Cache Complexity and Multicore Implementation for Univariate Real Root Isolation

Changbo Chen, Marc Moreno Maza and Yuzhen Xie
Ontario Research Centre for Computer Algebra, University of Western Ontario, London, Canada

Overview

Isolating the real roots of a univariate polynomial is a driving subject in computer algebra. Many researchers have studied this problem under various angles from algebraic algorithms to implementation techniques [1, 2, 7, 3, 5]. Today, multicores have become the most popular parallel hardware architectures. Besides, understanding the implications of hierarchical memory on performance software engineering has become essential. These observations motivate the work presented in this poster. First, we analyze the cache complexity of the core routine of many real root isolation algorithms, namely, the *Taylor shift*. Secondly, we present efficient multithreaded implementation targeting multicores.

Taylor shift and Real Root Isolation

For a squarefree univariate polynomial $f(x) \in \mathbb{Q}[x]$ the Vincent-Collins-Akritas Algorithm reduces the problem of isolating the real roots of p to that of computing the coefficients of $f(x+1)$, the Taylor shift of f. If $f(x)$ writes $a_n x^n + \cdots + a_1 x + a_0$, the Pascal Triangle relation $f_i(x) := f_{i+1}(x) * (x+1) + a_i$ for i successively equal to $n-1, \ldots, 1, 0$ with $f_n(x) := a_n$ produces $f_0(x) = f(x+1)$.

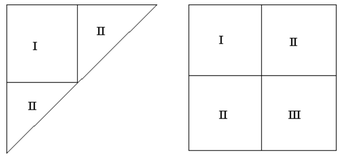
EXAMPLE. Let $f(x) = a_3x^3 + a_2x^2 + a_1x + a_0$. The diagram of the computation is illustrated as follows.

$$\begin{array}{cccc} 0 & 0 & 0 & 0 \\ a_3 + & + & + & + \to c_3 \\ a_2 + & + & + \to c_2 & \\ a_1 + & + \to c_1 \searrow & & \\ a_0 + \to c_0 & & & \end{array}$$

Proceeding the addition in diagonal direction is exactly the same as one computes $f(x+1)$ in Horner's rule.

Divide-and-conquer Taylor shift

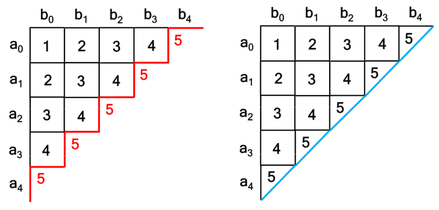
The elements of the Pascal Triangle and thus the coefficients of $f(x+1)$ can be computed in a divide-and-conquer manner sketched by the figures below. In this process, each triangular or square region is divided into smaller regions until a base case is reached. One observes that in both the triangular and square division, opportunities for concurrent execution and improved data locality are created.



The work and span of this algorithm are respectively $\Theta(n^2)$ and $\Theta(n^{log_2 3})$, In addition, this algorithm can be run in-place, in space $\Theta(n)$. Using the ideal cache model [4] , for a cache of Z words with cache line size L, we have shown that this algorithm incurs $\Theta(n^2/ZL)$ cache misses. Using the Hong-Kung lower bounds, we deduce that this latter result is optimal.

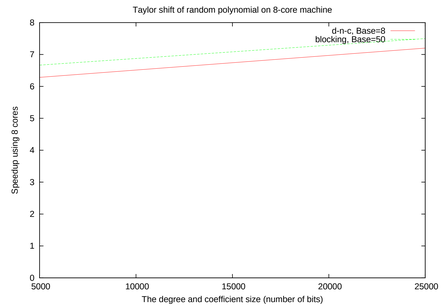
A Blocking Strategy

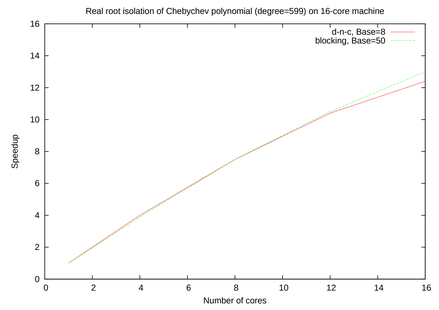
One can observe that, in the above triangle division, parts of the small triangle regions can be evaluated before completing region I. One can partition the entire Pascal Triangle into $B \times B$ blocks, where B should be tuned in order for a block to fit in cache. Then, the blocks are traversed one anti-diagonal band after another starting from the top left corner. Both span and parallelism are now $\Theta(Bn)$ and $\Theta(n/B)$ respectively. Moreover, our algorithm runs in place in space $\Theta(n)$. In addition, if B is well chosen, the above cache complexity estimate is preserved.

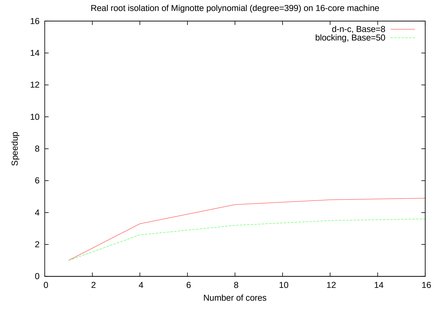


Experimentation

In `Cilk++` targeting multicores, relying on the `GMP` library, we have implemented these two approaches, that we denote d-n-c (for divide-and-conquer) and blocking (for the blocking strategy). We provide experimental data on a 8-core machine for both approaches and both problems of Taylor Shift and Real Root Isolation. For real root isolation, we also explore the parallelism of the binary tree for searching the roots. The machine has 8 GB memory, 6144 KB of L2 cache and each processor is Intel Xeon X5460 @3.16 GHz.







On one core (data not reported here) the blocking strategy and divide-and-conquer approach outperform all other data traversals that we have tried. Between the two, the winner varies from one architecture to another. On a given multicore architecture, for the problem of real root isolation, the winner varies from one test example to another, with, may be, a slight advantage on average to the blocking strategy.

References

[1] G. E. Collins and A. G. Akritas. Polynomial real root isolation using Descarte's rule of signs. *In SYMSAC'76*, pages 272–275.

[2] G. E. Collins, J. R. Johnson and W. Küchlin. Parallel real root isolation using the coefficient sign variation method. Number 584 in Lecture Notes in Computer Science, pages 71-87, 1992.

[3] T. Decker and W. Krandick. Parallel Real Root Isolation Using the Descartes Method. *In HiPC' 99*, pages 261–268.

[4] M. Frigo, C. E. Leiserson, H. Prokop and S. Ramachandran. Cache-Oblivious Algorithms. *In FOCS '99*, 1999.

[5] J. R. Johnson, W. Krandick, A. D. Ruslanov. Architecture-aware classical Taylor shift by 1. *In ISSAC '05*, pages 200–207.

[6] Jia-Wei Hong and H. T. Kung. I/O complexity: the red-blue pebbling game. *In STOC'81*, pages 326–333.

[7] J. Von zur Gathen and J. Gerhard. Fast algorithms for Taylor shifts and certain difference equations. *In ISSAC '97*, pages 40–47.