ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

8(144) 2008

ТЕОРЕТИЧЕСКИЙ И ПРИКЛАДНОЙ НАУЧНО-ТЕХНИЧЕСКИЙ ЖУРНАЛ

Издается с ноября 1995 г.

УЧРЕДИТЕЛЬ
Издательство "Новые технологии"

СОДЕРЖАНИЕ

Аннотации статей размещены на сайте журнала по адресу http://www.**informika.ru/text/magaz/it/** или **http://novtex.ru/IT.**

Журнал входит в Перечень научных журналов, в которых по рекомендации ВАК РФ должны быть опубликованы научные результаты диссертаций на соискание ученой степени доктора наук.

УДК 519.68

П. Е. Ильин, мл. научн. сотр., аспирант,
Московский энергетический институт

Многозадачная территориально распределенная вычислительная среда с учетом затрат на передачу данных

Приводится обзор проблем, возникающих при организации распределенных вычислений, рассматривается архитектура среды для организации распределенных вычислений в многозадачном режиме с возможностью динамического изменения приоритетов, предлагается алгоритм распределения задач с учетом затрат на передачу данных, проводится сравнение эффективности предложенной среды и кластера.

Ключевые слова: *многозадачность, распределенные вычисления, территориально распределенные вычисления, вычислительная среда, управление приоритетами, планировщик, планирование задач.*

Введение

Повышение производительности процессоров и рост пропускной способности каналов связи, наблюдавшиеся за последние несколько десятилетий, сделали возможным применение обычных персональных компьютеров во многих новых областях, в том числе и в области управления территориально распределенными промышленными объектами. Вместо централизованного сбора данных о состоянии объекта для обработки их на суперкомпьютере, применявшегося ранее, мощные персональные компьютеры сделали возможным другой подход: применение распределенных вычислений (являющихся частным случаем GRID-вычислений) — максимально возможный объем данных обрабатывается по месту их возникновения, а централизованно обрабатывается только та часть данных, которая необходима для анализа состояния объекта в целом.

Такой подход дает целый ряд преимуществ как с экономической точки зрения, так с точки зрения производительности и надежности. Преимущества с экономической точки зрения обусловливаются тем, что стоимость суперкомпьютера на несколько порядков выше, чем стоимость нескольких десятков (или сотен) типовых компьютеров, осуществляющих обработку данных на местах. Преимущество с точки зрения производительности возникает вследствие того, что во многих случаях основным ограничивающим фактором является не производительность процессора, а пропускная способность каналов связи, в результате чего задача решается быстрее за счет существенного уменьшения объема передаваемых данных. Преимущества с точки зрения надежности заключаются в том, что в случае отказа канала связи управление соответствующей частью объекта будет осуществляться с помощью локального компьютера. По этой же самой причине распределенный подход позволяет более оперативно реагировать на аварийные ситуации на самом управляемом объекте.

Однако переход к распределенным вычислениям создает ряд новых проблем, отсутствовавших при централизованной обработке данных. В частности, к таковым относятся:

- отслеживание доступных вычислительных ресурсов;
- организация обмена данными на уровне логических номеров параллельных процессов для обеспечения независимости от конфигурации сети;
- обеспечение возможности проводить асинхронные обмены данными;
- удаленный запуск процессов и задач;
- обработка ошибочных ситуаций (отказов одного из компьютеров, участвующих в вычислениях, или каналов связи с ним);
- сбор отладочной информации о ходе выполнения параллельных процессов.

Решение этих проблем является типичными для подавляющего большинства задач управления территориально распределенным объектами. В связи с этим возникла необходимость создания программной среды, которая избавила бы разработчиков от необходимости решать эти проблемы при реализации каждой задачи. Кроме того, так как при решении отдельных задач в общем случае компьютеры загружены неравномерно, то для их эффективного использования необходима реализация многозадачного режима с разграничением приоритетов задач.

Архитектура территориально распределенной вычислительной среды

Архитектура территориально распределенной вычислительной среды во многом похожа на ар-

хитектуру кластера, однако имеется ряд отличий. Наиболее существенным из них является необходимость регулярного отслеживания затрат времени на передачу данных между компьютерами в составе среды и учет различия этих затрат на этапе распределения параллельных процессов по компьютерам.

Территориально распределенная вычислительная сеть (ТРВС) представляет собой множество неспециализированных компьютеров (в дальнейших именуемых вычислителями), соединенных между собой каналами связи, и специализированного программного обеспечения, которое и выполняет поставленные перед ТРВС задачи по управлению распределенными вычислениями [1—3].

Программное обеспечение ТРВС состоит из нескольких компонентов: координирующего процесса, локального диспетчера, клиентской библиотеки, средств мониторинга.

Схема архитектуры ТРВС (без средств мониторинга) представлена на рисунке.

Задачи *координирующего процесса* — это хранение списка доступных вычислительных ресурсов, списка выполняемых задач, распределение задач и сбор информации о ходе выполнения вычислительных процессов. Непосредственно в решении самих вычислительных задач координирующий процесс не участвует.

Локальный диспетчер запускается на каждом из вычислителей и выполняет следующие функции: устанавливает соединение с координирующим процессом и передает ему информацию о параметрах данного вычислителя, удаленно запускает выполняемые задачи, организует обмены данными между параллельными процессами прикладных задач с промежуточной буферизацией данных, уведомляет координирующий процесс об ошибочных ситуациях, а также отправляет ему информацию о ходе выполнения задач.

Задачи *клиентской библиотеки* — установка соединения с локальным диспетчером, передача данных о требованиях запускаемой задачи, передача данных для обмена данными с другими параллельными процессами локальному диспетчеру, а также обработка сообщений об отказе одного из удаленных процессов и запросов на принудительное завершение задачи.

Средства мониторинга подключаются к координирующему процессу ТРВС, получают от него информацию о доступных вычислителях, выполняющихся задачах и их распределении по вычислителям, а также информацию о ходе выполнения задач и отображают ее в удобной для восприятия человеком форме.

Взаимодействие всех компонентов ТРВС осуществляется по собственному протоколу, реализованному поверх протокольного стека TCP/IP.

Логическая организация вычислительных процессов внутри задачи не связана напрямую с архитектурой территориально распределенной среды и определяется разработчиком прикладной задачи произвольно. На практике обычно применяется двухуровневая или трехуровневая организация вычислительных процессов. При двухуровневой организации задача состоит из одного координирующего вычислительного процесса и множества вычислительных процессов, осуществляющих первичную обработку данных. При трехуровневой организации к перечисленным выше процессам добавляется также промежуточный уровень — вычислительные процессы регионального уровня. Но и в том, и в другом случае с точки зрения ТРВС все эти вычислительные процессы будут полностью равноправными.

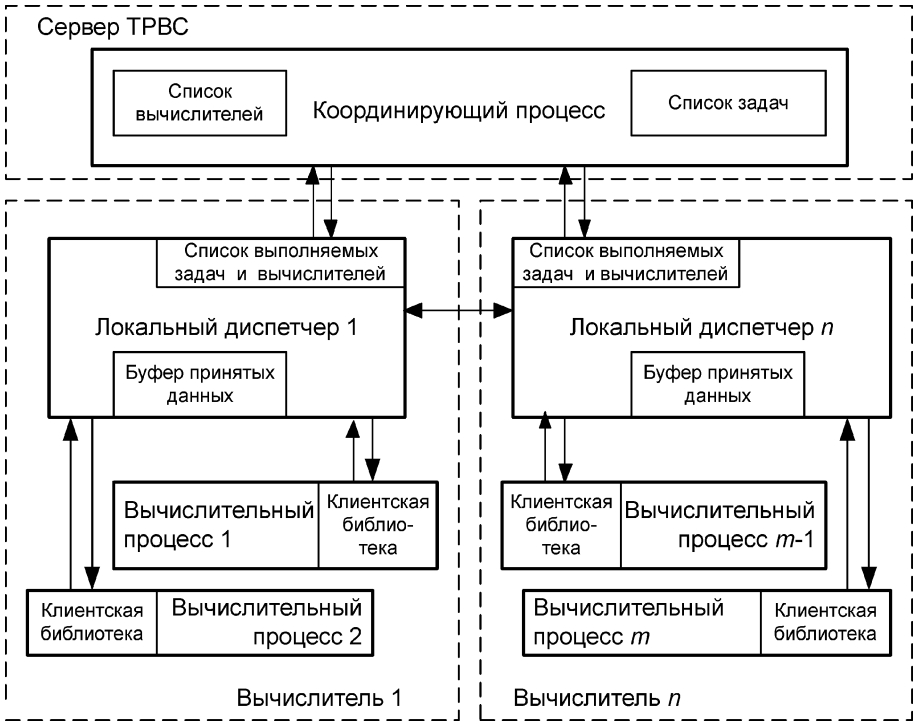


Схема архитектуры глобально распределенной вычислительной среды

Управление приоритетами задач

При использовании многозадачного режима возникает необходимость обеспечить выполнение задач с различными приоритетами. Во всех современных операционных системах имеются встроенные механизмы управле-

ния приоритетами процессов, в которых приоритет определяется числом квантов процессорного времени, выделяемых тому или иному процессу за единицу реального времени.

Но при реализации распределенных вычислений может возникнуть необходимость запуска задачи, своевременное выполнение которой насколько важно, что для его обеспечения становится необходимой приостановка других задач, при этом после завершения критической задачи исходные приоритеты должны быть восстановлены. Если учесть, что таких задач может быть несколько и они могут иметь разную степень критичности, а число уровней приоритетов в операционной системе невелико, то становится очевидным, что необходимы дополнительные средства по управлению приоритетами. Эти средства и предоставляет территориально распределенная вычислительная среда.

Изменение приоритетов может быть реализовано двумя способами. Первый — централизованное планирование в координирующем процессе ТРВС и рассылка команд каждому из вычислителей на изменение приоритетов для указанных процессов. Второй способ — использование механизма "скользящих приоритетов". Суть этого механизма заключается в том, что задаче при запуске назначается условный приоритет, и далее локальный диспетчер на основании соотношений условных приоритетов всех выполняющихся на нем задач рассчитывает, какой приоритет (в терминах операционной системы) назначить каждому конкретному процессу.

Алгоритм работы механизма "скользящих приоритетов" следующий.

1. При запуске задачи управляющий сервер присваивает ей уровень приоритета L и отправляет команду на запуск задачи локальному диспетчеру.

2. Осуществляется запуск процессов задачи, но при этом процессы находятся в остановленном состоянии.

3. Среди всех задач, выполняемых на каждом вычислителе, локальный диспетчер определяет максимальный относительный приоритет $L_{\max}$.

4. Для каждой из задач определяется величина $l = L - L_{\max} + L_{os}$, где L — относительный приоритет данной задачи, L_{os} — число уровней приоритетов в операционной системе.

5. Если l для данной задачи меньше или равна нулю, то все процессы этой задачи, которые выполняются на данном вычислителе (так называемые локальные процессы), приостанавливаются. Если же значение l положительно, то всем локальным процессам задачи присваивается уровень приоритета l средствами операционной системы.

6. При завершении одной из выполняющихся задач проводится проверка, есть ли еще задачи на данном вычислителе. Если задачи есть, то осуществляется переход к п. 3 алгоритма, в противном случае — обнуление величины $L_{\max}$ и ожидание запуска новых задач.

Рассмотренный механизм "скользящих приоритетов" позволяет передать большую часть действий по управлению приоритетами от координирующего процесса локальным диспетчерам вычислителей, что повышает надежность работы ТРВС в целом, уменьшает объем передаваемой служебной информации и время отклика на изменение ситуации на вычислителе. Механизм "скользящих приоритетов" может быть применен не только на уровне задач, но и на уровне параллельных процессов, порождаемых каждой задачей.

Особенности алгоритма планирования задач

Математическое представление вычислительной среды. Наиболее существенным отличием ТРВС от кластера является необходимость учета влияния территориальной распределенности при распределении процессов задач по вычислителям. Введем *коэффициент затрат передачи данных между двумя вычислителями*, который будет тем больше, чем менее эффективна передача данных между двумя конкретными вычислителями. На практике этот коэффициент может быть либо временем передачи пакета данных определенного размера (которое периодически замеряется средствами ТРВС), либо величиной, обратной пропускной способности канала, либо стоимостью передачи данных по каналу связи (в этом случае коэффициенты задаются явно при настройке ТРВС).

Теперь можно представить математическую модель ТРВС как граф $\mathbf{G0} = (\mathbf{P0}, \mathbf{V0})$, в котором множеству вершин **P0** соответствуют вычислители, а множеству ребер **V0** — каналы связи между вычислителями. Граф **G0** является взвешенным как по вершинам, так и по ребрам. Весовыми коэффициентами вершин $p0_i$ ($i = 1, ..., N$) являются показатели производительности вычислителей. Весовыми коэффициентами ребер $v0_{ij}$ являются коэффициенты затрат на передачу данных по каналу связи.

Назовем такой граф разбиением ТРВС нулевого уровня и выполним построение разбиений более высоких уровней. Для этого зададим некоторый шаг приращения коэффициента затрат на передачу Δv_1, затем в графе **G0** выберем некоторую вершину (например, $p0_1$) и те вершины, для которых весовой коэффициент ребер, соединяющих их с выбранной, не превышает величины Δv_1. В граф **G1** помещается вершина $p1_1$, весовой коэффициент которой равен сумме весов выбранных вершин в графе **G0**. (Выбранные вершины

графа **G0** называются вершинами, соответствующими вершине $p1_1$.) Выбранные вершины удаляются из графа **G0**. Вершины графа **G1** и графов более высоких уровней разбиения называются *вычислительными регионами*.

Далее выбирается следующая вершина в оставшейся части графа **G0**, и для нее повторяются те же самые действия. Выбираются те вершины, расстояние до которых не превышает Δv_1, и в граф **G1** помещается вершина $p1_2$, весовой коэффициент которой равен сумме весов выбранных вершин. Затем в граф **G1** добавляется ребро, весовой коэффициент которого определяется как величина, обратная сумме обратных величин весов тех ребер, которые соединяли в графе **G0** вершины, соответствующие вершинам $p1_1$ и $p1_2$ в графе **G1**. Аналогичные шаги повторяются до тех пор, пока из графа **G0** не будут удалены все вершины.

Аналогично выполняется построение разбиений более высоких уровней с увеличением шага до $\Delta v_l = l\Delta v_1$ (где l — номер шага) до тех пор, пока в результате очередного разбиения на шаге L не будет получен граф **GL**, состоящий всего из одной вершины. В итоге получаем множество разбиений ТРВС-среды $\mathbf{G} = \{\mathbf{G}l\}$ (где $l = 0, ..., L$).

Представление задач. Задачи, решаемые в ТРВС, представляются так же, как и при параллельных вычислениях: в виде ориентированных графов, где вершины — это этапы вычислений, а дуги — обмены данными, но при этом имеется одна особенность: у графа есть нулевой ярус, вершины которого имеют коэффициент 0, а вес дуг, соединяющих его вершины с первым ярусом, соответствует объему исходных данных (например, показаний каких-либо датчиков) для решения задачи.

Далее осуществляется процесс "сжатия" графа **T** задачи в граф параллельных процессов **TT**: из каждого яруса графа задачи выбирается одна вершина — вычислительный этап, который будет выполняться в рамках одного параллельного процесса. Весовой коэффициент новой вершины равен сумме весовых коэффициентов выбранных вершин графа **T**. После того, как в граф параллельных процессов помещены все вершины параллельных процессов, они связываются ребрами, весовые коэффициенты которых равны сумме весовых коэффициентов дуг, связывающих вершины в графе **T**, соответствующие данным параллельным процессам (в том случае, если сумма получилась равной нулю, дуга не добавляется).

Далее для каждого параллельного процесса, соответствующего вершине c_{Tj} графа **TT**, введем понятие *показателя свободы параллельного процесса*

$$f_j = <c_{Tj}, l_j, p_{lk}>,$$

где c_{Tj} — j-я вершина графа **TT**; p_{lk} — вычислительный регион, к которому привязан данный параллельный процесс; l_j — степень свободы данного параллельного процесса.

Степень свободы параллельного процесса — это величина, которая показывает, в пределах какого уровня разбиения ТРВС **G**l возможен выбор вычислительного региона для данного процесса (нулевая степень свободы означает жесткую привязку к конкретному вычислителю, а степень свободы $l = L$ — возможность выполнять данный процесс на любом вычислителе ТРВС).

Обобщенный алгоритм планирования задач. После того как множество показателей свободы размещения параллельных процессов $\mathbf{F} = \{f_j\}$ полностью задано, выполняется распределение задачи по вычислителям, которое состоит из следующих шагов.

Шаг 1. Определяется максимальная степень свободы $L_{\max}$ и минимальная $L_{\min}$ среди всех процессов данной задачи. Текущей степени свободы L присваивается значение $L_{\min}$.

Шаг 2. Проводится назначение процессам со степенью свободы L того вычислительного региона pL_k разбиения **GL**, который указан в показателе свободы параллельного процесса $f_j = <cT_j, l_j, pL_k>$. При этом вспомогательная степень свободы l приравнивается $L - 1$. Если $l < 0$, то переход к шагу 5.

Шаг 3. Выполняется распределение процессов со степенью свободы L по регионам на текущем вспомогательном уровне разбиения l, при этом соблюдается правило: процессу назначается одна из тех вершин графа, которые соответствуют предыдущему уровню разбиения.

Шаг 4. l уменьшается на 1. Если $l = 0$ (т. е. не достигнут самый низкий уровень разбиения), то повторяется шаг 3.

Шаг 5. L увеличивается на 1. Если $L \leqslant L_{\max}$, то повторяются шаги 2—4, в противном случае распределение параллельных процессов считается завершенным.

При каждом выполнении шага 3 данного алгоритма (кроме первого) осуществляется выбор для каждого из параллельных процессов вычислительного региона с учетом следующих параметров: объема вычислений в данном процессе и количества передаваемых данных при обменах с другими вычислителями, степени свободы процесса, затрат на передачу данных, а также производительности вычислителей и наличия других решаемых задач. При этом могут использоваться те же стратегии, что и при реализации параллельных вычислений, описанные, например, в [4, 5].

Обработка ошибочных ситуаций

Одной из вспомогательных задач ТРВС является сбор информации о ходе выполнения задач и об ошибочных ситуациях. ТРВС способна ав-

томатически обнаруживать ряд наиболее типичных ошибок в задаче, в том числе: попытки передать пакет несуществующему вычислителю; наличие "невостребованных" пакетов в приемном буфере после завершения задачи; попытка передать данные, адрес в памяти которых некорректен; попытка передать пакет с некорректной длиной.

Еще одна существенная проблема, возникающая при распределенных вычислителях, — обработка отказов вычислителей или каналов связи. Такие ситуации можно разделить на три категории:

- невозможность передать данные из-за отказа вычислителя-получателя;
- невозможность получить данные из-за отказа вычислителя-отправителя;
- потеря связи с локальным вычислительным процессом.

Во всех этих ситуациях применяется механизм тайм-аутов, в течение которых происходят попытки восстановить соединение. Если тайм-аут истек, а соединение не восстановлено, то информация об этом отправляется координирующему процессу и начинается процедура восстановления отказавшего процесса. Так как логика организации вычислительных процессов прикладной задачи в общем случае ТРВС неизвестна, то для восстановления задачи предлагается ряд заранее запрограммированных сценариев:

1) аварийное завершение оставшихся процессов и перезапуск всей задачи в целом;

2) рассылка уведомлений остальным процессам о том, что процесс с указанным номером более не доступен, и блокировка пересылки данных этому процессу;

3) перезапуск отказавшего процесса на другом вычислителе с рассылкой уведомлений об этом остальным процессам задачи, при этом передача данных, которые нужны процессу для продолжения решения задачи, осуществляется другими процессами самой задачи;

4) перезапуск отказавшего процесса на другом вычислителе, при этом рассылка данных для восстановленного процесса проводится средствами ТРВС-сети.

При использовании сценариев восстановления 3 и 4 может быть введено понятие *логического этапа задачи*. В этом случае задача разбивается на этапы таким образом, что в случае отказа процесса возможно продолжить ее выполнение с начала текущего этапа, а не перезапускать задачу целиком. Наиболее наглядным примером применения логических этапов являются задачи, решаемые с применением итераций, где началом нового логического этапа является обмен данными перед началом новой итерации вычислений.

Оценка эффективности применения территориально распределенной среды

В общем случае время решения задачи в распределенной среде складывается из следующих составляющих:

$$T_{\text{реш}} = \Sigma c_i/p_i + \Sigma u_j v_j + \Sigma w_k q_k,$$

где c_i — объем вычислений на i-м этапе; p_i — производительность вычислителя, на котором выполняется i-й этап; u_j — объем передаваемых данных на j-м обмене; v_j — затраты на передачу 1 байт данных на j-м обмене; w_k — объем данных, доставляемых от источника возникновения данных до вычислителя, на котором выполняется первый этап k-го процесса; q_k — затраты на передачу 1 байт данных при доставке начальных данных k-му процессу.

Примечание: при подсчете времени по приведенной формуле учитываются только те узлы и обмены, которые находятся на критическом пути графа задачи.

Если сопоставить время решения задачи на кластере (или суперкомпьютере) и на ТРВС, получаем следующее соотношение:

$$K_{\text{уск}} = T_{\text{реш.ТРВС}}/T_{\text{реш.кл}} = (\Sigma c_i/p_{i\text{ТРВС}} + \Sigma u_j v_{j\text{ТРВС}} + \Sigma w_k q_{k\text{ТРВС}})/(\Sigma c_i/p_{i\text{кл}} + \Sigma u_j v_{j\text{кл}} + \Sigma w_k q_{k\text{кл}}).$$

Очевидно, что при одинаковом методе решения объемы вычислений c_i, объемы начальных доставляемых данных w_k и объемы данных для обменов u_j будут одинаковыми. Производительность вычислителя p_i для ТРВС, как правило, бывает меньше или равна производительности узла кластера. Затраты на передачу данных v_j в процессе обмена на кластере значительно меньше, чем в ТРВС, за счет применения высокоскоростных каналов связи. С затратами на начальную доставку данных ситуация иная: вычислительные процессы в ТРВС распределяются так, чтобы обработка данных велась как можно ближе к месту их возникновения, поэтому затраты на передачу этих данных на вычислитель малы, тогда как затраты на передачу на кластер $w_{k\text{кл}}$ сопоставимы с затратами на обмен данных ТРВС ($v_{j\text{ТРВС}}$), так как передача данных осуществляется по тем же самым каналам связи.

Таким образом, получить время решения задачи на ТРВС-сети меньше, чем на кластере, возможно в том случае, если объем начальных данных значительно больше объема данных, передаваемых в процессе обменов между вычислителями, а затраты на доставку начальных данных к вычислителям ТРВС меньше, чем затраты на доставку этих же данных до кластера.

В частном случае при $p_{i\text{кл}} = p_{i\text{ТРВС}}$ и при одинаковых $v_{\text{кл}}$, $q_{\text{кл}}$, $v_{\text{ТРВС}}$ и $q_{\text{ТРВС}}$ получаем, что применение ТРВС будет эффективнее применения кластера при следующем соотношении объемов начальных данных и данных для обмена: $\Sigma w/\Sigma u \geqslant (v_{\text{ТРВС}} - v_{\text{кл}})/(q_{\text{ТРВС}} - q_{\text{кл}})$.

Заключение

Предложенная в данной статье территориально распределенная вычислительная среда с точки зрения архитектуры содержит в себе черты как кластера, так и GRID-технологии. Отличительной особенностью ТРВС является возможность учитывать затраты на передачу данных на этапе планирования, что позволяет достигнуть большей эффективности по сравнению с кластерными технологиями за счет уменьшения затрат времени на передачу начальных данных от источника их возникновения до вычислительного процесса.

В настоящее время на кафедре ВМСиС Московского энергетического института разработан макет территориально распределенной вычислительной сети, получившей название "параллельная мультикомпьютерная сеть КУРС-2004" [1—3].

Список литературы

1. **Дзегеленок И. И., Ильин П. Е.** и др. Декомпозиционный подход к осуществлению GRID-технологий // Информационная математика. 2005. № 1 (5). С. 110—119.

2. **Дзегеленок И. И., Ильин П. Е.** Проект глобально-распределенной мультикомпьютерной среды для реализации декомпозиционных моделей управления // Тр. III Междунар. конф. "Параллельные вычисления и задачи управления" PACO'2006, Москва, 2—4 октября 2006 г. М.: Институт проблем управления им. В. А. Трапезникова, 2006. С. 400—406.

3. **Ильин П. Е.** Стратегия распределения параллельных процессов в глобально-распределенной мультикомпьютерной среде с учетом многозадачности // Тр. III Междунар. конф. "Параллельные вычисления и задачи управления" PACO'2006, Москва, 2—4 октября 2006 г. М.: Институт проблем управления им. В. А. Трапезникова, 2006. С. 407—412.

4. **Воеводин В. В., Воеводин Вл. В.** Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

5. **Топорков В. В.** Модели распределенных вычислений. М.: Физматлит, 2004. 320 с.

УДК 517.9

В. А. Петров, аспирант,
А. Г. Тормасов, канд. физ.-мат. наук,
доц., зам. зав. каф.,
МФТИ

Длительность поиска в распределенной системе с регулярной структурой в условиях точечной загруженности

Рассмотрено поведение длительности поиска в распределенной системе при повышении загруженности сети вплоть до критических значений. Изучение проводится в рамках двух моделей. Результаты сравниваются друг с другом.

Ключевые слова: *распределенная система, тиражирование данных, прослойка специальных серверов, марковские процессы, сети Петри, N-k схема хранения файлов, поиск частей файла, регулярная структура графа, точечная загруженность, имитационное моделирование, эмпирическая функция распределения, вероятностное пространство, алгоритм Дейкстры.*

Введение

Любая реальная система имеет свои ограничения, пределы, за которыми ее использование перестает быть целесообразным. Узнать эти ограничения — одна из важнейших задач. Фактически это и есть познавание самой системы. Еще Аристотель утверждал, что различие сделанных из однородной материи объектов определяется их формой, т. е. границами.

Длительность поиска — ключевой параметр для скорости реакции системы на пользовательские команды. Поиск расположенных в разных частях распределенной системы частей файла лежит в основе операции чтения. Именно его длительность определяет, сколько времени пройдет от сигнала о желании пользователя прочесть файл до получения на экране его содержимого.

Данная работа посвящена изучению характера поведения длительности поиска в распределенной системе в условиях повышения загруженности сети, в частности, обучению находить предельно допустимую загруженность. Предполагается, что повышение загруженности происходит в результате роста числа запросов к определенным узлам распределенной системы. Такая схема не случайна, она взята из реальной ситуации: сайт, посвященный Олимпийским играм в Сиднее, во время их проведения одновременно посещали несколько миллионов пользователей, что сильно сказывалось на скорости получения данных каждым из пользователей. Во время некоторых соревнований сервер был недоступен из-за перегрузки.

Для определения характера поведения длительности поиска строятся две модели: первая основана на имитационном моделировании, вторая вычисляет длительность поиска по формуле — она условно названа теоретической оценкой. Обе модели можно назвать вероятностными: основные величины представлены в них случайными величинами. Сравнение полученных результатов показывает как искомую зависимость длительности поиска, так и достоинства и недостатки выбранных моделей.

1. Обзор

Поиск в распределенных системах

Pastry. Pastry была разработана в Microsoft Research в 2001 г. Она представляет собой прослойку для распределенных систем, обеспечивающую поиск и хранение объектов [4, 5].

Узлам и объектам хранения в Pastry назначены идентификаторы. На каждом шаге поиска узел отсылает запрос соседу с идентификатором, численно более близким к идентификатору искомого объекта. Узел с наиболее близким идентификатором гарантированно содержит данный объект. Такая процедура сходится за $\log N$ шагов, где N — общее число узлов в Pastry.

Предлагаемые решения оказались достаточно удачными: Pastry легла в основу таких распределенных систем, как PAST, SCRIBE, а также Beehive, описываемая ниже.

Beehive. Распределенная система Beehive была спроектирована исходя из требований высокой скорости доступа к хранимым данным, хорошей масштабируемости и устойчивости к внезапным вспышкам интереса к каким-то выделенным данным.

Высокая скорость доступа достигается в системе за счет оптимизации так называемых распределенных хэш-таблиц (PXT), таких, например, какой является Pastry. Обычно они доставляют требуемую информацию в сети из N компьютеров за время $O(\log N)$. Beehive же улучшает эту оценку до $O(1)$.

Чтобы увеличить скорость доставки, Beehive использует предварительное тиражирование данных (в отличие от систем, кэширующих данные прямо в процессе их запроса). Beehive распределяет значения уровней тиражирования исходя из эмпирического закона Зипфа (*Zipf's law, power law*). Он утверждает, что число запросов к объекту обратно пропорционально его индексу в списке самых популярных: $K(i) \sim 1/i^{\alpha}$. Закон Зипфа позволяет поддерживать большое число копий только для небольшого числа самых "популярных" объектов.

Подробнее о системе Beehive см. в [6].

Google File System (GFS). GFS — это кластерная распределенная файловая система, созданная компанией Google для своего внутреннего использования. GFS разбивает каждый файл на части (*chunks*) размером 64 Мбайт. Каждый из них хранится в нескольких копиях.

Архитектура GFS различает два типа серверов: мастер (*master*) и хранитель (*chunkserver*). Мастер-сервер инициирует все операции в системе, следит за ее целостностью, восстанавливает после сбоев и т. д. (таким образом, GFS — это централизованная распределенная система). Мастер-сервер хранит только метаданные файлов и местонахождение их частей. Сама информация располагается на серверах-хранителях.

Рассмотрим подробнее процедуру получения информации клиентом GFS [7]. Проситель, зная размер части файла в GFS, вычисляет индекс той, которая содержит интересующие его данные (пусть для простоты такая часть — одна). Свой запрос он направляет мастер-серверу. Мастер-сервер находит расположение всех копий (т. е. адреса содержащих их серверов-хранителей) требуемых частей и возвращает его клиенту, после чего клиент скачивает "ближайшую" копию части. При этом близость двух компьютеров определяется только близостью их IP адресов. Соответствующее распределение адресов среди серверов GFS предполагается.

Akamai. Проект Akamai как системы ускоренной доставки информации из Интернета появился в Массачусетсском технологическом институте (MIT) в начале 1995 г.

Доставка информации в Akamai ускоряется за счет "расширения" главных узких мест в Интернете. Ключевую роль в этом играет прослойка специальных серверов (*Edge Services*) между источником информации и получателем. "Расширение" достигается следующими способами:

- на участке между двумя Edge-серверами — благодаря их правильному расположению и хорошим соединениям между ними;
- на участке между Edge-сервером и клиентом — благодаря выбору ближайшего сервера. Специальные службы на серверах регулярно составляют карты трафика локальной подсети, определяют степень загруженности соседних машин. Исходя из этой карты и выбирается ближайший сервер.

Подробнее об Akamai можно узнать из работы [8].

Модели распределенных систем

Моделирование с помощью марковских процессов. При моделировании марковскими процессами распределенная система представляется графом, в котором узлы — это состояния системы, а

ребра — переходы между ними. Каждое ребро имеет свой вес — вероятность данного перехода.

Так, в работах [9, 10] с помощью марковских процессов моделируются распределенные системы управления космическим аппаратом и движением автомобильного транспорта, в работе [11] — системы мониторинга опасных промышленных объектов, системы охранной и пожарной безопасности и т. п.

Моделирование с помощью сетей Петри. *Обыкновенные сети Петри* [13] представляют собой двудольный ориентированный мультиграф $N = \{P, T, F\}$ с множеством вершин $P \cup T$ и функцией инцидентности F: $(P \times T) \cup (T \times P) \to N \cup \{0\}$ (N — множество натуральных чисел), задающей кратность дуг. Вершины из P называются позициями и соответствуют локальным состояниям системы, а вершины из множества T — переходами и задают действия или события. Текущее состояние сети Петри определяется его разметкой M: $P \to N \cup \{0\}$, сопоставляющей каждой позиции сети число фишек (маркеров) в ней.

В *сетях Петри высокого уровня* [14] фишка приобретает тип, переходы могут срабатывать в различных режимах, удаляя фишки из одних позиций и добавляя их в другие, кратности дуг становятся переменными.

Во *вложенных сетях Петри* [12] фишки в позициях сети сами могут быть сетями Петри.

В [12] показано, что, в отличие от сетей высокого уровня, вложенные сети строго выразительнее обыкновенных сетей Петри, но все же слабее по выразительности, чем универсальные вычислительные модели (например, машины Тьюринга). Для вложенных сетей Петри неразрешима проблема достижимости (по данным начальной и конечной разметкам для данной сети Петри определить, достижима ли конечная разметка из начальной), которая разрешима для обыкновенных сетей Петри. Однако для вложенных сетей Петри разрешимы проблемы останова, покрытия и некоторые другие.

2. Определение длительности поиска

Базовые понятия

Поиск в распределенной системе. Распределенную систему будем понимать как граф, узлы которого есть компьютеры, хранящие некоторые данные, а ребра — физические каналы передачи информации. Каждый элемент графа — узел или ребро — характеризуется определенным набором параметров, например, производительностью машины или пропускной способностью узла.

Каждый файл в хранилище разбивается на порции по так называемой ($N - k$)-схеме: на N порций, причем так, что из любых k порций можно собрать весь блок целиком ($N \geqslant k$) [1].

Чтобы собрать блок файла, необходимо в первую очередь получить информацию о местонахождении необходимых для этого k порций. Для этого центральный узел рассылает соответствующий запрос всем своим соседям.

Любой узел, получив впервые такой запрос, рассылает его всем своим соседям, кроме общих с узлом, пославшим ему запрос. Если он содержит требуемую порцию, также отсылает сообщение об этом по обратному пути.

Такой выбор способа поиска обусловлен следующими предпосылками:

- местоположение порций заранее неизвестно (произвольное);
- части файла требуется найти за минимально короткое время.

Глубина поиска ограничивается характеристикой TTL (*Time-To-Live*) пакета (подробнее см. в работе [2]).

Регулярность структуры. Регулярной назовем распределенную систему, каждый узел которой соединен ровно с l соседями. В данной работе все примеры были выполнены для $l = 4$, т. е. для графов в виде решеток.

Регулярность не является необходимым для решения задачи ограничением. Оно было выбрано для упрощения расчетов как один из вариантов.

На рис. 1 изображен решетчатый граф такой распределенной системы. Ромбом ограничена область поиска для глубины 5. При такой структуре, например, на i-м шаге достигается $D_i = 4i$ новых узлов (периметр ромба). А число узлов, вовлечен-

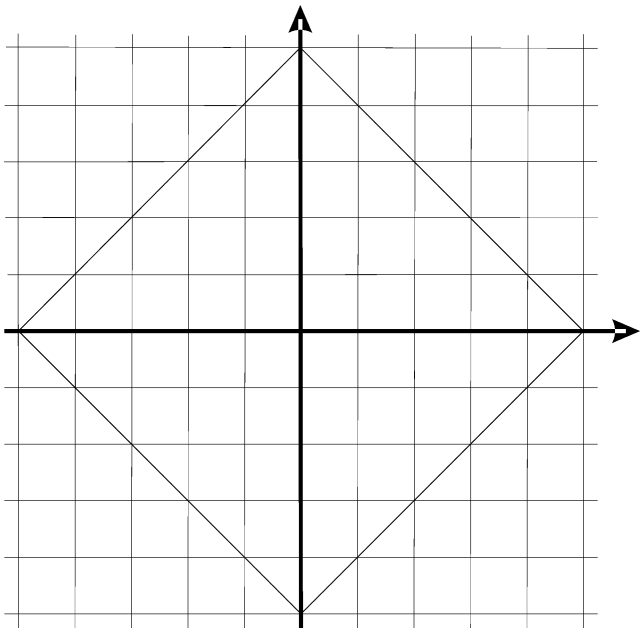


Рис. 1. Область поиска для глубины 5. Процедура запускается с узла в пересечении осей

ных в процедуру поиска, через i шагов равно B_i (считая, что ни один запрос не потерялся при передаче):

$$B_i = 1 + \sum_{j=1}^{i} D_j = 1 + 2i(i+1).$$

Заметим, что общее число узлов в окрестности поиска глубины d равно $B_d = 1 + 2d(d+1)$.

Точечная загруженность. Цель данной работы — изучить влияние загруженности соединений распределенной системы на длительность поиска. Предполагается следующий сценарий: на выбранных заранее компьютерах появляется актуальная информация, которая "притягивает" произвольно расположенных пользователей, повышая загруженность прилежащих соединений. Степень актуальности хранящейся на выбранном компьютере информации определяет число поступающих запросов, а следовательно, и прирост загруженности. Каждая порция актуальной информации увеличивает загруженность всех соединений системы на величину, убывающую в геометрической прогрессии в зависимости от их удаленности от выбранного компьютера.

Увеличение загруженности соединения выражается в пропорциональном росте длительности передачи данных по нему. Приращения загруженности, вызванные двумя различными узлами с актуальной информацией, суммируются.

На рис. 2 показано, как прирост загруженности снижается с расстоянием. Максимум R (рис. 2) соответствует числу запросов, приходящих на узел с актуальной информацией. На каждого из ближайших соседей будет приходиться в среднем в l раз (см. раздел "Регулярность структуры". Здесь $l = 4$) меньше запросов и т. д. Полная картина загруженности сети будет представлять собой несколько таких "горок" для каждого узла с актуальной информацией.

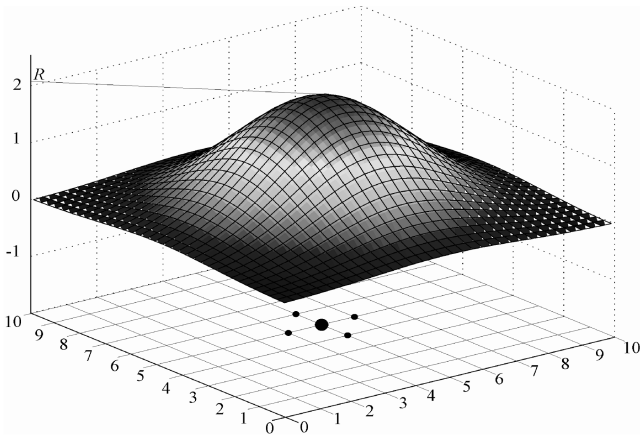


Рис. 2. Распределение загруженности в зависимости от удаленности от источника — узла с координатами (5, 5, 0). Он и его ближайшие соседи выделены точками

Имитационное моделирование

Вероятностная модель. Многие факторы в модели носят случайный характер, поэтому при описании происходящего была выбрана вероятностная модель. В ней параметры узлов и соединений системы представляются случайными величинами (рис. 3) [3]:

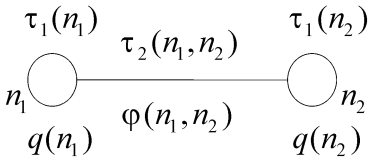


Рис. 3. Обозначения

- $\tau_1(n)$ — время обработки запроса (на сборку файла) на узле n;
- $\tau_2(n_1, n_2)$ — время передачи пакета с запросом между соседними узлами;
- $q(n)$ — число требуемых порций на машине n (считаем равномерно разбросанными);
- $\varphi(n_1, n_2) = \begin{cases} 1, \text{ если посылка пакета от } n_1 \text{ до } n_2 \text{ была успешной;} \\ 0, \text{ если при посылке пакета от } n_1 \text{ до } n_2 \text{ произошла ошибка (пакет не дошел до } n_2); \end{cases}$

$P\{\varphi(n_1, n_2) = 1\} = 0$ означает, что узлы n_1 и n_2 просто не соединены.

Сама длительность поиска также представляется случайной величиной. Требуется найти ее среднее значение. Оно определяется следующим образом: значение длительности вычисляется много раз как результат процесса моделирования сборки файла, случайные величины при этом генерируются каждый раз заново (для $\tau_1(n)$ и $\tau_2(n_1, n_2)$ используем нормальное распределение (один из вариантов), для $\varphi(n_1, n_2)$ — биномиальное). В результате мы получаем выборку искомой величины. Выборка дает полную информацию о случайной величине, в том числе и ее среднее значение. Подробнее о процессе определения длительности поиска см. в работе [2].

Математическая постановка. Ставится задача изучить поведение длительности поиска в регулярной распределенной системе в стрессовой ситуации — при возрастании числа запросов к актуальной информации.

Во введенных обозначениях это будет выглядеть следующим образом. Перенумеруем все узлы распределенной системы: 1, 2, ..., M. Пусть дан набор узлов с актуальной информацией: $i_1, i_2, \ldots, i_r$, где $\forall j,\ i_j \leqslant M$. И каждому такому узлу соответствует число запросов к нему, растущее со временем:

$$i_j \to R_j(t) = \alpha_j t.$$

Требуется найти зависимость среднего значения длительности поиска как случайной величины от времени $T(t)$.

Способ решения. Для решения поставленной задачи в рамках вероятностной модели необходи-

мо уметь получать длительность поиска T по входным параметрам. Для структуры распределенной системы, состоящей из узлов и схемы связей, входными параметрами будут:

- величины $\tau_1(n)$ и $q(n)$ для всех узлов, $\tau_2(n_1, n_2)$ и $\varphi(n_1, n_2)$ для всех соединений;
- параметры N и k схемы хранения файлов в распределенной системе;
- глубина поиска d;
- величина тайм-аута на операцию посылки запроса соседу, обработки его там и отправления ответа обратно;
- центральный узел, на котором запускается процедура поиска.

Сразу заметим, что случайными параметрами, т. е. генерируемыми многократно для получения выборки T, являются только величины $\tau_1(n)$ и $q(n)$ для всех узлов, $\tau_2(n_1, n_2)$ и $\varphi(n_1, n_2)$ для всех соединений.

Можно было бы составить математическое выражение зависимости длительности поиска от этих параметров: $T = f$(_параметры_) (см. [2]). Но оно было бы слишком сложным, чтобы напрямую получить из него функцию распределения T, зная распределения входных величин. А для вычисления значения T по значениям перечисленных параметров гораздо более естественным и точным способом является имитационное моделирование поиска.

При этом:

- посылка запроса соседу заключается в генерации значений случайных величин $\tau_1(n)$ и $\tau_2(n_1, n_2)$.
- каждый узел ищет минимальное время получения им запроса, определяя таким образом кратчайший (на текущий момент) путь до него;
- если узел содержит требуемую порцию файла, сразу "отсылает" сообщение обратно центральному узлу по кратчайшему пути;
- каждая пересылка данных между соседними узлами может завершиться неудачей с вероятностью $P\{\varphi(n_1, n_2\} = 0\}$;
- узел ожидает ответа от своих соседей только в течение заданного времени (тайм-аута). Все присланные по его истечении ответы игнорируются.

Процесс моделирования схематично изображен на рис. 4.

Процедура поиска может завершиться неуспешно, если:

- в окрестности поиска не окажется необходимого числа порций искомого блока информации;
- необходимое число порций не удастся обнаружить. Ведь если, например, при пересылке пакета с запросом от любого соседа узла с порцией произойдет ошибка, то эта порция останется невыявленной;

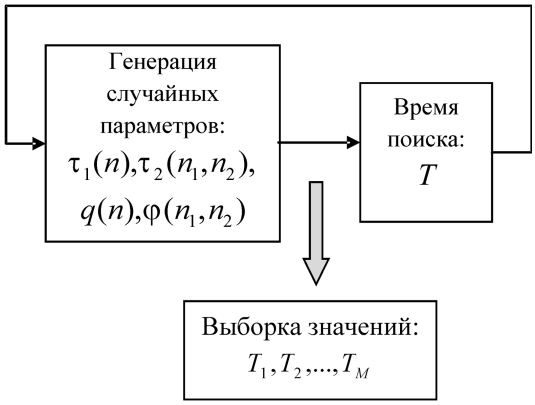


Рис. 4. Схема моделирования

- необходимое число ответов не дойдет до центрального узла.

По дороге пакет с ответом также может пропасть.

В этом случае длительность поиска принимается равной нулю.

Теоретическая оценка

Новая модель. Для того чтобы оценить правдоподобность получаемых имитационных моделированием результатов, было решено вычислить ту же самую зависимость средней длительности поиска T от времени другим способом — в рамках новой модели. Вторая модель также использует теорию вероятности, однако:

- длительности передачи данных между соседними узлами и обработки запроса постоянны:

$$\tau_2'(n_1, n_2) = E(\tau_2(n_1, n_2)) = \text{const};$$

$$\tau_1'(n) = E(\tau_1(n)) = \text{const};$$

- пересылка пакета всегда успешна:

$$\varphi(n_1, n_2) = 1 \ \forall (n_1, n_2);$$

- порции файла равномерно разбрасываются по узлам распределенной системы.

Таким образом, вероятностный характер новой модели придает именно разброс порций по узлам сети. Он будет осуществляться генерацией значений случайного вектора $\Psi^N = \begin{Vmatrix} \Psi_1 \\ \dots \\ \Psi_N \end{Vmatrix}$, где компоненты $\{\Psi_i\}_{i=1}^N$ — одинаково распределенные случайные величины, принимающие значения 1, ..., M с равной вероятностью $1/M$. Значение величины Ψ_i означает номер узла, на котором лежит i-я порция файла. Определим конкретнее вероятностное пространство вектора Ψ^N (будет использоваться в следующем разделе): множество элементарных

исходов Ω — всевозможные векторы с целыми компонентами из [1, M], алгебра событий — все подмножества Ω, вероятность определена классическим образом.

