Concurrent Computation and Time Complexity Bounds for Algebraic Fractals

F. Chris MacPhee[a] and Virendrakumar C. Bhavsar[b]

[a,b]Advanced Computational Research Laboratory, Faculty of Computer Science,
University of New Brunswick, Fredericton, NB, E3B 5A3, Canada
http://www.cs.unb.ca/acrl/

[a] *Chris.Macphee@unb.ca*

[b] *bhavsar@unb.ca*

This paper considers various parallelization schemes and applies them to concurrent computation of algebraic fractals on IBM SP3 and SGI Onyx 3400 systems. We adapt the time complexity bounds of parallel Monte Carlo methods to predict the execution time behavior of these schemes.

Nous étudions plusieurs méthodes de parallélisation appliquées au calcul simultané de plusieurs objets fractals algébriques sur un IBM SP3 et sur un SGI Onyx 3400. Nous adaptons les bornes de complexité temporelles des calculs Monte-Carlo parallèles afin de prédire le comportement du temps d'exécution de ces méthodes.

1. Introduction

The generation of fractal images from the self-squared function $z \leftarrow z^2 + c$ is widely discussed in the literature (for example, see [1]). The superset $z \leftarrow z^\alpha + c$ proposed by Gujar and Bhavsar [2] allows the exponent of z to be positive and negative, integer as well as real. Some techniques for the vectorization [4] and parallelization [chap. 3, 6; 7; 9] of fractal computations are given in the literature.

In the following section, we brielfly describe the computational aspects of algebraic fractals. Subsequently, we analyze the visual characteristics of the fractals. In §3, we consider the parallelization schemes proposed by MacPhee and Bhavsar [7] for fractal generation. Assuming independent and identically distributed (i.i.d.) execution times for pixel computations, we apply the time complexity bounds for parallel Monte Carlo methods in §4. We present the computational results of our experiments on IBM SP3 and SGI Onyx 3400 systems in §5. We investigate if the bounds could be used to describe the execution time behavior of these schemes. Finally, we conclude the paper with our findings.

2. Computational and Visual Characteristics

We consider the following parameters for algebraic fractal computations: the pixels map is 1024×1024 with $|\Re(z)| \leq 1.5$ and $|\Im(z)| \leq 1.5$, the maximum allowable number of iterations per pixel is 100, and z is said to diverge if $|z| > 10$.

We use the Open Visualization Data Explorer (OpenDX) package for visualizing fractals. The colour scheme used is intentionally skewed to provide more resolution in the [1, 10] and [91, 100] ranges, due to their higher concentration. Figure 1 shows sample images of various α; see http://www.cs.unb.ca/acrl/fractal/ for additional images.

Table 1 provides a statistical summary for various values of α. Algebraic fractals discussed within this paper have a bimodal distribution, with over 90% of the points having a value of [1, 10] or [91, 100], as seen in Figure 2.

As discussed in [7], as $|\alpha|$ increases, the number of nodes and satellites increase for $\alpha \geq 2$ and $\alpha \leq -2$, respectively. The rightmost images of Figure 1 shows that fractal images tend toward a unit circle centered on the origin as $|\alpha|$ tends toward infinity, approximated by $\alpha = \pm 100000$.

Table 1: Statistical summary of number of iterations in fractal images

	pixel		*column*	
α	*mean*	*std dev*	*mean*	*std dev*
-100000	65.7859	46.7158	67364.8	27608.5
-10	71.3106	44.1683	73022.1	23831.1
-2	83.2948	34.8595	85293.9	17162.6
2	22.2449	35.9684	22778.8	19381.1
10	29.4801	43.4767	30187.6	24781.6
100000	36.2092	46.7143	37078.2	27607.2

