Penerapan Pembangkit Bilangan Acak Semu untuk Simulasi Komputer

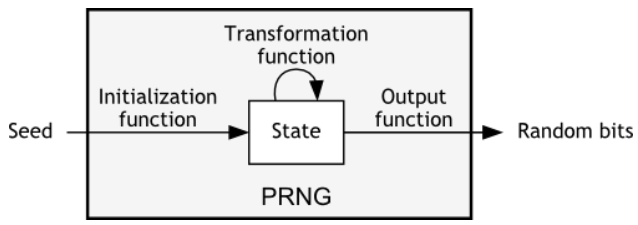
Joshua Aditya Kosasih
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514012@std.stei.itb.ac.id

***Abstract*—Pseudorandom Number Generator adalah pembangkit angka acak semu, yang memiliki banyak variasi. PRNG dipakai dalam berbagai bidang, termasuk simulasi komputer. Simulasi harus mengimitasi sistem pada kenyataan yang penuh ketidakpastian. Suatu metode terkenal bernama Monte Carlo Method umum digunakan menjadi metode simulasi. Metode tersebut memanfaatkan angka acak dalam jumlah yang banyak. Kebutuhan terhadap angka acak tersebut didapat dari sebuah PRNG yang sesuai.**

***Keywords*—Pseudorandom Number Generator (PRNG), Simulasi, Mersenne Twister (MT), Monte Carlo Method.**

I. PENDAHULUAN

Pseudorandom Number Generator (PRNG) atau dalam bahasa Indonesianya Pembangkit Bilangan Acak Semu adalah algoritma untuk menghasilkan rangkaian angka yang mendekati rangkaian angka acak (random). Rangkaian angka hasil dari PRNG tidak benar-benar acak, karena ditentukan dari kumpulan nilai awal yang umumnya disebut PRNG's seed. Walaupun rangkaian angka yang lebih acak bisa didapat dari Hardware Random Number Generators (pembangkit bilangan acak dengan cara manipulasi di tingkat perangkat keras), PRNG dibutuhkan untuk kemudahan dan kecepatannya dalam menghasilkan rangkaian angka acak.



PRNG memiliki aplikasi-aplikasi yang penting seperti dalam bidang simulasi, game, dan kriptografi. Aplikasi dalam bidang kriptografi memerlukan hasil keluaran yang tidak dapat diprediksi berdasarkan dari hasil keluaran sebelumnya, sehingga rangkaian angka acak PRNG dibutuhkan. Untuk pembahasan aplikasi-aplikasi PRNG lebih lanjut akan dibahas di bab 3.

Tingkat penilaian dari sebuah PRNG didapat dari penilaian statistik hasil rangkaian angka acak dari PRNG tersebut. Penilaian ini dibahas lebih lanjut di bab 2. Selain itu, penilaian bagus tidaknya PRNG juga didapat dari nilai perioda yang besar. Maksud dari perioda disini akan dijelaskan di bab 2. Tingkat penilaian ini dipakai untuk menentukan jenis PRNG yang sesuai untuk kebutuhan tertentu yang diinginkan.

II. TEORI

Pseudorandom Number Generator (PRNG) bukanlah sebuah algoritma tunggal yang menghasilkan rangkaian angka acak, melainkan lebih seperti sebuah sebutan klasifikasi bagi algoritma-algoritma yang memiliki jenis yang sama (membangkitkan angka acak). PRNG memiliki banyak jenis. Berikut adalah nama-nama dari jenis-jenis PRNG yang terkenal:

- Middle-square method
- Blum Blum Shub
- ISAAC
- XORShift
- Mersenne Twister (MT)
- SIMD-oriented Fast Mersenne Twister (SFMT)
- Well Equidistributed Long-period Linear (WELL)
- Linear congruential generator
- Stream ciphers

Analoginya adalah PRNG itu seperti sorting algorithm dan jenis-jenis PRNG adalah jenis-jenis sorting algorithm (misal: SFMT itu seperti quicksort, MT itu seperti mergesort, dst.).

