난수 발생기 성능 분석 보고서

안 윤 호, 오 광 진

KiSTi www.kisti.re.kr 슈퍼컴퓨팅센터

Ⅰ. 서 론

난수 발생기(Random Number Generator)는 과학적인 컴퓨터 시뮬레이션 분야에서부터 통계적인 표본 수집절차에 이르기까지 다양한 분야에서 사용되고 있다. 특히, 몬테카를로 방법(Monte Carlo method, MC), 확률론적 동역학(Stochastic Dynamics), 브라운 동역학(Brownian Dynamics)과 같은 계산과학(Computational science)에 많이 이용된다. 이러한 방법을 이용하는 응용 분야에서는 물리적인 시스템의 정확한 양을 계산하기 위해서 우수한 난수 발생기의 사용이 요구된다. 사용되는 난수 발생기에 따라 계산하고자 하는 물리적 측정값의 인위적인 결과도 초래할 수 있기 때문에, 난수 발생기의 질(Quality)은 매우 중요하다고 할 수 있다. 또한, 많은 양의 난수를 사용해야 하는 large-scale의 시뮬레이션에서는 난수 발생기의 역할이 매우 중요시될 수가 된다.

우리가 random number sequence를 발생하기 위해 사용하는 난수 발생기는 실제로 난수처럼 보이도록 하는 deterministic algorithm을 사용하기 때문에, 엄밀히 말하면 난수 발생기는 pseudo-random number(PRN) generator라고 불리어진다. 이러한 알고리즘에 의한 난수 시퀀스(PRN sequence)는 단지 어떤 한정된 의미에서 무작위적인 성격을 가지고 있을 수 있지만, 난수의 질을 그렇게 필수적으로 요구하지 않는 경우에는 무작위 특성에 대한 시늉만으로도 충분할 수가 있다. 하지만, 현대의 빠른 속도 컴퓨터와 정확한 값을 필요로 하는 계산들은 질적으로 우수한 난수 발생기에 대한 요구를 증가시키고 있다. 점차적으로 이러한 환경에서 난수발생기의 질은 매우 중요하다. 그러나, 질적으로 우수한 정도를 수학적으로 증명하기는 매우 어렵다.

여러 문헌에서 다양한 난수 발생기들에 의해 얻어진 결과 값을 확인할 수 있지만, 궁극적으로 난수 발생기의 질적 우수성 정도는 실험에서 나온 경험에 불과한 것이다. 다시 말해, 응용 기반의 테스트는 난수 발생기의 질을 보여줄 수 있는 적당한 방법이 될 수 없을 수도 있다. 그럼에도 불구하고, 그 테스트 결과를 통해 사

용된 난수 발생기 대한 우리의 확신 정도는 증가시킬 수 있는 것이다. 난수 발생기를 테스트하기 위하여 2차원 Ising 모델처럼 정확하게 풀 수 있는 시스템과 percolation 모델, random walks를 통한 방법들이 제안되어서 행해져 왔고[1-3], 이러한 테스트를 통하여 소위 좋다고 알려진 몇몇의 난수 발생기의 문제도 제기되었다[2]. 우리가 새로운 응용 프로그램을 다루게 될 때, 가장 안전한 방법은 다른 난수 발생기를 이용하여 여러 번 실행해 보는 것이다. 만약, 유사한 결과가 나온다면 그 결과는 받아 들일만 한 것이 된다. 각각의 다른 실행으로부터 나온 결과가 통계적인 오차를 줄일 수 있는 신뢰할 만한 값이 될 수 있기 때문에, 또 다른 여분의 노력은 필요하지 않을 수 있다.

몬테카를로(MC) 방법은 상당한 부분이 난수에 의존하는 수치적인 계산방법이다. 즉, 그 결과는 사용된 난수 발생기의 우수성, 결함 정도와도 직결된다고 할 수 있다. 본 연구에서는 먼저 무작위(ramdom), 확률적(stochastic)이라고 알려진 MC 방법인 pi(π) 계산을 통해 난수 발생기의 성능을 테스트하고, 적용 예로써 물리적 모델인 Diffusion-limited aggregation (DLA)의 소개와 함께 각기 다른 난수 발생기를 사용해 그 결과를 비교해 본다. DLA는 입자들이 브라운 운동 때문에 random walk을 겪으면서 하나의 집합체(aggregate)를 만들어가는 과정으로써, 일종의 결정 성장 모델이다. DLA에 대한 연구는 지금까지 여러 관련된 분야에서 연구되었고 물리적인 시스템의 응용에 관심이 많았지만, 사용된 난수 발생기의 영향에 대한 논의는 거의 없었다. 우리의 관심은 DLA에 사용된 난수 발생기와 측정하고자 하는 물리적 양과의 의존성에 관한 것이다.

본 보고서에서는 먼저 일반적으로 알려진 난수 발생기에 대해서 소개를 하고, pi 계산을 통해 난수 발생기들의 성능을 분석 비교한다. 그리고, DLA에 대한 물리적인 측정 결과를 비교해봄으로써 사용된 난수 발생기의 신뢰정도로 마무리를 한다.

Ⅱ. 난수 발생기(Random Number Generator)

오늘날 쓰이고 있는 많은 난수 발생기들은 컴퓨터 성능이 다소 느릴 때 개발되고 테스트되어졌다. 고성능 컴퓨터 등장과 성능이 빨라짐에 따라 MC 계산에 있어서 옛날보다 훨씬 많은 양의 난수를 소비하게 된다. 이것은, 이전에 확인되지 않았던 난수발생기의 문제를 드러내는데 충분하다. 따라서, 이론적으로나 실험적으로 난수 발생기를 테스트하는데 있어서 관심이 새로워지고 있다.

이상적인 순차(sequential) 난수 발생기는 다음의 조건들을 만족해야한다.

- Repeatability - 같은 초기값(initial seed)에 대해 같은 시퀀스를 만들어야 한다.
- Randomness - 서로 상관관계가 없고, 독립적이고 균일하게 분포된 난수를 생성시켜야 한다.
- Long period - 주기는 생성된 난수보다 훨씬 커야한다.
- Insensitivity to seeds - 주기(period)와 randomness는 초기값에 의존해서는 안된다.
- Portability - 다른 컴퓨터에서도 동일한 결과를 구현할 수 있어야 한다.
- Efficiency - 상대적으로 빠른 것이 좋으며, 너무 많은 메모리를 차지하지 않아야한다.

실제적으로, 이 모든 조건들을 동시에 충족하기는 불가능하다. 우리는 응용프로그램에서 필요한 총 난수 개수보다 훨씬 큰 흐름(stream)의 주기를 요구하는데, 이때 에는 상관관계(correlation)가 충분히 약할 수 있다. 이론적인 연구가 때로는 난수 시퀀스의 상관관계를 지적할 수 있지만, 많은 경우에 이러한 문제는 난수 발생기에 의한 결과들을 통계적으로 비교하는 실험 테스트들에서 잘 나타난다.

먼저, 난수 발생기들에서 많이 사용되는 주된 알고리즘을 다음과 같이 소개한다.

◉ Linear congruential generators(LCGs)

LCGs는 간단하고, 오래되고, 가장 일반적으로 널리 사용되는 난수 발생기이다. 난수 시퀀스는 다음 관계식을 이용하여 발생시킨다.

$$X_i = (a * X_{i-1} + c) \bmod M \tag{2.1}$$

식(2.1)에서, a는 승수(multiplier), M은 계수(modulus), 그리고 c는 상수(additive constant)이다. 변수 (a, c, M)은 큰 주기성과 균일하게 분포된 난수 특성을 나타내기 위해서 적절하게 선택되어져야 한다. 이 방법을 이용하는 난수 발생기 종류에는 Unix routine에서 사용되어지는 rand, drand48, 그리고 ranf가 있다.

◉ Lagged Fibonacci generators(LFGs)

LFGs는 매우 큰 주기를 얻을 수 있는 간단한 방법을 제공하고, 상대적으로 빨라질 수 있기 때문에 최근에 인기가 높아지고 있다. 관계식은 다음과 같다.

$$X_i = (X_{i-p} \odot X_{i-q}) \bmod M \tag{2.2}$$

식(2.2)에서, p와 q는($p>q$)는 lag이고, $\odot$는 +, −, * 및 $\otimes$(XOR)와 같은 수치 연산자이다. Lag은 긴 주기성을 나타내기 위하여 적절하게 선택되어 진다. 이 방법을 사용하는 난수 발생기는 Unix routine에서 사용되는 random이 있다.

◉ Shift register generators

Shift register (or Tausworthe) generators는 일반적으로 XOR을 사용하는 LFGs의 특수한 형태로 사용되어진다. 그러나, 상대적으로 좋지 않은 무작위(randomness) 특성 때문에 이 난수 발생기는 추천되지 않고 있다.

◉ Combined generators

이 방법은 앞서 소개된 서로 다른 난수 발생기를 이용하여 많은 환경에서 더 향상된 난수 발생기를 만들기 위하여 결합하는 것을 말한다.

다음은 병렬 난수 발생기에 대해서 소개를 한다. 우수한 질의 난수 발생기를 찾기란 힘들지만, 병렬 컴퓨터를 위한 좋은 난수 발생기를 찾는다는 것은 더더욱 어렵다. 병렬 난수발생기는 아래의 조건들을 만족해야 한다.

- 프로세서들 상호간에 상관(correlation) 의존성이 없어야한다.
- 각 프로세서에 생성되는 난수 시퀀스는 순차 난수 발생기의 질을 만족해야 한다.
- 각 다른 프로세서에 똑같은 난수 시퀀스 결과를 생성시켜야 하고, 프로세서 수에 관계없이 작동해야 한다.
- 프로세서 상호간 데이터 이동이 없어야 한다.

위에서 언급된 우수한 병렬 난수발생기가 만족해야 하는 조건들 중에서, 가장 중요한 것은 프로세서 상호간에 상관관계(correlation)가 없어야 한다는 것이다. 이 문제는 순차 난수 발생기에서는 나타나지 않았다. 병렬 컴퓨터가 처음 사용되기 시작하면서, 계산 과학자들은 순차 난수 발생기에서 생성되는 난수에 대한 병렬 난수 시퀀스를 재생하기 위하여 여러 방법을 고안하기 시작했다. 많은 병렬 난수 발생기들이 제안되었지만, 병렬화하는데 있어서 대부분은 비슷한 개념을 이용한다. 다음과 같이 기본적인 세가지 방법이 있다.

