ИНФОРМАЦИОННЫЕ СИСТЕМЫ.

СРЕДСТВА, ТЕХНОЛОГИИ, БЕЗОПАСНОСТЬ

ПРИМЕНЕНИЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ ОПРЕДЕЛЕНИЯ ЭФФЕКТИВНОСТИ СИСТЕМ ПАРОЛЬНОЙ ЗАЩИТЫ

А.Ю.Горбунов

Нижегородский госуниверситет

Актуальность задачи определения эффективности парольной защиты (ПЗ) обусловлена широким применением механизма ПЗ в современных операционных системах. Целью определения эффективности ПЗ является определение времени, необходимого на взлом ПЗ. Задача определения эффективности ПЗ сводится к задаче поисковой оптимизации на множестве, на котором задана целевая функция (ЦФ) в виде δ-функции.

Использование в качестве пароля квазислучайных последовательностей символов большой длины делает неэффективными используемые на настоящий момент методы решения поставленной задачи.

Поскольку генетические алгоритмы (ГА) широко используются для решения задач поисковой оптимизации [1], возникает идея исследовать ГА на возможность использования для решения задачи определения эффективности ПЗ.

В основе работы любого ГА лежит моделирование некоторых процессов, происходящих в популяциях особей в природе, существенных для развития особей. Каждая особь представлена генотипом – бинарной строкой, кодирующей точку в дискретном пространстве поиска. Особь оценивается мерой ее приспособленности согласно значению ЦФ в точке, значению которой соответствует генотип особи. Особи в популяции подвергаются естественному отбору (когда выживают только наиболее приспособленные), кроссоверу (скрещиванию, в результате которого генотип потомков повторяет участки генотипов каждого из предков) и мутациям (когда один или несколько генов в генотипе изменяются).

В процессе работы ГА неявно обрабатывает шимы (schema), которые представляют шаблоны подобия между строками [2,3]. Шима – строка длины, равной длине генотипа особей в популяции, состоящая из символов алфавита {0, 1, *}, где * – неопределенный символ. ГА практически не может заниматься полным перебором всех точек в пространстве поиска, однако он может производить выборку шим, соответствующих множеству похожих строк с высокой приспособленностью.

Эффективность работы ГА определяется теоремой шим, характеризующей скорость локализации областей с высокой приспособленностью. Теорема шим [3] показывает, что число особей с приспособленностью выше средней растет экспонен-

циально, а число особей с приспособленностью ниже средней экспоненциально уменьшается.

При адаптации ГА к решению задачи определения эффективности ПЗ операции кроссовера и мутации обеспечивают генерацию новых решений. Естественный отбор снижает эффективность ГА, поскольку подразумевает задание ЦФ в пространстве поиска и определение значения ЦФ для точек, соответствующих каждой особи в популяции на каждой итерации ГА. Из-за использования в качестве паролей квазислучайных последовательностей и отсутствия свойства глобальности у ГА, построение ЦФ на основе априорной информации о пароле является неэффективным. Большое количество вычислений ЦФ также снижает скорость поиска. В связи с этим целесообразно заменить естественный отбор операцией сравнения случайным образом выбранной особи из популяции на соответствие искомой точке в пространстве поиска с последующей буферизацией. Временная буферизация необходима для исключения зацикливания поиска и исключения из популяции особей, генотип которых совпадает с буферизованными значениями.

При мутации целесообразно вычислять значение вероятности мутации генов на основе статистического анализа буферизованных генотипов.

Для модернизированного таким образом ГА теорема шим будет иметь следующий вид:

$$m(H,t+1) \geq m(H,t)*(1-P_e)^{O(H)}*(1-P_c\frac{\delta(H)}{L-1}-O(H)*P_m), \qquad (1)$$

где $m(H,t)$ – число примеров шимы H на итерации t ГА, P_c – вероятность одноточечного кроссовера, P_m – вероятность мутации, P_e – вероятность совпадения битов в H и буферизованной шиме, L – длина генотипа, $O(H)$ – порядок H (число определенных битов), $\delta(H)$ – определенная длина H (расстояние между крайними определенными битами).

Из (1) видно, что для адаптированного ГА скорость локализации сохраняет экспоненциальный характер.

Таким образом, ГА можно адаптировать для решения задачи определения эффективности ПЗ с сохранением экспоненциального закона роста числа шим, генотип которых близок к генотипу искомой точки в пространстве поиска.

[1] Батищев Д.И., Исаев С.А., Ремер Е.К. //Межвузовский сборник научных трудов “Оптимизация и моделирование в автоматизированных системах”. –Воронеж: ВГТУ, 1998, с.20.
[2] Goldberg D.E. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, 1989.
[3] Holland J.H. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, 2nd edition, 1992.

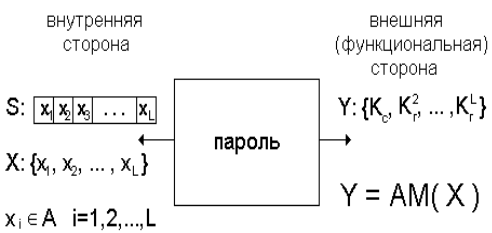
ИСПОЛЬЗОВАНИЕ КЛЕТОЧНЫХ АВТОМАТОВ ДЛЯ ОПРЕДЕЛЕНИЯ ХАРАКТЕРИСТИК ПАРОЛЬНОЙ ЗАЩИТЫ

А.С.Бажухин, А.Ю.Горбунов

Нижегородский госуниверситет

Защита информационных и системных ресурсов вычислительных сетей или отдельных вычислительных средств является одной их актуальнейших задач сегодняшнего дня. Для защиты широко используются парольные системы. Анализ проблемы защиты информации с помощью парольных систем позволяет выявить недостатки и уязвимость последних. Существующие методы, как правило, обладают большим временем поиска (полный перебор) или низкой вероятностью нахождения искомого пароля (случайный перебор). В связи с вышеуказанными аспектами проводилась разработка метода оценки характеристик парольной защиты с целью уменьшения времени поиска и увеличения вероятности нахождения пароля.

Модель парольной защиты, исследуемая в работе, представлена на рисунке. С внутренней стороны она представлена структурой и вектором параметров (S, $\mathbf{X}$), а с внешней стороны – набором частных критериев ($\mathbf{Y}$). Частные критерии K формируются на основании априорных вероятностей символов алфавита, полученных путем анализа больших объёмов (порядка 200 000 слов) текста (словарей). Из частных критериев формируется целевая функция, причем она может быть как аддитивной, так и мультипликативной. Весовые коэффициенты подбираются из соображений большей важности блоков из большего числа символов алфавита. В дальнейшем решается задача целочисленного нелинейного программирования (ЦНП) для отыскания наиболее вероятного (априорно) пароля.



Ввиду того, что парольная система – это система с минимумом информации о ней, модель достаточно проста и не может учитывать всех факторов, поэтому часто глобальный экстремум не совпадает с искомым паролем. После проведения экспериментов с моделью было выявлено, что искомый пароль находится в каком-то локальном экстремуме. После многочисленных экспериментов выяснилось, что важно отыскать не глобальный экстремум, а пробежаться по как можно большему числу локальных. Для решения этой задачи был предложен метод с использованием клеточных автоматов (ранее применялся метод ЦНП, разработанный Бугровым В.Н.)

