ISSN 1684–8853

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

45 лет – полету Юрия Гагарина!

1961–2006

2(21)/2006

НАУЧНО-ПРАКТИЧЕСКИЙ ЖУРНАЛ

2(21)/2006

РЕЦЕНЗИРУЕМОЕ ИЗДАНИЕ

ИНФОРМАЦИОННО-УПРАВЛЯЮЩИЕ СИСТЕМЫ

Адрес редакции: 190000, Санкт-Петербург,
Б. Морская ул., д. 67
Тел.: (812) 710-66-42, (812) 313-70-88
Факс: (812) 313-70-18
E-mail: ius@aanet.ru
Сайт: www.i-us.ru

Журнал зарегистрирован
в Министерстве РФ по делам печати,
телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-12412 от 19 апреля 2002 г.

Журнал распространяется по подписке.
Подписку можно оформить через редакцию, а также
в любом отделении связи по каталогам:
«Пресса России» – № 42476;
«Роспечать» («Газеты и журналы») – № 15385

ЛР № 010292 от 18.08.98.
Сдано в набор 24.03.2006. Подписано в печать 21.04.2006. Формат 60×90 1/8.
Бумага офсетная. Гарнитура SchoolBookC. Печать офсетная.
Усл. печ. л. 8,0. Уч.-изд. л. 9,0. Тираж 1000 экз. Заказ 189.

Оригинал-макет изготовлен
в отделе электронных публикаций и библиографии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

Отпечатано с готовых диапозитивов
в отделе оперативной полиграфии ГУАП.
190000, Санкт-Петербург, Б. Морская ул., 67.

УДК 681.516.7.015

ОПТИМАЛЬНОЕ ПРАВИЛО ОСТАНОВКИ НАБЛЮДЕНИЙ – СПОСОБ ДОСТИЖЕНИЯ НАИВЫСШЕЙ ВЕРОЯТНОСТИ ОБНАРУЖЕНИЯ СИГНАЛОВ

А. К. Розов,
доктор техн. наук
А. Н. Сырцев,
канд. техн. наук, преподаватель
Н. В. Кузина,
инженер-программист
Военно-морская академия им. Н. Г. Кузнецова

Излагается процедура применения оптимального правила остановки наблюдений для достижения наивысшей вероятности обнаружения слабых сигналов.

We discuss an optimal rule of stopping the observations to reach the maximal probability of finding weak signals.

Введение

В работе [1] было показано, как аппарат стохастических дифференциальных уравнений может быть использован для составления алгоритмов систем самонаведения. Отмечалось, что его использование позволяет применять оптимальное правило остановки наблюдений для максимизации вероятности обнаружения сигнала и тем самым увеличить дальность действия систем в 1,5 раза и более.

Вместе с тем для систем, использующих короткие зондирующие сигналы (порядка нескольких микросекунд), решение стохастических дифференциальных уравнений в реальном масштабе времени встречает трудности технического порядка, поскольку современные вычислительные средства еще не в состоянии обеспечить нужное быстродействие. Чтобы в таких условиях сохранить изложенные в статье [1] подходы, имеет смысл перейти к дискретному времени наблюдения.

Дискретное построение алгоритма позволяет также более наглядно интерпретировать основные положения теории оптимальных правил остановки, что немаловажно. Являясь основой для достижения наибольшей вероятности обнаружения, оптимальное правило остановки заслуживает того, чтобы подробнее остановиться на процедуре его использования.

Такому, своего рода, дополнению к статье [2] и посвящено дальнейшее изложение.

Оптимальное правило остановки – способ достижения наибольшего выигрыша

Теория оптимальных правил остановки наблюдений возникла как развитие статистического последовательного анализа А. Вальда.

В статистике случайных процессов значительный круг задач можно сформулировать в рамках следующей схемы. Задан частично наблюдаемый марковский процесс (Θ,η), $t\in T$, где Θ_t – его ненаблюдаемая, а η_t – наблюдаемая компонента. Требуется определить, как, последовательно наблюдая процесс η_t, наилучшим образом различить те или иные статистические гипотезы относительно значений Θ_t, оценить значения неизвестных параметров Θ_t.

Решением этих задач занимается статистический последовательный анализ, который можно определить как такой способ обработки текущих данных, в котором решения о значениях ненаблюдаемой компоненты выносятся не в фиксированный заранее момент времени, а по ходу наблюдений, с их прекращением в случайный момент времени ν.

Наиболее полно в статистическом последовательном анализе исследованы те случаи, когда время наблюдения дискретно, $T\{1, 2, ...\}$, ненаблюдаемый параметр λ_t не изменяется во времени, $\lambda_t = \lambda$. Именно к этому случаю относятся вальдовские задачи различения двух или нескольких гипотез о значениях неизвестного параметра λ.

Затем в работах ряда авторов подход А. Вальда был распространен для случаев, когда параметр сам является случайным процессом, например процессом, представляющим факт появления сигнала в случайный момент времени.

И если в задачах различения гипотез необходимость обращения к последовательным методам была не столь уж очевидной, то в приводимых дальше задачах обнаружения последовательный характер производимых наблюдений и проблема отыскания момента остановки обусловлены самим существом этих задач.

Эксцессивность выигрыша. В изменившихся условиях обстановка может быть представлена частично наблюдаемой последовательностью (Θ, η), в которой наблюдается лишь компонента η. Скрытый параметр Θ изменяется во времени случайным образом. Он характеризует момент изменения вероятностных характеристик наблюдаемого процесса (изменение плотности f_0 на f_1).

Все, что нам известно относительно скрытых данных о $\Theta_1^n = (\Theta_1, ..., \Theta_n)$, заключено в распределении вероятностей $\pi_n = \pi(\Theta_1, ..., \Theta_n)$, подсчитываемых в предположении известных $\eta_1^n = (\eta_1, ..., \eta_n)$,т. е. как вероятность $P(\Theta_1^n \eta_1^n)$. Поэтому вся информация о частично наблюдаемой последовательности (Θ, η) полностью описывается парой $\sigma_n = (\pi_n, \eta_1^n)$.

При каждом n заданы функции $g\left(\Theta_1^n, \eta_1^n\right)$, интерпретируемые как выигрыши, полученные в состоянии $\left(\Theta_1^n, \eta_1^n\right)$, если наблюдения прекращаются в момент времени n. Поскольку Θ_1^n для наблюдателя неизвестно, он знает не истинный выигрыш $g\left(\Theta_1^n, \eta_1^n\right)$, а лишь его среднее значение

$$g(\sigma_n) = \sum_{\Theta_1^n} g\left(\Theta_1^n, \eta_1^n\right)\pi_n.$$

Оптимальное правило остановки наблюдений предполагает нахождение такой процедуры, при которой достигался максимум функционала

$$U_1^N(\sigma_1) = \sup_{\nu \in K_n} T^\nu g(\sigma_1) = \sup_{\nu \in K_N} M\left[g(\sigma_\nu)|\sigma_1\right],$$

когда начальное состояние есть σ_1.

Для ее определения вводится выигрыш

$$U_n^N(\sigma_n) = \sup_{\nu \in K_{N-n}} T^\nu g(\sigma_n),$$

который можно трактовать как наибольший выигрыш, который мы получим, если наблюдения заведомо совершаются до момента времени n и могут прекращаться в любой из моментов $n, n+1, ..., N$.

Для нахождения оптимального правила остановки надо определить динамику изменения выигрыша $U_n^N\left(\sigma_1^n\right)$, $n = 1, ..., N$. Нахождение последнего возможно благодаря тому, что он может рассматриваться как эксцессивная функция, т. е. функция, удовлетворяющая условию

$$f(x) \geq Tf(x),$$

$$Tf(x) = M_x f(x),$$

более того – как наименьшая эксцессивная мажоранта функции $g(x)$, т. е. наименьшая из функций $f(x)$, удовлетворяющая условию

$$f(x) \geq g(x).$$

Иногда вместо эксцессивности используется возможность представления выигрыша процессом типа супермартингал.

Оказывается, и в этом смысл введения эксцессивности функции $f(x)$, что сам минимизируемый функционал $U_1^N(\sigma_1)$ является наименьшей эксцессивной мажорантой функции $g\left(\sigma_\nu|\sigma_1\right)$.

С точки зрения традиционных исследований в последовательном анализе, динамическом программировании, теории игр, «эксцессивная» характеризация функционала выигрыша важна потому, что из нее сразу следует рекуррентное уравнение

$$U_n^N\left(\sigma_1^n\right) = \max\left\{g\left(\sigma_1^n\right), TU_{n+1}^N\left(\sigma_1^n\right)\right\}. \qquad (1)$$

Решение рекуррентного уравнения для $U_n^N(\sigma_n)$ возможно методом индукции назад – методом Беллмана. В соответствии с этим методом для момента $n = N$ получаемый выигрыш приравнивается к значению $g(\sigma_N)$. В момент времени $N-1$ мы можем или прекратить наблюдения и получить тогда выигрыш $g(\sigma_{N-1})$, или же сделать еще одно наблюдение и тогда получить выигрыш, в среднем равный $M\left[g(\sigma_N)|\sigma_{N-1}\right]$. Аналогичным образом надо поступать на интервале $[N-2, N]$. Здесь выигрыш $g(\sigma_{N-2})$ сравнивается с суммарным выигрышем от остановки в моменты $N-1$ и N. В результате могут быть заранее определены значения $U_n^N\left(\sigma_1^n\right)$, с которыми последовательно должны сравниваться выигрыши $g(\sigma_n)$.

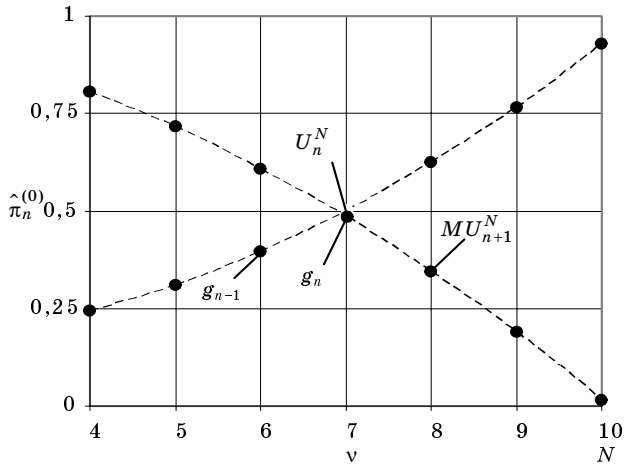
Здесь наиболее отчетливо видно, что выигрыш от продолжения наблюдений должен подсчитываться не только с учетом эволюции значений η_1^n на $n+1, ..., N$, но также с учетом ожидаемых решений после момента n – на $[n+1, N]$.

Принятие решения о прекращении наблюдений происходит в момент ν, когда $g(\sigma_n)$ сравняется с $U_n^N(\sigma_n)$ (рис. 1):

$$\nu = \min\left\{n : g(\sigma_n) = U_n^N(\sigma_n)\right\}. \qquad (2)$$

Предыдущее изложение относилось к варианту дискретного времени. В случае непрерывного времени функционал выигрыша $U_t^T(\sigma_n)$ (потерь) также допускает эксцессивную или регулярную характеризацию. Однако установление этих фактов и определение процедуры нахождения оптимальных моментов остановки требует привлечения довольно тонких результатов из теории марковских процессов и теории мартингалов.

Подобным образом можно поступать и в случае функционала, минимизирующего потери. Это за-



■ ***Рис. 1.*** *Выигрыши и их математические ожидания*

дачи классификации и оценивания сигнала, когда потери состоят из среднего времени наблюдения и вероятности ошибочных решений.

Обнаружение сигнала, появляющегося в случайный момент времени

Будем предполагать, что воздействие сигнала задается процессом

$$\chi_n = \begin{cases} 0, & n < \Theta \\ \Theta_n, & n = \Theta, \\ 0, & n > \Theta \end{cases}$$

где Θ – момент воздействия сигнала; Θ_n – сигнал известной или случайной величины.

Наблюдаемый процесс представляется в виде

$$\eta_t = \chi_n + x_n,$$

где x_n – помеха, имеющая нормальное распределение $x_n \sim N(0, 1)$.

В частном случае такая модель может точно соответствовать реальной обстановке. Имеется в виду случай, когда сигнал может появиться в одном из подынтервалов $[0, h]$, $[0, 2h]$, ... Тогда для каждого подынтервала могут быть образованы интегралы

$$\Theta_n = \int_{t_{n-1}}^{t_n} \Theta_s ds, \quad x_n = \int_{t_{n-1}}^{t_n} dw_s,$$

соответствующие воздействию сигнала и помех (w_s – винеровский процесс). Тем самым случайные величины χ_n и η_n приобретают вполне реальный смысл и без каких-либо приближений представляют реальную обстановку с квантованным во времени сигналом и помехой.

Необходимо в результате последовательного наблюдения за η_i, $i = 1, ..., n$, обнаружить появление сигнала, т. е. подать тревогу, когда он появится. В этом случае будет своевременное обнаружение.

Это случай, когда момент остановки должен совпадать с моментом появления сигнала, т. е. когда выигрыш $g(\sigma_n)$ представляется как

$$g(\sigma_n) = W(\nu, \Theta) P\left(\Theta \middle| \eta_1^n\right),$$

где функция выигрыша $W(\nu, \Theta)$ имеет вид

$$W(\nu, \Theta) = \begin{cases} 0, & \nu < \Theta \\ 1, & \nu \in [\Theta, \Theta + m]. \\ 0, & \nu > \Theta \end{cases}$$

Вид выигрыша (наличие апостериорной вероятности $P\left(\Theta \middle| \eta_1^n\right)$) определяет байесовский характер процедуры обнаружения – использование апостериорных вероятностей в качестве «рабочих» статистик. Это означает, что, согласно выражению (2), такая (или такие) статистика должна сравниваться с границей областей принятия решений и по достижении ею границ наблюдения должны прекращаться и приниматься решения об обнаружении сигнала.

Желательно, чтобы рабочие статистики были достаточными, но определение их в таком виде осложняется немарковостью процесса χ_n, представляющего воздействие сигнала. Интуитивно понятно, что одной из основных статистик должна быть апостериорная вероятность $\pi_n^{(0)} = P\left(\Theta \middle| \eta_1^n\right)$. В информационном смысле желательно большее их число, например $\pi_n^{(0)}$ и $\pi_n = P\left(\Theta \le n \middle| \eta_1^n\right)$.

Остановимся на характеристике использования этих двух вариантов.

Байесовское правило. Использование одной статистики $\pi_n^{(0)}$, когда $m = 0$, позволяет определить границу областей принятия решений. Последняя определяется методом индукции назад, основанном на рекуррентном соотношении (1), в котором $MU_{n+1}^N\left(\sigma_1^n\right)$ рассматривается как максимальный выигрыш от продолжения наблюдений. Максимальный, поскольку в его вычислении участвуют ранее определенные границы $\hat{\pi}_N^{(0)}$, ..., $\hat{\pi}_{n+1}^{(0)}$. Этот выигрыш приравнивается к $U_n^N(\sigma_n)$, который в соответствии с формулой (2) выступает как граница $\hat{\pi}_n^{(0)}$.

Итак, нахождение границы $\hat{\pi}_n^{(0)}$ сводится к ее приравниванию к математическому ожиданию выигрыша от обнаружения сигнала на интервале $[n + 1, N]$. Последний вычисляется как взвешенная сумма отношений числа своевременных обнаружений к числу разыгрываемых реализаций $\pi_n^{(0)}$, $n = 1, ..., N$, с весами, определяемыми априорными вероятностями появления сигнала в соответствующие моменты времени.

Видим, что процедура хотя и громоздка, но проста в своей основе. Она становится совсем простой

в случае довольно сильных сигналов, когда «апостериорный» характер подсчета числа своевременных обнаружений на интервале $[n+1, N]$ может быть заменен суммированием расположенных справа вероятностей $P(n+1), \ldots, P(n)$.

Полученная таким образом граница является оптимальной, т. е. гарантирующей получение наивысшей вероятности своевременного обнаружения сигнала с использованием статистики $\pi_n^{(0)}$, $n = 1, \ldots, N$.

Если допустить возможность небольшого (на m шагов) запаздывания в обнаружении сигнала и использовать для этого апостериорную вероятность $\pi_n^{(m)} = P\left(\Theta \in [n-m,\ n] \middle| \eta_1^n\right)$, то вероятность обнаружения сигнала может быть увеличена. При нахождении границ $\bar{\pi}_n^{(m)}$ в этом случае вместо числа своевременных обнаружений участвует число своевременных и запоздавших обнаружений.

Так, определенные границы будут зависеть как от величины сигнала, так и от априорного распределителя момента его появления.

Модифицированное байесовское правило. Как уже отмечалось, статистика может состоять из нескольких апостериорных вероятностей, в частности из $\pi_n^{(-)}$ и $\pi_n^{(0)}$, что обещает получение большей вероятности своевременного обнаружения сигнала. Но здесь возникают сложности в нахождении границы.

Приходится отказаться от учета границ на интервале $[n+1, N]$ и заменить эволюцию математического ожидания $MU_{n+1}^N(\sigma_n)$ на апостериорную вероятность появления сигнала в будущем – $\pi_m^{(+)} = P\left(\Theta > n \middle| \eta_1^n\right)$. При этом, конечно, теряется информация о характере обнаружения сигнала за моментом n.

