosening of the WCET bound.

Parametric timing analysis produces formulae, where the unknown values affecting the execution are parameterized [12]. These formulae can be quickly evaluated at run-time with real-time values to support dynamic scheduling decisions. However, current parametric approaches have difficulties with multiple loop nests that include non-rectangular loops, zero-trip loops, and/or loops with non-unit strides.

To count the number of loop iterations in the presence of potential zero-trip loops, Haghighat [5] proposes a method that exploits sums over special functions that are similar to the Dirac delta operator. Sakellariou [10] improves this and earlier work by Clauss on Ehrhart polynomials [3] by proposing the use of guarded sums. Unfortunately, the determination of the size of loop iteration spaces with the guarded sums proves impossible for certain complicated loop nests with symbolic loop bounds and non-unit strides. Another approach by Pugh uses Presburger formulas [8] to determine the iteration space size of a loop nest.

This paper presents a novel approach to parametric WCET estimation for rectangular and non-rectangular loop nests including those with zero-trip loops, non-unit strides, and multiple critical paths in the loop body. The method requires only very simple numeric/symbolic manipulation capabilities on standardized representations of polynomials. This paper significantly extends the framework that we first proposed in [11]. This includes the analysis of multiple critical iteration paths through the body of a loop.

The remainder of this paper is organized as follows. Section 2 introduces loop timing estimation using the Newton-Gregory formulae. A method for loop bounds analysis based on monotonic Newton series is presented in Section 3. Section 4 describes a technique to estimate the timing of critical loop iteration paths. Finally, some concluding remarks are given in Section 5.

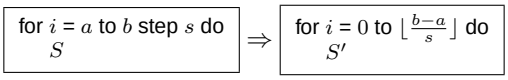
2. Loop Timing Estimation

This section introduces the timing estimation approach using the Newton-Gregory formulae. Most of the execution time of a typical program is spent in loops. Therefore, tight timing estimation of loop nests is crucial to estimate the total execution time of a program or time-critical task.

* Supported in part by NSF grants CCR-9904943, CCR-0105422, CCR-0208892, EIA-0072043, and DOE grant 20026126.

2.1. Bounding the Execution Time of Loops

To simplify the analysis of the timing of loop nests, loops in loop nests are normalized first if not normalized already. That is:



where the loop body statements S are adjusted S' such that all in-scope occurrences of i are replaced by $s * i + a$.

The total real execution time ω of a loop can be expressed as an accumulation of the real execution times of the individual operations in the loop, consisting of the loop header H and body S' operations

$$\omega = \omega(H(0)) + \sum_{i=0}^{\lfloor \frac{b-a}{s} \rfloor} \omega(S'(i)) + \omega(H(i+1)) \quad , \quad (1)$$

where $\omega(H(0))$ denotes the real start-up time of the loop header at iteration $i = 0$, $\omega(S'(i))$ denotes the real execution time of the loop body S' at iteration i, and $\omega(H(i+1))$ denotes the real execution time of the loop header operations required to commence with the next iteration or terminate the loop when $i + 1$ exceeds the loop bound.

The total real execution time ω of the loop can be bounded given sufficiently tight upper bounds on the real execution times of the loop operations H and S'

$$\begin{aligned} \omega(H(0)) &\leq c_0 \\ \omega(S'(i)) + \omega(H(i+1)) &\leq p(i) \quad , \end{aligned}$$

where c_0 is a constant and $p(i) \geq 0$ for $i = 0, \ldots, \lfloor \frac{b-a}{s} \rfloor$ is a bounding function. Given a sufficiently tight bounding constant c_0 and bounding function p, the real total execution time ω of the loop is bounded by

$$\omega \leq c_0 + \sum_{i=0}^{\lfloor \frac{b-a}{s} \rfloor} p(i) \quad . \quad (2)$$

The sums in (1) and (2) are only valid when $\lfloor \frac{b-a}{s} \rfloor \geq 0$, because a sum with negative bound requires evaluating $p(i)$ for negative values of i. In general, sums form a simple and attractive notation for loop iteration count problems, but their semantics provide a poor vehicle to accurately model loop iteration count problems. The sum in (2) is undefined for zero-trip loops because the total size of the iteration space of a loop is given by $\lfloor \frac{b-a}{s} \rfloor + 1$, which can be zero or even negative. Furthermore, because the sum operator satisfies $\sum_{i=a}^{b} f(i) = -\sum_{i=b}^{a} f(i)$, limiting the lowest value of the upper bound of the sum to -1, as in $\sum_{i=0}^{\max(-1, \lfloor \frac{b-a}{s} \rfloor)} p(i)$, still requires evaluating $p(-1)$.

2.2. Newton-Gregory Interpolating Polynomials

When the loop body S' includes non-constant time operations such as inner loops with bounds that can be expressed as polynomials in i, then the upper bound function p on the execution time of the outer loop header and body operations is polynomial. Because inner loops are commonly bounded by (symbolic) constants or affine functions of the outer loop counter variables, function p is polynomial of finite order k, where k is a relatively small constant. It follows, from the Newton-Gregory forward formula for the interpolating polynomial, that there exist coefficients ϕ_j, $j = 0, \ldots, k$, such that

$$p(i) = \sum_{j=0}^{k} \phi_j \binom{i}{j} \quad . \quad (3)$$

The coefficients ϕ_j form the Newton series of the polynomial p. The Newton series representation of a polynomial proves to be particularly useful to represent loop iteration count problems. The bound (2) requires a sum over an iteration space that is potentially empty. Given the Newton series of polynomial p, it follows from (3) that the sum can be written as

$$\sum_{i=0}^{n-1} p(i) = \sum_{j=0}^{k} \phi_j \binom{n}{j+1} \quad (4)$$

for all $n > 0$, since

$$\sum_{i=0}^{n-1} \sum_{j=0}^{k} \phi_j \binom{i}{j} = \sum_{j=0}^{k} \phi_j \sum_{i=0}^{n-1} \binom{i}{j} = \sum_{j=0}^{k} \phi_j \binom{n}{j+1} \quad .$$