Расчет длительности поиска. Пусть $\theta(n)$ — время достижения запросом узла n (т. е. время получения им первого пакета с запросом). Благодаря постоянству $\tau_1(n)$ и $\tau_2(n_1, n_2)$ длительность поиска здесь может принимать только дискретные значения, равные $T_n = 2\theta(n)$. Перенумеруем M узлов распределенной системы по возрастанию T_n:

$$T_1 \leqslant T_2 \leqslant ... \leqslant T_i \leqslant T_M$$

и сгруппируем рядом стоящие узлы с одинаковыми T_n в отдельные множества Q_1, ..., Q_r, $r \leqslant M$:

$$Q_i = \{j | T_j = T_i'\}$$

$$\forall i,\ \forall j_1 \in Q_i,\ \forall j_2 \in Q_{i+1}: T_{j_1} < T_{j_2}.$$

Тогда вероятность для длительности поиска T, равной T_i', $P\{T = T_i'\}|_{i=1}^{r}$, будет равна вероятности найти последнюю из необходимых, k-ю порцию файла на одном из узлов из множеств $Q_i|_{i=1}^{r}$ соответственно.

Итак, T — случайная величина, принимающая дискретный ряд значений T_i' с вероятностями $P\{T + T_i'\}$. Поэтому математическое ожидание $E(T)$ будет определяться выражением

$$E(T) = \sum_{i=1}^{r} T_i' P\{T = T_i'\}. \qquad (1)$$

Осталось только найти $P\{T = T_i'\}$.

Обозначим S_i область поиска, покрываемую за T_i': $S_i = \bigcup_{1}^{i} Q_j$. Пусть также для общности $Q_0 = S_0 = \varnothing$ (пустое множество). Рассмотрим набор событий

$$A_i = \bigcup_{j=0}^{k-1} (\{B\ S_{i-1}\ \textit{содержатся}\ j\ \textit{порций}\} \cap \cap \{Q_i \text{ содержит } \geqslant k - j \text{ порций}\}).$$

Докажем ряд утверждений относительно этих событий.

Утверждение 1. Все события A_1, A_2, ..., A_r независимы.

Доказательство. Покажем, что $\forall i \forall q: q > i\ A_i \cap \cap A_q = 0$, это докажет требуемое утверждение. Действительно, во всех элементарных исходах из множества A_q, $q > i$, узлы из $S_{q-1} = \bigcup_{1}^{q-1} Q_j$ содержат не более $k - 1$ порций. Поскольку $q > i$, то $S_i \subseteq S_{q-1}$. Но множество A_i содержит только такие элементарные исходы, для которых $S_i = S_{i-1} \cup Q_i$ содержит не менее k порций. Поэтому A_i и A_q никак не могут иметь общих элементарных исходов, т. е. не пересекаются.

Утверждение 2. $A_1 \cup A_2 \cup ... \cup A_r = \Omega$ (Ω — пространство всех элементарных исходов).

Доказательство. Для любого элементарного исхода из Ω (разброса N порций среди всех узлов) рассмотрим число порций на узлах из множеств S_i. Поскольку $S_{i-1} \subset S_i\ \forall i = 2,..., r$, то S_i содержит не меньше порций, чем S_{i-1}. Самое крупное множество узлов S_r содержит все N порций. Поэтому всегда найдется такой номер $2 \leqslant i \leqslant r$, для которого S_{i-1} содержит все еще не более $k - 1$ порций, а S_i содержит уже не менее k порций. Это как раз и означает, что выбранный нами элементарный исход принадлежит A_i. Иными словами,

$$\forall \omega \in \Omega\ \forall i: \omega \in A_i,$$

что и требовалось доказать.

Утверждение 3. Длительность поиска T может быть равна T_i' в том и только том случае, если в S_{i-1} содержатся $j \leqslant k - 1$ порций искомого файла и Q_i содержит не менее $k - j$ порций, т. е. идентичны события $\{T = T_i'\} = A_i$.

Доказательство. Поскольку для сборки файла необходимо не менее k порций, то условие "в S_{i-1} содержатся $j \leqslant k - 1$ порций искомого файла" гарантирует, что поиск не закончится раньше T_i', при этом "Q_i содержит не менее $k - j$ порций" обязывает его закончиться на одном из узлов множества Q_i. Длительность в этом случае будет равна T_i'. И наоборот, $T = T_i'$ означает, что поиск завершился на одном из узлов Q_i, не раньше.

Утверждение 4. Вероятности $P(A_i)$ определяются формулами

$$P(A_1) = \sum_{j=k}^{N} C_N^j \left(\frac{|Q_1|}{M}\right)^j \left(1 - \frac{|Q_1|}{M}\right)^{N-j}; \qquad (2)$$

$$P(A_i) = \sum_{j=k}^{N} C_N^j \left[\sum_{l=0}^{k-1} C_j^l \left(\frac{|S_{i-1}|}{M}\right)^l \left(\frac{|Q_i|}{M}\right)^{j-l}\right] \times \times \left(1 - \frac{|S_{i-1}| + |Q_i|}{M}\right)^{N-j}, \qquad (3)$$

$$2 \leqslant i \leqslant r - 1;$$

$$P(A_r) = \sum_{l=0}^{k-1} C_N^l \left(\frac{|S_{r-1}|}{M}\right)^l \left(\frac{|Q_r|}{M}\right)^{N-j}. \qquad (4)$$

Здесь C_N^k — биномиальный коэффициент.

Доказательство. Покажем, откуда берется формула (3). Формулы (2) и (4) — лишь ее частные случаи, расписывающие неопределенности типа 0^0:

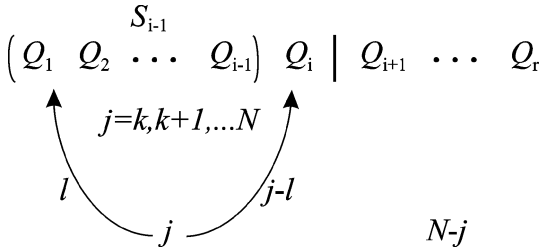


Рис. 5. Распределение порций в случае события A_i

ведь $|S_0| = \left(1 - \frac{|S_{r-1}| + |Q_r|}{M}\right) = 0$. Они получатся из (3), если считать, что $0^0 = 1$.

В формуле (3) для $P(A_i)$ по j суммируются вероятности разбить все множество узлов на две группы: $S_i = S_{i-1} \cup Q_i$ и остальные узлы так, чтобы S_i содержало $j \geqslant k$ порций.

Причем эти j порций должны быть распределены следующим образом: $l \leqslant k - 1$ порций в множестве S_{i-1} и остальные $j - l$ порций в множестве Q_i, за что и отвечает внутреннее суммирование по l (рис. 5).

Итак, утверждения 1—3 позволяют нам искать $P(A_i)$ вместо $P\{T = T_i'\}$, а утверждение 4 дает числовые выражения для них. Поставив их в (1), получим среднюю длительность поиска в рамках новой модели.

Расчет средней длительности поиска заключается в вычислении вероятностей по формулам (2)—(4) и кратчайших времен достижения узлов $\theta(n)$ по алгоритму Дейкстры.

Результаты

Представленные модели были проверены на конкретном примере. Для расчетов была выбрана решетчатая распределенная система из 121 узла (рис. 6). Параметры схемы хранения файлов: $N = 50$, $k = 25$. На трех узлах (6, 33 и 89) была "положена" информация, актуальность которой повышалась со временем по закону $R(t) = \alpha_i t$, где $\alpha_i = 200$, 500 и 300 соответственно. Шаг по времени, с которым пересчитывались средние значения длительности поиска, равен 1 с. Среднее время передачи данных между соседними узлами без учета загруженности равно 5000 мс.

Графики получившихся зависимостей средней длительности поиска от времени представлены на рис. 7 (см. вторую сторону обложки). Видно хорошее совпадение графиков разных моделей при небольшом росте загруженности (для этого вероятность успеха пересылки данных между соседними узлами в имитационной модели была взята равной 1 для всех соединений). Начиная с некоторого момента ($t \approx 30$ с), на имитационной кривой начинает сказываться рост неудачно завер-

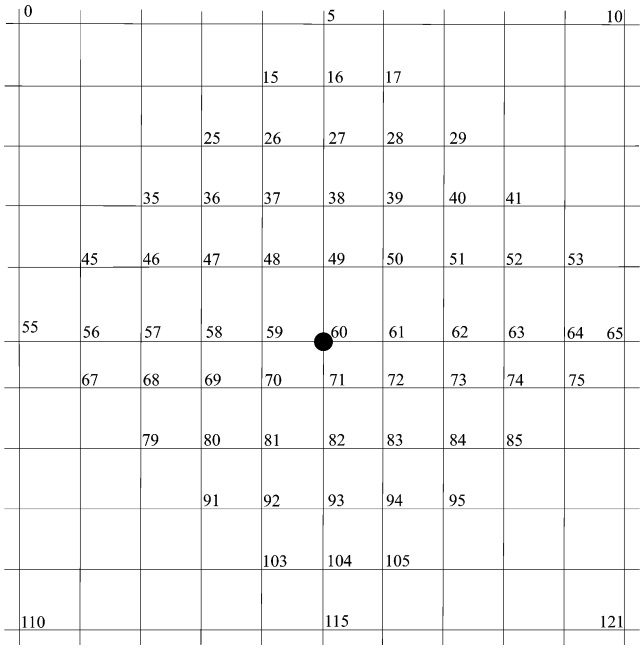


Рис. 6. Структура распределенной системы

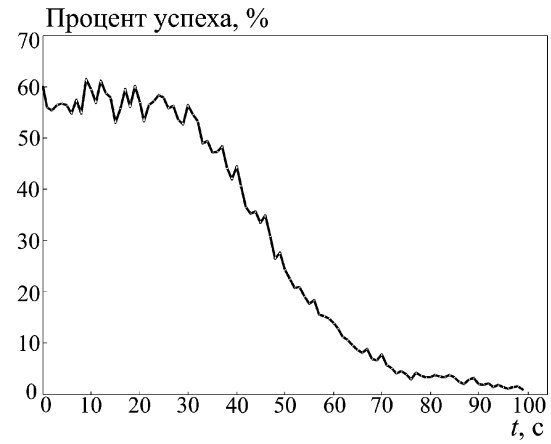


Рис. 8. Процент успешного завершения процесса поиска

шившихся процедур поиска: при некотором критическом уровне загруженности ответы присылаются узлам после истечения соответствующих тайм-аутов и поэтому игнорируются (см. раздел "Имитационное моделирование"). В этом убеждает график на рис. 8, также рассчитанный при имитационном моделировании. Лавинообразное падение вероятности успешного завершения процесса поиска также начинается примерно с 30 с.

Во второй модели тайм-ауты не учитываются, поэтому она дает строго возрастающую прямую.

Как мы видим, имитационная модель более гибкая. Она ближе к реальной ситуации, позволяет учесть больше параметров. Вторая модель важна для возможности сравнить, проверить результаты имитации. Неучет тех же тайм-аутов показывает, каких результатов можно было бы достичь при их увеличении. Ценное преимущество второй

модели также в гораздо (на порядки) более быстром расчете средней длительности по ней.

Помимо прочего, имитационная модель позволяет проследить зависимость средней длительности поиска от входных параметров. Например, на рис. 9 (см. вторую сторону обложки) графики изменения средней длительности поиска со временем в имитационной модели изображены при разных вероятностях успешной передачи данных между соседними узлами.

Как видно, средняя длительность поиска вообще очень чувствительна к этому параметру. Разница вероятностей в 0,02 демонстрирует существенные различия в графиках. При $p = 0{,}92$ график практически нечувствителен к изменению загруженности: вероятность успешной сборки файла крайне низка даже без нее.

Заключение

Был сделан обзор существующих методов поиска в распределенных системах, также приведены основные разновидности их моделей.

В классе рассматриваемых распределенных систем (использующих $N - k$ схему при хранении), в предположении неизвестности местоположения порций искомого файла, мы вынуждены искать их обычным поиском в ширину. Для такого поиска была рассчитана длительность поиска в условиях возрастающей загруженности сети. Расчет проводился в рамках двух различных моделей. Получившиеся результаты дают возможность в первую очередь сравнить модели: первая, имитационная, более приближена к реальной ситуации, поскольку учитывает больше реальных факторов, вплоть до индивидуальных параметров для каждого соединения и узла распределенной системы; вторая модель дает возможность проверить результаты имитационного моделирования, ее преимуществом является скорость расчета. Фактически, предпочтение одной модели другой — это выбор между точностью и скоростью получения результата.

Сам график изменения длительности поиска при повышении загруженности сети дает возможность оценить скорость роста длительности, а также пороговую загруженность, при которой начинается лавинообразное падение вероятности успеха процедуры поиска.

Список литературы

1. **Хасин М. А.** Модель распределенного хранилища в глобальной сети // Работа на соискание степени канд. физ.-мат. наук. М.: МФТИ, 2001.
2. **Петров В. А., Тормасов А. Г.** Доставка информации пользователю в распределенных системах // Проблемы вычислительной математики, математического моделирования и информатики: Сб. научн. тр. М.: МФТИ, 2006.
3. **Чернова Н. И.** Теория вероятностей: Курс лекций, http://www.nsu.ru/mmf/tvims/chernova/tv/portr.pdf.
4. **Rowstron A., Druschel P.** Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. http://research.microsoft.com/~antr/PAST/pastry.pdf.
5. **Pastry,** http://en.wikipedia.org/wiki/Pastry_%28DHT%29.
6. **Ramasubramanian V., Sirer E.** Beehive: O(1) Lookup Performance for Power-Law Query Distributions in Peer-to-Peer Overlays, http://www.cs.cornell.edu/People/egs/papers/beehive.pdf.
7. **Ghemawat S., Gobioff H., Leung S.-T.** The Google File System, http://labs.google.com/papers/gfs-sosp2003.pdf.
8. **A Distributed** Infrastructure for e-Business — Real Benefits, Measurable Returns — Akamai White Paper, October 2001, www.akamai.com.
9. **Бежитский С. С., Семенкин Е. С.** Эволюционные алгоритмы для автоматизации проектирования распределенных систем обработки информации и управления, http//raai.org/resurs/papers/kii-2006/seminar/Bezhitskiy.doc.
10. **Бежитский С. С.** Выбор оптимальной структуры аппаратно-программного комплекса системы управления движением автомобильного транспорта // Вестник университетского комплекса: Сб. научных трудов. Вып. 6 (20). Красноярск: ВСФ РГУИП, НИИ СУВПТ. 2005.
11. **Семенкин Е. С., Лебедев В. А.** Методы обобщенного адаптивного поиска для синтеза систем управления сложными системами. М.: МАКС Пресс, 2002.
12. **Ломазова И. А.** Вложенные сети Петри и моделирование распределенных систем, http://www.botik.ru/PSI/disk_20/e-book/e-book/1-4/02-Lomazova-Vlozhennye-seti-p-337.pdf.
13. **Котов В. Е.** Сети Петри. М.: Наука, 1984.
14. **Smith E.** Principles of high-level net theory // Lectures on Petri Nets: Advances in Petri Nets Lecture Notes in Computer Scince. Berlin; Heidelberg: Springer, 1998. P. 174—210. http://www.springerlink.com/content/78125201814583v8.

УДК 004.738.52

А. Ю. Орлов, ведущий инж.-конструктор,
НПК "Маджента Девелопмент",
А. В. Иващенко, канд. техн. наук, доц.,
руководитель проектов,
Самарский государственный
аэрокосмический университет

Организация виртуального сообщества в сети Интернет

Описана методика организации виртуальных сообществ пользователей (по интересам) в сети Интернет с использованием новых информационных технологий. Рассматриваются вопросы изучения, поддержки и управления виртуальным сообществом как сложной самоорганизующейся системой. Приведен пример организации виртуального сообщества в сети Интернет на основе онлайн браузера.

Ключевые слова: *Интернет, социальная сеть, виртуальное сообщество, онлайн-браузер.*

Введение

Лавинообразный рост популярности социальных сетей, наблюдаемый в настоящее время в сети Интернет, привлекает внимание различных исследователей, занимающихся информационными технологиями и социологией. У многих на устах такие громкие имена, как Flickr, Youtube, LiveJournal, LinkedIn, Facebook, а также их отечественные аналоги Яндекс.Фотки, RuTube, vContacte и многие другие.

Сами по себе социальные сети являются частным случаем виртуальных сообществ [1], исследованию которых посвящено много работ социологической направленности [2]. В частности, существует утверждение о том, что "виртуальные сетевые сообщества относятся к классу саморегулирующихся и самопреобразующихся социальных структур" [3]. Действительно, при большом числе пользователей виртуальные сообщества представляют собой сложные системы, а при объединении образуют новый организм, обладающий, в свою очередь, свойствами самоорганизации [4, 5].

Изучение этого явления и разработка механизмов поддержки и управления виртуальными сообществами весьма актуальны в современном мире. В частности, такие механизмы могут быть интересны, например, различным рекламодателям, ведь блогосфера сейчас является очень заманчивым и прибыльным местом [6].

Несмотря на то, что виртуальные сообщества могут существовать в контексте разнообразных телекоммуникационных технологий, сосредоточим наше внимание на рассмотрении виртуальных сообществ в сети Интернет в связи с высоким их распространением и большими техническими возможностями их организации.

Пользователям сети Интернет необходима возможность общаться и решать повседневные задачи наиболее комфортно и эффективно. Для удовлетворения этой потребности требуется обеспечить функциональность системы поддержки виртуальных сообществ, которая в идеале могла бы быть связующим звеном между всеми социальными сетями пользователя. Некоторым шагом в этом направлении является SocialStream, финансируемый Google [7], однако в полном объеме данная задача пока не решена.

При исследовании виртуальных сообществ в сети Интернет можно поставить задачи изучения, управления и поддержки, которым и посвящена данная статья.

Основные понятия

Ограничим область исследования *всемирной паутиной*, оставив без рассмотрения иные способы общения пользователей сети Интернет [8]. В этом случае объектом исследования будет *пользователь* — человек, возможно, имеющий один или несколько профилей в разнообразных социальных сетях, с учетом того, что каждая сеть не может изучаться независимо, а только в контексте других существующих сетей [9]. Соответственно, каждого пользователя следует рассматривать только в контексте всей деятельности, которую он производит в режиме online.

В связи с этим уточним понятие виртуального сообщества. Английское слово *virtual* несколько сложно перевести точно и емко на русский язык. В отличие от классических определений виртуальных сообществ [3], под *виртуальным сообществом* пользователей определим некоторую возможную группировку пользователей по определенному критерию. При этом не обязательно, чтобы между этими пользователями существовала какая-либо явно заданная связь (вроде дружбы или знакомства). Введенное понятие виртуального сообщества является более широким и позволяет рассматривать новые виды явлений.

Сказанное не означает, однако, что мы не принимаем к рассмотрению явно заданные отношения между пользователями. Такие отношения будем рассматривать как ребра ориентированного графа, вершины которого — пользователи, а ребра — связи. Будем называть такое отношение *заинтересованностью*, если оно носит односторонний характер, и *дружбой* в случае двухсторонней связи.

Определим также информационную единицу, информационный объект, который пользователи могут просматривать, комментировать и т. д.

При доступе к любому ресурсу всемирной паутины браузер в соответствии с протоколом HTTP [10] передает серверу тип запроса (GET, POST, HEAD и т. д.), запрашиваемый URL, набор различных HTTP заголовков и, собственно, тело запроса. Очень популярная в наши дни технология AJAX [11] и волна появившихся сейчас сайтов "web 2.0" [12] привели к тому, что браузер пользователя обычно посылает несколько запросов на сервер для того, чтобы отобразить страницу, независимо подгружая информационные блоки, из которых страница составлена.

Вместе с тем, как на уровне разработки и поддержки, так и с точки зрения пользователей потребляемая информация ассоциируется со страницей, отображаемой браузером. На наш взгляд, это связано с субъективным восприятием страницы, которая с точки зрения конечного пользователя представляет собой не набор подгружаемых в разные моменты времени фрагментов, а единый динамически меняющийся объект.

В связи с этим в качестве *информационной единицы*, или *информационного объекта*, наиболее логичным представляется выбрать страницу всемирной паутины, доступ к которой осуществляется путем открытия ее в окне браузера пользователя. Такая страница получается посылкой GET-запроса на сервер и может быть однозначно идентифицирована URL′ом.

Стоит отметить, что отдельные блоки на рассматриваемых нами страницах могут различаться при просмотре разными пользователями, т. е. зависеть от того, какой пользователь запросил страницу. Обычно такие блоки представляют собой или целевую рекламу или как-то связаны с профильными данными пользователя. Без ограничения общности будем считать такие страницы общими для всех пользователей, так как даже персональные блоки на этих страницах пользователи могут успешно обсуждать и оценивать.

Изменение содержимого просматриваемой страницы (как в случае, например, заглавной страницы новостного сайта) не означает появления нового информационного объекта. Обновление информации следует рассматривать как неотрывную часть процесса эволюции всемирной сети, а существующие связи между пользователями и страницами не следует моментально пересматривать — они сами трансформируются с течением времени в результате действий пользователя. Таким образом, до тех пор пока не ставится задача анализа содержимого страницы, можно игнорировать обновления в явном виде, продолжая под URL′ом, который не меняется, подразумевать сам информационный объект.

В связи с этим может возникнуть вполне обоснованный вопрос, насколько правомочно игнорировать все многообразие вариантов представления информации во всемирной паутине, и при этом следует сосредотачиваться только на независимых страницах. Действительно, излагаемый подход может быть с успехом экстраполирован и на вариант, когда объектом внимания становятся не страницы целиком, а их составные блоки. Примером реализации комментариев на такие блоки может служить [13], а ключом к экстраполяции может стать переход от URL к URI с XPath [14, 15].

При описании подхода будем также использовать понятия *тега* и *тегирования* страниц [16]. Пользователь может с любым информационным объектом проассоциировать набор тегов — произвольных слов. Обычно это имена существительные в именительном падеже, однако пользователь может в качестве тегов выбрать абсолютно любые слова. Тегирование применяется с целью организовать для пользователя окружающий мир и фактически отражает взгляд пользователя на этот мир.

Описанные выше понятия можно использовать при организации виртуального сообщества в сети Интернет.

Изучение виртуальных сообществ

Для системного изучения виртуальных сообществ сети Интернет построим модель, основанную на количественных характеристиках действий пользователей. В качестве исследуемого набора действий выберем:

- посещение информационного объекта (регулярное или разовое);
- голосование (за или против, либо оценка по некоторой шкале);
- обсуждение информационного объекта с другими пользователями;
- тегирование.

Приведенные действия отражают внутренний интерес пользователя, который меняется со временем. Отметим, что в ходе своей активности пользователь может иметь несколько параллельных интересов. Судить о внутреннем интересе можно по последовательности операций, которые совершает пользователь.

Проявление интереса пользователями может быть активным (связанным с размещением или переработкой информации, т. е. приводящим к изменению информационных объектов в результате активности пользователя) или пассивным (связанным только с потреблением информации).

Виртуальное сообщество имеет коллективный интерес, представляющий собой агрегацию частных интересов каждого входящего в него пользователя. В соответствии с этим интересом виртуальное сообщество обращается к одним и тем же информационным объектам глобальной сети: ходит на одни и те же сайты, обменивается однотипными сообщениями, использует специфический сленг и т. п.

Уточним сделанное выше определение виртуального сообщества на основе приведенных выше соображений. В качестве *виртуального сообщества* можно определить совокупность пользователей, у которых интересы в течение некоторого времени близки, т. е. они получают доступ к одинаковым или семантически связанным информационным объектам, размещенным на сайтах и представляющих интерес (т. е. информация является полезной или развлекательной).

В соответствии с различным проявлением интереса можно выделить два типа виртуальных сообществ:

- *аудитория* — совокупность пассивных читателей, которые в течение некоторого периода времени осуществляют неоднократный доступ на сайт;
- *содружество* — совокупность активных пользователей, которые самостоятельно создают и добавляют новые информационные объекты.

Стоит отметить, что в чистом виде ни одна из категорий в реальной жизни не встречается, а такая идеализация используется исходя из разных методов воздействия на две введенные категории.

Поддержка виртуальных сообществ

Перечисленные нами действия в популярных социальных сетях дополняются возможностью выложить на сайт набор своих фотографий, видеороликов, возможностью высказать свое мнение безотносительно какого-то информационного объекта (что является частным случаем рассматриваемой нами операции). Во всем многообразии социальных сетей каждая сеть имеет свою, иногда довольно узкую специализацию, в которой она является непревзойденным лидером:

- фотографии — Flickr, Slide;
- видеоролики — YouTube;
- связи между пользователями — LinkedIn, Facebook;
- горячие новости — Digg;
- хранение закладок — del.icio.us;
- личные дневники, тематические группы — LiveJournal, Blogger.

Большинство пользователей имеет по одной учетной записи в глобальной или локальной версии системы каждой из перечисленных направленностей (т. е. один аккаунт в Yandex.Фотки, один на RuTube и т. д.).

Для удобства использования нескольких учетных записей необходимо обеспечить пользователю единое представление гетерогенной сети источников информации. Для этого нужно интегрировать разнородные системы и обеспечить обмен информацией между ними с учетом отсутствия единого формата обмена данными.

Не менее серьезной проблемой является одновременная работа со всеми приложениями, в которых пользователь зарегистрирован. Как сделать так, чтобы каждая система не требовала свой независимый логин от пользователя? Как в качестве одной несложной операции добавить запись, прикрепив к ней фотографии, в свой сетевой дневник, выкладываемый на специализированный сервис, и видеофайлы, параллельно выкладываемые на свой специализированный сервис?

Для решения указанных проблем существует два класса систем:

- системы сборки (mashup) — сайты, позволяющие собрать в одном месте информацию из нескольких учетных записей из третьих систем [17];
- метабраузеры — надстройки над браузерами, интегрирующие в себя информацию из сервисов, а также предлагающие контекстные действия и информацию в соответствии с процессом навигации пользователя по сети [18].

С точки зрения авторов, именно второй подход является более перспективным для всемирной паутины, так как он модифицирует понятие навигации по сети Интернет (web surfing). В них пользователь может:

- в один клик сохранить просматриваемую ссылку в сети, присвоив ей теги;
- обсудить ее с другими пользователями;
- посмотреть, кто в данный момент просматривает ту же страницу;
- выполнить покупку, если просматриваемая страница содержит описание одного товара;
- найти описываемое место на карте;
- посмотреть, что есть в сети с содержанием, близким к просматриваемой странице, и т. д.

Инфраструктура действий и сервисов, которая строится вокруг процесса навигации по всемирной паутине, — наиболее современное и перспективное направление.

Согласно этой идее, в скором времени произойдет переход от классических устанавливаемых в операционную систему приложений метабраузеров к не требующим установки онлайн-браузерам. Последние же должны работать с любым браузером, встроенным в операционную систему пользователя.

Таким образом, при организации виртуальных сообществ сети Интернет в качестве инструмента поддержки сообщества можно использовать онлайн-браузер, пример которого, предлагаемый авторами, описывается ниже.

Управление виртуальным сообществом

Социальные сети и организующиеся на их основе виртуальные сообщества являются "лакомым куском" для рекламодателей, владельцев торговых марок, политиков и т. д. В связи с этим возникает задача управления существующими виртуальными сообществами.

Оставим за рамками данной статьи обсуждение приемлемости и этичности подобного рода управления. Заметим только, что в той или иной форме управление (а именно, политика ограничений) присутствует в большинстве организаций и учебных заведений, где общеиспользуемый прокси-сервер ограничивает доступ к определенным ресурсам.

Управление виртуальным сообществом, как и во множестве иных областей, может быть реализовано с применением схемы с обратной связью на верхнем уровне. При этом надо учитывать, что виртуальное сообщество по своей природе является сложной самоорганизующейся системой.

Управление виртуальным сообществом можно построить на основе мониторинга показателей интереса пользователей-членов путем автоматизированного воздействия следующим образом:

- непосредственным изменением концентрации пользовательского интереса путем включения новых пользователей, расширения состава сообщества и т. п.;
- сменой направленности сообщества путем создания новых источников интереса, мотивируя дополнительные волны обсуждения;
- искусственным исключением некоторых сайтов из ядра сообщества;
- созданием новых технических средств, обеспечивающих повышение взаимодействия между участниками и, тем самым, повышением концентрации интереса.

Управление сообществом может проводиться путем прямого изменения состава пользователей и сайтов либо инструментов для общения. Пользователи сами определяют состав обсуждаемых сайтов, поэтому вхождение нового пользователя, отсоединение старого пользователя, нахождение пользователя в составе сообщества нового сайта приводят к пересмотру состава сайтов, что является дестабилизирующим эффектом. Такой эффект может привести к полному разрушению сообщества (в силу потери интереса пользователей) либо к мутации интереса.

Для получения качественно новых сообществ можно стимулировать потерю интереса, а для сохранения состава сообщества можно искусственно поддерживать старый интерес. В этом случае одной из возможных методик является моделирование деятельности исчезнувших пользователей в целях постепенной переориентации всего сообщества.

Следует отметить, что на практике чрезвычайно сложно контролировать все действия пользователей. Так или иначе часть их не будет учитываться при принятии решений, что является дополнительным неопределенным входным воздействием.

В связи с различным поведением пользователей разных видов сообществ существуют различия и в структуре их жизненного цикла. На первом этапе жизненного цикла сообщества необходимо создать набор информационных объектов и предоставить всем (или ограниченному кругу) пользователей глобальной сети доступ к этим объектам. Поскольку виртуальное сообщество существует только в динамике, необходимо обеспечить постоянное обновление этих информационных объектов, тем самым, поддерживая интерес группы пользователей к сайту.

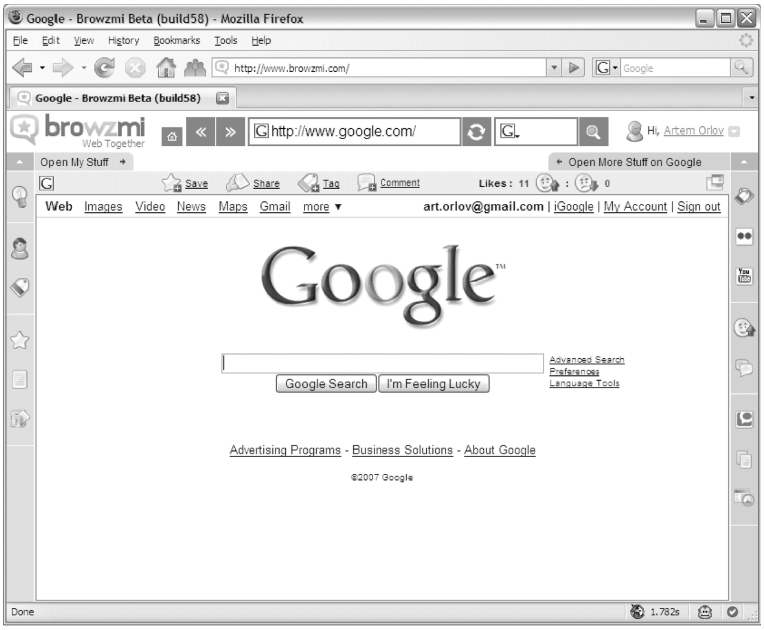
Далее возможно два пути развития. Если необходимо поддерживать аудиторию, достаточно создания удобного и быстрого средства обновления информационных объектов. Если необходимо создание содружества, в состав сайта нужно включить функциональность, инициирующую не только обратную связь, но и позволяющую вести виртуальное обсуждение материала сайта.

Стоит отметить, что виртуальные сообщества относятся к классу открытых систем, поэтому к ним не применимы аналоги законов сохранения энергии, так как исключительно сложно подсчитать "приток" энергии и ее "расход" вовне, ведь участники виртуального сообщества живут еще и реальной жизнью за границами сети.

Организация виртуального сообщества сети Интернет на примере реализации онлайн-браузера

Описанные выше концепции легли в основу онлайн-браузера (см. рисунок), осуществляющего поддержку и управление виртуальным сообществом пользователей сети Интернет. Целевой аудиторией системы являются англоязычные пользователи с достаточно разнообразными интересами, преимущественно молодые и среднего возраста, активно использующие сеть Интернет в развлекательных целях. В настоящее время этот сайт доступен лишь ограниченной группе пользователей (около 1000), однако в ближайшем будущем планируется открытие широкого доступа.

Следует отметить, что данная группа пользователей является достаточно сложной при организации виртуальных сообществ, поскольку, с одной стороны, обладает большой активностью и высоким интересом ко всему новому, с другой



Просмотр сайта средствами онлайн-браузера

стороны, содержит пользователей, не связанных между собой в реальном мире и общающихся исключительно на основе общего интереса.

В основе разработанного онлайн-браузера лежит возможность просматривать различные сайты, сохранять закладки, описывать их с помощью тегов и создавать статьи (posts), в которых рекомендовать эти сайты другим пользователям. Таким образом, виртуальное сообщество пользователей сети Интернет построено вокруг активного совместного веб-серфинга, а в основе его организации лежит максимально удобный и "забавный — cool" пользовательский интерфейс.

Опишем, как адресуются описанные выше концепции управления сообществом в приведенном онлайн-браузере:

- при потере пользовательского интереса возможно обращение к странице, отражающей всю деятельность пользователей;
- использование сайтов и страниц, пользователей, статей и т. п., близких к просматриваемой странице в настоящее время, позволяет поддержать концентрацию пользовательского интереса, так как пользователь всегда имеет возможность расширить число друзей или интересных ему сайтов;
- добавление ссылок на сайты, близкие к заданному, позволяет предлагать пользователям новые информационные объекты, изменять их интерес и мотивировать новые обсуждения;
- отслеживание активности всех пользователей позволяет идентифицировать информационные объекты, к которым проявляется повышенный интерес, и на основе анализа их содержания использовать этот факт в рекламных целях либо в целях поддержки сообщества;
- выбор наиболее близких сайтов проводится на основе поиска по тегам, однако результирующий набор может быть подвержен модификации;
- взаимодействие членов сообщества обеспечивается на любой странице онлайн-браузера, а обмен информацией осуществляется весьма легко.

Заключение

Описанные в данной статье подходы представляют собой лишь возможную методику организации виртуального сообщества как открытой сложной самоорганизующейся системы.

За рамками статьи остались вопросы формализации структуры, процессов эволюции виртуальных сообществ сети Интернет, описание математической модели и алгоритмов управления сообществом. Однако приведенный материал позволяет определить новые подходы к организации и управлению виртуальными сообществами и иллюстрирует применение их на практике.

Список литературы

1. http://en.wikipedia.org/wiki/Virtual_community
2. **Бондаренко С. В.** Жили они долго?!. и... // Компьютерра, 2004. http://offline.computerra.ru/offline/2004/529/32074/
3. **Бондаренко С. В.** Социальная структура виртуальных сетевых сообществ // Автореферат дисс. на соиск. уч. ст. докт. соц. наук, Ростов-на-Дону, 2004.
4. **Виттих В. А.** Эволюционное управление сложными системами // Известия Самарского научного центра РАН, 2000. Т. 2. № 1. С. 53—65.
5. **Эбелинг В., Файстель Р.** Хаос и космос. Синергетика эволюции. М.—Иж.: Институт компьютерных исследований; НИЦ "Регулярная и хаотическая динамика", 2005.
6. **Kivijarv L.** Podcast Advertising Gaining on Blogs // 2006. http://www.imediaconnection.com/content/9146.asp
7. **Socialstream** // Carnegie Mellon University HCII. http://www2.hcii.cs.cmu.edu/M-HCI/2006/Socialstream-Project/index.php
8. http://en.wikipedia.org/wiki/Internet
9. **Cai D., Shao Z., He X., Yan X., and Han J.** Mining hidden community in heterogeneous social networks. In Proceedings of the 3rd International Workshop on Link Discovery. ACM Press: Chicago, Illinois. 2005.
10. **RFC 2616** Hypertext Transfer Protocol — HTTP/1.1 http://tools.ietf.org/html/rfc2616
11. http://en.wikipedia.org/wiki/Ajax_%28programming%29
12. **Go2Web20.net** — The complete Web 2.0 directory http://www.go2web20.net/
13. **Fleck.com** — Fleck the Web! http://fleck.com/
14. **Uniform** Resource Identifier (URI): Generic Syntax http://www.ietf.org/rfc/rfc3986.txt
15. **XML** Path Language (XPath) 2.0 http://www.w3.org/TR/xpath20/
16. **Иващенко А. В., Минаков И. А., Орлов А. Ю.** Информационная поддержка сетевых сообществ на основе тегирования структур Интернет // Тр. IX Междунар. конф. "Проблемы управления и моделирования в сложных системах". Самара: СНЦ РАН, 2007. С. 549—555.
17. http://en.wikipedia.org/wiki/Mashup_%28web_application_hybrid%29
18. **Flock** — the social web browser http://www.flock.com/

УДК 004.658

В. А. Зелепухина,
ведущий программист отдела
internet-технологий,
Астраханский государственный университет

Разработка систем управления содержимым интернет-ресурсов на основе автоматической генерации web-интерфейса и SQL-запросов

Предлагается подход к разработке систем управления содержимым интернет-ресурсов, основанный на автоматической генерации web-интерфейса и SQL-запросов. Идея подхода — на основе формального описания каждого компонента модели данных и особенностей операций с данными сформировать информацию, необходимую для реализации конкретного web-интерфейса, и по запросу пользователя автоматически сгенерировать и обработать SQL-запрос к базе данных.

Ключевые слова: *интернет-ресурс, система управления содержимым, автоматическая генерация, web-интерфейс, SQL-запрос*

Введение

Современные системы управления интернет-ресурсами представляют собой полноценные приложения по обработке большого объема структурированных данных, хранящихся в СУБД (системы управления базами данных). Разработка таких систем с помощью традиционных средств является крайне трудоемкой задачей, так как любые изменения в структуре базы данных (БД) требуют модификации web-интерфейса и программного кода, включая SQL-запросы (*structured query language*). При этом видна типичность операций каждого модуля системы, работающего с БД, и достаточно определенным является набор визуальных компонентов его web-интерфейса.

Снизить стоимость разработки и упростить дальнейшую модификацию систем управления, работающих с большими объемами структурированных данных, можно за счет внедрения генераторов web-интерфейса и SQL-запросов. Рассматриваемая в статье технология создания подобных генераторов основана на замене процедурного кода декларативным описанием архитектурных компонентов web-интерфейса [1, 2].

Подходы к реализации генераторов web-интерфейса и программного кода

Проблемой автоматической генерации интерфейса и программного кода занимаются отечественные и зарубежные ученые: А. Н. Иванов, В. В. Грибова, Т. А. Гаврилова, О. Л. Перевозчикова, Э. В. Попов, П. И. Соснин, В. Ф. Хорошевский, Л. Л. Вышинский, P. Castells, J. Foley, M. Ivory, C. Janssen, J. Lowgren, B. Myers, A. Puerta, G. Singh, P. Sukaviriya, P. Szekely и др. В ходе проведенных исследований наметились два основных подхода к генерации интерфейса: дизайнерский и моделеориентированный [3—6]. Первый подход, используя технологию WYSIWYG, предлагает высокоуровневое проектирование интерфейса с автоматической генерацией кода. Несмотря на удобство применения, при использовании программного обеспечения, реализующего данный подход, требуется изменение сгенерированного кода вручную либо вследствие его избыточности, либо недостаточно реализованной им функциональности. За счет визуального проектирования web-интерфейса и автоматической генерации *html*-кода удается упростить разработку отдельно взятых модулей, но не всей системы в целом.

При втором подходе проектируется и сопровождается не web-интерфейс непосредственно, а его модель. Web-интерфейс и программный код генерируются автоматически по формальному описанию его модели. В отличие от дизайнерского, реализация данного подхода требует от разработчика либо знания существующих формализмов (UML), либо изучения специального декларативного языка для описания модели данных.

Концепция разработки систем управления содержимым на основе автоматической генерации web-интерфейса и SQL-запросов

Система управления содержимым web-сайта представляет собой типичную систему учета данных, которая обеспечивает хранение, просмотр и изменение информации в БД. Наличие четко определенных операций с данными (просмотр объектов и значений их атрибутов, добавление объектов, удаление объектов, изменение значений атрибутов) позволяет типизировать web-интерфейс и программный код с SQL-запросами. Применение моделеориентированного подхода к разработке такой системы позволит генерировать весь код "на лету" с использованием модели данных.

Модель данных. Особенности операций. Так как генератор рассчитан, в первую очередь, на работу с информацией, хранящейся в БД, то основой составления модели данных является модель классов: класс в реляционной БД представлен таблицей, а атрибуты класса полями таблицы. В каждой таблице может быть сгенерировано поле, являющееся ее первичным ключом и уникальным идентификатором для записей. Данный

идентификатор необходим для генерации интерфейса и запросов пользователя на удаление и редактирование существующих записей. При выполнении операции добавления новых записей значение идентификатора создается автоматически. Если редактируемая таблица из некоторого класса является зависимой от другого класса, то ссылка на класс-владелец представляется дополнительным полем, которое по своему смыслу является вторичным ключом.

Основные операции каждого модуля системы управления, работающего со структурированными данными, и соответствующие команды языка DML (язык манипулирования данными) следующие:

- *просмотр записей* (команда SELECT) с возможностью сортировки и фильтрации;
- *модифицирование выбранной записи* (команда UPDATE) с возможностью вывода отдельных полей для редактирования;
- *вставка новой записи* (команда INSERT); интерфейс подобен интерфейсу для модифицирования, при первоначальной загрузке страницы поля формы заполнены указанными в модели данных значениями по умолчанию;
- *удаление записей* (команда DELETE).

Вставку новых записей и модифицирование существующих разумно проводить посредством разработки единого генератора web-интерфейса с учетом некоторых особенностей.

1. *Идентификация типа операции*. Следует создать скрытое поле на *html*-форме, значение которого определяет тип операции (0 — вставка, 1 — модифицирование). Тогда в ходе дальнейшей обработки формы SQL-запрос будет формироваться в зависимости от данного значения (либо INSERT, либо UPDATE).

2. *Идентификация редактируемой записи*. В отличие от добавления новых записей для изменения существующих требуется наличие скрытого поля на web-странице с идентификатором редактируемой записи: его значение понадобится для генерации условия WHERE в запросе. При добавлении новых записей идентификатор создается автоматически.

Для внесения изменений в БД путем *добавления или редактирования записей* важно сгенерировать удобный для пользователя web-интерфейс. В случае связанных таблиц перебор записей в таблицах-владельцах и вывод их на страницу с использованием средств *html* (SELECT, RADIO) позволяет вводить значение в поле-ссылку не вручную, а выбирать из автоматически сгенерированного списка доступных значений.

При *удалении записей* следует учитывать случаи связанных некоторым отношением таблиц.

1. Классы K1 и K2 связаны агрегированием, где K1 является владельцем, а множественность роли K2 — "много".

2. Классы K1 и K2 связаны ассоциацией, где множественность роли K2 — "много", а K1 — "1".

3. Классы K1 и K2 связаны ассоциацией, где множественность роли K2 — "много", а K1 — "0...1".

В случаях 1 и 2 при удалении записей в таблице, соответствующей классу K1, происходит удаление всех записей из таблиц, соответствующих классу K2. В случае 3 в поле таблицы, соответствующей классу K2, которое является ссылкой на K1, будет проставлен 0.

Конфигурация моделей. С учетом описанных выше положений в модель данных следует включить следующие обязательные параметры.

1. *Параметры для операции удаления* (таблица-владелец с именем первичного ключа, список дочерних таблиц со ссылками на значения первичного ключа таблицы-владельца);

2. *Параметры сортировки*.

Указывается список полей, по значениям которых можно проводить сортировку при выполнении операции просмотра.

3. *Атрибуты класса*:

а) *владелец значения* в виде названия таблицы и имени столбца;

б) *значение по умолчанию*;

в) *тип данных*, используется для предварительной проверки введенных пользователем данных перед формированием запроса к БД;

г) *подпись* к атрибуту на web-странице (например, "Признак активности", "Категория", "Дата размещения" и т. д.);

д) *тип представления* на web-странице (text, textarea, select, radio, hidden и т. д.). Возможен вывод элементов не только с использованием стандартных возможностей *html*, но и путем создания новых пользовательских представлений. Это может быть визуальное представление даты на web-странице с использованием не текстового окна ввода, а календаря с выбором даты в виде *java*-апплета, либо интерактивное поле ввода в виде *flash*-объекта и т. д.

е) *источник значения* в виде названия таблицы, имени столбца с подписями и имени столбца со значениями. Является необязательным параметром и используется в случае связанных таблиц при множественности "один-ко-многим".