Тогда граница в момент n могла бы быть равной $\bar{\pi}_n^{(0)} = \pi_n^{(+)}$. Это неудобно, поскольку в этом случае она бы зависела от реализации η_1^n. Такой зависимости можно избежать, если воспользоваться соотношением

$$\pi_n^{(+)} = 1 - \pi_n^{(-)} - \pi_n^{(0)}$$

и ввести новую статистику

$$\Omega_n = \pi_n^{(-)} + 2\pi_n^{(0)}$$

с границей $\widehat{\Omega}_n = \widehat{\Omega} = 1$.

Моделирование процедуры обнаружения показало, что в действительности $\widehat{\Omega}_n$ несколько меньше единицы (в пределах от 0,95 до 1,0) и практически не критична к величине сигнала и априорному распределению момента его появления.

Пример. Воздействие сигнала величины r задается процессом

$$\chi_n = \begin{cases} 0, & n < \Theta \\ r, & n = \Theta, \\ 0, & n > \Theta \end{cases}$$

где Θ – момент воздействия сигнала.

Наблюдается величина

$$\eta_t = \chi_n + \sqrt{c_2}\, x_n, \quad x_n \sim N(0, 1).$$

Решение о поступлении сигнала принимается по достижению статистиками $\pi_n^{(0)}$ и $\Omega_n = \pi_n^{(0)} + 2\pi_n^{(0)}$ «своих» границ.

Рекуррентные соотношения[1]:

$$\pi_{n+1}^{(-)} = \frac{P(\Theta > n)\left(\pi_n^{(-)} + \pi_n^{(0)}\right)}{[*]}, \quad \pi_{n+1}^{(0)} = \frac{P_{n+1}\pi_n^{(+)}\varphi_{n+1}}{[*]},$$

$$\pi_{n+1}^{(+)} = \frac{P(\Theta > n+1)\pi_n^{(+)}}{[*]},$$

$$[*] = P_{n+1}\pi_n^{(+)}(\varphi_{n+1} - 1) + P(\Theta > n).$$

Начальные условия:

$$\pi_1^{(-)} = 0, \ \pi_1^{(0)} = \frac{P_1\varphi_1}{p_1(\varphi_1 - 1) + 1}, \ \pi_1^{(+)} = \frac{P(\Theta > 1)}{P_1(\varphi_1 - 1) + 1}.$$

Формулы для φ_n и ψ_n:

$$\varphi_n = e^{\frac{1}{c_2} r\, \eta_n - \frac{1}{2c_2} r^2},$$

$$\psi_n = \frac{1}{c_2} r\, \eta_n - \frac{1}{2c_2} r^2.$$

Значения P_n (табл. 1); $P(\Theta > n)$, $P(\Theta > n+1)$:

$$P(\Theta > n) = P_{n+1} + P_{n+2} + \ldots + P_{10};$$

$$P(\Theta > n+1) = P_{n+2} + P_{n+3} + \ldots + P_{10}.$$

Байесовский вариант. Значения границы, рассчитанные методом индукции назад, – табл. 2.

Моделирование проводилось при $r = 1$, $c_2 = 1$ с, $N = 10$. Вероятность ложной тревоги оказалась равной $P_{\text{л.т}} = 0{,}192$, а вероятность своевременного обнаружения $P_{\text{св.обн}} = 0{,}613$.

Если допустить запаздывание в обнаружении сигнала на величину $m = 1$ и использовать для этой цели апостериорную вероятность $\pi_n^{(m)}$, определяемую соотношением

$$\pi_n^{(1)} = (1 - \pi_n)\left[\varphi_{n-1}\frac{P_{n-1}}{P(\Theta > n-1)} + \varphi_n \frac{P_n}{P(\Theta > n)}\right],$$

то граница $\bar{\pi}_n^{(1)}$, определенная методом индукции назад, – табл. 3, а вероятность ложной тревоги $P_{\text{л.т}} = 0{,}2002$, при этом вероятность обнаружения сигнала с возможным запаздыванием увеличится до $P_{\text{обн}} = 0{,}762$.

Модифицированный байесовский вариант. В этом случае в качестве статистики принималась величина $\Omega_n = \pi_n^{(-)} + 2\pi_n^{(0)}$. Результаты моделирования – зависимость $P_{\text{св.обн}}$ от значений постоян-

[1] См. приложение.

■ *Таблица 1*

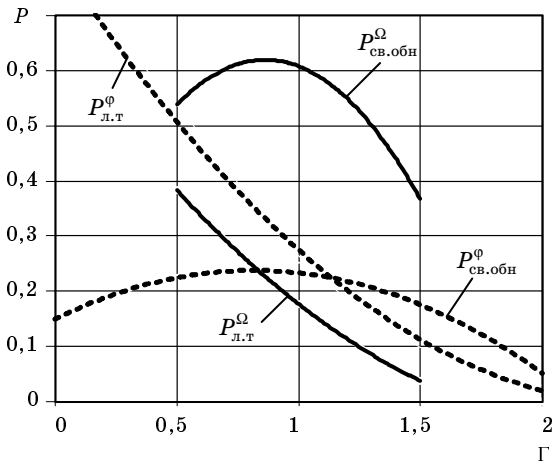
n	1	2	3	4	5	6	7	8	9	10
P_n	0,001	0,004	0,01	0,035	0,45	0,45	0,035	0,01	0,004	0,001

■ *Таблица 2*

n	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_n$	0,615	0,615	0,615	0,615	0,450	0,035	0,010	0,004	0,001	0

■ *Таблица 3*

n	1	2	3	4	5	6	7	8	9	10
$\hat{\pi}_n^{(1)}$	0,764	0,764	0,761	0,746	0,450	0,035	0,010	0,004	0,001	0



■ ***Рис. 2.*** *Вероятности обнаружения сигнала*

ных границ при использовании статистик Ω_n и $\psi_n = \ln\varphi_n$ представлена на рис. 2.

Видим, что предложенный метод обнаружения по сравнению с традиционным, использующим в качестве статистики отношение правдоподобия, увеличивает вероятность обнаружения сигнала с $P^{\psi}_{\text{св.обн}} = 0{,}23$ до $P^{\Omega}_{\text{св.обн}} = 0{,}62$ в случае модифицированного байесовского правила и до $P^{(1)}_{\text{обн}} = 0{,}76$ в случае байесовского правила. И хотя приведенное сопоставление относится к примеру с малым числом наблюдений (N), к сигналу постоянной величины r и т. д., оно все же значимо, поскольку относится к случаю, когда имеет место неопределенность в моменте появления сигнала – главное, чем отличается рассмотренная задача.

Полученные результаты свидетельствуют о практической значимости оптимальных правил остановки наблюдений для получения наибольшей вероятности обнаружения слабых сигналов и увеличения в связи с этим дальности действия систем обнаружения.

Приложение

Рекуррентные соотношения. Исходным моментом в составлении рекуррентных соотношений является формула Байеса. В соответствии с этой формулой статистика π_{n+1} может быть представлена в виде

$$\pi_{n+1} = P\left(\Theta \le n+1 \middle| \eta_1^{n+1}\right) =$$

$$= P\left(\Theta \le n \middle| \eta_1^{n+1}\right) + P\left(n+1 \middle| \eta_1^{n+1}\right) =$$

$$= P\left(\Theta \le n \middle| \eta_1^{n}, \eta_{n+1}\right) + P\left(n+1 \middle| \eta_1^{n}, \eta_{n+1}\right) =$$

$$= \frac{1}{f\left(\eta_{n+1}\middle|\eta_1^n\right)}\left[P\left(\Theta \le n \middle| \eta_1^{n}, \eta_{n+1}\right) f\left(\eta_{n+1}\middle|\eta_1^n\right) +\right.$$

$$\left. + P\left(n+1 \middle| \eta_1^{n}, \eta_{n+1}\right) f\left(\eta_{n+1}\middle|\eta_1^n\right)\right] =$$

$$= \frac{1}{f\left(\eta_{n+1}\middle|\eta_1^n\right)}\left[P\left(\Theta \le n \middle| \eta_1^{n}\right) f\left(\eta_{n+1}\middle|\ \Theta \le n, \eta_1^n\right) +\right.$$

$$\left. + P\left(n+1 \middle| \eta_1^{n}\right) f\left(\eta_{n+1}\middle| n+1, \eta_1^n\right)\right].$$

Поскольку при зафиксированной реализации η_1^n событие, состоящее в наступлении случайной величины η_{n+1}, может сочетаться с одним из событий: $\Theta \le n$, $\Theta > n+1$ или $\Theta > n+1$, – составля-

ющих полную группу событий, плотность распределения $f\left(\eta_{n+1}\middle|\eta_1^n\right)$ можем представить в виде

$$f\left(\eta_{n+1}\middle|\eta_1^n\right)=P\left(\Theta\le n\middle|\eta_1^n\right)f\left(\eta_{n+1}\middle|\Theta\le n,\eta_1^n\right)+$$

$$+P\left(n+1\middle|\eta_1^n\right)f\left(\eta_{n+1}\middle|n+1,\eta_1^n\right)+$$

$$+P\left(\Theta>n+1\middle|\eta_1^n\right)f\left(\eta_{n+1}\middle|\Theta>n+1,\eta_1^n\right).$$

Тогда

$$\pi_{n+1}=\frac{A(n,\,n+1)+B(n,\,n+1)}{A(n,\,n+1)+B(n,\,n+1)+C(n,\,n+1)},\quad(3)$$

где

$$A(n,\,n+1)=P\left(\Theta\le n\middle|\eta_1^n\right)f_0(\eta_{n+1}),$$

$$B(n,\,n+1)=P\left(n+1\middle|\eta_1^n\right)f_0(\eta_{n+1}),$$

$$C(n,\,n+1)=P\left(\Theta>n+1\middle|\eta_1^n\right)f_0(\eta_{n+1}),$$

причем индексы 0 и 1 означают соответственно отсутствие и наличие сигнала в момент $n+1$.

Необходимые для составления равенства (3) значения $A(n, n+1)$, $B(n, n+1)$ и $C(n, n+1)$ могут быть получены с помощью апостериорных вероятностей наличия сигнала в разные моменты времени. Последние могут быть представлены в виде

$$P\left(\Theta\middle|\eta_1^n\right)=\frac{dP\left(\eta_1^n|\Theta\right)P(\Theta)}{dP\left(\eta_1^n\right)}\cdot\frac{P\left(\Theta>n\middle|\eta_1^n\right)}{P\left(\Theta>n\middle|\eta_1^n\right)}=$$

$$=\frac{dP\left(\eta_1^n|\Theta\right)}{d\left(\eta_1^n|\Theta>n\right)}\cdot(1-\pi_n)\frac{P(\Theta)}{P(\Theta>n)}=$$

$$=\begin{cases}\varphi_\Theta\dfrac{1-\pi_n}{P(\Theta>n)}P(\Theta), & \Theta\le n,\\[2ex] \dfrac{1-\pi_n}{P(\Theta>n)}P(\Theta), & \Theta>n,\end{cases}\quad(4)$$

где $\pi_n=P\left(\Theta\le n\middle|\eta_1^n\right)$, а $\varphi_n=\dfrac{dP\left(\eta_1^n|\Theta\right)}{dP\left(\eta_1^n|\Theta>n\right)}$ – отношение правдоподобия.

Поэтому входящие в рекуррентное соотношение (3) вероятности могут быть определены следующим образом:

$$P\left(n+1\middle|\eta_1^n\right)=\frac{P(n+1)}{P(\Theta>n)}(1-\pi_n);$$

$$P\left(\Theta>n+1\middle|\eta_1^n\right)=1-P\left(\Theta\le n\middle|\eta_1^n\right)-P\left(n+1\middle|\eta_1^n\right).$$

Обозначив теперь через φ_{n+1} отношение $\dfrac{f_1(\eta_{n+1})}{f_0(\eta_{n+1})}$, получим рекуррентное соотношение для π_n в виде

$$\pi_{n+1}=\frac{P(n+1)(1-\pi_n)\varphi_{n+1}+P(\Theta>n)\pi_n}{P(n+1)(1-\pi_n)(\varphi_{n+1}-1)+P(\Theta>n)}.$$

Нетрудно видеть, что статистика π_n является неубывающей функцией времени.

Аналогично можно показать, что для $\pi_n^{(-)}=P\left(\Theta<n\middle|\eta_1^n\right)$, $\pi_n^{(0)}=P\left(\Theta=n\middle|\eta_1^n\right)$ и $\pi_n^{(+)}=P\left(\Theta>n\middle|\eta_1^n\right)$ справедливы рекуррентные соотношения

$$\pi_{n+1}^{(-)}=\frac{P(n)\varphi_n+P(\Theta>n)\pi_n^{(-)}}{P(n+1)\left(1-\pi_n^{(-)}\right)+P(\Theta>n)+P(n)\varphi_n};$$

$$\pi_{n+1}^{(0)}=\frac{\pi_n^{(0)}\varphi_{n+1}P(n+1)}{\pi_n^{(0)}(\varphi_{n+1}-1)P(n+1)+\varphi_nP(n)};$$

$$\pi_{n+1}^{(+)}=\frac{\left[P(\Theta>n)-P(n+1)\right]\pi_n^{(+)}}{P(n+1)\pi_n^{(+)}(\varphi_{n+1}-1)+P(\Theta>n)}.$$

С помощью уравнения (4) может быть определено и соотношение для вычисления вероятности $\pi_n^{(m)}$ наличия сигнала в текущий момент времени n или в одном из моментов $n-1$, ..., $n-m$. Поскольку наступление сигнала в эти моменты времени есть событие несовместное, то

$$\pi_n^{(m)}=P\left(\Theta\in[n-m,\,n]\middle|\eta_1^n\right)=$$

$$=(1-\pi_n)\sum_{i=0}^{m}\varphi_{n-1}\frac{P_{n-i}}{P(\Theta>n-i)},$$

и в случае $m=1$, приведенном в примере:

$$\pi_n^{(1)}=(1-\pi_n)\left[\varphi_{n-1}\frac{P_{n-1}}{P(\Theta>n-1)}+\varphi_n\frac{P_n}{P(\Theta>n)}\right].$$

Литература

1. **Роббинс Г., Сигмунд Д., Чао И.** Теория оптимальных правил остановки: Пер. с англ. М.: Наука, 1975. 188 с.
2. **Розов А. К., Лось А. П., Зелялютдинов А. Р.** Новые возможности в обнаружении движущихся объектов // Информационно-управляющие системы. 2004. № 4(11). С. 16–20.
3. **Ширяев А. Н.** Статистический последовательный анализ. М.: Наука, 1976. 272 с.

УДК 004.415.2.032.24

ФОРМАЛИЗМ ДЛЯ ОПИСАНИЯ ПРОГРАММНЫХ СИСТЕМ И ВЫЧИСЛИТЕЛЬНЫХ ПРОЦЕССОВ ЦИКЛИЧЕСКОЙ ПАРАЛЛЕЛЬНОЙ ОБРАБОТКИ ДАННЫХ РЕАЛЬНОГО ВРЕМЕНИ[1]

И. В. Стручков,
аспирант
В. М. Ицыксон,
канд. техн. наук
Санкт-Петербургский государственный политехнический университет

Рассматриваются вопросы формализованного описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени для специализированных информационно-вычислительных комплексов. Для указанного класса систем предлагается новый способ формализованного описания – граф генераторов и преобразователей данных, который может использоваться для выделения функциональных модулей-задач, имитационного моделирования, статического и динамического распределения задач по процессорам в многопроцессорной системе.

The article addresses a formal description of cyclic real-time parallel computational processes and corresponding software systems for specialized computing complexes. A new formal approach for describing the specified class of systems is introduced: the data generator-transformer graph. This graph can be utilized for the decomposition of task modules, simulation, static and dynamic task mapping in a multiprocessor system.

Задача параллельной обработки большого объема потоковой информации является типичной для многоканальных систем анализа сигналов о состоянии пространственно-распределенного объекта. Такого рода системы используются в различных прикладных областях, таких как гидроакустика, радиолокация и подобных им, где возникают следующие задачи: обнаружение и идентификация объектов, сопровождение обнаруженных объектов, прогнозирование местоположения и скорости движения объектов, классификация объектов, спектральный и корреляционный анализ сигналов. Для систем подобного класса характерны следующие особенности:

– данные от аналогового источника поступают в реальном времени с высокой частотой дискретизации (до сотен мегагерц);

– количество параллельных каналов ввода может составлять от нескольких сотен до нескольких тысяч;

– для данных, полученных от различных каналов ввода, предусмотрена одинаковая или однотипная математическая обработка;

– вычислительные процедуры обработки данных представляют собой последовательный многоэтапный процесс, допускающий конвейерную организацию вычислений;

– вычислительные процедуры обработки повторяются многократно для новых порций данных и, таким образом, являются циклическими процессами.

При этом разработчики математического и программного обеспечения должны учитывать следующие условия:

– ограниченность вычислительных ресурсов системы, часто связанную со специальными требованиями, предъявляемыми к используемым в составе аппаратуры компонентам;

– предопределенность архитектуры аппаратного обеспечения системы обработки, вызванную высокой стоимостью и большой длительностью процесса разработки аппаратуры;

– жесткие требования к временным характеристикам.