Сущность метода заключается в следующем. Пространство поиска представляется дискретной сеткой (естественно вытекающей из дискретности задачи), ячейка (клетка) которой представляет собой автомат и вычисляет своё следующее состоя-

ние на основе состояния соседей [1,2]. Внутренне состояние автомата определяется двумя параметрами (включен автомат или нет) и координатой этого автомата.

В связи с тем, что рассмотрение всех автоматов в пространстве поиска нецелесообразно ввиду слишком большого времени анализа, в предложенном методе используются только окрестности автомата.

Рассмотрим метод подробно. В пространстве поиска случайно выбирается начальная точка. В дальнейшем будем использовать не только точку, но и её окрестность. В n мерном пространстве у нее будет $2n$ соседей. Из всех соседей выбираем наихудшего (обладающего самым низким значением целевой функции) и убираем его из дальнейшего рассмотрения. Из оставшихся $2n$ автоматов собираем n пар, которые в результате взаимодействия дают нам новые $2n$ решения. Получившиеся решения (координаты) проверяются на значение целевой функции, и те из них, у кого оно хуже, чем у предыдущей точки, отсекаются. По оставшимся решениям и будут включаться новые автоматы в пространстве поиска. Таким образом, на каждом шаге используется все большее число новых автоматов.

Взаимодействие и генерация новых решений основаны на операциях кроссовера и мутации. Пара автоматов генерирует новую пару решений на следующий шаг посредством кроссовера, далее, происходит мутация. В данном методе был предложен одно- и двухточечный кроссовер, что обеспечивает поиск локальных экстремумов квазиравномерно во всем пространстве. После операции кроссовера происходит операция мутации. Операция мутации изменяет случайно одну из координат точки поиска, соответственно сдвигая её [3].

Данный метод был реализован программно для пространства поиска размерностью 6. Кроссовер обеспечивался по точке 3 (одноточечный), 2 и 4 (двухточечный). Исходя из тестирования, можно сказать, что для данной модели парольной защиты метод работает лучше, чем метод поиска глобального экстремума, а именно, несмотря на несколько возросшее время работы (в связи с большим количеством расчетов), возросла вероятность нахождения пароля (это связано с тем, что количество пробегаемых точек, подозрительных на искомый пароль, возросло). Данный метод позволяет распараллелить вычисления и существенно увеличить скорость поиска.

[1] Фон Нейман Дж. Теория самовоспроизводящихся автоматов. –М.: Мир, 1971. с.49.
[2] Тоффоли Т., Марголус Н. Машина клеточных автоматов. –М.: Мир, 1991. 280с.
[3] Андреев В.В. Генетические и нейронные алгоритмы: Конспект лекций. – Чебоксары: Изд-во Чуваш. ун-та, 2001. 38с.

ДИНАМИКА АППАРАТА ГЕНЕТИЧЕСКОГО КОДИРОВАНИЯ И ЭВОЛЮЦИЯ РАЗНООБРАЗИЯ

К.Г.Кирьянов[1)], Н.А.Добротина[1)], О.Л.Лебедев[2)], И.В.Семенчуков[1)]

[1)]Нижегородский госуниверситет, [2)]НИОПИК, Москва

Эволюции разнообразных живых существ и их важнейших «компонентов» – нуклеотидов ДНК, мРНК, тРНК и т.д., аминокислотных (АК) остатков в белковых цепочках и др., белковых и смешанных образованиях и комплексах посвящено огромное число публикаций [1-2]. В настоящей работе *продолжено* рассмотрение начатого нами модельного подхода к эволюции на основе рассмотренных ранее [3-7] и введённых вновь 1)*алфавитов* и 2)*состава* компонент состояния аппарата универсального генетического кодирования (АУГК), *уравнений* динамики и ограничений на АУГК, а также возможных «мелких и крупных» изменений АУГК под влиянием *внешних воздействий* [7]. Намеченный подход к работам по эволюции разнообразия позволяет относиться к эволюции не только как к эволюции одного типа биокомпонентов, а как к эволюции всего множества важнейших компонентов, «замешанных» в ферментативных реакциях. Процессы трансляции и её остановки являются ключевыми не только для зарождения, продолжения и остановки жизни АУГК, но и всего организма в целом [7,8]. Этот подход, по нашему мнению, дополняет в части эволюции АУГК наметившееся в последнее время совместное рассмотрение эволюции нуклеотидных и белковых образований, составляющее суть нового научного направления – «протеомики» (см. например, обзоры акад. Киселёва Л.Л. по проблеме «Генома человека»). Аминокислоты (АК), белки, ДНК, мРНК являются важнейшим классом органических соединений в АУГК, и, в первую очередь, интересна их эволюция.

Считается, что синтез белка в клетке протекает в результате процесса трансляции согласно таблице АУГК, а структура белковой цепочки определяется структурой матричной РНК (мРНК): каждой аминокислоте в белке соответствует в мРНК тройка нуклеотидов – кодон. Двадцать образующих белки аминокислот кодируются в мРНК 64-мя кодонами, то есть генетический код вырожденный. Возникают вопросы. Какой была таблица АУГК у первых организмов на Земле, и каким путем шла её эволюция? Или же таблица «с самого начала» была такой же, как сейчас? Есть ли сейчас организмы с разными АУГК? Для ответа

Таблица АУГК.

i	Классы эквивалентности кодонов в строках таблицы УБК						α_i	АК
1	UUU	UUC					2	Phe F
2	UUA	UUG	CUU	CUC	CUA	CUG	6	Leu L
3	UCU	UCC	UCA	UCG	AGU	AGC	6	Ser S
4	UAU	UAC					2	Tyr Y
5	UGU	UGC					2	Cys C
6	UGG						1	Trp W
7	CCU	CCC	CCA	CCG			4	Pro P
8	CAU	CAC					2	His H
9	CAA	CAG					2	Gln Q
10	CGU	CGC	CGA	CGG	AGA	AGG	6	Arg R
11	AUU	AUC	AUA				3	Ile I
12	AUG						1	Met M
13	ACU	ACC	ACA	ACG			4	Thr T
14	AAU	AAC					2	Asn N
15	AAA	AAG					2	Lys K
16	GUU	GUC	GUA	GUG			4	Val V
17	GCU	GCC	GCA	GCG			4	Ala A
18	GAU	GAC					2	Asp D
19	GAA	GAG					2	Glu E
20	GGU	GGC	GGA	GGG			4	Gly G
21	UAA	UAG	UGA				3	Stop