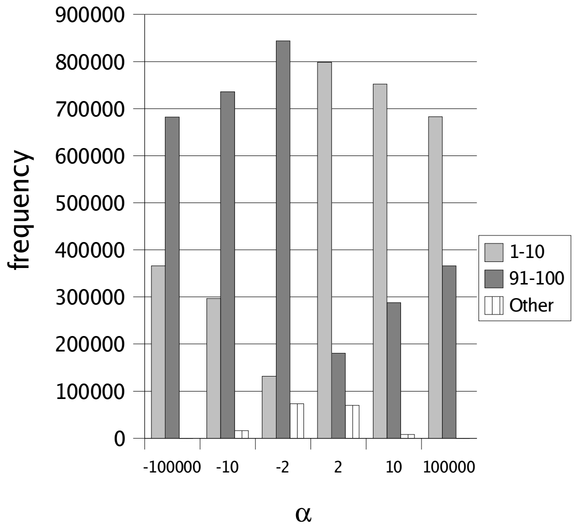


Figure 2: Distribution of number of iterations for pixel computations for various α

The unit circle forms due to the interior or exterior of the circle quickly diverging. We see that z^{α} can be defined by

$$\begin{aligned} z^{\alpha} &= |z|^{\alpha}\left[\cos(\alpha\,\Theta) + i\sin(\alpha\,\Theta)\right] \\ &= \sqrt{\Re(z)^2 + \Im(z)^2}^{\,\alpha}\left[\cos(\alpha\,\Theta) + i\sin(\alpha\,\Theta)\right] \end{aligned}$$

(1)

where $\Theta = \arctan\left(\frac{y}{x}\right)$. Therefore, the divergence of the pixel as $|\alpha|$ tends toward infinity is dependent upon $\Re(z)^2 + \Im(z)^2$, as seen in Table 2.

Table 2: Iterations of pixels as $|\alpha|$ tends toward infinity

	α	*Iterations of pixel*
$\Re(z)^2 + \Im(z)^2 > 1$	∞	2*
$\Re(z)^2 + \Im(z)^2 \le 1$	∞	100
$\Re(z)^2 + \Im(z)^2 > 1$	$-\infty$	100
$\Re(z)^2 + \Im(z)^2 \le 1$	$-\infty$	2*
** Note: z is initialized to* $0+i0$ *for* $\alpha>0$ *and* $1+i1$ *for* $\alpha\le0$*, so the first iteration sets z=c. z exceeds the limit during the second iteration, when* z^{α} *(i.e.* c^{α}*) occurs.*		

3. Parallelization Schemes

The three fractal programs with message passing given in [7, 9], static, smart static, and dynamic work assignments, are used to conduct the experiments. Note that the computation of each pixel is independent.

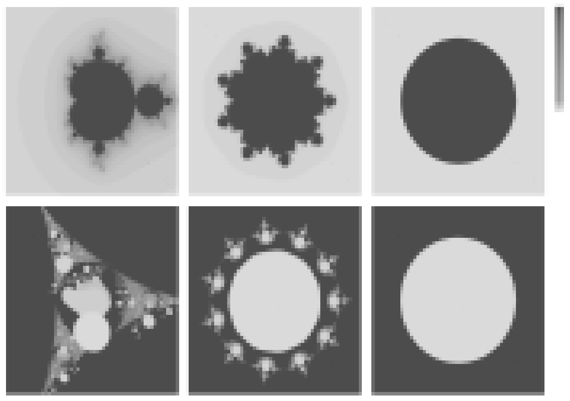


Figure 1: Various fractal images
Top row: α = 2, 10, 100000
Bottom row: α = -2, -10, -100000

In the static work assignment program, column groups are divided evenly among processors. This assignment scheme creates a load balancing problem, where either the inner or outer processors receive more computationally intensive columns for $\alpha \ge 2$ and $\alpha \le -2$, respectively.

In the smart static work assignment program, the workload is approximated by calculating the area under an approximation curve and dividing the workload, rather than columns, evenly over the processors.

In the dynamic work assignment program, the master passes work to the slaves in a block size of 2 columns. The block size of 2 columns was determined through experimentation with the machines discussed in §5 and, based upon these machines, approximating for larger numbers of processors.

4. Time Complexity Analysis

As stated earlier, the computation of each pixel is independent. Further, when pixel computations are carried out on a multiprocessor machine, the execution time depends on many factors, including scheduling by the operating system, shared memory conflicts, nondeterminism in I/O and interprocessor communication, and the work assignment scheme. In this context, we hypothesize that the execution times of pixels are i.i.d. random variables. Therefore, we believe that the time complexity bounds given by Bhavsar and Isaac [8] for parallel Monte Carlo methods could be applied to parallel algebraic fractal generation schemes discussed in the previous section.