[1] Статья выполнена в рамках работы по гранту для поддержки научно-исследовательской работы аспирантов вузов Федерального агентства по образованию 2004 года № А04-3. 16-482.

Совместный учет перечисленных особенностей и условий при разработке программного обеспечения требует специальной методики. При этом возможность применения стандартных технологий и средств разработки существенно ограничена.

Основным способом достижения требуемой производительности является параллельная организация вычислений, поэтому первостепенная задача – это эффективное распараллеливание вычислений. Она имеет два пути решения:

статическое распараллеливание – осуществляется на этапе проектирования либо сборки программной системы на основе анализа моделей вычислительных процессов и потоков данных;

динамическое распараллеливание – осуществляется в процессе функционирования программной системы с учетом фактических данных о загруженности процессоров и коммуникационных каналов.

При любом пути решения существенным оказывается учет особенностей системы и задачи обработки данных. Такой учет возможен при наличии формализованной модели вычислительного процесса, адекватно отражающей суть происходящих явлений.

Формализованное описание вычислительного процесса является важнейшей частью всего процесса разработки, и от качества выполнения данного этапа зависит качество системы в целом.

На различных этапах жизненного цикла программного обеспечения требуются различные типы формализованного описания:

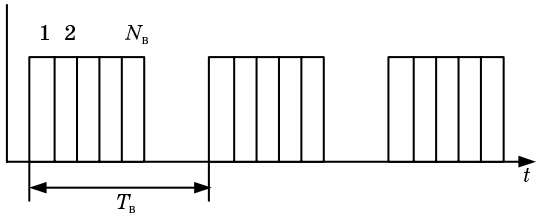
на этапе проектирования создаются структурные и поведенческие описания для выделения функциональных модулей-задач и определения информационных связей между ними;

на этапе разработки и компиляции формируются графы задач, потоков данных и соединений процессоров для статического распределения задач по имеющимся вычислительным ресурсам [1];

на этапе исполнения аналогичные графы используются для динамической балансировки нагрузки [2], автоматической диагностики и контроля работоспособности системы.

При исследовании многопроцессорного комплекса для обработки гидроакустических сигналов авторами была разработана теоретико-графовая модель параллельных вычислительных процессов – граф генераторов и преобразователей данных (ГГПД). Данная модель ориентирована на описание программ циклической параллельной обработки данных реального времени. Отличительной особенностью разработанной модели является возможность ее использования на всех трех перечисленных этапах жизненного цикла программного обеспечения.

При разработке модели ГГПД учитывалось, что важнейшей особенностью специализированных систем потоковой обработки информации является цикличность. Входные данные от внешних источников поступают, как правило, регулярными порциями фиксированного размера. Асинхронными и непериодическими событиями в системе являются изменения режима работы или параметров системы. Для функционирования системы в реальном времени необходимо, чтобы обработка очередной порции данных полностью завершалась в течение периода поступления входных данных.



■ ***Рис. 1.*** *Детерминированный всплесковой поток данных*

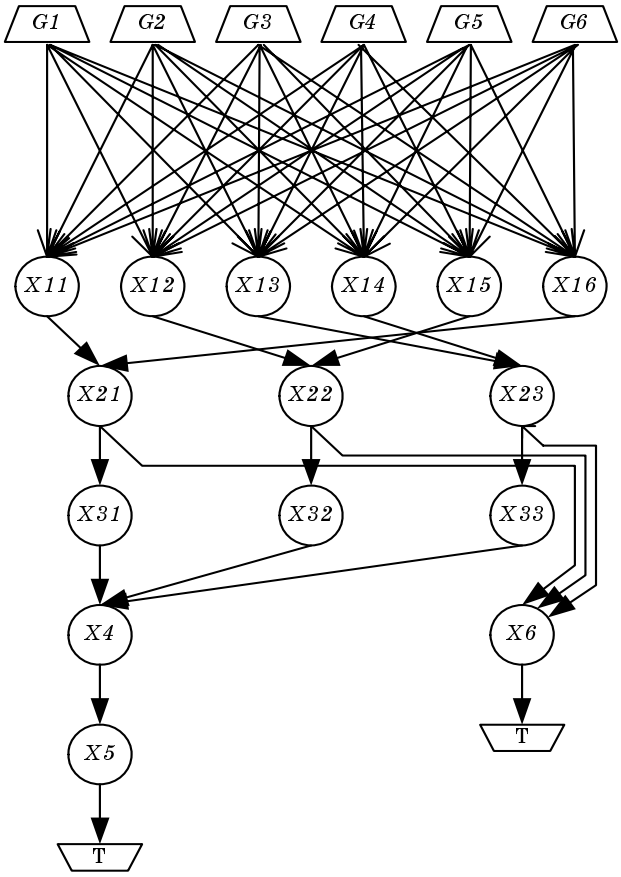
Анализ нескольких реальных систем обработки информации [3] показал, что потоки данных, создаваемые отдельными параллельными задачами, имеют всплесковый характер. Всплески состоят из одного или нескольких фрагментов данных, следующих непосредственно друг за другом (рис. 1). Периодичность всплесков задается детерминированной временной константой. Размер фрагмента данных фиксирован и задается коммуникационной подсистемой. Такие потоки далее будем называть детерминированными всплесковыми потоками.

Цикличность процессов в системе и всплесковый детерминированный характер потоков данных являются основой модели ГГПД. Граф ГПД содержит вершины трех типов: генераторы данных, преобразователи данных и терминаторы. Генератор автономно формирует всплески согласно заданной временной константе. Один генератор может создавать потоки одновременно для нескольких получателей. Генератор характеризуют следующие параметры: период всплесков ($T_в$); количество фрагментов во всплеске ($N_в$); получатели данных.

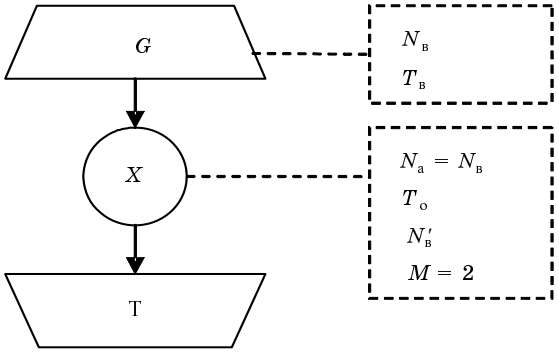
Преобразователь данных, в отличие от генератора, активизируется не автономно, а после получения заданного количества входных данных. Преобразователь характеризуют следующие параметры: количество входных фрагментов для активизации ($N_а$); время обработки ($T_о$); получатели данных; для каждого получателя – количество фрагментов во всплеске ($N_в$); для каждого получателя – кратность, т. е. количество активизаций, после которого генерируется всплеск (M).

Терминатор является просто заглушкой, не имеет параметров и на графе обозначает конец вычислений.

Таким образом, каждая задача в рассматриваемых системах выполняется циклически с обновленными входными данными. Период цикла зада-



■ ***Рис. 2.*** *ГГПД для программы параллельной обработки гидроакустических сигналов*



■ ***Рис. 3.*** *Пример простого ГГПД*

чи-генератора равен $T_в$. Период цикла задачи-преобразователя зависит от скорости поступления входных данных. Полному циклу соответствует M поступлений по $N_а$ фрагментов данных (активизаций), причем при каждой активизации затрачивается время на обработку $T_о$.

Граф генераторов и преобразователей данных позволяет описывать параллельные программы из класса циклических программ параллельной обработки данных с детерминированными всплесковыми потоками данных. В частности, определенная комбинация генераторов и преобразователей на заданной топологии позволила адекватно моделировать реально исследованную задачу параллельной обработки гидроакустических сигналов [3]. Для исследованной параллельной программы ГГПД представлен на рис. 2 (задачи-генераторы обозначены G_i, задачи-преобразователи – X_{jk}, терминаторы – Т). Формально, с каждой вершиной графа связаны соответствующие параметры (рис. 3). На рис. 2 данные параметры не показаны в целях наглядности.

Поскольку ГГПД отражает логическую структуру программного обеспечения, данное описание может создаваться уже на этапе проектирования. На этапе разработки и компиляции необходимо решить задачу оптимального статического распараллеливания. Возможны два пути применения ГГПД на этом этапе:

– преобразование ГГПД в иную модель, для которой существует методика статического распараллеливания;

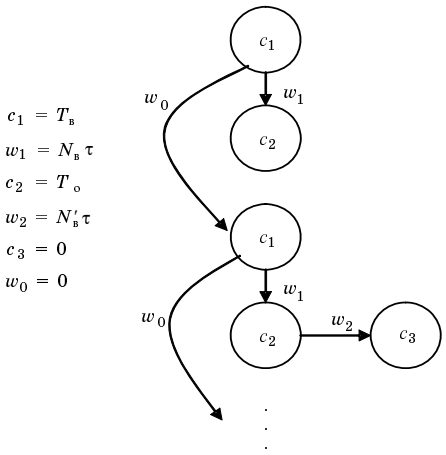
– создание методики статического распараллеливания, использующей ГГПД непосредственно в качестве исходной информации.

Покажем возможность эквивалентного перехода от ГГПД к другим моделям описания параллельных процессов. За основу возьмем простой граф (см. рис. 3).

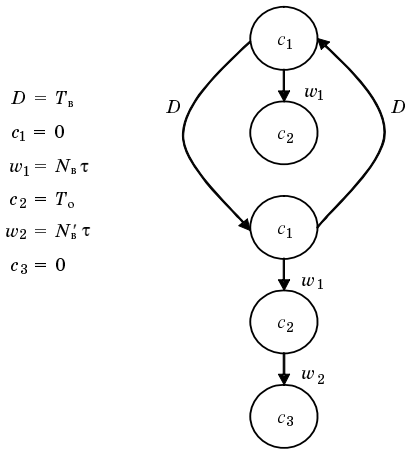
Наиболее распространенным способом формализованного описания параллельных процессов являются графы задач. В настоящее время предложено множество подходов к построению графов задач [4, 5], из которых в данной работе рассматриваются направленные ациклические графы (DAG), итеративные графы задач (ITG) и временные коммуникационные графы (TCG).

Рассмотрим переход от графа на рис. 3 к направленному ациклическому графу (DAG) на рис. 4.

Направленные ациклические графы являются наиболее простым и часто используемым в алгоритмах распределения нагрузки способом описания параллельных программ. В таком графе вершины представляют задачи, а ребра – передачу данных между задачами. При описании параллельных программ обычно используют взвешенные направленные ациклические графы. Веса, придаваемые вершинам, отражают вычислительную сложность данной задачи, а веса, придаваемые ребрам, – затраты на передачу данных между двумя данными задачами. Основным недостатком ациклических графов является невозможность адекватно описать итеративные вычисления. Вследствие этого с помощью ациклических графов обычно описывают только параллельные программы с крупным делением на задачи. В этом случае



■ ***Рис. 4.*** *Эквивалентный направленный ациклический граф (τ – время передачи одного фрагмента данных)*



■ ***Рис. 5.*** *Эквивалентный итеративный граф задач*

все итеративные вычисления считаются упрятанными внутри одной задачи (и, следовательно, являются последовательными). Другим решением является развертывание циклов в виде вложенных ациклических подграфов. Однако это допустимо только в том случае, когда число итераций известно на этапе компиляции, что случается достаточно редко.

На рис. 4 периоду генератора данных соответствует вершина графа c_1, времени обработки данных преобразователем соответствует вершина c_2. Отметим, что для описания кратности преобразователя данных M в DAG требуется дублирование пары вершин c_1, c_2. В нашем случае $M = 2$, при больших значениях M потребуется соответствующее число повторений вершин в DAG. Фиктивная передача данных с $w_0 = 0$ необходима для обеспечения связности графа. Ациклическая природа графа DAG не позволяет адекватно описать бесконечный процесс, поэтому граф неограничен.

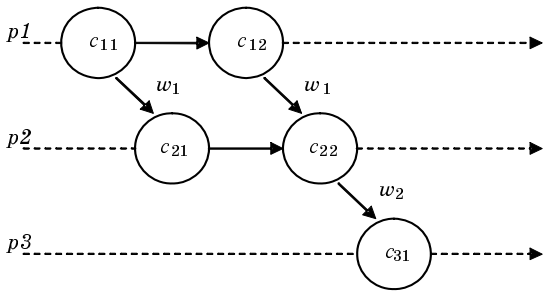
Преимуществом итеративного графа задач (ITG) является возможность компактного описания циклов, поэтому эквивалентный ITG является ограниченным. В итеративных графах задач ребру графа помимо затрат на передачу данных может присваиваться также дополнительное значение – задержка. При этом обязательно должно соблюдаться условие: в любом цикле должна находиться, по крайней мере, одна задержка. Ребра без задержек моделируют зависимости внутри одной итерации, связи с задержками – зависимости между итерациями. Тем не менее, описание кратности преобразователя данных также требует дублирования вершин. Период генератора в данном случае удобнее описать не весом вершины c_1, а задержкой D, c_1 при этом принимается равным нулю. В результате получаем граф, представленный на рис. 5.

Временной коммуникационный граф (TCG) для рассматриваемого случая является неограниченным, как и DAG. Некоторым преимуществом данного описания (рис. 6) является явное выделение параллельных процессов (задач): $p1$ соответствует задаче-генератору, $p2$ – задаче-преобразователю, $p3$ – задаче-заглушке.

Из рассмотренных примеров очевидно, что возможно построение формальных схем перехода от описания ГТПД к иным моделям графового представления параллельных вычислительных процессов. Большинство существующих алгоритмов статического и динамического распределения нагрузки в многопроцессорных системах [1, 2] базируются на рассмотренных стандартных графовых моделях описания вычислительных процессов в качестве исходной информации. Наличие формальных процедур преобразования описания ГТПД в одну из таких моделей позволит автоматически адаптировать имеющееся описание к стандартным средствам.

Имеющиеся подходы были разработаны для описания широкого класса параллельных программ и, как показано выше, не вполне эффективны для описания циклических многоэтапных процессов, подразумевающих однотипную математическую обработку. Предложенная модель ГТПД не только компактнее и нагляднее описывает процессы циклической параллельной обработки, но и может быть непосредственно использована в процессе разработки и оптимизации параллельных программ.

Имея описание вычислительного процесса в виде ГТПД, разработчик системы получает возмож-



$c_{11} = c_{12} = T_в$
$w_1 = N_в\tau$
$c_{21} = c_{22} = T_о$
$w_2 = N'_в\tau$
$c_{31} = 0$

■ ***Рис. 6.*** *Эквивалентный временной коммуникационный граф*

ность осуществить имитационное моделирование ее функционирования до написания какого-либо программного кода. Такое моделирование на этапе проектирования позволит избежать множества ошибок при разработке. В ходе исследования эффективности коммуникационных подсистем для многопроцессорного комплекса [3] были разработаны имитационные модели аппаратной архитектуры, коммуникационных подсистем и прикладных задач. Для имитационных экспериментов использовалась среда моделирования OMNet++. Имитационные модели параллельных задач базируются на введенных абстракциях генераторов и преобразователей данных, что позволяет использовать описание ГГПД в качестве исходных данных без предварительного преобразования.

Описание ГГПД также может являться входной информацией алгоритма распределения задач по процессорам. Распределение задач по процессорам сводится к решению следующей многокритериальной и, в общем случае, нелинейной задачи оптимизации:

$$\min_{\forall M: S \xrightarrow{M} H} CF(M),$$

где S – граф задач (в нашем случае, ГГПД); H – направленный граф соединений процессоров (вершинам соответствуют процессоры, дугам – межпроцессорные соединения); M принимает значение всех возможных отображений графа задач на граф соединений процессоров.

Для решения данной задачи применяются итеративные алгоритмы оптимизации. Общий вид целевой (минимизируемой) функции для итеративного алгоритма оптимизации можно представить в виде суммы трех компонент [1]:

$$CF(M, t) = F_{верш}(M, t) + F_{дуг}(M, t) + \\ + F_{марш}(M, t, RA),$$

где M – отображение задач на процессоры; t – шаг алгоритма оптимизации; RA – функция маршрутизации сообщений и данных.

Покажем, как сформировать целевую функцию алгоритма оптимизации распределения задач на основе описания ГГПД.

Целевая функция вершин ($F_{верш}$) соответствует затратам на вычисления в процессорах. Для каждой задачи-преобразователя определяется вычислительный вес:

$$c_i = \frac{T_{oi}\sum_{j \in I}\lambda_j}{N_{ai}},\ i \in X; \\ \lambda_j = \frac{N_{вj}}{T_{cj}}; \qquad (*) \\ T_{cj} = \begin{cases} \dfrac{M_j T_{oTask(j)}}{c_{Task(j)}},\ Task(j) \in X; \\ T_{вj},\ Task(j) \in G, \end{cases}$$

где T_{oi} – время обработки i-й задачи-преобразователя; λ_j – интенсивность j-го входящего потока; I – множество входящих потоков задачи; N_{ai} – количество фрагментов для активизации i-й задачи-преобразователя; X – множество задач-преобразователей; $N_{вj}$ – количество фрагментов во всплеске для j-го входящего потока; T_{cj} – период всплесков j-го входящего потока (как от генератора, так и от преобразователя); M_j – кратность для j-го входящего потока; $Task(j)$ – индекс задачи, генерирующей входящий поток j; $T_{вj}$ – период всплесков для j-го входящего потока (только от генератора); G – множество задач-генераторов.