◉ Leapfrog method

이 방법은, 카드 놀이를 하는 사람들에게 한 벌의 카드처럼 여러 프로세서들 사이에서 교대로 순차 난수열이 쪼개어 지는 것을 말한다. 만약 총 N개의 프로세서가 있다면, 각 프로세서는 난수 시퀀스에 있어서 N만큼 껑충 뛰게 된다. 가령, i번 프로세서는 난수열이 X_i, X_{i+N}, X_{i+2N},... 순서대로 나타난다. 이 방법의 경우, 원

래 난수 시퀀스에 long-range 상관관계가 있으면, 병렬화에 있어서 short-range inter-stream 상관관계가 발생할 수 있는 문제가 있다.

◉ Sequence splitting

난수 발생기를 병렬화하는 또 다른 방법은, 각각의 프로세서들에 있어서 겹치지 않게 인접하는 구역으로 나누는 것이다. 만약 프로세서 총 개수가 N개이고 순차 난수 시퀀스의 주기가 P이라면, 첫 번째 프로세서는 P/N만큼의 난수를 갖게 되고 두 번째 세 번째도 동일한 양만큼 갖게 된다. 이 방법의 단점은, 사용자가 기대한 것보다 더 많은 시퀀스를 소비하게 되면, 난수 시퀀스가 겹칠 수 있다. 이 방법 또한, 원래 순차 난수 발생기에 long-range 상관관계가 존재하면, 병렬화하는데 있어서 short-range inter-stream이나 inter-processor 상관관계가 생길 수 있다.

◉ Independent sequences

이 방법은 sequence splitting 방법과 유사한 방법으로 각 프로세서는 난수 시퀀스에 대한 다른 인접한 구역을 생성시킨다. 그러나, 이 경우에는 미리 규칙적인 증가분을 사용하여 계산되어지기보다 난수 시퀀스에 있는 시작점이 각 프로세서에서 임의적으로 선택되어진다. 이것은 가능한 long-range 상관관계를 피할 수 있는 (최소한 줄일 수 있는) 이점이 있다. 하지만, 이 방법의 잠재적인 단점은 초기 seed가 임의적으로 선택되기 때문에, 다른 프로세서에서 생성되는 난수 시퀀스가 겹쳐지지 않는다는 보장이 없다. 각 프로세서에 있어서 seed 테이블의 초기화도 이 알고리즘의 중요한 부분이 된다.

우수한 병렬 난수 발생기를 찾는다는 것은 매우 도전적인 문제로 증명이 되었고, 또 여전히 연구와 논쟁의 주제로 남아있다. 그 이유 중에 하나가, 순차 난수 발생기에 존재하는 작은 상관관계라도 프로세서들 사이에서 난수 시퀀스를 생성시키기 위한 방법에 의해 증폭될 수 있고, 각 프로세서 위의 난수 시퀀스에서 더 강한 상관관계를 만들 수가 있기 때문이다.

이번에는 우리가 테스트에 사용한 난수 발생기들을 간단히 소개한다. 사용된 대부분 난수발생기가 널리 알려진 것이기 때문에, 구체적인 알고리즘에 관한 설명은 각 주석을 참고하면 되겠다.

먼저, Scalable Parallel Random Number Generator(SPRNG)는 순차 계산뿐 아니라 병렬 계산을 위해서 몇개의 난수 발생기를 포함하고 있는 잘 알려진 라이브러리이다[4]. SPRNG는 University of Southern Mississippi와 NCSA (the National Center for Supercomputing Application)의 연구자들이 개발하여 자유롭게 배포되었다. 구성은 컴퓨터 언어인 Fortran, C, 그리고 C++로 짜여져 있고, 통계적인 물리 테스트 방법도 소개하고 있다. 여기에 포함된 난수발생기들의 속도도 어느 난수발생기와 견주어 볼 때 빠르다고 할 수 있다. SPRNG에 포함하고 있는 난수 발생기는 아래와 같다.

- Modified Additive Lagged-Fibonacci Generator (LFG)
- Multiplicative Lagged Fibonacci Generator (MLFG)
- 48-bit Linear Congruential Generator (LCG)
- 64-bit Linear Congruential Generator (LCG64)
- Combined Multiple Recursive Generator (CMRG)

다음은 Unix routine에서 기본적으로 제공하는 srand와 drand48을 사용하였는데, 우리가 아주 일반적으로 흔히 쓰고 있는 난수발생기이다. 그리고, Mersenne Twister(MT)라는 난수발생기는 1996년/1997년에 일본의 M. Matsumoto와 T. Nishimura에 의해 개발되었다[5]. 알고리즘 자체는 Linear Congruential Generator를 쓰는 Twisted Generalized Feedback Register(TGSFR)[6]의 향상된 종으로 알려져 있다. 마지막으로 KISTI의 IBM 머신 ESSL 라이브러리에 있는 DURAND와 DURXOR 난수 발생기를 사용하였다[7].

Ⅲ. 결과 및 논의

Ⅲ-1. Monte Carlo 방법에 의한 Pi 계산 성능 비교

우리는 π를 계산하기 위해 반복적인 순환방식을 통한 통계적인 MC 방법을 사용하였다. 이미, 정확한 π값을 알고 있으므로 난수 발생기의 성능을 테스트하는데 이 방법이 유용하게 이용될 수 있다.

■ Algorithm

π 계산은 단위 반지름을 가진 원의 면적을 구함으로써 행해진다. 그림 3.1에서처럼 사각형 안에 원점을 중심으로 하는 원을 그릴 수 있다. 어떤 조건안에서 사건이 일어날 확률을 P라고 했을 때, 컴퓨터는 그러한 조건을 반복적으로 재생하는데 사용될 수 있기에 편리하다.

수많은 시행 화살이 직사각형 $OABC$안에 던져진다고 가정해 보자. 우리는 (0,1) 사이의 균일한 분포로부터 선택되어진 독립적인 두 개의 난수를 발생시킨다. 이 값들은 그대로 화살이 떨어진 점의 좌표로 사용되고, 원점으로부터 그 점까지 거리를 계산하다. 만약, 그 거리가 1보다 같거나 작으면, 화살은 OAC의 검은 부채꼴 면에 존재하는 것이고 카운터를 하게 된다. 이런 식으로 반복하게 되면 π는 다음과 같이 구할 수가 있다.

$$\pi \approx 4 \times \frac{\text{부채꼴 } OAC\text{안의 점 개수}}{\text{사각형 } OABC\text{안의 점 개수}} = 4P \qquad (3.1)$$

π값이 얼마나 정확한가는 MC 반복 횟수가 얼마나 많은지에 의존하게 된다. 이와 같이, MC 방법은 물리나 수학에서 해석학적으로 풀 수 없는 문제들을 푸는데 종종 이용될 수 있지만, π값을 계산하는 데는 다소 느린 편이다. 하지만, π값 계산 과정과 결과 비교를 통해 난수 발생기들의 성능이나 우수성을 비교해 보기에

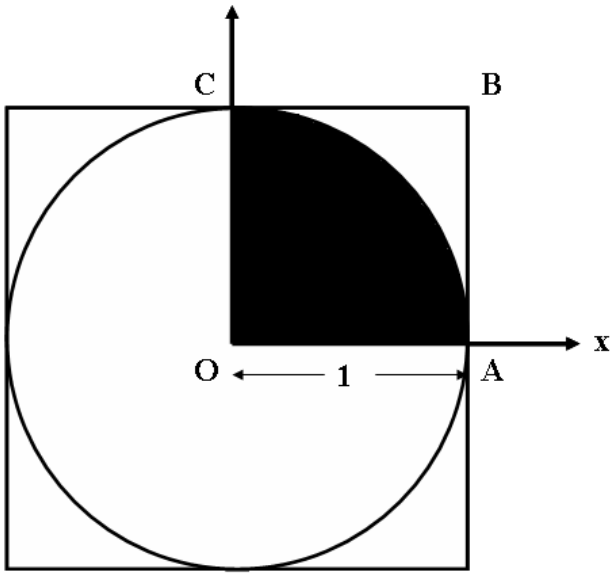


그림 3.1 MC 방법으로 π계산을 위한 단위 반지름 원

충분하다.

■ 순차(sequential) 계산 결과

MC 방법에 의한 π계산을 통해 난수 발생기들의 성능을 비교, 분석하였다. MC 반복 시행횟수 10^8번을 거쳐 유도되는 π값과 계산에 소요된 시간을 비교하였다. 생성된 난수 시퀀스의 무작위성(randomness)이 좋을수록, 계산된 π값의 정확도도 증가한다고 볼 수 있다.

먼저, 그림 3.2에서 π계산에 소요된 시간을 비교해보면, IBM 머신의 ESSL 라이브러리에 있는 난수발생기들이 가장 빠르게 나타났다. DURXOR이 6.38초로 가장 빠르게 나타났고, DURAND가 7.84초로 다음 순위였다. SPRNG 라이브러리안에 있는 LFG(21.81초), LCG(17.72초), LCG64(15.10초), CMRG(27.65초), MLFG(22.41초)의 계산 시간은 상대적으로 다른 난수 발생기와 견줄 만하였고, Mersenne Twister도 17.42초로 양호한 것으로 나타났다. Unix routine에 있는 DRAND48은 27.27초, SRAND는 28.17초로 기록되었으며 가장 느린 편이었다. 가장 빠른 DURAND와 가장 느린 SRAND의 시간 차이를 비교해 보면 무려 21.79초의 시간차로 나타났다.

그림 3.3는 각 난수발생기의 의한 π값 계산 결과에서 정확도를 보여주고 있다. 계산에 앞서, 우리는 정확한 pi값을 3.14159265358979332로 알고 있다. 반복 시행횟수가 10^8인 경우에는 모든 난수 발생기에 대해 99.990%이상의 우수한 결과를 보였다. 그 중에서도 DRAND48이 가장 높은 정확도를 보였고, 상대적으로 SRAND가 가장 낮게 나타났다. SPRNG에 있는 모든 난수 발생기들도 양호한 결과를 나타내었다. 이번 π계산의 순차 계산에서 그림 3.2과 그림 3.3의 결과만을 놓고 보면, 우리가 일반적으로 많이 쓰는 SRAND가 정확도면에서 가장 떨어지고, 계산시간도 다른 난수 발생기에 비해 가장 느린 것으로 판명됐다.