на такие вопросы предложен метод анализа нуклеотидных последовательностей с помощью не статических, а динамических спектров чисел вырожденности $\beta(t) = \{\beta_1(t),\beta_2(t),\ldots,\beta_{S(t)}(t)\}$, намечающий подход к определению «возраста» нуклеотидной последовательности. В описываемой далее ММ дискретной динамической системы (ДДС) состояние АУГК характеризуется, в отличие от статической модели АУГК [4-6], уже зависимым от дискретного времени t=0,1,2,… вектором $C(t) = \{\beta(t), S(t), P(t)\}$, который вычисляется на основе C(0) и *уравнений* динамики градиентным методом по изменению полной динамической энтропии спектра чисел вырожденности, например, в форме $H(t) = 2\cdot[P(t)\cdot H_{чет}(t) + Q(t)\cdot H_{нечет}(t)]$, где $P(t) + Q(t)$=1, $Q(t)\geq 0$, $P(t)\geq 0$,– коэффициенты асимметрии критерия полной динамической энтропии. Эта ММ указывает на «стохастическую синхронизацию» компонентов состояния АУГК (чисел АК, кодонов и т.д.) на начальных и последующих этапах эволюции и даёт ответы на многие из поставленных выше вопросов [7]. Например, известно, что у простейших организмов найдены «диалекты» (малые отличия) таблицы АУГК [8]. Кроме того, найдены сложные органические соединения АУГК в продуктах вулканических извержений [9]. Известно также, что паутина состоит из небольшого количества аминокислот (преобладают аланин (25%) и глицин (около 40%) [10]. Разнообразие других фактов и соображений, как и перечисленные, удаётся объяснить качественно и количественно разнообразием возможных начальных условий, параметров, типов движений, внешних воздействий при моделировании упомянутых выше уравнений динамики.

Работа выполнена при частичной поддержке РФФИ (грант 02-02-175173).

[1] Шредингер Э. Что такое жизнь с точки зрения физики?
[2] Ичас М. Биологический код. – М.: Мир, 1971.
[3] Кирьянов К.Г., Лебедев О.Л. Почему биологические алфавиты имеют 4 и 20 букв? //Биофизика. 1995. Т.40, вып.3. С.536.
[4] Кирьянов К.Г., Лебедев О.Л. Об оптимальности чисел вырожденности универсального биологического кода. //Вестник ВВО Академии технологических наук РФ, 1996, с.117.
[5] Добротина Н.А., Кирьянов К.Г. //В кн.: Тр. 3-й научн. конф. по радиофизике 7 мая 1999 г. /Ред. А.В.Якимов. –Н. Новгород: ННГУ, 1999, с.122.
[6] Добротина Н.А., Кирьянов К.Г. К созданию концепции и определения специфики жизни. // Сб. Жизнь в аспекте экологии и биологии (материалы и тезисы международного научного совещания “Жизнь и факторы биогенеза”, Сентябрь, 1999г. – Ижевск: Удмурдский государственный университет, 1999, с.17.
[7] Кирьянов К.Г. Генетический код и тексты: динамические и информационные модели. –Н.Новгород: Изд-во ТАЛАМ, 2002, 100с.
[8] Инге-Вечтомов С.Г. Трансляция как способ существования живых систем, или в чем смысл «бессмысленных» кодонов. //Соросовский образовательный журнал 12/96. С.2.
[9] Мархинин Е.К. Вулканы и жизнь. – М.: Мысль, 1980. 196с.
[10] Chemical and Engineering News. 1997. №40. P.25.

МАТЕМАТИЧЕСКАЯ МОДЕЛЬ КРИПТОСИСТЕМ С ЗАКРЫТЫМ КЛЮЧОМ

А.А.Горбунов, К.Г.Кирьянов

Нижегородский госуниверситет

Установлено, что для достаточно широкого круга рассмотренных нами классических криптосистем (КС) с закрытых ключом [1, 2] удается получить унифицированный вид математических моделей (ММ), единые правила описания различных элементов внутри самой КС. Блоки шифратора и дешифратора КС представляются в виде q-уровневого линейного цифрового автомата (ЛЦА), ММ которого описывается на языке ABCD-формализма. Выявлена возможность по имеющейся модели шифратора получения модели дешифратора и всей КС в целом.

Система уравнений, описывающая модель ЛЦА, записывается в виде:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) + Du(t) \\ x(0) = x_0 \end{cases} \qquad (1)$$

В случае шифратора КС под входным сигналом $u(t)$ подразумевается открытый текст, а под выходным сигналом $y(t)$ – шифрограмма. Основная идея восстановления входных сигналов по сигналам на выходе автомата заключена в следующих выражениях:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ Du(t) = y(t) - Cx(t) \\ x(0) = x_0 \end{cases}, \text{ если } \exists\ \mathrm{D}^{-1}: \begin{cases} x(t+1) = Ax(t) + Bu(t) \\ u(t) = -D^{-1}Cx(t) + D^{-1}y(t) \\ x(0) = x_0 \end{cases} \qquad (2)$$

Подставляя в первое уравнение системы (2) вместо $u(t)$ его значение, задаваемое вторым из уравнений (2), имеем:

$$\begin{cases} x(t+1) = [A - BD^{-1}C]x(t) + [BD^{-1}]y(t) \\ u(t) = [-D^{-1}C]x(t) + [D^{-1}]y(t) \\ x(0) = x_0 \end{cases} \qquad (3)$$

Таким образом, можно построить *восстанавливающий автомат* (*):

$$\begin{cases} x^*(t+1) = A^*x^*(t) + B^*u^*(t) \\ y^*(t) = C^*x^*(t) + D^*u^*(t) \\ x^*(0) = x_0{}^* = x_0 \end{cases}, \qquad (4)$$

где: $A^*=[A\text{-}BD^{-1}C]$, $B^*=[BD^{-1}]$, $C^*=[-D^{-1}C]$, $D^*=[D^{-1}]$, $x_0{}^*=x_0$ (5)

На вход этого автомата будет поступать $u^*(t)=y(t)$, на выходе же будет $y^*(t)=u(t)$ – восстановленный сигнал. Таким образом, (1) представляет собой модель линейного автомата, осуществляющего преобразование входного сигнала $u(t)$ в шифр-сигнал $y(t)$; (3) и (4) – модель автомата, производящего обратную операцию – пре-

образование зашифрованного сигнала $y(t)=u^*(t)$ в исходный. Используя равенства (5), можно перейти к уравнениям, однозначно описывающим конкретную КС (т.е. модели шифратора и дешифратора) в ABCD-формализме через пять матричных параметров: A, B, C, D и начальных условий x_0, описывающих модель шифратора.

Данный подход был реализован программно. В качестве тестовых примеров рассматривались: шифр Цезаря, шифр Вижинера, шифрование методом гаммирования на основе псевдослучайной последовательности, вырабатываемой линейным конгруэнтным генератором [1], ММ которых удалось построить в алгебраической структуре (АС) GF(q) – поля Галуа (или соответствующего кольца – GR(q)) по модулю q. Пример шифра Петра I демонстрирует возможность сведения модели к представленному описанию подбором соответствующей АС, отличной от GF(q). Результаты обработки ряда тестовых последовательностей, символы которых задаются своими номерами в алфавите размерности q, приведены в следующей таблице. Следует также отметить, что длина последовательностей, обрабатываемых ЛЦА (1) и (3), (4), может быть произвольной.