Обработка моделей. Web-интерфейс автоматически создается по его модели таким образом, чтобы его *html*-кода было достаточно для последующей генерации SQL (для составления интерфейса и запроса используется единая модель). Поэтому шаблон для элементов *html*-формы на генерируемой web-странице удобен следующий:

{подпись к элементу} < INPUT TYPE = {тип элемента} NAME = '{имя таблицы} {разделитель} {имя столбца}' VALUE = '{значение}' >

Источником переменных шаблона является соответствующая модель данных. Если web-интерфейс сгенерирован для добавления новой записи, то значение элемента формы является значением по умолчанию из соответствующей модели данных. Следующий листинг иллюстрирует применение указанного шаблона к генерации *html*-формы для редактирования записи.

```
<FORM METHOD = "POST" ACTION = "?mode =
3&act = edit&id = 1">
// Скрытое поле, указывающее на тип операции
(0 — добавление, 1 — редактирование)
<INPUT TYPE = "hidden" NAME = "saveType"
VALUE = "1">
// Скрытое поле, содержащее один из атрибутов модели класса — идентификатор редактируемой записи.
< INPUT TYPE = "hidden" NAME = "newsTexts$id"
VALUE = "9" >
<br> <b> Категория новости </b>
<SELECT NAME = "news$idCategory">
<OPTION VALUE = "1" SELECTED>
События </OPTION>
<OPTION VALUE = "2"> Конференции </OPTION>
<OPTION VALUE = "3"> Объявления </OPTION>
</SELECT>
<br> <b> Заголовок новости </b>
<INPUT TYPE = "text" NAME = "newsTexts$caption"
SIZE = "50" VALUE = "Упрощение процедуры получения субсидий">
<br> <b> Текст новости </b >
<TEXTAREA NAME = "newsTexts$text">
ПУСТО </TEXTAREA> <br> <INPUT TYPE = "submit" VALUE = "Сохранить">
```

Для обработки данной формы на сервер передается ассоциированный массив с именами элементов, соответствующими именам переменных *html*-формы, и со значениями, соответствующими значениям элементов той же формы. Используя данный массив (либо HTTP_POST_VARS, либо HTTP_GET_VARS, в зависимости от метода, указанного в заголовке формы), можно автоматически сгенерировать запрос к БД, не прибегая к какой-либо дополнительной информации. Реализуя цикл по HTTP_POST_VARS или HTTP_GET_VARS, необходимо выделить из имен его элементов названия таблиц БД и полей и сохранить в соответствующих массивах. Сделав данные массивы уникальными, можно сформировать непосредственно SQL-запрос в соответствии с его синтаксисом. Таким образом, процесс проходит три основные стадии: разбор модели на параметры; обработка параметров и выделение ключевых элементов; генерация запроса.

Итоги

Достаточно трудоемким является сам процесс написания обработчика моделей, по которым должен генерироваться web-интерфейс и программный код с использованием предлагаемого подхода. Однако временные затраты окупаются уже на стадии наполнения системы новыми модулями, не требующей больших усилий и кропотливой работы разработчика. Время разработки такой системы, в целом, сокращается в несколько раз, ее сопровождение намного упрощается. Основные преимущества систем, разработанных с применением такой технологии, очевидны — автоматическое изменение интерфейса без модифицирования его кода, возможность быстрого внедрения модулей в систему, возможность повторного использования компонентов, исправление ошибок на уровне обработчика моделей.

Разработка программного обеспечения с использованием предлагаемого подхода частично поддержана грантом Российского фонда фундаментальных исследований (*проект РФФИ № 07-07-00128-а*) и Фондом содействия развития малых форм предприятий в научно-технической сфере (проект У.М.Н.И.К. "Интернет-ориентированная кроссплатформенная система визуализации данных для сопряжения со специализированными пакетами научного программного обеспечения и базами данных"). Описанная технология применима не только к разработке систем управления интернет-ресурсами, но и для любого программного обеспечения, где требуется автоматическая генерация интерфейса и запросов к БД. В частности, подход использован для создания интерактивной системы визуализации научных данных.

Автор выражает благодарность д-ру физ.-мат. наук Тарасевичу Юрию Юрьевичу за полезные обсуждения и советы в процессе написания данной статьи.

Список литературы

1. **Иванов А. Н., Стригун С. А.** Технологическое решение REAL-IT: Моделирование и генерация пользовательского интерфейса // Системное программирование. 2004. С. 124—147.

2. **Грибова В. В., Клещев А. С.** Методы и средства разработки пользовательского интерфейса: современное состояние // Программные продукты и системы. 2001. № 1. С. 2—6.

3. **Грибова В. В., Тарасов А. В**. Инструментальное средство ONTO DEV для проектирования и автоматической генерации пользовательского интерфейса // Информатика и системы управления. 2006. № 1. С. 152—158.

4. **Гультяев А. К., Машин В. А.** Проектирование и дизайн пользовательского интерфейса. СПб.: КОРОНА принт, 2000. 352 с.

5. **Ceri S., Fraternali P., Bongio A., Brambilla M., Comai S., Matera M.** Designing Data-Intensive Web Applications // Elsevier Science (USA) 2003.

6. **Puerta A. R. I**ssues in Automatic Generation of User Interfaces in Model-Based Systems. 1996. http://smi-web.stanford.edu/projects/mecano/publicat.htm

УДК 614.8(043)

В. А. Немтинов, д-р техн. наук, доц.,
А. М. Манаенков, студент,
В. В. Морозов, студент,
К. В. Немтинов, студент,
Тамбовский государственный
технический университет

Технология создания пространственных моделей территориально распределенных объектов с использованием геоинформационных систем

Описывается технология создания пространственных моделей территориально распределенных объектов с использованием геоинформационных систем на примере объектов культуры, в частности, виртуального музея мемориальных комплексов, посвященных Великой Отечественной войне, которая включает в себя создание 3D-моделей объектов, базы атрибутивных данных для каждого комплекса и инструментария для их анализа.

Ключевые слова: *пространственные модели территориально распределенных объектов, геоинформационная система.*

Геоинформационные системы (ГИС) в настоящее время широко применяются во всем мире и России во многих областях знаний и промышленности. Это связано с тем, что ГИС становятся универсальной средой для интеграции самых различных информационных технологий и построения многофункциональных информационно-аналитических и управляющих систем. Одним из основных элементов таких систем является пространственная модель территории с включением в нее всех объектов, образующих единое информационное пространство, упорядоченно взаимодействующих друг с другом в процессах обмена информации и потребления материально-энергетических ресурсов.

В данной статье описывается технология создания пространственных моделей территориально распределенных объектов с использованием ГИС на примере объектов культуры. Технология включает в себя создание 3D-моделей объектов, базы атрибутивных данных для каждого комплекса и инструментария для их анализа.

При создании моделей в качестве базовой информационной системы используется ГИС, имеющая средства трехмерного моделирования, позволяющая построить пространственную модель отдельного комплекса, включающую все его объекты, а также объединить совокупность комплексов в единое целое — виртуальный музей. В качестве базового программного обеспечения использована ГИС ArcGis 9.2 корпорации ESRI. Системные средства ГИС позволяют объединить все комплексы виртуального музея по территориальному, тематическому или какому-либо другому признаку. Посетитель виртуального музея по собственному желанию может моделировать сценарий знакомства с объектами комплексов (экспонатами музея).

Пространственная модель отдельного комплекса включает в себя:

- трехмерное фотореалистичное изображение всех объектов с их географической привязкой;
- атрибутивную информацию различного назначения, связанную как с событиями военных лет и историей создания комплекса, так и персональными данными воинов, которым он посвящен.

Рассмотрим стадии создания музея, экспонатами которого являются модели ряда комплексов, размещенных на территории Тамбовской области.

Исходными данными для создания виртуальных копий реальных мемориальных комплексов являются:

- географическая карта Тамбовской области масштаба 1:200 000 с местоположением населенных пунктов;
- географические карты населенных пунктов различных масштабов с местоположением их основных объектов, в том числе и объектов мемориальных комплексов;
- планы размещения объектов отдельных комплексов масштабов 1:500 — 1:1000;
- фотографии объектов комплексов и их отдельных элементов, сделанные цифровым фотоаппаратом.

В качестве основы для создания электронных карт населенных пунктов и планов мемориальных комплексов могут быть использованы космические снимки местности с высоким разрешением или снимки аэрофотосъемки.

На начальном этапе создания электронного виртуального музея создаются 2D-виды карт (моделей) области, населенных пунктов и мемориальных комплексов. Переход от одних моделей к другим осуществляется с помощью функции "горячие связи" системы ArcGis 9.2.

В зависимости от сложности формы объекта для его описания может быть использован тот или иной графический примитив.

Точечные объекты — это объекты, расположенные только в одной точке пространства. Линейные объекты представляются как одномерные в координатном пространстве. Полигональные (площадные) объекты — это объекты, проекции которых на координатную плоскость *xoy* представляет собой области, аппроксимируемые многоугольниками. Так, например, промышленные объекты задаются в форме параллелепипедов или цилиндров:

$$O_i = xp_i \cdot yp_i \cdot zp_i,\ O_j = \pi \cdot xd_i^2 \cdot zp_i,\ i, j \in [1, ..., N],$$

где xp_i, yp_i, zp_i, xd_i — соответственно размеры объекта по каждой оси (xd_i — для объектов, проекция которых на плоскость задана в виде окружности); N — число объектов.

Для представления поверхностей чаще всего используется модель, называемая нерегулярной триангуляционной сетью (*triangulated irregular network* (*TIN*)) [1].

Помимо данных о геометрической форме объекта каждый из них может быть снабжен разнообразной атрибутивной информацией, хранящейся либо как отдельные таблицы внутри одной базы данных, либо как самостоятельные наборы данных, связанные набором указателей и объединенные в банке геоданных. Таким образом, каждая t_i-я точка 3D-модели мемориального комплекса может быть представлена следующим образом:

$$t_i = \{x_i, y_i, z_i, v_j, u_{jk}, \bar{a}_i\},\ i \in \Delta,$$

где x_i, y_i, z_i — координаты t_i; Δ — множество точек 3D-модели; v_j — тип объекта определенного назначения, которому принадлежит t_i, $v_j \in V$, V — множество типов объектов, входящих в пространственную 3D-модель; u_{jk} — k-й объект v_j -го типа, $u_{jk} \in U_j$, U_j — множество объектов типа v_j; $\bar{a}_i$ — множество атрибутивных данных об u_{jk}-м объекте, имеющим отношение к t_i-й точке 3D-модели.

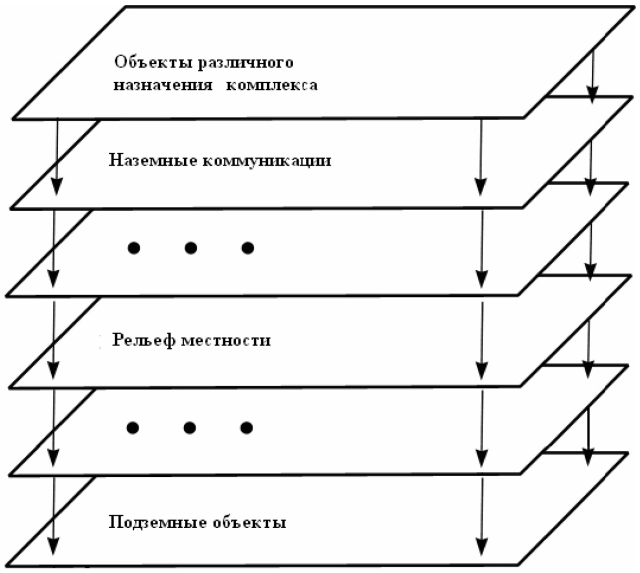


Рис. 1. Схема представления данных об объектах комплекса в виде совокупностей тематических слоев

При построении пространственной модели с использованием ГИС-технологий объекты различного назначения комплексов представляются в виде совокупностей тематических слоев и свя-

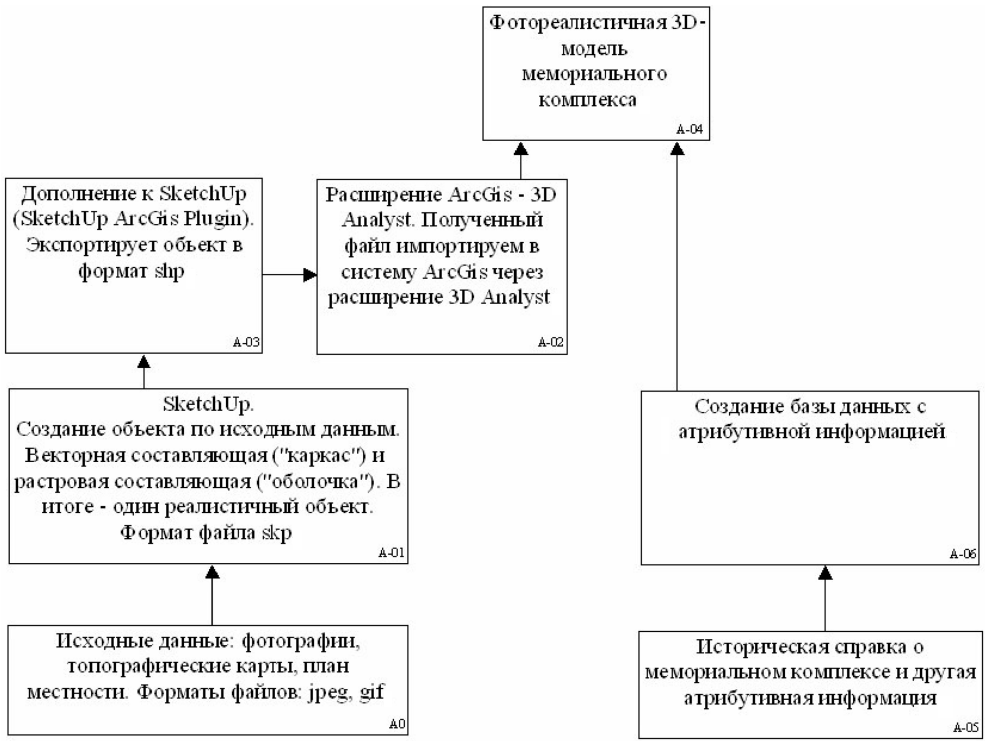


Рис. 2. Функциональная схема создания 3D-модели объекта

занных с ними атрибутивных данных в табличной и текстовой формах, схематично представленных на рис. 1.

При создании 3D-видов моделей сложных объектов в ArcGis для расширения функциональных возможностей системы было использовано приложение Google SketchUp компании Google [2], позволяющее не только создавать сложные трехмерные модели, но и накладывать на них текстуры. Это позволяет получить 3D-модели объектов, наиболее точно совпадающие с реальными образами. Функциональная схема создания 3D-вида объекта изображена на рис. 2.

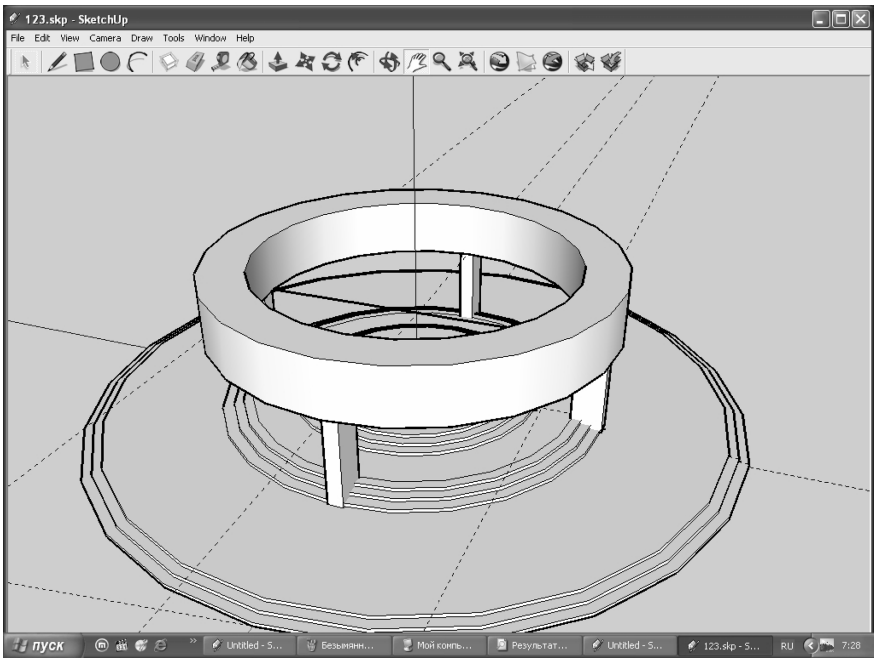


Рис. 3. Стадия № 1 создания 3D-модели объекта

Вначале создается геометрическая модель объектов мемориального комплекса с учетом реальных размеров (рис. 3). С использованием элементов стандартной базы SketchUp (деревья, кустарники, уличные фонари и др.) модели придается вид, приближенный к реальному. Далее инструментами SketchUp на геометрическую модель "натягиваются" фотореалистичные текстуры, полученные с помощью цифровой фотосъемки (рис. 4, см. четвертую сторону обложки). Далее осуществляется экспорт модели объекта в формат, поддерживаемый ArcGis. Таким образом, с помощью продукции кампании Google Sketchup получается фотореалистичная 3D-модель объекта мемориального комплекса в ArcGis.

Для полного ознакомления с конкретным мемориальным комплексом недостаточно его 3D-модели, поэтому следующей стадией создания виртуального музея является создание базы данных с различной атрибутивной информацией. С помощью SQL-запросов можно осуществить выборку по конкретному мемориальному комплексу, памятнику или отдельному воину.

В качестве дополнительной информации посетителю виртуального музея предоставлены коллекции цветных фотоснимков, используемых при построении пространственной модели, а также схем-карт, посвященных военным действиям в данной местности, детальные описания особенностей взаимосвязей между событиями, которым посвящен данный комплекс, и отдельными его объектами в пространственном и временном разрезах.

В перспективе результаты работы будут представлены в двух версиях: локальном (на CD-диске) и сетевом (на сервере Тамбовского государственного технического университета) для пользователей сети Интернет. В качестве примера на рис. 5 и 6 (см. четвертую сторону обложки) приведены фрагменты 3D-видов моделей двух комплексов: монумента Вечной славы в г. Тамбове и мемориального комплекса на станции РАДА Тамбовского района. Посещение виртуального музея (просмотр 3D-моделей и знакомство с атрибутивной информацией с CD-носителей) можно осуществлять с помощью таких программ-просмотрщиков, как ArcView 9.x Evaluation, ArcReader, ArcExplorer. Также планируется создание web-сервера на базе ArcGIS Server. Преимущество сетевой версии очевидно: пользователям достаточно иметь на своем компьютере только Интернет-браузер для связи с web-приложением, опубликованным разработчиком.

Работа выполнена по программе гранта Президента Российской Федерации для поддержки творческих проектов общенационального значения в области культуры и искусства "Технология создания виртуального музея мемориальных комплексов, посвященных Великой Отечественной войне, с использованием геоинформационных систем" (распоряжение Президента РФ от 16.05.2008 № 264-рп).

Список литературы

1. **Майкл де Мерс.** Географические информационные системы. М.: Дата+, 2000. 490 с.
2. **Intro** to Google SketchUp Book Bundle. http://sketchup.google.com/tutorials.html (01.01.07).

УДК 004.9:55

Д. Н. Кобзаренко, канд. техн. наук,
ст. науч. сотр.,
Институт проблем геотермии Дагестанского НЦ

Ускорение поиска ближайших узлов в задаче 2D-интерполяции на регулярной сетке при большом объеме исходных и результирующих данных

Излагается методика ускорения поиска ближайших узлов в задаче 2D-интерполяции на регулярной сетке при большом объеме исходных и результирующих данных. На основе предлагаемой методики разработан алгоритм интерполяции на регулярной сетке, заключающейся в поиске ближайшего соседа. Тестирование на реальном наборе данных показало, что данная методика дает приблизительно двукратное ускорение вычисления по сравнению с широко известным программным пакетом Golden Software Surfer 7.0.

Ключевые слова: *геоинформационные технологии, двумерная интерполяция.*

Введение

На сегодняшний день в геоинформационном моделировании (и не только) при расчете и визуализации данных функции $z = f(x, y)$ используются два основных метода: регулярная сетка и триангуляция Делоне [1]. Преимущество триангуляции Делоне состоит в том, что полученная модель более эффективна для визуализации, так как содержит минимально необходимое число объектов (треугольников). В свою очередь регулярная сетка позволяет построить более точную (точность зависит от шага между узлами) модель распределяемого параметра и является очень удобной структурой данных для быстрого доступа и извлечения необходимой информации.

Двумерная (2D) интерполяция функции $z = f(x, y)$ на регулярной сетке заключается в нахождении z-координаты для каждого ее узла с известными координатами x и y. Исходными данными являются: границы регулярной сетки: $\min X$, $\min Y$, $\max X$, $\max Y$ и шаг между ее узлами — *Space*; массив известных значений функции $z = f(x, y)$ (или исходных узлов для построения) — $R_0(x_0, y_0, z_0)$, $R_1(x_1, y_1, z_1)$, ..., $R_{N-1}(x_{N-1}, y_{N-1}, z_{N-1})$ длиной N.

Существуют несколько методов интерполяции значений функции $z = f(x, y)$, наиболее популярные из которых: метод инверсных расстояний и метод Кригинга. Возможность построения цифровой модели рельефа на регулярной сетке, наряду с триангуляцией Делоне, имеется в 3D геоинформационных системах общего назначения, таких как *ArcView 3D-Analyst* [2].

Для выполнения 2D-интерполяции на регулярной сетке существует очень удобный программный пакет *Surfer* фирмы *Golden Software*, в котором реализовано несколько методов интерполяции[1] [3]. Его удобство состоит в открытости для пользователя форматов входных и выходных данных. При построении геоинформационных моделей в различных задачах авторы часто использовали этот программный продукт. Однако, работая с очень большими массивами входных и выходных данных, пришлось столкнуться с определенной проблемой. Она заключается в том, что в некоторых случаях время, затрачиваемое на интерполяцию, достигает нескольких часов.

Процедура интерполяции значения Z_0 для каждого узла регулярной сетки с координатами X_0, Y_0 состоит из двух частей: 1) поиск одного или нескольких ближайших (к точке X_0Y_0) исходных узлов с известными значениями Z; 2) вычисление Z_0 на основе координат найденных узлов по формулам одного из методов интерполяции. Подавляющая часть времени, затрачиваемого на интерполяцию, расходуется на поиск ближайших исходных узлов. Традиционно поиск осуществляется следующим образом. Из начальной точки испускается круговая "волна". И при каждом очередном значении увеличивающегося радиуса осуществляется проверка элементов массива исходных данных на принадлежность внутренней области окружности. Предварительно массив элементов исходных данных может быть упорядочен по определенному признаку. Именно на перебор и сравнение элементов этого массива при большом их числе и затрачивается много машинного времени.

В данной работе предлагается методика для ускорения поиска ближайших узлов в рамках 2D-интерполяции на регулярной сетке. Методика реализована нами на практике в виде алгоритма интерполяции, заключающейся в поиске ближайшего соседа. Это позволило сравнить скорость интерполяции, основанной на предлагаемой методике, с тем, что предлагает тот же *Surfer*, и сделать выводы о возможности ее использования в 2D-интерполяции.

1. Загрузка и начальная обработка исходных данных

Описывать методику будем с позиций решения задачи о поиске ближайшего соседа для каждого

[1] Версия 8.0 пакета *Surfer* содержит уже возможность выполнения и 3D-интерполяции $w = f(x, y, z)$.

интерполируемого узла. Первым шагом является задание параметров регулярной сетки: Min$X$ — минимальное значение по X; Min$Y$ — минимальное значение по Y; Max$X$ — максимальное значение по X; Max$Y$ — максимальное значение по Y; *Space* — шаг между узлами регулярной сетки (в работе делаем допущение, что шаги вдоль оси X и вдоль оси Y равны). Далее определяются число столбцов и строк в регулярной сетке. Число столбцов определяется как *ColCount* = (Max$X$ − − Min$Y$)/*Space* + 1, а число строк как *RowCount* = = (Max$Y$ − Min$Y$)/*Space* + 1.

Следующим шагом является загрузка исходных узлов в массив *Nodes*, содержащий элементы, соответствующие 3D-узлу. Число исходных узлов определяется как *Count*. Узлы индексируются от 0 до *Count* − 1. Сразу после загрузки проводится сортировка элементов массива *Nodes* по возрастанию ключей X и Y. Причем, если *RowCount* >= *ColCount*, то X — первичный ключ, Y — вторичный, иначе Y — первичный ключ, X — вторичный. Алгоритмам сортировки и поиска посвящен т. 3 классического труда Дональда Э. Кнута "Искусство программирования" [4]. В книге приведено множество алгоритмов сортировки. Автор данной статьи остановил свой выбор на пирамидальной сортировке (из семейства методов сортировки посредством выбора).

После сортировки массива исходных данных *Nodes* вычисляются границы периметра, оконтуривающего данные исходного массива узлов в плоскости XY: MinX_{Nodes}, MinY_{Nodes}, MaxX_{Nodes}, MaxY_{Nodes}.

2. Принцип построения регулярной сетки

Принцип построения регулярной сетки в рамках предлагаемой методики состоит в следующем. Само собой разумеется, чтобы перебрать все узлы регулярной сетки, для которых надо найти $z = f(x, y)$, необходимо два цикла. Если число строк превышает число столбцов (*RowCount* >= *ColCount*), то внешний цикл открывается по X (x возрастает от MinX_{Nodes} до MaxX_{Nodes} с шагом *Space*), а внутренний — по Y (y возрастает от MinY_{Nodes} до MaxY_{Nodes} с шагом *Space*). В противном случае внешний цикл открывается по Y, а внутренний по X. Далее по ходу статьи при рассмотрении методики опишем случай (*RowCount* >= *ColCount*), т. е. внешний цикли — по X, а внутренний — по Y.

Пусть $X = X_0$. Для текущего значения X_0 активизируется выполнение интерполяции в узлах регулярной сетки, у которых координата x равна X_0, а y меняется в пределах границы регулярной сетки от MinY_{Nodes} до MaxY_{Nodes} с шагом *Space*. Алгоритм поиска основан на построении окружности, т. е. испускании волны из точки поиска. Для более четкого понимания ускоренного алгоритма интерполяции и собственно самой идеи ускорения опишем вначале некоторые локальные алгоритмы.

3. Алгоритмы предварительной выборки данных из массива исходных узлов

Предварительная выборка данных из массива исходных узлов *Nodes* состоит в следующем (рис. 1). Пусть на какой-то стадии поиска ближайшего узла текущий радиус поиска равен R, тогда выбираем в массиве исходных данных те элементы $Nodes_i(x, y, z)$, у которых x принадлежит промежутку $[X_0 - R, X_0 + R]$. Поскольку изначально массив *Nodes* отсортирован по ключам X, Y, то нам достаточно найти ссылки на первый элемент выборки (*Start*) и последний элемент выборки (*Finish*) (рис. 1). А все промежуточные элементы массива будут удовлетворять условию выборки.

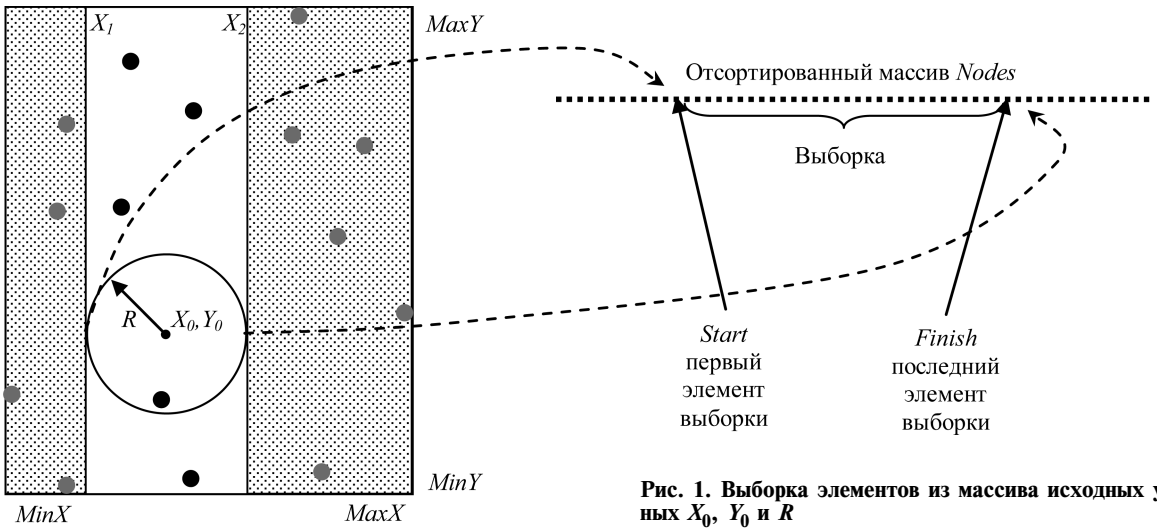
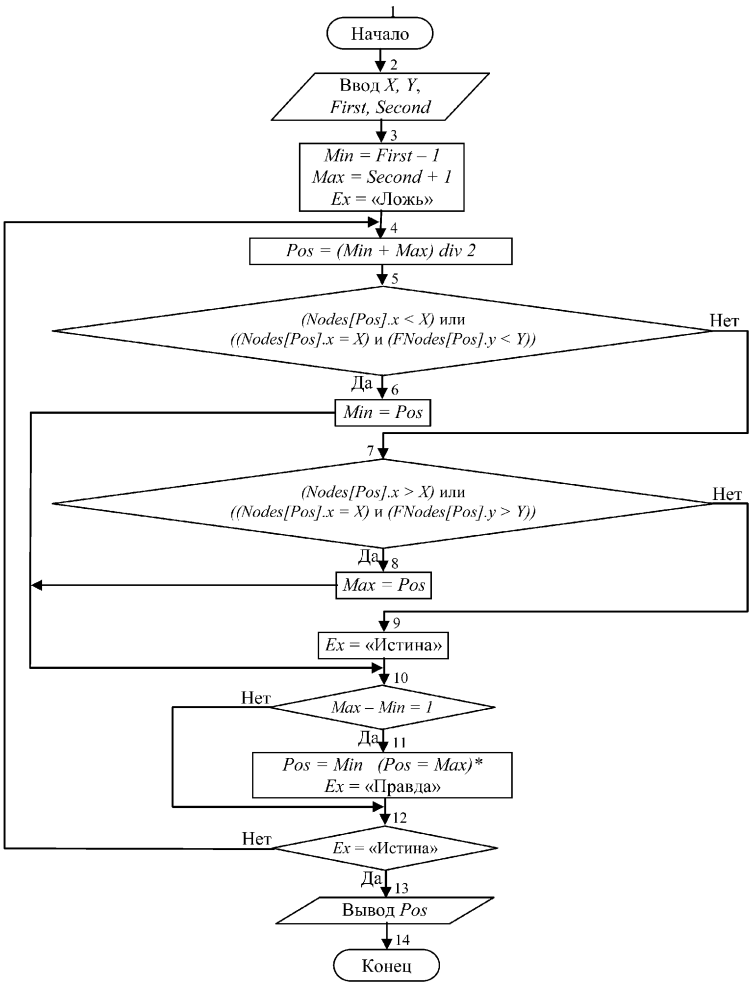


Рис. 1. Выборка элементов из массива исходных узлов при заданных X_0, Y_0 и R



* - Единственное отличие алгоритмов для первого и последнего элемента в блоке 11: *Pos = Min* – для первого элемента *Start*, *Pos = Max* – для последнего элемента *Finish*.

Рис. 2. Блок-схема алгоритма получения первого и последнего элементов выборки из массива исходных узлов

На рис. 1 показано, что таким образом сразу отсекаются те узлы (показаны серым цветом в заштрихованных областях), которые не имеет смысла рассматривать, так как они не принадлежат промежутку $[X_0 - R, X_0 + R]$ и, само собой, не принадлежат окружности поиска. Алгоритмы поиска первого и последнего элементов выборки основаны на половинном делении и приведены на рис. 2. Исходными данными для алгоритмов поиска являются: X, Y — координаты, в соответствии с которыми проводится поиск; *First*, *Second* — диапазон рассматриваемых элементов в массиве *Nodes*. При $First = 0$, $Second = Count - 1$ рассматриваются все элементы массива *Nodes*.

Для четкого понимания, как работают алгоритмы получения первого и последнего элементов выборки, приведем следующий пример. Пусть массив *Nodes* (уже отсортированный по ключам X, Y) представляет собой 13 элементов: $N_0[x = 2, y = 3, z = 1]$, $N_1[x = 2, y = 15, z = 1]$, $N_2[x = 2, y = 20, z = 1]$, $N_3[x = 2, y = 30, z = 1]$, $N_4[x = 5, y = 5, z = 1]$, $N_5[x = 5, y = 10, z = 1]$, $N_6[x = 5, y = 17, z = 1]$, $N_7[x = 5, y = 30, z = 1]$, $N_8[x = 7, y = 45, z = 1]$, $N_9[x = 10, y = 2, z = 1]$, $N_{10}[x = 10, y = 10, z = 1]$, $N_{11}[x = 10, y = 12, z = 1]$, $N_{12}[x = 15, y = 2, z = 1]$. Для такого массива исходных данных представим некоторые примеры входных данных и результатов работы алгоритма (см. таблицу).

4. Структура данных выборки

Прежде чем перейти к описанию структуры данных выборки, дадим пояснение к поиску ближайшего соседа по волне окружности. Поиск инициируется окружностью с начальным значением радиуса $R = R_0$ (рис. 3). Начальному радиусу ставится в соответствие начальный шаг поиска $Step = 0$. Далее окружность поиска растет со значениями радиусов $R = R_0 \cdot 2$ (при $Step = 1$), $R = R_0 \cdot 3$ (при $Step = 2$), ..., $R = R_0 \cdot (N + 1)$ (при $Step = N$).

Структура данных выборки динамически создается при переходе к интерполяции в узлах очередного столбца регулярной сетки с координатой X_0 и разрушается по выполнению интерполяции для всех узлов этого столбца. Структура данных состоит из следующих элементов: X_0 — координата текущего столбца регулярной сетки; R_0 — начальное значение радиуса поиска;

№№	Входные данные					Результат
	X	*Y*	*First*	*Second*	Тип операции	
1	1	50	0	12	*Start*	0
2	2	16	0	12	*Start*	2
3	5	11	0	12	*Start*	6
4	20	1	0	12	*Start*	13*
5	3	10	7	10	*Start*	7
6	5	16	0	12	*Finish*	5
7	6	10	0	12	*Finish*	7
8	0	30	0	12	*Finish*	−1*
9	2	40	0	12	*Finish*	3
10	2	40	5	10	*Finish*	4*

* При поиске позиции *Start*, если ни один элемент не удовлетворяет условию *X*, *Y*, то результат равен *Second* + 1; при поиске позиции *Finish*, если ни один элемент не удовлетворяет условию *X*, *Y*, то результат равен *First* − 1.

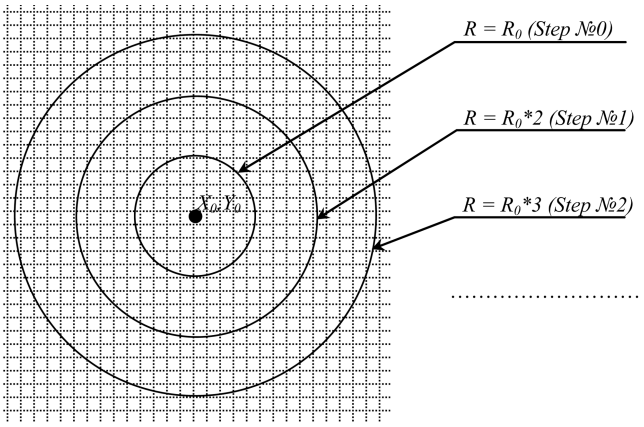


Рис. 3. Растущие окружности поиска ближайшего узла

Pos — массив элементов типа *TPos*; *Count* — число элементов в массиве *Pos*. Тип *TPos* представляет собой запись для хранения данных, соответствующих очередному шагу испускания волны поиска *Step*. Запись состоит из следующих элементов: *Start* — ссылка на первый элемент выборки из массива исходных узлов; *Finish* — ссылка на последний элемент выборки из массива исходных узлов; *Active* — переменная логического типа, которая принимает значение "истина", если из массива исходных узлов выбран один или более элементов; R — значение радиуса на данном шаге.

Работа со структурой данных происходит следующим образом. Посылается запрос на получение *Start*, *Finish*, R и *Active* для текущего шага поиска *Step*. Делается проверка на наличие данных для этого шага. Эти данные рассчитываются, в случае их отсутствия, с сохранением результата. Запрос в любом случае удовлетворяется.

Такая структура данных и принцип ее работы позволяют существенно экономить время, затрачиваемое на интерполяцию, поскольку она обеспечивает более быстрое получение данных, являющихся общими для узлов одного столбца регулярной сетки.

5. Алгоритм поиска ближайшего соседа при заданных *Start*, *Finish*, R, X_0, Y_0

В предыдущих разделах мы рассмотрели вопросы выборки элементов из массива исходных узлов *Nodes*. Теперь рассмотрим алгоритм поиска ближайшего узла (если таковой существует) из тех, что были выбраны.

Пусть на очередном шаге поиска *Step* нам известны данные выборки *Start*, *Finish*, а также: *Active* — наличие узлов в выборке; R — радиус поиска на данном шаге; X_0, Y_0 — координаты интерполируемого узла регулярной сетки. Требуется либо найти ближайший узел из тех, что попадают

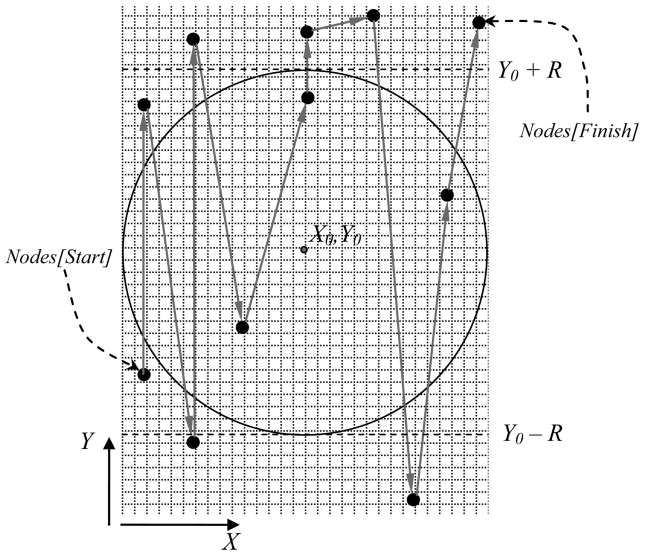


Рис. 4. Поиск ближайшего соседа при заданных *Start*, *Finish*, R, X_0, Y_0

в окружность поиска, либо констатировать факт отсутствия исходных узлов в окружности поиска. Рис. 4 иллюстрирует пример такого поиска. Здесь серыми стрелками показана последовательность узлов из массива *Nodes* в выбранных изначально пределах *Start* и *Finish*.

В рамках решения текущей задачи мы последовательно должны просмотреть все узлы из массива *Nodes* в диапазоне индексов *Start* и *Finish* и принять решение по каждому узлу. Выполнение текущей задачи иллюстрирует блок-схема алгоритма, представленная на рис. 5.

Принцип работы алгоритма заключается в том, что, просматривая все узлы промежутка от *Start* до *Finish*, проверяется вначале принадлежность узла промежутку $[Y_0 - R, Y_0 + R]$ и только потом принадлежность его окружности поиска.

6. Использование обратной связи для установки начального шага *Step* при интерполяции очередного узла регулярной сетки

Использование обратной связи для установки начального шага *Step* заключается в следующем. Пусть нам известно, что для предыдущего узла текущего столбца регулярной сетки значение шага, при котором найден ближайший сосед, равно $Step = N$. Тогда для текущего узла текущего столбца можно принять шаг *Step* исходя из следующих выражений: $R = (N + 1)R_0 - Space$. Если $R < R_0$, тогда $R = R_0$ и $Step = Round(R/R_0) - 1$.

Это позволяет экономить машинное время, не проводя поиск для шагов, при которых заведомо известно об отсутствии исходных узлов в пределах окружности поиска.

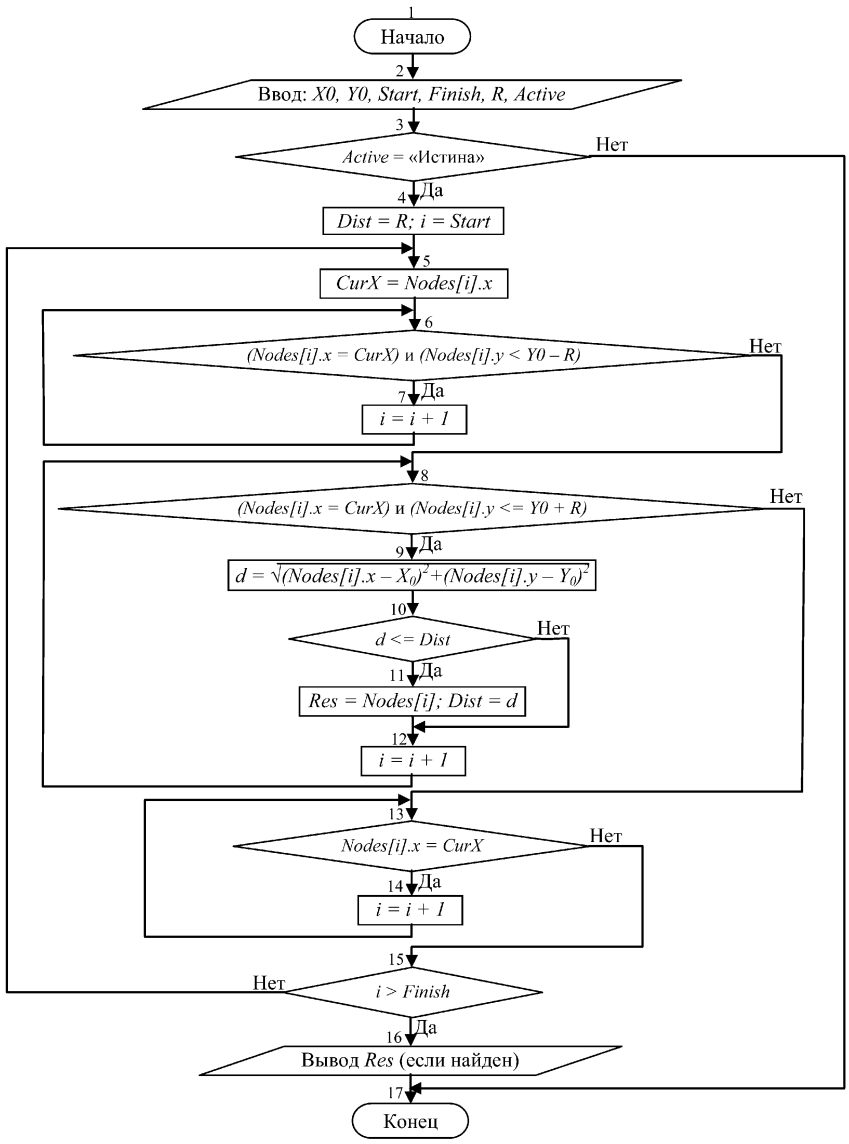


Рис. 5. Блок-схема алгоритма поиска ближайшего соседа при заданных *Start*, *Finish*, *R*, X_0, Y_0

7. Алгоритм интерполяции на базе предложенной методики

Опишем пошагово алгоритм интерполяции на базе предложенной нами методики и описанных локальных алгоритмов и структур данных.

1. Ввод исходных данных регулярной сетки Min*X*, Min*Y*, Max*X*, Max*Y*, *Space*.

2. Ввод массива исходных узлов *Nodes* с предварительной сортировкой элементов по ключам *X* и *Y*.

3. Определение начального значения радиуса поиска R_0.

4. Для очередного столбца с координатой X_0 выполняются пункты 5—16.

5. Создается структура данных выборки с параметрами X_0 и R_0.

6. *Step* = 0.

7. Для очередного узла регулярной детки X_0, Y_0 выполняются пункты 8—14.

8. Посылается запрос для текущего шага поиска *Step* в структуру данных выборки и в результате выдается *Start*, *Finish*, *R*, *Active*.

9. На базе *Start*, *Finish*, *R*, X_0, Y_0 находится ближайший сосед по алгоритму на рис. 5.

10. Если ближайший сосед (узел) найден, то присваиваем его значение *Z* значению узла регулярной сетки и переходим к пункту 11, иначе *Step* = *Step* + 1 и переходим к пункту 8.

11. $R = (Step + 1)R_0 - Space$.

12. Если $R < R_0$, то $R = R_0$.

13. $Step = Round(R/R_0) - 1$.

14. Если выполнена интерполяция для всех узлов столбца с координатой X_0, то переход к пункту 15, иначе $Y_0 = Y_0 + Space$ и переходим к пункту 7.

15. Разрушается структура данных выборки.

16. Если выполнена интерполяция для всех узлов регулярной сетки, то переход к пункту 17, иначе $X_0 = X_0 + Space$ и переходим к пункту 4.

17. Конец.

Следует еще раз отметить, что приведенный алгоритм учитывает то, что число строк регулярной сетки превышает число столбцов. В противном случае целесообразнее *X* и *Y* поменять местами для повышения быстродействия вычислений.

Заключение и выводы

На основе предложенной методики нами разработан программный модуль на *Delphi*, реализующий интерполяцию методом поиска ближайшего соседа. Для тестирования были взяты данные из реальной практической задачи — массив, состоящий из 1 132 568 элементов (узлов). Были заданы следующие параметры регулярной сетки: $\min X = -1430$, $\min Y = -1890$, $\max X = 4480$, $\max Y = 6760$, *Step* = 5, итого 2 047 773 узлов.

Тестирование временных показателей интерполяции проводилось на средней по мощности машине на базе процессора *Celeron*. При тестировании не учитывалось время считывания исходных данных и их предварительной сортировки (как для нашего модуля, так и в *Surfer*).