그림 3.4 (a), (b)의 결과는 MC 반복 횟수를 함수로 한 π계산 결과다. 10번부터 시작하여 10^8번까지 점차적으로 반복횟수를 늘이면서 π를 계산하였다. 시행 횟수가 1000번까지는 대체적으로 통계적인 오차 변동 폭이 커다가, 그 이후부터 수렴

해나가는 것을 볼 수 있다. 시행횟수가 작은 10번의 경우에, SPRNG의 LCG와 Mersenne Twister가 정확한 π값에 가장 근접했으며, MLFG와 DURAND에 의한 π값이 가장 큰 오차를 보였다. DURXOR은 큰 주기의 난수를 발생시키는 난수 발생기로 시행횟수가 작을 때는 측정이 불가하였고, 1000번의 시행횟수에서도 가장 좋지 않은 π계산 결과를 보였다. 다른 난수발생기에 비해 Mersenne Twister가 근접한 π값으로 시작하여 가장 안정적인 변동폭으로 정확한 π값에 수렴해나가는 것을 볼 수 있다. 전체적인 π계산 결과 추이를 살펴보면, 정확한 π값을 얻기 위해서는 높은 차수의 MC 반복횟수가 요구되는 통계적인 사실을 알 수 있다.

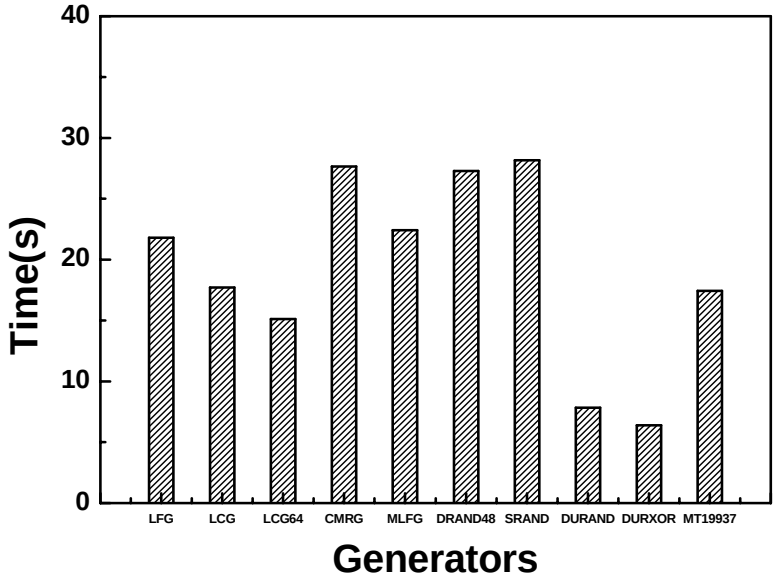


그림 3.2 난수 발생기에 따른 π계산에 시간

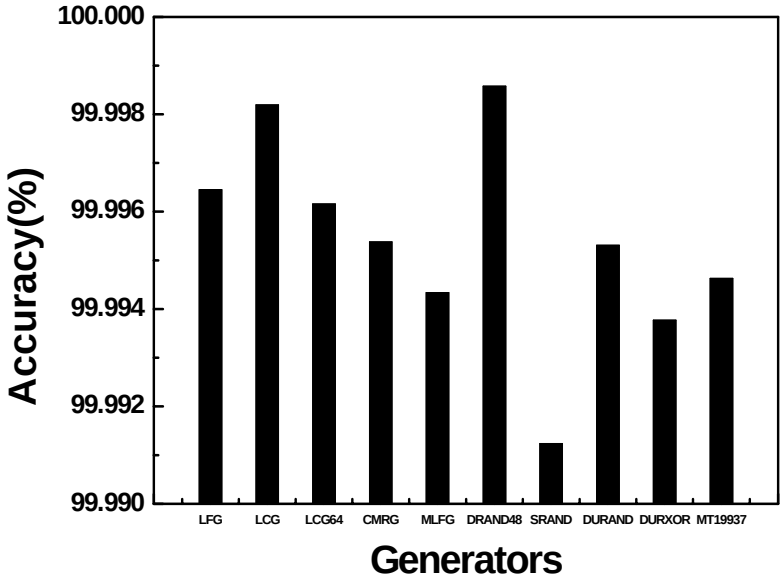


그림 3.3 난수 발생기에 따른 π계산 정확도

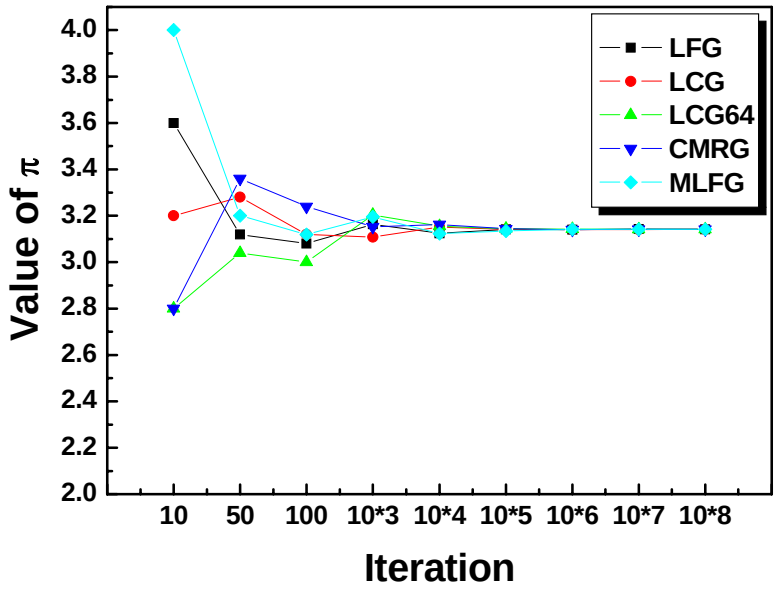


그림 3.4 (a) MC 반복 시행 횟수에 따른 π계산 결과 ①

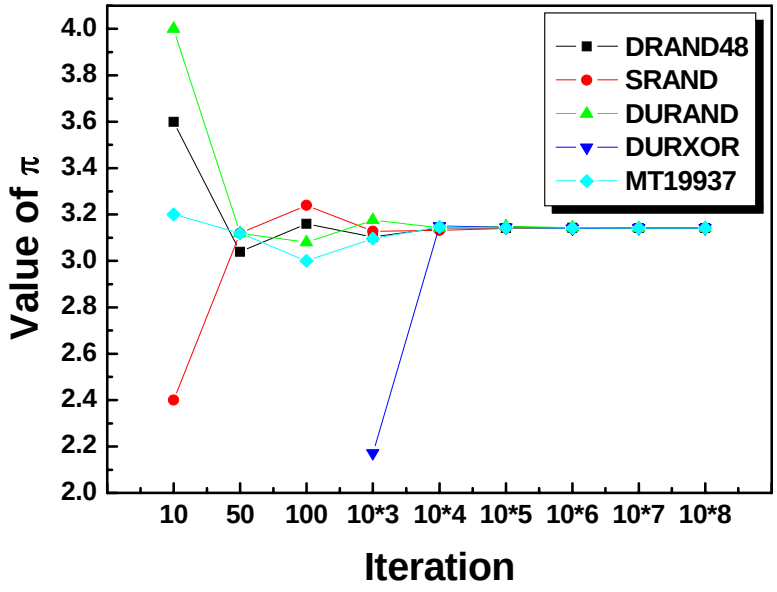


그림 3.4 (b) MC 반복 시행 횟수에 따른 π계산 결과 ②

■ 병렬(Parallel) 계산 결과

앞서, 순차적으로 진행되었던 π 계산을 동일 조건의 MC 시행 횟수로 CPU 여러 개로 나누어 여러 프로세서에서 동시에 수행되도록 하였다. 이렇게 프로그램으로 병렬 처리를 함으로써 각 난수 발생기의 성능을 테스트해 보았다. 그림 3.5 (a), (b)는 CPU 개수 증가에 따라 π계산에 소요된 시간 변화 그래프이고, 그림 3.6 (a), (b)는 이에 따른 성능 향상도(Speed-up) 그래프이다. 여기에서 성능 향상도라 하는 것은, 순차 프로그램에 대한 병렬 프로그램의 성능 이득 정도로써 다음과 같이 정의된다.

$$S(N) = \frac{\text{순차 프로그램의 실행시간}}{\text{병렬 프로그램의 실행시간}(N\text{개의 프로세서})} = \frac{t_s}{t_p} \tag{3.2}$$

가령, 실행시간이 100초가 걸리는 순차 프로그램을 병렬화하여, 10개의 프로세서로 50초만에 실행이 되었다면, 성능 향상도는 $S(10) = \frac{100}{50} = 2$가 되는 것이다. 이번 π계산에서는 CPU 개수를 2개씩 증가하여 총 14개까지 테스트하였다. 그림 3.5 (a), (b)와 그림 3.6 (a), (b)의 결과를 함께 살펴보도록 한다. 먼저, SPRNG에 있는 난수 발생기들에 의한 결과에 의하면 단일 CPU의 순차 계산에 비해 CPU 2개를 사용했을 때 모두 2이상의 성능 향상도(2.35~2.5)를 보였다. DRAND48과 DURAND도 각각 2.61과 2.93으로 나타났고, SRAND의 경우에는 4.44로 무려 4이상의 성능 향상을 보였다. 반면, DURXOR과 Mersenne Twister는 각각 1.86과 1.53으로 다른 난수 발생기에 비해 상대적으로 낮았다. 결과 그래프에서 보는 것처럼, CPU 개수를 점차적으로 늘였을 때 SPRNG에 포함된 난수 발생기는 전체적으로 성능 향상도가 고르게 증가하였으며, CPU 개수가 14개인 경우에는 S(14)=16.6~17.56의 비슷한 성능 향상을 보였다. 이에 대한 성능 효율(efficiency)를 계산해보면 E(14)=118.57~125.43%의 효율을 나타내었다. 그러나, 그림 3.6 (b)에 있는 난수 발생기들의 비교 결과는 성능 향상도가 고르게 증가하지 않으면서 그 결과도 차이가 난다. 상대적인 결과 비교에 의하면, CPU 개수가 14개일 때

DURXOR은 S(14)=7.42, E(14)=53%로 가장 낮았고, Mersenne Twister가 S(14)=10.13, E(14)=72.36%로 나타났다. 다음으로 DURAND는 S(14)=12.06, E(14)=86.14%였고, DRAND48은 S(14)=23.92, E(14)=170.86%의 순으로 높게 나타났다. SRAND가 S(14)=30.29, E(14)=216.36%로 가장 높은 성능 향상도을 보였다. 그림 3.7 (a), (b)그래프는 CPU 증가에 따른 π계산에서, 정확한 π값으로부터 오차 변화를 보여준다. 이전 순차계산 결과에서 0.01%이내의 오차 범위를 보이던 난수 발생기에 대한 결과 값이, 병렬 계산에 의하면 최대 0.021%의 오차 편동 폭을 보여주기도 한다. 이것은, 단일 CPU 계산에 비해 병렬 처리를 함으로써 발생될 수 있는 문제 즉, 프로세서 상호간의 상관관계 및 한 프로세서 내에서도 난수 시퀀스의 질이 순차 계산보다 우수하지 못하는 것으로 판단된다.