Метод шифрования	Входная последовательность – исходный текст – $u(t)$	Выходная последовательность – шифротекст $y(t)=u^*(t)$	Восстановленная последовательность – $y^*(t)$ при $x_0=x_0^*$	q
Шифр Цезаря	1 2 3 4 5 6 7 8 9 0	4 5 6 7 8 9 0 1 2 3	1 2 3 4 5 6 7 8 9 0	10
	3 0 9 4 2 5 9 1 2 0 4 3 8 6 1 7	6 3 2 7 5 8 2 4 5 3 7 6 1 9 4 0	3 0 9 4 2 5 9 1 2 0 4 3 8 6 1 7	10
Шифр Вижинера	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3 7 5 3 7 5 3 7 5 3 7 5 3 7 5 3	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	10
	0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0	3 8 7 6 1 0 9 4 3 2 5 2 9 2 9 6 9 6 3	0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0	10
Гаммирование	0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0	2 5 20 4 1 20 9 18 6 8 14 16 13 21 14 17 21 9 0	0 1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1 0	23
	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	23 58 117 136 51 54 81 68 207 178 173 128 235 174 137 60 135 42 229 120 163	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	256
	0 0 255 255 48 32 75 117 36 4 90 255 255 0 0 0	23 58 116 135 99 86 156 185 243 182 7 127 234 174 137 60	0 0 255 255 48 32 75 117 36 4 90 255 255 0 0 0	256
Шифр Петра I	3 4 2 7 5 5 8 1 3 4 6	1 3 2 6 8 8 4 5 1 3 7	3 4 2 7 5 5 8 1 3 4 6	8
	1 8 4 6 3 6 32 4 7 1	5 4 3 7 1 7 32 3 6 5	1 8 4 6 3 6 32 4 7 1	8

Представляет интерес продолжить исследование для более широкого ряда КС, в том числе применительно к более сложным КС, включая КС с открытым ключом.

Работа выполнена при частичной поддержке РФФИ (грант 02-02-17573).

[1] Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. –М.: Радио и связь, 1999.

[2] Соболева Т.А. Тайнопись в истории России (История криптографической службы России XVIII – XX вв.). –М.: Международные отношения, 1994.

ТЕСТИРОВАНИЕ ПРОГРАММ ДЛЯ ОПРЕДЕЛЕНИЯ СПЕКТРАЛЬНЫХ СЕТОК

А.В.Грачев[1)], К.Г.Кирьянов[2)], А.В.Пастухов[2)], Р.Ф.Фахруллин[3)]

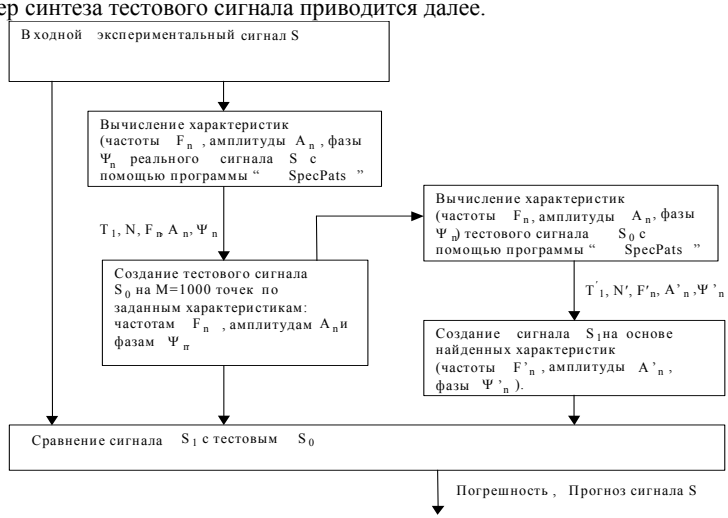
[1)]ННГУ, [2)]ННИПИ «КВАРЦ», [3)]НГТУ

В ходе проведения исследовательских работ по изучению влияния глобальных космофизических факторов на приземные процессы различной природы для автоматизации расчетов был создан программный пакет для спектральной обработки дискретных сигналов “SpecPats” [1]. В соответствии с гипотезой о законе глобальной синхронизации природных явлений основной его задачей является поиск спектральных “сеток” с периодами

$$T_n = T_1 \cdot \frac{1}{n^2}, \text{где } n = 1,2,...,N \qquad (1)$$

в сложном спектре исследуемого приземного природного процесса, зависящего, например, от периода T_1 вращающейся массы небесного тела.

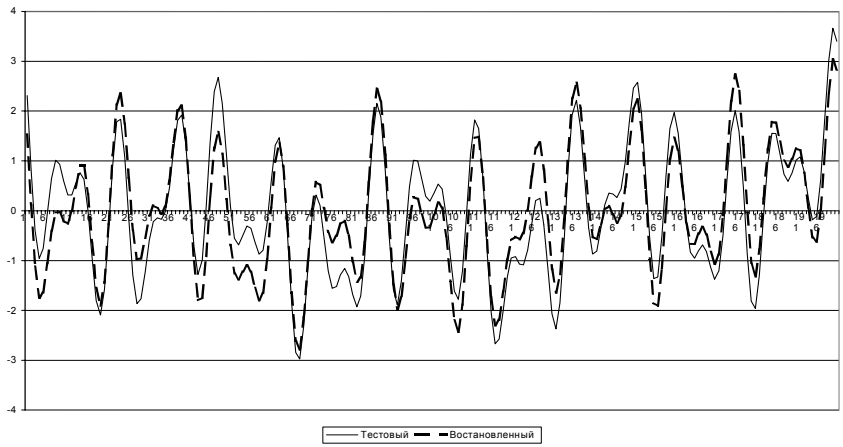
Аттестация точности частот и амплитуд спектральных сеток не является тривиальной задачей, так как результаты оценки спектров и характеристик сеток любым устройством спектральной обработки зависят не только от самого процесса, но и от логики работы программы, программной реализации базовых математических функций, разрядности вычислений, округлений и т.п. Поэтому тестирование программы и оценка погрешности проводилась нами по приведенной далее схеме, допускающей различные режимы работы в зависимости от априорной информации о реальном сигнале и целей использования результата тестирования и аттестации. Пример синтеза тестового сигнала приводится далее.



Согласно указанной схеме генерируется M выборок тестового полигармонического сигнала (2) из N гармонических составляющих сетки. В примере нами были взяты числа M =1000, N = 5, частоты (периоды) выбраны согласно формуле (1), а амплитуды и фазы – произвольно.

$$S_0(t) = \sum_{n=1}^{N} A_n \cdot \sin\left(2\pi \frac{t}{T_n} + \Psi_n\right), \quad F_n = \frac{1}{T_n} \qquad (2)$$

Преобразование Фурье тестового полигармонического сигнала (2) выполняется программой “SpecPats” с дополнением нулевыми выборками до длины, ближайшей к степени двойки (в примере – 1024 точки). Точность совпадения тестового и восстановленного сигнала (см. их графики накладки) можно оценить количественно в квадратичной и чебышевской метриках (3) (**$\Delta_{кв} \approx 0.1$,$\Delta_{чб} \approx 0.3$**).



$$\Delta_{кв} = \frac{\sum_{t=0}^{t_{max}} (S_0(t) - S_1(t))^2}{\sum_{t=0}^{t_{max}} (S_0(t))^2}, \quad \Delta_{чб} = \frac{\sum_{t=0}^{t_{max}} |S_0(t) - S_1(t)|}{\sum_{t=0}^{t_{max}} |S_0(t)|} \qquad (3)$$

Работа выполнена при частичной поддержке РФФИ (грант 02-02-17573).