Therefore, a loop iteration count problem that involves the evaluation of symbolic loop bounds can be translated into a simpler summation problem with constant bounds, i.e. the constant degree k of the polynomial p. More importantly, the formulation is applicable to zero-trip loops, because

$$\sum_{j=0}^{k} \phi_j \binom{0}{j+1} = 0 \quad (5)$$

by the properties of the binomial coefficients. Equations (4) and (5) provide the basic requirements to define a loop timing function.

Definition 2.1 *Let $\Phi(i) = \langle \phi_0, \ldots, \phi_k \rangle_i$ denote the Newton series of a polynomial in i. The* sum reduction *$\sigma(\Phi(i), n)$ of $\Phi(i)$ over a domain $i = 0, \ldots, n-1$ of size $n \geq 0$ is the polynomial defined by*

$$\sigma(\Phi(i), n) \stackrel{\text{def}}{=} \sum_{j=0}^{k} \phi_j \binom{n}{j+1} \quad .$$

The sum reduction $\sigma(\Phi(i), n)$ of a Newton series $\Phi(i)$ is a polynomial in n. This important property of sum reduc-

tions enables the composition of σ functions over polynomials represented by Newton series to compute the size of multi-dimensional iteration spaces of nested loops.

The sum reduction of a Newton series is related to the Riemann approximation of the integral

$$\sigma(\Phi(i), x) \approx \int_0^x p(y)\,\mathrm{d}y$$

However, the function σ determines the exact size of the iteration space of a loop by exploiting the discrete formulation of Newton-Gregory interpolating polynomials.

Note that $\sigma(\Phi(i), x)$ can be evaluated for any (symbolic) real value $x \geq 0$, because the binomial coefficients are defined by

$$\binom{x}{j+1} = \frac{1}{(j+1)!}x(x-1)(x-2)\cdots(x-j) \quad .$$

This combined with the fact that polynomial $p(i) \geq 0$ bounds the real execution time of the loop header and body operations for all iterations $i = 0, \ldots, n-1$, leads to the observation that $\sigma(\Phi(i), n)$ is monotonically increasing with increasing iteration space size $n \geq 0$. Therefore, non-unit loop strides can be incorporated in the σ function by noting that

$$\sigma(\Phi(i), \lfloor \tfrac{b-a}{s} \rfloor + 1) \leq \sigma(\Phi(i), \tfrac{b-a}{s} + 1)$$

if $(b-a)/s + 1 \geq 0$, with equality when s evenly divides $b-a$. When the remainder $(b-a) \bmod s$ is small, the bound is tight.

Definition 2.2 *The worst-case execution time (WCET) bound of a (possibly zero-trip) loop with iteration $i = a, \ldots, b$ and stride $s \neq 0$ is*

$$WCET = c_0 + \sigma(\Phi(i), \max(0, \tfrac{b-a}{s} + 1))$$

which bounds the real execution time ω of the loop, where $\Phi(i)$ is the Newton series representation of the polynomial $p(i)$ over $i = 0, \ldots, \lfloor \frac{b-a}{s} \rfloor$ that bounds the execution time of the normalized loop header and body operations $\omega(S'(i)) + \omega(H(i+1)) \leq p(i)$, and c_0 bounds the initial loop header execution time $\omega(H(0)) \leq c_0$.

When the loop bounds a and/or b are symbolic, the WCET estimation is parameterized.

2.3. Newton Series Conversions

The Newton series of a polynomial is unique and can be efficiently computed. It is noteworthy to mention that the coefficients of the Chains of Recurrences (CR) pure-sum representation of a polynomial are equivalent to the coefficients of the Newton series of the polynomial [1, 13]. Therefore, the CR algebra construction rules for polynomials can be applied to e.g. the Horner form of the polynomial. However, a more efficient construction method exists that is based on Newton's triangle [1]. In this paper, the triangle is represented as a matrix to apply series conversions via matrix-vector products.

Given the coefficients p_j, $j = 0, \ldots, k$, of polynomial[1] $p(i) = p_0 + p_1 i + p_2 i^2 + \cdots + p_k i^k$, the $k+1$ Newton coefficients ϕ_j, $j = 0, \ldots, k$, can be directly obtained from the coefficients p_j of the polynomial using Newton's triangle. For $k = 3$, the Newton triangle in matrix form is

$$\mathbf{N}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 0 & 6 \end{pmatrix}$$

The Newton series $\Phi(i)$ of a polynomial is obtained by the matrix-vector product $\mathbf{N}_k p(i)$ of the Newton triangle $\mathbf{N}_k$ and the vector of polynomial coefficients $p(i) = [p_0, \ldots, p_k]_i$.

The Newton triangle is formed by a two-term recurrence [1]. The algorithm shown in Figure 1**a** computes the coefficients ϕ_j for p_j with $\mathcal{O}(k^2)$ operations while requiring $\mathcal{O}(k)$ temporary storage space. The algorithm computes the matrix-vector product $\Phi(i) = \mathbf{N}_k p(i)$, where $\mathbf{N}_k$ is the Newton triangle, $p(i) = [p_0, \ldots, p_k]_i$ is a vector of polynomial coefficients, and $\Phi(i) = \langle \phi_0, \ldots, \phi_k \rangle_i$ is a Newton series represented by a vector of Newton coefficients ϕ_j, $j = 0, \ldots, k$.