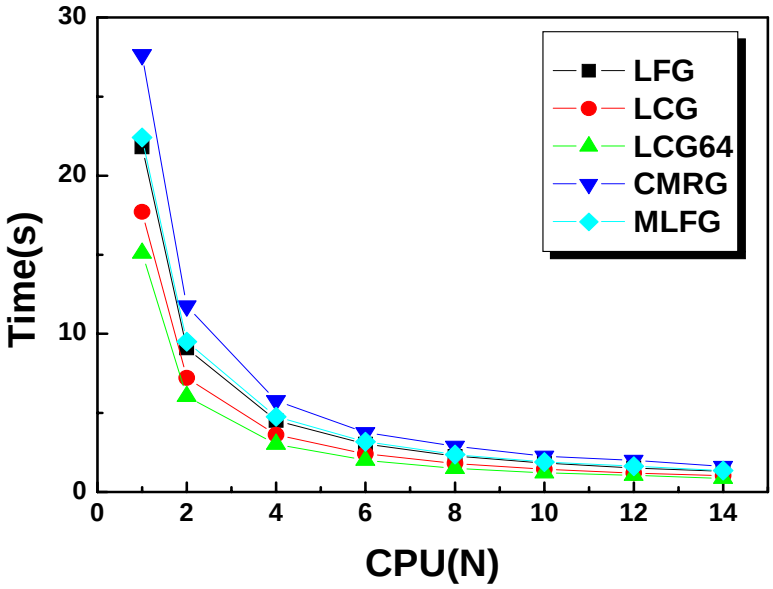


그림 3.5 (a) CPU 개수 증가에 따른 π계산 시간 변화 ①

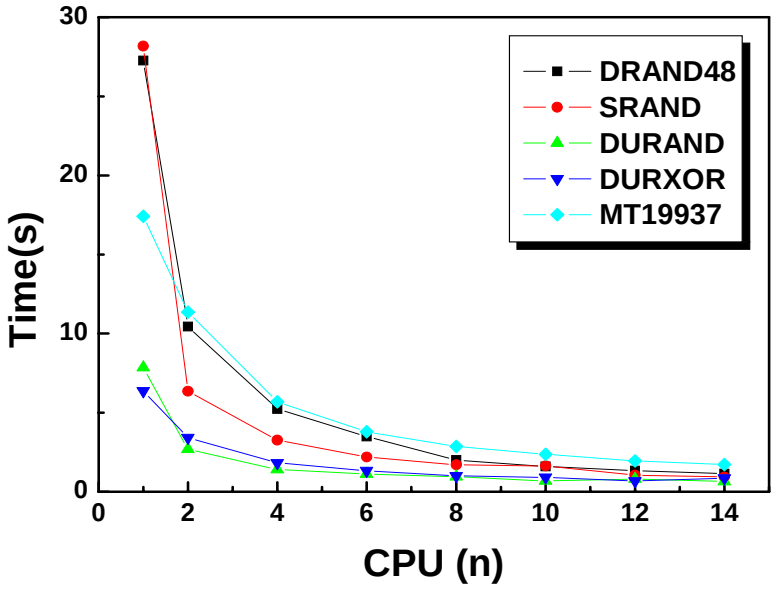


그림 3.5 (b) CPU 개수 증가에 따른 π계산 시간 변화 ②

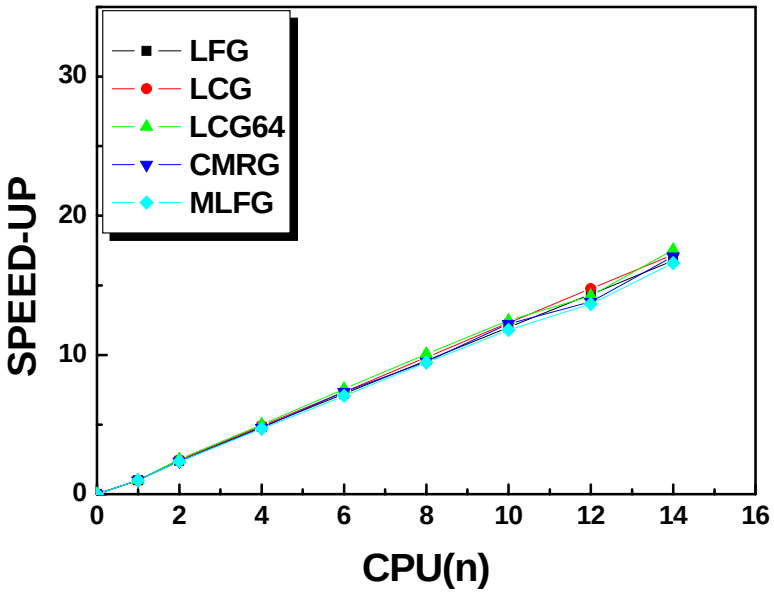


그림 3.6 (a) CPU 개수 증가에 따른 성능 향상도 ①

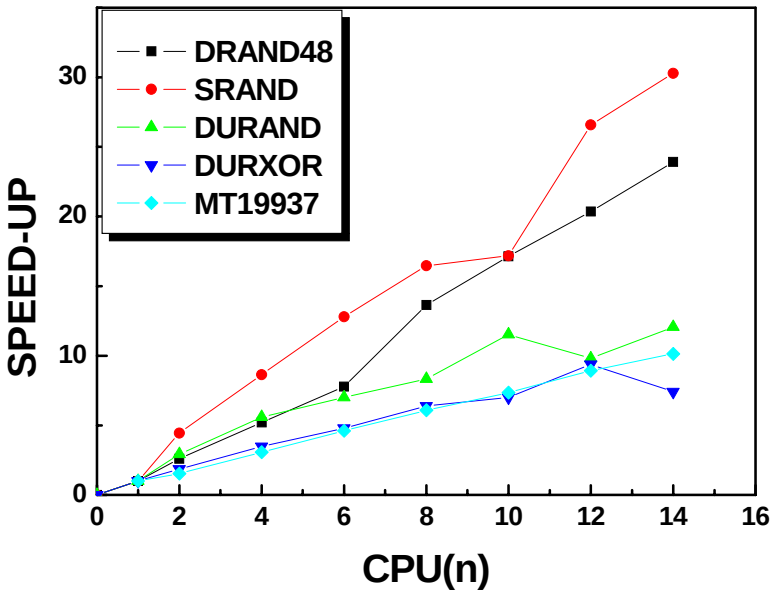


그림 3.6 (b) CPU 개수 증가에 따른 성능 향상도 ②

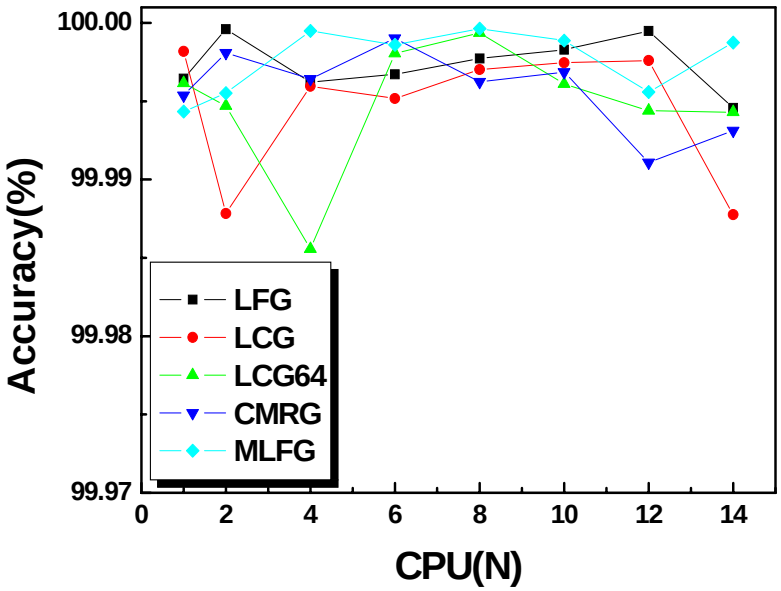


그림 3.7 (a) CPU 증가에 따른 π계산 결과 정확도 변화 ①

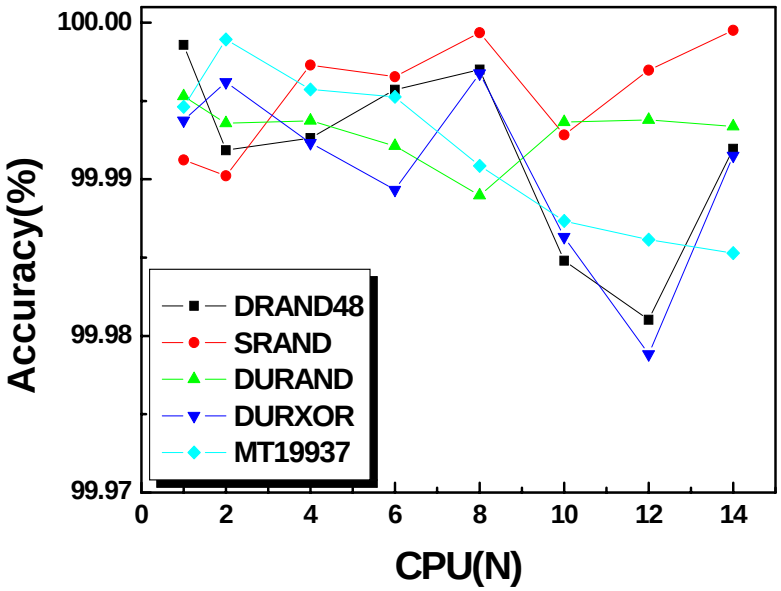


그림 3.7 (b) CPU 증가에 따른 π계산 결과 정확도 변화 ②

Ⅲ-2. Diffusion-limited aggregation(DLA)의 fractal dimension 비교

우리는 실생활에서부터 과학적인 현상에 이르기까지 입자들이 결합해 가면서 하나의 큰 결정체로 성장하는 현상을 흔히 관찰한다. 이러한 성장 과정과 그것을 설명하려는 모델은 그 관련성 때문에 많은 분야에서 연구되어지고 있다. 특히, Diffusion-limted aggregation (DLA)은 1981년 Witten, T.A. and Sander, L. M[8]에 의해 제안된 이후 계속해서 활발하게 연구되고 있고, 입자들의 패턴 형성 성장에 있어서 가장 표준적인 모델이 되었다. DLA는 그림 3.8와 같이 눈의 결정(snowflake growth), 산호초 성장(coral growth)에서 부터 전해질 석출(electrodeposition), 유전체 방전(dielectric breakdown), 결정 성장(crystal growth)과 같은 현상을 설명할 수 있는 흥미로운 모델이다[9,10]. 비교적 간단해 보이는 DLA는 다소 적은 물리량의 사용으로 많은 자연의 필수적인 현상을 설명할 수 있도록 기대한다. 물리적인 관점에서 DLA를 흥미롭게 만드는 것은 생성된 결정체의 프랙탈(fractal) 구조이다. 간단한 알고리즘이 자기 유사한(self-similar) 프랙탈 구조를 만든다는 수학적 사실은 놀란 만한 일이다. 임의적이면서 중요치 않아 보이는 성장 과정이 실제 시스템을 많이 닮아 보이고, 자기 유사한 복잡성을 가진 결정체를 형성한다. 흥미를 끄는 이유가 자연에서 흔하게 일어나기 때문이다. 이 모델이 널리 알려지고 연구되어 지고 있음에도 불구하고, 우리가 정확하게 분석하고 풀어갈 만한 이론은 아직까지 요원한 단계다. “Diffusion-limited"란 하나의 결정체를 생성해 갈 때, 입자들이 낮은 농도에서 서로 충돌하지 않고 한 번에 한 입자씩 결정체에 결합해가는 것을 의미한다. 이러한 과정이 복잡하면서도 신기한 프랙탈(fractal) 구조를 성장시킨다. 지금까지, DLA에 관한 이론적인 노력들은 이 프랙탈 구조와 멀티 프랙탈의 특성에 두었고, 무엇보다 프랙탈 구조의 차원을 계산하는데 집중하여 왔다. 프랙탈 차원을 구하기 위한 정확한 공식은 없다. 광범위한 연구에도 불구하고, 프랙탈 차원에 대한 정확한 계산값을 얻기 위한 일들은 여전

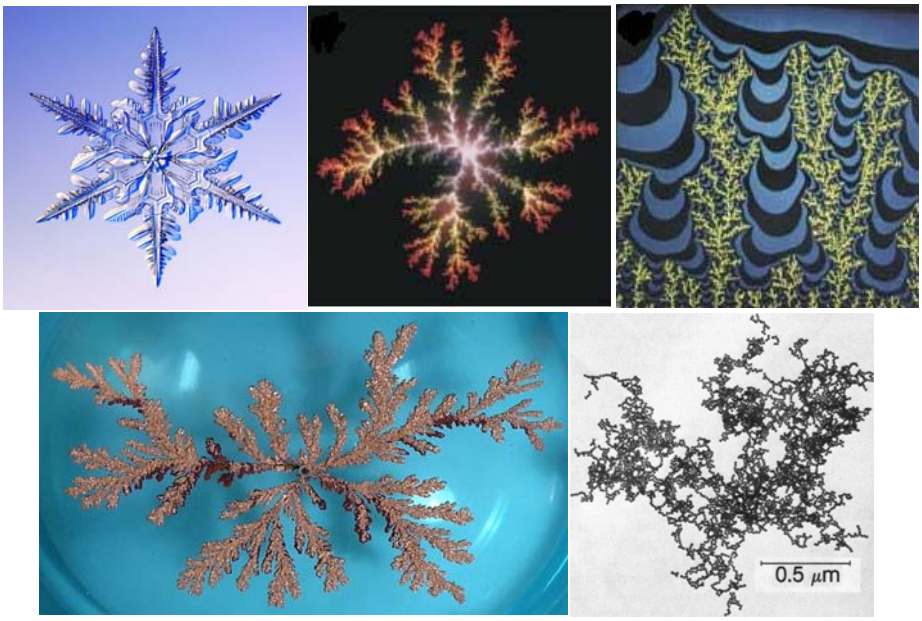


그림 3.8 DLA cluster의 응용분야

: 눈의 결정, 방전 현상, 산호초 성장, 전해질 석출, 콜로이드 입자 결정체