[1] Пастухов А.В., Кирьянов К.Г., Фахруллин Р.Ф., Шабельников А.В. //В кн.: Тр. 6-й научн. конф. по радиофизике 5 мая 2002 г., посвященной 100-летию со дня рождения А.А.Андронова, /Ред. А.В.Якимов. –Н.Новгород: ТАЛАМ, 2002, с.310.

ЦИКЛИЧНОСТЬ ОСТРЫХ СОСУДИСТЫХ КАТАСТРОФ И ПРИРОДНЫХ ФАКТОРОВ

Е.А.Грунина, К.Г.Кирьянов, Р.Ф.Фахруллин, А.Г.Фролов

Нижегородская медицинская академия, Нижегородский университет

Вместо введения. Инфаркт миокарда и инсульт – тяжелые осложнения сердечно-сосудистых заболеваний, стоящие на первом месте среди причин смертности. В нашей стране смертность от сердечно-сосудистых заболеваний составляет у мужчин 52%, а у женщин – 63% (Р.Г. Оганов, 1991). В структуре смертности от сердечно-сосудистых заболеваний два заболевания составляют 90% всех случаев смерти – ишемическая болезнь сердца и мозговой инсульт [1]. Частота развития этих острых состояний в популяции больных не постоянна во времени и в разные дни может различаться в несколько раз. В последнее десятилетие появился ряд работ, выявляющие связи между данными медицинской статистики (например, вызовов скорой помощи, параметров крови и т.п.) с параметрами, характеризующими состояние солнечной, геомагнитной активности и метеорологическими параметрами [2–3].

Суточный. Кр-индекс

№	Частота	Период	Ампл.	Фаза	N	%
1	0.03320	30.11765	7.27266	1.14764	1	7.8
2	**0.03662**	**27.30667**	**16.50869**	**0.61042**	**1**	**1.6**
3	0.03857	25.92405	6.78972	-0.38074	1	7.1
4	0.13037	7.67041	7.31380	-2.18879	2	9.5
5	0.13184	7.58519	9.37224	1.03759	2	8.5
6	0.13379	7.47445	4.42183	-0.02272	2	7.2
7	0.13525	7.39350	6.19796	-1.01567	2	6.1
8	0.13672	7.31429	13.17053	-2.14109	2	5.1
9	0.13867	7.21127	4.76290	-2.15533	2	3.8
10	0.14063	7.11111	4.63702	2.24696	2	2.4
11	0.14307	6.98976	13.96136	-0.83762	2	0.7
12	**0.14453**	**6.91892**	**11.55103**	**-1.03508**	**2**	**0.3**
13	0.14600	6.84950	6.61626	1.93278	2	1.3
14	0.14795	6.75908	9.69822	1.63721	2	2.7
15	0.14893	6.71475	9.54005	3.04839	2	3.3
16	0.14990	6.67101	4.71420	2.57171	2	4.0
17	0.15088	6.62783	7.95497	1.46334	2	4.7
18	0.15234	6.56410	6.87574	-0.42056	2	5.7
19	0.15332	6.52229	6.98342	0.75743	2	6.4
20	0.15430	6.48101	6.59246	1.63498	2	7.1

[1] Среднесуточное атмосферное давление

№	Частота	Период	Ампл.	Фаза	N	%
[1] 1	0,03272	30,56716	1,76620	-1,80629	1	9.2
2	0,03418	29,25714	1,28764	0,49770	1	5.1
3	**0,03565**	**28,05479**	**1,57057**	**-2,73794**	**1**	**1.1**
4	0,03662	27,30667	0,98644	-2,48754	1	1.6
5	0,03809	26,25641	1,11293	0,36218	1	5.7
6	0,03906	25,60000	0,78005	0,10411	1	8.4
7	0,12988	7,69925	1,12423	-0,88121	2	9.9
8	0,13135	7,61338	0,98761	2,94807	2	8.9
9	0,13281	7,52941	0,45868	-0,81282	2	7.8
10	0,13428	7,44727	0,46583	1,92959	2	6.8
11	0,13574	7,36691	1,07707	-0,19052	2	5.8
12	0,13818	7,23675	0,56885	-1,90496	2	4.1
13	0,14014	7,13589	0,61699	-2,84221	2	2.8
14	0,14160	7,06207	0,56644	2,90685	2	1.7
15	0,14307	6,98976	1,02074	2,27912	2	0.7
16	**0,14404**	**6,94237**	**0,58465**	**1,08094**	**2**	**0.0**
17	0,14600	6,84950	0,80103	1,07363	2	1.3
18	0,14746	6,78146	0,97793	-1,68491	2	2.3
19	0,14844	6,73684	0,68979	2,75205	2	3.0

Цель. Так как коэффициент корреляции – не лучший параметр для определения степени связи между рядами [4], нами выбран новый подход – метод выделения «спектральных сеток», фактически содержащий фильтрацию [5].

Материалы и методы. Анализировали количество ежедневных госпитализаций в отделение интенсивной терапии городской больницы скорой медицинской помощи Н.Новгорода и количество смертей в связи с острым инфарктом миокарда

Поступления с инфарктом миокарда

№	Частота	Период	Ампл.	Фаза	N	%
1	0.03272	30.56716	0.06514	-2.36229	1	9.2
2	0.03418	29.25714	0.09298	2.26152	1	5.1
3	**0.03516**	**28.44444**	**0.08242**	**2.62623**	**1**	**2.4**
4	0.03711	26.94737	0.10504	2.86579	1	3.0
5	0.03906	25.60000	0.06206	2.47582	1	8.4
6	0.13184	7.58519	0.03053	2.79382	2	8.5
7	0.13281	7.52941	0.02472	-1.50094	2	7.8
8	0.13428	7.44727	0.06715	-1.48251	2	6.8
9	0.13574	7.36691	0.05398	2.55862	2	5.8
10	0.13770	7.26241	0.05373	2.24787	2	4.5
11	0.13867	7.21127	0.02179	1.91216	2	3.8
12	0.14063	7.11111	0.10373	2.43553	2	2.4
13	0.14258	7.01370	0.16704	2.32593	2	1.1
14	**0.14502**	**6.89562**	**0.11440**	**0.46776**	**2**	**0.6**
15	0.14648	6.82667	0.06982	-0.06262	2	1.6
16	0.14795	6.75908	0.09780	-0.85487	2	2.7
17	0.15039	6.64935	0.09959	2.83605	2	4.4
18	0.15381	6.50159	0.05064	-2.19463	2	6.7
19	0.15576	6.42006	0.05787	0.19273	2	8.1
20	0.15820	6.32099	0.05743	-1.73062	2	9.8

и острым нарушением мозгового кровообращения за период 1997-1999 гг. Кроме того, учитывался геомагнитный планетарный среднесуточный Kp-индекс, а также средний суточный уровень атмосферного давления и влажности за этот же срок.