Table 3: PIC-time distribution-free time complexity results for the various schemes for parallel Monte Carlo algorithms

Scheme	*Expected time complexity*
SCA-1 $(P \ll K)$	$\frac{K\mu}{P} \le E[T(P,K)] \le \frac{K\mu}{P} + \frac{\sqrt{K}\cdot\sigma}{2}$
SCA-2 $(\frac{P}{K} \sim 10)$	$\frac{K\mu}{P} \le E[T(P,K)] \le \frac{K\mu}{P} + \frac{(P-1)\sqrt{K}}{\sqrt{(2P-1)P}}\cdot\sigma$
SCA-3 $(P = K)$	$\frac{\mu}{P} \le E[T(P,K)] \le \mu + \frac{(K-1)}{\sqrt{2K-1}}\cdot\sigma$
SCA-4 $(P > K)$	*PIC-time distribution-free time complexity results are not possible.*
DCA-1	$E[T(P,K)]\, etd\, \frac{K\mu}{P} + \frac{(P-1)(\mu^2+\sigma^2)}{2P\mu}$
DCA-2	$\tau \leqslant E[T(P,K)] \leqslant \tau + \frac{(P-1)}{\sqrt{2P-1}}\cdot\sigma$ $where\ \tau\ etd\ \frac{K\mu}{P} + \frac{(P-1)(\mu^2-\sigma^2)}{2P\mu}$

K = desired total number of primary estimates of a problem variable.
μ = *mean of i.i.d. PIC times*
σ = *standard deviation of i.i.d. PIC times*
P = number of processors

Bhavsar and Isaac [8] proposed two basic static and dynamic computational assignment schemes for parallel Monte Carlo algorithms based on renewal theory and order statistics. These schemes assign the primary estimate computations to processors. For the fractal generation, the same schemes could be applied to pixel computations (PICs).

Table 3 provides a list of the PIC time complexity results for the assignment schemes based on results given in [8]. The number of PICs to be calculated, *K*, is the total number of rows in the fractal image. For fractals within this paper, we have set *K*=1024.

In the static computation assignment (SCA) schemes, a fixed number of PICs are assigned to the processors before any computations are initiated. The SCA time complexity bounds are composed of multiple SCA schemes, determined by the relationship of *P* to *K*.

In SCA-1 and 2, *K* PICs are assigned over *P* processors, such that each processor is assigned the same number of PICs. In SCA-3, one PIC is assigned to each processor. In SCA-4, one PIC is assigned to each processor, but computations are terminated once *K* PICs are completed. Only SCA-1, 2, and 3 are used for the purposes of algebraic fractal computation.

In the dynamic computation assignment (DCA) schemes, PICs are farmed out to the processors; the decision to initiate further PICs is based on a preset criteria. In DCA-1, PICs are farmed to processors until *K* PICs have been completed; any incomplete PICs on the remaining processors are aborted. In DCA-2, only *K* PICs are farmed to the processors; unlike DCA-1, no PICs are aborted. Only DCA-2 is used for the purpose of algebraic fractal computation, as all PICs must be completed.

The means and standard deviations of the algebraic fractals found in Table 1 are used to obtain the bounds of the speedup defined by Table 3. Figures 3 and 4 depict the upper and lower bounds of the speedup for the generation of $\alpha = 10$, along with the estimated performance of the static, smart static, and dynamic assignment schemes.

4.1 SCA Schemes

The makeup of the SCA time complexity bounds are considered to be as follows:

SCA-1 $(P\ trèssupérieuràn\ K)$: for *P* = 2, 4, 8, .:. , 64.

SCA-2 $(P / K \sim 10)$: for *P* = 128, 256, and 512.

SCA-3 $(P = K)$: for *P* = 1024.