Значение целевой функции вершин рассчитывается, как суммарный дисбаланс нагрузок по всем процессорам:

$$F_{верш}(M, t) = \alpha(t)\sum_{i=1}^{N} | c_i - c_{ср} |,$$

где $\alpha(t)$ – нормировочный множитель, зависящий от шага алгоритма; N – число процессоров; c_i – суммарный вычислительный вес всех задач, распределенных на данный процессор; $c_{ср}$ – средний суммарный вычислительный вес по всем процессорам.

Целевая функция дуг ($F_{дуг}$) соответствует затратам на передачу данных:

$$F_{дуг} = \beta(t) \sum_{i,\ j \in T} \lambda_{ij} d_{ij} w,$$

где $\beta(t)$ – нормировочный множитель, зависящий от шага алгоритма; T – множество всех задач; λ_{ij} – интенсивность потока данных от задачи i к задаче j из формулы (*); d_{ij} – расстояние от задачи i до задачи j в узлах, согласно таблице маршрутиза-

ции; w – затраты на передачу единицы (фрагмента) данных между соседними узлами.

Затраты на маршрутизацию ($F_{\text{марш}}$) учитывают загруженность коммуникационных каналов (линков). Загруженность l-го линка определяется по формуле

$$AD_l = \sum_{i,\ j \in T} \lambda_{ij} w RA(i,\ j,\ l),$$

где $RA(i, j, t)$ – функция маршрутизации ($RA = 1$, если данные от задачи i до задачи j проходят через линк l).

Целевая функция маршрутизации $F_{\text{марш}}$ определяется, как суммарный дисбаланс загруженностей линков:

$$F_{\text{марш}}(M,\ t,\ RA) = \gamma(t) \sum_{l=1}^{L} | AD_l - AD_{\text{ср}} |,$$

где $\gamma(t)$ – нормировочный множитель, зависящий от шага алгоритма; L – число линков; AD_l – загруженность l-го линка; $AD_{\text{ср}}$ – среднее значение загруженности по всем линкам.

Выводы

В данной статье были рассмотрены способы формализованного описания параллельных программ для вычислительных систем с распределенной памятью и представлена новая модель графового описания параллельных программ – граф генераторов и преобразователей данных. ГГПД является расширением графа потоков данных и применим для описания параллельных программных систем и вычислительных процессов циклической обработки данных реального времени. Примером систем данного класса является программное обеспечение вычислительного комплекса многоканальной обработки гидроакустических сигналов.

В вычислительных процессах указанного класса потоки данных имеют периодический всплесковый характер. Параметры вершин в ГГПД основаны на данном представлении потоков данных. Анализ применения ГГПД для описания реальной параллельной программы и сравнение с другими моделями описания позволяют сделать следующие выводы.

1. Описание ГГПД является естественным и наглядным визуальным формализмом для разработчиков системы, так как каждой вершине графа соответствует вполне реальная программная сущность – задача.

2. Описание ГГПД является наиболее компактным для указанного класса задач, по сравнению с эквивалентными описаниями DAG, ITG и TCG, так как скрывает цикличность и кратность обработки данных в семантике вершин.

3. Можно предложить формальную схему перехода от ГГПД к любой из рассмотренных моделей, что позволит разработать автоматизированный программный инструмент для подобных преобразований.

4. Описание ГГПД может быть непосредственно использовано как входная информация для имитационного моделирования системы или алгоритма распределения задач.

Таким образом, модель ГГПД является новой и более эффективной формой описания параллельных программ и вычислительных процессов в классе циклических задач обработки данных реального времени. Модель ГГПД программы, благодаря своей наглядности, может быть построена разработчиком (системным архитектором) на этапе проектирования. В процессе разработки описание ГГПД может быть переведено в эквивалентное представление другого типа, что может оказаться необходимым условием для автоматического (автоматизированного) распределения задач по процессорам вычислительной системы.

Направлением дальнейшего развития предложенного подхода является исследование возможности применения модели ГГПД для статического и динамического распределения задач. В результате исследования должна быть предложена эффективная методика такого распределения и разработан прототип системы управления параллельными задачами на основе данной методики.

Литература

1. **Hluchэ L., Dobrovodskэ M., Dobruckэ M.** Static Mapping Methods for Processor Networks. http://citeseer.ist.psu.edu/472693.html
2. **Hu Y., Blake R.** An optimal dynamic load balancing algorithm. Technical Report DL-P95 -011. Daresbury Laboratory. Warrington, UK, 1995. http://citeseer.ist.psu.edu/hu95optimal.html
3. **Стручков И. В., Ицыксон В. М., Мелехин В. Ф.** Исследование эффективности коммуникационных систем для многопроцессорного комплекса с распределенной памятью // XXXII Неделя науки СПбГПУ: Материалы межвуз. науч. конф. / СПбГПУ. СПб., 2004. С. 87–89.
4. **Lo Virginia M.** Temporal Communication Graphs: Lamport's Process-Time Graphs Augmented for the Purpose of Mapping and Scheduling // Journal of Parallel and Distributed Computing. 1992. N 16(4). December. ftp://ftp.cs.uoregon.edu/pub/lo/tcg.ps.Z
5. **Sinnen O., Sousa L.** A Platform Independent Parallelising Tool Based on Graph Theoretic Models // Lecture Notes in Computer Science. 2001. Vol. 1981. http://www.ece.auckland.ac.nz/~sinnen/articles/Sinnen2001pip.ps

УДК 65.012.122

СУБОПТИМАЛЬНЫЙ АЛГОРИТМ ПОСТРОЕНИЯ РАСПИСАНИЙ ДЛЯ ИЕРАРХИЧЕСКИХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ

Н. В. Колесов,
доктор техн. наук, профессор, начальник сектора
М. В. Толмачева,
ведущий инженер
ГНЦ РФ ЦНИИ «Электроприбор» (*Санкт-Петербург*)

Рассматривается субоптимальный алгоритм составления расписаний в иерархических вычислительных системах, практически не требующий перебора вариантов. Приводятся результаты исследований его эффективности на основе случайного генерирования примеров.

We propose a suboptimal scheduling algorithm for hierarchical computing systems which requires almost no case study of schedule versions. It is based on several optimal no-case-study scheduling algorithms. Each of the algorithms is intended for its own, rather narrow basic class of systems.

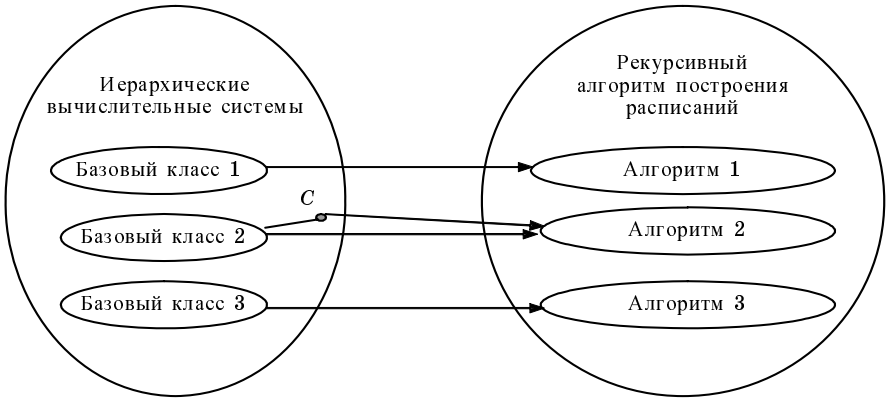
Введение

Проблема построения расписаний выполнения задач в многопроцессорных системах обработки информации является достаточно важной, и ей посвящены многочисленные публикации, ссылки на которые можно найти, например, в работах [1–3]. Данная проблема может рассматриваться в различной постановке. В настоящей статье она рассматривается применительно к иерархической вычислительной системе, которая решает некоторую последовательность задач. При этом каждая задача разбита на фрагменты по числу процессоров. Предполагаются известными времена решения для всех фрагментов каждой из задач. Известно [1–3], что решение проблемы составления расписаний в общем случае в той или иной степени предполагает перебор вариантов возможных расписаний и характеризуется высокой алгоритмической сложностью. Ниже предлагается субоптимальный алгоритм построения расписаний, практически не требующий перебора вариантов. Дальнейшее изложение отчасти коррелируется с предыдущей работой авторов [4], которая была посвящена существенно более узкому классу вычислительных систем – конвейерным системам.

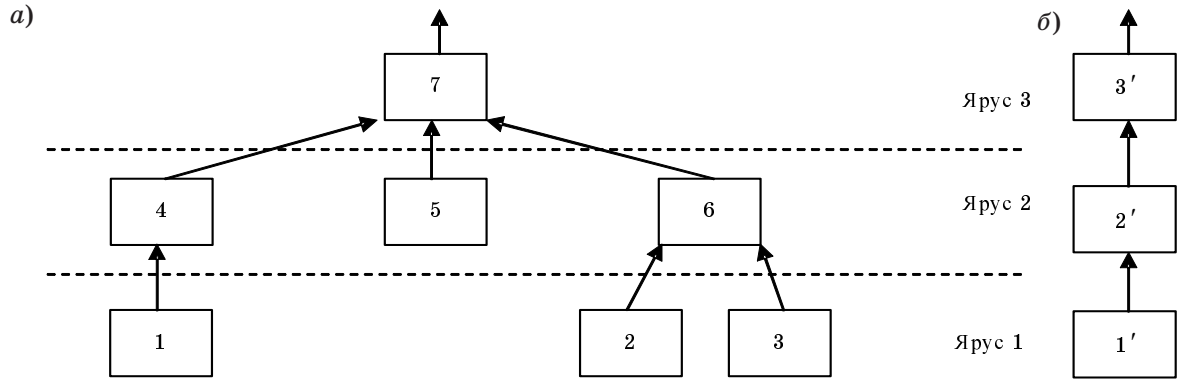
Постановка задачи и основная идея предлагаемого алгоритма

Пусть рассматриваемая система представляет собой иерархию из m процессоров $L = \{L_1, L_2, ..., L_m\}$, на вход которой поступает последовательность $(J_1, J_2, ..., J_n)$ из n задач. Каждая задача разбивается на m фрагментов (по числу процессоров). При этом i-й фрагмент j-й задачи решается на i-м процессоре за время $\tau_{i,j}$ $i = \overline{1, m}$, $j = \overline{1, n}$ и $\tau_{i,j} = t^{к}_{i,j} - t^{н}_{i,j}$, где $t^{н}_{i,j}$ и $t^{к}_{i,j}$ – времена начала и конца решения j-й задачи на i-м процессоре. Проблема состоит в определении расписания решения задач, которое не существенно проигрывает минимальному расписанию по суммарному времени решения всех задач. Заметим, что под минимальным расписанием понимается расписание, характеризующееся минимальным временем решения всех задач.

Для решения поставленной задачи предлагается субоптимальный рекурсивный алгоритм, основная идея которого сводится к следующему (рис. 1). В основе предлагаемого алгоритма лежат три одношаговых алгоритма построения расписаний, каждый из которых ориентирован на свой, в общем случае достаточно узкий, базовый класс систем. Важно то, что эти алгоритмы являются беспереборными и оптимальными. В предложенном алгоритме на каждом шаге рекурсии в будущем расписании размещаются одна или две задачи из числа не размещенных на предыдущих шагах. При этом используемый одношаговый алгоритм размещения соответствует тому базовому классу, к которому наиболее близка рассматриваемая на данном шаге система. Эти действия повторяются до исчерпания заданного списка задач.



■ ***Рис. 1.*** *Иллюстрация основной идеи предлагаемого алгоритма*



■ ***Рис. 2.*** *Примеры иерархических систем: а – иерархическая система общего вида; б – конвейерная система*

Базовые классы иерархических систем

Обсудим предварительно основные понятия и, прежде всего, понятие иерархической системы. На рис. 2, *а* приведен пример трехъярусной иерархической системы из семи процессоров. Следует подчеркнуть, что в данном контексте речь идет об иерархических информационных связях. При этом их физическая реализация может быть другой и иметь, например, магистральную архитектуру. Система на рис. 2, *а* не имеет циклов и разветвлений, включает четыре входных процессора *1–3* и *5* и один выходной *7*. Отметим, что конвейерная система является частным случаем иерархической системы, когда каждый ярус содержит лишь один процессор (рис. 2, *б*).

Каждый входной процессор L_i связан с выходным процессором L_m некоторым вычислительным путем (последовательностью процессоров) $p_k = L_i, L_j, ..., L_m$. Назовем величину $t_{k,j}$ временем выполнения вычислительного пути p_k на j-й задаче. Определим ее как сумму времен выполнения фрагментов j-й задачи процессорами, принадлежащими этому пути, что с использованием нумерации процессоров вдоль пути p_k можно записать:

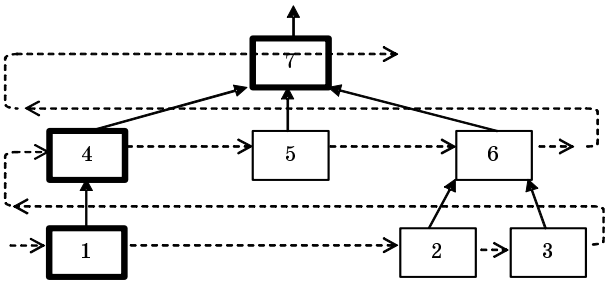
$$t_{k,j} = \sum_{i=1}^{m_k} \tau_{i,j},$$

где m_k – число процессоров, принадлежащих пути p_k.

Заметим, что в общем случае реальное время выполнения вычислений на этом пути будет больше из-за необходимости ожидания готовности исходных данных для процессоров вычислительного пути p_k, вычисляемых в процессорах, не принадлежащих этому пути.

Назовем вычислительный путь p_j^* критическим для j-й задачи, если время его выполнения на j-й задаче является максимальным среди всех остальных путей системы. Очевидно, что для разных задач, решаемых в одной и той же системе, критические вычислительные пути могут быть различными.

Для любой иерархической системы время $t_{i,j}^{н}$ начала обработки информации в некотором процессоре L_i определяется временем подготовки для него всех исходных данных, которые, в свою очередь, складываются из ряда частных данных, подготавливаемых предшествующими процессорами



■ ***Рис. 3.*** *Иллюстрация отношения доминирования для систем класса 1*

L_r по каждому r-му информационному входу L_i (так, на рис. 1, *а* процессору *7* предшествуют процессоры *4*, *5* и *6*). В результате время готовности данных для процессора L_i определяется максимальным среди времен $t^{к}_{r,j}$ подготовки частных данных, т. е.

$$t^{н}_{i,j} = \max_{r}\{t^{к}_{r,j}\}.$$

В свою очередь, время $t^{к}_{r,j}$ подготовки частных данных по некоторому вычислительному пути определяется не только временем обработки данных в процессорах этого пути, но также временем ожидания готовности данных и временем начала вычислений во входном процессоре на рассматриваемом вычислительном пути. Последняя величина имеет в общем случае разные значения для разных вычислительных путей для любой задачи, кроме первой.

Рассмотрим три базовых для данного рассмотрения класса иерархических систем. В дальнейшем этот список может быть расширен. Особенностью всех рассматриваемых классов является существование характерных упорядоченностей на множестве процессоров по определяемому ниже отношению доминирования. Первый класс связан с наличием убывающей упорядоченности, второй класс – с наличием возрастающей упорядоченности, третий класс – с наличием двух упорядоченностей (возрастающей и убывающей).

Введем на множестве процессоров отношение доминирования «>».

Определение. Процессор L_q доминирует над процессором L_r ($L_q > L_r$), если

$$\min_{j}\tau_{q,j} \;\geq\; \max_{j}\tau_{r,j}.$$

Определим три базовых класса иерархических систем.

Класс 1. Множество процессоров представляет собой последовательность $L_m > L_2 > ... > L_m$, убывающую по отношению доминирования как вдоль каждого из ярусов системы, так и от нижнего яруса к верхнему таким образом, что процессоры, являющиеся максимальными элементами каждого из ярусов, образуют путь (критический).

Класс 2. Множество процессоров представляет собой последовательность $L_1 < L_2 <...< L_m$, возрастающую по отношению доминирования как вдоль каждого из ярусов системы, так и от нижнего яруса к верхнему таким образом, что процессоры, являющиеся максимальными элементами каждого из ярусов, образуют путь (критический).

Класс 3. Множество процессоров представляет собой пару соединенных последовательностей $L_1 < L_2 <...< L_h >...> L_{m-1} > L_m$, $1 \le h \le m$, первая из которых возрастает, а вторая убывает по отношению доминирования как вдоль каждого из ярусов системы, так и от нижнего яруса к верхнему таким образом, что процессоры, являющиеся максимальными элементами каждого из ярусов, образуют путь (критический путь).

На рис. 3 приведена иерархическая система, соответствующая классу 1. Штриховой линией показано отношение доминирования, а жирными линиями – процессоры *1*, *4* и *7*, являющиеся максимальными элементами ярусов. Путь, который образуют эти процессоры, является критическим для всех задач системы.

Построение оптимальных расписаний для базовых классов

Покажем теперь, как сконструировать оптимальное расписание в системе из класса 1. При этом характеристики критического пути будем отмечать звездочкой (*).