Результаты. Разработанная авторами программа анализа спектров приведенных текстов осуществляет поиск периодов «спектральных сеток» ($T(n)=T_1/n^2$), T_1 – «вынуждающий период» [5]). Далее приводим примеры найденных серий периодов спектральных сеток ряда текстов: 1) суточного Kp-индекса, 2) среднесуточного атмосферного давления, 3) количества поступлений с острым инфарктом миокарда.

Все приведенные сигналы содержат 1095 выборок (за период 1997-1999 гг.), Период выборки 24.00000000 (Час). Добивка нулями до 2048 выборок. Основной период сетки 666.15997314 (Час) ≈ 22 года / (17^2), Максимальная погрешность 10.0 %%. Частота указана в (сутках) $^{-1}$.

Выводы. При спектральной обработке частоты госпитализаций и частоты смертей больных инфарктом миокарда и инсультом выявлены периоды ≈27,8 и 6,9 (≈27,8/4) суток. Это примерно соответствует циклу фаз Луны и недельному циклу. Количество поступлений и смертей коррелировало между собой, что можно объяснить тем, что эти заболевания имеют стабильный уровень летальных исходов.

Работа выполнена при частичной поддержке РФФИ (грант 02-02-17573).

[1] Сумароков А.В., Моисеев В.С. Клиническая кардиология. Руководство для врачей. – М.: Универсум Паблишинг, 1996, с.6.
[2] Агулова Л.П. //Биофизика. 1998. Т.43, вып.4. С.571.
[3] Атлас временных вариаций природных, антропогенных и социальных процессов. Т.2 Циклическая динамика в природе и обществе. –М.: Научный мир, 1998.
[4] Хабарова О.В., Руденчик Е.А. //Биомедицинские технологии и радиоэлектроника. 2002. №10-11. С.32.
[5] Кирьянов К.Г., Шабельников А.В. //Радиотехника и электроника. 1995. Вып.5. С.753.

ГЕНЕРАЦИЯ РАЗРЕШЁННЫХ НАБОРОВ КЛЮЧЕЙ КОМПИЛЯЦИИ ПРИ РЕШЕНИИ ЗАДАЧИ ОПТИМИЗАЦИИ ИСПОЛНЯЕМОГО КОДА

А.Ю.Виценко, А.Н.Хурсевич

Нижегородский госуниверситет

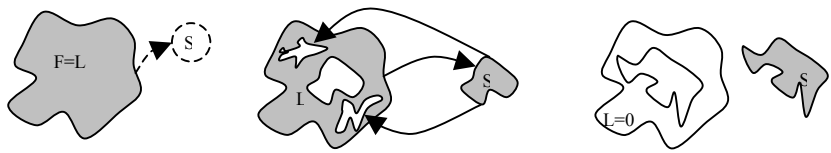
На данный момент создано много программных продуктов для тестирования производительности вычислительных систем [1]. Все они обладают теми или иными достоинствами и недостатками. К примеру, программы, измеряющие время выполнения частей кода различных приложений, занижают производительность некоторых систем (фрагменты, кода могут быть скомпилированы не оптимально для отдельных тестируемых систем). Поэтому мы получаем скорее оценку оптимальности исполняемого кода (не алгоритма или его реализации) для конкретной вычислительной системы. Другие тесты пропускают через процессор некий абстрактный код, представляющий собой “смесь” команд различного типа. Такой подход имеет как положительные, так и отрицательные моменты. С одной стороны, предпочтение не отдаётся каким-то типам систем. А с другой – при использовании псевдохаотического кода не задействуются многие возможности систем (которые могут весьма существенно влиять на производительность).

В работе реализована идея оптимизации выходного кода под различные системы с использованием одних и тех же исходных текстов. Это позволяет оценить эффективность выполнения различных алгоритмов и их реализаций на различных системах, т.е. оценить пригодность систем для решения различных типов задач.

Задача может быть решена в два этапа. На первом этапе, учитывая возможности компилятора по созданию выходного кода (набор ключей управления), получаем все возможные комбинации ключей компиляции. На втором этапе создаём исполняемые файлы и выбираем из них лучший по производительности.

На данный момент решена первая часть проблемы. Если принять во внимание, что количество ключей может достигать нескольких десятков, при этом каждый ключ может иметь порядка десяти модификаций, становится ясно, что простой перестановкой решение данной задачи неосуществимо. В то же время ключи логически связаны между собой. Например, компиляция кода с использованием MMX™ инструкций невозможна для процессоров младше Intel® Pentium® processor with MMX™ technology [2].

Рассмотрим алгоритм, позволяющий эффективно использовать эту особенность (см. рис.). При использовании данного алгоритма отношение времени, затраченного на получение “правильных” комбинаций, к общему времени работы стремится к единице, что позволяет при сильной логической связности ключей осуществлять перебор комбинаций со значительным количеством ключей.



Обозначим полный набор ключей за {F}, уже подобранную комбинацию ключей за {S}, а оставшиеся ключи за {L}. В начале работы программы массив {L} равен {F}, а подобранная комбинация {S} пуста. На первом шаге берётся ключ из массива {L} и помещается в {S}. При этом часть ключей {L} может оказаться не совместимой с ключом, только что помещённым в {S}. Такие ключи удаляются из {L}. Потом берётся следующий ключ из {L} и помещается в массив {S}. Соответственно, из {L} вновь удаляются ключи, несовместимые с только что добавленным. И так далее, пока в {L} не закончатся ключи.

Для определения ключей, которые будут удалены из {L}, используются “запреты” и “связи”. Рассмотрим сначала “запреты”. Для каждого ключа хранится набор ключей, которые с ним несовместимы. К примеру, создание кода для систем на базе процессоров Itanium® с использованием набора инструкций SSE [3]. “Связи” являются условными “запретами”, и срабатывают в случае, если были использованы (или пропущены) заданные ключи.

Для эффективного ввода данных в программу разработан специальный язык описания компиляторов. Он является достаточно простым и гибким (одной и той же цели можно добиться разными способами). Имеются различные синонимы команд. Гибкость не влияет на производительность, т.к. все данные транслируются в соответствующие структуры, а команды – в “связи” и “запреты”, которые дополнительно оптимизируются. Под оптимизацией в данном контексте понимается удаление избыточных “связей’ и “запретов”, что позволяет генерировать практически одинаковые внутренние структуры данных при использовании различных команд для их описания.

Таким образом, к настоящему моменту решена задача подготовки наборов ключей компиляции за разумное время с использованием ограниченных вычислительных. Следующим шагом будет решение задачи определения набора ключей, позволяющих генерировать максимально производительный исполняемый код.

[1] http://www.citforum.ru/hardware/app_kis/glava_6.shtml#_3
[2] http://cedar.intel.com/cgi-bin/ids.dll/content/content.jsp?cntKey=Legacy%3a%3airtmp_PRM_10660&cntType=IDS_EDITORIAL&catCode=BBU
[3] http://cedar.intel.com/cgi-bin/ids.dll/topic.jsp?catCode=BMC

АЛГОРИТМ ОБНАРУЖЕНИЯ ИЗМЕНЕНИЙ В КОМПЬЮТЕРНЫХ СИСТЕМАХ

С.В.Корелов, Л.Ю.Ротков

Нижегородский госуниверситет

В статье представлены теоретические подходы к построению алгоритма обнаружения вторжений в компьютерной системе на основе работы биологической иммунной системы.