A similar two-term recurrence exists for the inverse Newton triangle $\mathbf{N}_k^{-1}$ required to compute the polynomial $p(i) = \mathbf{N}_k^{-1}\Phi(i)$ of a Newton series $\Phi(i)$. For $k = 3$, the inverse Newton triangle matrix is

$$\mathbf{N}_3^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -\frac{1}{2} & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \\ 0 & 0 & 0 & \frac{1}{6} \end{pmatrix}$$

An algorithm to compute the polynomial of a Newton series is shown in Figure 1**b**.

The inverse Newton triangle matrix has an important property.

Lemma 2.3 *Let $\sigma(\Phi(i), n)$ be the sum reduction of $\Phi(i)$ over domain $i = 0, \ldots, n-1$. Then, the coefficients $s_0, \ldots, s_{k+1}$ of the polynomial $s(n) = \sigma(\Phi(i), n)$ are given by the matrix-vector product*

$$s(n) = [s_0, \ldots, s_{k+1}]_n = \mathbf{N}_{k+1}^{-1}\langle 0, \phi_0, \ldots, \phi_k \rangle_i$$

1 The polynomial representations $p(i) = p_0 + p_1 i + p_2 i^2 + \cdots + p_k i^k$ and its vector form $p(i) = [p_0, p_1, \ldots, p_k]_i$ will be used interchangeably throughout this paper.

```
Input:  p[0 : k]
Output: φ[0 : k]
Declare array m[0 : k] of integer
for j = 0 to k do
    φ[j] := 0
    m[j] := 0
φ[0] := p[0]
if k > 0 then
    φ[1] := p[1]
    m[1] := 1
    for i = 2 to k do
        for j = i to 1 step −1 do
            m[j] := j * (m[j − 1] + m[j])
            φ[j] += m[j] * p[i]
```

a. Algorithm to compute $\Phi = \mathbf{N}_k p$

```
Input:  φ[0 : k]
Output: p[0 : k]
Declare array m[0 : k] of rational
for j = 0 to k do
    p[j] := 0
    m[j] := 0
p[0] := φ[0]
if k > 0 then
    p[1] := φ[1]
    m[1] := 1
    for i = 2 to k do
        for j = i to 1 step −1 do
            m[j] := (m[j − 1] − (i − 1)*m[j])/i
            p[j] += m[j]*φ[i]
```

b. Algorithm to compute $p = \mathbf{N}_k^{-1}\Phi$

Figure 1. Newton Series Conversion Algorithms

Therefore, the inverse Newton triangle can be used to efficiently compute sum reductions of Newton series.

To demonstrate the applicability of the Newton-Gregory approach to loop timing estimation, two examples are presented and the results are compared to existing worst-case execution time estimation techniques.

Example 2.4 *Consider the following loop nest with triangular iteration space:*

```
for I = 1 to N do
    for J = I to N do
        S
```

The normalized iteration space is

$$\begin{aligned} i &= 0, \ldots, N-1 \\ j &= 0, \ldots, N-1-i \quad . \end{aligned}$$

Suppose the following execution time bounds are obtained

$$\begin{aligned} \omega(H_1(0)) &\leq c_0 \\ \omega(H_2'(i,0)) &\leq c_1 \\ \omega(S'(i,j)) + \omega(H_2'(i,j+1)) &\leq c_2 \quad , \end{aligned}$$

where c_1, c_2, and c_3 are constants bounding the time of the loop header H_1 of the outer loop, the normalized loop header H_2' of the inner loop, and the normalized loop body S', respectively. Constant bounds c_0, c_1, and c_2 can be established with an algorithm that simulates the (cache) behavior of operations iteratively until the timing of the loop body converges, see e.g. [7, 12]. This simulation technique does not accurately handle the timing of (triangular) inner loops. The combination of the loop timing estimation presented in this paper and the simulation of non-looping statements by the iterative algorithm is particularly attractive.

The worst-case execution time of the two-dimensional loop nest is expressed by the composition of two σ functions and two Newton series conversions. The evaluation of this WCET estimation expression proceeds as follows

$$\begin{aligned} &WCET \\ &= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(\mathbf{N}_0 c_2, \max(0, N-i))), \max(0, N)) \\ &\quad (0 \leq i \leq N-1, \text{ so replace } \max(0, N-i) \text{ by } N-i) \\ &= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(\mathbf{N}_0 c_2, N-i)), \max(0, N)) \\ &= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(\langle c_2 \rangle_i, N-i)), \max(0, N)) \\ &= c_0 + \sigma(\mathbf{N}_1(c_1 + c_2 N - c_2 i), \max(0, N)) \\ &= c_0 + \sigma(\langle c_1 + c_2 N, -c_2 \rangle_i, \max(0, N)) \\ &= c_0 + c_1 + c_2(N + \tfrac{1}{2}) \max(0, N) - \tfrac{1}{2} c_2 \max(0, N)^2 \end{aligned}$$

Given constants c_0, c_1, and c_2 and (symbolic) N, the parameterized WCET of the loop nest is the tight upper bound $c_0 + c_1 + c_2(N + \frac{1}{2}) \max(0, N) - \frac{1}{2} c_2 \max(0, N)^2$. $\diamond$

Example 2.5 *Consider the following loop nest with a non-linear iteration space and a non-unit stride:*

```
for I = 1 to N do
    for J = I to I * I − 2 step 2 do
        S
```

The normalized iteration space is

$$\begin{aligned} i &= 0, \ldots, N-1 \\ j &= 0, \ldots, \lfloor \tfrac{i+i^2}{2} \rfloor - 1 \quad . \end{aligned}$$

Suppose the following execution time bounds are obtained

$$\begin{aligned} \omega(H_1(0)) &\leq c_0 \\ \omega(H_2'(i,0)) &\leq c_1 \\ \omega(S'(i,j)) + \omega(H_2'(i,j+1)) &\leq c_2 \quad , \end{aligned}$$

where c_0, c_1, and c_2 are constants.