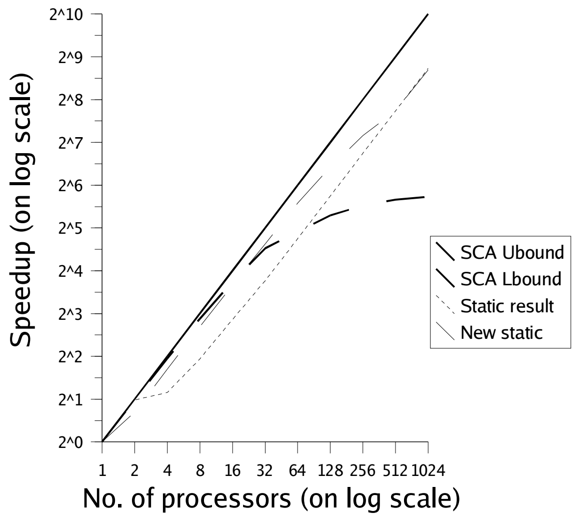


Figure 3: Theoretical and estimated performance of static schemes for parallel Monte Carlo algorithms for generating a fractal image with $\alpha = 10$

As seen in Figure 3, neither static assignment scheme follows the lower bound at $\alpha = 10$ for small values of *P*. SCA-1 requires i.i.d. random variables to satisfy a normal distribution through the application of the Central Limit theorem, but in this case the PIC times do not uphold our assumption of being i.i.d. random variables.

As α increases, the smart static assignment scheme distributes PICs more efficiently to the processors, which emulates the normal distribution required by SCA-1. For large α, the smart static assignment scheme results are within the upper and lower bounds given by SCA-1.

Both static assignment schemes are within the SCA-2 and 3 bounds, as the lower bound is not dependent on i.i.d. random variables.

4.2 DCA-2 Scheme

As seen in Figure 4, the dynamic assignment scheme results are within the bounds given by DCA-2. Due to the low standard deviation of processor execution times, the dynamic assignment scheme achieves a high efficiency, only slightly below the upper bound given by DCA-2.

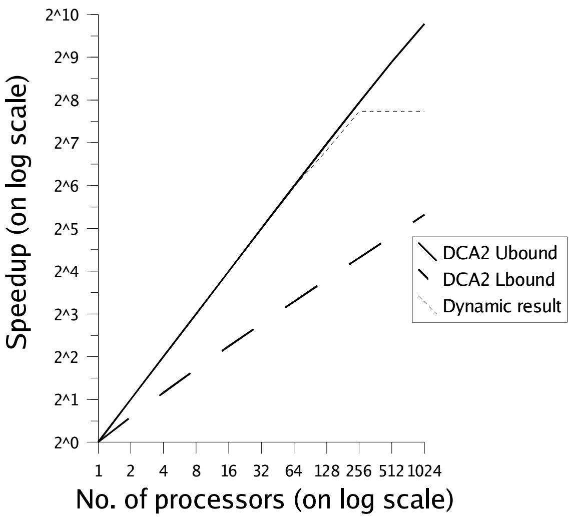


Figure 4: Theoretical and estimated performance of various dynamic schemes for parallel Monte Carlo algorithms for generating a fractal image with row division = 2 and α = 10

5. Experimental Results

Experiments were performed on the IBM SP3 in the Advanced Computational Research Laboratory of the University of New Brunswick (http://www.cs.unb.ca/acrl/) and the SGI Onyx 3400 in the Computation and Visualization Centre of the Memorial University of Newfoundland (http://www.cvc.mun.ca/).

The IBM SP3 is configured with 4 compute nodes, with each node containing 2 x 2-way 375 MHz POWER3 64-bit Winterhawk II Processor cards and 1 GB of memory. The SGI Onyx 3400 is configured with 28 compute nodes with each node containing one 400MHz IP35 MIPS R12000 processor and 0.5 GB of memory.

A total of 9 processors were used during the experiments, configured as 1 master and 8 slaves.

5.1 IBM SP

Figures 5 and 6 show the DCA-2 bounds and experimental performance of the dynamic assignment scheme for various α on the IBM SP3.

While α ≤ -2 are within the DCA-2 bounds, α ≥ 2 falls below the DCA-2 lower bound. Due to the higher mean of the pixels and rows of α ≤ -2 fractals, more time is spent on computation rather than communication. This allows α ≤ -2 fractals to achieve a larger speedup than their α ≥ 2 counterparts.