В рамках тестирования варьировали начальное значение радиуса поиска. Для начала вычислили его исходя из значения площади, занимаемой окружностью, на которую попадает в среднем один узел из массива исходных узлов:

$$R_0 = \sqrt{\frac{(\mathrm{Max}X - \mathrm{Min}X) \cdot (\mathrm{Max}Y - \mathrm{Min}Y)}{\pi\, Count}}. \qquad (1)$$

Для нашего массива исходных данных вычисленное по формуле (1) значение начального радиуса поиска составляет $R_0 = 3{,}7$. Кроме вычисленного значения при тестировании мы задавали значения начального радиуса $R_0 = 5$, $R_0 = 10$ и $R_0 = 15$. В результате были получены следующие временные показатели: при $R_0 = 3{,}7$ время выполнения интерполяции составляет 26 мин 48 с, при $R_0 = 5$ время равно 18 мин 55 с, при $R_0 = 10$ — 18 мин 11 с, при $R_0 = 15$ — 18 мин 19 с. Отсюда следует, что для оптимальной скорости работы алгоритма нужно внести изменение в формулу расчета начального радиуса поиска, добавив умножение правой части равенства (1) на 3.

Для сравнения программный пакет *Surfer 7* с тем же набором входных данных и теми же параметрами регулярной сетки выполняет интерполяцию (методом поиска ближайшего соседа) за 35 мин. Таким образом, ускорение поиска ближайших узлов в задаче 2D-интерполяции на регулярной сетке, предложенное нами, по сравнению с аналогом в *Surfer* приблизительно двукратно.

В заключение следует отметить, что метод интерполяции, заключающийся в поиске ближайшего соседа, взят всего лишь как пример, для того чтобы показать преимущества разработанной методики ускорения поиска ближайших узлов. Такую же методику поиска с некоторыми модификациями и дополнениями можно применить, например, для интерполяции методом инверсных расстояний, что будет выполнено в дальнейшем.

Список литературы

1. **Скворцов А. В.** Обзор алгоритмов построения триангуляции Делоне // Вычислительные методы и программирование. 2002. Т. 3. С. 14—39.
2. http://www.esri.com/
3. http://www.goldensoftware.com/
4. **Дональд Э. Кнут.** Искусство программирования. Том 3. Сортировка и поиск. / Под ред. Ю. В. Козаченко. М.—СПб.—Киев: Вильямс, 2000. 822 с.

МОДЕЛИРОВАНИЕ

УДК 539.3

Ю. И. Димитриенко, д-р физ.-мат. наук, проф., зав. каф.,
А. П. Соколов, аспирант,
МГТУ им. Н. Э. Баумана

Система автоматизированного прогнозирования свойств композиционных материалов

Представлены результаты разработки методики автоматизированного вычисления эффективных упругих характеристик композиционных материалов с различной структурой армирования: пространственно-армированной по трем ортогональным направлениям, четырехнаправленной по диагоналям куба и тканевой. Расчет осуществлен на основе конечно-элементного метода решения так называемых локальных задач $Ж_{pq}$, возникающих при использовании метода асимптотического осреднения Бахвалова — Победри. Представлены результаты разработки программного комплекса для вычисления эффективных упругих характеристик композитов и некоторые характерные результаты его работы для указанных типов структур армирования.

Ключевые слова: *периодические структуры, композитные материалы, гомогенизация, гомогенный, гетерогенный, асимптотическое осреднение, упругость.*

Введение

Композиционные материалы благодаря уникальному сочетанию их эксплуатационных свойств — низкому весу по отношению даже к алюминиевым сплавам, высокой жесткости и прочности, высокой ударной вязкости и стойкости к распространению трещин, высокой химической стойкости, технологичности и другим — в настоящее время являются одним из наиболее перспективных классов конструкционных материалов. Существенный недостаток этих материалов — высокая цена, ранее ограничивавшая их применение в гражданской технике, в настоящее время во многом уже не является препятствием,

вследствие того, что цена на конечную продукцию (например, самолеты, морские суда) в настоящее время в значительно меньшей степени определяется стоимостью материалов и комплектующих.

Таких уникальных свойств композиционных материалов (КМ или композитов) удается достичь благодаря тому, что они состоят из совокупности совершенно различных компонентов, например, из высокопрочных армирующих волокон, пластичной матрицы, обеспечивающей хороший контакт с волокнами, а также имеют четко выраженную границу раздела матрицы и волокон, препятствующую распространению трещин. При синтезе новых композитов возникают несколько теоретических задач, одной из них является задача прогнозирования свойств новых композитов по свойствам их компонентов, успешное решение которой помогает найти наиболее рациональное сочетание компонентов и достичь наилучших эксплуатационных свойств композитов. Экспериментальный перебор всех возможных вариантов композиций, как правило, технически затруднен и является дорогостоящим. Расчетные методы позволяют значительно уменьшить затраты на создание новых материалов, а иногда даже обеспечивают возможность синтеза новых материалов "с наперед заданными свойствами".

Еще в 70-х годах прошлого столетия были созданы основные методы расчета свойств "классических" композитов [1—6] (слоистых, однонаправленно-армированных, слоисто-волокнистых), основанные, главным образом, на приближенном анализе физических полей (полей упругих напряжений и деформаций, тепловых, электромагнитных полей). Однако для новых современных типов композитов, у которых армирование осуществляется по трем и более направлениям, и особенно для перспективных типов — нанокомпозитов, имеющих сложную пространственную структуру соединения компонентов, эти приближенные методы дают большую погрешность и зачастую оказываются неприемлемыми. В 80-х годах прошлого столетия появился новый математический метод Бахвалова — Победри [7, 8], основанный на теории периодических структур, который позволяет в принципе вычислять свойства композитов с любой наперед заданной точностью. Однако в этом методе необходимо решать специальные краевые задачи — так называемые "задачи на ячейках" (локальные задачи), которые, вообще говоря, являются интегродифференциальными и имеют неклассические условия периодичности. Эти задачи оказались сложными даже для решения их численными методами, вследствие чего к настоящему времени удалось решить лишь несколько таких задач аналитически и численно для сравнительно простых типов структур композитов [8—13]. В работах [14—17], выполненных в МГТУ им. Н. Э. Баумана, было осуществлено дальнейшее развитие метода Бахвалова — Победри, в результате которого удалось свести задачи на ячейке периодичности к задачам "классического" типа, для которых может быть эффективно применен метод конечного элемента. Этот результат открыл возможность для разработки программного продукта, позволяющего автоматизировать расчет свойств композитов по свойствам их компонентов практически для любых геометрических структур компонентов. Следует отметить, что широко известные коммерческие конечно-элементные программные продукты, такие как ANSYS, NASTRAN, которые содержат в своих библиотеках конечных элементов модули для расчета композитов, также "не умеют решать" указанные задачи на ячейке периодичности. Применение этих программных продуктов оказывается возможным, только после сведения задач на ячейке к краевым задачам теории упругости, теплопроводности и другим, имеющим "классически тип". Именно такой способ редуцирования задач на ячейках и был предложен в работах [14—17], а в работе [14] было проведено сравнение результатов решения задач на ячейке, получаемых с помощью программного продукта, разработанного в МГТУ им. Н. Э. Баумана, и с помощью пакета ANSYS (который использовался только как решатель системы линейных алгебраических элементов, возникающей в конечно-элементных методах), которое показало хорошее совпадение результатов.

В настоящей работе представлены результаты дальнейшей разработки метода решения задач на ячейке периодичности, выполненные на кафедре "Вычислительной математики и математической физики" МГТУ им. Н. Э. Баумана, которые позволили создать систему автоматизированного прогнозирования свойств композитов.

Математическая формулировка локальной задачи

Ограничимся только рассмотрением проблемы прогнозирования жесткостных (упругих) свойств композитов. Возможности системы по прогнозу других физико-механических, теплофизических и электромагнитных свойств композитов отражены в работах [18].

Для прогнозирования упругих свойств композитов (их называют эффективными свойствами), согласно методу Бахвалова — Победри, модифицированному в работах [10—13], необходимо решить так называемые локальные задачи $Ж_{pq}$ на 1/8 части ячейки периодичности (ЯП) (рис. 1, *а*), которые представляют собой серию контактных задач теории упругости и имеют следующий вид:

$$
\begin{cases}
\sigma^{\alpha}_{ij(pq)/j} = 0 \quad \text{в } \tilde{V}_{\xi}, & \text{— уравнения равновесия;} \\
\sigma^{\alpha}_{ij(pq)} = C^{\alpha}_{ijkl}\,\varepsilon^{\alpha}_{kl(pq)} = \lambda^{\alpha}\varepsilon^{\alpha}_{kr(pq)}\,\delta_{ij} + 2\mu^{\alpha}\varepsilon^{\alpha}_{kl(pq)}, & \text{в } \tilde{V}_{\xi} \cup \Sigma'_s \cup \Sigma_s \text{ — определяющие соотношения;} \\
\varepsilon^{\alpha}_{ij(pq)} = \dfrac{1}{2}\,(U^{\alpha}_{i(pq)/j} + U^{\alpha}_{j(pq)/i}), & \text{в } \tilde{V}_{\xi} \text{ — соотношения Коши;} \\
\left.\begin{array}{l} U^{\alpha}_{i(pq)} = U^{N}_{i(pq)} \\ (\sigma^{\alpha}_{ij(pq)} - \sigma^{N}_{ij(pq)})\,n_j = 0 \end{array}\right\} \text{ на } \tilde{\Sigma}_{\xi\alpha N} & \text{— условия идеального контакта.}
\end{cases} \tag{1}
$$

Здесь функции $U^{\alpha}_{i(pq)}$, называемые псевдоперемещениями, являются решением задач $Ж_{pq}$ на 1/8 ЯП ($\tilde{V}_{\xi} = V_{\xi} \cap (\xi^i \geqslant 0)$), где V_{ξ} — полная ЯП; ξ^i — так называемые локальные координаты, изменяющиеся на ЯП; $\sigma^{\alpha}_{ij(pq)}\ \varepsilon^{\alpha}_{ij(pq)}$ — напряжения и деформации в компонентах композита [10]; C^{α}_{ijkl} — компоненты тензоров модулей упругости компонентов композита; λ^{α}, μ^{α} — константы Ламе α-го компонента КМ; $U^{\alpha}_{j(pq)/i} = \dfrac{\partial}{\partial\xi^i}\,U^{N}_{j(pq)}$ — частные производные по локальным координатам ξ^i; n_i — компоненты вектора нормали к поверхности раздела $\tilde{\Sigma}_{\xi\alpha N}$ компонентов композита.

К системе (1), кроме условий идеального контакта на границе компонентов композита, присоединяются условия на торцевых поверхностях ЯП Σ'_s и на плоскостях симметрии $\Sigma_s = \{\xi_s = 0\}$ [14]:

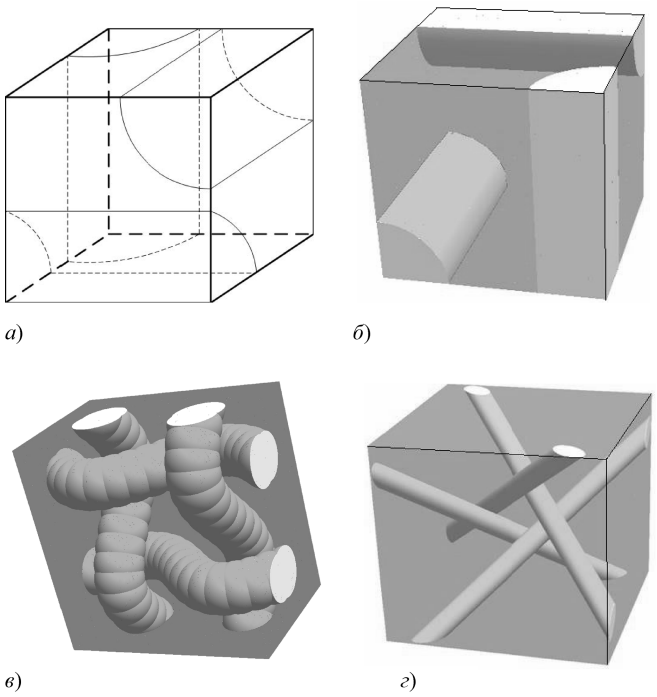
при $p = q$: $U^{\alpha}_{i(pq)} = \dfrac{1}{2}\,\bar{\varepsilon}_{pq}\delta_{ip}$, $U^{\alpha}_{j(pq)/i} = 0$, $U^{\alpha}_{k(pq)/i} = 0$ на Σ'_i, $i \neq j \neq k \neq i$;

при $p \neq q$: $U^{\alpha}_{i(pq)} = \dfrac{1}{4}\,\bar{\varepsilon}_{pq}\delta_{ip}$, $U^{\alpha}_{j(pq)/j} = 0$, $U^{\alpha}_{k(pq)/j} = 0$ на Σ'_j; (2)

$$U^{\alpha}_{i(pq)/k} = 0,\ U^{\alpha}_{j(pq)/j} = 0,\ U^{\alpha}_{k(pq)} = 0 \text{ на } \Sigma'_j;$$
$$i, j = \{p, q\},\ i \neq j \neq k \neq i,$$

где $\bar{\varepsilon}_{pq}$ — эффективные деформации композита, которые в задаче (1), (2) являются заданными величинами.

Вариационная формулировка задачи (1), (2) и конечно-элементный метод ее решения предложены в работах [14—17].



а) *б*) *в*) *г*)

Рис. 1. Различные структуры композиционного материала: *а*, *б* — 3D-ортогонально-армированный композит; *в* — тканевый композит; *г* — 4D-армированный композит

Метод вычисления эффективных упругих характеристик

Если задача (1), (2) решена, то проинтегрировав полученные напряжения по областям $\tilde{V}^{\alpha}_{\xi}$, соответствующим компонентам композита для 1/8 ЯП, и воспользовавшись определяющими соотношениями в (1), находим тензор эффективных модулей упругости композита $\bar{C}_{ijpq}$ [14]:

$$\bar{\sigma}_{ij(pq)} = <\sigma^{(\alpha)}_{ij(pq)}(\xi^k)> = 8\sum_{\alpha=1}^{N}\int_{\tilde{V}^{\alpha}_{\xi}}\sigma^{\alpha}_{ij(pq)}(\xi^k)\,dV^{\alpha}_{\xi}; \tag{3}$$

$$\bar{C}_{ijpq} = \frac{\bar{\sigma}_{ij(pq)}}{\bar{\varepsilon}_{pq}}. \tag{4}$$

После расчета тензора эффективных модулей упругости $\bar{C}_{ijpq}$ вычисляется тензор эффективных упругих податливостей $\bar{\Pi}_{ijpq} = (\bar{C}_{ijpq})^{-1}$ и по его компонентам находятся эффективные технические константы. В частности, если композит в целом оказывается ортотропным, то для него вычисляются: $E_i = \dfrac{1}{\Pi_{iiii}}$ — эффективные модули Юнга; $\nu_{ij} = -\dfrac{\Pi_{iijj}}{\Pi_{iiii}}$ — эффективные коэффициен-

ты Пуассона; $G_{ij} = \overline{C}_{ijij}$ — эффективные модули сдвига композита (здесь по i, j суммирования нет).

Разработка системы автоматизированного вычисления эффективных упругих свойств композитов

В данном разделе кратко освещены результаты разработки автоматизированной технологии, которая предназначена для решения локальных задач $Ж_{pq}$ и вычисления эффективных упругих свойств композитов.

Был разработан специализированный программный комплекс (ПК), который построен с использованием принципов модульности: в нем проведено жесткое разделение задач на подэтапы — от постановки до ее решения. Каждый подэтап постановки задачи — задание граничных условий, задание свойств материалов, формирование геометрии — был выполнен в виде самостоятельных модулей (плагинов).

Разработанный ПК построен с потенциальной возможностью расширения, что чрезвычайно важно для дальнейших исследований, в частности, для возможности анализа новых структур КМ.

Модульный принцип разработки позволил существенно улучшить функциональные возможности всего программного комплекса, при этом были решены следующие задачи:

- реализована возможность "сборки" проектов/частей проектов программного комплекса из стандартных частей;
- отлажена работа генератора кода, который позволил осуществлять автоматизацию процесса разработки новых модулей системы;
- сформировано дерево репозитория объектов/проектов/БД и прочих компонентов системы;
- созданы шаблоны для формирования форм/ модулей (файлов *.cpp), проектов плагинов/приложений с поддержкой и без поддержки подключения к БД.

В составе ПК разработаны основные модули: решатель задач $Ж_{pq}$, препроцессор (задание произвольных граничных условий (ГУ), характеристик материалов, формирование достаточно произвольной геометрии расчетной области), постпроцессор (расчет эффективных упругих характеристик на основе серий решенных задач $Ж_{pq}$).

Был разработан образец для создания новых модулей системы, а также комплекс сервисных средств, облегчающих разработку новых программных модулей системы. Выбранная для разработки ПК технология позволяет масштабировать ПК. Имеется возможность создавать "заготовки" будущих проектов в исходных кодах (хост-приложения, плагины, *dll*-проекты и прочее в исходных кодах на *C++*), а также автоматически включать документацию в проекты в исходных кодах TeX.

В разработанном ПК имеется также возможность генерации программных компонентов меньшего масштаба: модули(*.cpp/*.h), формы, а также есть возможность простого создания новых шаблонов любой сложности.

Новая архитектура разработки в стандарте позволяет легко интегрировать программные сред-

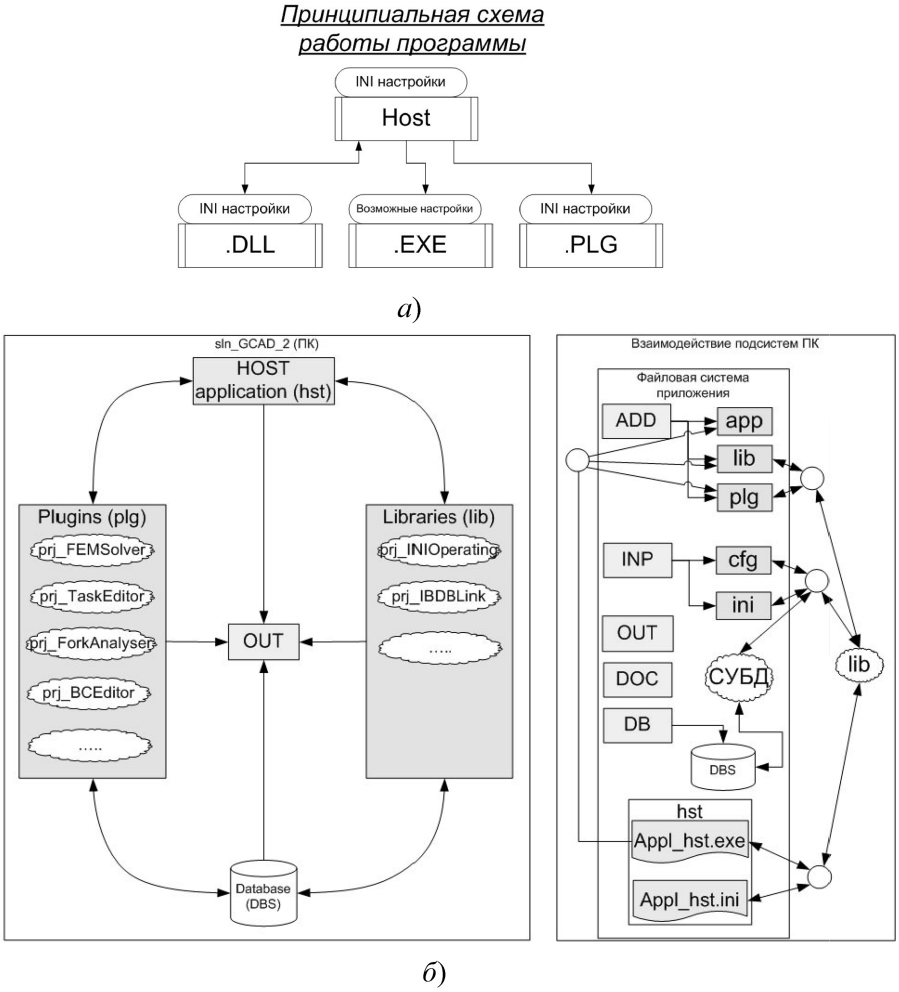


Рис. 2. Принципиальная схема работы модулей автоматизированного ПК:
а — основные модули хост-приложений; *б* — взаимодействие модулей в ПК

ства сторонних производителей (использование технологии плагин-компонентов). Разработка новых программных решений видится в рамках единого репозитория объектов, что позволяет систематизировать все работы, которые ведутся по данному проекту.

На рис. 2 представлена принципиальная схема работы ПК. Введены следующие обозначения:
ADD — *addons* — дополнительные модули системы;
INP — *input* — входные данные системы;
OUT — *output* — результаты работы системы;
DOC — *documentation* — документация;
app — *application* — внешние программы;
lib — *libraries* — библиотеки системы;
plg — *plugins* — автоматически подключаемые модули;
cfg — *configuration* — исходные данные с текстами;
ini — *initials* — параметры библиотек/плагинов;
prj — *project* — проект (программная единица системы);
sln — *solution* — решение (структурная единица системы);
prj_FEMSolver — решатель МКЭ для линейной задачи упругости;
prj_TaskEditor — редактор постановки задачи;
prj_BCEditor — редактор задания граничных условий краевой задачи;
prj_ForkAnalyser — постпроцессор-анализатор ЭУХ по методам Фойгта—Рейсса и Хашина — Штрикмана;
prj_INIOperating — библиотека по работе с INI файлами;

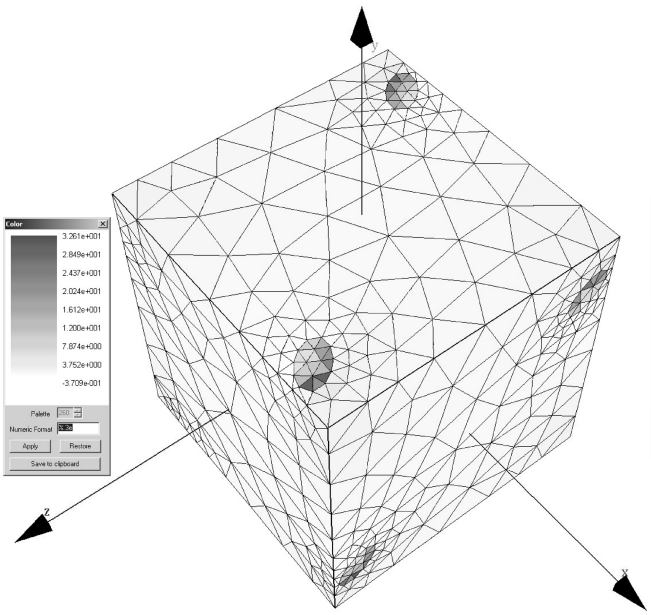


Рис. 3. Распределение псевдонапряжений $\sigma^{\alpha}_{12(12)}$ (МПа) в задаче $Ж_{12}$ для 4DKM

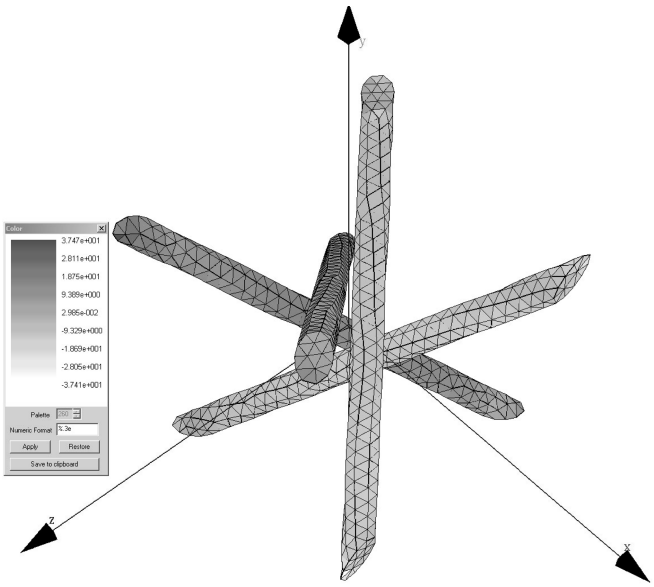


Рис. 4. Распределение псевдонапряжений $\sigma^{\alpha}_{12(13)}$ (МПа) в задаче $Ж_{13}$ для 4DKM

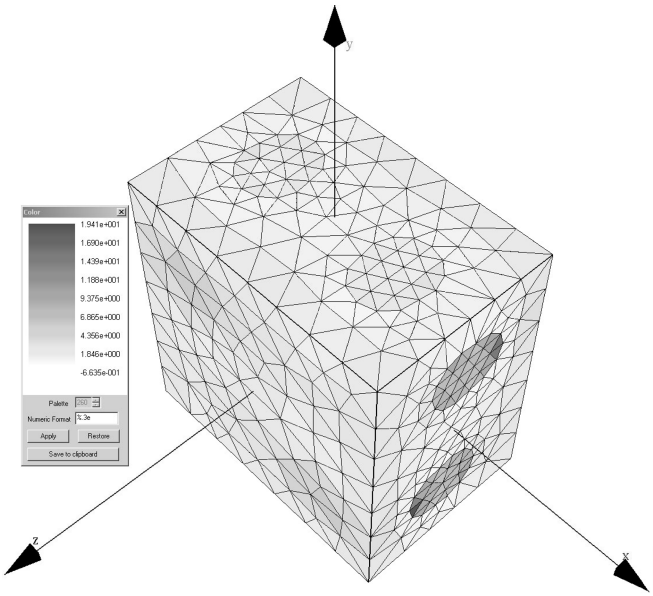


Рис. 5. Распределение псевдонапряжений $\sigma^{\alpha}_{13(13)}$ (МПа) в задаче $Ж_{13}$ для TKKM

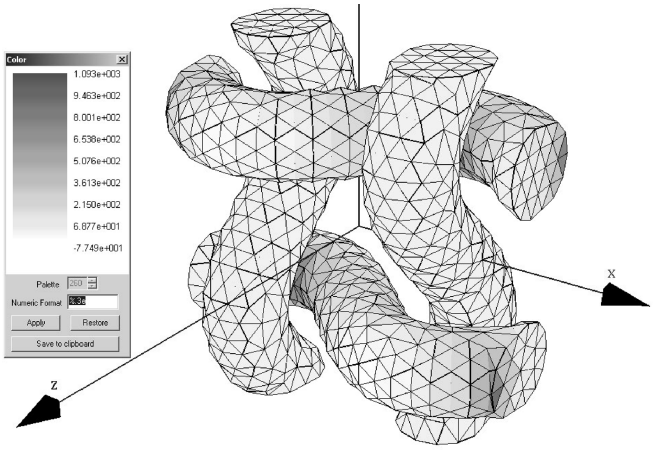


Рис. 6. Распределение псевдонапряжений $\sigma^{\alpha}_{11(11)}$ (МПа) в задаче $Ж_{11}$ для TKKM

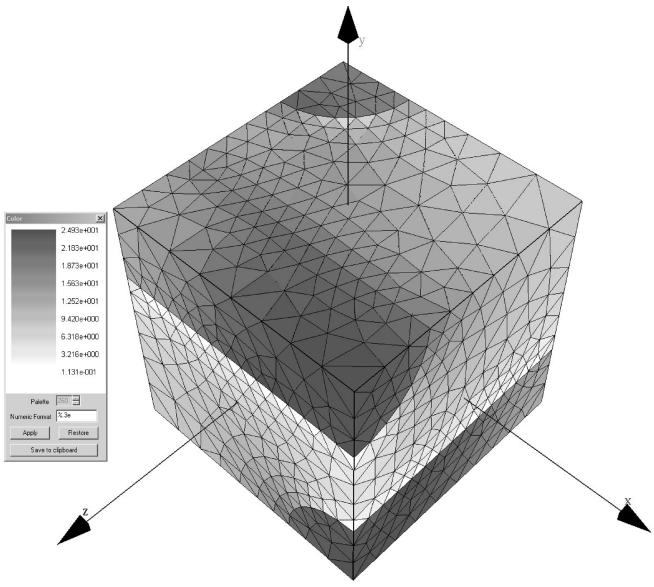


Рис. 7. Распределение псевдонапряжений $\sigma^{\alpha}_{13(13)}$ (МПа) в задаче $Ж_{13}$ для 3DKM

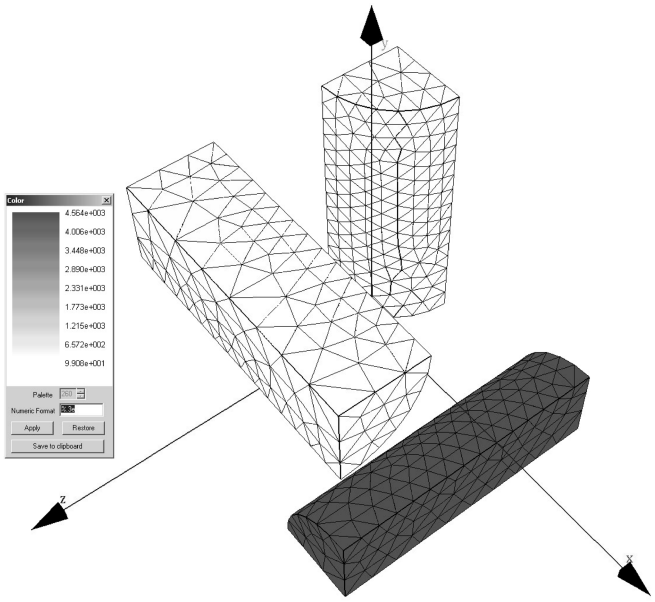


Рис. 8. Распределение псевдонапряжений $\sigma^{\alpha}_{33(33)}$ (МПа) в задаче $Ж_{33}$ для 3DKM

prj_IBDBLink — библиотека с возможностью обеспечения подключения к БД системы.

Стрелками обозначено взаимодействие отдельных подсистем ПК.

Численные результаты

Рассматривались три типа структур армирования композиционного материала с различным числом армирующих волокон (нитей) в ЯП:

2D-композит, армированный системой двух нитей, с тканевым плетением (см. рис. 1, *б*) (TKKM);

3D-ортогонально-армированный композит (см. рис. 1, *а*) (3DKM);

4D-композит, армированный системой четырех нитей, ориентированный по главным диагоналям куба (4DKM).

Все эти структуры освоены в промышленности, для них существуют специальные технологии, позволяющие изготовлять различные элементы конструкций на основе этих композитов [9]. В зависимости от функционального назначения конструкции, ее геометрической формы, заданных упругих и других технических характеристик при проектировании изделий может оказаться более эффективной та или иная структура. Для расчета упруго-прочностных характеристик различных структур КМ с разным содержанием волокон φ_f (которое может варьироваться в различных зонах конструкции) и предназначен разрабатываемый автоматизированный комплекс.

Цель проведенных численных исследований заключалась в демонстрации возможностей разработанного метода расчета и автоматизированного комплекса на примере вычисления эффективных упругих модулей КМ с различными структурами армирования. В табл. 1 представлены сведения о задачах $Ж_{pq}$, которые решаются для различных структур. Число и вид этих задач различны для разных структур КМ — они определяются группой симметрии композита [14]. Так,

Таблица 1

Наборы решенных задач $Ж_{pq}$ для различных структур КМ

№	Тип КМ	Содержание волокон φ_f	Задачи $Ж_{pq}$
1	3DKM	0,45	$Ж_{33}$, $Ж_{13}$, $Ж_{31}$
2	TKKM	0,11	$Ж_{33}$, $Ж_{11}$, $Ж_{12} + Ж_{21}$, $Ж_{13} + Ж_{31}$
3	4DKM	0,04	$Ж_{33}$, $Ж_{11}$, $Ж_{12} + Ж_{21}$, $Ж_{13} + Ж_{31}$

Таблица 2

Эффективные упругие характеристики КМ с различной структурой армирования

Обозначения	3DKM $\varphi_f = 0,45$	4DKM $\varphi_f = 0,04$	TKKM $\varphi_f = 0,11$
Модули Юнга E, ГПа			
E_1	39,758	2,406	6,417
E_2	39,758	2,407	6,645
E_3	39,758	2,381	3,448
Коэффициенты Пуассона ν			
ν_{13}	0,073	0,356	0,451
ν_{31}	0,074	0,361	0,275
ν_{12}	0,073	0,358	0,202
ν_{21}	0,073	0,358	0,114
ν_{23}	0,073	0,364	0,498
ν_{32}	0,0743	0,360	0,187
Модули сдвига G, ГПа			
G_{13}	2,227	1,283	1,296
G_{12}	2,227	1,278	1,094
G_{23}	2,227	1,283	1,296

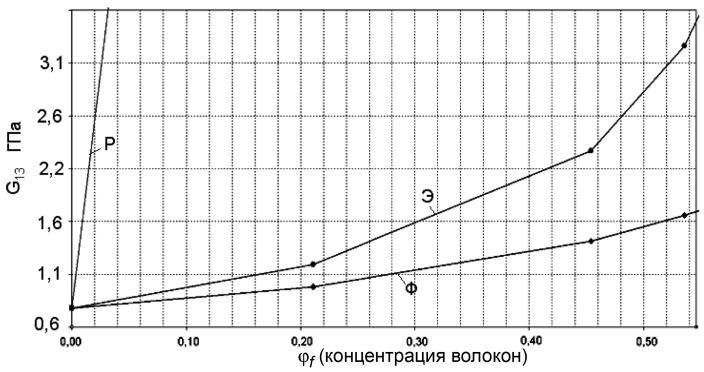


Рис. 9. Модуль сдвига G_{13} композита 3DKM, вычисленный по Фойгту (Ф), Рейссу (Р), и эффективный модуль сдвига, вычисленный по методу асимптотического осреднения

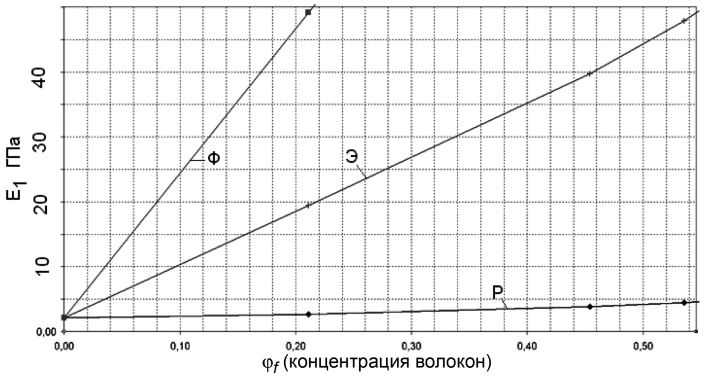


Рис. 10. Модуль Юнга E_1 композита 3DKM, вычисленный по Фойгту (Ф), Рейссу (Р), и эффективный модуль Юнга (Э), вычисленный по методу асимптотического осреднения

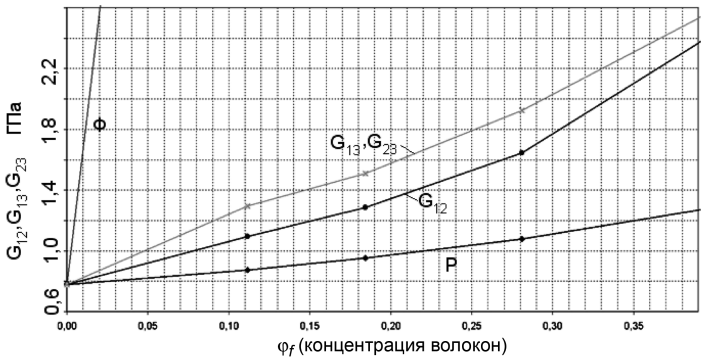


Рис. 11. Модули сдвига тканевого композита ТККМ, вычисленные по Фойгту (Ф), Рейссу (Р), и эффективные модули сдвига G_{12}, G_{13}, G_{23}, вычисленные по методу асимптотического осреднения

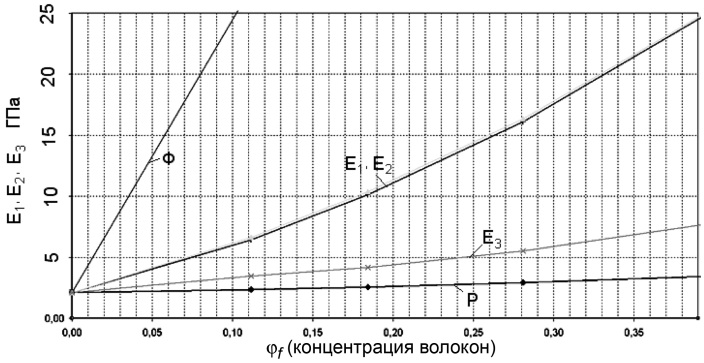


Рис. 12. Модули Юнга тканевого композита ТККМ, вычисленные по Фойгту (Ф), Рейссу (Р), и эффективные модули Юнга E_{12}, E_{13}, E_{23}, вычисленные по методу асимптотического осреднения

3DKM имеет ортотропную симметрию, а если все три системы нитей одинаковы — то кубическую симметрию, а ТККМ и 4DKM всегда имеют лишь ортотропную симметрию. Поэтому для 3DKM необходимо решить только две локальные задачи $Ж_{33}$ и $Ж_{13} + Ж_{31}$, а для ТККМ и 4DKM — четыре задачи: $Ж_{33}$, $Ж_{11}$, $Ж_{12} + Ж_{21}$, $Ж_{13} + Ж_{31}$. Сумма $Ж_{13} + Ж_{31}$ означает, что указанные задачи решались совместно.

Для всех рассмотренных типов КМ решались только указанные выше типы задач, а решения остальных задач $Ж_{pq}$ получались простой перестановкой решений указанных в табл. 1 задач, что возможно за счет типа анизотропии рассматриваемого КМ.

В качестве КЭ использовался четырехузловой тетраэдр.

Для всех исследуемых КМ использовались следующие механические характеристики, которые соответствуют углеродным волокнам и эпоксифенольной матрице:

матрица $E = 2$ ГПа, $\nu = 0{,}35$;

волокна (нити) $E = 250$ ГПа, $\nu = 0{,}25$.

Содержание волокон φ_f для указанных типов КМ было различным, его значение приведено в табл. 1. При расчете задач растяжения $Ж_{pp}$ значения осредненной деформации $\overline{\varepsilon}_{pp}$ выбирались равными 1%.

Некоторые типичные результаты решения локальных задач $Ж_{pq}$ для различных структур КМ приведены на рис. 3—8 в виде фоновых значений компонент тензоров псевдонапряжений $\sigma^{\alpha}_{ij(pq)}$ в ЯП.

В табл. 2 приведены значения эффективных технических констант композитов с различными структурами армирования. На рис. 9—12 представлены графики зависимости эффективных модулей упругости композитов от содержания волокон φ_f. На этих же рисунках приведены верхние и нижние границы эффективных модулей (вилки Фойгта и Рейсса [2]). Вычисления показывают, что вилки Фойгта и Рейсса оказываются очень широкими и ими нельзя пользоваться даже для приближенной оценки эффективных модулей при расчете — этот результат хорошо известен [2], однако надежные методы вычисления точных значений эффективных характеристик композитов со сложными пространственными структурами армирования практически отсутствуют. Разработанный метод и автоматизированный программный комплекс в значительной мере восполняют этот недостаток.

Выводы. В работе разработана методика автоматизированного решения задач "на ячейке периодичности" $Ж_{pq}$ и автоматизированного

вычисления эффективных упругих характеристик КМ с различными структурами армирования. С ее помощью проведены численные расчеты эффективных упругих характеристик для трех различных типов КМ, демонстрирующие широкие возможности методики для автоматизированного прогнозирования свойств новых композиционных материалов с требуемыми характеристиками.

Работа выполнена при поддержке грантов РФФИ № 06-08-01448а и 07-08-00574.

Список литературы

1. **Pagano N. J.** Exact Moduli of Anisotripic Laminates, in Mechanics of Composite Materials. Vol. 2. / Ed. G. P. Sendecky. New York: Academic Press, 1974. P. 38—60.

2. **Chamis C. C.** Micromecahical Failure Theory, in Fracture nad Fatigue. Vol. 5. / Ed. Broutman L. J., Krock R. H. New York: Academic Press, 1974. P. 106—153.

3. **Jones R. M.** Mechanics of Composite Materials. Hemispere, USA, 1975.

4. **Christensen R. M.** Mecanics of Composite Materials. John Wiley & Sons, 1979.

5. **Болотин В. В., Новичков Ю. Н.** Механика многослойных конструкций. М.: Машиностроение, 1980. 376 с.

6. **Ван Фо Фы.** Конструкции из армированных пластмасс. Киев: Техника, 1971. 220 с.

7. **Бахвалов Н. С., Панасенко Г. П.** Осреднение процессов в периодических средах М.: Наука, 1984.

8. **Победря Б. Е.** Механика композиционных материалов. М.: МГУ, 1984. 336 с.

9. **Bensoussan A., Lions J. L., Papanicolau G.** Asymptotic Analysis for Periodic Structures. Amsterdan. North-Holland. 1978.

10. **Димитриенко Ю. И.** Механика композиционного материала при высоких температурах. М.: Машиностроение, 1997.

11. **Li J., Dunn M.** Viscoelectroelastic behavior of heterogeneous piezoelectric solids // J. of Applied Physics. 2001. Vol. 89. N 5. P. 2893—2903.

12. **Liu S., Chen K., Feng X.** Prediction of viscoelastic property of layered materials / Int. J. of Solids and Struct. 2004. Vol. 41. P. 3675—3688.

13. **Maghous S., Creus G. J.** Periodic homogenization in thermoviscoelasticity: case of multilayered media with ageing // Int. J. of Solids and Structures. 2003. Vol. 40, P. 851—870.

14. **Oleinik O. A., Shamaev A. S., Yosifian G. A.** Mathematical problems in elasticity and homogenization. Amsterdan: North-Holland, 1992. 324 p.

15. **Димитриенко Ю. И., Кашкаров А. И.** Конечно-элементный метод для вычисления эффективных характеристик пространственно-армированных композитов // Вестник МГТУ им. Н. Э. Баумана. Сер. Естественные науки. 2002. № 2. С. 95—108.

16. **Димитриенко Ю. И., Кашкаров А. И., Макашов А. А.** Разработка конечно-элементного метода решения локальных задач теории упругости "на ячейке периодичности" для композитов с периодической пространственной структурой // Математика в современном мире / Под ред. Ю. А. Дробышева. Калуга: Изд-во КГПУ, 2004. С. 177—191.

17. **Димитриенко Ю. И., Кашкаров А. И., Макашов А. А.** Конечно-элементное моделирование процесса разрушения пространственно-армированных композитов с периодической структурой // Современные естественно-научные и гуманитарные проблемы. Сб. Трудов конференции "40 лет факультета ФН". М.: Логос. 2004. С. 485—498.

18. **Димитриенко Ю. И., Соколов А. П., Кашкаров А. И.** Разработка конечно-элементного метода решения задач расчета эффективных характеристик композиционных материалов на многопроцессорных вычислительных системах // Аэрокосмические технологии. М.: Изд-во МГТУ им. Н. Э. Баумана, 2004. С. 113—114.

19. **Димитриенко Ю. И., Макашов А. А., Кашкаров А. И., Соколов А. П., Ничеговский Е. С.** Конечно-элементное моделирование эффективных физико-механических характеристик пространственно-армированных композитов // Труды конференции, посвященной 90-летию В. И. Феодосьева, май 2006. М.: Изд-во МГТУ им. Н. Э. Баумана, 2006.

УДК 004.414.2.3:627.13

А. И. Гирча, мл. научн. сотр., НИИ механики МГУ им. М. В. Ломоносова

Разработка программного комплекса для численного моделирования процессов нестационарной гидродинамики методом вязких вихревых доменов

Рассмотрены основные особенности программного комплекса, разработанного на базе численного метода вязких вихревых доменов, для моделирования процессов нестационарной вычислительной гидродинамики. Особое внимание уделено механизмам расширения его функциональных возможностей для решения новых классов задач. Представлены особенности программной реализации. В частности, для повышения гибкости и удобства использования программного комплекса предложен и обсуждается подход на основе совместного использования скриптовых и традиционных языков программирования.

Ключевые слова: *течения вязкой жидкости, вычислительная аэродинамика, численные методы, комплексы программ, расширяемость программного обеспечения, гибкость программного обеспечения, скриптовые языки.*

Введение

Метод вязких вихревых доменов [1—3] представляет собой одну из последних разработок в области дискретных вихревых методов численного моделирования нестационарных отрывных течений вязкой несжимаемой жидкости в лагранжевых координатах. С помощью этого метода удается эффективно исследовать различные сопряженные задачи динамики и аэромеханики. Подходы на его основе с успехом применяются в вычисли-

тельных экспериментах по исследованию механизмов машущего полета, моделированию ветродвигателей роторного и волнового типа, по оптимизации систем стабилизации отрыва на толстых крыльях с помощью вихревых ячеек, оснащенных системами управления с обратной связью.

Численный метод вязких вихревых доменов предназначен для моделирования различных типов плоских течений вязкой среды. Этот метод продолжает активно развиваться. Последнее обстоятельство — одна из причин того, что строго определить область применимости рассматриваемого метода на сегодняшний день не представляется возможным. В частности, данный метод позволяет моделировать следующие типы течений.

1. Внутренние течения в неподвижной замкнутой, ограниченной области произвольной геометрической формы. Такие течения можно рассматривать в качестве моделей течений, получаемых, например, в экспериментах с использованием аэродинамической трубы. Они задаются специальными условиями на границе области: на одних участках границы задается условие вдува среды, на других — отбора. На остальных частях границы задается условие непротекания. Дополнительно в область течения может быть добавлено множество областей, соответствующих профилям обтекаемых тел. Число степеней свободы любого тела может быть равно 0, 1, 2 или 3. Предполагается, что в процессе движения тела не сталкиваются и не выходят за пределы области течения. На границах областей, соответствующих профилям обтекаемых тел, могут быть заданы различные условия (условие непротекания, условие проницаемости, движущиеся стенки). Между областями также могут быть заданы различные дополнительные условия (условия соединения посредством пружины, шарнира). На границах обтекаемых тел, а также в области течения может быть задан набор точек (с законом их движения), соответствующих областям отбора (вдува) среды.