Алгоритм 1

1. Выделим задачу J_s, которая удовлетворяет условию

$$\sum_{i=2}^{m^*}\tau^{*}_{i,s} = \min_{j}\left\{\sum_{i=2}^{m^*}\tau^{*}_{i,j}\right\}.$$

2. Сформируем расписание $\pi_1 = [\tilde{\pi}\;\; J_s]$, где $\tilde{\pi}$ – произвольное расписание для $(n-1)$ задач, не содержащее задачи J_s.

Покажем оптимальность алгоритма 1. Предварительно отметим важную особенность систем из класса 1, заключающуюся в том, что ни перед одним из процессоров, принадлежащих критическому пути, кроме входного, никогда не образуется очередь из фрагментов задач, ожидающих решения. Очередь на входе процессора может возникать по двум причинам: из-за ожидания подготовленными на критическом пути исходными частными данными готовности других частных данных, а также из-за ожидания момента окончания решения на данном процессоре фрагмента предыдущей задачи. Ожидание в силу первой причины исключается, поскольку при любой задаче решение на любом процессоре критического пути будет всегда заканчиваться позже, чем на других процессорах того же яруса. Это будет происходить ввиду

того, что этот процессор является максимальным элементом яруса по отношению доминирования, а время начала вычислений на входном процессоре критического пути всегда будет наиболее поздним среди всех входных процессоров. Ожидание в силу второй причины также исключается, поскольку ввиду доминирования предшествующего процессора критического пути над последующим самый короткий для предшествующего процессора фрагмент любой задачи будет решен за большее время, нежели самый длинный фрагмент любой другой задачи на последующем процессоре.

С учетом сказанного можно записать выражение для времени $T_1(\pi)$ выполнения произвольного расписания π в системе из класса 1. Это время можно представить в виде суммы двух составляющих: времени ожидания в исходной очереди для фрагмента последней n-й задачи, соответствующего входному процессору критического пути, а также времени t_n^* выполнения критического пути для этой задачи. Такое представление возможно, поскольку момент начала ожидания для входного фрагмента критического пути n-й задачи совпадает с моментом начала выполнения расписания, а момент окончания выполнения критического пути – с моментом окончания выполнения расписания. Причем время ожидания будет определяться суммой времен решения входных фрагментов критического пути для всех задач, кроме последней. В результате с использованием нумерации процессоров вдоль критического пути имеем

$$T_1(\pi) = \sum_{j=1}^{n-1} \tau_{1,j}^* + t_n^* = \sum_{j=1}^{n-1} \tau_{1,j}^* + \sum_{i=1}^{m^*} \tau_{i,n}^*,$$

где $\tau_{i,j}^*$ – время решения фрагмента j-й задачи на i-м процессоре критического пути системы; m^* – число процессоров, принадлежащих критическому пути.

Перенося для удобства анализа первое слагаемое из второй суммы в первую сумму, получаем

$$T_1(\pi) = \sum_{j=1}^{n} \tau_{1,j}^* + \sum_{i=2}^{m^*} \tau_{i,n}^*.$$

Значение первой суммы представляет собой суммарное время, затрачиваемое на решение входных фрагментов критического пути для всех задач. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. В отличие от первой суммы, вторая зависит от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой суммы. Отсюда следует, что расписание, формируемое в алгоритме 1, является оптимальным для класса 1.

Отметим два существенных обстоятельства. Во-первых, в данном алгоритме отсутствует перебор вариантов расписаний. Во-вторых, для оптимальности расписания достаточно лишь правильного выбора последней исполняемой задачи, которая должна быть наиболее быстро решаемой на всех процессорах критического пути системы, кроме, возможно, первого. Упорядоченность же остальных задач не влияет на время исполнения расписания.

Рассмотрим правила формирования расписания во втором случае, который является в определенном смысле противоположным первому. Он также характеризуется определяющим свойством, но оно заключается в том, что перед каждым из процессоров критического пути системы всегда образуется очередь из фрагментов задач, ожидающих решения. В результате каждая задача, кроме первой, будет стартовать на любом процессоре критического пути с некоторой задержкой относительно момента ее прибытия на вход процессора, поскольку любой i-й фрагмент j-й задачи будет заканчиваться позже, нежели $(i-1)$-й фрагмент следующей за ней $(j+1)$-й задачи. Этот факт также следует из определения отношения доминирования. Алгоритм конструирования расписания в этом случае будет выглядеть следующим образом.

Алгоритм 2

1. Выделим задачу J_l, которая удовлетворяет условию

$$\sum_{i=1}^{m^*-1} \tau_{i,l}^* = \min_j \left\{ \sum_{i=1}^{m^*-1} \tau_{i,j}^* \right\}.$$

2. Сформируем расписание $\pi_2 = [J_l \tilde{\pi}]$, где $\tilde{\pi}$ – произвольное расписание для $(n-1)$ задач, не содержащее задачи J_l.

Покажем оптимальность этого алгоритма. Запишем выражение для времени выполнения произвольного расписания для системы из класса 2. Учтем, что перед последним m^*-м процессором критического пути системы (как и перед всеми другими) всегда существует очередь, а значит, этот процессор всегда занят с момента появления на его входе первой и до момента ухода с него последней задачи. В связи с этим время $T_2(\pi)$ выполнения произвольного расписания π можно представить в виде суммы времени t_1, необходимого для решения первой задачи, и времени решения последних m^*-х фрагментов критического пути для всех остальных задач. Заметив, что время решения первой задачи равно времени выполнения критического пути, получаем

$$T_2(\pi) = t_1 + \sum_{j=2}^{n} \tau_{m^*,j}^* = \sum_{i=1}^{m^*} \tau_{i,1}^* + \sum_{j=2}^{n} \tau_{m^*,j}^*.$$

Перенося для удобства анализа последнее слагаемое из первой суммы во вторую сумму, получаем

$$T_2(\pi) = \sum_{i=1}^{m^*-1} \tau_{i,1}^* + \sum_{j=1}^{n} \tau_{m^*,j}^*. \qquad (*)$$

Значение второй суммы представляет собой суммарное время, необходимое для решения пос-

ледних фрагментов критического пути для всех задач. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. В отличие от второй суммы первая зависит от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой суммы. Отсюда следует, что расписание, формируемое в алгоритме 2, является оптимальным для класса 2.

Так же как и в предыдущем случае, в данном алгоритме отсутствует перебор вариантов расписаний, а для оптимальности расписания достаточно лишь правильного выбора первой исполняемой задачи, которая должна быть наиболее быстро решаемой на всех процессорах критического пути, кроме, возможно, последнего. Упорядоченность же остальных задач не влияет на время выполнения расписания.

Рассмотрим задачу построения оптимального расписания для систем из класса 3.

Алгоритм 3

1. Выделим задачу J_s, которая удовлетворяет условию

$$\sum_{i=1}^{h-1}\tau^*_{i,s}=\min_j\left\{\sum_{i=1}^{h^*-1}\tau^*_{i,j}\right\}.$$

2. Выделим задачу J_p, которая удовлетворяет условию

$$\sum_{i=h^*+1}^{m^*}\tau^*_{i,p}=\min_{\substack{j\\ j\neq s}}\left\{\sum_{i=h^*+1}^{m^*}\tau^*_{i,j}\right\}.$$

3. Сформируем расписание $\pi_3=[J_s\ \tilde{\pi}\ J_p]$, где $\tilde{\pi}$ – произвольное расписание для $(n-2)$ задач, не содержащее задач J_s и J_p.

4. Сформируем расписание $\pi_4=[J_q\ \tilde{\pi}\ J_r]$, повторив пп. 1–3, но выполнив пп. 1 и 2 в другой последовательности.

5. Выберем среди расписаний π_3 и π_4 наилучшее расписание.

Покажем оптимальность этого алгоритма. Разобьем критический путь системы и, соответственно, всю систему на две части. В первую часть критического пути включим процессоры с первого по (h^*-1)-й, а во вторую – с h^*-го по m^*-й процессоры. Соответственно в первую часть системы включим процессоры ярусов с 1-го по (h^*-1)-й, а во вторую – все остальные процессоры. Очевидно, что первая часть системы соответствует системе из класса 2. В результате фрагмент каждой задачи перед решением на любом процессоре соответствующей части критического пути попадает в очередь. Поскольку то же самое справедливо и для h^*-го процессора, можно утверждать, что вторая часть системы соответствует системе из класса 1. Действительно, не только имеет место необходимая упорядоченность процессоров, но и фрагменты задач выходят на соответствующую часть критического пути без каких-либо дополнительных задержек так, как будто они все одновременно стоят в очереди к h^*-му процессору. Запишем время $T_3(\pi)$ выполнения произвольного расписания в рассматриваемой системе. Представим искомое время $T_3(\pi)$ как сумму двух величин. Первая величина – это время $t_n(1, h^*-1)$ решения всех n задач в первой части системы (от начала решения входного фрагмента первой задачи на первом процессоре критического пути до начала решения фрагмента n-й задачи на h^*-м процессоре), вторая величина – это время $t_n(h, m)$ решения n-й задачи на второй части системы:

$$T_3(\pi)=t_n(1,h^*-1)+t_n(h^*,m^*)\ .$$

Очевидно, что время $t_n(1, h^*-1)$ будет равняться времени решения $(n-1)$-й задачи на процессорах с 1-го по h^*-й, которое определяется выражением (*) и имеет вид

$$t_n(1,\ h^*-1)=t_{n-1}(1,\ h^*)=\sum_{i=1}^{h^*}\tau^*_{i,1}+\sum_{j=2}^{n-1}\tau^*_{h^*,j}.$$

Время $t_n(h^*, m^*)$ решения n-й задачи на второй части системы определяется очевидным выражением

$$t_n(h^*,\ m^*)=\sum_{i=h^*}^{m^*}\tau^*_{i,n}.$$

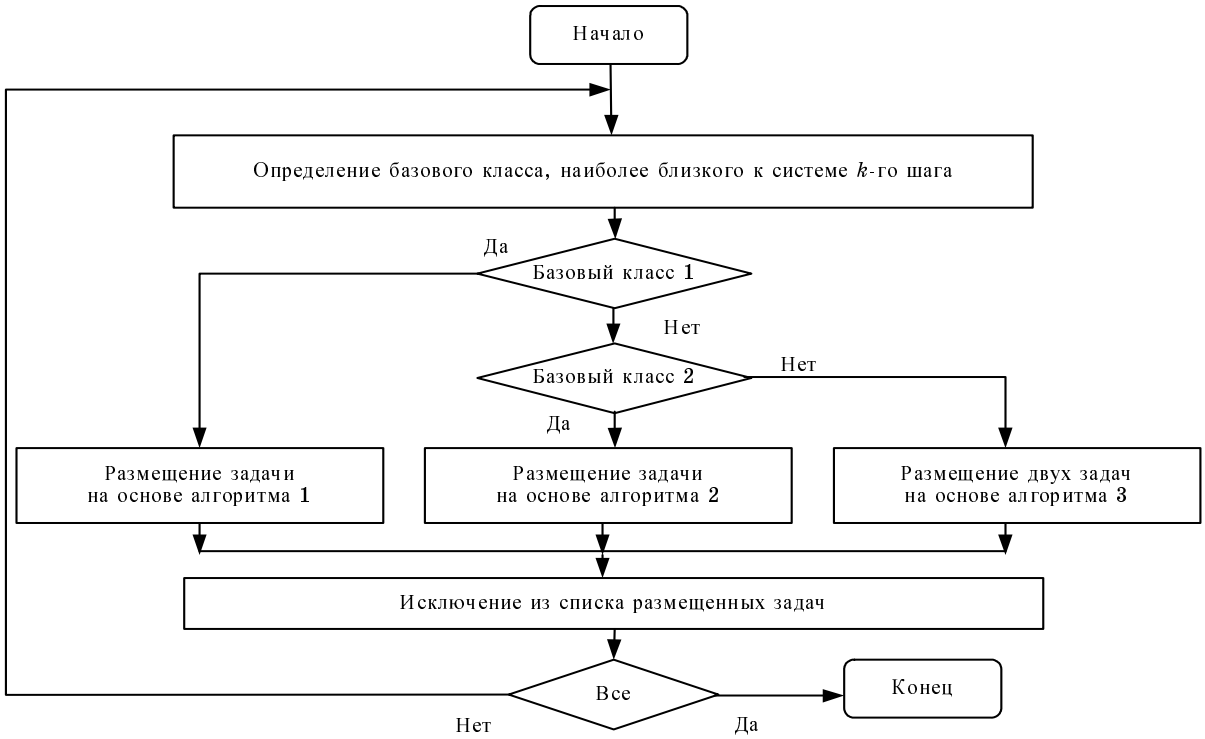
В результате для $T_3(\pi)$ получаем

$$T_3(\pi)=\sum_{i=1}^{h^*}\tau^*_{i,1}+\sum_{j=2}^{n-1}\tau^*_{h^*,j}+\sum_{i=h^*}^{m^*}\tau^*_{i,n}.$$

Перенося для удобства анализа последнее слагаемое из первой суммы и первое слагаемое из третьей суммы во вторую, получаем

$$T_3(\pi)=\sum_{i=1}^{h^*-1}\tau^*_{i,1}+\sum_{j=1}^{n}\tau^*_{h^*,j}+\sum_{i=h^*+1}^{m^*}\tau^*_{i,n}.$$

Значение второй суммы представляет собой суммарное время, затрачиваемое на решение соответствующих фрагментов всех задач на h^*-м процессоре. Очевидно, что для заданной очереди ее величина фиксирована и не зависит от выбора расписания. В отличие от второй суммы, первая и третья суммы зависят от выбора расписания. Причем оптимальным будет то расписание, которое характеризуется минимальным значением этой части выражения. Минимальность значения, в свою очередь, будет достигаться, если в расписании первой будет решаться задача, требующая наименьшего времени для первых (h^*-1) фрагментов, выполняемых на процессорах критического пути, а последней – задача, наиболее быстро решаемая на последних (m^*-h^*) процессорах критического пути. Поскольку одна и та же задача может удовлетворять обоим требованиям, ее надо попробовать разместить на каждой из этих позиций, а затем выбрать наилучшее из этих двух расписаний. Отсюда следует, что расписание, формируемое в алго-



■ ***Рис. 4.*** *Блок-схема субоптимального алгоритма построения расписаний*

ритме 3, является оптимальным для системы из класса 3.

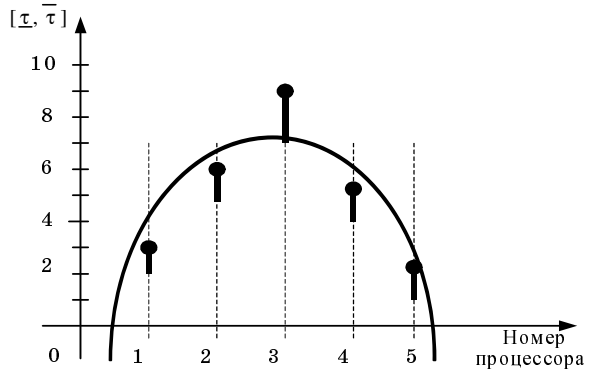
Отметим, что в данном алгоритме перебор расписаний также практически отсутствует. При построении расписания достаточно проанализировать лишь два возможных варианта, а для оптимальности расписания достаточно лишь правильного выбора первой и последней исполняемых задач. Упорядоченность же остальных задач не влияет на время исполнения расписания.

Субоптимальный алгоритм построения расписаний

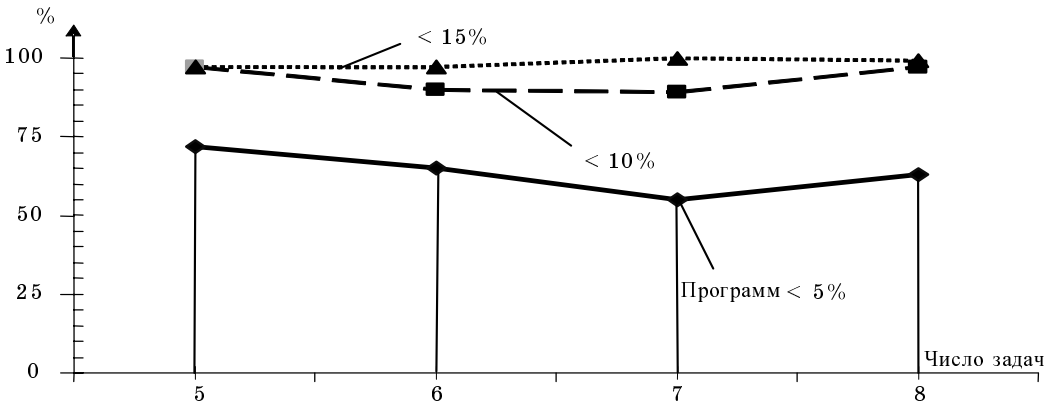
Конечно же, оптимальность приведенных выше алгоритмов имеет место лишь в случаях, когда рассматриваемая система полностью удовлетворяет требованиям какого-либо из базовых классов. На практике же чаще всего эти требования не выполняются, и тогда, в частности, утверждение о независимости времени выполнения расписания от варианта упорядочивания некрайних задач теряет силу. Напротив, представляется правдоподобным, что расхождение времен выполнения оптимального расписания и расписания с упорядочиванием лишь крайних задач будет тем меньше, чем меньше время выполнения расписания для всех некрайних задач. Таким образом, приходим к следующему рекурсивному алгоритму (рис. 4).