The worst-case execution time of the two-dimensional loop nest is expressed by the composition of two σ functions and two Newton series conversions. The evaluation of

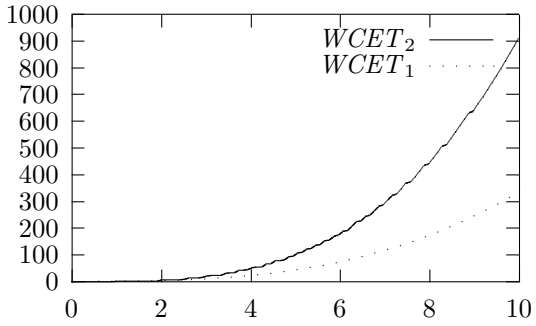


Figure 2. Comparison of WCET Estimations

this WCET estimation expression proceeds as follows

$$
\begin{aligned}
&WCET_1\\
&= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(\mathbf{N}_0 c_2, \max(0, \tfrac{i+i^2}{2}))), \max(0, N))\\
&\quad \textit{($i + i^2 \geq 0$, so replace } \max(0, \tfrac{i+i^2}{2}) \textit{ by } \tfrac{i+i^2}{2}\textit{)}\\
&= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(\mathbf{N}_0 c_2, \tfrac{i+i^2}{2})), \max(0, N))\\
&= c_0 + \sigma(\mathbf{N}_1(c_1 + \sigma(c_2, \tfrac{i+i^2}{2})), \max(0, N))\\
&= c_0 + \sigma(\mathbf{N}_1(c_1 + \tfrac{1}{2}c_2(i + i^2)), \max(0, N))\\
&= c_0 + \sigma(\langle c_1, c_2, c_2\rangle_i, \max(0, N))\\
&= c_0 + (c_1 - \tfrac{1}{6}c_2)\max(0, N) + \tfrac{1}{6}c_2 \max(0, N)^3
\end{aligned}
$$

Conventional worst-case execution time estimation techniques based on guarded sums [10], Ehrhart polynomials [3] or Presburger formulas [8] cannot handle this loop nest accurately because the inner loop bound is non-linear. These methods have to resort to computing overestimations by establishing the maximum size of the inner loop iteration space using the bounds $1 \leq I \leq N$. As a result, the inner loop's iteration space size is overestimated by $\lfloor \frac{N^2-N}{2} \rfloor$. Therefore, the resulting overestimated parameterized WCET expression is

$$WCET_2 = c_0 + c_1\, N + c_2\, N \lfloor \tfrac{N^2-N}{2} \rfloor \qquad (6)$$

for $N \geq 1$. This approach can result in very loose bounds for triangular loop nests, because the inner loop is bounded by a non-negative constant over the entire iteration space of the inner loop.

Figure 2 shows the worst-case execution time estimation bounds using Newton-Gregory ($WCET_1$) compared to the simple WCET (6) approach ($WCET_2$) for increasing values of N (shown on the x-axis). The sample bounds $c_0 = 1$, $c_1 = 1$, and $c_2 = 2$ are used in the graph. ◇

3. Loop Bounds Analysis

The Newton-Gregory approach to timing analysis is important not only to demonstrate that a potentially problematic loop iteration count problem that involves loops with symbolic bounds can be translated into a simpler summation problem with constant bounds, but also to facilitate accurate loop bound analysis. In many cases, the loop bound estimation enables the elimination of the limiting $\max$ operation from the WCET formulation given in Definition 2.2.

Bounds analysis is at the heart of tight timing analysis. The value range of a symbolic expression E is denoted by $[\mathcal{L}(E), \mathcal{U}(E)]$, where $\mathcal{L}(E)$ denotes the lower bound and $\mathcal{U}(E)$ denotes the upper bound of E, respectively. The value range of an expression is derived by the application of rules such as $\mathcal{L}(E_1 + E_2) = \mathcal{L}(E_1) + \mathcal{L}(E_2)$. Rules for arithmetic operators are mostly straightforward, see e.g. [2, 4, 6, 5, 9].

The range of values of a symbolic expression evaluated on an interval is of particular interest. The Newton series representation of a polynomial provides a means to accurately determine the value range of the polynomial on a given interval.

It is well known that when a function f is monotonically increasing on an interval $[a, b]$, the full range of values of f on the interval is given by $[f(a), f(b)]$. Similarly, when f is monotonically decreasing on $[a, b]$, then the range of f is $[f(b), f(a)]$.

