8. Заико А. И. Оценивание плотности вероятности эргодического случайного процесса // Матер. II междунар. научно-практич. конф. «Современные проблемы науки и образования в техническом вузе».–Т.1.–Уфа: УГАТУ, 2015.–С.146-152.

К.Е. Климентьев

ВЫБОР И РЕАЛИЗАЦИЯ ПРОГРАММНОГО ГЕНЕРАТОРА ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ ДЛЯ СИСТЕМЫ МУЛЬТИАГЕНТНОГО МОДЕЛИРОВАНИЯ

(Самарский университет)

Введение. На кафедре ИСТ Самарского Университета продолжается разработка и реализация программной среды для мультиагентного моделирования поведения «инфицирующих» сущностей [1], примерами которых могут служить некоторые разновидности вредоносных программ (компьютерные вирусы и черви), болезнетворные микроорганизмы, участки возгорания во время пожаров и т.п. Очевидно, неотъемлемым и важным компонентом любой системы имитационного моделирования, предполагающей проведение статистических экспериментов, является «качественный» генератор псевдослучайных чисел (ГПСЧ). Более того, для получения достоверных результатов рекомендуется выполнение нескольких статистических экспериментов с одинаковыми начальными условиями, но с использованием разных ГПСЧ.

Ядром рассматриваемой системы реализовано средствами XDS-Modula-2 фирмы Excelsior (см. https://www.excelsior.ru/products/xds). Недостатком данной системы программирования является невозможность самостоятельной инициализации встроенного ГПСЧ начальным состоянием, что исключает возможность повторения экспериментов. Другим важным недостатком является отсутствие поддержки 64-битной целочисленной арифметики.

1. Постановка задачи. Таким образом, ставится задача выбора и реализации генератора (или нескольких генераторов) псевдослучайных чисел со свойствами, удовлетворяющими условиям применения.

2. Анализ условий задачи. (Псевдо-) случайными элементами имитационной модели, реализуемой в системе с помощью ГПСЧ, являются (см. [1]): топология пространств; начальные состояния и законы изменения значений атрибутов; потоки событий, описывающие поведение и взаимодействие агентов и прочее. Очевидно, (псевдо-) случайный характер несут как «непрерывные», так и «дискретные» аспекты функционирования модели.

Следовательно, ГПСЧ должен служить «хорошим» источником как вещественных чисел с разными вероятностными распределениями, так и целых чисел, представимых в виде совокупности отдельных битов. Обычно к псевдослучайным числовым последовательностям предъявляются требования: 1) «равномерности», то есть равной вероятности появления различных чисел, би-

товых фрагментов чисел и групп чисел; 2) «случайности», то есть непредсказуемости появления отдельных чисел или групп чисел.

3. Анализ предметной области. Анализ источников [2-10] позволил сделать выводы об общей структуре, характерной для большинства программных ГПСЧ (см. рис. 1).

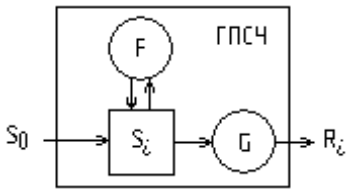

Рис. 1. Типовая структура ГПСЧ

На этом рисунке S_i – внутреннее состояние ГПСЧ на *i*-ом шаге работы; S_0 – начальное состояние ГПСЧ; F – функция перехода, описывающая переход ГПСЧ из состояния в состояние на каждом очередном шаге; G – функция генерации, описывающая формирование на каждом шаге работы очередного псевдослучайного числа R_i. Все числовые последовательности, генерируемые ГПСЧ, периодичны. На определенном шаге внутреннее состояние S вновь приобретает значение S_0, и вся последовательность циклически повторяется.