На первом шаге алгоритма для системы, характеризующейся множеством исполняемых задач мощностью n, на основе эвристического классификационного правила определяется базовый класс систем, к которому наиболее близка рассматриваемая система. Далее определяется одна из крайних (первая или последняя) задач будущего расписания (классы 1 и 2) либо обе крайние задачи (класс 3). Затем эти действия повторяются для $(n-1)$ задач (классы 1 и 2) или для $(n-2)$ задач (класс 3) и т. д. до тех пор, пока все задачи не будут размещены в расписании.

Классификация осуществляется путем выделения у рассматриваемой системы определяющего



■ ***Рис. 5.*** *Иллюстрация правила классификации систем*



■ ***Рис. 6.*** *Эффективность субоптимального алгоритма*

признака – наличия либо убывающей последовательности, либо возрастающей последовательности, либо состыкованных убывающей и возрастающей последовательностей. Значение определяющего признака вычисляется с использованием аппроксимации параболой верхних границ интервальной зависимости времени выполнения всех задач от номера процессора (рис. 5). На рисунке интервалы времен показаны отрезками жирных линий, а их верхние границы – черными кружками. При этом, если вершина параболы оказывается слева от интервала номеров процессоров, то система относится к классу 1, если справа,– то к классу 2, если внутри интервала,– то к классу 3.

Оценка эффективности приближенного алгоритма

Оценка эффективности предложенного алгоритма осуществлялась на основе компьютерного моделирования и в целях упрощения производилась авторами по отношению к классу конвейерных систем, составляющих важный подкласс иерархических систем. В основу использованного подхода было положено случайное генерирование примеров. При этом фиксировалось число процессоров ($m = 10$), образующих конвейер, а число задач варьировалось в интервале от 5 до 8. Для каждого значения n (число задач) из этого интервала генерировалось 100 примеров. Причем времена решения задач на каждом из процессоров формировались как случайные величины, равномерно распределенные на заданном интервале. Для каждого получаемого примера строилось расписание на основе предложенного алгоритма и путем полного перебора (оптимальное расписание). Эти расписания сравнивались между собой по относительной разности времен выполнения. Результаты моделирования приведены на рис. 6.

Результаты показывают, что, например, при $n = 5$ предлагаемый алгоритм примерно в 75% случаев строил расписания, которые уступали по длительности оптимальному расписанию не более 5%, а практически для всех примеров были построены расписания, уступающие оптимальному не более 10 %. Самым неблагополучным оказался случай при $n = 7$, когда лишь для 59% примеров были построены расписания, уступающие оптимальному не более 5%, а для 90% примеров были построены расписания, уступающие оптимальному не более 10%.

Заключение

В настоящей статье рассмотрен субоптимальный рекурсивный алгоритм составления расписаний в иерархических вычислительных системах, практически не требующий перебора вариантов и вследствие этого обладающий низкой временной сложностью. В его основе лежат несколько беспереборных оптимальных алгоритмов построения расписаний, каждый из которых ориентирован на некоторый свой в общем случае достаточно узкий базовый класс систем. Проведенное исследование эффективности алгоритма показало, что при его использовании в 90% случаях строится расписание, уступающее по длительности выполнения оптимальному расписанию не более 10%.

Литература

1. Теория расписаний и вычислительные машины: Пер. с англ. / Под ред. Э. Г. Коффмана. М.: Наука, 1984. 334 с.
2. **Левин В. И.** Структурно-логические методы исследования сложных систем с применением ЭВМ. М.: Наука, 1987. 304 с.
3. **Топорков В. В.** Модели распределенных вычислений. М.: Физматлит, 2004. 320 с.
4. **Колесов Н. В., Толмачева М. В.** Составление расписаний решения задач в конвейерных вычислительных системах // Информационно-управляющие системы. 2005. № 5. С. 16–21.

УДК 681.324:681.3.001.57

ВИРТУАЛЬНЫЕ МАШИНЫ И СЕТИ

А. В. Гордеев,
доктор техн. наук, профессор
Санкт-Петербургский государственный университет аэрокосмического приборостроения

В статье кратко описываются наиболее распространенные средства моделирования персональных компьютеров и локальных сетей на их основе, известные как виртуальные машины. В результате сравнительного анализа средств моделирования излагаются основные их возможности, достоинства и недостатки. Приведены примеры использования виртуальных машин и сетей как в учебном процессе, так и в реальной практике построения вычислительных сетей.

We describe most popular tools of emulations of PCs and networks, known as virtual machines. There are very few publications on this topic in Russian. As a result of comparison of virtual machines, the author talks about their basic features, advantages and disadvantages. The article narrates about examples of using virtual PCs and virtual networks both in the educational aspect and in the case of real-life networks.

Успехи в построении компьютерных моделей разных объектов и систем привели к появлению виртуальных объектов и миров. Одним из таких объектов, который может быть построен с помощью компьютера, является сам компьютер. Другими словами, речь идет о том, что посредством специально созданного программного обеспечения мы на компьютере можем достаточно реально моделировать работу другого компьютера. В этом случае говорят о виртуальных машинах. А если запустить на одном или нескольких компьютерах несколько виртуальных машин и обеспечить возможность взаимодействовать этим виртуальным машинам посредством сетевых технологий, то можно говорить и о виртуальных сетях. К сожалению, на эту тему совсем немного публикаций [1, 2]. Наибольшим быстродействием обладают те виртуальные компьютеры, архитектура которых соответствует архитектуре компьютеров, на которых и организуется моделирование. Само собой, что полноценное и эффективное функционирование виртуальных машин возможно только на тех компьютерах, аппаратные средства которых позволяют организовать независимые защищенные вычисления.

В настоящее время существует несколько программных продуктов, которые позволяют получить на обычных персональных компьютерах полноценные виртуальные вычислительные машины. Мы ограничимся рассмотрением всего двух[1], но, пожалуй, самых известных – это VMware Workstation и Microsoft Virtual PC. Кроме этих программ, которые организуют функционирование модели компьютера непосредственно на том компьютере, на котором работает пользователь, имеются программные средства, позволяющие создавать виртуальные машины и работать с ними по технологии клиент-сервер. Наиболее известными являются VMware GSX Server и VMware ESX Server. Эти программные средства позволяют создавать и запускать виртуальные машины на сервере. Работа с виртуальными машинами в этом случае осуществляется через консольное приложение. Такая работа напоминает работу с терминальным сервером.

Тот компьютер с операционной системой, в среде которой функционирует запускаемая виртуальная машина, называется основным, или главным (в своей документации компания VMware Inc. употребляет термин host). Та операционная система, которая устанавливается на виртуальный компьютер, называется гостевой (guest VM).

И VMware Workstation, и Microsoft Virtual PC предназначены для построения виртуальных машин, работающих под управлением широко распространенных операционных систем. Так, например, VMware Workstation 4.5.2 позволяет установить на виртуальный компьютер следующие гостевые операционные системы:

- простейшие однопрограммные DOS-системы типа Microsoft MS-DOS 6.x;
- так называемые DOS-based Windows, т. е. операционные системы Windows, использующие DOS для своей загрузки. Это и уже всеми забытые 16-раз-

[1] Помимо этих программ, можно также упомянуть программу twoOStwo и некоторые другие, однако моделирующие возможности, надежность и производительность остальных средств существенно ниже.

рядные Windows 3.x, и более современные 32-разрядные Windows 95 / 98 / 98 SE / ME;

• наиболее мощные и совершенные в плане своей архитектуры и используемой модели защиты системы Microsoft поколения New Technology – Windows NT 4.0 (Workstation и Server 4.0, в том числе и Terminal Server Edition), системы семейства Windows 2000 (Professional, Server и Advanced Server), системы Windows XP (Professional и Home Edition), серверные системы Windows Server 2003 (Web Edition, Standard Edition и Enterprise Edition);

• известные сетевые системы Novell Netware 5.1 и Novell Netware 6;

• системы класса UNIX – Red Hat Linux 7.0, 7.1, 7.2, 7.3, 8.0, Linux Mandrake 8.2, 9.0, SuSe Linux 7.3, SuSe Linux Enterprise Server 7.0, 8.0, 8.1 ,Turbolinux Server 7.0, 8.0, Turbolinux Workstation 8.0, FreeBSD 4.0-4.6.2, 4.8.

Работа вышеперечисленных систем была всесторонне протестирована компанией VMware, и при создании новой виртуальной машины пользователю предлагается именно такой список из возможных гостевых систем. Кроме этого, как показывает практика, достаточно успешно в роли гостевых работают и другие операционные системы, не вошедшие в него. Так, например, можно создать даже бездисковую машину и работать с операционной системой, которая будет загружаться либо с компакт-диска, либо с его образа. В качестве такого диска при обучении студентов мы часто используем компакт-диск, на котором имеется и Windows XPE, и Linux Life CD (Linux BCS Linux).

Microsoft Virtual PC 2004 также позволяет работать со следующими наиболее распространенными клиентскими операционными системами: MSDOS, Windows 95, Windows 98, Windows Millennium Edition, Windows NT Workstation, Windows 2000 (Professional), Windows XP (Home Edition и Professional), OS/2 (разных версий, в том числе и OS/2 Warp). Из серверных операционных систем в диалоговом окне выбора инсталлируемой виртуальной машины упомянуты Windows NT Server, Windows 2000 Server, Windows Server 2003. Кроме этого, имеется пункт Other, что означает возможность установки и других операционных систем.

Первое важное различие, которое имеется между VMware Workstation и Microsoft Virtual PC, заключается в том, что для VMware есть реализации, когда основной компьютер работает под Linux, и есть реализации, работающие в среде Win32API. Для Microsoft Virtual PC нет реализаций, работающих на основной машине с системами Linux.

Второе различие в этих программных продуктах, которое, безусловно, обращает на себя внимание, заключается в том виртуальном оборудовании, которое может быть включено в состав виртуального компьютера. Обе они основаны на хорошо известном чипсете от Intel – 440BX – и центральном процессоре с архитектурой ia32, частота которого соответствует частоте процессора основного компьютера. Таким образом, платформа реального и виртуального компьютеров могут не совпадать. Например, реальный компьютер может быть собран на базе процессора AMD Athlon и материнской платы с чипсетом NVidia NForce, в то время как виртуальный компьютер будет иметь процессор Intel Pentium и материнскую плату на чипсете Intel 440BX. Правда, работать процессор виртуального компьютера будет на частоте реального процессора. Более того, некоторые тестирующие программы, непосредственно опрашивающие сигнатуру процессора (например, известная многим программа AIDA32), укажут на процессор, реально установленный на host-машине.

Итак, первое главное отличие между основной и гостевой машинами начинается с контроллера памяти и контроллера ввода/вывода и может продолжиться наличием или отсутствием SCSI-контроллеров, числом и типом накопителей на жестких дисках, сетевыми адаптерами и другим периферийным оборудованием. Отличие виртуального процессора от реального заключается, в частности, в том, что виртуальный может иметь коэффициент умножения внутренней частоты процессора по отношению к частоте FSB, который не соответствует реальному процессору. В результате некоторые тестовые программы могут выдавать некорректную информацию о быстродействии памяти. Объем оперативной памяти, который указывается при создании и последующем конфигурировании виртуальной машины, зависит от объема реальной памяти. Объем оперативной памяти виртуальной машины не может быть больше, чем объем памяти в основном (реальном) компьютере. Остальные компоненты у VMware Workstation и Microsoft Virtual PC – разные.

Так, например, машины VMware могут включать в свой состав следующее оборудование:

• сетевые адаптеры AMD PCnet-PCI II (всего до трех адаптеров), которые работают по технологии Ethernet, т. е. скорость обмена данными по сети составляет не более 10 Мб/с;

• жесткие диски с интерфейсом IDE. Как и в настоящем компьютере, может быть до четырех устройств, подключенных через этот интерфейс, в том числе и приводы CD-ROM, один из которых будет входить в состав виртуальной машины по умолчанию, если это устройство имеется на основном компьютере. Следует заметить, что если на реальном компьютере нет привода CD-ROM, то на виртуальном он может быть, но в этом случае он может работать только с виртуальными компакт-дисками, т. е. с так называемыми образами дисков. Поскольку образы компакт-дисков представлены файлами на реальных жестких дисках (а их скорость, как известно, существенно выше, нежели у оптических приводов), то некоторые задачи, которые связаны с работой приводов CD-ROM, на виртуальных машинах могут быть выполнены с существенно более высокой скоростью, чем на ре-

альных. Последнее справедливо и в некоторых других случаях, и об этом будет изложено более подробно;

• контроллеры SCSI (BusLogic или LSI Logic), посредством которых можно работать с соответствующими SCSI-устройствами. Этими устройствами могут быть и жесткие диски, и приводы CD-ROM. Интерфейс SCSI в соответствии с известной спецификацией поддерживает работу до семи SCSI-устройств. Эти устройства могут быть виртуальными либо могут быть задействованы реально существующие. Безусловно, реальные жесткие диски работают намного быстрее виртуальных. Однако для большей безопасности рекомендуется для виртуальных машин использовать отдельные жесткие диски, не задействованные основным компьютером;

• VMware Workstation 4.5 допускает, что при наличии в основном (реальном) компьютере 4 Гбайт памяти, под виртуальный компьютер может быть отведено до 3,6 Гбайт оперативной памяти.

• дисковод (Floppy Drive). Его наличие или отсутствие в виртуальном компьютере связано с тем, имеется ли такое устройство в основной машине. Если нет, то дисковод по умолчанию не будет включен. Если же его добавить в виртуальный компьютер, то это устройство должно быть либо связано с реально существующим дисководом, либо представлено образом тех данных, которые могут быть считаны с этого виртуального дисковода, либо записываются на некую виртуальную дискету.

Виртуальные машины, создаваемые в среде Microsoft Virtual PC 2004, имеют следующие особенности:

• объем памяти виртуальной машины может определяться с шагом 1 Мб (а не с шагом 4 Мб, как у VMware Workstation), но не более чем объем памяти основной машины за вычетом объема, необходимого основной операционной системе и самой программе Microsoft Virtual PC;

• имеется явное ограничение на количество виртуальных жестких дисков – их может быть не более трех. Это ограничение может стать препятствием при моделировании и изучении RAID-массивов;

• в качестве виртуальных сетевых адаптеров могут выступать только те реальные, что установлены на основной машине. Максимальное количество их не может превышать четырех. В дополнение к этому следует отметить, что Microsoft Virtual PC не имеет механизма конфигурирования виртуальных сетей, в то время как в VMware Workstation имеется несколько очень важных настроек и обеспечивается возможность построения до десяти виртуальных подсетей.

К сожалению, мультипроцессорную систему промоделировать нельзя ни в VMware Workstation, ни в Microsoft Virtual PC, хотя чипсет 440BX и допускает возможность построения компьютеров с двумя процессорами. Двухпроцессорные SMP-системы на этом чипсете в свое время были не редкостью, и было бы интересно получить возможность исследовать работу SMP-систем.

Наконец, нельзя не заметить различие в быстродействии виртуальных машин. Виртуальные машины VMware Workstation работают в два раза быстрее, нежели машины Microsoft Virtual PC. В подтверждение сказанному можно привести следующий пример. Процесс инсталляции операционной системы Microsoft Windows Millennium Edition в среде VMware Workstation 4.0 занял 55 мин, в то время как инсталляция этой же системы в среде Microsoft Virtual PC потребовал 1 ч и 45 мин. Основной компьютер работал под операционной системой Microsoft Windows XP SP2 и имел следующую конфигурацию: материнская плата на чипсете Intel 440BX, процессор iP-III 800MHz, оперативная память PC133 объемом 512 Мбайт, два накопителя на жестких дисках. Винчестер, на котором создавались виртуальные машины, – это достаточно быстродействующий даже по сегодняшним меркам IC35L120AVVA207, причем файлы виртуальных машин по очереди создавались в одном и том же разделе. Этот раздел был специально отведен для экспериментов и располагался на внешних цилиндрах, что обеспечивает максимальное быстродействие дисковой подсистемы. Необходимо отметить, что наибольшее замедление в работу виртуальных машин вносит именно дисковая подсистема. Использование даже простейшего контроллера с RAID-0 на основной машине позволяет заметно ускорить работу виртуальных машин.

Итак, можно констатировать, что предпочтительнее использовать именно продукты компании VMware. Более того, у компании VMware есть программные продукты, позволяющие использовать идеологию клиент-сервер и запускать виртуальные машины на мощных многопроцессорных серверах с большим объемом оперативной памяти и высокой производительностью. Как правило, такие серверы имеют быструю дисковую подсистему, основанную на технологиях RAID-массивов. Одной из таких программ является уже упомянутый VMware GSX Server 3.1. При наличии нескольких серверов с VMware GSX Server ими можно управлять с помощью специальных средств. Одним из таких средств является VMware Virtual Center, который позволяет с одного места управлять тысячами виртуальных машин. Дополнительно можно использовать такое сильное средство, как VMware ACE. Эти средства позволяют клонировать машины, собирать информацию о потребляемых вычислительных ресурсах, останавливать и запускать те или иные виртуальные машины, управлять виртуальными сетями, создавать и использовать специальные политики, которые помогают администрировать. В результате администраторы могут более эффективно и строго управлять информационными и вычислительными ресурсами, которые предоставляются пользователям посредством виртуальных машин.