5.2 SGI Onyx 3400

Figures 7 and 8 show the DCA-2 bounds and experimental performance of the dynamic assignment scheme for various α on the SGI Onyx 3400.

Both α ≥ 2 and α ≤ -2 results are within the DCA-2 bounds on the SGI Onyx 3400. This is due to its improved interconnection network and the higher computation time required to complete results on the SGI Onyx 3400. Both factors increase the computation-to-communication ratio, which increases speedup.

6. Conclusion

The computation and visualization characteristics of algebraic fractals were discussed and a bimodal distribution of the pixel values was observed for α ≥ 2 and α ≤ -2. As |α| tends toward infinity, the distribution of the number of iterations for each pixel of the fractals focuses almost exclusively on the two local maxima of [1, 10] and [91, 100], due to pixels either quickly diverging or converging.

The assumption of i.i.d. pixel execution times is used to apply time complexity bounds of parallel Monte Carlo methods. For the static assignment scheme, the execution time was found to be below the lower bound of the SCA-1 scheme, as it is dependent upon a normal distribution, which the assignment scheme does not provide. The execution time for the smart static assignment scheme was found to be below the lower bound of the SCA-1 scheme when |α| is small, but as |α| increases, which increases the assignment schemes efficiency, a normal distribution is achieved and both bounds apply. Both static assignment schemes are within both of the SCA-2 and SCA-3 bounds. The dynamic assignment scheme is within the DCA-2 bounds and is only slightly below the given upper bound.

Experimental results on the IBM SP and SGI Onyx 3400 show that the DCA-2 bounds hold in practice, although a lower computation-to-communication ratio may pull the speedup below the DCA-2 lower bound.

In conclusion, the time complexity bounds given in this paper can be used to predict the execution time behavior of concurrent computation of algebraic fractals.

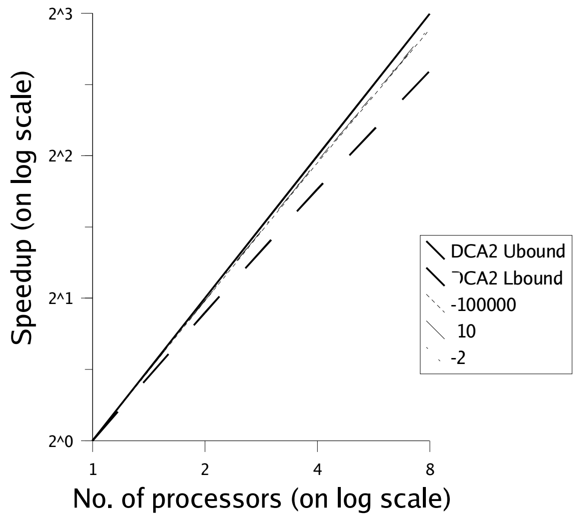
References

[1] H. O. Peitgen and P. Richter, *The Beauty of Fractals*, Springer-Verlag, Berlin, 1996.

[2] U. G. Gujar and V. C. Bhavsar, "Fractals from $z \leftarrow z^{\alpha} + c$ in the Complex z-plane", *Comp. and Graph.*, 16(1), pp. 45-49, 1992.

[3] S. V. Dhurandhar, V. C. Bhavsar, and U. G. Gujar, "Analysis of z-plane fractal images from $z \leftarrow z^{\alpha} + c$ for $\alpha < 0$", *Comp. and Graph.*, 17(1), pp. 89-94, 1993.

[4] V. C. Bhavsar, U. G. Gujar, N. Vangala, "Vectorization of generation of fractals from $z \leftarrow z^{\alpha} + c$ on IBM 3090 / 180VF", *Comp. and Graph.*, 17(2), pp. 169-174, 1993.