Lemma 3.1 *Let $\Phi(i) = \langle \phi_0, \phi_1, \ldots, \phi_k \rangle_i$ denote the Newton series of a polynomial $p(i)$ in i and let $h(i) = \mathbf{N}_k^{-1} \langle \phi_1, \ldots, \phi_k \rangle_i$. If $h(i) \geq 0$ for all $i = 0, \ldots, n-2$, $n \geq 0$, then $p(i)$ is monotonically increasing on the interval $[0, n-1]$. Similarly, if $h(i) \leq 0$ for all $i = 0, \ldots, n-2$, then $p(i)$ is monotonically decreasing on the interval $[0, n-1]$.*

Proof. The polynomial $h(i)$ is constructed from the Newton series $\langle \phi_1, \ldots, \phi_k \rangle_i$ starting with the second coefficient of $\Phi(i)$. Therefore, by the Newton-Gregory formula we have

$$h(i) = \sum_{j=1}^{k} \phi_j \binom{i}{j-1}$$

which leads to

$$
\begin{aligned}
p(i) &= \sum_{j=0}^{k} \phi_j \binom{i}{j}\\
&= \phi_0 + \sum_{j=1}^{k} \phi_j \binom{i}{j}\\
&= \phi_0 + \sum_{j=1}^{k} \phi_j \sum_{\ell=0}^{i-1} \binom{\ell}{j-1}\\
&= \phi_0 + \sum_{\ell=0}^{i-1} \sum_{j=1}^{k} \phi_j \binom{\ell}{j-1}\\
&= \phi_0 + \sum_{\ell=0}^{i-1} h(\ell)
\end{aligned}
$$

for all $i \geq 1$ with $p(0) = \phi_0$. Because $h(i) \geq 0$ for all $i = 0, \ldots, n-2$, we observe that $\sum_{\ell=0}^{i-1} h(\ell)$ is monotonically increasing with increasing $i = 1, \ldots, n-1$ and therefore $p(i)$ is monotonically increasing with increasing $i = 0, \ldots, n-1$. The proof of the monotonically decreasing case is similar. □

The fact that monotonicity of a polynomial can be observed from its Newton series combined with the fact that the range of a monotonic function can be easily bounded leads to the following recursive definition of a lower and upper bound on the range of values of a polynomial.

Definition 3.2 *Let $\Phi(i) = \langle \phi_0, \ldots, \phi_k \rangle_i$ denote the Newton series of a polynomial evaluated on $i = 0, \ldots, n-1$ with $n \geq 0$. The lower bound of the polynomial defined by $\Phi(i)$ is*

$$\mathcal{L}(\Phi(i)) = \begin{cases} \mathcal{L}(\phi_0) & \text{if } \mathcal{L}(\langle \phi_1, \ldots, \phi_k \rangle_i) \geq 0 \\ \mathcal{L}(\mathbf{N}_k^{-1}\Phi(n-1)) & \text{if } \mathcal{U}(\langle \phi_1, \ldots, \phi_k \rangle_i) \leq 0 \\ \mathcal{L}(\mathbf{N}_k^{-1}\Phi(i)) & \text{otherwise} \end{cases}$$

and the upper bound of the polynomial defined by $\Phi(i)$ is

$$\mathcal{U}(\Phi(i)) = \begin{cases} \mathcal{U}(\phi_0) & \text{if } \mathcal{U}(\langle \phi_1, \ldots, \phi_k \rangle_i) \leq 0 \\ \mathcal{U}(\mathbf{N}_k^{-1}\Phi(n-1)) & \text{if } \mathcal{L}(\langle \phi_1, \ldots, \phi_k \rangle_i) \geq 0 \\ \mathcal{U}(\mathbf{N}_k^{-1}\Phi(i)) & \text{otherwise} \end{cases}$$

where $\mathbf{N}_k^{-1}\Phi(n-1)$ denotes the expression obtained by converting Φ to a polynomial that is evaluated at $i = n-1$, that is, by replacing i with $n-1$.

The conditions in the guards in this definition exploit the fact that when $\mathcal{L}(\Phi(i)) \geq 0$ on an interval the polynomial of $\Phi(i)$ is non-negative on the interval and when $\mathcal{U}(\Phi(i)) \leq 0$ then the polynomial is non-positive on the interval.

Lemma 3.3 *Let $\Phi(i)$ denote the Newton series of a polynomial $p(i)$ evaluated on $i = 0, \ldots, n-1$ with $n \geq 0$. Then,*

$$\mathcal{L}(\Phi(i)) \leq p(i) \leq \mathcal{U}(\Phi(i))$$

for all $i = 0, \ldots, n-1$.

The following example demonstrates that the bound is more tight compared to conventional value range analysis applied to polynomials.

Example 3.4 *Consider the polynomial $p(i) = \frac{1}{2}i^2 - \frac{1}{2}i$ with Newton series $\langle 0, 0, 1 \rangle_i = \mathbf{N}_2[0, -\frac{1}{2}, \frac{1}{2}]_i$. Given the domain $i = 0, \ldots, N$, the value range of this polynomial is $[\mathcal{L}(\langle 0, 0, 1 \rangle_i), \mathcal{U}(\langle 0, 0, 1 \rangle_i)] = [0, \frac{1}{2}N^2 - \frac{1}{2}N]$.* ◇

Conventional value range analysis methods applied to polynomials fail to establish accurate bounds [2]. For the example polynomial $p(i)$ we obtain the range overestimation $\frac{1}{2}[0, N]^2 - \frac{1}{2}[0, N] = [-\frac{1}{2}N, \frac{1}{2}N^2]$. In fact, different closed-form polynomial representations result in different bound estimations, none of which is exact when variable i occurs twice or more times in the expression. For example, the Horner form of this polynomial gives the bound $[-\frac{1}{2}N, \frac{1}{2}N^2 - \frac{1}{2}N]$.