히 과제로 남아있다.

■ Algorithm

DLA는 확률적 과정(Stochastic process)으로 난수발생기에 의한 random walk 방법이 지배적이다. 수치적으로 DLA을 계산하는 방법은 여러 가지가 있지만, 가장 일반적인 알고리즘은 다음과 같다.

먼저, 원점에 핵 입자(seed particle) 입자를 놓는다. 그 핵입자로부터 떨어진 임의의 지점에 또 다른 입자를 놓고 random walk을 실행시킨다. 이 입자가 핵입자 경계지점에 도착하면 하나의 결정체를 이루는 것이고, 만약 너무 멀리 떨어진 지점에 도착하면 무시하게 된다. 또 다른 두 번째 입자가 멀리 떨어진 지점에 놓여지고, 확산에 의한 random walk이 진행되면서 이 결정체에 근접하면 종료된다. 하나의 큰 결정 집합체가 이루어지까지 이러한 과정이 계속해서 반복되어진다. 이러한 알고리즘이 매우 간단해 보일지 모르지만, 여러 응용분야에 적용하고 정확한 답을 구하기 위해서 많은 이론 과학자들에게 관심을 불러일으켰다.

■ 프랙탈 구조의 차원(fractal dimension) 계산

DLA는 프랙탈 성장의 중요한 원형(prototype) 모델로써, 지금까지 프랙탈 차원의 정량적인 측정에 많은 관심을 보여 왔다. 프랙탈 차원은 결정 구조의 특성을 정의하는 중요한 변수로 작용한다. 이러한 차원의 문제에 접근하는 것에는 몇몇의 방법이 있지만, 여기서는 가장 일반적 방법을 소개한다.

우리가 직관적으로 잘 알고 있듯이, 직선은 1차원이고, 평판의 구조는 2차원이다. 2차원 균질한 평판에서 반지름 r의 원에 포함된 질량을 고려하면 다음의 식이 된다.

$$m(r) = \sigma \pi r^2 \tag{3.3}$$

σ는 단위 면적당 질량이다. 여기서 질량은 r^2에 비례하고 2승은 대상의 차원이다.

만약, 직선이나 유사한 곡선 모양을 고려한다면 질량은 다음과 같다.

$$m(r) = 2\lambda r \tag{3.4}$$

λ는 단위길이 당 질량이다. 질량은 r^1에 비례하게 되고, 1승은 다시 대상의 공간 차원을 나타낸다. 우리는 이러한 기본적인 생각으로부터 DLA cluster의 효율적인 차원을 계산할 수 있다. 다음은 이로부터 유도된 정의이다.

$$m(r) \sim r^{d_f} \tag{3.5}$$

식(3.5)의 d_f가 우리가 측정하고자 하는 대상에 대한 프랙탈 차원(fractal dimension)된다. 이로부터, DLA에 의한 프랙탈 구조에 적용하기 위해 질량중심으로부터 거리 r 만큼 떨어진 원내부의 질량을 계산할 필요가 있다. 편의상 원점의 초기 seed 입자를 선택한다. 모든 입자들이 똑같은 질량이라고 가정했을 때 원점에서 r만큼 떨어진 거리 안에서 입자의 수를 헤아리면 $m(r)$이 된다. DLA cluster에서 r에 대한 함수의 질량값을 다음과 같이 로그 스케일(logarithmic scale)로 그려봄으로써 우리가 원하는 d_f값을 쉽게 얻을 수가 있다.

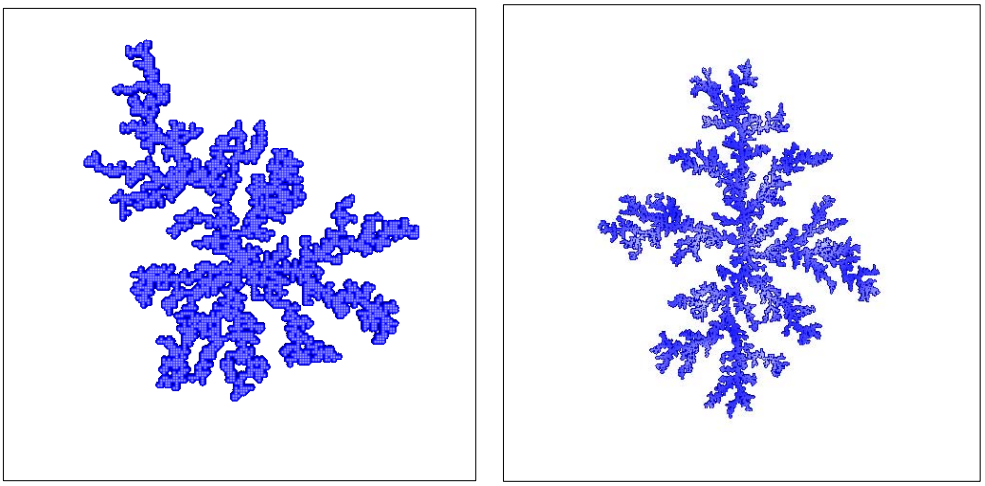
$$\log m \sim d_f \log r \tag{3.6}$$

식(3.6)에서 log-log plot의 기울기가 프랙탈 차원(d_f)이 되는 것이다. DLA cluster의 프랙탈 구조와 그 차원에 대한 연구는, 결정 구조가 정량적으로 얼마나 밀집한지 가늠할 수 있는 한 방법을 제시해 준다고 할 수 있다.

DLA에 대한 컴퓨터 시뮬레이션은 2-dimension square lattice 위의 동일한 조건에서 각각 다른 난수 발생기를 이용하여 앞서 소개한 알고리즘을 수행시켰다.