В большинстве ГПСЧ состояние S описывается либо единственным целым числом, либо массивом чисел, дополненным при необходимости набором индексов-указателей, счетчиков количества шагов и т.п. Выходное значение чаще всего представляет собой целое $R\in[0..R_{max}]$, которое для преобразования в вещественное $r\in[0..1]$ предлагается нормировать: $r=R/R_{max}$. Однако существуют ГПСЧ, внутреннее состояние которых описывается комбинацией простых и вещественных или исключительно вещественными числовыми значениями (например, генератор Витчмена-Хилла, встроенный в некоторые версии MS Excel). В этом случае актуально обратное преобразование: $R=\lfloor r\times R_{max}\rfloor$, где $\lfloor . \rfloor$ означает округление «вниз» до ближайшего целого.

4. Краткий обзор алгоритмов. В рассмотрение попали алгоритмы с 32-битным выходом, использующие 32-битную целочисленную арифметику.

1. LCG - смешанные линейные конгруэнтные ГПСЧ (или «генераторы Лемера»), использующие алгоритм $R_i=(R_{i-1}\times a+c) \bmod m$, где a, c и m – некоторые константы, правила выбора которых описаны в [2]. Обычно $m=2^L$ определяется разрядностью L машинного слова. При правильном выборе констант a и c период так же равен m. Достоинства: простота, быстродействие, наличие развитой теории. Недостатки: 1) неслучайность младших битов; 2) линейная зависимость между любыми двумя элементами последовательности. Для нейтрализации первого недостатка используется отбрасывание до 16-20 младших битов результата. В итоге приходится мириться с «коротким» диапазоном возможных значений.

2. MLCG - мультипликативные линейные конгруэнтные ГПСЧ (или «генераторы Парка-Миллера»), использующие алгоритм $R_i=(R_{i-1}\times a) \bmod m$, где a и m – некоторые константы, правила выбора которых описаны в [2]. Обычно в

качестве m выступает большое простое число, например $m=2^{31}-1 = 2147483647$, в этом случае период достигает m–1. Достоинства: простота, быстродействие, наличие развитой теории. Недостатки: 1) относительно небольшой период; 2) линейная зависимость между любыми двумя элементами последовательности. Для нейтрализации первого из недостатков рекомендуется комбинировать вычитанием $R_i=(P_i-Q_i) \bmod \max(m1,m2)$ результаты работы двух параллельно работающих ГПСЧ с периодами m1 и m2, в этом случае период достигает (m1–1)×(m2–1).

3. Линейные конгруэнтные ГПСЧ с перемешиванием по методу Бейса-Дурхема (Bays–Durham). Числовые значения, сгенерированные ДПСЧ, накапливаются в постоянно обновляемом буфере размером K элементов, из которого потом извлекаются в случайном порядке [2]. Период имеет порядок $(m!K!)^{1/2}$. Достоинства: 1) в выходной числовой последовательности нейтрализуются корреляционные зависимости; 2) при увеличении K возможно достижение очень больших периодов. Теоретически обоснованные недостатки не выявлены.

4. Комбинированные смешанные линейные конгруэнтные ГПСЧ. Варианты комбинирования двух генераторов с разными константами (a1, c1) и (a2, c2) и общим периодом m [2-10]:

1) вычитание: если $P_i>Q_i$ то $R_i=P_i-Q_i$, иначе $R_i=(P_i+m)-Q_i$.

2) сложение: $R_i=(P_i+Q_i) \bmod m$;

3) сложение по модулю 2: $R_i=P_i \oplus Q_i$;

4) конкатенация «старших» половинок: $R_i=\lfloor P_i/m^{1/2} \rfloor+\lfloor Q_i/m^{1/2} \rfloor \times m^{1/2}$;

5) конкатенация комбинированных «старших» и «младших» половинок (алгоритм, известный как MWC1616 Дж. Марсальи).

Для увеличения периода комбинированных ГПСЧ в настоящей работе предлагается формальная новинка (не описанная, по крайней мере, в доступных источниках), а именно: вместо $R_i=P_i \bullet Q_i$ использовать $R_i=P_i \bullet Q_{i+j}$, где • – один из описанных выше методов комбинирования; *i* – номер элемента в числовой последовательности; j – номер повторения «периодичной» числовой последовательности. Благодаря этому период достигнет m×m.