2. Внешние течения в бесконечном пространстве, заполненном неподвижной на бесконечности средой с набором погруженных в нее движущихся областей. Такие течения соответствуют течениям, получаемым в условиях натурного эксперимента. Движущиеся области соответствуют профилям обтекаемых тел. Предположения о характере движения областей, условиях, заданных между ними и на их границах, аналогичны изложенным в предыдущем пункте.

В настоящее время существуют несколько программных реализаций данного численного метода для проведения вычислительных экспериментов по исследованию течений для различных задач обтекания. Однако использование большинства из них является неэффективным, поскольку при разработке данных программных решений не учитывались особенности, связанные с последующим их применением в рамках эксперимента такого типа. Одна из причин неэффективности заключается в монолитном характере архитектуры имеющихся программных реализаций. Данный факт влечет за собой ряд широко известных проблемных вопросов, связанных с поддержкой и расширением функциональности данных программ. Например, добавление возможности расчета нового класса задач обтекания приводит к изменению значительной части их исходного кода. Последнее обстоятельство увеличивает сложность имеющихся программных реализаций, а также нередко приводит к появлению новой монолитной версии старой программы, ориентированной на новый класс задач. Увеличение количества программных реализаций рассматриваемого численного метода и их возрастающая сложность привели к необходимости систематизации существующих программных решений и разработки гибкого универсального программного комплекса. Такой программный комплекс должен предоставлять возможность моделирования различных классов задач обтекания и иметь универсальный механизм расширения собственной функциональности.

Настоящая работа посвящена особенностям программного комплекса (далее по тексту для краткости — инструментария) такого типа. Необходимые свойства программного комплекса достигаются за счет учета соответствующих требований на ранних этапах процесса разработки, к которым относятся этапы анализа и разработки архитектуры программного комплекса. Далее более подробно сформулированы основные требования к разрабатываемому инструментарию, описаны особенности его архитектуры. Показаны архитектурные решения, за счет которых достигаются характеристики, предъявленные в требованиях к программному комплексу. Заключительная часть статьи посвящена описанию особенностей программной реализации разрабатываемого инструментария. Представлен также современный подход повышения гибкости конечной программной реализации за счет использования скриптовых языков.

Автор не стремился к всестороннему и полному описанию разрабатываемого программного решения, поскольку это могло бы привести к значительному увеличению объема статьи. Основной акцент настоящей работы приходится на идеи и подходы, которые, в первую очередь, связаны с построением универсального механизма расширения функциональности разрабатываемого программного комплекса.

Цель работы. Определение объекта и предмета исследования

Далее формулируются основная цель и объект исследования. Приводится множество ролей, в которых может выступать специалист относительно указанного объекта. Предмет исследования определяется посредством указания той роли, которая соответствует характеру проводимой автором работы.

Цель настоящей работы заключается в создании эффективного с практической точки зрения программного комплекса, удовлетворяющего следующим требованиям:

- программный комплекс должен предоставлять возможность моделирования различных классов задач обтекания методом вязких вихревых доменов;
- для классов задач обтекания, которые уже реализованы в программном комплексе, должен быть предусмотрен способ задания каждой из задач без дополнительного кодирования;
- инструментарий должен иметь механизм расширения своей функциональности на новые классы задач обтекания.

В рамках выбранной области исследования можно выделить следующие роли, в которых может выступать специалист, относительно определенного выше инструментария (названия ролей обусловлены произволом авторского выбора):

- механик[1];
- разработчик численных методов;
- системный аналитик.

Опишем особенности каждой из выделенных ролей.

Основной задачей механика является изучение свойств различных явлений, возникающих при обтекании тел. Механик выступает в качестве конечного пользователя разрабатываемого инструментария. Он, как правило, не несет ответственности за разработку последнего, однако обеспечивает осмысленность исходных данных и получаемых результатов с точки зрения их соответствия математическим моделям явлений, которые реализованы в инструментарии.

С разработкой рассматриваемого программного комплекса для моделирования различных задач обтекания тел связаны две оставшиеся роли — разработчика численных методов и системного аналитика. Данные роли сопряжены с различными аспектами процесса разработки. Для прояснения отличий между этими ролями обратимся к энциклопедии по истории философии [4]:

"ФОРМА и СОДЕРЖАНИЕ — *философские категории, традиционно используемые для характеристики отношения между способом организации вещи и собственно материалом, из которого данная вещь состоит...* "

Рассмотрим разрабатываемый инструментарий с точки зрения представленных философских категорий. Данный подход позволяет разделить процесс разработки на два взаимосвязанных направления: разработка содержания инструментария и исследование его формы. Первому направлению соответствует роль разработчика численных методов, второму — роль системного аналитика.

Основной задачей разработчика численных методов является создание алгоритмов численного решения уравнений, описывающих различные типы течений вязкой среды. В рассматриваемом случае такими уравнениями являются классические уравнения динамики и гидродинамики. Процесс разработки численного метода рассматривается как процесс подготовки исходного материала для создания на его базе программного комплекса с требуемыми свойствами. Разработчик численного метода также отвечает за качество создаваемого материала. Под качеством здесь понимаются сложность разработанного численного метода, а также область его применимости.

Работу системного аналитика можно охарактеризовать как изучение формы создаваемого инструментария. Предполагается, что программный комплекс представляет собой систему — большое число взаимосвязанных элементов. Обязанности специалиста, выступающего в данной роли, включают в себя следующие пункты:

- определение и описание основных элементов, составляющих программный комплекс, изучение взаимосвязей между элементами; описание отношения между выделенными сущностями программного комплекса и понятиями рассматриваемого численного метода (описание отношения форма—содержание);
- исследование особенностей разрабатываемого инструментария в более широком контексте, изучение взаимосвязей между ним и элементами "внешнего окружения". Таким образом, здесь инструментарий рассматривается как элемент некоторой более широкой системы и изучаются отношения между ним и другими элементами системы.

Каждый подход к исследованию систем имеет набор характеристик, которые дополнительно накладываются на понятия элемента и взаимосвязи системы. Применение различных подходов к анализу и разработке систем приводит к созданию систем различных типов.

В настоящей работе приняты следующие соглашения о характере элементов программного

[1] Под механиком здесь и далее подразумевается специалист в области механики жидкости и газа, владеющий математическими моделями, описывающими физическое явление обтекания тел.

комплекса и взаимосвязей между ними. Элементы представляют собой пассивные хранилища данных, называемые далее также типами данных. Обработку данных осуществляют активные компоненты программного комплекса. Между ними распределена функциональность разрабатываемого инструментария. Такие компоненты рассматриваются как взаимосвязи между элементами системы (типами данных). Активные компоненты программного комплекса именуются также модулями.

Относительно разрабатываемого программного комплекса автор выступает в роли системного аналитика.

Общая схема численного метода вязких вихревых доменов

Численный метод вязких вихревых доменов допускает ряд вариаций для моделирования различных течений вязкой жидкости. В силу отмеченного обстоятельства его можно рассматривать в виде совокупности вычислительных схем, каждая из которых соответствует одной из задач обтекания. Для определения понятия общей схемы выбранного численного метода рассмотрим следующее конечное множество с операцией. В качестве такого множества примем совокупность вычислительных схем метода, в качестве операции — операцию перехода между различными задачами обтекания. Под общей схемой численного метода будем понимать набор свойств его вычислительных схем, инвариантных относительно введенной операции. Данный набор свойств будет служить основой построения механизма расширения функциональности разрабатываемого программного комплекса.

К инвариантным характеристикам различных вычислительных схем численного метода вязких вихревых доменов, в первую очередь, относятся понятия, используемые в методе. Данный набор понятий включает в себя следующие элементы: понятия вихря и вектора, понятия обтекаемого тела, среды, матрицы действительных чисел.

Общая схема метода также включает в себя инвариантный набор этапов вычислений, характерный для различных вычислительных схем выбранного численного метода. Представим основные подшаги одного шага по времени рассматриваемого численного метода с указанием основных элементов данного набора.

1. Решение системы линейных уравнений, в результате которого находятся различные характеристики, необходимые для удовлетворения граничных условий на обтекаемых телах (и внешней границе области течения для задач внутреннего обтекания). Дополнительные условия на обтекаемых телах (число степеней свободы, характер взаимосвязей между телами) приводят к включению дополнительных уравнений в данную систему на их кинематические характеристики (например, вектор скорости, угловая скорость обтекаемых тел). Данному подшагу алгоритма соответствуют три элемента инвариантного набора вычислений: этап заполнения матрицы системы и ее правой части; этап решения системы линейных уравнений; этап извлечения результатов из полученного решения. Вычислительные схемы, соответствующие различным задачам обтекания, реализуют данные этапы различными способами.

2. Этап, соответствующий операции объединения (и расщепления) элементов, моделирующих среду (свободных вихрей). Данный подшаг осуществляется с целью получения более регулярного распределения свободных вихрей. Он необходим для последующих вычислений, в частности, для вычисления диффузионной скорости свободных вихрей. Различным вычислительным схемам численного метода соответствуют различные критерии регуляризации.

3. Нахождение множества векторов скорости каждого из элементов, моделирующих среду. Данному подшагу соответствуют два элемента инвариантного набора: вычисление конвективной и вычисление диффузионной скоростей свободных вихрей. Различным вычислительным схемам численного метода вязких вихревых доменов соответствуют различные способы нахождения перечисленных характеристик. Конвективную составляющую скорости каждого свободного вихря можно рассматривать как сумму двух слагаемых, первое из которых соответствует влиянию остальных свободных вихрей на рассматриваемый, второе — влиянию элементов, расположенных на границе обтекаемых тел (для моделирования соответствующих граничных условий). Нахождение первых слагаемых конвективной составляющей векторов скоростей свободных вихрей представляет собой так называемую гравитационную задачу N тел [5], где в качестве закона взаимодействия выступает функция, являющаяся ядром в законе Био-Савара. Диффузионная составляющая векторов скорости вычисляется с помощью некоторой функции, полученной в результате преобразований уравнений Навье—Стокса.

4. Перемещение среды на следующий шаг по времени.

5. Перемещение обтекаемых тел на следующий шаг по времени с использованием характеристик, найденных на первом подшаге алгоритма.

Архитектура программного комплекса

Архитектура программного комплекса должна обладать свойствами, соответствующими предъ-

явленным требованиям. Основное внимание в данной работе уделено особенностям реализации нефункциональных требований к программному комплексу. Необходимо отметить, что характер требований такого типа не позволяет провести строгое доказательство наличия соответствующих свойств в предлагаемых архитектурных решениях. Далее представлены основные идеи, направленные на достижение требуемых качественных характеристик программного комплекса.

Архитектура разрабатываемого инструментария состоит из двух уровней: уровня данных и уровня модулей. Элементы уровня данных соответствуют выделенным компонентам численного метода вязких вихревых доменов. Взаимоотношения между выделенными компонентами метода формализованы в виде элементов уровня модулей (в виде модулей).

Элементы уровня данных представляют собой формализацию основных понятий, использующихся в методе. Каждому понятию соответствует тип данных. Множество типов данных программного комплекса будем обозначать $\mathbb{D}$.

Для определения понятия модуля программного комплекса рассмотрим множество T всех задач обтекания, которые (на сегодняшний день) позволяет моделировать численный метод вязких вихревых доменов. Как было отмечено ранее, всем вычислительным схемам, соответствующим данному множеству, характерен общий набор этапов вычислений. На каждом этапе осуществляется изменение наборов данных в соответствии с алгоритмом численного метода. Таким образом, этап вычисления любой вычислительной схемы можно рассматривать как функцию вида

$$f: A \to B;\ A \in \mathbb{D}^m,\ B \in \mathbb{D}^n.$$

Зафиксируем некоторый этап вычисления α. В самом общем случае он определяет некоторое множество функций

$$F^\alpha = \{f_t^\alpha: A_t \to B_t;\ A_t \to \mathbb{D}^{m_t},\ B_t \in \mathbb{D}^{n_t},\ \forall t \in T\},$$

соответствующих рассматриваемому этапу всех вычислительных схем множества T задач обтекания (множество F^α можно рассматривать как срез всех вычислительных схем по выбранному этапу α). Множество F^α можно разбить на пересекающиеся подмножества F_k^α. Каждое подмножество определяется некоторой параметризованной функцией

$$m_k^\alpha : A_k \times \mathrm{PARAM} \to B_k;\ A_k \in \mathbb{D}^{m_k},\ B_k \in \mathbb{D}^{n_k},$$

где PARAM — множество параметров. Таким образом,

$$F_k^\alpha = \{f_t^\alpha \in F^\alpha : \exists p_t \in \mathrm{PARAM},\ m_k^\alpha(p_t) = f_t^\alpha\}.$$

Функции m_k^α будем называть модулями. Заметим, что по множеству F_k^α однозначно определяется множество задач $T_k \subset T$, для которых возможно применение данного модуля на выбранном этапе вычислений α. Таким образом, для любого этапа вычислений α множеству модулей $\mathbb{M}^\alpha$ данного этапа соответствует некоторое покрытие множества T задач обтекания его подмножествами.

Программный комплекс предполагает следующий алгоритм использования. Для решения той или иной задачи обтекания необходимо выяснить, для каждого ли этапа ее основного шага моделирования по времени существует модуль, реализующий необходимую функциональность. Если все требуемые модули реализованы в программном комплексе, то для решения выбранной задачи можно использовать соответствующий набор модулей, в противном случае — необходимо реализовать недостающие модули. Таким образом, с течением времени число реализованных модулей будет расти, и, следовательно, программный комплекс можно будет применять для более широкого круга задач обтекания тел.

Модуль представляет собой обобщенный подшаг моделирования для некоторого класса задач обтекания. Отсюда следует, что при соответствующей его программной реализации требование об отсутствии дополнительного кодирования при решении задач обтекания существующими модулями программного комплекса выполняется автоматически.

Особенности программной реализации и интерфейс

В качестве языка для программной реализации инструментария был выбран язык *C++*. Все элементы множеств типов данных и модулей реализованы в виде классов выбранного языка. Каждому элементу соответствует свой класс. Конечная реализация инструментария представляет собой две динамически подгружаемые библиотеки (по одной для множества типов данных и множества модулей) и набор заголовочных файлов языка *C++* (*.h*-файлов) с описаниями интерфейсов классов, соответствующих элементам упомянутых множеств. С технической точки зрения инструментарий предполагает стандартный способ использования, заключающийся во включении необходимых заголовочных файлов в исходные тексты разрабатываемых программ с последующим связыванием результирующего исполняемого файла с необходимыми библиотеками.

Для повышения гибкости работы с инструментарием для всех его компонентов с помощью библиотеки *Boost Python* [6] разработан специальный код, осуществляющий их привязку к скриптовому

Число шагов по времени	Время работы, с *C++*	Время работы, с C++/*Python*
50	197	205
100	500	503

языку *Python*. Использование в качестве интерфейса к инструментарию языка *Python* дает следующие преимущества.

- Разработку и запуск приложений можно осуществлять в интерактивном режиме с возможностью "на лету" изменять различные параметры моделирования.
- Скриптовый язык можно рассматривать как средство для интеграции разработанного инструментария с другими библиотеками, также имеющими интерфейс к данному языку. Здесь, в первую очередь, подразумевается интеграция с библиотеками, реализующими различные вспомогательные функции: предварительную обработку данных, визуализацию данных, передачу данных по сети.

Применение языка *Python* в качестве программируемого интерфейса к разработанному инструментарию незначительно сказывается на производительности вычислений. Сравнительная характеристика времени счета соответствующих запусков инструментария с интерфейсом и без него для одной из задач обтекания приведена в таблице.

Заключение

Представлены особенности программного комплекса для проведения вычислительных аэродинамических экспериментов на базе численного метода вязких вихревых доменов. Основное внимание уделено вопросам расширяемости разрабатываемого инструментария. В настоящее время программный комплекс представляет собой набор модулей, ориентированных на решение одного из классов задач внутренней аэродинамики. Данный набор модулей используется при проведении исследований в рамках европейского проекта VortexCell2050 [7] для определения полезных свойств вихревых ячеек. Эти свойства, например способность предотвращать отрыв с гладкой поверхности крылового профиля, могут быть использованы при проектировании летательных аппаратов нового типа.

Список литературы

1. **Дынникова Г. Я.** Лагранжев подход к решению нестационарных уравнений Навье—Стокса // Доклады Академии наук. — 2004. — Т. 399. № 1. — С. 42—46.

2. **Андронов П. Р., Гувернюк С. В., Дынникова Г. Я.** Вихревые методы расчета давлений и нестационарных нагрузок при движении тел в сплошной среде. — М.: Изд-во МГУ, 2006.

3. **Гувернюк С. В., Дынникова Г. Я.** Моделирование обтекания колеблющегося профиля методом вязких вихревых доменов // Изв. РАН МЖГ. 2007. — № 1.

4. **Определение** понятий формы и содержания // Энциклопедия по истории философии. — http: //velikanov. ru/philosophy/ soderzhanie_ i_forma.asp.

5. **Rankin III W. T.** Efficient parallel implementations of multi pole based N-body algorithms. — 1999. — April. — http://www.ee.duke.edu/~wrankin/disser/rankin_dissertation.pdf.

6. **Boost** Python library documentation. — http://www.boost.org/ libs/ python/doc/.

7. **VortexCen2050** project homesite. — http://www.vortexcell2050.org.

УДК 620.179.14

В. Я. Гальченко, д-р техн. наук, проф., зав. каф., **Д. Л. Остапущенко,** аспирант,
Луганский государственный медицинский университет

Численный анализ пространственной конфигурации магнитных полей объектов сложной геометрической формы с учетом нелинейных характеристик веществ

Рассматривается программное обеспечение, разработанное авторами, позволяющее проводить расчет поля в открытых и замкнутых расчетных областях магнитных систем, содержащих ферромагнитные детали сложной геометрической формы, с учетом нелинейных магнитных свойств.

Ключевые слова: *программное обеспечение, магнитное поле, магнитная система, ферромагнетик, нелинейные магнитные характеристики, пространственные интегральные уравнения, адаптивная дискретизация, численное интегрирование, системы линейных и нелинейных уравнений, метод Ньютона.*

Магнитные системы, включающие в себя элементы, изготовленные из ферромагнитных материалов, широко применяются в различных областях техники. Примерами устройств, содержащих такие магнитные системы, могут служить устройства электронной и ионной оптики, ускорительной техники, магнитной дефектоскопии и структуроскопии, различные электротехнические уст-

ройства, датчики. При разработке таких устройств возникает необходимость проводить анализ пространственной конфигурации статического магнитного поля как в объеме ферромагнитных тел, так и в окружающем пространстве. Наиболее удобным и гибким инструментом для этих целей является компьютерное моделирование. При его использовании имеется возможность варьирования в широком диапазоне геометрии магнитной системы, выбора различных магнитных характеристик материалов и получения в результате расчета исчерпывающей информации о конфигурации магнитного поля. Особенно важно то, что, проведя расчеты для различных вариантов конструкции магнитной системы, существует возможность выбрать ее наилучший вариант.

Наиболее перспективным методом анализа магнитного поля, особенно в открытых расчетных областях магнитных систем, является метод пространственных интегральных уравнений (ПрИУ) [1, 2].

Существует много примеров использования ПрИУ для решения различных технических задач. Однако существующее на данный момент подобное программное обеспечение имеет ряд существенных недостатков, ограничивающих возможность его применения для анализа магнитных систем, содержащих детали со сложной геометрической формой, что значительно сужает круг практических задач, решаемых с использованием таких программ. Так, модели, описанные в работах [3—6], при дискретизации объектов допускают применение элементарных объемов только в виде параллелепипедов, грани которых параллельны плоскостям системы координат, или кольцеобразных областей для представления аксиально-симметричных тел, что существенно упрощает интегрирование при численных экспериментах и позволяет реализовать его аналитически. При этом анализу подвергается не геометрическая модель магнитной системы, близкая к реальности, а некоторое ее грубое приближение. Такое излишне идеализированное представление геометрии объектов обусловливает узкую специализацию программного обеспечения. Целью настоящей работы является описание программного обеспечения, разработанного авторами и не имеющего подобных ограничений.

При построении математической модели с использованием ПрИУ [1] объем тела, помещенного в магнитное поле с напряженностью $\boldsymbol{H}_0$, создаваемое внешними источниками, разбивается на элементарные объемы, т. е. представляется в виде теоретико-множественного объединения N геометрических объектов более простой формы, пересекающихся только по границам. В пределах элементарного объема намагниченность считается постоянной. Каждый элементарный объем создает в пространстве магнитное поле, напряженность которого можно определить по формуле

$$\boldsymbol{H}(Q) = \frac{1}{4\pi} \oiint\limits_{\partial V} \frac{(\boldsymbol{M} \cdot \boldsymbol{n}_p)\boldsymbol{r}_{PQ}}{r_{PQ}^3} dS_P, \qquad (1)$$

где Q — точка наблюдения; P — точка истока; $\boldsymbol{r}_{PQ} = \boldsymbol{r}_Q - \boldsymbol{r}_P$ — вектор, направленный из точки P в точку Q; $\boldsymbol{n}_P$ — вектор внешней нормали к границе ∂V элементарного объема V в точке истока; $\boldsymbol{M}$ — намагниченность элементарного объема. Напряженность магнитного поля в присутствии ферромагнитного тела может быть определена по формуле

$$\boldsymbol{H}(Q) = \sum_{j=1}^{N} \boldsymbol{H}_j(Q) + \boldsymbol{H}_0(Q), \qquad (2)$$

где $\boldsymbol{H}_j(Q)$ — напряженность, создаваемая в точке Q j-м элементарным объемом. С учетом (1) выражение (2) может быть представлено в виде

$$\boldsymbol{H}(Q) = \frac{1}{4\pi} \sum_{j=1}^{N} \oiint\limits_{\partial V_j} \frac{(\boldsymbol{M}_j \cdot \boldsymbol{n}_p)\boldsymbol{r}_{PQ}}{r_{PQ}^3} dS_P + \boldsymbol{H}_0(Q), \quad (3)$$

где $\boldsymbol{M}_j$ — намагниченность j-го элементарного объема. Поочередно помещая точку наблюдения в центры элементарных объемов $\{Q_j\}_{j=1}^{N}$, можно получить

$$\boldsymbol{H}_i = \frac{1}{4\pi} \sum_{j=1}^{N} \oiint\limits_{\partial V_j} \frac{(\boldsymbol{M}_j \cdot \boldsymbol{n}_p)\boldsymbol{r}_{PQ_j}}{r_{PQ_j}^3} dS_P + \boldsymbol{H}_{i0},\ i = \overline{1, N}. \quad (4)$$

Выполнив интегрирование, выражение (4) можно представить в виде

$$\boldsymbol{H}_i = \sum_{j=1}^{N} \mathbf{A}_{ij}\boldsymbol{M}_j + \boldsymbol{H}_{i0},\ i = \overline{1, N}, \qquad (5)$$

где $\mathbf{A}_{ij}$ — матрица размером 3 × 3, учитывающая вклад намагниченности j-го элементарного объема в напряженность в i-м. Вводя следующие обозначения:

$$\mathbf{H} = \begin{pmatrix} \boldsymbol{H}_1 \\ \boldsymbol{H}_2 \\ \dots \\ \boldsymbol{H}_N \end{pmatrix},\ \mathbf{H}_0 = \begin{pmatrix} \boldsymbol{H}_{10} \\ \boldsymbol{H}_{20} \\ \dots \\ \boldsymbol{H}_{N0} \end{pmatrix},\ \mathbf{M} = \begin{pmatrix} \boldsymbol{M}_1 \\ \boldsymbol{M}_2 \\ \dots \\ \boldsymbol{M}_N \end{pmatrix},$$

$$\mathbf{A} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1N} \\ A_{21} & A_{22} & \dots & A_{2N} \\ \dots & \dots & \dots & \dots \\ A_{N1} & A_{N2} & \dots & A_{NN} \end{pmatrix}, \qquad (6)$$

выражение (5) можно представить в виде

$$\mathbf{H} = \mathbf{A} \cdot \mathbf{M} + \mathbf{H}_0. \quad (7)$$

В предположении отсутствия гистерезиса существует однозначная, но, вообще говоря, нелинейная зависимость между намагниченностью j-го элементарного объема и напряженностью магнитного поля в нем

$$\boldsymbol{M} = F_j(\boldsymbol{H}). \quad (8)$$

В общем случае характер этой зависимости может изменяться от элементарного объема к элементарному объему. Введем в рассмотрение нелинейный оператор, определенный по правилу

$$\mathbf{M} = F(\mathbf{H}) = \begin{pmatrix} F_1(\boldsymbol{H}_1) \\ F_2(\boldsymbol{H}_2) \\ \ldots \\ F_N(\boldsymbol{H}_N) \end{pmatrix}. \quad (9)$$

С его использованием (7) может быть представлено в виде нелинейного уравнения

$$\mathbf{H} = \mathbf{A} \cdot F(\mathbf{H}) + \mathbf{H}_0. \quad (10)$$

Решением этого уравнения являются матрица $\mathbf{H}$, содержащая напряженности магнитного поля в элементарных объемах. С использованием оператора (9) они могут быть пересчитаны в намагниченности. Для расчета поля в интересующих точках пространства достаточно воспользоваться формулой (3). Такая математическая модель применима для расчета поля как односвязных, так и многосвязных объектов.

Решение прямой задачи магнитостатики с использованием ПрИУ предполагает выполнение следующих этапов:

1) подготовка исходных данных;

2) разбиение (дискретизация) ферромагнитных тел на элементарные объемы;

3) вычисление элементов матрицы $\mathbf{A}$ уравнения (10);

4) вычисление элементов матрицы $\mathbf{H}_0$, содержащей компоненты напряженности магнитного поля, создаваемого внешними источниками;

5) решение нелинейного уравнения (10);

6) вычисление напряженности магнитного поля в интересующих точках пространства.

На этапе подготовки исходных данных в компьютер заносится информация о геометрии областей пространства, заполненных ферромагнетиком, магнитных характеристиках веществ, интенсивности и месторасположении в пространстве источников магнитного поля (катушек с постоянным током, постоянных магнитов и др.).

Несмотря на кажущуюся простоту, этап проведения разбиений является одним из важнейших, и от качества его проведения в значительной мере зависит эффективность анализа, а в конечном итоге — адекватность полученного результата. При разбиении областей пространства, заполненных ферромагнетиком, в качестве элементарных объемов удобнее всего использовать трехмерные симплексы — тетраэдры. Любая многогранная область может быть представлена в виде теоретико-множественного объединения тетраэдров, которые не пересекаются или точки пересечения которых образуют их общую грань [7]. Границы области произвольной формы приближаются многогранными поверхностями, гранями которых являются треугольники. Любая достаточно гладкая поверхность может быть приближена таким образом с произвольной точностью. В различных частях разбиваемой области возможно использование тетраэдров различного размера, т. е. использование адаптивного разбиения. Необходимость использования адаптивного разбиения может быть вызвана рядом причин. При наличии у разбиваемой области элементов, размеры которых малы по сравнению с размерами самой области, приходится уменьшать размер тетраэдров вблизи таких элементов в целях более детального их представления. Такая же необходимость возникает при наличии у области сильно искривленных участков границы. В целях уменьшения числа неизвестных в системе нелинейных уравнений, а также для снижения возможных погрешностей следует использовать априорную информацию о характере распределения намагниченности в разбиваемой области. В частях области с большой скоростью изменения намагниченности от точки к точке следует использовать тетраэдры меньшего размера по сравнению с частями, где ожидается плавное ее изменение. Чрезмерно крупные тетраэдры приводят к высоким погрешностям расчета, а мелкие к значительному возрастанию числа неизвестных в системе уравнений. Использование адаптивного разбиения позволяет найти компромисс в этой ситуации. Априорная информация о характере распределения намагниченности может быть получена из опыта проведения расчетов. Пример дискретизации, выполненной с учетом сформулированных требований, приведен на рис. 1.

Матрица $\mathbf{A}$ формируется в виде файла на жестком диске, при этом ее элементы записываются в порядке их расположения в строках.

Вычисление элементов проводится поблочно. Это позволяет хранить в оперативной памяти только три строки матрицы $\mathbf{A}$, которые после заполнения дописываются в файл.

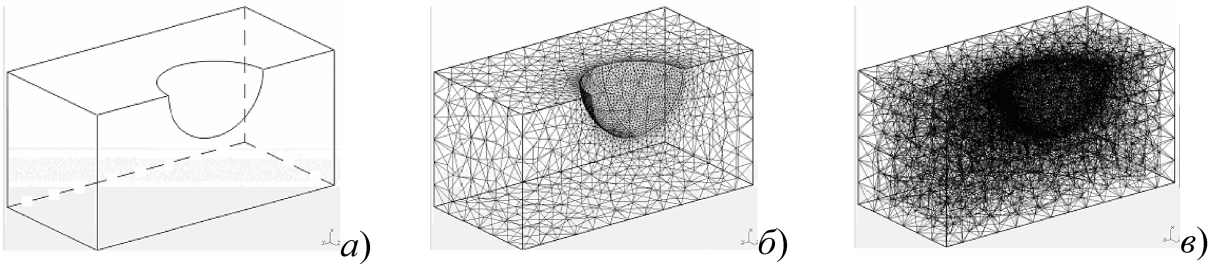


Рис. 1. Этапы проведения процедуры дискретизации:
а — исходная геометрия тела; *б* — дискретизация поверхности; *в* — дискретизация объема

Рассмотрим расчет элементов блока матрицы **A**. Формула

$$\boldsymbol{H} = \frac{1}{4\pi} \oiint\limits_{\partial V} \frac{(\boldsymbol{M} \cdot \boldsymbol{n}) \boldsymbol{r}_{PQ}}{\boldsymbol{r}_{PQ}^3} \, dV_P \qquad (11)$$

выражает линейную связь между векторами $\boldsymbol{M}$ и $\boldsymbol{H}$, которая может быть записана в виде

$$\boldsymbol{H} = \mathbf{A} \cdot \boldsymbol{M}, \qquad (12)$$

где матрица **A** в случае, когда в качестве элементарного объема выступает тетраэдр, представляется следующим образом:

$$\mathbf{A} = \begin{pmatrix} \sum\limits_{k=1}^{4} n_{xk} I_{xk} & \sum\limits_{k=1}^{4} n_{yk} I_{xk} & \sum\limits_{k=1}^{4} n_{zk} I_{xk} \\ \sum\limits_{k=1}^{4} n_{xk} I_{yk} & \sum\limits_{k=1}^{4} n_{yk} I_{yk} & \sum\limits_{k=1}^{4} n_{zk} I_{yk} \\ \sum\limits_{k=1}^{4} n_{xk} I_{zk} & \sum\limits_{k=1}^{4} n_{yk} I_{zk} & \sum\limits_{k=1}^{4} n_{zk} I_{zk} \end{pmatrix}, \qquad (13)$$

где

$$I_k = \frac{1}{4\pi} \iint\limits_{\Delta_k} \frac{\boldsymbol{r}_{PQ}}{\boldsymbol{r}_{PQ}^3} \, dS_P, \qquad (14)$$

где интеграл вычисляется по k-й грани тетраэдра ($\partial V = \bigcup\limits_{k=1}^{4} \Delta_k$). При переходе в локальную систему координат, связанную с гранью тетраэдра, интеграл по треугольной области преобразуется в криволинейный, вычисление которого проводится численно, с использованием адаптивного квадратурного алгоритма, основанного на формуле Ньютона—Котеса восьмого порядка [8].

Для контроля точности вычисления интегралов найдем сумму диагональных элементов матрицы **A**:

$$\mathrm{Sp}\mathbf{A} = \sum_{k=1}^{4} n_{xk} I_{xk} + \sum_{k=1}^{4} n_{yk} I_{yk} + \sum_{k=1}^{4} n_{zk} I_{zk} =$$

$$= \sum_{k=1}^{4} \boldsymbol{n}_k \boldsymbol{I}_k = \frac{1}{4\pi} \sum_{k=1}^{4} \iint\limits_{\Delta_k} \frac{\boldsymbol{r}_{PQ} \cdot n_k}{\boldsymbol{r}_{PQ}^3} \, dS_P =$$

$$= \frac{1}{4\pi} \oiint\limits_{\partial V} \frac{\boldsymbol{r}_{PQ}}{\boldsymbol{r}_{PQ}^3} \, dS_P.$$

Для интеграла Гаусса известно [9]

$$\oiint\limits_{\partial V} \frac{\boldsymbol{r}_{QP}}{\boldsymbol{r}_{QP}^3} \, d\boldsymbol{S}_P = \begin{cases} 4\pi, \; Q \in V; \\ 0, \; Q \notin V. \end{cases}; \qquad (15)$$

Откуда следует

$$\mathrm{Sp}\mathbf{A} = \begin{cases} -1, \; Q \in V; \\ 0, \; Q \notin V. \end{cases}; \qquad (16)$$

При вычислении элементов матрицы $\mathbf{H}_0$ необходимо проводить расчет напряженности магнитного поля различных источников. Программный комплекс содержит библиотеку моделей источников магнитного поля. Часто магнитные системы включают в себя катушки с постоянным током. Расчет поля круглой катушки с прямоугольным сечением подробно рассматривается в работах [10, 11]. В работе [10] также приведены зависимости для расчета поля некоторых идеализированных источников, таких как, например, контур с током, плоская катушка и тонкий соленоид. Некоторые катушки сложной формы удобно разделить на сегменты в виде параллелепипеда или прямой призмы, тогда поле катушки представляет собой суперпозицию полей сегментов. Формулы для вычисления полей таких сегментов приведены в работе [12]. Методика расчета поля кольцеобразного постоянного магнита заимствована из работы [13], полосового магнита — из работы [14].

Для вычисления значений нелинейного оператора $\mathbf{F}(\mathbf{H})$ должны быть заданы магнитные характеристики веществ $\boldsymbol{M} = F_j(\boldsymbol{H})$. Программный комплекс включает в себя библиотеку аналитических зависимостей для описания нелинейных свойств ферромагнетиков [15—17]. Также предусмотрена возможность учета анизотропии [2].

Для решения системы нелинейных уравнений применяется метод Ньютона [18]. Вводя матрицу-столбец $\mathbf{G}(\mathbf{H}) = \mathbf{A} \cdot \mathbf{F}(\mathbf{H}) - \mathbf{H} + \mathbf{H}_0$, перепишем уравнение (10) в виде

$$\mathbf{G}(\mathbf{H}) = 0. \qquad (17)$$

Матрица Якоби для $\mathbf{G}(\mathbf{H})$ будет иметь следующий вид:

$$\mathbf{W} = \mathbf{AJ} - \mathbf{E}, \tag{18}$$

где $\mathbf{J}$ — матрица Якоби для $\mathbf{F}(\mathbf{H})$; $\mathbf{E}$ — единичная матрица. В силу структуры нелинейного оператора $\mathbf{F}(\mathbf{H})$ матрица $\mathbf{J}$ состоит из блоков размером 3×3, расположенных по диагонали:

$$\mathbf{J} = \begin{pmatrix} J_1 & 0 & \ldots & 0 \\ 0 & J_2 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots & J_N \end{pmatrix}, \tag{19}$$

где J_j — матрица Якоби для $F_j(\boldsymbol{H})$.

Итерационный процесс организуется по схеме

$$\mathbf{H}_n = \mathbf{H}_{n-1} - \mathbf{h}, \tag{20}$$

где $\mathbf{h}$ — решение системы линейных уравнений

$$\mathbf{Wh} = \mathbf{G}(\mathbf{H}_{n-1}). \tag{21}$$

Матрица $\mathbf{W}$, так же как матрица $\mathbf{A}$, формируется на жестком диске. Наличие у матрицы $\mathbf{J}$ блочной структуры существенно упрощает процесс вычисления произведения $\mathbf{A} \cdot \mathbf{J}$.

Итерационный процесс, по всей видимости, является сходящимся для любого начального приближения. Однако для уменьшения числа итераций необходимо использовать априорную информацию о характере распределения напряженности или намагниченности. В качестве условия остановки итерационного процесса можно принять условие $\|\mathbf{h}\|_\infty < \varepsilon$, где ε — положительное число, значительно меньшее, чем $\|\mathbf{H}_n\|_\infty$.

В силу особенностей метода на каждой итерации приходится решать систему линейных уравнений с несимметричной заполненной матрицей большого порядка (50 000 и более). Подобная задача не является тривиальной и требует применения специфических подходов. Решение линейной системы уравнений (16) проводится с использованием специальной реализации метода Гаусса, в которой не требуется размещать матрицу системы в оперативную память целиком, а достаточно считывать с жесткого диска блок, состоящий из k строк, размер которого выбирается исходя из объема доступной памяти. Считанный блок приводится к трапециевидной форме. Оставшиеся строки матрицы последовательно считываются в оперативную память, где с ними проводятся преобразования, в результате которых k первых элементов строки обращаются в ноль, затем они сохраняются в новый файл. Такая процедура повторяется необходимое число раз. На каждом этапе блок, приведенный к трапециевидной форме, дописывается в отдельный файл. В результате в этом файле сформируется треугольная матрица, полученная из исходной посредством преобразований Гаусса. Для решения системы уравнений с треугольной матрицей, размещенной на диске, достаточно считывать с конца файла небольшими порциями элементы матрицы, последовательно вычисляя значения неизвестных.

Для контроля точности решения системы линейных уравнений предусмотрен ряд следующих мероприятий:

- система уравнений (16) решается одновременно с несколькими столбцами свободных членов, один из которых — $\mathbf{G}(\mathbf{H}_{n-1})$, а остальные генерируются так, чтобы соответствующие им решения были заранее известны, и точность их нахождения можно было проконтролировать;
- проводятся вычисление и оценка нормы невязки;
- оценивается число обусловленности матрицы.

Верификация работы программного обеспечения проводилась на задачах, имеющих аналитическое решение.

1. Ферромагнитный шар, помещенный в однородное поле. Известным является факт его однородного намагничивания. Для напряженности магнитного поля внутри шара радиусом R с центром в точке P, помещенного в однородное поле напряженностью $\boldsymbol{H}_0$, согласно [19] имеем

$$\boldsymbol{H} = \frac{3\mu_0}{\mu + 2\mu_0} \boldsymbol{H}_0, \tag{22}$$

где $\mu = \mu_0 \mu_r$ — абсолютная и μ_r — относительная магнитные проницаемости. Откуда для намагниченности получаем

$$\boldsymbol{M} = \frac{3\chi}{3 + \chi} \boldsymbol{H}_0, \tag{23}$$

где $\chi = \mu_r + 1$ — магнитная восприимчивость вещества.

Во внешнем пространстве в точке Q поле шара совпадает с полем магнитного диполя, расположенного в центре шара:

$$\boldsymbol{H}(Q) = \frac{1}{4\pi} \left(\frac{\boldsymbol{p}}{r_{PQ}^3} - \frac{3(\boldsymbol{p} \cdot \boldsymbol{r}_{PQ}) \boldsymbol{r}_{PQ}}{r_{PQ}^5} \right), \tag{24}$$

магнитный момент которого совпадает с магнитным моментом шара

$$\boldsymbol{p} = \frac{4}{3} \pi R^3 \boldsymbol{M}. \tag{25}$$

Из формул (24) и (25) следует, что расчет может проводиться с использованием относительных единиц длины. Численный эксперимент проводился для шара $R = 1$, помещенного в однородное поле, направленное вдоль оси z напряженностью

100 А/м. Магнитная восприимчивость вещества $\chi = 100$. Объем шара разбивался на 15 899 элементарных объемов. Результат сопоставления численного и аналитического решения приведен на рис. 2. Расчет поля проводился на отрезке, параллельном оси y ($x = 0$, $z = 1{,}1$). При этом максимальное расхождение по абсолютной величине составляло не более 2 А/м, что соответствует погрешности 1,4 %.

Значения намагниченности элементарных объемов, полученные в результате численного эксперимента, отличались от рассчитанных по формуле (18) менее чем на 7 %. Аналогичный численный эксперимент, проведенный с учетом нелинейной магнитной характеристики вещества, также показал адекватные результаты.

2. Магнитный экран в форме полого шара, помещенный во внешнее магнитное поле с индукцией $\boldsymbol{B}_0$. Индукция в полости шара рассчитывается по формуле [19]

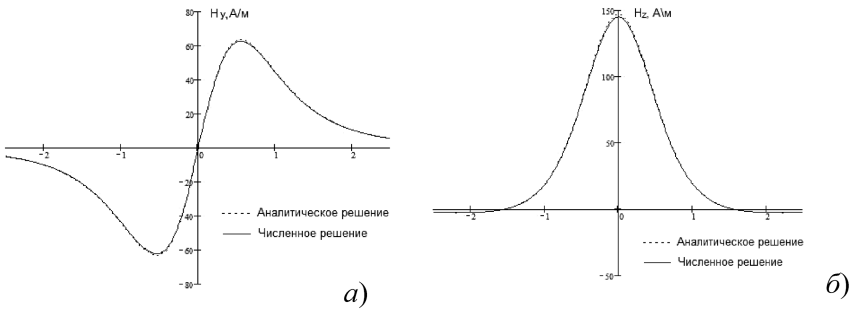
$$\boldsymbol{B} = \boldsymbol{B}_0 \frac{1}{1 + \frac{2}{9}\left(1 - \frac{r^3}{R^3}\right)\left(\frac{\mu_0}{\mu} + \frac{\mu}{\mu_0} - 2\right)}, \quad (26)$$

где R — радиус экрана; r — радиус полости. Откуда для напряженности находим

$$\boldsymbol{H} = \boldsymbol{H}_0 \frac{1}{1 + \frac{2}{9}\left(1 - \frac{r^3}{R^3}\right)\left(\frac{1}{\chi + 1} + \chi - 1\right)}. \quad (27)$$

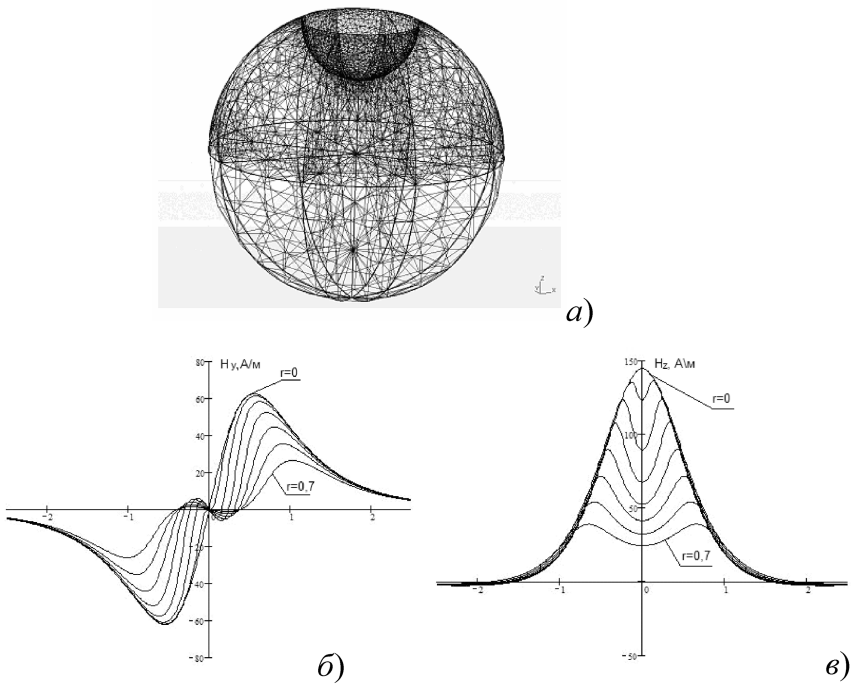
Расчет поля проводился для контрольных точек, расположенных в полости экрана радиусом $R = 1$, радиус полости $r = 0{,}9R$. Напряженность внешнего магнитного поля составляла 100 А/м. Объем тела экрана разбивался на 9194 элементарных объема. Максимальное расхождение численного и аналитического решений составило 0,18 А/м (1,25 %) при $\chi = 100$, и 0,2 А/м (6,2 %) при $\chi = 500$.

По результатам верификации можно сделать вывод о применимости программного обеспечения для анализа магнитных полей объектов сложной формы. В качестве примера рассмотрим расчет поля шара радиусом $R = 1$ с полусферическим углублением радиусом r, помещенного в однородное магнитное поле, направленное вдоль оси z напряженностью 100 А/м (рис. 3, *а*). Магнитная восприимчивость вещества $\chi = 100$. Расчет проводился для ряда значений r, меняющихся от 0 до 0,7 с шагом в 0,1. Поле рассчитывалось на отрезке, параллельном оси y ($x = 0$, $z = 1{,}1$). Графики результатов расчета приведены на рис. 3, *б*, *в*.



а) *б*)

Рис. 2. Напряженность магнитного поля, создаваемая намагниченным шаром



а) *б*) *в*)

Рис. 3. Шар с полусферическим углублением, помещенный в однородное поле

Следовательно, разработанное программное обеспечение позволяет проводить расчет поля в открытых и замкнутых расчетных областях в присутствии ферромагнитных тел со сложной геометрией и с учетом нелинейных магнитных свойств, и может быть использовано для исследования пространственной конфигурации магнитного поля в магнитных системах при их проектировании для теоретического исследования явлений и процессов, в которых происходит взаимодействие ферромагнитных тел со статическими магнитными полями, созданными различными внешними источниками.