■ DLA cluster 형성 패턴

먼저 시뮬레이션 결과에 따른 DLA cluster 패턴에 대해 살펴보았다. DLA 패턴 결과는 난수 발생기 및 동일한 난수발생기에서도 각각 다르게 주어진 initial seed에 의해 다양한 모양을 생성시켰다. 그림 3.9는 계산된 DLA cluster의 한 형태를 보여주고 있다. 그림 3.9 (a) 이미지는 격자 크기가 200×200이며 DLA cluster를 구성하는 있는 입자수는 3,167개 이다. 그리고, 그림 3.9 (b)는 격자 크기가 400×400이며 입자수가 13,910개인 DLA cluster이다. 패턴 결과에 의하면, 형성된 DLA cluster의 입자수가 적은 경우에는 그 패턴이 비대칭적이며, 가지(branch) 모양도 지그재그(zigzag)인 것처럼 보인다. 격자 크기가 큰 경우 즉, 생성된 DLA cluster의 입자수가 클수록 가지 수도 많아지고, 상대적으로 패턴도 균일하면서 대칭적인 모양으로 변해가는 것을 볼 수 있다. DLA cluster가 형성해 나갈 때 즉, 프랙탈 구조로 변해가면서 성장할 때 입자는 원점에 가까운 안쪽 깊숙이 들어가기 보다는 경계면에 성장한 가지들에 달라붙을 확률이 더 많아지게 된다. 가지 수가 많아질수록 대칭적으로 변해가는 DLA cluster 구조의 변화는 쉽게 이해할 만한 사실이다. 그림 3.9 (a)가 그림 3.9 (b)에 비해 패턴이 커 보이고 입자수가 많아 보이지만, 실제적으로 그림 3.9 (b)가 격자 크기가 더 크고 입자수가 많은 경우이다. 그림을 명확하게 보여주기 위하여 가시화 프로그램을 같은 크기로 그렸기 때문이다.



(a) Lattice size 200×200　　(b) Lattice size 400×400

그림 3.9 DLA cluster 형성 패턴

■ 프랙탈 차원(Fractal dimension, d_f)

앞서 결과에서, DLA cluster의 패턴은 사용한 난수 발생기, 주어진 initial seed, 격자 크기(형성된 입자수)에 따라 다르게 나타난다는 것을 간략하게 언급하였다. 우리는 DLA cluster에 사용된 난수 발생기들의 의존성을 살펴보기 위하여, 물리적인 양인 프랙탈 차원(fractal dimension)을 측정하였다. 그림 3.10 그래프는 프랙탈 차원을 계산하기 위한 과정을 나타내었다. 앞에서 설명했던 식(3.6)에 의해 $\log m$ vs $\log r$의 기울기가 d_f가 되는 것이다. 이때, 각 데이터들의 값으로부터 기울기를 구하는 방법은 Least-squres fitting을 이용하였다. 그림 3.10의 결과를 보면, r이 작은 경우에는 그래프의 결과가 직선의 기울기와 일치한다. 그러나, r이 큰 경우에는 데이터 커브가 약간 굽어지면서 평편하게 된다. 이것은 DLA cluster의 크기가 유한하기 때문이다. 이러한 방법을 이용하여 계산한 하나의 결과값 d_f는 1.69로 계산되었다. 우리가 직관적으로 알 수 있듯이, DLA cluster의 d_f는 1보다 크고 2보다 작은 값을 가지게 된다. 이것은, 실제로 DLA cluster가 왜 프랙탈인지 이유가 되는 것이다. DLA cluster가 유효한 차원을 나타내기 위해서 질량은 r^2보다 더욱 천천히 증가하여야 한다. 이것은 우리가 시뮬레이션 결과에서 관찰할 수 있었던 것처럼 hole이나 crack을 포함하고 있어야 된다는 것을 의미하게 된다.

다음은 사용한 각 난수 발생기에 대해서 initial seed값을 바꿔가면서 2개에서 100개에 이르는 DLA cluster를 생성시켜 d_f를 계산하였다. 이때, d_f는 연속된 개수에 대해 독립적인 값이고, 각각 생성된 개수만큼 평균되어졌다. 에러바(error bar)는 통계적인 변동(statistical fluctuation) 값으로부터 구하였다. 우리가 원하는 d_f값이 일관되고 신뢰할 만한 값을 갖기 위해서는 계산된 모든 값들이 통계적인 에러 범위 안에 있어야 한다. 그림 3.11 결과들에서 보듯이, 10개 미만에 대한 각각의 평균값에서 d_f가 변동을 하면서 그 값이 차이가 많이 벌어지는 경우도 볼 수 있다. 그럼, 과연 하나의 난수발생기를 사용하여, 한 두번의 측정으로 우리가 계산한 DLA cluster의 d_f값이 신뢰할 만한지 의문을 갖게 된다. 이번 연구 결과를 보면 긍정적인 답은 아니다. SPRNG에 있는 난수 발생기와 대부분의 난수발생기에서 d_f값이 에러 범위를 벗어나는 경우를 볼 수 있다.

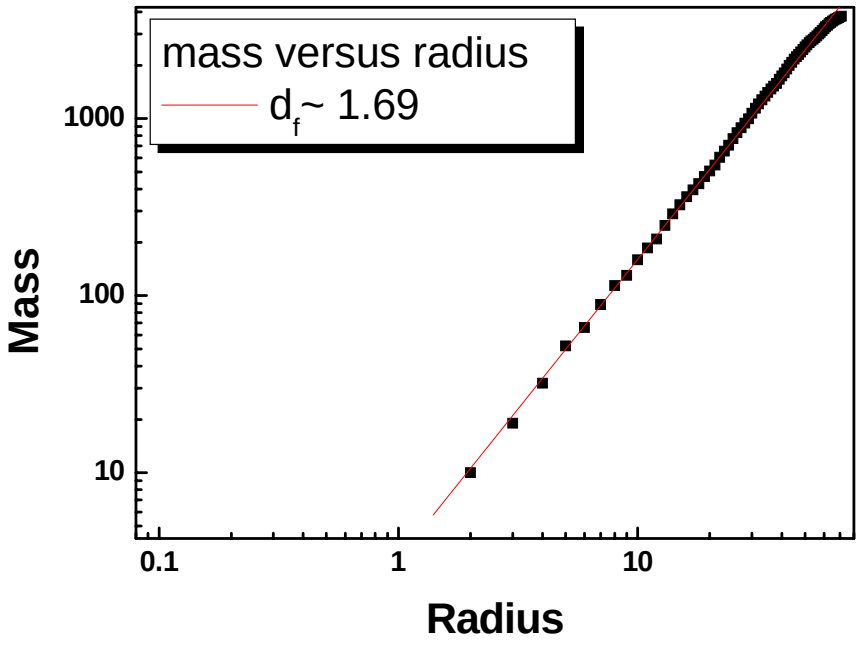


그림 3.10 DLA cluster에 대한 logm vs logr의 그래프.

Least-squres fitting에 의한 직선의 기울기 값이 d_f가 된다.

이번 DLA cluster에 적용한 난수 발생기의 테스트 결과를 분석해보면, 최소한 한 두 번의 수행으로 계산한 d_f는 우리가 신뢰할 만한 결과 값이 될 수 없다는 결론이 된다. 당연한 결과겠지만, 통계적인 횟수를 증가시킴에 따라 d_f는 점점 더 수렴해나가고 받아들일 만한 측정값이 된다고 볼 수 있다.

그림 3.12 그래프는 사용된 각 난수발생기에 대해서 DLA cluster 100개의 d_f평균값이고 d_f를 직접적으로 비교한 것이다. 대부분의 난수 발생기 결과에 의하면 통계적인 에러 범위 안에서 d_f값이 거의 일치하는 것을 알 수 있다. 여러 문헌에 의하면 d_f는 격자구조, 입자크기, cluster 중심에서 부터의 상대적인 거리에 따라 변할 수 있다고 알려져 있다[11-13]. 본 DLA cluster에 대한 난수 발생기 테스트는 조건이 모두 동일하였다. 결과 그래프에서, 난수발생기 SPRNG를 포함하여 다른 난수발생기를 이용해 계산한 DLA cluster의 d_f는 통계적인 에러 범위안에서 수렴하여 받아들일 만하지만, 정확한 결과값에 있어서는 조금씩 차이가 나는 것을 알 수 있다. 이 결과만 놓고 보면, 어느 난수 발생기가 우수한 것인지 단정짓기는 어렵지만, 무난하게 비슷한 결과를 보이는 것으로 알 수 있다. MC 반복 횟수를 무한대로 늘여가면서 비교한다면, 물론 모든 난수 발생기에 의해 계산한 값들이 수렴해가면서 거의 일치할 것으로 판단된다.

만약, 특정한 난수 발생기를 DLA에 적용했을 때 다른 난수 발생기의 결과와 차이가 난다면 당연히 그 난수 발생기의 사용은 피해야한다. 통계적인 테스트는 무난히 통과했지만, 응용 프로그램 테스트에서 실패하는 경우가 있기 때문이다. 우리의 목표는 MC 방법을 이용한 컴퓨터 시뮬레이션에서 특정한 응용 프로그램을 테스트함으로써 그 결과를 비교하고자 한 것이다. 이를 통하여 일반적으로 쓰이고 있는 난수 발생기의 사용을 신뢰하기 위함이고, 또한 계산과학에서 요구되는 응용 프로그램을 위한 더 나은 난수 발생기를 개발하는데 쓰이기 위한 바람이다.