F

igure 5: Experimental performance of the dynamic assignment scheme for various $\alpha \leq -2$ on the IBM SP3

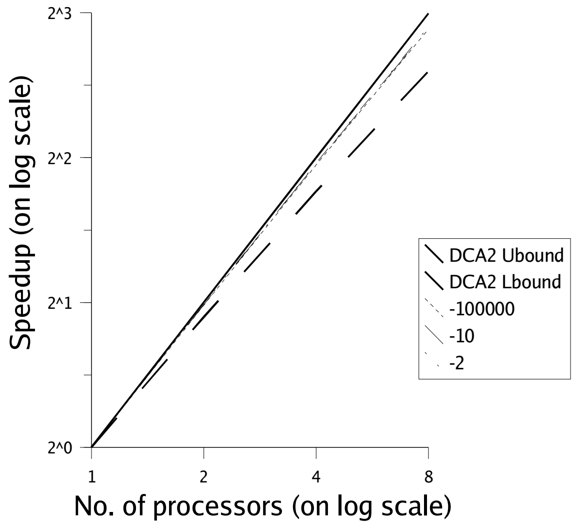


Figure 7: Experimental performance of the dynamic assignment scheme for various $\alpha \leq -2$ on the SGI Onyx 3400

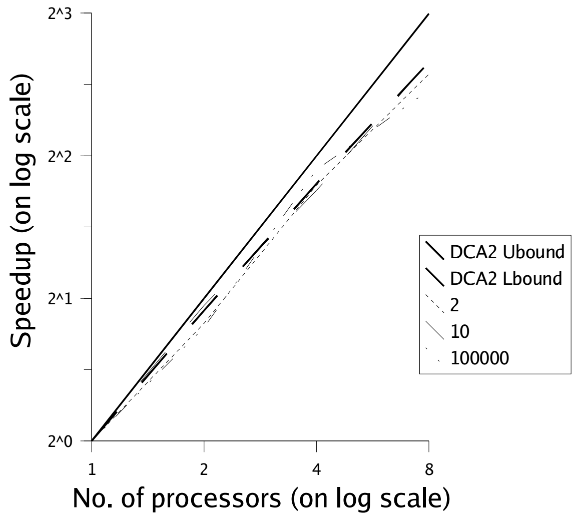


Figure 6: Experimental performance of the dynamic assignment scheme for various $\alpha \geq 2$ on the IBM SP3

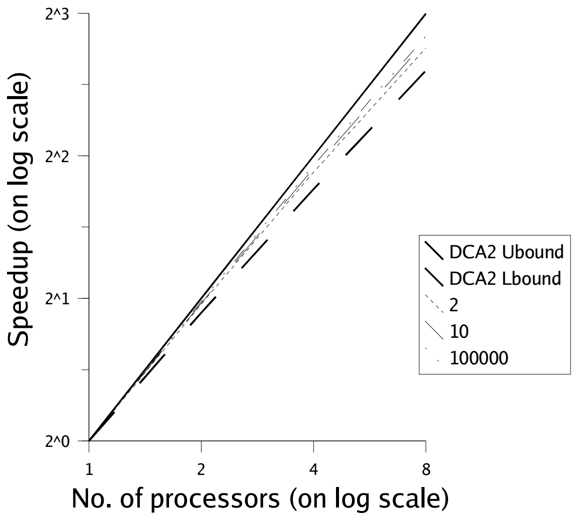


Figure 8: Experimental performance of the dynamic assignment scheme for various $\alpha \geq -2$ on the SGI Onyx 3400

[5] E. Aubanel, “Parallel Programming with Generalized Fractals,” Faculty of Computer Science, University of New Brunswick, , February 2002, http://www.cs.unb.ca/profs/aubanel/aubanel_fractals.html.

[6] B. Wilkinson and M. Allen, *Parallel Programming*, Prentice Hall, Upper Saddle River, 1999.

[7] F. Chris MacPhee and V. C. Bhavsar, Efficient Parallelization of Algebraic Fractals, Technical Report, Faculty of Computer Science, University of New Brunswick, Fredericton, N.B., Canada, March 2002.

[8] V. C. Bhavsar and J. R. Isaac, “Design and Analysis of Parallel Monte Carlo Algorithms,” *SIAM J. Stat. Comput.*, 8(1), pp. 73-95, 1987.

[9] F. Chris MacPhee and V. C. Bhavsar, “An Efficient Static Assignment scheme for Algebraic Fractals,” *Proceedings of High Performance Computing Systems & Applications*, J. Almhana and V. Bhavsar (Eds.), IEEE, Los Alamitos, CA, pp. 232-233, June 2002.