4. Loop Iteration Path Timing Estimation

The objective is to find the critical paths in the body of a loop, i.e. the paths that are the most time consuming across the iterations of a loop. The following loop fragment generalizes this concept:

```
for i = a to b step s do
    if C then
      S1
    else
      S2
```

If condition C partitions the iteration space of i, e.g. C is a condition such as $i > c$, then loop timing analysis can be applied by splitting the loop into partitioned loops which can be separately analyzed. The partitioning is not performed on the physical code but virtually to aid the timing analysis. For example:

```
for i = a to min(b, c) step s do
    S1
for i = max(a, c + 1) to b step s do
    S2
```

However, it is far more common in practice that condition C exhibits switching behavior that is unknown at compile time. Therefore, a conservative timing estimation must be made for a loop nest by determining the critical path(s) through the body of a loop. Worst-case execution timing analysis then assumes that the critical path(s) are always taken at each iteration of the loop nest.

Whether a path is critical may also depend on the current iteration executed in the loop. Interesting cases arise when one path is critical at the first few loop iterations but then gradually decreases in execution time during subsequent iterations while another path increases in execution time eventually exceeding the time of the former path. The analysis should take both paths into consideration. Unfortunately, finding the break-even point is not always possible, because it requires root-finding on an interval with symbolic bounds.

However, the path timing analysis can be performed independently on each path, even when these paths include the same loop body operations, because the choice of path taken in the loop body cannot be determined at compile time. Furthermore, no assumptions of data dependences carried from one iteration to the next have to be made for timing estimation of multiple paths. Therefore, to align and compare the timing of a path with a descending time curve to a path with an ascending time curve, the iteration order of one of the paths in the loop can be virtually reversed. If after alignment it can be observed that one path always dom-

inates the other in the estimated execution time, then it is safe to assume that the dominating path determines the total estimated execution time of the loop.

Without loss of generality, the paper presents an approach to analyze two paths in the body of a loop. The extension to multiple paths is straightforward. If the worst-case execution time bounds on S_1 and S_2 are polynomial in i (including constant), then the worst-case execution time of the entire loop nest can be accurately determined. To this end, the timing of the paths is compared by computing the difference series of the Newton series of the polynomials that bound the execution time of the paths.

Definition 4.1 *Let $\Phi(i) = \langle\phi_0, \ldots, \phi_k\rangle_i$ and $\Psi(i) = \langle\psi_0, \ldots, \psi_\ell\rangle_i$ denote Newton series. Then, the difference series $\Delta(i) = \Phi(i) - \Psi(i) = \langle\delta_0, \ldots, \delta_{\max(k,\ell)}\rangle_i$ is defined by*

$$\delta_j = \begin{cases} \phi_j - \psi_j & \text{if } j \le k \text{ and } j \le \ell \\ \phi_j & \text{if } j \le k \text{ and } j > \ell \\ -\psi_j & \text{if } j > k \text{ and } j \le \ell \end{cases}$$

for all $j = 0, \ldots, \max(k, \ell)$.

The difference series of the Newton series of two polynomials gives the necessary information to determine which of the two polynomials is the largest on an interval.

Lemma 4.2 *Let $\Phi(i)$ be the Newton series of a polynomial $p(i)$ and let $\Psi(i)$ be the Newton series of a polynomial $q(i)$. If $\mathcal{L}(\Phi(i) - \Psi(i)) \ge 0$ then $p(i) \ge q(i)$ for all $i = 0, \ldots, n-1$, $n \ge 0$, and if $\mathcal{U}(\Phi(i) - \Psi(i)) \le 0$ then $p(i) \le q(i)$ for all $i = 0, \ldots, n-1$.*

A maximizing operator on Newton series is introduced that exploits iteration reversal. The maximizing operator returns the larger of two series. If neither of the two series dominates, a new series is constructed that bounds both polynomials of the series.

Definition 4.3 *Let $i = 0, \ldots, n-1$, $n \ge 0$, be the iteration space of a (normalized) loop with two execution paths through the loop body. Let $\Phi(i) = \langle\phi_0, \ldots, \phi_k\rangle_i$ denote the Newton series of the parameterized WCET of the first path and let $\Psi(i) = \langle\psi_0, \ldots, \psi_\ell\rangle_i$ denote the Newton series of the parameterized WCET of the second path. We define the maximizing operator $\uparrow$ of the two series by*

$$\Phi(i) \uparrow \Psi(i) = \begin{cases} \Phi(i) & \text{if } \mathcal{L}(\Delta_1(i)) \ge 0 \\ & \quad \text{or } \mathcal{L}(\Delta_2(i)) \ge 0 \\ & \quad \text{or } \mathcal{L}(\Delta_3(i)) \ge 0 \\ & \quad \text{or } \mathcal{L}(\Delta_4(i)) \ge 0 \\ \Psi(i) & \text{if } \mathcal{U}(\Delta_1(i)) \le 0 \\ & \quad \text{or } \mathcal{U}(\Delta_2(i)) \le 0 \\ & \quad \text{or } \mathcal{U}(\Delta_3(i)) \le 0 \\ & \quad \text{or } \mathcal{U}(\Delta_4(i)) \le 0 \\ \Theta(i) & \text{otherwise} \end{cases}$$

where

$$\begin{aligned} \Delta_1(i) &= \Phi(i) - \Psi(i) \\ \Delta_2(i) &= \Phi(i) - \Psi(n-i-1) \\ \Delta_3(i) &= \Phi(n-i-1) - \Psi(i) \\ \Delta_4(i) &= \Phi(n-i-1) - \Psi(n-i-1) \end{aligned}$$

and

$$\Theta(i) = \langle\theta_0, \ldots, \theta_{\max(k,\ell)}\rangle_i$$

with coefficients

$$\theta_j = \begin{cases} \max(\phi_j, \psi_j) & \text{if } j \le k \text{ and } j \le \ell \\ \max(0, \phi_j) & \text{if } j \le k \text{ and } j > \ell \\ \max(0, \psi_j) & \text{if } j > k \text{ and } j \le \ell \end{cases}$$

for all $j = 0, \ldots, \max(k, \ell)$.