PRNG juga memiliki sebuah subklasifikasi yaitu cryptographically secure pseudorandom number generator (CSPRNG) yang berarti PRNG yang cocok digunakan untuk kriptografi.

Tetapi, tidak seperti sorting algorithm, jenis-jenis PRNG walaupun dapat dinilai, tidak bisa dikomparasikan secara

langsung karena perbedaan kebutuhan untuk pemakaian tiap jenisnya.

Terdapat 4 buah kriteria yang ditetapkan dalam BSI evaluation criteria untuk menilai sebuah PRNG. Berikut keempat buah kriteria untuk PRNG:

1. Menghasilkan rangkaian angka acak dengan kemungkinan yang rendah untuk mempunyai kemunculan urutan yang identik.
2. Menghasilkan rangkaian angka yang menyerupai rangkaian angka yang benar-benar acak. Diuji dengan beberapa tes yang sudah umum dipakai untuk menguji sebuah PRNG. Contoh-contoh tes yang dipakai antara lain: Monobit test, Chi-squared test, runs test, autocorrelation test, dll.
3. Menghasilkan rangkaian angka yang tidak dapat dihitung, dikalkulasi, atau ditebak oleh orang luar yang sudah mendapat sebagian rangkaian, baik nilai sebelum atau sesudah dari rangkaian tersebut.
4. Menghasilkan rangkaian angka dimana angka-angka sebelumnya tidak dapat dihitung, dikalkulasi, atau ditebak oleh orang dalam yang mengetahui cara pembangkitnya

Untuk CSPRNG, hanya PRNG yang dapat memenuhi kriteria nomor 3 dan nomor 4 yang diterima.

Berikut akan dijelaskan mengenai perioda PRNG yang juga mempengaruhi penilaian dari PRNG.

Sebuah PRNG dapat dimulai dari sebuah state awal menggunakan seed state. PRNG tersebut akan selalu menghasilkan rangkaian angka acak yang sama jika dimulai dari state tersebut. Dengan pengertian tersebut, periode PRNG adalah panjang maksimum, dari semua state awal, dari prefiks rangkaian angka yang tidak berulang. Periode ini dibatasi oleh jumlah state, biasa dihitung dalam bit. Tetapi, karena panjang periode dapat dilipat gandakan dengan penambahan setiap bit state, adalah cukup mudah untuk membuat PRNG dengan periode yang cukup panjang untuk dapat digunakan.

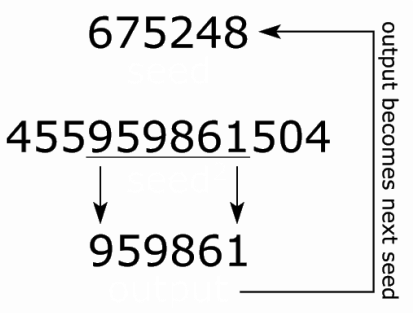
Jika sebuah PRNG memiliki state yang berjumlah n bits, periodanya tidak bisa lebih panjang dari 2^n dan malah mungkin lebih pendek. Beberapa PRNG dapat dihitung periodanya. Contohnya: Linear Feedback Shift Registers (LFSRs) memiliki perioda 2^n – 1. Linear congruential generators memiliki perioda yang dapat dihitung dengan pemfaktoran.

Jika PRNG sudah mencapai akhir perioda, rangkaian angka yang dihasilkan akan terulang kembali. Itulah mengapa namanya perioda. Tetapi, dari hasil yang terulang tidak selalu berarti PRNG tersebut sudah mencapai periodanya. Umumnya, semakin besar periodanya, semakin panjang rangkaian angka yang benar-benar acak yang dapat dihasilkan.

Kebanyakan PRNG menghasilkan rangkaian angka yang nilainya terdistribusi secara merata.

PRNG berbasis komputer yang pertama bernama Middle-square method. Algoritma tersebut diciptakan oleh John von Neumann di tahun 1946. Dalam prakteknya algoritma tersebut kurang baik, karena periodenya sangat pendek dan algoritma tersebut memiliki beberapa kelemahan yang cukup parah, seperti rangkaian angka hampir selalu menuju ke nol. Berikut gambaran mengenai cara kerja algoritmanya.