5. LFG - генераторы Фибоначчи с запаздыванием [2], использующие алгоритмы $R_i=(R_{i-p}+R_{i-q}) \bmod m$ или $R_i=(R_{i-p}-R_{i-q}) \bmod m$, где p и q – некоторые числовые запаздывания («лаги»); $m=2^E$ – ограничение, обусловленное конечной длиной разрядной сетки. Если (для определенности) p<q, то период составит $2^{E-1} \times (2^q-1)$. Достоинства: большая скорость работы; высокие «случайность» и «равномерность» результирующей числовой последовательности. Недостатки: 1) повышенное количество «1» в двоичном представлении чисел выходной последовательности; 2) «сгущения» близких по величине чисел выходной последовательности. Для нейтрализации недостатков используют прореживание («децимация») выходной последовательности, например, согласно [2], отбрасывают каждое второе число. Более радикальный, теоретически обоснованный способ предполагает неравномерное прореживание, а именно: если (для определенности) p<q, то использовать q чисел, потом отбросить 10×q чисел, потом опять использовать q чисел и т.д.

6. Генераторы семейства XORSHIFT, предложенные Дж.Марсальей и использующие для генерации псевдослучайных чисел побитовые логические операции «сложение по модулю 2» (XOR) и «сдвиг вправо/влево» (SHR/SHL). Эти ГПСЧ представляют собой попытку получить последовательность псевдослучайных чисел из последовательности «хорошо перемешанных» битов (теоретическое обоснование такого подхода см., например, в [11]). Достоинства: 1) простота; 2) сверхвысокая скорость работы. Недостатки: 1) «сгущения» битов в двоичном представлении чисел; 2) недостаточная равномерность распределения чисел в относительно «коротких» числовых последовательностях. В рассмотрение не попали разновидности XORSHIFT64, XORSHIFT128, XORSHIFT128+, HORSHIFT128*, WOWSHIFT, XOROSHIRO и др., а так же комбинированные генераторы типа KISS, требующие для реализации наличия «длинной арифметики».

7. Генераторы типа «вихрь Мерсенна», предложенные М.Мацумото и Т.Нисимурой (см. [3], [10]), на текущий момент считаются самыми лучшими с точки зрения статистического качества генерируемой числовой последовательности и с точки зрения длины периода ($2^{19937}-1 \approx 4.3 \times 10^{6001}$). Однако отличаются сложным, неочевидным алгоритмом работы и длинным, медленным при выполнении программным кодом. Причем это относится не только к алгоритму генерации псевдослучайных чисел, но и к алгоритму начальной инициализации внутреннего состоянии ГПСЧ.

5. Тестирование ГПСЧ. Все описанные выше генераторы были реализованы в среде XDS Modula-2 и подвергнуты тестированию [12]. В качестве средств тестирования были рассмотрены.

1. Ent (http://www.fourmilab.ch/random/) - программная утилита Дж.Уолкера, которая предназначена для проведения небольшого количества базовых тестов на «случайность» и «равномерность», применимых к последовательности отдельных битов и 8-битовых байтов, считываемых из файла. Утилита позволяет сделать лишь самые общие выводы о возможностях ГПСЧ, сгенерировавшего исследуемую последовательность.

2. Dieharder (https://webhome.phy.duke.edu/~rgb/General/dieharder.php) - программный комплекс Р.Брауна, служащий развитием более ранней разработки Diehard от Дж. Марсальи. Содержит комплекс очень строгих статистических тестов, предназначенных для поиска мелких, необнаружимых другими средствами отклонений от «случайности» и «равномерности» в числовых последовательностях, производимых ГПСЧ.

3. NIST SP 800-22 [13] – программный комплекс, разработанный и поддерживаемый Национальным Агентством по Стандартизации США. Содержит сравнительно небольшой, тщательно отобранный комплект статистических тестов, предназначенных для исследования «случайности» и «равномерности» битовых последовательностей, производимых «криптографически стойкими» ГПСЧ.