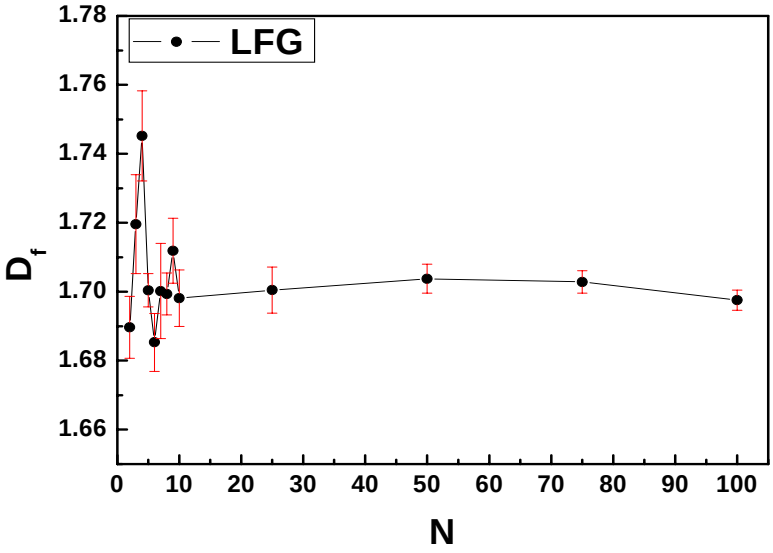


그림 3.11 (a) DLA cluster에 대한 LFG(SPRNG)의 d_f결과

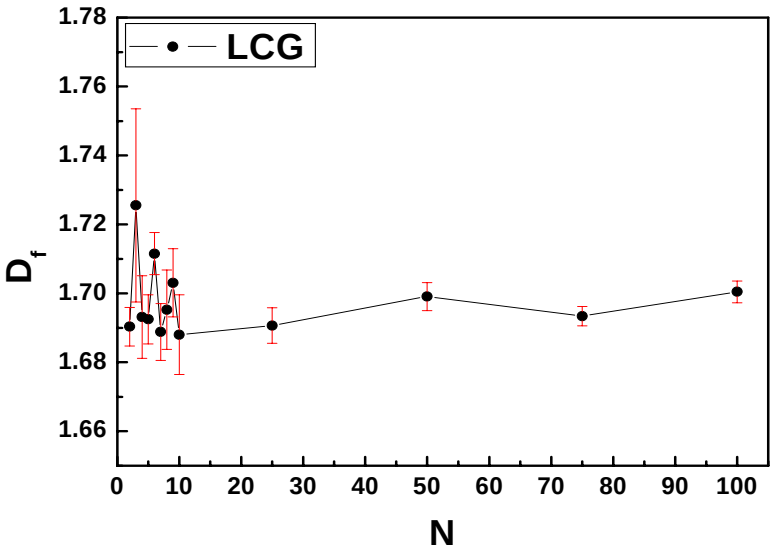


그림 3.11 (b) DLA cluster에 대한 LCG(SPRNG)의 d_f결과

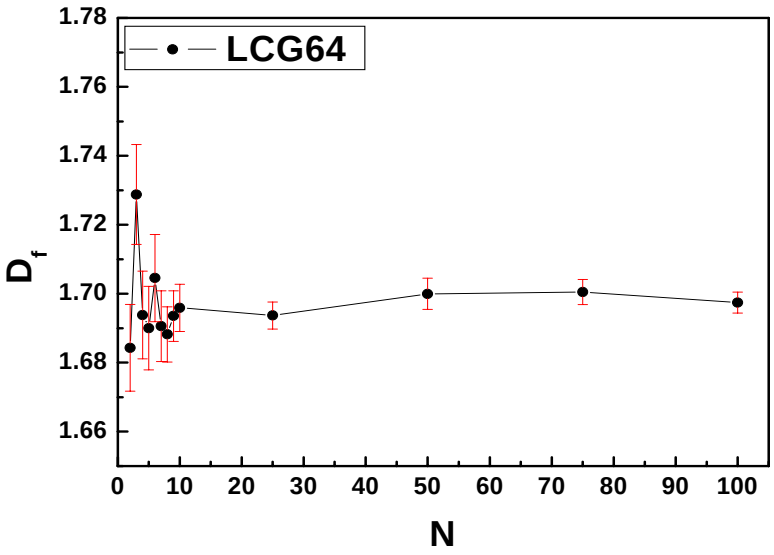


그림 3.11 (c) DLA cluster에 대한 LCG64(SPRNG)의 d_f결과

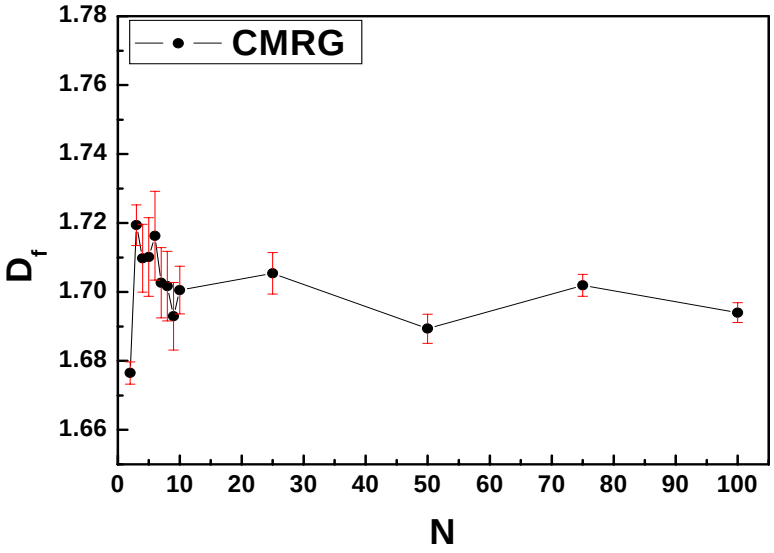


그림 3.11 (d) DLA cluster에 대한 CMRG(SPRNG)의 d_f결과

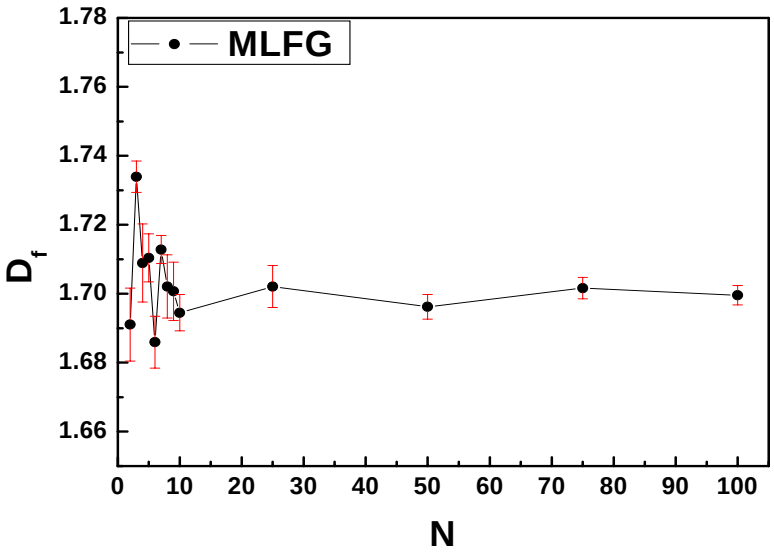


그림 3.11 (e) DLA cluster에 대한 MLFG(SPRNG)의 d_f결과

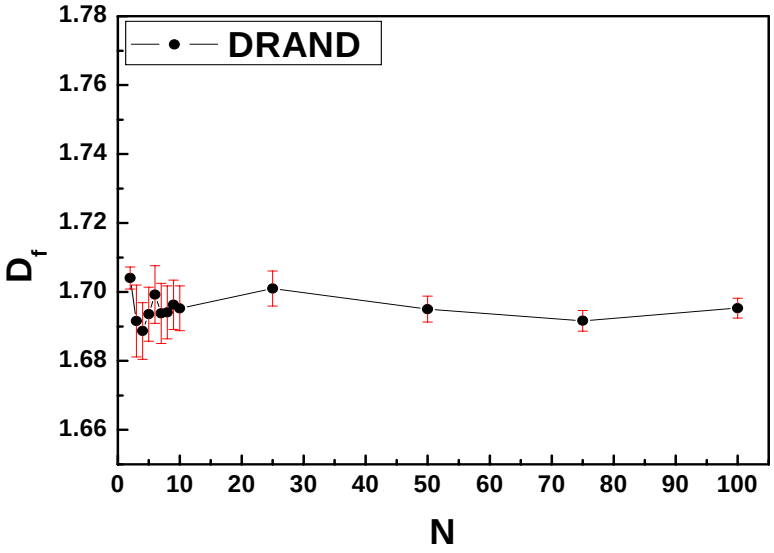


그림 3.11 (f) DLA cluster에 대한 DRAND의 d_f결과

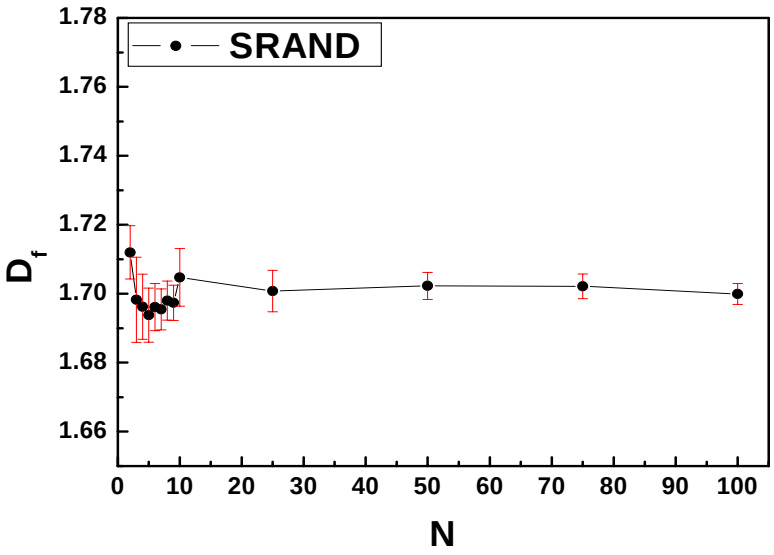


그림 3.11 (g) DLA cluster에 대한 SRAND의 d_f결과

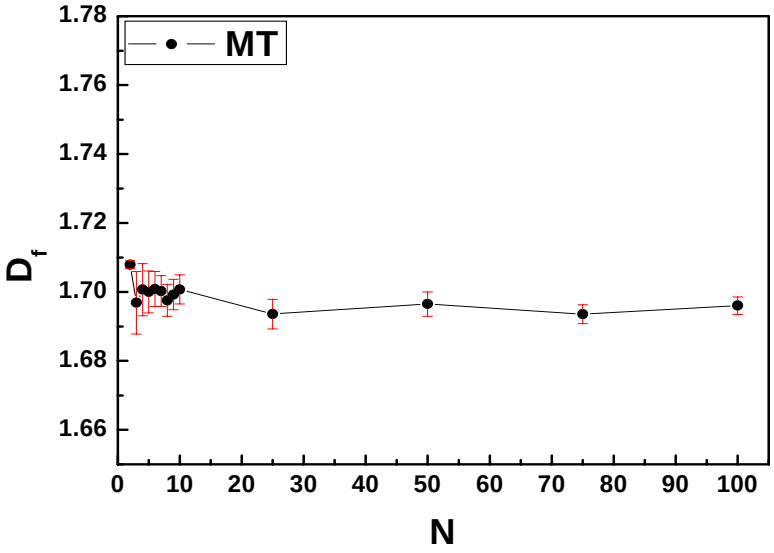


그림 3.11 (h) DLA cluster에 대한 Mersenne Twister의 d_f결과

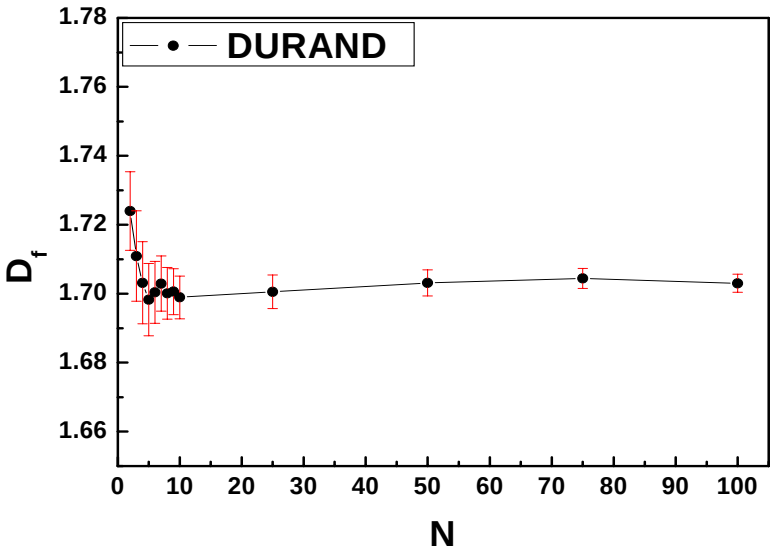


그림 3.11 (i) DLA cluster에 대한 DURAND(ESSL)의 d_f결과

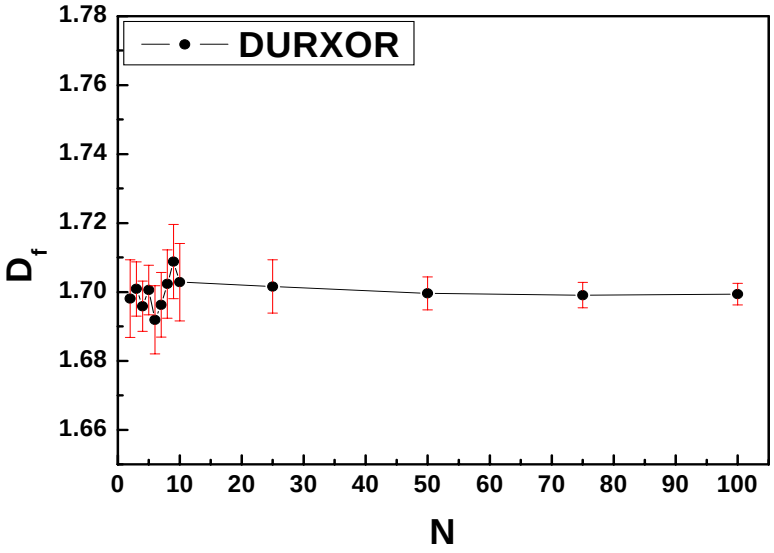


그림 3.11 (j) DLA cluster에 대한 DURXOR(ESSL)의 d_f결과

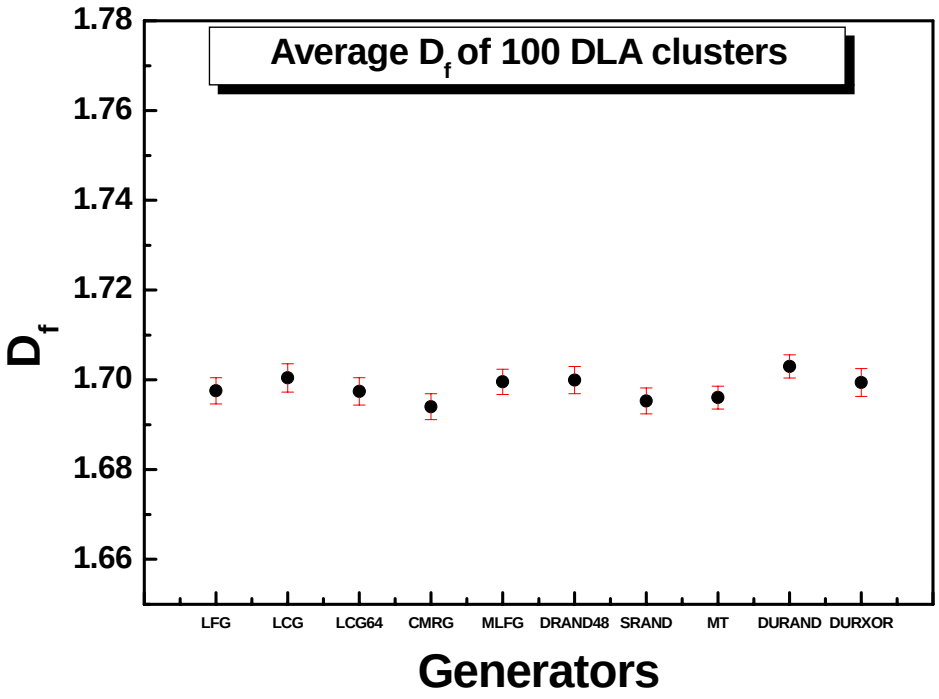


그림 3.12 DLA cluster 100개에 대한 난수 발생기들의 평균 d_f결과

Ⅳ. 결 론

이번 연구에서는 컴퓨터 수치 계산에 흔히 쓰이는 난수 발생기를 테스트하기 위하여, 우리가 정확하게 알고 있는 π값에 대해 순차 및 병렬 계산 결과를 비교해 봄으로써 그 성능을 살펴보았다. 순차 계산 결과에 의하면, 독립적으로 제공되는 SPRNG 라이브러리에 포함된 난수 발생기들은 계산 시간 및 정확도 면에서도 다른 난수 발생기에 비해 견줄 만하였고, 계산 속도는 IBM머신에서 제공되는 최적화된 ESSL라이브러리의 난수 발생기들이 가장 빠르게 나타났다. 병렬 계산 결과는 SPRNG 난수 발생기들이 대체적으로 균일한 성능 향상도를 보였고, SRAND가 가장 높았다. 정확도면에서는 순차 결과에 비해 병렬 계산에서 상대적으로 큰 오차도 보였지만, 비교적 비슷한 폭의 오차가 나타났다.

다음은 물리적인 DLA 모델에 난수 발생기들을 적용하였다. 사용된 난수 발생기와 주어진 initial seed값에 따라 DLA cluster의 패턴이 영향을 받는 다는 것을 보여주었고, 특히 다양한 난수 발생기를 이용하여 프랙탈 구조의 차원(d_f)을 비교해 봄으로써 그 신뢰도를 평가할 수 있었다. 난수 발생기에 대한 이론적 이해는 다소 제한되어 있고, 사실 어떤 통계적인 테스트도 그 질적인 우수성을 결정하는 데는 다소 무리가 있다고 보여 진다. 표준화된 통계적 테스트를 통과한 난수 발생기도 때로는 응용 기반의 테스트에서 실패하는 경우가 나타난다. 따라서, 가능하다면 광범위하고 많은 실험적인 테스트를 거치는 것이 중요하다. 우리가 컴퓨터 시뮬레이션을 이용하여 원하는 값을 얻기 위해서는 적어도 두 개의 난수 발생기를 사용하여야 할 것이고, 그 난수 발생기가 한쪽으로 치우치지 않는다는 것을 확신하기 위해서라도 그 결과는 비교되어져야 한다.