Untuk menghasilkan rangkaian angka acak 4 digit, dibuat sebuah nilai 4 digit (seed) dan dikuadratkan, menghasilkan nilai sebesar 8 digit. Jika hasilnya kurang dari 8 digit, nol ditambahkan di depan agar jumlah digitnya 8. 4 digit angka yang berada di tengah dari hasilnya akan menjadi angka berikutnya dalam urutan, dan outputkan sebagai hasilnya. Proses ini kemudian diulang untuk menghasilkan angka baru.

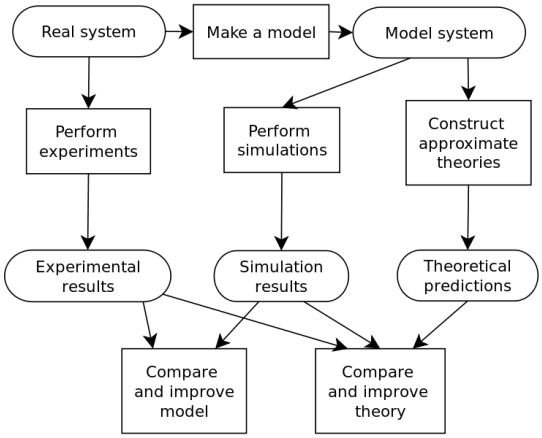


Untuk menghasilkan angka n digit, periode tidak dapat bernilai lebih dari 8n. Jika keempat digit semua nol, maka akan terus menghasilkan nol. Jika setengah digit dari seed bernilai nol, maka hasil akan terus menuju ke nol. Hal ini menunjukkan kalau membuat sebuah PRNG tidaklah mudah.

III. Implementasi PNGR dalam Simulasi

A. Simulasi Komputer

Simulasi komputer adalah upaya untuk memodelkan kehidupan nyata atau situasi hipotetis pada komputer sehingga dapat dipelajari untuk melihat bagaimana sistem tersebut bekerja. Dengan mengubah variabel dalam simulasi, prediksi dapat dibuat tentang perilaku sistem. Ini adalah alat untuk hampir menyelidiki perilaku sistem yang diteliti.



Simulasi komputer telah banyak berguna untuk memodelkan banyak sistem alam dalam fisika, kimia dan biologi, dan sistem manusia di bidang ekonomi dan ilmu sosial (misalnya, sosiologi komputasi) serta di bidang teknik untuk mendapatkan informasi tentang pengoperasian sistem-sistem. Contohnya simulasi di bidang lalu lintas jaringan. Dalam simulasi tersebut, perilaku model akan berubah pada setiap simulasinya sesuai dengan parameter awal yang ditetapkan yang diasumsikan ada di lingkungan sebenarnya.

Dulunya, pemodelan sebuah sistem dilakukan melalui pemodelan matematika. Pemodelan matematis ini digunakan untuk menemukan solusi analitis yang memungkinkan orang untuk memprediksi perilaku sistem berdasarkan beberapa parameter dan kondisi-kondisi awal. Simulasi komputer sering digunakan sebagai tambahan/substitusi untuk memodelkan sistem yang tidak dapat diambil solusi analitik tertutupnya. Ada berbagai jenis simulasi komputer. Pada umumnya simulasi komputer berusaha untuk menghasilkan sebuah sampel yang dapat merepresentasikan skenario-skenario untuk sebuah model di mana pencacahan lengkap dari semua state yang ada akan susah atau tidak mungkin.

Simulasi/pemodelan di komputer bisa diklasifikasikan menjadi beberapa pasangan sifat:

- Stochastic atau deterministic
- Steady-state atau dinamis
- Kontigu atau diskrit
- Lokal atau terdistribusi

Yang menarik dari pasangan sifat-sifat diatas adalah pasangan sifat poin pertama. Stochastic secara singkat berarti acak, berubah secara tidak teratur seiring waktu, berevolusi dengan nilai acak seiring waktu. Deterministic berarti kebalikannya, proses yang stabil, dengan input yang sama akan selalu menghasilkan output yang sama.