The differences $\Theta(i) - \Phi(i)$ and $\Theta(i) - \Psi(i)$ are always positive and therefore $\Theta(i)$ bounds both $\Phi(i)$ and $\Psi(i)$.

The Newton series $\Phi(n-i-1)$ is obtained from the series $\Phi(i)$ by reversing the evaluation of the polynomial on the interval $[0, n-1]$. There are several ways this can be accomplished. A simple implementation is to convert the Newton series back to a polynomial using $p(i) = \mathbf{N}_k^{-1}\Phi(i)$ which gives the coefficients of $p(i)$ or with $p(i) = \sum_{j=0}^{k} \phi_j \binom{i}{j}$ which gives a closed-form polynomial expression. Then replace i with $n-i-1$ in $p(i)$, simplify the polynomial, and convert to Newton series with $\Phi(n-i-1) = \mathbf{N}_k p(n-i-1)$.

The generalization of this approach to multiple paths and multiple loops follows from the fact that $\Phi(i) \uparrow \Psi(i)$ is closed under polynomial formation. Therefore it can be applied to multiple paths to maximize the Newton series representation of the timing of the paths.

Example 4.4 *Consider the following loop with a loop body that contains two execution paths:*

```
1: for i = 0 to N do
2:     if a[i] > 0 then
3:         for j = 0 to i do
4:             S4
5:     else
6:         for j = 0 to N − i do
7:             for k = 0 to j do
8:                 S8
```

Suppose the following execution time bounds are obtained

$$\begin{aligned} \omega(H_1(0)) &\le 1 \\ \omega(C_2(i)) &\le 1 \\ \omega(H_3(i,0)) &\le 1 \\ \omega(S_4(i,j)) + \omega(H_3(i,j+1)) &\le 2 \\ \omega(H_6(i,0)) &\le 1 \\ \omega(H_7(i,0)) &\le 1 \\ \omega(S_8(i,j)) + \omega(H_7(i,j+1)) &\le 2 \quad . \end{aligned}$$

The analysis begins with the inner loops and proceeds to the outer loop. Loop bounds analysis revealed that none of

Difference Series	Value Range on Domain $i = 0 \ldots, N$
$\Delta_1(i) = \Phi(i) - \Psi(i) = \langle -1 - 4N - N^2, 5 + 2N, -2\rangle_i$	$[\mathcal{L}(\Delta_1), \mathcal{U}(\Delta_1)] = [-1 - 4N - N^2, -1 + 2N]$
$\Delta_2(i) = \Phi(i) - \Psi(N - i) = \langle -1, -3, -2\rangle_i$	$[\mathcal{L}(\Delta_2), \mathcal{U}(\Delta_2)] = [-1 - 2N - N^2, -1]$
$\Delta_3(i) = \Phi(N - i) - \Psi(i) = \langle -1 - 2N - N^2, 1 + 2N, -1\rangle_i$	$[\mathcal{L}(\Delta_3), \mathcal{U}(\Delta_3)] = [-1 - 2N - N^2, -1]$
$\Delta_4(i) = \Phi(N - i) - \Psi(N - i) = \langle -1 + 2N, -7, -2\rangle_i$	$[\mathcal{L}(\Delta_4), \mathcal{U}(\Delta_4)] = [-1 - 4N - N^2, -1 + 2N]$

Table 1. Difference Series of $\Phi(i)$ and $\Psi(i)$ and Their Value Range for Example 4.4

the iteration space sizes of the loops is negative and therefore the max *operations can be eliminated from the σ functions. The first path spans statements* 2 *to* 4 *and the second path spans statements* 2 *and* 6 *to* 8*. The Newton series corresponding to the two paths are*

$$\begin{aligned}
\Phi(i) &= \mathbf{N}_1(3 + \sigma(\mathbf{N}_0 2, i + 1)) \\
&= \mathbf{N}_1(5 + 2i) \\
&= \langle 5, 2\rangle_i \\
\Psi(i) &= \mathbf{N}_2(3 + \sigma(\mathbf{N}_1(1 + \sigma(\mathbf{N}_0 2, j + 1)), N - i + 1) \\
&= \mathbf{N}_2(6 + 4N + N^2 - (2N + 4)i + i^2) \\
&= \langle 6 + 4N + N^2, -3 - 2N, 2\rangle_i
\end{aligned}$$

and the reversed series are

$$\begin{aligned}
\Phi(N - i) &= \mathbf{N}_1(3 + \sigma(\mathbf{N}_0 2, N - i + 1)) \\
&= \mathbf{N}_1(5 + 2N - 2i) \\
&= \langle 5 + 2N, -2\rangle_i \\
\Psi(N - i) &= \mathbf{N}_2(3 + \sigma(\langle 3, 2\rangle_j, i + 1) \\
&= \mathbf{N}_2(6 + 4i + i^2) \\
&= \langle 6, 5, 2\rangle_i \quad .
\end{aligned}$$

The difference series and their value ranges on the domain $i = 0, \ldots, N$ are shown in Table 1. The value range of $\Delta_2(i)$ is negative. Therefore, it can be concluded that the polynomial of the series $\Psi(N - i)$ bounds the polynomial of the series $\Phi(i)$. That is, $\Psi(i)$ (and the reversed $\Psi(N-i)$) represents the execution time of the critical path through the loop body. The total worst-case execution time of the loop fragment is given by

$$WCET_1 = 1 + \sigma(\Psi(i), N + 1) = 7 + 6\tfrac{1}{6}N + 2\tfrac{1}{2}N^2 + \tfrac{1}{3}N^3$$