컴퓨터를 이용한 MC 계산과 다른 확률적인 시뮬레이션의 정확성을 향상시키기 위해서도 계속하여 더 좋은 난수 발생기의 알고리즘 개발이 필요할 것이고, 또한 그 우수성과 결함을 테스트하는 일도 병행되어져야 할 것이다.

Acknowledgement

본 보고서에서 테스트되어진 난수 발생기와 DLA 모델에 대한 수치적 계산은 모두 KISTI IBM p690 머신에서 실행되었다.

참고 문헌

[1] I. Vattulaine, T. Ala-Nissila, and K. Kankaala, "Physical Tests for Random Numbers in Simulaions", *Phys. Rev. Lett.* **73**. 2513 (1994)

[2] Alan M. Ferrenberg and D. P. Landau, "Monte Carlo Simulations: Hidden Errors from "Good" Random Number Generators", *Phys. Rev. Lett.* **69**. 3382 (1992)

[3] A. Srinivasan, M. Mascagni, David Ceperley, "Testing parallel random number generators", *Parallel Comput.* **29**. (2003) 69-94

[4] http://sprng.cs.fsu.edu/

[5] http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html

[6] Mersenne Twister:A Study on Random Number Generators, Project Prestudy, Nihat Karaoglu, Vrije Universiteit Brussel, April 2004

[7] http://www.gwdg.de/service/rechenanlagen/parallelrechner/sp_documentation/essl_3-2.0/essl002.html

[8] T. A. Witten and L. M. Sander, "Diffusion-Limited Aggregation, a Kinetic Critical Phenomenon", *Phys. Rev. Lett.* **47**. 1400 (1991)

[9] L. M. Sander, "Diffusion-Limited aggregation: a kinetic critical phenomenon?", *Contemp. Phys.* **41**, 203 (2000)

[10] http://en.wikipedia.org/wiki/Diffusion-limited_aggregation

[11] W. Ouyang, Z. J. Tan, X. W. Zou and Z. Z. Jin, "Pattern of diffusion-limited aggregation on nonuniform substrate", *Chaos Solitons & Fractals*, **17**. (2003) 189-193

[12] Z. J. Tan, X. W. Zou, W. B. Zhang, and Z, Z. Jin, "Influence of particle size on diffusion-limited aggregation", *Phys. Rev. E.* **60**. 6202 (1999)

[13] P. Ossadnik, "Multiscaling analysis and width of the active zone of large

off-lattice DLA", *Physica A* **195** (1993) 319-323