Simulasi komputer yang memodelkan sistem alam, manusia, dan bidang teknik umumnya memakai sifat stochastic. Hal tersebut dikarenakan dalam kehidupan nyata ini, sistem-sistem tersebut tidak selalu menghasilkan keluaran yang pasti. Terdapat terlalu banyak parameter bebas yang dapat mengubah hasil keluaran. Berbeda dengan pemodelan sistem matematika, yang pasti, yang menggunakan sifat deterministic.

B. Implementasi

Pemodelan dengan sifat stochastic memerlukan sebuah PRNG yang baik dan metode pemodelan yang memproses rangkaian angka acak tersebut. Metode yang dipakai umumnya adalah Monte Carlo method. Sedangkan PRNG yang dinilai baik atau cocok untuk masalah simulasi/pemodelan terutama menggunakan Monte Carlo method adalah Mersenne Twister (MT) dan Well Equidistributed Long-period Linear (WELL).

Mersenne Twister termasuk sebagai PRNG yang cocok untuk simulasi karena beberapa kelebihan berikut:

1. Memiliki perioda yang besar, $2^{19937} - 1$
2. Memiliki *k*-distribusi sampai akurasi 32-bit untuk setiap $1 \leq k \leq 623$
3. Sudah lolos uji beragam tes statistik, termasuk Diehard Tests (Paket tes khusus untuk menguji sebuah PRNG)

Walaupun begitu, MT juga memiliki beberapa kelemahan yaitu: distribusi yang kalah merata dibanding WELL dan kecepatannya yang cukup lambat (walaupun jauh lebih cepat dari WELL) untuk standar sekarang. Untuk mengatasi kelemahan MT tadi sudah dibuat SFMT yang lebih cepat dan memberi hasil yang lebih acak daripada leluhurnya. Tetapi yang akan dibahas adalah MT.