Замечательным является то, что VMware GSX Server позволяет работать с виртуальными маши-

нами даже с маломощных рабочих станций, поскольку последние в этом случае выступают просто как терминалы. Этот продукт, так же как и VMware Workstation, существует в двух вариантах – и под Linux, и под Windows. Он устанавливается на сервер. Интересно отметить, что на компьютер, работающий под операционной системой, имеющей статус рабочей станции, его установить нельзя.

На рабочие станции, с которых пользователи будут работать со своими виртуальными машинами, устанавливается либо VMware Workstation (если ресурсы компьютера позволяют запускать виртуальные машины), либо специальное клиентское программное обеспечение – VMware Virtual Machine Console. Оно тоже бывает в двух вариантах – и под Linux, и под Windows. Таким образом, мы можем иметь четыре возможные комбинации клиентских и серверных программ, обеспечивающих работу с виртуальными машинами. Пожалуй, наиболее распространенной является ситуация, когда VMware GSX Server работает в среде Linux, а клиенты работают в среде Windows. Однако возможна и обратная ситуация, когда VMware GSX Server работает в операционной системе Windows Server 2003, а модули клиентов работают в среде Linux. Само собой, что может быть ситуация, когда и сервер и клиент работают в среде Win32 API, и, наконец, когда все работают в среде Linux.

При использовании VMware GSX Server все моделирование работы виртуальных машин, запущенных на основной машине, осуществляется на стороне сервера. Клиенту лишь передаются графические изображения дисплея гостевой машины, полученные на сервере. В обратную сторону (от клиента к серверу) передаются данные, полученные от перемещения мыши, и данные, полученные при нажатиях на клавиатуру. Полезным является то обстоятельство, что при закрытии окна клиентского приложения, в котором была работающая виртуальная машина, последняя продолжает работать без каких-либо последствий. При повторных открытиях сеанса работы с удаленной виртуальной машиной пользователь получает экран своего виртуального монитора. Все эти действия практически полностью совпадают с той ситуацией, которая многим нынче известна по работе с терминальным сервером. Самым интересным является то, что сам терминальный сервер при использовании VMware GSX Server также может быть виртуальным.

Виртуальные сети, имеющиеся в программах компании VMware Inc., можно представить как IP-подсети, которые могут быть связаны не только с виртуальными машинами, но и с основными компьютерами. Виртуальные сетевые адаптеры виртуальных машин могут быть либо связанными с реальными сетевыми адаптерами посредством мостового соединения (Bridged-networking), либо использовать технологию трансляции сетевых адресов (NAT – Network Address Translation). В первом случае все кадры данных с реальной сетевой карты передаются на виртуальный сетевой адаптер и обратно. Это обеспечивается установкой на сетевой адаптер основной машины специального протокола (VMware Bridge Protokol). Виртуальный сетевой адаптер должен иметь свой собственный IP-адрес, отличающийся от адреса основной машины. Возможность принимать и обрабатывать информацию, приходящую на виртуальную машину через сетевое соединение, определяется настройками стека протоколов TCP/IP основной и виртуальной машины. Если они обе принадлежат одной и той же подсети или подсети, в которых они находятся, имеют соединение через маршрутизатор, то между этими машинами возможно нормальное сетевое взаимодействие. В противном случае передаваемые пакеты не смогут дойти с одной машины на другую, и сетевое взаимодействие между основной и гостевой машинами будет отсутствовать. Отметим, что, если на основной и виртуальной машинах установить ныне уже почти неиспользуемый протокол NetBEUI, работающий в немаршрутизируемых сетях, то обе машины смогут передавать между собой данные через свои сетевые интерфейсы. Этот очевидный факт хорошо демонстрирует, что при мостовом соединении кадры данных передаются с реального сетевого адаптера на виртуальный и обратно.

В случае использования трансляции сетевых адресов виртуальный адаптер виртуальной машины, имеющий внутренний или нелегальный IP-адрес, может связываться с внешними сетями. Пакеты данных, уходящие с виртуальной машины, получают IP-адрес основной машины. Настройки NAT в VMware осуществляются для виртуальных подсетей, в которых работает виртуальный DHCP-сервер.

Наконец, на виртуальной машине может быть host-only-соединение, при котором ее сетевой адаптер имеет соединение с одной из виртуальных подсетей. Если с этой же виртуальной подсетью связана еще какая-нибудь виртуальная машина или несколько виртуальных машин, то эти виртуальные машины могут полноценно использовать сетевые технологии для передачи между собой данных.

Две из десяти виртуальных сетей по умолчанию имеют DHCP-сервер. Обычно это первая и восьмая виртуальные подсети. Это позволяет иметь виртуальные машины, сконфигурированные как клиенты DHCP.

Технология виртуальных машин может использоваться для разных целей. Перечислим несколько таких применений.

Прежде всего, виртуальные машины предназначены для удобства разработчиков программных продуктов. Чтобы проверить и отладить работу создаваемых программных продуктов в той или иной среде, для которых они создаются, разработ-

чику нужно иметь под рукой эти самые среды. И вместо того, чтобы иметь несколько компьютеров, каждый из которых работает под другой операционной системой, программист имеет возможность протестировать свои решения на виртуальных машинах. Например, создаваемая программа или даже целый программный комплекс должны работать в системах Windows компании Microsoft. Как всем известно, таких операционных систем у компании Microsoft немало. Эти системы отличаются не только архитектурой, своими возможностями и графическим интерфейсом, но и интерфейсом прикладного программиста (API). Поэтому приложение, созданное для одной системы, может и не выполняться в другой системе. Иметь рядом все эти системы слишком дорого, а порой и вообще невозможно. Поэтому виртуальные машины, запускаемые программистом, позволяют ускорить и удешевить столь трудоемкий процесс тестирования и отладки.

Во-вторых, виртуальные машины используют с целью промоделировать ту или иную ситуацию, которую невозможно воссоздать в реально существующей среде, поскольку это может быть, например, небезопасно. Изучив эту ситуацию на модели из виртуальных машин (возможно, с привлечением даже нескольких реальных компьютеров), специалист может найти то правильное решение, которое без такого полунатурного моделирования и нет возможности получить. Достаточно часто виртуальные машины используются администраторами для проверки решений, которые они разрабатывают и затем реализуют в жизни, поскольку это позволяет избежать серьезных неприятностей в работе реальной сети.

В-третьих, виртуальные машины используют с целью запуска программ, работающих в операционной среде, которой нет на основном компьютере. И для того, чтобы иметь столь необходимую операционную среду, которой нет в основной операционной системе, как раз и создается и запускается виртуальная машина.

В-четвертых, виртуальные машины можно (и нужно!) использовать в учебном процессе. Очевидно, что виртуальные машины с большим успехом могут быть применены для изучения разных операционных систем. Кроме этого, их можно использовать и для изучения основ построения, конфигурирования и администрирования локальных вычислительных сетей, которые в подавляющем большинстве случаев теперь строятся на персональных компьютерах. Для этого в программные средства VMware и были включены мощные средства виртуализации вычислительных сетей.

Наконец, виртуальные машины могут с большим успехом применяться в тех случаях, когда не хватает средств на создание отдельных серверов, выполняющих те или иные функции, необходимые для нормального функционирования сети. Например, сервер, функционирующий под управлением операционной системы Linux и реализующий основные службы локальной сети, может наряду с вычислениями, выполняющимися в Linux, организовать работу одной или нескольких виртуальных машин с системами Windows. Одной из машин может быть, например, виртуальный Windows Server 2000, выполняющий роль контроллера домена. Вторая виртуальная машина может работать в роли сервера терминалов, что дает возможность с маломощных рабочих станций запускать на выполнение ресурсоемкие приложения. Часто такая виртуальная машина работает под управлением системы Windows Server 2003, поскольку она имеет протокол, который обеспечивает полноцветный графический режим с достаточно большим разрешением. Сервер с VMware GSX Server, работающий под Linux, обеспечивает выход в Интернет и надежную защиту всей сети, работу WWW- и FTP-сервисов, отправку и получение электронной почты, файловый SMB-сервис и некоторые другие службы. Кроме этого, естественно, у пользователей остается возможность при наличии клиента VMware Virtual Client Console работать со своими виртуальными машинами.

Эта последняя область применения начинает в последнее время быть столь значимой, что некоторые компании, в том числе и Microsoft, стали разрабатывать системы, которые сразу (без установки дополнительного программного обеспечения) имеют возможность запускать на одном реальном компьютере несколько операционных систем. И даже архитектура процессоров изменяется таким образом, чтобы можно было более эффективно организовать на них одновременную работу нескольких операционных систем, причем, в каждой из них может выполняться параллельно несколько задач.

Следует отметить следующие интересные факты. Виртуальные Windows-машины часто работают существенно быстрее, чем когда они запускаются как основные. Казалось бы, такого не должно быть, поскольку при виртуализации мы должны терять в производительности. Однако объясняется этот любопытный факт тем, что основная машина (при наличии достаточного объема оперативной памяти) кэширует в своей памяти те файлы, которые для виртуальных машин воспринимаются как виртуальные диски. И поскольку Linux делает кэширование файлов достаточно эффективно, то Windows при обращении к своим виртуальным дискам начинает работать в несколько раз быстрее. Проведенные эксперименты показывают, что при работе с большим количеством файлов виртуальная машина Windows может ускориться даже в десять (и более!) раз по сравнению с тем решением, когда она устанавливается как основная система.

Автор, работая в Санкт-Петербургском государственном университете аэрокосмического приборостроения, уже достаточно длительное время использует виртуальные машины и сети для обу-

чения студентов. Использование этой технологии позволяет организовать работу студентов таким образом, что каждый из них получает статус администратора всех своих виртуальных машин и своей локальной сети, которую студенты строят в рамках курсового проектирования. Очевидно, что без виртуальных машин невозможно обеспечить каждому студенту статус администратора, тем более, что эти же компьютеры используются не только в учебном процессе. На них могут работать и преподаватели, и сотрудники кафедры, причем для решения самых разнообразных задач. Большое удобство виртуальных машин заключается в том, что любую работающую виртуальную машину можно в любой момент отправить в состояние «сна» (suspend). При этом программа моделирования компьютера сохраняет в специальных файлах состояние виртуальной машины. При «пробуждении» машина продолжает свою работу с того же места. Получается, что по окончании занятий студенты могут быстро сохранить свою работу и продолжить ее, например, на следующей неделе.

Опыт испльзования виртуальных машин и сетей нашел отражение в работе [3].

Литература

1. **Бешков А.** VMware – виртуальный полигон для администратора и разработчика // Системный администратор. 2003. № 9. С. 8–13.
2. **Ерижоков А. А.** VMware – одновременная работа с несколькими операционными системами. http://docs.luksian.com/hardware/emulator/vmware/
3. **Балберин В. В., Гордеев А. В., Лавров В. Э.** Работа с VMware Workstation и VMware Server. Спб.: Институт дополнительного профессионального образования, 2005. 52 с.

Коваленко В. В.
Частично инфинитное моделирование (основания, примеры, парадоксы). – СПб.: Политехника, 2005. – 483 с.
ISBN 5-7325-0337-4.

В книге достаточно полно излагаются основы частично инфинитного моделирования. На примерах из истории, географии и механики жидкости показывается технология, слабые и сильные стороны рассматриваемого подхода к моделированию развивающихся систем. Формируется логико-негеоцентрический парадокс, исключающий претензии на самодостаточность частично инфинитного моделирования.

Предназначена специалистам, аспирантам и студентам вузов, занимающихся методологией науки и моделированием. Не рекомендуется любителям доказывать теоремы и ученым, склонным к обстоятельности и серьезности (которым «не до шуток»).

Повопросам приобретения книги обращаться по адресу:
191023, г. Санкт-Петербург, ул. Инженерная, д. 6, 3-й этаж, ОАО «Издательство “Политехника”»
телефон/факс: **312-44-95** (отдел реализации)
e-mail: lara_politehnika@hotbox.ru, olesya_politehnika@hotbox.ru,
www.polytechnics.ru

УДК 621.391

ИСПОЛЬЗОВАНИЕ АДРЕСОВ АБОНЕНТОВ ДЛЯ РАЗРЕШЕНИЯ КОНФЛИКТОВ В КАНАЛЕ С ШУМОМ

С. Г. Марковский,
старший преподаватель
А. М. Тюрликов,
канд. техн. наук, доцент
Санкт-Петербургский государственный университет аэрокосмического приборостроения

Рассматриваются алгоритмы случайного множественного доступа, использующие адреса абонентов для разрешения конфликтов в системе с конечным числом абонентов. Показывается, что при определенной модификации данные алгоритмы работоспособны в канале с шумом, приводящим к возникновению ложных конфликтов. Приведена методика определения скорости и средней задержки передачи сообщения в канале с ложными конфликтами.

The paper considers random multiple access algorithms that use subscribers addresses for collisions resolution in finite system. With some modifications, these algorithms are able to work in the channel with noise, false collisions appearing. A method of algorithm rate and average delay definition in the channel with false collision is presented.

Введение

Случайный множественный доступ (СМД) широко используется для организации доступа множества абонентов к общему каналу связи как в существующих, так и в перспективных системах. Примерами таких систем являются организация передачи данных в стандартах 802.3 и 802.11 [1], резервирование времени при пакетной передаче данных (GPRS) в стандарте GSM. В разрабатываемом в настоящее время стандарте 802.16 случайный доступ планируется использовать для резервирования времени при передаче данных от абонентов к базовой станции [2]. Во всех этих системах при организации доступа используются чисто случайные механизмы, не опирающиеся на адреса абонентов. Собственно способ использования адресов для разрешения конфликтов был предложен более 25 лет назад, и была показана его эффективность по сравнению с чисто случайными механизмами [3, 4].

В большинстве работ, посвященных этой тематике, предполагается отсутствие ошибок в канале связи, когда все абоненты безошибочно определяют ситуацию в канале. Это предположение, являющееся одним из допущений базовой (идеализированной) модели СМД [3, 4], не всегда справедливо для реальных каналов связи. Наличие шума в канале может приводить к ошибкам в распознавании ситуации в канале.

Для стандартного древовидного алгоритма разрешения конфликта (АРК) [3, 4] модель канала с шумом впервые рассматривалась Г. С. Евсеевым и Н. Г. Ермолаевым [5]. Авторы исследовали характеристики АРК со случайным выбором адресов (так называемые случайные паспорта [3]) в канале с ложными конфликтами и установили, что модифицированный древовидный алгоритм неработоспособен в канале с шумом. Кроме того, они отметили, что в канале с шумом непосредственно использовать адреса абонентов для разрешения конфликтов нельзя. Возможно, это и явилось одной из причин того, что такой подход не нашел применения на практике. Единственно известной системой, в которой используется близкий к приведенному в работе [4] подход, является так называемый CAN-интерфейс [6].

В настоящей статье описывается вариант, позволяющий применять адреса абонентов при разрешении конфликтов в канале с шумом, и предлагается методика анализа характеристик алгоритмов, использующих такой вариант. При этом алгоритмы интерпретируются в виде стека и в работе называются немодифицированный и модифицированный блокированный стек-алгоритмы. Показывается, что оба алгоритма работоспособны в канале с ложными конфликтами.

Описание модели системы

Рассмотрим систему множественного доступа с конечным числом абонентов M, связанных об-

щим каналом связи. Подобная модель была рассмотрена в статье [4] для случая бесшумного канала. Абонентам присваиваются конкретные физические адреса в двоичной системе счисления из диапазона 0, 1, ..., $M-1$ (так называемые фиксированные паспорта [3]), которые используются для разрешения конфликтов. Для простоты будем считать, что $M = 2^l$, где l – разрядность адреса.

Положим, что все сообщения имеют одинаковую длину, поэтому сообщение будем называть *пакетом*. Длительность передачи пакета по общему каналу принимается за единицу времени. Абонентам разрешается начинать передачу пакета только в моменты времени $t = 0, 1, 2, ...$ (так называемый синхронный доступ). Единичный интервал времени $(t, t + 1)$ называется *окном* t. В системе имеется обратная связь, посредством которой каждый из абонентов узнает о состоянии общего канала. В каждом окне канал находится в одном из трех возможных состояний: пустое окно П (пакет не передавался, т. е. канал свободен), успешная передача У (передавался только один пакет) и конфликт К (одновременно передавались два и более пакетов).

Наличие шумов в прямом канале приводит к возможности ошибочного определения абонентами состояния канала (появлению ложных конфликтов). С вероятностью q_0 абонент принимает пустое окно за конфликт, а с вероятностью q_1 он воспринимает успешную передачу как конфликт. Считается, что все абоненты одинаково принимают решение о состоянии канала, т. е. либо все абоненты правильно оценивают ситуацию, либо все одновременно ошибаются.

Число пакетов, вступающих в конфликт, называется *кратностью конфликта*. Ложные конфликты имеют кратность 0 или 1.

Каждый абонент имеет буфер, состоящий из двух ячеек (ячейка IN и ячейка OUT). Пакеты могут поступать к абоненту в ячейку IN, а передаваться в канал из ячейки OUT, т. е. в каждый момент времени у абонента может находиться не более двух пакетов.

Подробное описание двухбуферной модели (вероятностная модель поступления пакетов и правило передачи пакетов в канал) приведено в статье [7].