Список литературы

1. **Курбатов П. А., Аринчин С. А.** Численный расчет электромагнитных полей. М.: Энергоатомиздат, 1984. 168 с.
2. **Майергойз И. Д.** Итерационные методы расчета статических полей в неоднородных, анизотропных и нелинейных средах. — К.: Наукова думка, 1979. — 210 с.
3. **Матюк В. Ф., Чурило В. Р., Стрелюхин А. В.** Численное моделирование магнитного состояния ферромагнетика в неоднородном постоянном поле методом пространственных интегральных уравнений. I. Описание методики расчета // Дефектоскопия. 2003. № 8. С. 71—78.
4. **Печенков А. Н., Щербинин В. Е.** Некоторые прямые и обратные задачи технической магнитостатики. Екатеринбург: Изд-во УрО РАН, 2004. 177 с.
5. **Яковенко В. В., Шевченко А. И., Романенко А. В.** Модель магнитного поля намагниченного нелинейного ферромагнетика. // Вісник СНУ ім. В. Даля. 2000. № 3. С. 202—203.
6. **Калашникова О. Н., Яковенко В. В., Тхафер Бассим.** Математическая модель магнитного поля для расчета металлоискателей // Вісник СНУ ім. В. Даля. 2000. № 10. С. 151—154.
7. **Жермен-Лакур П., Жорж П. Л., Пистер Ф., Безе П.** Математика и САПР: В 2-х кн. Кн. 2. М.: Мир, 1989. 264 с.
8. **Гальченко В. Я.** Информационные модели в теории и практике электромагнитной дефектоскопии. Луганск: Изд-во Восточноукраинского государственного университета, 1997. 262 с.
9. **Фихтенгольц Г. М.** Курс дифференциального и интегрального исчисления. Т. 3. М.-Л.: ГТТИ, 1949. 783 с.
10. **Алиевский Б. Л., Орлов В. Л.** Расчет параметров магнитных полей осесимметричных катушек. М.: Энергоатомиздат, 1983. 112 с.
11. **Печенков А. Н.** Расчет трехмерного магнитного поля круглой катушки с прямоугольным сечением и постоянным током // Дефектоскопия. 2006. № 9. С. 65—71.
12. **Печенков А. Н.** Аппроксимация сложных катушек с конечной толщиной для расчета трехмерных магнитных полей этих катушек // Дефектоскопия. 2006. № 10. С. 27—32.
13. **Расчет** электромагнитных цепей и электромагнитных полей на ЭВМ / Под ред. Л. В. Данилова, Е. С. Филипова. М.: Радио и связь, 1983. 344 с.
14. **Том Р., Тарр Дж.** Магнитные системы МГД-генераторов и термоядерных установок. М.: Энергоатомиздат, 1985. 272 с.
15. **Панасенков М. А.** Электромагнитные расчеты устройств с нелинейными распределенными параметрами. М.: Энергия, 1971. 216 с.
16. **Мельгуй М. А.** Формулы для описания нелинейных и гистерезисных свойств ферромагнетиков // Дефектоскопия. 1987. № 11. С. 3—10.
17. **Афанасьев Ю. В.** Феррозондовые приборы. Л.: Энергоатомиздат, 1986. 188 с.
18. **Треногин В. А.** Функциональный анализ. М.: Наука, 1980. 496 с.
19. **Нейман Л. Р., Демирчян К. С.** Теоретические основы электротехники. Т. 2. М.: Энергия, 1967. 408 с.

ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ

УДК 004.032.26

А. Б. Барский, д-р техн. наук, проф.
каф. "Вычислительные системы и сети",
МИИТ, Москва
E-mail: arkbarsk@mail.ru

Применение логической нейронной сети для распознавания объектов временного ряда по заданному набору признаков

Рассматривается возможность выбора объектов временного ряда по значениям множества признаков с помощью аппарата логических нейронных сетей. Данный аппарат позволяет проводить параллельное, одновременное сравнение значений всех признаков с эталонными. Совместное использование нескольких эталонов позволяет построить систему принятия решений.

Ключевые слова: *объект временного ряда, множество признаков, распознавание, логическая нейронная сеть, система принятия решений.*

Агафья Тихоновна. ...Если бы губы Никанора Ивановича да приставить к носу Ивана Кузьмича, да взять сколько-нибудь развязности, какая у Балтазара Балтазарыча, да, пожалуй, прибавить к этому еще дородности Ивана Павловича — я бы тогда тотчас же решилась.

Н. В. Гоголь. "Женитьба".

Введение

Временной ряд является важным объектом анализа и исследования в области информационных технологий. Он характеризуется реальным временем поступления данных, например, — от регистрирующей аппаратуры при проведении испытаний сложных систем, и столь же оперативной обработкой этих данных. Обработка возможна и с меньшим темпом – при частичном накоплении информации. Однако всегда можно выделить ту первичную обработку, которая проводится в темпе поступления информации, т. е. в реальном масштабе времени.

Первичная обработка временного ряда связана с распознаванием и классификацией данных для запуска средств последующей их обработки. Это порождает надежду на эффективное применение на данном этапе таких средств искусственного интеллекта, как логические нейронные сети [1—3].

1. Детерминированные оценки объектов временного ряда

Маша хочет замуж... Нескончаемой чередой мимо ее окошка по городской улице проходят мужчины... Идеал настоящего мужчины, достойного стать ее мужем, Маша определила по следующим свойствам-признакам:

- он брюнет;
- его рост — не менее 180 см;
- он обладатель длинного носа;
- он должен носить шляпу.

Поток мужчин настолько интенсивен, что Маша принимает единственно правильное решение: она должна автоматизировать процесс селекции мужчин, обладающих указанными признаками, для того чтобы в реальном времени, с применением методов распараллеливания, проводить (в своих мечтах) обработку встретившегося "кандидата".

Итак, имеется временной ряд — поток объектов, характеризующийся набором данных по каждому из них. Необходимо также в реальном времени решать задачи обработки этих объектов. Эти задачи разнообразны: аппроксимация функциями времени, вероятностная обработка результатов испытаний, обработка динамики состояния фондового рынка, динамический контроль состояния сложной системы и т. д.

Среди этих задач важное место занимает задача селекции и идентификации — задача ассоциативного поиска по эталону. Такая задача возникает, например, при сопровождении спутника, вновь появившегося в секторе обзора.

В простейшей постановке задача формулируется следующим образом.

Объекты, составляющие временной ряд, т. е. последовательно предъявляемые для обработки, имеют множество признаков $A = \{a_1, ..., a_m\}$. Задан эталон, соответствующий фиксированному значению этих признаков $\{b_1, ..., b_m\}$. Необходимо в динамике последовательного анализа объектов временного ряда выделять (и направлять на специальную обработку) те объекты, где все значения признаков совпадают с эталонными, $a_i = b_i$, $i = 1, ..., m$.

Обобщением постановки этой задачи, превращающим ее в задачу классификации, является следующее дополнение: провести распознавание каждого возможного набора значений признаков $A = \{a_1, ..., a_m\}$ для принятия решения по дальнейшей обработке объекта, имеющего эти значения признаков. Очевидно решение данной задачи включает в себя решение задачи, поставленной ранее.

Основным элементом данной задачи является распознавание. В общем случае оно усложняется тем, что точное совпадение признаков с эталонными маловероятно. В этом случае можно лишь говорить о допустимой степени достоверности этого совпадения. Правомочен вопрос: "На какой эталон в значительной степени похож анализируемый объект?" Это указывает на эффективность применения параллельных нейросетевых технологий, опирающихся на несложный расчет значений передаточной функции нейроподобного элемента. Этот расчет лежит в основе модели ассоциативного мышления и адекватно отображается *логической нейронной сетью.*

Конечно, Маша могла бы построить традиционный алгоритм *последовательного* анализа признаков проходящих мужчин. Затем бы она решила проблему минимизации сложности полученного алгоритма — опять же в традиционной постановке, изложенной, например, в работе [4].

Она бы догадалась, что минимизировать число основных операций сравнения можно, если проводить это сравнение, упорядочив признаки по неубыванию частоты появления интересующего значения.

Например, она заметила, что реже всего появляются мужчины в шляпах. Тогда, если она увидит мужчину в шляпе, это повлечет анализ других признаков. Значит, чем реже будут встречаться мужчины в шляпах, тем реже придется отвлекаться на дополнительное сравнение. В то же время сокращение затрат на последующий анализ мужчины в шляпе требует, чтобы интересующее значение признака, с которого этот анализ начинается, также был редко встречающимся и т. д.

Однако такой порядок анализа признаков требует периодического исследования выборок из временного ряда для выявления частоты появления требуемых значений признаков. В комплексе исследование ряда и частотный анализ значений признаков (по каждому эталону!) можно рассматривать как путь построения самонастраивающегося (по минимуму требуемой производительности) алгоритма поиска с помощью последовательного сравнения значений признаков с эталонными.

Машу такой путь не устраивает. Ей хочется получить алгоритм *параллельного* анализа всех признаков сразу методом, подобным методу ассоциативного мышления, — для дальнейшего обоснования применения параллельных вычислительных средств: многопроцессорного суперкомпьютера, кластера локальной вычислительной сети или нейрокомпьютера. Кроме того, она рассчитывает в дальнейшем принимать решения не по единственному эталону и даже — не по единственному идеалу.

Приступим к формированию логической нейронной сети – основы машинной системы принятия решений (рис. 1, см. третью сторону обложки).

Для формирования нейронов рецепторного слоя проведем градацию "мужских" признаков на основе понятия *полного множества событий* [2, 3].

1. Пусть в Машином представлении мужчины бывают *брюнеты*, *блондины*, *прочие*. Это требует введения трех нейронов-рецепторов.

2. В области интересов, касающихся роста, Маша выделила четыре исчерпывающих диапазона, которым соответствуют высказывания: "*ниже* 180 см", "180—190 см", "190—200 см", "*выше* 2 м".

Это требует использования еще четырех нейронов рецепторного слоя.

3. В части длины носа у Маши выработался свой стереотип, в соответствии с которым она делит носы на "*короткие*", "*картошечкой*" и "*длинные*", также предполагая, что все возможности исчерпаны, и трех рецепторов ей достаточно.

4. Наличию шляпы соответствует булевы переменные "*шляпа есть*" и "*шляпы нет*". (Не забывать, что они задаются значениями истинности соответствующих высказываний!) Значения этих переменных преобразуются в действительные. Для данных переменных необходимы два рецептора.

Для краткого описания системы принятия решений (СПР) введем обозначения всех фигурирующих высказываний (это отражено на рис. 1):

x_1 = "*Брюнет*";
x_2 = "*Блондин*";
x_3 = "*Прочий*";
y_1 = "*Ниже* 180 см";
y_2 = "*Между* 180 и 190 см";
y_3 = "*Между* 190 см и 2 м";
y_4 = "*Выше* 2 м";
z_1 = "*Нос короткий*";
z = "*Нос картошечкой*";
z_3 = "*Нос длинный*";
k_1 = "*Шляпа есть*";
k_2 = "*Шляпы нет*".

Маша создает уже обученную сеть, вводя необходимые связи с единичными весами. Прежде всего она формирует нейрон выходного слоя *Вых*1, который будет максимально возбуждаться при появлении идеального мужчины. Маша связывает с этим нейроном все рецепторы, реагирующие на интересующие ее значения признаков.

Предположим, что кроме "идеальной" комбинации Машу интересуют не все возможные комбинации признаков проходящих мужчин. Некоторые комбинации она вовсе не удостаивает вниманием. Однако по некоторым она готова сделать замечания, хотя бы связанные с сожалением о близком счастье. Поэтому создаваемая Машей нейронная сеть содержит несколько нейронов выходного слоя, соответствующих лишь тем решениям, замечаниям и сожалениям, которые готовы слететь с губ за чашкой чая у раскрытого окна.

Тогда логическое описание СПР имеет вид:

$x_1 \wedge (y_2 \vee y_3) \wedge z_3 \wedge k_1 \to R_1 =$ < Сладко помечтать о возможном счастье > ;

$x_1 \wedge y_4 \wedge z_3 \wedge k_1 \to R_2 =$ < Воскликнуть: "Ну почему он такой высокий!" > ;

$x_2 \wedge (y_1 \vee y_4) \wedge (z_1 \vee z_2) \wedge k_2 \to R_3 =$ < Воскликнуть: "Как только таких на улицу выпускают!" > ;

$x_3 \wedge y_4 \wedge z_3 \wedge k_2 \to R_4 =$ < Воскликнуть: "А ведь хорош! Но..." >

Необходимо отметить, что при решении общей задачи анализа всех возможных ситуаций число N нейронов выходного слоя в данном случае должно составлять $3 \times 4 \times 3 \times 2 = 72$. Каждый нейрон выходного слоя в этом случае указывал бы на решение, принимаемое для соответствующей ему комбинации значений признаков. В общем случае обработки временного ряда справедливо учитывать все возможные ситуации, принимая по каждой из них отдельное решение.

Учет не всех комбинаций признаков приводит к необходимости тщательного подбора параметров передаточной функции. Ведь решение должно приниматься не просто по максимальному возбуждению нейронов выходного слоя (оно существует всегда!), а по возбуждению, превысившему достаточно высокий порог.

Воспользовавшись рекомендациями работы [2], Маша выбирает передаточную функцию на основе счета значений следующих выражений:

$$f_i = \sum_j \omega_j V_j;$$

$$f_{Вых\,i} = \begin{cases} f_i, \text{ если } f_i \geqslant h, \\ 0 \text{ в противном случае.} \end{cases}$$

Здесь V_j — значение возбуждения на j-м входе нейрона выходного слоя, поступившего от связанного с ним рецептора; ω_j — вес входа (связи). На этапе обучения значение возбуждения рецептора принимается равным единице, если значение высказывания о наличии соответствующего признака является истинным, и равным нулю в противном случае.

Порог h выбирается из следующих соображений. Поскольку возбуждение нейрона выходного слоя должно быть высоким только в случае поступления на его вход ровно четырех единиц, следует выбрать значение $4 > h > 3$. Легко проверить, что все ситуации, по которым предусмотрено решение, обнаруживаются с помощью высокого возбуждения единственного нейрона выходного слоя. Если ситуация относится к тем, по которым решение не предусмотрено, то ни один нейрон выходного слоя не возбудится, и Маша просто проигнорирует появление на улице очередного мужчины.

На рис. 1 система принятия решений представлена полностью. Все веса связей равны единице и потому на рисунке не обозначены.

Конечно, в виде, представленном на рисунке, нейронная сеть не обрабатывается компьютером. Информационная технология обработки логической нейронной сети основана на представлении ее матрицей следования (рис. 2). Каждая строка и

	x_1	x_2	x_3	y_1	y_2	y_3	y_4	z_1	z_2	z_3	k_1	k_2
*Вых*1	1				1	1				1	1	
*Вых*2	1						1			1	1	
*Вых*3		1		1			1	1	1			1
*Вых*4			1				1			1	1	

Рис. 2. Матрица следования

столбец соответствуют нейрону, а связи показаны их весами в том столбце, который соответствует их исходу.

Однако, несколько подумав, Маша приходит к выводу, что наличие длинного носа чревато серьезными последствиями в семейной жизни. Нос "картошечкой", несомненно, благотворнее влияет на мягкость нрава. Тогда Маша решает, что при последующей модификации она обязательно положит вес $\omega_{z3 \to Вых1}$ связи, ведущей из нейрона-рецептора z_3 к нейрону *Вых*1, равным 0,5. В то же время она введет новую связь, соединяющую нейрон z_2 с нейроном *Вых*1. Вес этой связи $\omega_{z2 \to Вых1}$ она выберет также равным 0,5. Приведенный проект не отражен на рисунке. Он лишь иллюстрирует прекрасные возможности взвешенного учета различных факторов при модификации и развитии логической нейронной сети.

Очевидно, что с помощью построенной однослойной логической нейронной сети можно параллельно (!) исследовать значения всех признаков проходящих мужчин (в общем случае — всего множества признаков каждого объекта временного ряда). Это отражает тот факт, что искусственная нейронная сеть, как и естественная, имитирует параллельную обработку сигналов, одновременно поступивших на все рецепторы, т. е. проводит распараллеливание выполнения сложных логических конструкций вида "если... то...". Относительно матрицы следования это означает возможность параллельной обработки всех строк, соответствующих нейронам выходного слоя, с помощью передаточной функции. В общем случае эта обработка параллельно-последовательная. Элемент последовательности возникает в связи с преемственностью информации внутри логических цепочек, образуемых нейронами.

Однако при программном исполнении на компьютере непараллельной архитектуры обработка нейроподобных элементов сети осуществляется последовательно. Этого вполне достаточно для Маши, но крайне затруднительно в общем случае обработки временного ряда.

Поэтому построение нейрокомпьютера, допускающего параллельную обработку многих нейронов по SPMD-технологии [5], следует считать обоснованным и актуальным.

Загрузка входного, рецепторного, слоя заключается в формировании массива, каждый элемент которого соответствует рецептору и должен принять значение возбуждения этого рецептора.

Как же автоматизировать возбуждение рецепторов, придавая каждому из них значение 0 или 1 в соответствии с признаком проходящего мужчины?

Предположим, в результате подключения аппаратуры наружного наблюдения к компьютеру появилась возможность "считывания" значения указанных признаков для каждого проходящего мужчины.

Чтобы поместить эти значения "на место", следует воспользоваться базированием и индексацией. В качестве значения базы B_i выбирается известный, предусмотренный при программировании, адрес первого элемента подмассива рецепторов, закрепленного за признаком i ($i = 1, ..., m$). Например, для построенной нейронной сети $B_1 = 0$, $B_2 = 3$, $B_3 = 7$, $B_4 = 10$.

Индекс, или смещение, находится на основе способа градации каждого признака. Например, признак масти мужчины предполагает нумерацию с нуля элементов множества {*брюнет*, *блондин*, *прочие*}. Тогда адрес рецептора, соответствующего высказыванию "*брюнет*", равен $B_1 + 0 = 0$, адрес рецептора, соответствующего высказыванию "*блондин*", равен $B_1 + 1 = 1$, и, соответственно, следующий адрес находится как $B_1 + 2 = 2$.

Адреса рецепторов, определяющих рост, найти труднее. Для этого необходимо исследовать значения роста и соотносить их с принятым шагом градации Δ, принимая во внимание минимальное учитываемое значение параметра роста. А именно, в данном случае $\Delta = 10$ см. Минимальное учитываемое значение признака равно 180 см. Чтобы его получить, необходимо 0 раз применить (сложить) значение Δ. Следовательно, 0 — индекс (смещение) для получения с помощью базы B_2 адреса рецептора, соответствующего высказыванию "*рост меньше* 180 см".

Для получения адреса рецептора, соответствующего высказыванию "*рост лежит в пределах от* 180 до 190 см", необходимо один раз применить значение Δ при суммировании с минимальным значением признака. Следовательно, 1 — индекс (смещение) для получения (с помощью B_2) адреса рецептора, соответствующего данному высказыванию, и т. д.

Для остальных признаков индексы при расчете адресов соответствующих рецепторов находятся с помощью баз B_3 и B_4 аналогично признаку масти.

2. Структурированная, не однослойная логическая нейронная сеть

Маша готова к компромиссу. Ее представления о идеальном мужчине расширяются. В качестве желанного жениха она готова рассмотреть не менее шикарный вариант {*блондин*, *рост ниже* 180 см, *нос картошечкой*, *шляпа отсутствует*}.

Казалось бы, достаточно дополнить нейронную сеть на рис. 1 (см. третью сторону обложки) связями, ведущими из рецепторов, соответствующих указанным параметрам, к нейрону *Вых*1, указывающему на решение R_1. Однако это не так. В работе [2] представлена методика построения обученной логической нейронной сети, в соответствии с которой при попытке ограничиться одним слоем необходимо проверить, не поглотил ли обобщенный эталон, предполагающий некоторое решение, другой обобщенный эталон, ведущий к другому решению. Следовательно, может случиться, что для некоторых ситуаций возможно неоднозначное решение.

В данном случае такая опасность отсутствует. Однако при расширении системы принятия решений, т. е. при формировании новых решений по некоторым, пока не рассмотренным ситуациям, возможна неоднозначность рекомендуемых решений. Например, Маша скоро захочет сформировать новое решение R_5 на основе логического выражения

$$(x_1 \wedge y_2 \wedge z_2 \wedge k_1) \rightarrow R_5 = \\ = (\textit{Воскликнуть: "Душечка!"}).$$

Легко видеть, что при непосредственном связывании рецепторов с нейронами выходного слоя в однослойной сети задание (единицами) указанной ситуации приведет к активизации двух решений — R_1 и R_5.

В работе [2] предлагаются два способа ликвидации неоднозначности принимаемых решений.

Первый способ заключается в строгом воспроизведении структуры нового логического выражения, определяющего (в данном случае) решение R_1:

$$(x_1 \wedge (y_2 \vee y_3) \wedge z_3 \wedge k_1) \vee (x_2 \wedge y_2 \wedge z_2 \wedge k_2) \rightarrow R_1.$$

Наличие скобок указывает на способ структуризации формируемой сети (рис. 3), которая теряет свойство однослойности.

Целесообразно в этом случае использовать ранее рассмотренную передаточную функцию с высоким значением порога. Важность использования высокого порога обусловлена тем, что при замене логических операций дизъюнкции и конъюнкции операцией суммирования входных сигналов нейроподобного элемента, при возникновении соответствующей ситуации должен возбудиться лишь один из нейронов — № 1 или № 2. В противном случае значения возбуждения этих нейронов будут складываться при формировании возбуждения нейрона *Вых*1. При этом значение возбуждения нейрона y_2 будет учтено дважды. Тогда нейроны выходного слоя будут иметь неодинаковые возможные значения максимально высокого возбуждения, что снизит достоверность выводов нейронной сети.

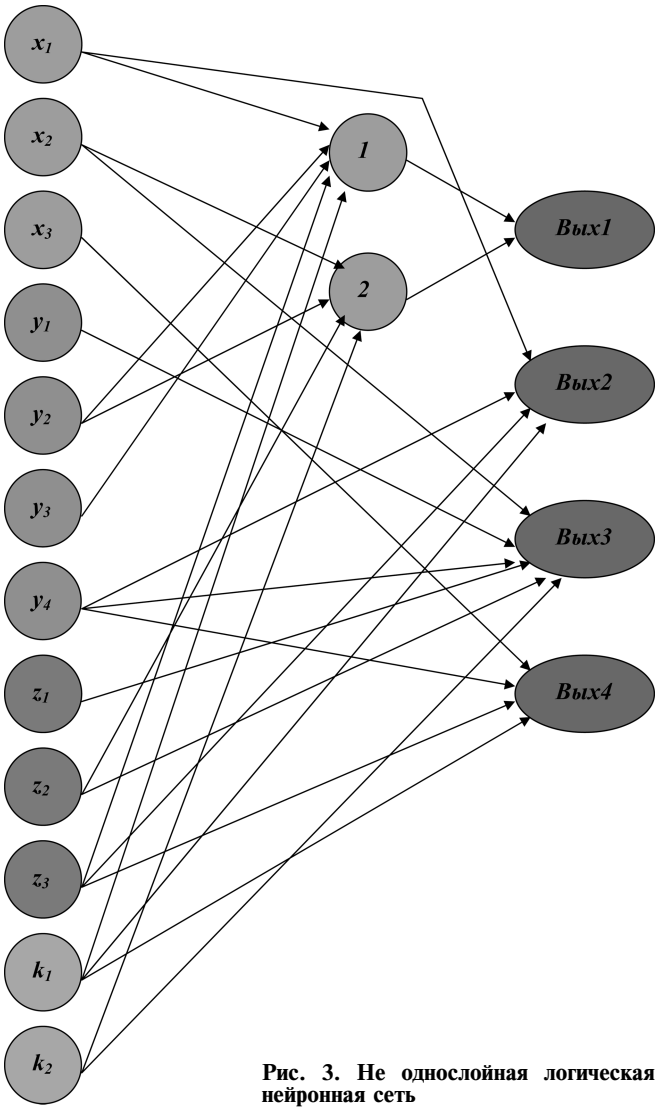


Рис. 3. Не однослойная логическая нейронная сеть

Второй способ называется *размножением решений*. Он заключается в том, что каждая конъюнкция в выражении рассмотренного вида приводит к возбуждению одного нейрона выходного слоя. То есть каждый нейрон, выполняющий роль конъюнктора, инициирует одно и то же решение.

В этом случае можно записать два логических выражения, обусловливающих решение R_1: заданное первоначально и новое

$$x_2 \wedge y_2 \wedge z_2 \wedge k_2 \to R_1.$$

Появившаяся интерпретация того же решения вводит в искушение развития системы принятия решений. Например, первое вхождение R_1 в описание системы может быть дополнено разъяснением:

$R_{11} = <$*Сладко помечтать о возможном счастье с высоким брюнетом*$>$.

Второе вхождение R_1 может привести к аналогичному изменению:

$R_{12} = <$*Сладко помечтать о возможном счастье с симпатичным блондином*$>$.

Таким образом, принимаемые решения в действительности отличаются одно от другого, а нейронная сеть остается однослойной. Это облегчает ее обработку.

3. Нечеткие оценки признаков объектов временного ряда

Проходят годы, а Маша все сидит за самоваром у раскрытого окошка и слабеющими глазами пытается угадать свой идеал. Конечно, только с долей вероятности (достоверности) она может оценить значение каждого из интересующих ее признаков, в комплексе влияющих на ее настроение.

В этих условиях неопределенности мы спешим на помощь, вооруженные современными положениями теории нечетких множеств [6], призванными обосновать задание исходной информации на рецепторном слое Машиной нейронной сети.

Первоначально мы вынуждены признать, что каждое значение любого признака (не следует забывать, что эти значения соответствуют исчерпывающему множеству событий) определяется с достоверностью $P \leqslant 1$. А куда девать остальные $1 - P$?

Практический, "инженерный" подход заключается в рассмотрении нормального закона распределения вероятностей. Согласно ему, все вероятности событий исчерпывающего множества, в сумме образующие единицу, распределены между "смежными" по смыслу и содержанию высказываниями, т. е. служат возбуждению соответствующих рецепторов так, чтобы измеренное или предполагаемое значение P было близко к математическому ожиданию значения признака или совпадало с ним.

Осуществляя практический подход, продемонстрируем возбуждение нейронов-рецепторов на основе нечетких оценок значений параметров для системы принятия решений, построенной Машей (см. рис. 1 на третьей стороне обложки).

Пусть, оглядывая очередного проходящего мужчину, Маша быстро прикидывает:

- процентов на 70 я уверена, что он брюнет, хотя, возможно, он блондин или шатен, ведь он, кажется, в шляпе;
- его рост не менее 190 см, в этом я уверена процентов на 80;
- носа я совершенно не вижу: возможно, он закрыт тенью от шляпы или шевелюры;
- не пойму: шляпа у него или такая шевелюра? Скорее, все же, шляпа.

Не смея долго раздумывать (ведь из соседней закусочной показался следующий мужчина), Маша формирует возбуждение рецепторов:

$$V_{x1} = 0{,}7,\ V_{x2} = V_{x3} = 0{,}15,$$
$$V_{y1} = 0,\ V_{y2} = 0{,}15,\ V_{y3} = 0{,}8,\ V_{y4} = 0{,}05,$$
$$V_{z1} = 0{,}33,\ V_{z2} = 0{,}33,\ V_{z3} = 0{,}33,$$
$$V_{k1} = 0{,}6,\ V_{k2} = 0{,}4.$$

По-видимому, формирование эвристического алгоритма такого "разбрасывания" вероятностей, на глазок воспроизводящего нормальный закон, не представляется трудным.

В условиях допущенной неопределенности необходимо снизить значение порога h. Он должен подбираться экспериментально, чтобы Маша не лишилась окончательно удовольствия и надежды. Положим $h = 2{,}5$ и рассчитаем возбуждение нейронов выходного слоя:

$$V_{Вых1} = 2{,}58,\ V_{Вых2} = V_{Вых3} = V_{Вых4} = 0.$$

Так что радости еще есть место.

В общем случае признаки объектов временного ряда преимущественно имеют действительные значения из широкого диапазона, являясь отражением измерений различных физических, динамических, временных и прочих характеристик. Разбивка этого диапазона на большое число малых интервалов значений — для закрепления за ними рецепторов — позволяет более точно воспроизводить нормальный закон распределения вероятностей. Это соответствует условному воспроизведению гистограммы.

Заключение

Логические нейронные сети являются важным аппаратом параллельного выполнения сложных логических конструкций. Параллельная обработка сигналов, поступивших на рецепторный слой, является непременным условием высокой скорости ассоциативного мышления, свойственного мозгу. Обработка объектов (информационных кадров, состояния средств испытываемых сложных систем, потактового формирования управляющих воздействий и т. д.), образующих вре-

менной ряд, методами ассоциативной селекции позволяют приблизить эту обработку к режиму реального времени. В то же время только разработка нейрокомпьютеров — параллельных вычислительных систем, специализированных для параллельного счета значений несложных передаточных функций нейронов сети, позволяет практически решать проблему оперативной обработки временных рядов.

Популяризация этих идей требует их доступного и иллюстративного изложения. Это определило стиль представления настоящего материала.

Список литературы

1. **Барский А. Б.** Нейронные сети: распознавание, управление, принятие решений. — М.: Финансы и статистика, 2004.
2. **Барский А. Б.** Логические нейронные сети. — М.: ИНТУИТ; БИНОМ.ru, 2007.
3. **Барский А. Б.** Математическая логика событий и логические нейронные сети // Информационные технологии. Приложение. 2007. № 7.
4. **Вирт Н.** Алгоритмы и структуры данных. — М.: Мир, 1989.
5. **Барский А. Б.** Параллельные информационные технологии. — М.: ИНТУИТ; БИНОМ.ru, 2007.
6. **Ярушкина Н. Г.** Основы теории нечетких и гибридных систем. — М.: Финансы и статистика, 2004.

УДК 519.711.3

С. Д. Махортов, канд. физ.-мат. наук, доц.,
С. Л. Подвальный, д-р техн. наук, проф.,
Воронежский государственный
технический университет

Алгебраический подход к исследованию и оптимизации баз знаний продукционного типа

Рассматривается алгебраическая система, содержащая семантику расширенной продукционной логики. Операции и отношения этой системы соответствуют полному набору логических связок пропозиционального исчисления. Определены также бесконечные операции, реализующие для модели продукционной логики кванторы общности и существования. Доказаны новые теоремы о структуре логического замыкания и о существовании логической редукции рассматриваемой алгебраической системы. Указан эффективный способ построения логической редукции.

Полученные результаты могут быть применены для исследования и автоматической оптимизации баз знаний продукционного типа. К таковым, в частности, относятся базы знаний систем символьной математики.

Ключевые слова: *продукционная логика, алгебраическая система, полная решетка, логическое замыкание, логическая редукция.*

Введение

Среди моделей представления знаний в интеллектуальных информационных системах можно выделить две категории: логические, основанные на строгой математической теории (логика высказываний, предикатов и т. д.), и продукционно-сетевые, построенные на базе упрощенных схем "человеческой логики". Первая категория характеризуется серьезными математическими обоснованиями, вторая находит более широкое применение на практике, имея при этом слабую теоретическую базу [1].

Эффективным подходом к построению и исследованию формальных логических систем является "алгебраизация логики". Известная теория Линденбаума—Тарского [2] рассматривает логику нулевого порядка как универсальную алгебру, операции которой соответствуют логическим связкам пропозиционального языка. Примерами алгебраизации исчисления предикатов первого порядка являются полиадические алгебры Халмоша [3] и цилиндрические алгебры [4]. Однако общие алгебраические подходы, расширяя возможности исследования самих логических теорий, существенно не облегчают их практического применения.

В работах [5—6] была сделана попытка создания некоторой математической основы систем второй категории. Предложены алгебраические структуры, формализующие продукционно-логические связи на основе теории решеток и отношений. Основная идея состоит в моделировании продукционных (импликативных) связей отдельным бинарным отношением с соответствующими свойствами. Решетку вместе с заданным на ней дополнительным отношением можно назвать LP-системой (*Lattice-* или *Logical-Production System*). Семантика предложенной в [5—6] логики содержит операции, соответствующие двум логическим связкам — конъюнкции и импликации. Такой минимум операций соответствует набору правил типичной продукционной системы.

Одно из направлений развития теории LP-систем состоит в расширении логики. Продукционные системы в своих правилах могут содержать не только конъюнкции термов, но также дизъюнкции и отрицания. Кроме того, некоторые задачи

требуют использования логических кванторов. Такие модели могут возникать при разработке систем символьной математики. Большинство математических теорем имеют форму продукции "дано" → "требуется доказать", где предпосылка и заключение являются формулами логики первого порядка. Соответственно возникла идея о распространении методов работ [5—6] на такую логику. Это позволило бы формально преобразовывать математические знания, а также осуществлять их верификацию и оптимизацию. Начало такому исследованию положено в статье [7], где была введена LP-система, операции и отношения которой соответствуют полному набору логических связок с кванторами общности и существования. Доказано существование логического замыкания, а также исследованы возможности эквивалентных преобразований LP-системы.

Настоящая работа является продолжением статьи [7]. В качестве основных результатов здесь доказаны теоремы о структуре логического замыкания и о существовании логической редукции произвольного LP-отношения. Указан эффективный способ построения логической редукции.

Заметим, что возможности алгебраического исследования продукционно-логических систем содержит и теория ультраоператоров [8]. В приложении к математической логике она также предлагает рассматривать импликации в виде отдельного соответствия. Однако к этому соответствию предъявляется больше требований (ультраоператор является отображением). Кроме того, авторам настоящей статьи неизвестны более подробные исследования ультраоператоров в данном направлении.

Основные понятия и обозначения

Вначале введем ряд обозначений, а также напомним несколько определений и фактов, связанных с отношениями и решетками.

Бинарное отношение R на произвольном множестве F называется рефлексивным, если для любого $a \in F$ справедливо $(a, a) \in R$. Оно называется транзитивным, если для любых $a, b, c \in F$ из (a, b), $(b, c) \in R$ следует $(a, c) \in R$. Существует замыкание R^* произвольного отношения R относительно свойств рефлексивности и транзитивности [9]. Пара элементов $a, c \in F$ называется транзитивной в R, если $(a, c) \in R_1^*$, где R_1^* — транзитивное замыкание отношения $R_1 = R \backslash \{(a, c)\}$.

Существует понятие транзитивной редукции бинарного отношения — в определенном смысле обратное по отношению к транзитивному замыканию. Для данного отношения R строится минимальное отношение R', имеющее то же самое транзитивное замыкание. Как обычно, для частично упорядоченных множеств мы будем различать понятия минимального элемента (для него нет меньшего элемента) и наименьшего элемента (он меньше всех). В [9] приведен алгоритм построения транзитивной редукции конечных отношений; показано, что эта задача вычислительно эквивалентна задаче построения транзитивного замыкания, и доказана единственность транзитивной редукции ациклического графа.

Необходимые для чтения данной статьи сведения о решетках содержатся в [10]. *Решеткой* называется полуупорядоченное множество $\mathbb{F}$, в котором наряду с отношением частичного порядка $\leqslant$ ("не больше", "содержится") введены также два двуместных оператора $\wedge$ ("пересечение") и $\vee$ ("объединение"). Элемент $a \wedge b$ является точной нижней гранью, а элемент $a \vee b$ — точной верхней гранью пары элементов $a, b \in \mathbb{F}$ в смысле отношения $\leqslant$.

Решетка $\mathbb{F}$ называется *ограниченной*, если она содержит нижнюю и верхнюю грани — такие два элемента O, I, что $O \leqslant a \leqslant I$ для любого $a \in \mathbb{F}$. Решетка $\mathbb{F}$ называется *полной*, если каждое ее подмножество имеет в $\mathbb{F}$ точные нижнюю и верхнюю грани. В полной решетке для любого подмножества элементов можно определить многоместные пересечение и объединение. Для таких операций используются обозначения $\bigwedge$ и $\bigvee$ с соответствующими индексами.

Решетка называется *дистрибутивной*, если в ней при любых a, b, c выполняются равенства: $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$ и $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. Под дополнением элемента a в ограниченной решетке $\mathbb{F}$ подразумевают элемент $a' \in \mathbb{F}$ такой, что $a \wedge a' = O$ и $a \vee a' = I$. Дистрибутивная решетка с дополнениями называется *булевой*. В ней каждый элемент имеет единственное дополнение, причем справедливы тождества (законы де Моргана) $(a \wedge b)' = a' \vee b'$; $(a \vee b)' = a' \wedge b'$. В полной булевой решетке справедливы "бесконечные" аналоги (см. [11]) $(\bigwedge_{t \in T} a_t)' = \bigvee_{t \in T} a_t'$; $(\bigvee_{t \in T} a_t)' = \bigwedge_{t \in T} a_t'$. Нам понадобятся также свойства ассоциативности и дистрибутивности, связывающие конечные операции с "бесконечными" [11]:

$$b \wedge (\bigwedge_{t \in T} a_t) = \bigwedge_{t \in T} (b \wedge a_t);\ b \vee (\bigvee_{t \in T} a_t) = \bigvee_{t \in T} (b \vee a_t);$$

$$b \vee (\bigwedge_{t \in T} a_t) = \bigwedge_{t \in T} (b \vee a_t);\ b \wedge (\bigvee_{t \in T} a_t) = \bigvee_{t \in T} (b \wedge a_t).$$

В приложении к продукционной логике предлагаемая нами модель ориентируется на решетку Линденбаума—Тарского, дополненную принятой в [2] алгебраической интерпретацией кванторов общности и существования в виде бесконечноместных операций пересечения и объединения.

Продукционно-логические отношения на решетках

В этом разделе мы рассматриваем бинарные отношения на полной булевой решетке $\mathbb{F}$. Заметим, что отношение частичного порядка $\leqslant$ на $\mathbb{F}$ по определению является рефлексивным и транзитивным. Сформулируем несколько базовых определений.

Определение 1. Бинарное отношение R на решетке с дополнениями называется контрапозиционным, если из $(a, b) \in R$ следует $(b', a') \in R$.

Определение 2. Бинарное отношение R на полной решетке назовем $\bigwedge$-дистрибутивным, если для любого подмножества элементов решетки $\{b_t | t \in T\}$, такого что $(a, b_t) \in R$, $\forall t \in T$, справедливо $(a, \bigwedge\limits_{t \in T} b_t) \in R$.

Определение 3. Бинарное отношение R на полной решетке назовем $\bigvee$-дистрибутивным, если для любого подмножества элементов решетки $\{a_t | t \in T\}$ из $(a_t, b) \in R$, $\forall t \in T$ следует $(\bigvee\limits_{t \in T} a_t, b) \in R$.

Определение 4. Отношение R на полной решетке называется вполне дистрибутивным, если оно одновременно удовлетворяет определениям 2 и 3.

Определение 5. Отношение R на полной булевой решетке называется продукционно-логическим (или просто логическим), если оно содержит отношение $\leqslant$, а также является контрапозиционным, вполне дистрибутивным и транзитивным.

Из определения 5 следует, что отношение $\leqslant$ на полной булевой решетке является логическим.

Определение 6. Логическим замыканием отношения R, заданного на полной булевой решетке, называется наименьшее логическое отношение, содержащее R. Два отношения R_1 и R_2 на общей решетке $\mathbb{F}$ называются логически эквивалентными (или просто эквивалентными), если их логические замыкания совпадают ($R_1 \sim R_2$).

Логической редукцией данного отношения называется эквивалентное ему минимальное отношение.

Определение 7. Пусть дано произвольное отношение R на полной булевой решетке $\mathbb{F}$. Будем говорить, что упорядоченная пара $a, b \in \mathbb{F}$ логически связана отношением R ($a \xrightarrow{R} b$), если выполнено одно из следующих условий:

1) $(a, b) \in R$;

2) $a \leqslant b$;

3) $b' \xrightarrow{R} a'$;

4) существуют такие $b_t \in \mathbb{F}$, $t \in T$, что $\bigwedge\limits_{t \in T} b_t = b$, причем $a \xrightarrow{R} b_t$, $\forall t \in T$;

5) существуют такие $a_t \in \mathbb{F}$, $t \in T$, что $\bigvee\limits_{t \in T} a_t = a$, причем $a_t \xrightarrow{R} b$, $\forall t \in T$;

6) существует элемент $c \in \mathbb{F}$ такой, что $a \xrightarrow{R} c$ и $c \xrightarrow{R} b$.

Определение 7 по данному R задает новое отношение $\xrightarrow{R}$ на той же решетке $\mathbb{F}$, которое содержит отношения R, $\leqslant$, а также обладает некоторыми дополнительными свойствами, которые мы обсудим ниже. Условия 1)—6) определения 7 мы будем также называть *правилами* (вывода).

При получении некоторой логической связи *шагом вывода* мы будем называть применение ровно одного правила, возможно, одновременно к бесконечному множеству элементов решетки. Например:

- если $a_t \xrightarrow{R} c_t$, $c_t \xrightarrow{R} b_t$, $t \in T$, то $a_t \xrightarrow{R} b_t$, $t \in T$.

Определение 7 представляет лишь такие логические связи, которые могут быть получены за конечное число шагов. Уровнем рекурсии в соотношении $a \xrightarrow{R} b$ мы будем называть число шагов вывода, необходимое для получения этой связи. При этом учитываются лишь применения правил 3)—6). Для связи, основанной только на правиле 1) или 2), уровень рекурсии считается равным нулю.

Замечание 1. В связи со сказанным при получении любого соотношения $a \xrightarrow{R} b$ длины применяемых транзитивных цепочек (правило 6) ограничены в совокупности.

Поскольку в общем случае данная логическая связь $a \xrightarrow{R} b$ может быть получена не единственным набором шагов определения 7, мы будем лишь оценивать сверху ее уровень рекурсии, не указывая его точного значения.

Следующие две теоремы, взятые из работы [7], мы приводим здесь без доказательств.

Теорема 1. Для произвольного отношения R на полной булевой решетке $\mathbb{F}$ логическое замыкание существует и совпадает с множеством $\xrightarrow{R}$ всех упорядоченных пар, логически связанных отношением R.

Из теоремы 1 нетрудно получить следующее утверждение.

Следствие 1. Пусть R — бинарное отношение на полной булевой решетке $\mathbb{F}$ и $a_t \xrightarrow{R} b_t$, $\forall t \in T$. Тогда отношение $R' = R \cup \{(a_t, b_t) | t \in T\}$ логически эквивалентно R.

Пусть дано произвольное бинарное отношение R на решетке $\mathbb{F}$. Его эквивалентным преобразованием называется такая замена множества упорядоченных пар R (всего или некоторой его части), что полученное в результате новое отношение P логически эквивалентно R, т. е. $P \sim R$.

Теорема 2. Пусть R_1, R_2, R — отношения на полной булевой решетке $\mathbb{F}$. Если при этом $R_1 \sim R_2$, то $R_1 \cup R \sim R_2 \cup R$.

Следствие 1 и теорема 2 обосновывают принцип локальности эквивалентных преобразований логических отношений.

Далее мы покажем, что логическое замыкание произвольного отношения R совпадает с транзи-

тивным замыканием другого отношения $\tilde{R} \supseteq R$, построенного по данному R в виде "контрапозиционно-дистрибутивного многообразия". Это позволяет свести изучение некоторых важных вопросов, касающихся логических отношений, к соответствующим проблемам транзитивных отношений. В частности, построение логического замыкания или редукции можно осуществить с помощью быстрых алгоритмов (типа Уоршолла), разработанных для транзитивных отношений [9]. С этой целью предварительно докажем вспомогательные утверждения.

Лемма 1. Пусть R — бинарное отношение на полной булевой решетке $\mathbb{F}$ и $a_t \xrightarrow{R} b_t$, $\forall t \in T$. Тогда справедливы соотношения $\bigwedge\limits_{t \in T} a_t \xrightarrow{R} \bigwedge\limits_{t \in T} b_t$ и $\bigvee\limits_{t \in T} a_t \xrightarrow{R} \bigvee\limits_{t \in T} b_t$.

Доказательство. Чтобы установить первое из требуемых соотношений, введем обозначения $\tilde{a} = \bigwedge\limits_{t \in T} a_t$, $\tilde{b} = \bigwedge\limits_{t \in T} b_t$, $t \in T$. Поскольку $\tilde{a} \leqslant a_t$, $\forall t \in T$, то в силу $a_t \xrightarrow{R} b_t$ и условия 6) определения 7 справедливо $\tilde{a} \xrightarrow{R} b_t$, $\forall t \in T$. Применяя к последнему соотношению правило 4), получим $\tilde{a} \xrightarrow{R} \tilde{b}$.

Для доказательства второй части леммы обозначим $\tilde{a} = \bigvee\limits_{t \in T} a_t$, $\tilde{b} = \bigvee\limits_{t \in T} b_t$, $t \in T$. Далее, пользуясь неравенствами $b_t \leqslant \tilde{b}$, $\forall t \in T$, из $a_t \xrightarrow{R} b_t$ с помощью транзитивного условия 6) имеем $a_t \xrightarrow{R} \tilde{b}$, $\forall t \in T$. Наконец, используя правило 5) определения 7, приходим к соотношению $\tilde{a} \xrightarrow{R} \tilde{b}$. □

Следствие 2. На основе леммы 1 можно ввести обобщенные правила вывода, соответствующие правилам 4) и 5) определения 7. Упорядоченная пара $a, b \in \mathbb{F}$ логически связана отношением R ($a \xrightarrow{R} b$), если выполнено одно из следующих условий:

4′) существуют такие $a_t, b_t \in \mathbb{F}$, $t \in T$, что $\bigwedge\limits_{t \in T} a_t = a$, $\bigwedge\limits_{t \in T} b_t = b$, причем $a_t \xrightarrow{R} b_t$, $\forall t \in T$;

5′) существуют такие $a_t, b_t \in \mathbb{F}$, $t \in T$, что $\bigvee\limits_{t \in T} a_t = a$, $\bigvee\limits_{t \in T} b_t = b$, причем $a_t \xrightarrow{R} b_t$, $\forall t \in T$.