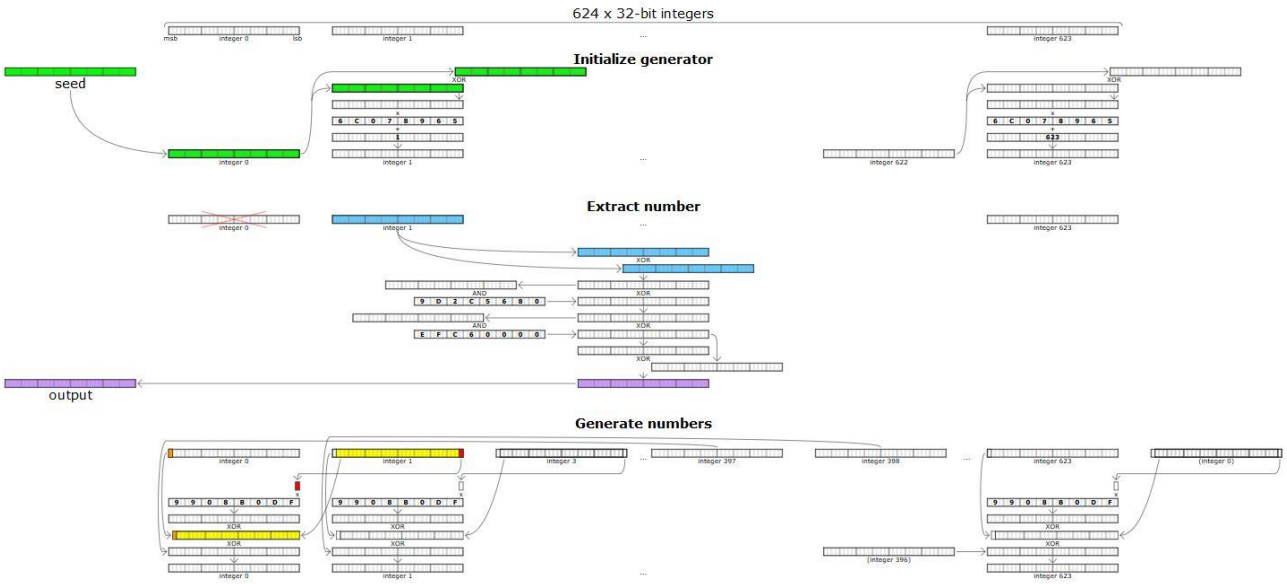
C. Mersenne Twister

Mersenne Twister adalah PRNG yang paling banyak digunakan untuk berbagai keperluan. MT didesain untuk memperbaiki kekurangan dari kebanyakan PRNG sebelumnya. MT adalah PRNG pertama yang menghasilkan rangkaian angka acak semu dengan kualitas acak yang baik dengan cepat. MT dipakai sebagai PRNG pada sistem-sistem berikut: Python, Ruby, Free Pascal, MATLAB, C++11, Apache, dll. Walaupun begitu, MT tidak cocok menjadi CSPRNG.

Berikut pseudocode untuk algoritma MT secara umum.

```
// Membuat array sepanjang n untuk
menyimpan state dari generator
 int[0..n-1] MT
 int index := n+1
 const int lower_mask = (1 << r) - 1
const int upper_mask = lowest w bits of
(not lower_mask)

// Lanjut ke halaman berikutnya
// Inisialisasi generator dengan seed
```



Visualisasi dari proses pembangkitan bilangan acak dengan MT .

```
// Inisialisasi generator dengan seed
 function seed_mt(int seed) {
     index := n
     MT[0] := seed
     for i from 1 to (n - 1) { // loop
setiap element
         MT[i] := lowest w bits of (f *
(MT[i-1] xor (MT[i-1] >> (w-2))) + i)
     }
 }

 // Mengambil nilai menengah berdasar
dari MT[index]
 // memanggil twist() setiap n buah
 function extract_number() {
     if index >= n {
         if index > n {
           error "Generator was never
seeded"
         }
         twist()
     }

     int y := MT[index]
     y := y xor ((y >> u) and d)
     y := y xor ((y << s) and b)
     y := y xor ((y << t) and c)
     y := y xor (y >> l)

     index := index + 1
     return lowest w bits of (y)
 }

// Lanjut ke kolom sebelah kanan
// Generate setiap nilai n dari series
x_i
```

```
// Generate setiap nilai n dari series
x_i
 function twist() {
     for i from 0 to (n-1) {
         int x := (MT[i] and upper_mask)
                   + (MT[(i+1) mod n] and
lower_mask)
         int xA := x >> 1
         if (x mod 2) != 0 {
xA := xA xor a
         }
         MT[i] := MT[(i + m) mod n] xor
xA
     }
     index := 0
 }
```

D. Monte Carlo Method

Metode Monte Carlo bukanlah sebuah metode tunggal yang memproses serta memberi estimasi dari rangkaian angka acak. Metode Monte Carlo adalah sebuah klasifikasi bagi algoritma komputasi yang bergantung pada pengulangan sampel data acak untuk mendapatkan hasil yang kuantitatif. Biasanya dipakai dalam masalah fisika dan matematika yang susah atau tidak mungkin bisa dipecahkan dengan cara matematis. Metode Monte Carlo dipakai dalam 3 permasalahan yang berbeda: optimisasi, integrasi numerik, dan pembangkitan seri dari distribusi probabilistik.

Metode Monte Carlo berguna untuk mensimulasikan sistem dengan banyak parameter bebas, seperti pada simulasi benda cair, struktur cell, hingga simulasi resiko

dalam bisnis. Prinsipnya, metode ini bisa digunakan untuk menyelesaikan masalah apapun yang bertemakan intrepetasi probabilistik.