Now consider a conventional WCET estimation that is determined by decoupling the outer loop from the inner loops. This WCET is the best estimation that current methods can achieve. This estimation is obtained by simplifying the problem by assuming that the inner loops are bounded by the maximum value of i which is N. Effectively, this WCET is computed from the Newton series of the inner loops by replacing i by N giving $\mathbf{N}_1^{-1}\Phi(N + 1) = 5 + 2N$ and $\mathbf{N}_2^{-1}\Psi(N + 1) = 6 + 4N + N^2$. Because the outer loop iterates $N + 1$ times and the loop header execution time is 1*, the simplified worst-case execution time estimation is*

$$WCET_2 = 1 + (N + 1)\max(5 + 2N, 6 + 4N + N^2)$$

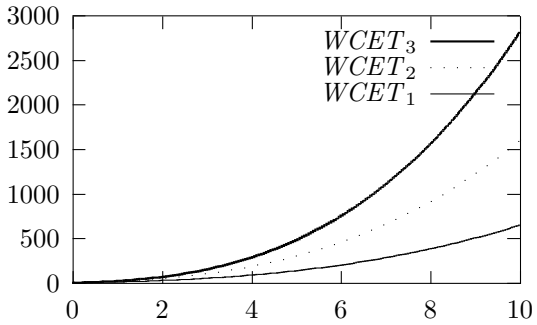


Figure 3. Comparison of WCET Estimations

Yet even more simple estimation methods are commonly used in practice that bound the iteration space of loops by using value range analysis that provides only very simple bounding expressions on the loop iteration space sizes. These methods cannot accurately estimate the timing of triangular loops, because inner loops are decoupled from outer loops. For example, loop statement 3 *iterates at most $N + 1$ times thereby executing a loop body with cost* 2*. This gives $2(N + 1) + 3$, where* 3 *is the cost of the header of loop statement* 1*, the cost of the conditional statement* 2*, and the cost of the header of loop statement* 3*. Loop statement* 8 *iterates at most $N + 1$ times thereby executing a loop body with cost* 2*. Loop statement* 7 *iterates at most $N + 1$ times thereby executing the inner loop with cost $2(N + 1) + 1$. The cost of the loop nest statements* 6 *to* 8 *is $(2(N + 1) + 1)(N + 1) + 3)$. The total estimated execution time using this simple approach is*

$$WCET_3 = 1 + (N + 1)\max(5 + 2N, 6 + 5N + 2N^2)$$

Figure 3 shows the worst-case execution time estimation bounds using Newton-Gregory estimation ($WCET_1$), decoupled estimation ($WCET_2$), and simple estimation ($WCET_3$), for increasing values of N (shown on the x-axis). ◇

5. Conclusions and Future Work

This paper presented a novel approach to parametric WCET estimation for rectangular and non-rectangular loop

nests including those with zero-trip loops, non-unit strides, and multiple critical iteration paths. Future work includes the elimination of the limiting $\max$ operations from the σ sum function through the addition of guarding conditions to the Newton series, to represent piecewise polynomials. This is necessary when the bounds of an inner loop includes unknowns such that the iteration space is possibly negative. In addition, the bounding polynomial Θ used by the maximizing operator $\uparrow$ will be improved.

References

[1] O. Bachmann. *Chains of Recurrences*. PhD thesis, Kent State University, College of Arts and Sciences, 1996.

[2] W. Blume and R. Eigenmann. Demand-driven, symbolic range propagation. In 8^{th} *International workshop on Languages and Compilers for Parallel Computing*, pages 141–160, Columbus, Ohio, USA, Aug. 1995.

[3] P. Clauss. Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: Applications to analyze and transform scientific programs. In *proceedings of the 1996 International Conference on Supercomputing*, pages 278–285. ACM Press, 1996.

[4] T. Fahringer. Efficient symbolic analysis for parallelizing compilers and performance estimators. *Supercomputing*, 12(3):227–252, May 1998.

[5] M. R. Haghighat. *Symbolic Analysis for Parallelizing Compilers*. Kluwer Academic Publishers, 1995.

[6] W. H. Harrison. Compiler analysis of the value ranges of variables. *IEEE Transactions on Software Engineering*, 3(3):243–250, May 1977.

[7] C. Healy, M. Sjödin, V. Rustagi, D. Whalley, and R. van Engelen. Supporting timing analysis by automatic bounding of loop iterations. *Real-Time Systems*, pages 121–148, May 2000.

[8] W. Pugh. Counting solutions to Presburger formulas: How and why. In *ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 121–134, Orlando, FL, June 1994.

[9] H. Ratchek and J. Rokne. *Computer Methods for the Range of Functions*. Ellis Horwood Limited, Chichester, West Sussex, PO19 1EB England, 1984.

[10] R. Sakellariou. Symbolic evaluation of sums for parallelising compilers. In *IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*. Wissenshaft & Technik Verlag, 1997.

[11] R. A. van Engelen and K. Gallivan. Tight non-linear loop timing estimation. In *International Workshop on Innovative Architectures for Future Generation High-Performance Processors and Systems (IWIA) 2002*, pages 21–26, Maui, Hawaii, 2002.

[12] E. Vivancos, C. Healy, F. Mueller, and D. Whalley. Parametric timing analysis. In *LCTES 2001*, pages 88–93, 2001.

[13] E. Zima. Simplification and optimization transformations of chains of recurrences. In *Proc. of the International Symposium on Symbolic and Algebraic Computing*, Montreal, Canada, 1995. ACM.