Очевидно, что правила 4) и 5) определения 7 являются частными случаями правил 4′) и 5′).

Лемма 2. Пусть R — бинарное отношение на полной булевой решетке $\mathbb{F}$. Тогда при выводе произвольной логической связи $a \xrightarrow{R} b$ все дистрибутивные правила 4′)—5′) могут быть применены без участия правила 2) со строгим неравенством. Роль последнего можно свести к присутствию в качестве компонента для транзитивного правила 6).

Доказательство. Предположим, что при получении вывода $a \xrightarrow{R} b$ использовалось условие 4′). Разобьем имеющееся множество пар $\{(a_t, b_t) \mid t \in T\}$ на два подмножества. В первое из них войдут пары $a_t < b_t$; соответствующее им множество индексов $\{t\}$ обозначим T_1. Ко второму подмножеству отнесем все остальные пары (a_t, b_t), $t \in T_2$. Тогда, вводя обозначения $\bigwedge\limits_{t \in T} a_t = a^1$, $\bigwedge\limits_{t \in T} b_t = b^1$, $\bigwedge\limits_{t \in T} a_t = a^2$, $\bigwedge\limits_{t \in T} b_t = b^2$, получим $a = a^1 \wedge a^2$, $b = b^1 \wedge b^2$. При этом $a^1 \leqslant b^1$ (соответственно $a^1 \xrightarrow{R} b^1$) и по условию 4′) имеет место $a^2 \xrightarrow{R} b^2$. Очевидно, что указанное выше применение правила 4′) для $\{(a_t, b_t) | t \in T\}$ можно заменить аналогичным правилом для (a^1, b^1), (a^2, b^2).

Если при этом $T_1 = \varnothing$, то для вывода $a \xrightarrow{R} b$ сформулированное в лемме утверждение относительно правила 4′) выполнено сразу. Если же $T_2 = \varnothing$ либо $a^2 \leqslant b^2$, то применение правила 4′) не требуется вовсе. Рассмотрим оставшийся нетривиальный случай ($T_1 \neq \varnothing$, $T_2 \neq \varnothing$ и не имеет места $a^2 \leqslant b^2$), в котором поступим так. Применим правило 4′) к соотношениям $b^1 \xrightarrow{R} b^1$, $a^2 \xrightarrow{R} b^2$, тогда получим $b^1 \wedge a^2 \xrightarrow{R} b^1 \wedge b^2$. Возвращаясь к имеющемуся неравенству $a^1 \leqslant b^1$, заметим, что в силу свойств решетки $\mathbb{F}$ из него следует $a^1 \wedge a^2 \leqslant b^1 \wedge a^2$. Поэтому, применяя к парам $a^1 \wedge a^2 \leqslant b^1 \wedge a^2$ и $b^1 \wedge a^2 \xrightarrow{R} b^1 \wedge b^2$ транзитивное правило 6), приходим к соотношению $a^1 \wedge a^2 \xrightarrow{R} b^1 \wedge b^2$, т. е. $a \xrightarrow{R} b$. При этом мы исключили в применяемом правиле 4′) участие строгих неравенств вида $a_t < b_t$. Таким образом, для правила 4′) утверждение леммы доказано.

Другой вариант, когда при выводе $a \xrightarrow{R} b$ было применено правило 5′), рассматривается аналогично с заменой операции $\wedge$ на $\vee$. □

Техника доказательства следующих двух лемм аналогична технике доказательства леммы 2. Поэтому в целях экономии места мы его здесь не приводим.

Лемма 3. Пусть R — отношение на полной булевой решетке. Тогда при выводе любой логической связи $a \xrightarrow{R} b$ все применения контрапозиционного правила 3) определения 7 могут быть исключены либо перенесены в начальную стадию этого процесса.

Лемма 4. Пусть R — отношение на полной булевой решетке. Тогда при выводе любой логической связи $a \xrightarrow{R} b$ все применения транзитивного правила 6) могут быть исключены либо перенесены в заключительную стадию данного процесса.

Для произвольного отношения R на полной булевой решетке $\mathbb{F}$ рассмотрим отношение $\tilde{R}$, построенное по данному R последовательным выполнением следующих шагов:

1) добавить к R все пары вида (a, a), где $a \in \mathbb{F}$ (рефлексивные пары), и обозначить новое отношение R_1;

2) добавить к R_1 все пары (a, b), для которых $(b', a') \in R$, и обозначить новое отношение R_2;

3) добавить к R_2 всевозможные пары вида $(a_1 o_1 a_2 o_2 ... o_{m-1} a_m, b_1 o_1 b_2 o_2 ... o_{m-1} b_m)$, где $(a_i, b_i) \in \in R_2$, символ o_i обозначает операцию $\wedge$ либо $\vee$, и обозначить новое отношение R_3;

4) добавить к R_3 всевозможные пары вида $(Q_1 Q_2 ... Q_m a_{t_1 t_2 ... t_m}, \quad Q_1 Q_2 ... Q_m b_{t_1 t_2 ... t_m})$, где $(a_{t_1 t_2 ... t_m}, b_{t_1 t_2 ... t_m}) \in R_3$, символ Q_i обозначает бесконечную операцию вида $\bigwedge\limits_{t_i \in T_i}$ либо $\bigvee\limits_{t_i \in T_i}$;

5) объединить полученное отношение с отношением $\leqslant$.

Заметим, что в силу дистрибутивности булевой решетки шаг 3) можно заменить двумя более простыми последовательными шагами: вначале использовать лишь одну из операций $\wedge$ либо $\vee$, затем — только вторую.

Применяя конечное число раз лемму 1 и следствие 1, нетрудно убедиться в том, что отношение $\tilde{R}$ логически эквивалентно R.

Лемма 5. Пусть R — бинарное отношение на полной булевой решетке $\mathbb{F}$. Тогда, если $a \xrightarrow{R} b$, и это соотношение может быть получено без использования транзитивного правила 6) определения 7, то $(a, b) \in \tilde{R}$.

Доказательство. Пусть имеет место $a \xrightarrow{R} b$, полученное без учета правила 6). Если это соотношение получено непосредственно из условия 1) или 2) определения 7, то сразу имеем $(a, b) \in \tilde{R}$. Остается рассмотреть нетривиальный случай применения цепочки правил 2), 3), 4′), 5′).

Поскольку при выводе правило 6) не используется, то по лемме 2 не применяется и правило 2) со строгим неравенством. Далее по лемме 3 все применения правила 3) могут быть осуществлены на начальном этане процесса вывода. Следовательно, правила 4′)—5′) завершают этот процесс. Также заметим, что все правила 4′)—5′) с бесконечными операциями $\bigwedge$ и $\bigvee$ могут быть применены после всех применений правил 4′)—5′) с конечными операциями $\wedge$ и $\vee$. Этот факт следует из свойств операций $\bigwedge$ и $\bigvee$ по отношению к операциям $\wedge$ и $\vee$ и является алгебраической интерпретацией теоремы о предваренной форме в логике первого порядка [2].

Если сопоставить перечисленные этапы вывода с последовательностью построения отношения $\tilde{R}$, то окажется, что вывод $a \xrightarrow{R} b$ представляет собой построение некоторого подмножества $\tilde{R}$, что и доказывает включение $(a, b) \in \tilde{R}$. □

Теорема 3. Логическое замыкание отношения R на полной булевой решетке $\mathbb{F}$ совпадает с транзитивным замыканием $\tilde{R}^*$ соответствующего отношения $\tilde{R}$.

Доказательство. Как было отмечено, отношение $\tilde{R}$ эквивалентно R. Следовательно, по определению логического замыкания имеем $\tilde{R} \subseteq \xrightarrow{R}$. Отсюда, поскольку $\xrightarrow{R}$ транзитивно, получаем $\tilde{R}^* \subseteq \xrightarrow{R}$.

Докажем обратное включение. Пусть $a \xrightarrow{R} b$. По лемме 4 при получении этого вывода все применения правила 6) могут быть перенесены в заключительную стадию: существует цепочка элементов $a = c_0, c_1, ..., c_n = b$ такая, что выполнены соотношения $c_{i-1} \xrightarrow{R} c_i$, $i = 1, ..., n$, при выводе которых правило 6) не используется. Тогда по лемме 5 имеем $(c_{i-1}, c_i) \in \tilde{R}$, откуда сразу получаем $(a, b) \in \tilde{R}^*$. Итак, $\xrightarrow{R} \subseteq \tilde{R}^*$. □

Рассмотрим далее вопрос о существовании и построении логической редукции бинарных отношений.

Для произвольного отношения R на полной булевой решетке $\mathbb{F}$ рассмотрим отношение $\underset{\sim}{R}$, построенное по данному R последовательным выполнением шагов, обратных построению $\tilde{R}$, а именно:

1) исключить из R все пары (a, b), для которых $a < b$, и обозначить новое отношение R_{-1};

2) исключить из R_{-1} все пары (a, b) вида $(Q_1 Q_2 ... Q_m a_{t_1 t_2 ... t_m}, \quad Q_1 Q_2 ... Q_m b_{t_1 t_2 ... t_m})$, где $(a_{t_1 t_2 ... t_m}, b_{t_1 t_2 ... t_m}) \in R_{-1}$ либо $(b'_{t_1 t_2 ... t_m}, a'_{t_1 t_2 ... t_m}) \in \in R_{-1}$, символ Q_i обозначает бесконечную операцию вида $\bigwedge\limits_{t_i \in T_i}$ либо $\bigvee\limits_{t_i \in T_i}$, причем (a, b) не совпадает ни с одной парой $(a_{t_1 t_2 ... t_m}, b_{t_1 t_2 ... t_m})$, и обозначить новое отношение R_{-2};

3) исключить из R_{-2} все пары (a, b) вида $(a_1 o_1 a_2 o_2 ... o_{m-1} a_m, b_1 o_1 b_2 o_2 ... o_{m-1} b_m)$, где $(a_i, b_i) \in \in R_{-2}$ либо $(b_i', a_i') \in R_{-2}$, символ o_i обозначает операцию $\wedge$ либо $\vee$, причем (a, b) не совпадает ни с одной парой (a_i, b_i), и обозначить новое отношение R_{-3};

4) исключить из R_{-3} все контрапозиционные пары, т. е. при (a, b), $(b', a') \in R_{-3}$ оставить в R_{-3} лишь одну из двух пар, и обозначить новое отношение R_{-4};

5) исключить из R_{-4} все пары вида (a, a), где $a \in \mathbb{F}$ (рефлексивные пары).

Как и выше, для отношения $\tilde{R}$ в силу дистрибутивности булевой решетки шаг 3) может быть заменен двумя более простыми последовательны-

ми шагами: вначале использовать лишь одну из операций $\wedge$ либо $\vee$, затем — только вторую. Также с помощью леммы 1 и следствия 1 нетрудно убедиться в том, что отношение $\tilde{R}$ логически эквивалентно R.

Лемма 6. Пусть R — бинарное отношение на полной булевой решетке $\mathbb{F}$. Для того чтобы R являлось логической редукцией, необходимо и достаточно, чтобы R не содержало ни одной такой пары (a, b), что выполнено соотношение $a \xrightarrow{R\setminus\{(a,b)\}} b$.

Доказательство этой леммы легко проводится методом от противного в обоих направлениях. Следующая теорема указывает достаточное условие существования и способ построения логической редукции.

Теорема 4. Пусть для бинарного отношения R, заданного на полной булевой решетке $\mathbb{F}$, построено соответствующее отношение $\tilde{R}$ (см. выше). Тогда, если для $\tilde{R}$ существует транзитивная редукция R^0, то соответствующее ей отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию исходного отношения R.

Доказательство. Из вышеизложенного следует, что указанное в теореме отношение $\underset{\sim}{R}^0$ логически эквивалентно R. Осталось показать, что $\underset{\sim}{R}^0$ является логической редукцией вообще. Для этого достаточно проверить выполнение для $\underset{\sim}{R}^0$ условия леммы 6.

Пусть $(a, b) \in \underset{\sim}{R}^0$. Требуется показать невозможность логической связи $a \xrightarrow{R^0\setminus\{(a,b)\}} b$. Предположим противное — эта связь существует. Тогда в силу следствия 1 отношение $\underset{\sim}{R}^0\setminus\{(a, b)\}$ эквивалентно $\underset{\sim}{R}^0$. Сразу заметим, что применение правила 1) определения 7 для вывода $a \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} b$ невозможно, поскольку пара (a, b) не содержится в множестве $\underset{\sim}{R}^0\setminus\{(a, b)\}$.

По лемме 4 любая логическая связь может быть построена таким образом, что все транзитивности будут присутствовать лишь в завершающей стадии ее вывода: существует цепочка элементов $a = c_0, c_1, \ldots, c_n = b$ такая, что выполнены соотношения $c_{i-1} \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} c_i$, $i = 1, \ldots, n$, при выводе каждого из которых правило 6) не используется. Отсюда по лемме 5 имеем $(c_{i-1}, c_i) \in \tilde{R}$. Таким образом, при $n > 1$ пара (a, b) оказывается транзитивной в $\tilde{R}$. Она не может содержаться в $\underset{\sim}{R}^0$, являющемся подмножеством транзитивной редукции отношения $\tilde{R}$. Мы получили противоречие исходному предположению $(a, b) \in \underset{\sim}{R}^0$.

Остается исследовать случай $n = 1$. В этой ситуации по лемме 3 рассматриваемая логическая связь $a \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} b$ может содержать применения правил 4′) и 5′) лишь в заключительной стадии вывода. В силу дистрибутивных свойств полной булевой решетки (см. [11]) любая полученная таким образом пара (a, b) описывается шагами 3) и 4) процесса построения отношения $\tilde{R}$. Соответственно при нахождении $\underset{\sim}{R}^0$ (обратный процесс) она будет исключена. Таким образом, мы снова пришли к противоречию исходному предположению $(a, b) \in \underset{\sim}{R}^0$.

При выводе $a \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} b$ не могло использоваться и правило 3) определения 7, поскольку в этом случае при $(a, b) \in \underset{\sim}{R}^0$ соответствующая контрапозиционная пара (b', a') должна быть исключена на шаге 4) построения $\underset{\sim}{R}^0$. Применение правила 2) при $a \neq b$ исключается леммой 2, а случай $a = b$ невозможен ввиду шага 5) процесса построения отношения $\underset{\sim}{R}^0$.

Таким образом, мы исследовали возможные варианты предполагаемого логического вывода $a \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} b$. В результате установлено, что наличие связи $a \xrightarrow{\underset{\sim}{R}^0\setminus\{(a,b)\}} b$ противоречит факту $(a, b) \in \underset{\sim}{R}^0$. Следовательно, отношение $\underset{\sim}{R}^0$ представляет собой логическую редукцию. □

Список литературы

1. **Представление** и использование знаний: Пер. с япон. / Под ред. Х. Уэно, М. Исидзука. М.: Мир, 1989. 220 с.
2. **Расева Е., Сикорский Р.** Математика метаматематики: Пер. с англ. М.: Наука, 1972. 591 с.
3. **Halmos P.** Algebraic logic, IV: Equality in polyadic algebras // Trans. Amer. Math. Soc. 1957. N 86. P. 1—27.
4. **Henkin L., Tarski A.** Cylindric algebras // Proc. of Symposia in Pure Mathematics II, Lattice theory. 1961. P. 83—113.
5. **Махортов С. Д.** Логические отношения на решетках // Вестник ВГУ. Сер. физика, математика. Воронеж, 2003. Вып. 2. С. 203—209.
6. **Махортов С. Д.** О редукции логических отношений на решетках // Вестник факультета ПММ. Воронеж: ВГУ, 2004. Вып. 5. С. 172—179.
7. **Махортов С. Д.** Продукционная логика первого порядка и ее алгебраическая интерпретация // Системы управления и информационные технологии. 2007. № 3 (29). С. 21—26.
8. **Чечкин А. В.** Математическая информатика. М.: Наука, 1991. 416 с.
9. **Aho A. V., Garey M. R., Ulman J. D.** The transitive reduction of a directed graph // SIAM J. Computing 1:2. 1972. P. 131—137.
10. **Биркгоф Г.** Теория решеток: Пер. с англ. М.: Наука, 1984. 568 с.
11. **Сикорский Р.** Булевы алгебры: Пер. с англ. М.: Мир, 1969. 375 с.

УДК 004.415.532.2

С. В. Миронов, нач. отд.,
ФГУП "Концерн "Системпром"

Тестирование компиляторов на программные закладки

Рассмотрена проблема выявления программных закладок в компиляторах и предлагается метод тестирования компиляторов без исходных текстов на программные закладки на основании методов тестирования черного ящика и теории игр. Представлена математическая модель программной закладки, внедряемой в компилятор. Сформулированы рекомендации по практическому применению разработанного метода.

Ключевые слова: *тестирование, компилятор, программные закладки, черный ящик, теория игр, сертификация.*

Введение

Основополагающим подходом к оценке безопасности информационных технологий является использование совокупности определенным образом упорядоченных требований к функциональным механизмам обеспечения безопасности, их эффективности и гарантированности реализации. Результирующая оценка безопасности информационных технологий, проведенная на основе предъявленных требований, имеет качественное выражение. Такой подход положен в основу всех действующих нормативно-правовых документов по сертификации и оценке безопасности информационных технологий.

К основным мероприятиям по оценке степени безопасности программного обеспечения (ПО) относят обязательную сертификацию по требованиям безопасности информации [3, 5].

Однако возможности сертификации ограничены как временными рамками, так и нормативно-правовыми и конструкторскими требованиями. Кроме того, в случае использования разработчиками несертифицированных компиляторов, при проведении сертификационных испытаний программного обеспечения не существует эффективных методов выявления программных закладок, внедряемых компилятором. Еще одной проблемой является несовершенство существующих методов и средств, используемых при сертификационных испытаниях компиляторов.

Для исключения указанных проблем разрабатывается новый метод тестирования компиляторов без исходных текстов на отсутствие программных закладок.

Проблемы сертификации программного обеспечения и компиляторов в ходе существующей нормативной базы

Основным документом, регламентирующим сертификацию ПО, является руководящий документ [3]. На основании этого документа основной задачей сертификационных испытаний является выявление недекларированных возможностей в ПО. Руководящий документ устанавливает классификацию ПО (как отечественного, так и импортного производства) средств защиты информации по уровню контроля отсутствия в нем недекларированных возможностей. Проведенный анализ нормативной и инструментальной базы показал, что существует ряд проблем, связанных с неэффективностью существующих методов и средств, используемых при сертификационных испытаниях программного обеспечения и компиляторов на соответствие требованиям [3].

1. Несовершенство нормативной базы:

- отсутствие сигнатурного анализа в требованиях к испытаниям ПО ниже 2-го уровня контроля, т. е. для программ, которые обрабатывают информацию с уровнем "секретно" и "конфиденциально", что существенно уменьшает вероятность выявления программных закладок;
- отсутствие требований к содержанию базы потенциально опасных конструкций; эта проблема делает неэффективными методы сигнатурного анализа;
- отсутствие механизмов выявления программных закладок, связанных с переполнением буфера, выявление гонок, вызовов ПО программ не из своего адресного пространства, отсутствие очистки памяти;
- отсутствие декларированных способов построения перечня маршрутов выполнения функциональных объектов (ветвей), а также методов корректной оценки его правильности. Этот недостаток сводит проверку к формальному действию и не увеличивает вероятность выявления программных закладок;
- отсутствие механизмов подсчета полноты покрытия маршрутов выполнения функциональ-

ных объектов при динамическом анализе. Данный недостаток сводит проверку к формальному действию и не увеличивает вероятность выявления программных закладок.

2. Несовершенство существующих инструментальных средств, используемых в рамках сертификационных испытаний ПО по требованиям безопасности:

- некорректная обработка исходных текстов программ, написанных на современных языках программирования. Не все анализаторы исходных текстов в достаточной мере обрабатывают возможные конструкции, которые могут быть сформированы исходя из грамматик языков высокого уровня;
- анализаторы исходных текстов имеют ограничения на объем исследуемых текстов;
- реализация для ограниченного числа языков программирования.

3. Организационные проблемы:

- легитимность инструментария. В системе сертификации ФСТЭК России требуется при проведении сертификационных испытаний ПО использовать только сертифицированные инструментальные средства. В настоящее время в системе сертификации ФСТЭК России сертифицирован только один анализатор исходных текстов для с/с++-кода, созданный в 1990-х годах и не обновляемый с того времени. Вследствие этого нельзя использовать новые средства в рамках сертификационных испытаний в системе ФСТЭК России;
- процедуры при изменении исходных текстов ПО (обновлении). В нормативной базе отсутствует информация о разрешении вопросов при изменении исходных текстов: когда должна осуществляться пересертификация, а когда инспекционный контроль. Названные проверки значительно отличаются между собой по трудоемкости.

В настоящее время существует ограниченное число сертифицированных компиляторов. Предпосылкой для этого является еще одна важная проблема, связанная с неэффективностью существующих методов и отсутствием отдельных требований по проверке компиляторов. Данная проблема связана с отсутствием выделения компилятора из множества программного обеспечения. Компилятор является очень сложным инструментарием для создания программного обеспечения, он участвует в интерпретации и трансляции любого программного обеспечения. Компилятор представляется в виде многомодульной структуры, для анализа которой существующими методами потребуются годы исследований множествами экспертов. Отсутствие сертифицированных компиляторов может привести к наличию программных закладок в готовом программном обеспечении без ведома разработчика. Требуется новый метод тестирования компиляторов в рамках сертификационных испытаний, на основании методов тестирования черного ящика, который повысит вероятность и скорость выявления программных закладок.

Модель программной закладки, внедряемой в компилятор

Метод тестирования основывается на модели программной закладки, описанной ниже.

Модель программной закладки должна содержать, по крайней мере, два механизма: механизм активизации и исполнительный механизм. Данные механизмы должны иметь определенную специфику, задаваемую целевым назначением элементов системы программирования.

Исполнительный механизм такой программной конструкции должен реализовывать функцию внедрения программной закладки в генерируемый объектный или загрузочный код.

Механизм активизации, находящийся в составе компилятора, должен быть настроен на семантику преобразуемого исходного кода ПО, определяемую аббревиатурами операндов и содержанием комментариев. Таким образом, механизм активизации должен однозначно определять, в какую программную компоненту необходимо встраивать программную закладку, а в какую нет. В противном случае недекларированная возможность элемента системы программирования будет выявлена экспертом за достаточно короткое время.

Исходя из указанных предпосылок строится следующая модель программной закладки.

Программная закладка состоит из механизма активации U и исполнительного механизма I.

Пусть A — конечный алфавит, тогда X — текст произвольной программы, написанный в соответствии со стандартом языка программирования, который является конечным подмножеством множества слов из A.

Текст программы можно представить в виде графа G, в котором вершины V соответствуют функциям из текста программы, а дуги D — вызовам функций:

$$X \Rightarrow G(V_x, D_x).$$

Механизм активации U можно представить в виде функции u:

$$U : u(X) = a, \text{ где } a = \{0, 1\}.$$

При $a = 0$ программная закладка внедряться не будет, при $a = 1$ будет внедряться.

Исполнительный механизм I можно представить в виде функции i:

$$I(X) = \begin{cases} X \text{ при } a = 0; \\ X' = i(X) \text{ при } a = 1, \end{cases}$$

где $X' \Rightarrow G'(V'_{x'}, D'_{x'})$.

Факт внедрения программной закладки можно обнаружить путем сравнения графов G и G'.

При внедрении программной закладки в компилятор можно составить следующую модель компилятора, в которой его функционирование описывается механизмом компиляции K.

Механизм компиляции зависит от параметров по умолчанию компилятора и настраиваемых параметров. Данный механизм можно представить в виде функции k:

K: $k(X,$ параметры$) = X''$.

Необходимо отметить, что в связи со спецификой компиляторов, для всех компиляторов для одних и тех же X разные X''.

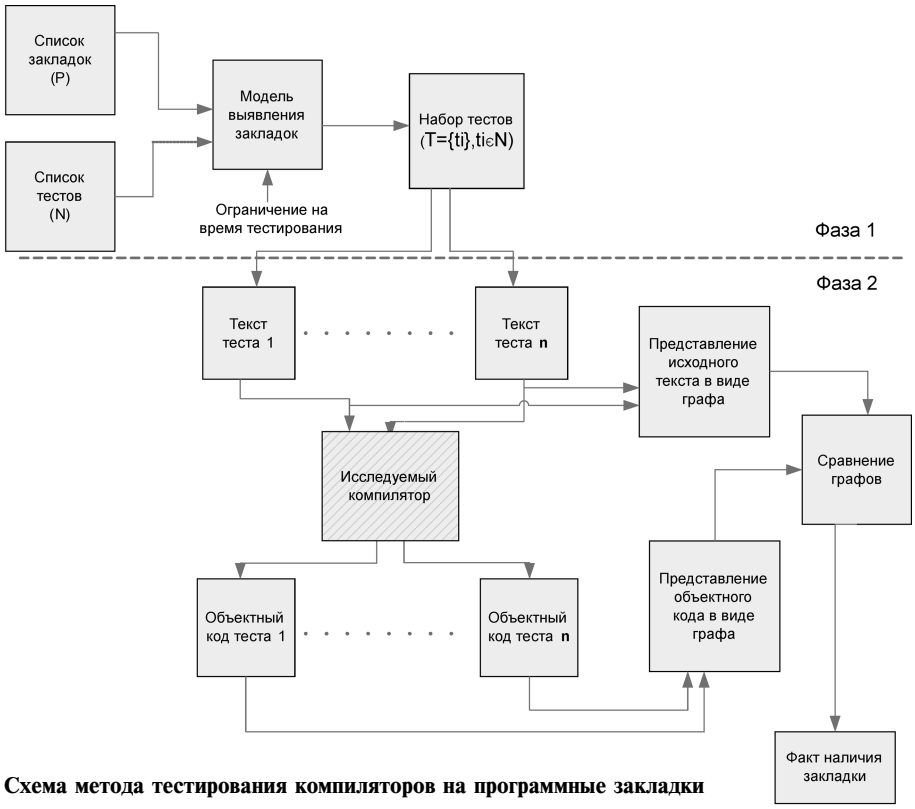


Схема метода тестирования компиляторов на программные закладки

С учетом моделей программной компиляции и самого компилятора компиляция текста программы проходит следующие фазы:

$$X \xrightarrow{i(X)} X' \xrightarrow{k} X''.$$

Метод тестирования компиляторов без исходных кодов на программные закладки

Исходя из особенностей тестирования компилятора без исходных текстов можно свести этот процесс к следующим этапам:

1. Формирование тестового набора.
2. Компиляция тестов из набора.
3. Анализ результатов компиляции.

При тестировании компилятора на программные закладки остановимся на методе тестирования черного ящика — тестирования потока управления и данных [1, 4]. Суть данного метода состоит в представлении исходных текстов и компилированного текста в виде графов и сравнении этих графов для выявления факта внедрения программной закладки.

Для формирования тестового набора с учетом ограничений на время и качество тестирования можно воспользоваться аппаратом теории игр [2], где в качестве одного игрока выступает разработчик компилятора, который хочет внедрить закладку, в качестве другого — эксперт, способный с помощью тестовых наборов выявить факт внедрения. В качестве стоимостной функции могут выступать такие показатели, как вероятность выявления тестов программной закладки, время теста, критичность закладки.

С учетом вышесказанного метод тестирования компиляторов на программные закладки представляется в виде схемы, представленной на рисунке.

Метод разбивается на две фазы.

Фаза 1. *Выбор тестов для тестирования компилятора.*

На этой фазе формируется возможный набор тестов и программных закладок.

Строится игра с учетом допустимой точности результата и максимального времени тестирования компилятора. На основании этой игры строится оптимальная стратегия поведения проверяющего. Для выбранной стратегии поведения отбираются тесты.

Фаза 2. *Анализ объектного кода на предмет наличия внедренных программных закладок.*

На этой фазе происходит непосредственное выявление программных закладок. Сформированные тестовые данные подаются на вход исследуемого компилятора с целью получения транслированного (объектного) кода. На основании сравнения структур объектного кода происходит анализ результата тестирования. В случае выявления различия делается вывод о возможности присутствия в компиляторе программной закладки.

Разработанный метод позволяет тестировать компиляторы на программные закладки при условии как наличия, так и отсутствия программных закладок. Данный метод учитывает специфику сертификационных испытаний и современной динамики развития технологий автоматизированных систем военного назначения.

Рекомендации по практическому применению

Метод может использоваться как в ходе сертификационных испытаний, так и при разработке самих компиляторов. При формировании перечня тестов можно с учетом ограничения на время проведения испытаний и точность результатов подобрать оптимальные тестовые наборы, которые будут использоваться для тестирования компиляторов. Метод не имеет ограничений на количество тестов и программных закладок.

Выводы

Таким образом, в статье представлены проблемы и несовершенства нормативной базы по сертификации как программного обеспечения, так и компиляторов. Описана модель программной закладки, на основе которой построен метод тестирования компиляторов без исходных кодов на программные закладки. Данный метод основан на методах тестирования "черного ящика" и теории игр. Представленный метод позволит повысить вероятность выявления программных закладок в компиляторах в условиях современной нормативной базы в области сертификации программного обеспечения.

Список литературы

1. **Бейзер Б.** Тестирование черного ящика. Технологии функционального тестирования программного обеспечения систем. СПб.: Питер, 2004.
2. **Венцель Е. С.** Элементы теории игр. Популярные выпуски по математике. Вып. 32.
3. **Гостехкомиссия РФ.** Руководящий документ. Защита от несанкционированного доступа к информации. Часть 1. Программное обеспечение средств защиты информации. Классификация по уровню контроля отсутствия недекларированных возможностей. М.: Военное издательство, 1999.
4. **Липаев В. В.** Обеспечение качества программных средств. Методы и стандарты. М.: СИНТЕГ, 2001.
5. **Марков А. С., Миронов С. В., Цирлов В. Л.** Выявление уязвимостей в программном коде // Открытые системы. 2005. № 12.

ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ В ОБРАЗОВАНИИ

УДК 004.89.001

Л. Г. Гагарина, д-р техн. наук, проф., зав. каф.,
Н. С. Фоминова, аспирант,
И. С. Калинников, студент,
Московский институт электронной техники
(технический университет)

Теоретические основы методики интеллектуального тестирования

Рассматриваются такие аспекты компьютерного тестирования, как правомерность замены им процесса собеседования, а также существование соответствующей модели программного обеспечения тестирования, являющегося аналогом собеседования. Предложена концепция интеллектуального тестирования, рассмотрены три метода ее реализации.

Ключевые слова: *интеллектуальное тестирование, автоматизированная оценка знаний, смысловые связи вопросов, интерполяция, аппроксимация, получаемая информация.*

Объектом исследования данной работы является адекватность оценки знаний, эталоном которой, как правило, считается решение преподавателя, принятое в результате собеседования при условии его вопросной полноты.

В качестве предмета исследования рассматриваются такие аспекты компьютерного тестирования, как правомерность замены им процесса собеседования преподавателя и обучаемого, а также наличие соответствующей модели программного обеспечения тестирования, являющегося аналогом собеседования.

Решение поставленных проблем получено на основе анализа особенностей существующих компьютерных систем массового опроса и требований к программному обеспечению, способному приблизить компьютерный опрос (тест) к диалогу обучающего и обучаемого. Такое приближение в дальнейшем называется системой интеллектуального тестирования (СИТ).

Основными характеристиками для оценки существующих компьютерных систем тестирования во время проведения работы были информационная полнота и структурная связность опросов, созданных системами.

Роль структуры опроса в оценке знаний

Известно, что во время обучения периодически проводятся контрольные мероприятия, целью которых является проверка уровня усвоения и понимания преподаваемой дисциплины. При этом оценка должна быть как можно более адекватной, так как она влияет на суть учебного процесса в дальнейшем. Традиционно одним из самых предпочтительных методов оценки знаний является собеседование, но практически в учреждениях высшего, среднего профессионального образования и аналогичных им диалоги преподавателя и обучаемого в силу ряда объективных причин заменяют контрольными работами, диктантами, тестами, самостоятельными работами и т. д.

На проведение контрольного мероприятия уходит меньше времени, чем на собеседование (коллоквиум), однако процесс составления заданий и проверки ответов является достаточно длительным. Наличие вариантов заданий и автоматизированная проверка помогают достигнуть требуемой скорости проведения контрольных мероприятий, но, если принимать во внимание только скорость проведения опросов, утрачивается суть проверки знаний.

Дело в том, что **собеседование** предполагает возможность преподавателя оценить каждый ответ обучаемого и, учитывая всю предысторию вопросов/ответов, задать уточняющий вопрос. Таким образом, выбор множества вопросов не случаен, а приводит к наискорейшей адекватной оценке знаний. Отметим, что в данных рассуждениях не учитывается элемент субъективности, т. е., по сути, преподаватель является некоторой машиной с объемной памятью и способностью к самостоятельному анализу. Такая схема крайне примитивна по многим причинам, однако она помогает наметить базис: опрос имеет динамическую структуру (все вопросы задаются в контексте предыдущих ответов), при этом все изменения структуры служат одной цели — однозначной оценке знаний. Такой опрос будем называть **контекстно-связанным**, так как суть опроса зависит от порядка задания вопросов и имеющегося контекста.

Контрольная работа предполагает оценку знаний обучаемого по имеющимся ответам на эталонные задания, которые, однако, сгруппированы определенным образом. При этом преподаватель, проанализировав результаты работы на основе представленных ответов, вынужден без привлечения дополнительной информации оценить знания однозначно. Очевидно здесь следует говорить об опросе со статической структурой, причем выбор множества вопросов не случаен, а фактически является решением задачи поиска оптимального алгоритма опроса. Такой опрос также можно считать контекстно-зависимым, хотя и с большими ограничениями.

Компьютерное тестирование предполагает возможность оценки знаний на основе ответов на случайно выбранные вопросы. По крайней мере, так устроено абсолютное большинство существующих компьютерных тестовых программ. При этом оценка проводится по процентному отношению верных ответов к общему числу вопросов. В данном случае опрос не имеет структуры (в том смысле, что опрос не является контекстно-зависимым, суть опроса не меняется от перестановки вопросов и т. д.), что, на наш взгляд, не дает возможности адекватной оценки знаний. Ведь такая методика фактически основывается на полной независимости знаний по отдельным вопросам в смысле усвоения этих знаний учащимся, что заранее неверно.

Собеседование, контрольные работы и компьютерное тестирование имеют существенные различия в том плане, что первые два метода предполагают ответы в так называемой "свободной" форме, а последний — в "закрытой".

Положим, что любой опрос с ответами в "свободной" форме можно свести к опросу на основе множества вопросов с ответами в "закрытой" форме, при этом структура опроса изменится.

Для проведения дальнейших рассуждений введем рабочее определение термина "количество знаний". Количеством знаний будем называть характеристику способностей человека к решению нетривиальных неэталонных задач в той области, в которой следует оценить знания. Положим также, что количество знаний прямо характеризует их качество. *Будем считать, что только бесконечное множество несвязанных эталонных заданий позволяет судить о знаниях с максимально возможной степенью адекватности.* Фактически данное требование имеет следующий педагогический смысл: ответы на любое конечное число эталонных заданий могут быть просто заучены, без понимания теории, которая позволяет дать ответы на данное множество вопросов и еще бесконечное множество других.

Очевидно, что длина компьютерного теста (в данном случае для оценки знаний тестируемого) должна стремиться к бесконечности. Логично потребовать того же от длины собеседования, контрольной работы и вообще любого другого контрольного мероприятия, однако такое заключение является слишком поверхностным.

Отвечая на вопрос в ситуации контекстно-независимого опроса, мы получаем только ответ на данный вопрос и ничего больше. То есть, как следует из теории информации Шеннона [1], мы снимаем неопределенность, равную $i = \log_2$ (*чис-*

ло вариантов ответа). Когда речь идет о нескольких ответах, информация просто складывается $i = \sum_{i=1}^{n} \log_2(w_i)$, где n — число заданных вопросов, а w_i — число вариантов ответа.

В случае же с опросом, имеющим некоторую структуру, ситуация становится несколько сложнее: задавая некоторые вопросы, мы аналогично получим информацию $i = \sum_{i=1}^{n} \log_2(w_i)$, но еще мы получим некоторую информацию и об ответах на другие вопросы $i^* = \sum_{j=1}^{n} \log_2\left(\frac{w_j}{w_j^*}\right)$ (w_j — число вариантов ответа до задания вопроса, связанного с данным, а w_j^* — число вариантов ответа после задания вопроса), которая, по сути, определяется наличием структурных связей между вопросами. Неверно считать, что такая информация отсутствовала в первом случае, она существовала вне зависимости от того, как был устроен опрос. Дело лишь в том, что в первом случае мы не имели понятия о соотношениях между вопросами и ответами на них, т. е. отсутствовала какая-либо логическая структура, позволяющая данную информацию зафиксировать или осмыслить, а во втором случае она существовала, что позволило получить соответствующую информацию [2].

Приведенные выше рассуждения позволяют построить качественное описание явлений, связанных с оценкой знаний, и сделать некоторые предположения относительно соответствующей методики компьютерного тестирования.

Реализация контекстно-зависимого компьютерного тестирования

Как показано выше, опрос, создаваемый компьютером, должен иметь динамическую структуру, т. е. компьютер каким-то образом должен анализировать набор полученных ответов на уже заданные вопросы и задавать новый вопрос в контексте информации, полученной в результате этого анализа. При этом новый вопрос должен приносить максимально возможное количество информации, которая будет использована компьютером в процессе предъявления следующего вопроса.

Оценка знаний, полученная путем вычисления процентного отношения верных ответов к общему числу вопросов, не является верной в случае наличия смысловых связей между вопросами. Наоборот, можно констатировать, что она является наиболее неверной в этом конкретном случае. Поэтому компьютер, помимо простой проверки ответа на достоверность, должен оценивать верность ответа в контексте предыдущих и даже будущих вопросов, именно такая трактовка ответов в "закрытой" форме позволяет достичь той же гибкости, которая присуща ответам в форме "открытой".

Удовлетворить всем перечисленным требованиям крайне непросто, однако только такое компьютерное тестирование может претендовать на некоторую объективность и являться не просто генератором случайных чисел. *Договоримся также такое тестирование называть интеллектуальным,* чтобы обозначить его отличие от традиционно существующих методик компьютерного тестирования.

Все представленные выше рассуждения нуждаются в важном пояснении: говоря о контрольных мероприятиях, включающих использование компьютера на стадиях создания, проведения и проверки результатов опроса, мы сразу переходим к тестированию, особенностью которого является так называемая "закрытость" ответа. Такой переход не случаен — ведь компьютер не имеет ни малейшего понятия о семантике вопросов и ответов, следовательно, правильность/неправильность одного конкретного ответа может быть оценена только в случае определенности всех возможных вариантов ответа и при наличии информации о верном ответе.

Вместе с тем, для того чтобы компьютер мог генерировать опросы с динамической структурой, не используя списков или деревьев опроса, необходимо, чтобы вопросы были упорядочены по смыслу. Поэтому можно рассматривать базу вопросов как часть некоторого пространства Δ, метрика которого такова, что из схожести вопросов по смыслу, сложности и другим характеристикам следует, что они расположены близко, и, наоборот, из их взаимной близости следует то, что их характеристики схожи.

Если каждому заданному вопросу ставить в соответствие верность ответа от 0 до 1 ($0 \leqslant h \leqslant 1$), то на каждом шаге мы будем иметь дискретное отображение, однозначно ставящее некоторым векторам пространства вопросов в соответствие верность ответа:

$$I = \{(\boldsymbol{x},\ h(\boldsymbol{x}) | \boldsymbol{x} \in \Delta\}_{i-1}^{n},$$

где I — однозначное дискретное отображение; $\boldsymbol{x}$ — вектор пространства вопросов Δ; $h(\boldsymbol{x})$ — верность ответа; n — число заданных на текущий момент вопросов.

Такое представление базы вопросов и ответов позволяет упростить рассуждения о возможных методах реализации интеллектуального тестирования. Вместе с тем такое представление интерес-

но само по себе, так как позволяет программе, не имеющей понятия о смысле вопросов, группировать их и работать с ними по смыслу, как это делает человек.

Самым простым решением проблемы интеллектуального тестирования может быть передача преподавателем некоторых методов построения опроса компьютеру, при этом нет даже необходимости в упорядочивании вопросов по смыслу.

Для тестирования создаются логические деревья опроса: вводятся несколько предикатов, например, P_t — предикат верности ответа, P_s — предикат достаточной скорости ответа и т. д., а затем в компьютере реализуется алгоритм опроса на некотором языке программирования или в виде дерева с условными переходами. При этом не предполагается необходимость указания дерева для каждого вопроса, а возможно создание деревьев с множествами вопросов в узлах и возможна замкнутость дерева (рис. 1, 2).

```
10 ASK (RND (SET (4,67,8,758,56,75)))
REM выбрать случайный вопрос из множества
вопросов в базе, имеющих номера
4,67,8,758,56,75, задать вопрос и ждать
ответа
20 IF AND (Pt,Ps) THEN 40
REM если ответ верный и скорость
достаточна, перейти к инструкции 40
30 GOTO 10
REM в противном случае вернуться назад и
снова выбрать случайный вопрос из того же
множества
40 ASK (1)
REM задать вопрос, в базе имеющий номер 1
```

Рис. 1. Пример программы, описывающей алгоритм задания вопросов

Язык программирования для реализации такой системы может быть любым, хотя, очевидно, наилучшим выбором, наверное, был бы Lisp, так как он изначально предназначался для создания и исполнения логических деревьев [3].

При использовании описанных выше подходов возникают трудности двух видов: во-первых, неудобно перечислять все вопросы по номерам; во-вторых, непонятно, как организовать контекстную оценку знаний учащегося.

Первая проблема решается достаточно просто, учитывая, что теперь степень похожести вопросов легко определяется расстоянием между векторами вопросов в пространстве вопросов Δ.

Очевидным решением второй проблемы является введение правил изменения оценки, как это делается в атрибутных грамматиках Кнута [4], где каждому правилу порождения ставится в соответствие правило изменения некоторых переменных, отражающих смысловую суть кода. Это значит, что каждому переходу в логическом дереве необходимо поставить в соответствие некоторое правило изменения переменной, которая характеризует оценку учащегося в текущий момент.

Предложенное выше решение принципиально удовлетворяет данному нами определению интеллектуального тестирования, однако такая реализация не может считаться эталонной по нескольким причинам. К ним следует отнести большую трудоемкость составления и отладки алгоритмов опроса, их ограниченную применимость (каждый алгоритм привязан к своей базе вопросов), большую скорость исчерпания различных вариантов развития событий в заранее заданном алгоритме и т. д. Преимуществами алгоритма являются простота, легкость реализации и полная предсказуемость.

Другим интересным решением проблемы интеллектуального тестирования является использование алгоритма, основанного на нейросетевых классификаторах. Суть алгоритма в следующем: ответы тестируемого позволяют выделить классы похожих друг на друга вопросов — в зависимости от верности ответов. Далее, выявляя закономерности в распределениях вопросов тех или иных классов в пространстве вопросов Δ, мы можем все остальные еще не заданные вопросы, принадлежащие Δ, отнести в один из классов и, выбрав по

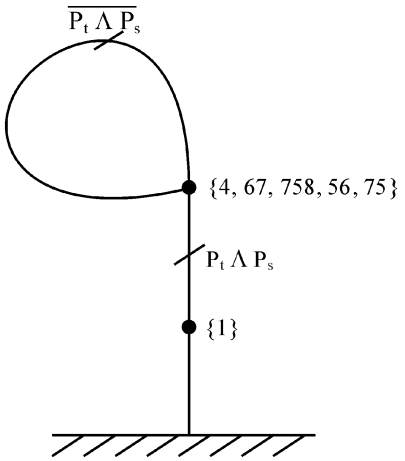


Рис. 2. Пример логического дерева, описывающего алгоритм задания вопросов (алгоритм аналогичен программе на рис. 1. Дерево содержит два узла с множествами, в которых перечислены номера вопросов в базе и три перехода, один из которых означает завершение опроса)

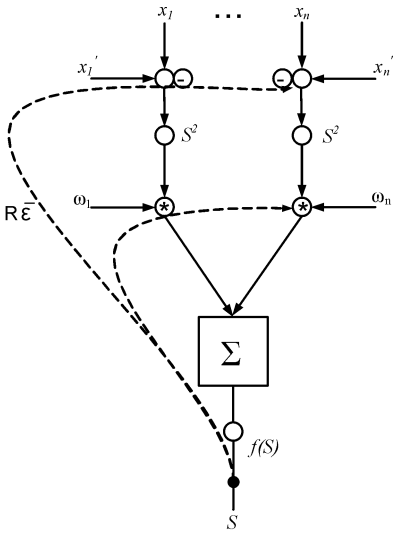


Рис. 3. Структура нейрона классификатора.
На вход подаются два вектора: $\boldsymbol{x}$ — классифицируемый и $\boldsymbol{x}'$ — вектор самого класса — далее вычисляется модуль разности между векторами по каждому из измерений в пространстве Δ, затем модули домножаются на весовые коэффициенты, суммируются и прогоняются через некоторое отображение f

некоторым критериям один из классов, задать новый вопрос именно из этого класса. Получив ответ, мы имеем возможность оценить точность совершенного индуктивного перехода (определение границ классов и перенесение на них одной верности ответа).