4. TestU01 [14] – большой программный комплекс, разработанный в Монреальском университете и содержащий несколько тестовых комплектов,

каждый из которых включает десятки самых разнообразных статистических тестов. Комплект «Alphabit» предназначен для тестирования битовых последовательностей. Комплект «Rabbit» предназначен для тестирования последовательностей битов и групп битов (то есть целых чисел). Комплекты «SmallCrush», «Crush» и «BigCrush» характеризуются возрастающей строгостью тестов для последовательностей вещественных чисел, равномерно распределенных на интервале [0..1]. Если «SmallCrush» позволяет довольно грубо «ранжировать» ГПСЧ, разделив их на «посредственные», «хорошие» и «отличные», то «BigCrush» (содержащий тесты из пакетов Diehard и Dieharder) пытается найти мелкие недостатки в «отличных».

Для тестирования ГПСЧ были использованы: 1) утилита Ent для определения потенциальной возможности использования реализованных (возможно, с ошибками) программ в качестве ГПСЧ; 2) комплект «Rabbit» из комплекса TestU01 для тестирования применимости ГПСЧ в задачах моделирования дискретных объектов и процессов (случайных расстановок, адресных пространств, графовых топологий и т.п.); 3) комплект «SmallCrush» из комплекса TestU01 для тестирования применимости ГПСЧ в задачах моделирования непрерывных объектов и процессов (вероятностных распределений, случайных процессов, потоков событий и т.п.).

В результате проведения тестов средствами утилиты Ent все реализации ГПСЧ подтвердили способность генерировать числовые последовательности, обладающие свойствами «случайности» и «равномерности».

Результаты тестирования средствами «Rabbit» (26 тестов) и «SmallCrush» (10 тестов) представлены в табл. 1. Следует иметь в виду, что тесты типа «RandomWalks» применяются группами, но засчитываются как единый тест. Поэтому, например, обозначение «6=5+3» означает, что всего не пройдено 5 «обычных» тестов и 3 теста типа «RandomWalks», но общее количество «проваленных» тестов составляет 6 из 10 возможных.

Табл.1 – Результаты тестирования ГПСЧ

Генератор	SmallCrush	Rabbit	Числовые параметры ГПСЧ
LCG	10=9+3	19=17+4+2	(1103515245, 12345)
LCG>>16	8=7+5	23	- «» -
LCG BD-shuffled	5	8	(1103515245, 12345) K=37
LCG1|LCG2	5	6	(1103515245, 12345) (69069, 1234567)
LCG1–LCG2	8=7+5	17=16+14	- «» -
LCG1+LCG2	9=8+5	17=16+14	- «» -
LCG1⊕LCG2	9=8+5	17=16+14	- «» -
LCG1|LCG2 BD-shuffled	**0**	**6**	- «» -
MLCG	5	21=18+5+5+5	16807, $m=2^{31}-1$
LFG	2	6	(55, 24)
LFG Decimated	**0**	**7+1**	(55, 24) 1/2
XORSHIFT32	6=5+1	9+2	(13, 17, 5)
Mersenne Twister (MT)	0	8+1	
MWC1616	3	9+2	(18000, 30903)

Выводы. Тесты продемонстрировали не только ожидаемо посредственные эксплуатационные характеристики линейных конгруэнтных генераторов и их комбинаций, но и неожиданно удачные результаты применения к ним же некоторых простейших модификаций.

По итогам тестирования выбраны к использованию: 1) генератор Фибоначчи с прореживанием; 2) генератор, использующий конкатенацию старших 16-битовых «половинок», полученных от двух разных линейных конгруэнтных генераторов (со смещением на один шаг после исчерпания цикла – см. выше), с последующим перемешиванием выходной последовательности по методу Бейса и Дурхема.