Metode-metode Monte Carlo bervariasi, tetapi memiliki beberapa kesamaan:

1. Mendefinisikan domain dari input yang mungkin
2. Membangkitkan input yang acak dari distribusi probabilistik dari domain
3. Melakukan komputasi yang deterministik pada input
4. Mengaggregasi hasilnya

Untuk keperluan nomor 2 itu dipakailah PRNG. Metode Monte Carlo tidak selalu memerlukan angka yang benar-benar acak untuk dapat berfungsi. Tetapi tetap diperlukan PRNG untuk menghasilkan rangkaian angka seacak mungkin. Berikut kebutuhan rangkaian angka acak oleh Metode Monte Carlo:

1. PRNG harus memiliki karakteristik tertentu (seperti perioda yang panjang)
2. PRNG harus menghasilkan output yang telah teruji keacakannya
3. Jumlah data yang mencukupi untuk mendapatkan hasil maksimum

Penggunaan Metode Monte Carlo memerlukan angka-angka acak dalam jumlah yang besar, dan itu jugalah salah satu factor yang menyebabkan berkembangnya Pseudorandom Number Generator. PRNG jauh lebih cepat daripada penggunaan tabel berisi angka acak yang sudah digunakan dalam pengambilan sampel data secara statistik.

Bidang-bidang yang dapat memanfaatkan aplikasi metode Monte Carlo adalah:

- Fisika
- Teknik
- Biologi komputasi
- Statistik terapan
- AI untuk games
- Desain dan visual
- Search and Rescue
- Keuangan dan bisnis

Selain itu juga terdapat aplikasi yang sangat popular bagi penggunaan angka acak dalam simulasi numeric yaitu optimisasi numerik. Contoh dari persoalan optimisasi numerik adalah traveling salesman problem (TSP), tetapi alih-alih menghitung jarak terpendek, yang diinginkan adalah mendapat waktu yang tersingkat. Persoalan ini melebihi optimisasi konvensional/biasa karena waktu perjalanan tidak dapat diprediksi. Untuk itu agar dapat menyelasaikan persoalan ini diperlukan simulasi.

IV. KESIMPULAN

Pembangkit angka acak semu (PNRG) selain berguna dalam bidang kriptografi dan game juga berperan penting dalam proses simulasi. Simulasi yang dimaksud adalah simulasi stochastic yang berdasar pada sistem riil di kehidupan nyata, dimana tidak ada yang pasti. Simulasi tsb umumnya memakai Monte Carlo method yang mengestimasi hasil dari rangkaian angka acak yang didapat dari PRNG. PRNG yang umumnya dipakai untuk simulasi adalah Mersenne Twister (MT) dan Well Equidistributed Long-period Linear (WELL). MT juga merupakan PRNG yang terkenal dan banyak dipakai oleh berbagai sistem.

VII. TERIMA KASIH

Saya berterimakasih pada Tuhan Yang Maha Esa karena berkat bantuan-Nya makalah ini dapat selesai. Saya juga berterimakasih pada dosen saya, Pak Rinaldi Munir yang telah mengajari saya tentang Matematika Diskrit yang menginspirasi saya untuk mengambil topik teori bilangan sebagai topik makalah saya. Terima kasih juga untuk segenap keluarga dan teman-teman yang telah membantu menguatkan saya dalam menyelesaikan makalah ini.

REFERENSI

- NMCS4ALL: Random number generators - Youtube https://www.youtube.com/watch?v=_tN2ev3hO14 Diakses 8-12-2015
- L'Ecuyer P. (2010), "Uniform random number generators", *International Encyclopedia of Statistical Science* (editor—Lovric M.) Springer.
- *Wescott, Bob (2013)*. *The Every Computer Performance Book, Chapter 7: Modeling Computer Performance*. ISBN 1482657759.
- Matsumoto, M.; Nishimura, T. (1998). *"Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator"*. ACM Transactions on Modeling and Computer Simulation **8** (1): 3–30. doi:10.1145/272991.272995
- Best pseudo random number generator – Stack Overflow http://stackoverflow.com/questions/4720822/best-pseudo-random-number-generator Diakses 8-12-2015
- Fishman, G. S. (1995). *Monte Carlo: Concepts, Algorithms, and Applications.* New York: Springer. ISBN 0-387-94527-X

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Desember 2015

Joshua Aditya Kosasih, 13514012