Практическая реализация такого подхода требует наличия двух нейронных сетей — классификатора вопросов и сети оценки верности ответа, а также анализатора. Классификатор разбивает все имеющиеся в базе вопросы по классам, которые созданы анализатором на основе верности ответов тестируемого и тематической близости вопросов. Затем анализатор выбирает самый расплывчатый в смысле отношения верности к числу вопросов класс и задает вопрос из этого класса. После чего нейронная сеть оценки верности передает анализатору верность ответа тестируемого на этот вопрос, а анализатор ищет класс, к которому относится этот вопрос, или создает новый класс. При этом нейроны классификатора имеют структуру, представленную на рис. 3.

По сути, их задачей является проверка, лежит ли конец вектора вопроса в некоторой окрестности эталонного вектора, вектора класса. По форме эта окрестность представляет собой n-мерный эллипс [5]. Нейронная сеть оценки верности ответа имеет в принципе ту же структуру, но на конечный сигнал S, получаемый после суммирования, требуется воздействовать другой функцией.

Основной проблемой данного решения является сложная динамика собственных векторов классов $\boldsymbol{x}'$, которые изменяются после появления в классе нового элемента (на это указывают связи, обозначенные штриховой линией на рис. 3). Явная зависимость этого вектора от порядка вхождения вопросов в класс только усложняет задачу. Кроме этого, каждый вопрос может входить в большое число классов, что разумно с точки зрения построения опроса, но усложняет учет заданных и незаданных вопросов.

Реализация интеллектуального тестирования описанным методом сложна в силу описанных проблем, однако формально она аналогична решению данной проблемы с помощью метода интерполяции функции знаний, о котором и будет идти речь далее.

Интерполяция функции сложности заданий как доступный метод реализации интеллектуального тестирования

Одним из возможных методов реализации интеллектуального тестирования является интерполяция функции сложности заданий. Принципиально проведя опрос, мы получаем дискретное отображение (I, см. ранее), где всем заданным вопросам из пространства вопросов ставится в соответствие верность от 0 до 1. Однако мы можем предполагать, что верность ответов тестируемого в окрестности любого вопроса (в пространстве вопросов Δ) не будет сильно изменяться, так как заданный вопрос и вопросы, лежащие в его окрестности, будут схожи и по теме, и по сложности (см. ранее описание пространства Δ).

Тогда принципиально можно провести оценку знаний учащегося, используя функцию сложности заданий (назовем ее $h(x)$ по аналогии с нашим обозначением для верности h, так как функция ставит в соответствие каждому вопросу из Δ верность ответа), т. е. на основе предположений о верности его ответов на все множество вопросов Δ, связи между которыми нам уже известны. Рассмотрим следующий частный случай. Все вопросы характеризуются только темой, при этом тематическая близость вопросов определяется модулем расстояния между точками-вопросами. Пространство Δ — отрезок [0, 1] и часть $\Re^1$. Поскольку число вопросов в базе конечное, то естественно бесконечному числу точек из Δ не будут соответствовать никакие вопросы из базы, однако это не мешает делать предположения об ответах на вопросы с такими координатами (как если бы они существовали). После некоторого шага опроса полученный результат будет выглядеть так, как показано на рис. 4 (см. вторую сторону обложки).

Теперь можем получить приближение функции $\boldsymbol{h}(\boldsymbol{x})$ сложности заданий для учащегося ($h(x)_n$ непр. на Δ, имеющую $n-1$ производных на Δ) на основе дискретного отображения I путем интерполяции. Свойства функции $h(x)_n$ таковы, так как, используя n точек из I и применяя метод ин-

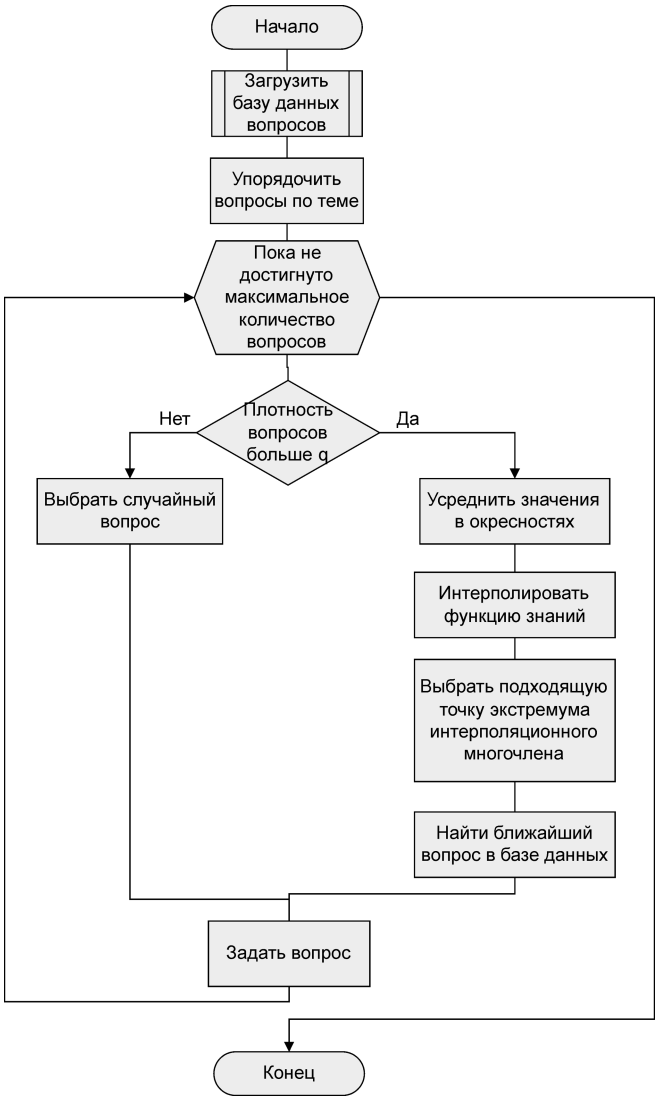


Рис. 5. Схема алгоритма интеллектуального тестирования

терполяции Ньютона, мы получим многочлен n-й степени, а он непрерывен и имеет n производных. Кроме того, вопрос о непрерывности решается из других посылок: если вопросы тематически близки, то $|x_1 - x_2| \to 0$ (см. описание Δ), и на вопросы, которые по сути являются разными формулировками одного вопроса, тестируемый, если он не просто щелкает по кнопкам, не может ответить по-разному, $|h(x_1) - h(x_2)| \to 0$.

Однако пусть есть еще некоторое число вопросов, которые можно задать для того, чтобы прояснить характер знаний учащегося там, где данная функция $(h(x)_n)$ ведет себя наиболее странным образом. Очевидно, что схожим принципом руководствуется педагог, выбирая новые вопросы при собеседовании. Фактически на каждом шаге проводим интерполяцию и выбираем новый вопрос, анализируем $h(x)_n$ и задаем его, снова интерполируем функцию — $h(x)_{n+1}$ и т. д.

Остается решить вопрос о методе выбора новых точек. Конечно, если мы ничего не знаем о знаниях человека, случайный выбор, наверное, будет наилучшим [6], однако по мере уточнения интерполяции $h(x)$, которую мы называем $h(x)_n$ — по номеру шага интерполяции, возможно, есть другой, лучший метод выбора. Для ответа на этот вопрос можно рассмотреть $\int_0^1 |h(x) - h(x)_n| dx$ — общую ошибку интерполяции на отрезке. Оказывается, она будет минимальной в среднем, если выбирать новые точки-вопросы по экстремумам производных $h(x)_n$, что в среднем совпадает с максимумом получаемой информации.

В таком случае алгоритм интеллектуального тестирования выглядит, как показано на рис. 5.

Оценку следует выставлять на основе полученной интерполяции функции знаний $h(x)$, используя, например, такую формулу:

$$m = \Sigma k_m < h(x)^{(n)} >_x,$$

где $h(x)$ — конечный интерполяционный многочлен; k — коэффициент влияния соответствующей производной.

Описанный метод оценки явно имеет преимущество перед традиционными по следующим причинам.

1. Благодаря тематическому упорядочиванию вопросов и связям между ними стало возможно говорить о функции сложности заданий $h(x)$.

2. Благодаря пошаговому процессу и возможности выбора максимума получаемой информации мы получили интерполяцию $h(x)$ с возможной точностью.

3. Используя эту функцию, мы смогли выставить оценку, учитывая не только число верных ответов, но и более глубокую динамику прохождения теста учеником.

Следовательно, возможно использование данного метода в качестве технологии реализации интеллектуального тестирования, которым предлагается заменить все остальные методы компьютерного тестирования.

Некоторые итоги

В работе в сжатой форме сформулирована предлагаемая концепция интеллектуального тестирования, рассмотрены три метода реализации этой концепции, при этом проанализированы сложности, которые могут возникнуть при использовании того или иного алгоритма.

Мы уверены, что данная теория, позволяющая только качественно описывать некоторые явления, может со временем быть развита в теорию,

дающую количественное описание явлений, связанных с зарождением новой информации, имеющем место при ответе ученика на структурно связанные вопросы.

На основании приведенных выше теоретических положений осуществлена программная реализация комплекса интеллектуального тестирования студентов Московского института электронной техники; в течение последующего полугодия будет завершена его опытная эксплуатация.

Список литературы

1. **Андерсон Д. А.** Дискретная математика и комбинаторика: Пер. с англ. М.: Вильямс, 2003. 960 с.
2. **Чернавский Д. С.** Синергетика и информация (динамическая теория информации). М.: Едиториал УРСС, 2004. 288 с.
3. **Пратт Т., Зелковиц М**. Языки программирования: разработка и реализация. СПб.: Питер, 2002. 686 с.
4. **Карпов Ю. Г.** Теория и технология программирования. Основы построения трансляторов. СПб.: БХВ-Петербург, 2005. 272 с.
5. **Хайкин С.** Нейронные сети: полный курс, 2-е изд. М.: Вильямс, 2006. — 1104 с.
6. **Винер Н.** Кибернетика или управление и связь в животном и машине. М.: Сов. Радио, 1968.

УДК 004.3.06

Г. А. Доррер, д-р техн. наук, проф.,
А. А. Попов, аспирант,
Г. М. Рудакова, канд. физ.-мат. наук, доц.,
К. В. Сысенко, аспирант,
Сибирский государственный технологический университет, г. Красноярск

Оптимальная группировка разделяемых единиц контента в учебные модули на базе системы БиГОР

При использовании модульно-рейтинговой технологии обучения важную роль играет рациональное построение учебных модулей. Большие возможности для решения этой задачи, особенно в условиях частой смены образовательных программ, открывает использование системы БиГОР, разработанной в МГТУ им. Н. Э. Баумана. Предложена методика оптимальной группировки разделов курса (разделяемых единиц контента) в равноценные модули. Методика проиллюстрирована на примере курса "Основы теории управления".

Ключевые слова: *учебные модули, разделяемые единицы контента, система БиГОР, модульно-рейтинговая технология обучения, оптимизация структуры учебного курса.*

Введение

Переход высшей школы на новые образовательные стандарты, использование компетентностного подхода и модульно-рейтинговых технологий обучения при планировании и проведении учебного процесса требуют использования эффективных средств генерации образовательных ресурсов. Актуальность этой проблемы также связана с растущей мобильностью учебного процесса, вызванного как научно-техническим прогрессом, так и быстро меняющимися требованиями потребителей образовательных услуг — предприятий, фирм и отдельных выпускников. Особенно велики требования к мобильности создания образовательных ресурсов в системах повышения квалификации и переподготовки кадров. Это вызвано большим разнообразием учебных программ, междисциплинарным характером многих курсов, необходимостью адаптации к уровню и потребностям слушателей, а также короткими сроками, отводимыми на создание и использование новых учебно-методических комплексов.

К числу эффективных программно-методических средств, которые позволяют облегчить выполнение указанных требований, относится, на наш взгляд, система БиГОР, разработанная в МГТУ им. Н. Э. Баумана под руководством И. П. Норенкова [1]. Эта система предназначена для создания и сопровождения баз учебных материалов, синтеза новых электронных учебных пособий (ЭУП) в соответствии с технологией разделяемых единиц контента (ТРЕК), а также для использования созданных пособий обучаемыми.

В Сибирском государственном технологическом университете (СибГТУ) система БиГОР используется с 2006 г. на факультете автоматизации и информационных технологий, а также в институте повышения квалификации преподавателей. Возможности этой системы рассматривались и были высоко оценены научно-методическим советом вуза. Были отмечены следующие достоинства системы БиГОР:

- простота создания новых учебных пособий пользователями-преподавателями;
- автоматическое преобразование текста в гипертекст;
- автоматическое упорядочение модулей, отобранных для включения в учебное пособие;

- широкие возможности представления текстов с разнообразными специальными символами (например, математическими), с включением разнообразных иллюстраций, внешних документов и т. п.

При проведении учебного процесса в СибГТУ использовались как готовые пакеты, созданные в МГТУ им. Н. Э. Баумана и вошедшие в базу учебных материалов, так и новые образовательные ресурсы, разработанные авторами с помощью средств системы. По тематике, связанной с системой БиГОР, выполнены два дипломных проекта.

Накопленный к настоящему времени опыт работы с данной системой позволяет оценить ее достаточно высоко как эффективное средство создания новых образовательных ресурсов с широким использованием уже имеющихся в системе разделяемых единиц контента. Важное значение имеют встроенные средства, облегчающие создание и работу с образовательными ресурсами, такие как авторская подсистема, допускающая выделение в тексте понятий, имеющихся в тезаурусе, формирование метаданных модулей и получение гипертекста. Достаточно удобны при работе также обучающая, информационно-поисковая и диагностическая системы.

Некоторое неудобство при работе с системой связано с отсутствием в ее составе развитого тестирующего компонента. Поэтому систему БиГОР мы использовали совместно с тестирующими системами LearningSpace v. 2.5 и АСТ.

При подготовке учебно-методических материалов в рамках модульно-рейтинговой технологии обучения типовой задачей является разбивка курса на учебные модули. При этом под модулем, в отличие от [1], мы понимаем не одну из разделяемых единиц контента, а логически и методически завершенную часть учебного курса, изучение которой завершается прохождением контрольных процедур (тестирования; защиты лабораторных работ, рефератов и др.) и начислением обучаемому определенного числа зачетных (кредитных) единиц. Таким образом, модуль может содержать несколько единиц контента.

В ряде случаев, особенно при разработке объемных курсов, содержащих разнообразную информацию из различных научных дисциплин, построение логически связанной последовательности равноценных по сложности модулей является непростой задачей. В таких случаях целесообразно использовать формализованные процедуры синтеза модулей из набора заданных блоков (единиц контента). При этом возможности авторской подсистемы БиГОР, которая отслеживает появление новых терминов во вновь вводимых единицах контента, оказываются весьма полезными.

В настоящей статье предлагается методика синтеза курса из имеющегося набора единиц контента. Ставятся требования к сохранению логической последовательности материала, минимальной трудоемкости изучения курса и, кроме того, к одинаковой (по возможности) трудоемкости отдельных модулей. Близкие по смыслу задачи рассматривались ранее в работе [2].

Постановка задачи

Рассмотрим формальную постановку задачи. Пусть учебный курс в системе БиГОР состоит из множества априори заданных тематических блоков (единиц контента), дополненных контрольными мероприятиями:

$$\Sigma = \{S_1, \ldots, S_n\}.$$

Каждый блок S_i, $i = 1, \ldots, n$, характеризуется следующими параметрами:

R_i — общая трудоемкость блока; Θ_i — объем учебного материала (страниц или академических часов); t_i — число вновь введенных терминов и понятий; λ_i — коэффициент сложности материала блока; l_i — коэффициент сложности терминов и понятий; z_i — трудоемкость организации контрольных мероприятий; u_i — коэффициент увеличения трудоемкости за счет консультаций и дополнительных мероприятий, проводимых вне системы БиГОР.

Трудоемкость изучения i-го блока измеряется в некоторых условных единицах, которые могут быть привязаны либо к числу академических часов, отводимых на изучение блока, либо к объему соответствующего текста в учебном пособии. Эта величина определяется формулой, полученной авторами в [2] на основе теории обучения Л. А. Растригина [3] и в настоящей работе адаптированной к особенностям системы БиГОР:

$$R_i = \mathbf{e}^{\lambda_i \theta_i}(u_i \theta_i + l_i t_i) + z_i. \qquad (1)$$

Кроме того, для каждой пары блоков S_i, $S_j \in \Sigma$ установлен логический порядок их изучения: отношение $S_i < S_j$ означает, что блок S_i должен быть изучен раньше блока S_j, а отношение $S_i = S_j$ означает, что эти блоки могут изучаться в произвольном порядке.

Задача состоит в том, что указанные блоки должны быть скомплектованы в модули таким образом, чтобы трудоемкость изучения курса в целом была бы наименьшей. Эффект оптимизации состоит в следующем. При большом объеме учебного материала модуля снижается вероятность его освоения с первого раза, требуется повторное изучение и прохождение контрольных мероприятий. При уменьшении объема модуля повышается вероятность его успешного прохождения, однако

число контрольных мероприятий также увеличивается, что сказывается на общей трудоемкости. Как показано в [2], существует такой объем модуля, при котором суммарная трудоемкость изучения курса минимальна.

Каждый модуль может включать один или несколько блоков, причем упорядоченность блоков должна сохраняться как внутри каждого модуля, так и в целом по курсу. Требуется сформировать $m \leqslant n$ модулей, причем это число априори не известно. Каждый из модулей содержит множество блоков

$$\Sigma_j = \{S_{1j}, S_{2j}, ..., S_{lj}\}, j = 1, ..., m, \quad (2)$$

где S_{kj} — k-й блок j-го модуля, причем

$$\bigcup_{j=1}^{m} \Sigma_j = \Sigma, \Sigma_j \cap \Sigma_l = 0 \text{ при } j \neq l. \quad (3)$$

Первое равенство в этом выражении означает, что должны быть использованы все блоки, а второе — что разные модули не должны содержать одинаковые блоки.

Характеристики каждого модуля определяются на основе характеристик вошедших в него блоков:

$$\bar{\Theta}_j = \sum_{S_k \in \Sigma_j} \Theta_k; \ \bar{t}_j = \sum_{S_k \in \Sigma_j} t_k; \ \bar{z}_j = \max z_k;$$

$$\bar{\lambda}_j = \frac{\sum_{S_k \in \Sigma_j} \lambda_k \Theta_k}{\bar{\Theta}_j}; \ \bar{u}_j = \frac{\sum_{S_k \in \Sigma_j} u_k \Theta_k}{\bar{\Theta}_j}; \quad (4)$$

общая трудоемкость изучения j-го модуля определится формулой

$$\bar{R}_j = \mathbf{e}^{\bar{\lambda}_j \bar{\theta}_j} (\bar{u}_j \bar{\theta}_j + \bar{l}_j \bar{t}_j) + \bar{z}_j, j = 1, ..., m, \quad (5)$$

а общая трудоемкость курса в соответствии с (4), (5) равна

$$R = \sum_{j=1}^{m} \bar{R}_j.$$

Этот показатель должен быть минимизирован путем выбора числа и состава модулей, т. е. должна быть решена задача нахождения величины

$$\hat{R} = \min_{m, \Sigma_1, ..., \Sigma_l} R. \quad (6)$$

Задачи, подобные описанной, характерны для систем структурного синтеза и оптимизации в системах автоматизированного проектирования (САПР). Для решения таких задач разработаны эффективные алгоритмы, которые могут быть использованы и в данном случае.

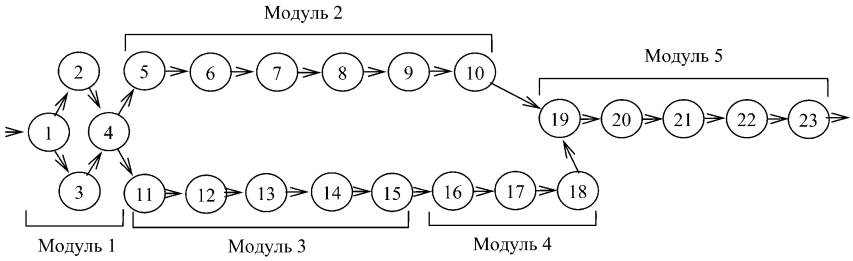
Таблица 1

Характеристики блоков курса "Основы теории управления"

N	Наименование блока (единицы контента)	θ_i	t_i	u_i	R_i
1	Введение. Принципы управления. Классификация систем управления	2,0	5	1	4,52
2	Уравнение динамики идеализированного двигателя	1,0	1	1	2,74
3	Макроэкономическая модель управления производством	2,0	3	1	4,29
4	Линеаризация моделей объекта. Безразмерная форма уравнений	1,0	1	1	2,74
5	Понятие управляемой динамической системы	0,5	3	1	2,36
6	Гладкие динамические системы. Линейные системы с дискретным и непрерывным временем	1,0	3	1	2,96
7	Переходные функции линейных динамических систем	1,0	2	1,2	2,85
8	Спектральное разложение фундаментальных матриц	0,5	2	1,1	2,25
9	Области достижимости управляемых динамических систем, оптимальное управление	2,5	5	1,3	5,27
10	Управляемость, наблюдаемость и устойчивость динамических систем по Ляпунову	1,5	2	1,3	3,15
11	Операторные методы анализа и синтеза систем управления. Преобразования Лапласа, Фурье, Z-преобразование	1,0	2	1	2,85
12	Понятие передаточных функций, их свойства, дискретные передаточные функции	1,0	3	1,1	2,96
13	Структурные схемы систем управления и их преобразования	1,0	1	1	2,74
14	Переходные процессы в непрерывных и дискретных динамических системах	1,0	2	1,2	2,85
15	Частотные характеристики динамических систем	1,0	5	1,1	3,17
16	Элементарные динамические звенья	2,0	4	1,1	4,40
17	Устойчивость замкнутых динамических систем. Критерий устойчивости Рауса—Гурвица	2,0	4	1,3	4,40
18	Частотные критерии устойчивости Михайлова и Найквиста	2,0	2	1,2	4,17
19	Уравнение движения волнового фронта	1,0	3	1,2	2,96
20	Уравнения движения локализационных сил и средств	0,5	1	1,2	2,15
21	Формулировка и решение задач локализационного управления	1,0	4	1,2	3,06
22	Понятие о синтезе систем управления, аналитическое конструирование регулятора	2,0	2	1,3	4,40
23	Заключение. Перспективы развития современных систем управления	2,0	4	1	4,40

Пример

В качестве простого примера рассмотрим синтез модулей интерактивного курса из набора тематических блоков на примере дисциплины "Основы теории управления". В соответствии с действующим Государственным образовательным стандартом направления 230100 эта дисциплина входит в блок общепрофессиональных дисциплин, ее объем составляет 120 ч. Рассматриваемая ниже реализация курса в системе БиГОР составлена на основе учебного пособия [4].



Логическая последовательность блоков курса "Основы теории управления"

Список блоков с их оценками приведен в табл. 1. Ряд параметров принят одинаковым для всех блоков: $\lambda_i = 0{,}075$; $z_i = 1{,}5$; $l_i = 0{,}1$, $i = 1, ..., 23$.

Логическая последовательность изучения блоков показана на рисунке. Стрелки на линиях показывают последовательность прохождения блоков. Параллельные ветви схемы свидетельствуют о том, что цепочки блоков могут изучаться в любом порядке.

Выбор оптимальной структуры и состава модулей осуществлялся по разработанному нами алгоритму на основе приведенной выше модели. Кроме того, полученные решения оценивались ведущим преподавателем, и среди близких по формальным показателям вариантов выбирался тот, который в наибольшей степени удовлетворяет его требованиям.

В качестве иллюстрации мы приведем три варианта разбивки курса на модули и оценки их трудоемкости:

1. Весь курс рассматривается как один модуль ($m = 1$). При этом общая трудоемкость составляет $R = 363$ ед.

2. Число модулей равно числу единиц контента в табл. 1, т. е. $m = 23$. В этом случае трудоемкость $R = 77{,}3$ ед.

3. Число модулей $m = 5$. Это оптимальный вариант, удовлетворяющий дополнительным экспертным требованиям. Его трудоемкость составляет $R = 67{,}5$ ед. В этом случае структура модулей определена табл. 2.

В первом варианте большое значение трудоемкости объясняется сложностью одновременного освоения всего объема курса и большой вероятностью неоднократного повторного прохождения курса. Во втором варианте изучение каждого модуля (отдельной единицы контента) не представляет сложностей, однако число контрольных мероприятий оказывается чрезмерным.

Таблица 2

Характеристики модулей

Номер модуля j	Блоки, вошедшие в модуль	Трудоемкость R_j
1	1, 2, 3, 4	12,94
2	5, 6, 7, 8, 9, 10	15,22
3	11, 12, 13, 14, 15	15,38
4	16, 17, 18	11,03
5	19, 20, 21, 22, 23	12,95
Общая трудоемкость R		67,52

Из таблицы видно, что полученный вариант помимо снижения общей трудоемкости курса обеспечивает также достаточно равномерное распределение трудоемкости среди модулей.

***

В заключение отметим, что предложенная методика может, на наш взгляд, явиться некоторым дополнением к возможностям системы БиГОР. Понятно, что при создании учебно-методических материалов нельзя чрезмерно увлекаться формальными методами, окончательное решение в любом случае принадлежит педагогу-создателю образовательного ресурса. Однако, как показывает наш опыт, разумное применение таких методов может значительно облегчить работу преподавателя, особенно в случаях, о которых говорилось в начале статьи.

Список литературы

1. **Норенков И. П., Уваров М. Ю.** База и генератор образовательных ресурсов // Информационные технологии. 2005. № 9. С. 60—65.

2. **Доррер Г. А., Рудакова Г. М.** Технология моделирования и разработки учебных электронных изданий / Отв. ред. В. С. Соколов. Новосибирск: Изд-во СО РАН, 2006. 272 с.

3. **Растригин Л. А., Эренштейн М. Х.** Адаптивное обучение с моделью обучаемого. Рига: Зинатне, 1988. 160 с.

4. **Доррер Г. А.** Основы теории управления: Учеб. пос. для студентов направлений 552800 и 654600. Красноярск: СибГТУ, 2003. 228 с.

УДК 004.332.3

О. Д. Жуков, канд. техн. наук,
Московский государственный университет
им. М. В. Ломоносова

Экспоненциальные вычисления в криптографии

Для применения криптографических методов, основанных на экспоненциальных вычислениях, всегда актуальна проблема ускорения этих вычислений. Предлагаются метод и алгоритм, обеспечивающие существенное повышение эффективности выполнения модулярных экспоненциальных вычислений. Метод основан на том, что вычет (остаток) по модулю P конструируется двумя вычетами по двум другим модулям P + 1 и P + 2.

Вычислительная сложность предлагаемого метода при его параллельном исполнении меньше, чем у обычного прямого метода на основе одного модуля. В частности, когда P + 1 и P + 2 суть произведения F факторов, вычислительная сложность стремится к 1/F. С помощью данного метода могут быть улучшены скоростные параметры не только в криптографии, но и в цифровой обработке сигналов.

Ключевые слова: *модулярные числовые системы, вычет, экспоненциальные вычисления, информационная безопасность.*

Введение

Существует много криптографических методов, которые базируются на экспоненциальных вычислениях вида $X^e \bmod P = |X^e|_P$ [1]. К ним относятся методы RSA, Rabin [2], Gamar [3], эллиптических кривых [4—6], стандартов цифровой подписи [7], Diffie-Hellman [8] и Okamoto [9].

Причина, из-за которой существует постоянная необходимость в ускорении экспоненциальных вычислений, состоит в том, что все методы оперируют числами длиной в сотни и тысячи битов. Это приводит к замедлению вычислительного процесса вследствие того, что двоичные числа такой длины не укладываются в машинное слово и для их обработки требуется использовать не отдельные машинные инструкции, а целые программные сегменты.

Отметим, что число модулярных умножений в процессе экспоненциальных вычислений может быть сокращено путем комбинирования операций умножения и возведения в квадрат. Существуют и другие подходы, способные ускорить вычисления и изложенные, например, в [10—13]. Предлагаемый метод отличается от них, но может быть скомбинирован с ними. Основное его отличие состоит в том, что вычет по модулю P конструируется из двух вычетов по модулям $P + 1$ и $P + 2$.

В методе RSA модуль P есть составное число, но его факторизация известна только для дешифратора. В методе El Gamar модуль есть простое число. В нашем случае, если модули $P + 1$ и $P + 2$ легко факторизуемы, то для них может быть применена Китайская теорема о вычетах (КТ).

Алгоритм вычисления вычета на основе КТ

При делении некоторого числа A на модуль P вычет a ($0 \leqslant a \leqslant P$) обозначается как

$$a \equiv |A|_P. \tag{1}$$

В соответствии с КТ число A в системе вычетов с модулями $m_1, m_2, \ldots, m_n$ определяется как

$$A \equiv \left| \sum_{i=1}^{n} a_i M_i q_i \right|_M, \tag{2}$$

где $a_i \equiv |A|_{m_i}$, $M = \prod_{i=1}^{n} m_i$, $M_i = M/m_i$, $|q_i M_i|_{m_i} = 1$.

Модулярная экспонента $|A^E|_P$ может быть вычислена в два этапа:

- возведение в степень: A^E;
- деление степени на модуль P и определение вычета $|A^E|_P$.

Однако такой прямой способ является трудоемким. Возможен другой подход, который сводится к делению на P и определению вычета результата на каждом шаге вычисления степени. При этом число шагов умножения может быть сокращено приблизительно до $2\log_2 E$ с помощью метода [10]. Представляемый метод и алгоритм заключаются в следующем.

Пусть

$$Y \equiv |X|_P, \tag{3}$$

где

$$0 \leqslant X \leqslant (P-1)^2. \tag{4}$$

Условие (4) справедливо, потому что X в нашем случае представляет квадрат исходного числа $A < P$ или произведение $X = AB$, где $A, B < P$. Введем обозначения:

$$Y_1 \equiv |X|_{P+1}; \tag{5}$$

$$Y_2 \equiv |X|_{P+2}. \tag{6}$$

Покажем, что на основе формул (4)—(6) имеем:

$$X = (P + 2)Y_1 - (P + 1)Y_2 + k(P + 1)(P + 2), \quad (7)$$

где $k = 0$ или $k = 1$.

Действительно, поскольку $|P + 2|_{P+1} \equiv 1$ и $|P + 1|_{P+2} = -1$, то $|X|_{P+1} \equiv Y_1$ и $|X|_{P+2} \equiv Y_2$. Таким образом, выражения (5) и (6) справедливы. Что касается значения k, то из неравенств $Y_1 \leqslant P$ и $Y_2 \geqslant 0$ следует, что правая часть в (7) меньше или равна $(P + 2)Y_1 - (P + 1)Y_2 - (P + 1)(P + 2) \leqslant (P + 2)P - (P + 1)(P + 2) = -(P + 2) < 0$. Но $X \geqslant 0$, и поэтому k не может быть отрицательным.

Затем допустим, что $k > 1$. Из неравенств $Y_1 \geqslant 0$ и $Y_2 \leqslant P + 1$ следует, что правая часть в (7) больше или равна $-(P + 1)^2 + 2(P + 1)(P + 2) \geqslant (P + 1)(P + 2) > (P - 1)^2$. Но $X \leqslant (P - 1)^2$, и поэтому k не может быть больше 1. Таким образом, $k = 0$ или $k = 1$.

Тогда с учетом формул (5)—(7) имеем:

при $Y_1 \geqslant Y_2$

$$Y \equiv |2Y_1 - Y_2|_P; \quad (8)$$

при $Y_1 < Y_2$

$$Y \equiv |2Y_1 - Y_2 + 2|_P. \quad (9)$$

Действительно, из (7) следует, что $X \equiv |2Y_1 - Y_2 + 2k|_P$ и (7) может быть переписано в следующем виде:

$$X = (P + 1)(Y_1 - Y_2) + Y_1 + k(P + 1)(P + 2).$$

Из выражений (5)—(7) вытекают следующие свойства. Если $Y_1 \geqslant Y_2$, справедливо выражение (8), поскольку $k = 0$. Если $Y_1 < Y_2$, справедливо выражение (9), так как $k = 1$.

Дополнительно отметим следующее обстоятельство. Для вычисления (8) и (9) не требуется выполнения деления. Это следует из того, что $-P < Y < 2P$. Таким образом, для получения корректного результата достаточно один раз вычесть или прибавить P.

Пример 1.

Пусть $P = 19$ и $X = x^2$.

- Если $x = 11$, $Y_1 \equiv |121|_{20} \equiv 1$ и $Y_2 \equiv |121|_{21} \equiv 16$.

В этом случае $Y_1 < Y_2$. На основе (9) $Y \equiv |2 - 16 + 2|_{19} \equiv 7$. Действительно, $7 \equiv |121|_{19}$.

- Если $X = 13$, $Y_1 \equiv |169|_{20} \equiv 9$ и $Y_2 \equiv |169|_{21} \equiv 1$.

В этом случае $Y_1 > Y_2$. На основе (8) имеем $Y \equiv |18 - 1|_{19} \equiv 17$. Действительно, $17 \equiv |169|_{19}$.

Пример 2.

Этот пример взят из источника [1] и относится к криптографическим системам типа RSA.

Пусть $P = 2773$ и $X = x^2$. Тогда $P + 1 = 2774 = 2 \cdot 19 \cdot 73$ и $P + 2 = 2775 = 3 \cdot 5^2 \cdot 37$.

Если $x = 920$, $Y_1 \equiv |920^2|_{2774} \equiv 330$ и $Y_2 \equiv |920^2|_{2775} \equiv 25$.

В этом случае $Y_1 > Y_2$. На основе (8) $Y \equiv |2 \cdot 330 - 25|_{2773} \equiv 635$. Действительно, $635 \equiv |920^2|_{2773}$.

В приведенных примерах присутствуют вычисления по двум модулям $P + 1$ и $P + 2$ вместо вычислений по модулю P. Однако, как будет показано в следующем разделе, демонстрируемый метод имеет определенное преимущество.

Алгоритм модулярного возведения в квадрат и умножения

Вычисление экспоненты может быть выполнено на основе комбинации вычислений квадратов и произведений вида

$$|x^2|_P; \ |xu|_P. \quad (10)$$

Для вычисления этих выражений предлагается следующий алгоритм на основе КТ и полученных значений Y_1 и Y_2. Предварительно составные модули $P + 1$ и $P + 2$ декомпозируются на взаимно простые факторы:

$$P + 1 = \prod_{i=1}^{h} p_i \text{ и } P + 2 = \prod_{i=1}^{g} q_i.$$

Тогда следующий алгоритм возведения в квадрат определяет значение x такое, что $0 \leqslant x \leqslant P - 1$ и $Y \equiv |x^2|_{p_i}$; $i = 1, \ldots, h$.

Алгоритм

Ввод: x; $0 \leqslant x \leqslant P - 1$; p_i; $i = 1, \ldots, h$.

Вывод: $Y \equiv |x^2|_{p_i}$; $i = 1, \ldots, h$.

1. Вычислить $x_i \equiv |x|_{p_i}$; $i = 1, \ldots, h$.
2. Вычислить $a_i = x_i^2$; $i = 1, \ldots, h$.
3. Вычислить $a_i \equiv |a_i|_{p_i}$; $i = 1, \ldots, h$.
4. Вычислить Y.

В п. 4 алгоритма определяются значения $Y_1 \equiv |x^2|_{P+1}$ и $Y_2 \equiv |x^2|_{P+2}$. Соответствующие вычеты произведения xu определяются подобным образом. Это предполагает выполнение $u_i = |u|_{p_i}$ на шаге 1 и использование $a_i = x_i u_i$ вместо $a_i = x_i^2$ на шаге 2.

Пример 3.

Для этого примера используются исходные данные примера 2: $P = 2773$.

Предварительно декомпозируем модули $P + 1$ и $P + 2$:

$$P + 1 = 2774 = 2 \cdot 19 \cdot 73$$

$$\text{и } P + 2 = 2775 = 3 \cdot 5^2 \cdot 37.$$

Пусть модули p_i и q_j суть $p_1 = 2 \cdot 19 = 38$, $p_2 = 73$, $q_1 = 3 \cdot 5^2 = 75$, $q_2 = 37$. И пусть $x = 920$. Требуется определить квадратичную экспоненту по модулю P: $Y \equiv |x^2|_P$.

Процедура вычисления Y_1 для этой экспоненты приведена ниже:

1. $x_1 \equiv |920|_{38} \equiv 8$; $x^2 \equiv |920|_{73} \equiv 44$.
2. $a_1 = 8^2 = 64$; $a_2 = 44^2 = 1936$.
3. $a_1 \equiv |64|_{38} \equiv 26$; $a_2 \equiv |1936|_{73} \equiv 38$.
4. На основе решения системы конгруэнций

$$Y_1 \equiv |26|_{38} \text{ и } Y_2 \equiv |38|_{73}$$

получим $Y_1 = 330$. С помощью этой же процедуры вычисляется $Y_2 = 25$. С учетом (8) получим $Y = 2 \cdot 330 - 25 = 635$. Данное значение Y подтверждается результатом примера 2.

Заключение

Если модуль P есть простое число или составное число, для которого факторизация является чрезвычайно трудоемкой, эффективность процесса вычисления вычета экспоненты по модулю P может быть существенно повышена с помощью КТ. Это достигается за счет использования системы вычетов с модулями $P + 1$ и $P + 2$, для которых факторизация является достаточно простой. Эффективность представленного метода тем выше, чем меньше значения факторов, составляющих эти модули. Сами факторы представляют собой взаимно простые модули, относительно которых выполняются необходимые операции умножения и сложения. Поэтому скорость вычислительного процесса будет выше по сравнению с прямым методом определения вычета экспоненты по модулю P, имеющему эквивалентную длину до тысячи и более битов.

Список литературы

1. **Rivest R., Shamir A., Adleman L.** A method for obtaining digital signatures and public key cryptosystems // Commun. ACM. 1978. **21.** P. 120—126.
2. **Rabin M. O.** Digital signatures and public-key functions as intractable as factirization. MIT/LCS/TR-212 Tech. memo. Massachusetts institute of technology. 1979.
3. **El Gamal T. A.** A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE Trans. inf. theory. 1985. N 31. P. 469—472.
4. **Koyama K., Maurer U., Okamoto T., Vanstone S. A**. New public-key schemes based on elliptic curves over the ring Z_n // Proc. CRYPTO'91, Santa Barbara (USA). 1991. P. 252—266.
5. **Menezes A. J., Vanstone S. A.** Elliptic curve cryptosystems and their implementation // J. Cryptol. 1993. N 6. P. 209—224.
6. **Kuwakado K., Koyama K.** Efficient cryptosystems based on a form-free primes // IEICE Trans. Fundam. 1994. E77-A. P. 1309—1318.
7. **NIST.** Specifications for a digital signature standard. Federal inf. proc. standard publication 186. 1991.
8. **Diffie W., Hellman M.** New directions in cryptography // IEEE trans. inf. theory. 1976. N 22. P. 644—654.
9. **Okamoto T.** Provably secure and practical identification schemes and corresponding signature schemes // Proc. CRYPTO'92, Santa Barbara. 1992. P. 31—53.
10. **Knuth D. E.** The art of computer programming. 2nd ed., Addison Wesley; 1980. Vol. 2, chapter 4.
11. **Quisquater J. J., Cuovreur C.** Fast decipherment algorithm for RSA public-key cryptosystem // Electron. lett. 1982. N 18. P. 905—907.
12. **Ito T., Sako K.** Algorithm on finite field and high-speed algorithm for multiple length remainder operation // Inf. proc. 1993. N 34. P. 170—179.
13. **Morita H.** Cryptograthy and hign-speed algorithm // Inf. proc. 1993. N 34. P. 336—342.
14. **Koblitz N. A.** A course in number theory and cryptograthy. Springer. 1987. Chapter 1.

Поздравляем юбиляра!

Заслуженному деятелю науки и техники РФ, заведующему кафедрой "Системы автоматизации проектирования" МГТУ им. Н. Э. Баумана, академику РАЕН, лауреату Государственной премии СССР, главному редактору журнала "Информационные технологии", доктору технических наук, профессору

Игорю Петровичу НОРЕНКОВУ,

известному ученому в области современных компьютерных и информационных технологий исполнилось *75* лет.

Игорь Петрович является одним из пионеров автоматизации проектирования в СССР, создателем первой отечественной программы анализа электронных схем, руководителем одной из ведущих отечественных школ в области моделирования и оптимизации технических систем, синтеза проектных решений на основе технологий эволюционных вычислений, а также разработки электронных образовательных ресурсов. Он является автором более 230 печатных работ, включая 18 монографий, учебников и учебных пособий.

Возглавляемая И. П. Норенковым кафедра занимает ведущее место в системе подготовки высококвалифицированных специалистов в области автоматизации проектирования. Под его руководством выпущена широко известная серия учебных пособий "Информационные технологии в техническом университете", а также создана оригинальная автоматизированная обучающая система БиГОР (База и Генератор Образовательных Ресурсов), успешно используемая в учебном процессе. Игорь Петрович ведет активную плодотворную работу по подготовке научно-педагогических кадров высшей квалификации.

И. П. Норенков является создателем и главным редактором нашего любимого журнала "Информационные технологии", который за десять лет существования стал всероссийской трибуной для специалистов в области современных информационных технологий.

Талантливый ученый, блестящий педагог, высококвалифицированный специалист, чуткий, отзывчивый и обаятельный человек, Игорь Петрович пользуется глубоким уважением учеников, коллег, товарищей.

Сердечно поздравляем юбиляра, желаем ему крепкого здоровья, благополучия и новых творческих успехов в научной и педагогической деятельности.

Редколлегия и редакция журнала.

CONTENTS

This article deals with the software developed by the authors which ensures field intensity calculation in limited and unlimited calculation region of magnetic systems confining ferromagnetic parts of complex geometric form taking into consideration nonlinear magnetic properties.
Keywords: software, magnetic system, ferromagnetic, nonlinear magnetic property, volume integral equation, adaptive discretization, numerical integration, system of linear and nonlinear equations, Newton's method.

Possibilities of object choice from the time series on base of its signs by means of logical neural network are examined. This apparatus allow making parallel analyze all signs with etalon. Simultaneous comparison with few etalons allows constructing the decision make system.
Keywords: time series object, set of signs, recognition, logical network, decision make system.

In this article the algebraic system containing semantics of the expanded production logic is considered. Operations and relations of this system correspond to the full set of logical connectives of a propositional calculus. In this system the infinite operations realizing universal and existential quantifiers for the production logic model are defined. Theorems on logical closure structure and on existence of a logical reduction of the considered algebraic system are proved. The effective way of a logical reduction construction is specified.
The received results can be applied to research and automatic optimization of knowledge bases of production type, such as knowledge bases of systems of symbolical mathematics.
Keywords: production logic, algebraic system, complete lattice, logical closure, logical reduction.

This article describes the problem of detecting logic bombs in compilers and an offered method of testing compilers without source codes on logic bombs. This method is based on methods of black-box testing and game theory. The suitable model of logic bomb has been developed. The recommendations of practice for using developed method are stated.
Keywords: testing, compiler, logic bomb, black box, game theory, certification.

In this article we concentrate our attention on problem of adequate knowledge estimation. As the measure of adequateness we take the student's knowledge estimation been made by a teacher after subject discussion. The main questions this article tries to answer to are those as follows: Whether it is possible to use computer testing despite the subject discussion with the teacher? Is there any model of software for testing that makes it an analogue of subject discussion with the teacher in a field of knowledge estimation?
Keywords: intellectual testing, automatic knowledge estimation, meaning based relation of questions, interpolation, approximation, received information.

When the module-rating educational technology is using, the problem of rational synthesis of learning modules became an important one. The good opportunities for solving of this problem, especially when the educational programs are change often give the using of BiGOR system which was developed in Moscow Bauman University. In the work presented the method of optimal combination the learning course components (the divided units of content) to some number of equivalent modules is suggested. This method is illustrated on example of the course "The Basis of Control Theory".
Keywords: learning modules, divided units of content, BiGOR system, module-rating educational technology, optimization of learning course structure.

The method proposed for cryptography is based on an idea that the remainder for moduls SPS is constructed from the remainders with moduli SP+2S and SP+2S. The efficiency of the method depends strongly on whether or not SP+1S and SP+2S can be decomposed into smaller prime factors
Keywords: modular number system, remainder, exponentian computing information secuirity.

Адрес редакции:

107076, Москва, Стромынский пер., 4/1

Телефон редакции журнала **(495) 269-5510**
E-mail: it@novtex.ru

Дизайнер *Т.Н. Погорелова*. Художник *В.Н. Погорелов*.
Технический редактор *О. А. Ефремова*. Корректор *Т. В. Арбузова*

Сдано в набор 09.06.2008. Подписано в печать 18.07.2008. Формат 60×88 1/8. Бумага офсетная. Печать офсетная.
Усл. печ. л. 9,8. Уч.-изд. л. 10,86. Заказ 854. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати,
телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Отпечатано в ООО "Подольская Периодика"
142110, Московская обл., г. Подольск, ул. Кирова, 15