Любая компьютерная система характеризуется признаковым пространством, включающим в себя наиболее значимые характеристики вычислительной системы. Его формирование включает следующие этапы:

- отбор на основе практических исследований наиболее информативных первичных признаков;
- минимизация признакового пространства.

Множество первичных признаков (характеристик) компьютерной системы может быть представлено такими основными группами, как процессор, оперативная память, файловая система, открытые сервисы, процессы. Нормальное поведение системы можно характеризовать набором допустимых значений данных характеристик. Однако это множество достаточно велико. Перебор всех состояний и сравнение их со значением в данный момент занимает очень много времени. Более эффективным является создание базы данных детекторов, отличных от защищаемых строк. Их количество будет значительно меньше и, следовательно, потребует меньше времени для обнаружения несанкционированного процесса.

Корректное представление контролируемых данных обеспечивается. Так некоторые характеристики измеряются относительными величинами, например, загруженность процессора измеряется в процентах. Значения таких величин необходимо перевести в цифровую форму. Для этого необходимо выбрать оптимальное количество уровней квантования по величине и шаг дискретизации по времени.

После этого необходимо измерить эти характеристики, выбрав отрезок, однозначно характеризующий поведение системы, и перевести их в двоичную строку.

После получения бинарной строки, характеризующей поведение системы, необходимо построить алгоритм распознавания нормального поведения системы от аномального (поведение при наличии несанкционированных процессов). Ниже описан один из подходов, позволяющий произвести такое распознавание [1].

Данный алгоритм имеет две основные фазы:

1) Генерация набора детекторов. Каждый из детекторов представляет собой бинарную строку, не совпадающую по определенному правилу с исходной строкой.
2) Наблюдение за защищаемыми данными путем их сравнения с детекторами из сгенерированного набора.

Работа данного алгоритма основана на очень точном описании нормального поведения системы и имеет ряд отличительных особенностей:

- нет необходимости в априорной информации о характере вторжения;

- обнаружение носит вероятностный характер, однако мы можем изменять количество детекторов в наборе;
- есть возможность запускать на выполнение несколько независимых копий процессов, реализующих данный алгоритм;
- набор детекторов в каждой из этих копий уникален, что позволяет повысить вероятность обнаружения, и оставить систему работоспособной в случае компрометации одного из процессов.

Для генерации набора детекторов, общая схема которой представлена на рис.1, необходимо вначале разбить исходную бинарную строку на сегменты одинаковой длины.

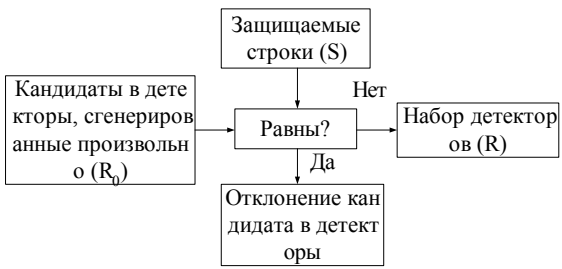


Рис. 1

Обобщенная схема обнаружения вторжений представлена на рис.2.

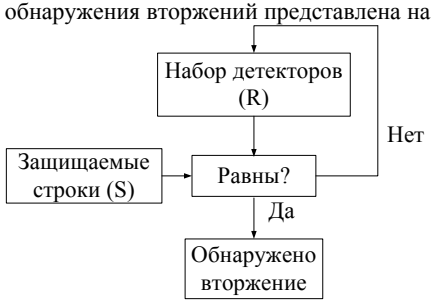


Рис. 2

Определение изменений в контролируемых данных носит вероятностный характер. Необходимо определить достаточное количество детекторов и их длину для достижения приемлемых значений вероятности обнаружения изменений, а также зависимость вероятности пропуска изменений от количества определенных детекторов.

[1] Forrest S., Perelson A., Allen L., Cherukuri R. Self-nonself discrimination in a computer. In: *Proceedings of the 1994 IEEE Symposium on Research in Security and Privacy*, 1994, pp. 202-212, Los Alamos, CA.

ОЦЕНКА ЗАДЕРЖЕК СООБЩЕНИЙ В СЕТЯХ СОТОВОЙ ПОДВИЖНОЙ СВЯЗИ СТАНДАРТА GSM

Е.С.Золотницкий, А.Е.Литвинов, Л.Ю.Ротков

Нижегородский госуниверситет

Одной из существующих проблем сетей GSM является кратковременное прерывание связи. Не всегда можно чётко сказать, что является причиной этого: плохая работа системы прерывистой передачи речи, задержки при прохождении пакетов сквозь систему, обрыв связи при эстафетной передаче (хендовере), или слабый сигнал.

С задержками пакетов бороться можно, например, повышая число обрабатывающих каналов сети. Известно, что обычно сотовая сеть не может предоставить доступ одновременно всем абонентам. Впервые о распределении вызовов в ТфОП задумался А.К. Эрланг, который, исходя из Гамма-распределения, построил своё распределение (1) применительно к телефонной сети [1]. По этому принципу можно рассчитывать любое обслуживающее оборудование (BTS, BSC, MSC и т. д.). В качестве “абонентов” будут выступать пакеты проходящие сеть от абонента к абоненту.

$$P_c = \frac{1}{\sum_{n=0}^{N-1} \frac{A^n}{n!} + \frac{A^N N}{N!(N-A)}} \cdot \frac{A^N N}{N!(N-A)}, \qquad (1)$$

где A – средний трафик ($A=XT$, X – средняя частота поступления пакетов, T – среднее время обработки пакета), N – число каналов, т.е. число пакетов, которые могут обслуживаться одновременно, P – вероятность задержки (вероятность того, что пришедший пакет не обрабатывается немедленно, а становится в очередь). Первый множитель – вероятность того, что все каналы свободны.

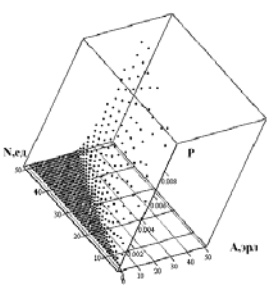


Рис. 1

На рис.1 показана зависимость вероятности P_c от величины трафика и числа каналов в устройстве. Для наглядности этот же график приведен в планарном виде на рис.2, где можно увидеть, сколько потребуется каналов на данный трафик при требуемой вероятности отказа в 1%. На пути пакета стоит не одно устройство, поэтому нужно считать суммарную вероятность задержки. Для расчёта вероятности того, что в n независимых испытаниях “успех” наступит k раз, применим формулу Бернулли:

$$P(k) = \frac{n!}{k!(n-k)!} \cdot p^k (1-p)^{n-k}, \qquad (2)$$

где p – вероятность успеха одного испытания, т.е. вероятность, рассчитанная по Эрлангу С (1).