Литература

1. Климентьев К.Е. Мультиагентное моделирование процессов распространения и взаимодействия инфицирующих сущностей // Программные продукты и системы. - Тверь, 2018. - 1(31) - с. 744-748.

2. Кнут Д. Искусство программирования, том 2. Получисленные алгоритмы, 3 изд. - М.: Издательский дом «Вильямс», 2001. - 832 с.

3. Kneusel R. Random Numbers and Computers. - Springer, 2018. - 259 pp.

4. Press E. et al. Numerical Recipes. Third Edition. - Cambridge University Press, 2007. - 1235 pp.

5. Gentle J. Random Number Generation and Monte-Carlo Methods. - George Mason University, 2002. - 381 pp.

6. Beebe N. The Matematical Function Computation Handbook. - Springer, 2017. - 1115 pp.

7. Newman M.E.J., Barkema G.T. Monte Carlo Methods in Statistical Physics. - Oxword University Press, 2001. - 475 pp.

8. Лоу А., Кельтон Д. Имитационное моделирование. - СПб.: БХВ-Пресс, 2004. - 847 с.

9. ГОСТ Р ИСО 24153-2012. Статистические методы. Процедуры рандомизации и отбора случайной выборки. - М.: Стандартинформ, 2014. - 30 с.

10. ГОСТ Р ИСО 28640-2012. Статистические методы. Генерация случайных чисел. - М.: Стандартинформ, 2014. - 52 с.

11. Михайлов Г.А., Войтишек А.В. Численное статистическое моделирование. Методы Монте-Карло. - М.: Издательский центр «Академия», 2006. - 367 с.

12. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. - М.: Кудиц-образ, 2003. - 240 с.

13. Rukhin A. et al. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. U.S. Department of Commerce, National Institute of Standards and Technology, 2010. - URL: http://ws680.nist.gov/publication/get_pdf.cfm?pub_id=906762 (дата обращения: 12.05.2019).

14, L'Ecuyer P., Simard R. TestU01. A Software Library in ANSI C for Empirical Testing of Random Numb er Generators. User's guide, compact version, 2013. - 214 pp. URL: http://www.iro.umontreal.ca/~simardr/ testu01/guideshorttestu01.pdf (дата обращения: 12.05.2019).

И.В. Кузьмина, В.В. Котлякова

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТОВ ВИРТУАЛИЗАЦИИ

(Нижегородский государственный университет им. Н.И. Лобачевского)

Сложность распределенных информационных систем, в том числе и применяемых в научных исследованиях, обусловлена многими факторами, такими как необходимость поддержки протоколов взаимодействия между компонентами этой системы, периодическая недоступность сервисов, сложность управления распределенными транзакциями. При разработке информационных систем важной задачей также является обеспечение информационной безопасности, реализующей доступность, целостность и конфиденциальность данных. В связи с этим в последние годы растёт потребность в защищённых решениях для структур, работающих с конфиденциальной информацией. Одним из них является Astra Linux Special Edition – операционная система (ОС) специального назначения на базе Linux-ядра, созданная для нужд организаций, которые работают с информацией ограниченного доступа.

Кроме того при тестировании сложных информационных систем часто возникают следующие проблемы:

- необходимость испытаний программного обеспечения (ПО) в различных пользовательских конфигурациях, количество которых превышает количество физических компьютеров, выделенных для тестирования;
- большие временные затраты на развертывание и настройку тестовых стендов, содержащих множество различных компонентов, между которыми обеспечивается сетевое взаимодействие;
- большие временные затраты на создание резервных копий систем и их конфигураций, а также восстановление состояния этих систем после запуска тестов;
- невозможность воспроизведения дефекта, найденного специалистом по тестированию, на машине разработчика, потеря времени на его поиск и исправление;
- необходимость в испытаниях программы в условиях аппаратной среды, которой нет в распоряжении команды тестирования;
- необходимость тестирования программного продукта в условиях, требующих быстрого переключения между пользовательскими конфигурациями.