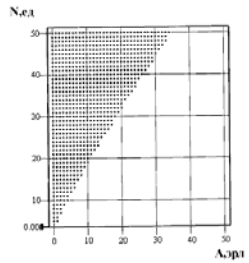


Рис. 2

На рис.3 приведен график зависимости вероятности P (попадания пакета k раз в очередь при прохождении всей сети) от числа устройств n, на которых он терпит попадание в очередь, и от вероятности P(erl), рассчитанной по формуле (1). Всего устройств 20. Из графика на рис.3 следует, что зависимость роста количества устройств, на которых пакет встает в очередь, от вероятности (1) линейная.

Можно взглянуть на проблему задержек (обрывов связи) с другой стороны. Известно, что в сетях GSM хендовер осуществляется жёстко (т.е. с временным разрывом соединения), но теоретически можно часть из них проводить мягко (без разрыва). Это можно осуществить, подготовив в новой соте свободный канал на той же частоте с тем же тайм-слотом. Также нужно следить за отсутствием интерференции и динамическим перевыделением каналов. Такой приём, вероятно, будет эффективнее работать в сети с небольшими нагрузками. Зная, что сеть загружена полностью только в час пик (обычно 2 раза в сутки), можно использовать его в остальное время. В правильно спроектированной сети избыточность числа обрабатывающих каналов присутствует большую часть времени. Эту избыточность можно использовать. Для примера такого рода расчета была написана программа. Она оценивает процент удачно проведённых мягких хендоверов. На рис.4 фрагмент программы расчетов. В углу слева: soft – число проведённых мягких хендоверов, hard – число проведённых жёстких хендоверов, soft – процент мягких от общего числа хендоверов. Данный пример работает на сети из 22-х сот, на каждой из которых 4 частоты, на каждой частоте в соответствии со стандартом – 8 тайм слотов. Цифры – номера каналов (частот), точки – абоненты.

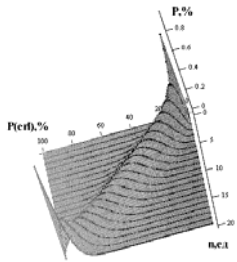


Рис. 3

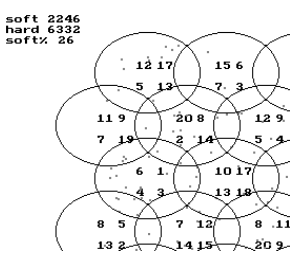


Рис. 4

В примере, приведённом на рис.4, расчёт показывает, что в данной конфигурации системы примерно каждый четвёртый хендовер проходит мягко.

[1] Ивченко Г.И., Каштанов В.А., Коваленко И.Н. Теория массового обслуживания. –М., 1982.

СРАВНИТЕЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ СОРТИРОВКИ ПРИМЕНИТЕЛЬНО К СЛОЖНЫМ СТРУКТУРАМ ДАННЫХ

Д.Н.Смородинов, А.Ю.Виценко

Нижегородский госуниверситет

Большая часть информации хранится в виде массивов разнородных элементов. Примером массива такого рода является таблица маршрутизации.

Поставщики вычислительных машин считают, что в среднем более 25% программного времени систематически тратится на сортировку. Из этой статистики можно заключить, что сортировка имеет много важных применений, либо ею часто пользуются без нужды, либо применяются в основном неэффективные алгоритмы сортировки. В работе рассмотрены: алгоритм “быстрой” сортировки сложности $O(n\log_2 n)$ [1] и алгоритмы сортировки, имеющие более низкую эффективность, но более простые $O(n^2)$ – “пузырьковая” сортировка и сортировка вставками [1] и [2]. Основной задачей было сравнение “быстрой” сортировки в двух вариантах: с использованием встроенной функции qsort и в собственной программной реализации. Алгоритмы протестированы на трех типах данных: целочисленный массив, массив структур с шестью элементами типа unsigned int, массив структур с двумя элементами unsigned int и void. Результатами измерений было количество тактов процессора, затраченное на выполнение того или иного алгоритма. Для замера количества затраченных тактов использовалась процессорная инструкция RDTSC (read-time stamp counter) [3]. Тестовая машина: 1890 MHz Athlon XP, 256 KB Cache, 512 MB RAM.

В работе было исследовано применение алгоритмов, и доказано, что время работы $O(n^2)$ алгоритмов заметно превышает время работы $O(n\log_2 n)$ алгоритмов. Для целочисленного 100K массива сортировка вставками в 2,5 раза быстрее, чем сортировка “пузырьком”, qsort (штатная реализация алгоритма “быстрой” сортировки) примерно в 1900 раз быстрее “пузырька”, quicksort (собственная программная реализация алгоритма “быстрой” сортировки) примерно в 2300 раз быстрее “пузырька”. Сортировка вставками примерно в 760 раз медленнее, чем qsort и в 910 раз медленнее, чем quicksort.

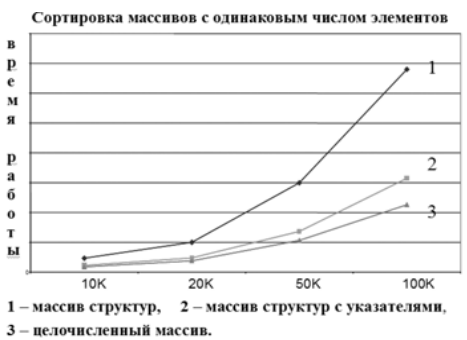


Рис. 1

Видно, что лучшим выбором для сортировки большого массива является алгоритм “быстрой” сортировки. Автором разработана собственная его реализация, превосходящая по быстродействию стандартные средства, присутствующие в сис-

теме. Сортировка 20 К массива двухэлементных структур заняла 2 мс ($4*10^6$ тактов процессора), 100 К целочисленного массива – 13 мс ($24*10^6$ тактов процессора).

Возможен вариант комбинированного использования сортировок. Для исходного, не отсортированного массива применяется метод “быстрой” сортировки, затем, во время использования в массив вносятся небольшие изменения, происходит пересортировка вставками. Это, несомненно, дает выигрыш в производительности. Сортировка вставками массива целых чисел, отсортированного по возрастанию с измененными первым, средним и последним элементами, для 10 К элементов в 4 раза быстрее, чем quicksort, для 100 К элементов в – 5,5 раз быстрее.

Для большого массива многоэлементных структур, (например, таблицы маршрутизации) целесообразнее разбить этот массив на два. Один будет состоять из двухэлементных структур: критерия сортировки и указателя на элемент второго массива. Второй содержит всю оставшуюся информацию. Это сильно экономит процессорное время и уменьшит объемы пересылаемой информации (рис.1 и рис.2).

Рис. 2

Разработано программное обеспечение, реализующее данные алгоритмы, написанное на языке Си как для операционных систем семейства Windows, так и для операционных систем семейства Linux.

Тем самым задача по оптимальной сортировке успешно решена, и намечены пути по реализации алгоритмов поиска и обработки сложных массивов.

[1] Кнут Д. Искусство программирования для ЭВМ. Т.3 Сортировка и поиск. Пер. с англ. – М.: Мир, 1978.
[2] Robert Sedgewick. “Algorithms in C++”. Third Edition. Addison-Wesley, 1999.
[3] Intel® Developer Services - Using the RDTSC Instruction for Performance Monitoring.