Алгоритмы доступа для канала без шума

В данном разделе приведены инструкции для блокированного немодифицированного стек-алгоритма. Подобные инструкции для модифицированного стек-алгоритма рассмотрены в работе [7] и здесь не приводятся. Данные алгоритмы работоспособны при отсутствии шумов в канале связи и рассматривались ранее в работе [3], но в несколько иной интерпретации и применительно к другой модели системы.

Для описания алгоритмов введем понятие сеанса. Сеанс – это последовательность окон, конец которой определяется при помощи дополнительной целочисленной переменной, называемой *меткой* сеанса и обозначаемой $h(t)$. Первый сеанс начинается в момент времени $t = 0$, когда система начинает работать. Если окно t пустое или в нем передавался один пакет, то сеанс имеет длину 1 и в момент времени $t + 1$ начинается следующий сеанс. В случае, если окно t – первое окно сеанса и в нем происходит конфликт определенной кратности, то для определения момента окончания текущего сеанса и начала следующего используется метка сеанса $h(t)$. Сеанс заканчивается, если $h(t) = 0$.

Пусть $w_i(t)$ – состояние ячейки OUT i-го абонента в окне $t (w_i(t) = \{0, 1\}$, при $w_i(t) = 1$ в ячейке OUT находится пакет, при $w_i(t) = 0$ ячейка свободна). Будем считать абонент с номером i активным, если он имеет пакет для передачи, т. е. $w_i(t) = 1$. Тогда под *кратностью* сеанса будем понимать число активных абонентов в первом окне сеанса.

Разрешение конфликтов в системе производится при помощи адресов абонентов. При описании алгоритмов доступа использованы три целочисленные переменные: $SP_i(t)$ – указатель стека i-го абонента в окне t; $n_i(t)$ – номер анализируемого бита в двоичном представлении адреса абонента $a_i = a_{il}$, a_{il-1}, ..., a_{i1} в окне t; $h(t)$ – метка сеанса в окне t. Обозначим через $\eta(t) = \{П, У, К\}$ состояние канала в окне t. Пакет от i -го абонента передается в канал в окне t, если $w_i(t) = 1$, $SP_i(t) = 0$.

В начале первого окна сеанса (окна с номером t) для алгоритмов доступа задаются начальные значения переменных: если пакет появился в ячейке OUT у абонента с номером i в окне t, т. е. $w_i(t) = 1$, то $SP_i(t) = 0$, $n_i(t) = l$, $h(t) = 1$.

Инструкции немодифицированного стек-алгоритма

Инструкции для $SP_i(t)$ и $n_i(t)$

1. Если пакет передавался в окне t, т. е. $SP_i(t) = 0$, то:

• если $\eta(t) = У$, то пакет покидает стек и систему связи: $w_i(t + 1) = 0$;

• если $\eta(t) = К$, то $SP_i(t + 1) = 1 - a_i(n_i(t))$, $n_i(t + 1) = n_i(t) - 1$.

2. Если пакет не передавался в окне, т. е. $SP_i(t) >= 1$, то:

• если $\eta(t) = К$, то $SP_i(t + 1) = SP_i(t) + 1$, $n_i(t + 1) = n_i(t)$;

• если $\eta(t) = \{П, У\}$, то $SP_i(t + 1) = SP_i(t) - 1$, $n_i(t + 1) = n_i(t)$.

Инструкции для $h(t)$

1. Если $h(t) = 0$, то сеанс закончен и в системе происходят следующие события:

• пакеты, поступившие во время этого сеанса в ячейки IN абонентов, переписываются в ячейки OUT;

• ячейки IN освобождаются;

• начинается следующий сеанс.

2. Если $h(t) \geq 1$, то:
- если $\eta(t) = К$, то $h(t + 1) = h(t) + 1$;
- если $\eta(t) = \{П, У\}$, то $h(t + 1) = h(t) - 1$.

Алгоритмы разрешения конфликта можно описать, используя терминологию [3], поставив каждому сеансу во взаимно-однозначное соответствие дерево. При этом корень дерева соответствует первому окну сеанса, а концевые вершины дерева – окнам с ситуациями «пусто» и «успех». Вид дерева полностью определяется номерами активных абонентов.

Модификация алгоритмов доступа для канала с шумом

Доопределим алгоритм доступа для канала с ложными конфликтами. Для этого рассмотрим дерево, описывающее разрешение конфликта для бесшумного канала, и на основе свойств дерева определим модификацию дерева для канала с шумом.

Пусть в системе имеется $M \leq 2^l$ абонентов с адресами из множества $\{a_0, a_1, ..., a_{2^l-1}\}$. Каждый адрес a_i будем представлять в виде l-разрядного двоичного числа. Обозначим через A множество адресов M абонентов.

Введем в рассмотрение граф G_X, описывающий дерево разрешения конфликта в бесшумном канале для абонентов с адресами из множества X.

Непосредственно из определения графа G_X вытекает следующее утверждение.

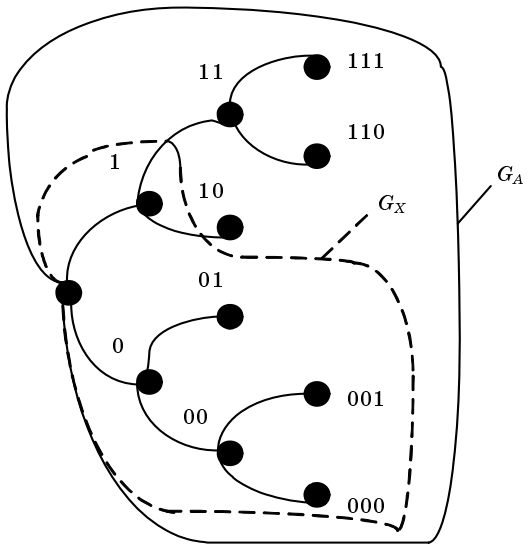
Утверждение 1. Пусть $X \subset A$, тогда дерево G_X является поддеревом дерева G_A. Концевые вершины деревьев G_A и G_X соответствуют окнам, в которых имеют место ситуации «пусто» или «успех».

Доказательство утверждения 1 основано на том, что сначала строится полное дерево разрешения конфликта для 2^l абонентов и далее, из полного дерева, путем удаления ряда вершин и дуг сначала формируется дерево G_A, а затем дерево G_X. В данной работе доказательство не приводится. На рис. 1 дан пример деревьев G_A и G_X при $A = \{000, 001, 110, 111\}$ и $X = \{000, 001, 111\}$.

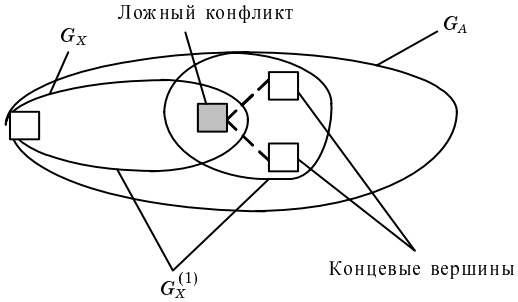
Пусть для канала с ложными конфликтами используется тот же самый алгоритм, что и для бесшумного канала, и G_A – дерево, описывающее разрешение конфликта для всех M абонентов. Доопределим алгоритм следующим образом.

Если для некоторого абонента в процессе разрешения конфликта передавался пакет в окне, которое соответствует концевой вершине дерева G_A, и пакет оказался не переданным (это происходит из-за ложных конфликтов), то данный абонент перестает участвовать в разрешении конфликта.

Пусть разрешается конфликт для абонентов из множества $X \subset A$. При отсутствии ложных конфликтов множеству X соответствует единственное дерево G_X (см. утверждение 1). Из-за наличия ложных конфликтов множеству абонентов X будут соответствовать несколько различных деревьев



■ ***Рис. 1.*** *Деревья G_A и G_X*



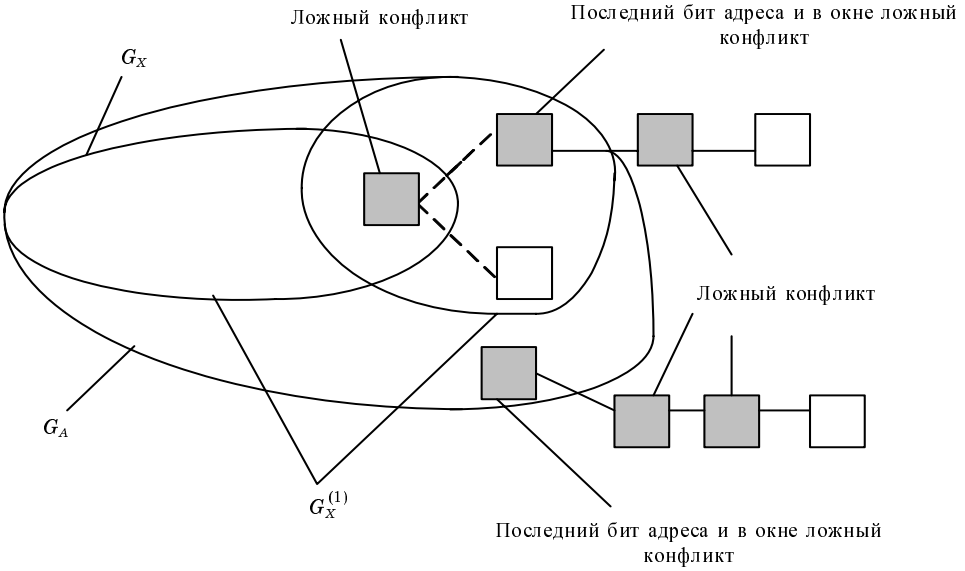
■ ***Рис. 2.*** *Деревья G_A, G_X и $G_X^{(1)}$*

разрешения конфликта. Множество таких деревьев обозначим $\Gamma_X = \left\{G_X^{(1)}, G_X^{((2)}, ...\right\}$. Справедливо следующее утверждение.

Утверждение 2. Для любого $X \subset A$ любое дерево $G_X^{(i)} \in \Gamma_X$, является поддеревом дерева G_A, и корни этих деревьев совпадают. Дерево G_X является поддеревом для любого дерева $G_X^{(i)} \in \Gamma_X$, и корни этих деревьев совпадают (рис. 2).

Аналогично утверждению 1, доказательство можно провести конструктивным способом, построив все необходимые деревья и убедившись в справедливости утверждения. Доказательство утверждения 2 в работе опускается.

Утверждение 3. Если некоторый абонент в соответствии с алгоритмом принял решение о передаче в некотором окне и после передачи определил, что в этом окне был конфликт, то данный конфликт является ложным, если данное окно является концевой вершиной дерева G_A.



■ ***Рис. 3.*** *Дерево разрешения конфликта для случая, когда в концевых вершинах возникают ложные конфликты*

Доказательство утверждения 3 непосредственно следует из построения дерева G_A. В любой концевой вершине может иметь место либо ситуация «успех», либо ситуация «пусто».

Из утверждения 3 следует, что порядок действий абонента в канале с ложными конфликтами должен быть изменен следующим образом. Для каждой вершины, в которой абонент наблюдает конфликт, определяется, является вершина концевой или нет. Если вершина *не концевая*, то продолжается работа исходного алгоритма, если вершина *концевая*, то (рис. 3):

• если абонент передавал в данном окне, то абонент повторяет свою передачу в последующих окнах до того момента, пока его пакет не будет успешно передан. После успешной передачи абонент продолжает работу по обычному алгоритму;

• если абонент не передавал в данном окне, то абонент наблюдает за выходом канала и продолжает работу по обычному алгоритму после того, как в канале будет иметь место ситуация «пусто» или «успех».

Модификация стек-алгоритмов, описанных во втором разделе настоящей статьи, заключается в следующем. Если ложный конфликт возникает в концевой вершине дерева, то абонент не изменяет величины $SP_i(t)$ и $h(t)$ до тех пор, пока в канале не будет ситуации «пусто» или «успех».

Следствием утверждения 3 является то, что в алгоритме с ложными конфликтами абонент должен уметь различать, является ли вершина дерева концевой или нет. Для определения концевой вершины дерева к абоненту добавляется дополнительный стек, который будем называть *терминальным*.

Терминальный стек

Будем предполагать, что число абонентов $M = 2^l$ в системе известно всем абонентам. Это допущение справедливо, так как в блокированном алгоритме состав системы во время функционирования изменяться не может.

Терминальный стек – это стек ограниченной глубины, которая определяется как $Dep = \log_2 M + 1$ и совпадает с числом ярусов в дереве разрешения конфликта максимальной кратности. В ячейках терминального стека могут храниться числа, соответствующие максимальному числу пакетов, которые могут находиться в вершине дерева разрешения конфликта максимальной кратности.

При инициализации системы ячейки терминального стека обнуляются, т. е. $TS(0) = TS(1) = \ldots = TS(Dep-1) = 0$, где $TS(i)$ – содержимое ячейки терминального стека с номером i.

Обозначим через $TS(i)(t)$ содержимое ячейки терминального стека с номером i в окне с номером t. Если $TS(i)(t) = 1$, то это говорит о том, что вершина дерева является концевой.

Инструкции работы с терминальным стеком

1. В начале первого окна сеанса $TS(0) = M$, (в верхнюю ячейку стека заносится число абонентов).

2. Если $\eta(t) = К$ и $TS(0)(t) \neq 1$, то:

$$TS(0)(t+1) = TS(1)(t+1) = TS(0)(t)/2;$$
$$TS(2)(t+1) = TS(1)(t);$$
$$\ldots$$
$$TS(Dep-1)(t+1) = TS(Dep-2)(t).$$

3. Если $\eta(t) = К$ и $TS(0)(t) = 1$, то вершина дерева является концевой и содержимое терминально-

го стека не изменяется до появления ситуации $\eta(t) = \{П,У\}$.

4. Если $\eta(t) = \{П, У\}$, то:

$$TS(0)(t+1) = TS(1)(t);$$
$$TS(1)(t+1) = TS(2)(t);$$
$$\ldots$$
$$TS(Dep-2)(t+1) = TS(Dep-1)(t).$$
$$TS(Dep-1)(t+1) = 0.$$

Пример 1. При $M = 8$ построим терминальный стек для немодифицированного стек-алгоритма (рис. 4) для следующего дерева разрешения конфликта (рис. 5). Диаграммы стека на рис. 4 соответствуют состоянию стека на момент начала очередного окна.

Анализируя терминальный стек, приведенный в примере, можно прийти к выводу, что после завершения последнего окна дерева разрешения конфликта (в примере – окно с номером $t + 8$) стек будет пуст.

Непосредственно из вышерассмотренного описания функционирования терминального стека вытекает следующее утверждение.

Утверждение 4. Терминальный стек может быть использован как для определения концевых вершин в дереве разрешения конфликта, так и для определения границ сеансов, т. е. реализует функцию метки сеанса.

Первые три инструкции работы с терминальным стеком модифицированного блокированного стек-алгоритма совпадают с соответствующими инструкциями немодифицированного алгоритма. Инструкция 4 видоизменяется и подразделяется на две. Кроме того, добавляется инструкция 5.

• Если $\eta(t) = \{У\}$ или $\eta(t) = \{П\}$ и $B(t) = \{У\}$, то:

$$TS(0)(t+1) = TS(1)(t);$$
$$TS(1)(t+1) = TS(2)(t);$$
$$\ldots$$
$$TS(Dep-2)(t+1) = TS(Dep-1)(t).$$
$$TS(Dep-1)(t+1) = 0.$$

• Если $\eta(t) = \{П\}$, $B(t) = \{К\}$, то:

$$TS(0)(t+1) = TS(1)(t)/2;$$
$$TS(1)(t+1) = TS(1)(t)/2,$$

где $B(t) = \{У, К\}$ – переменная, которая хранит предыдущее состояние канала, кроме состояния «пусто».

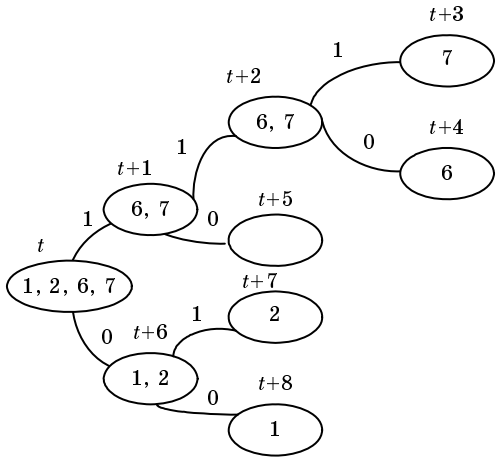
5. Если $TS(0)(t) = 1$, то $B(t) = \{У\}$.

Пример 2. При $M = 8$ построим терминальный стек для модифицированного стек-алгоритма (рис. 6) для следующего дерева разрешения конфликта (рис. 7).

Количественными характеристиками алгоритма доступа являются скорость и средняя задержка. В последующих разделах рассмотрена методика определения этих характеристик.

Окно >	t	$t+1$	$t+2$	$t+3$	$t+4$	$t+5$	$t+6$	$t+7$	$t+8$
Ячейка 0	8	4	2	1	1	2	4	2	2
Ячейка 1	0	4	2	1	2	4	0	2	0
Ячейка 2	0	0	4	2	4	0	0	0	0
Ячейка 3	0	0	0	4	0	0	0	0	0

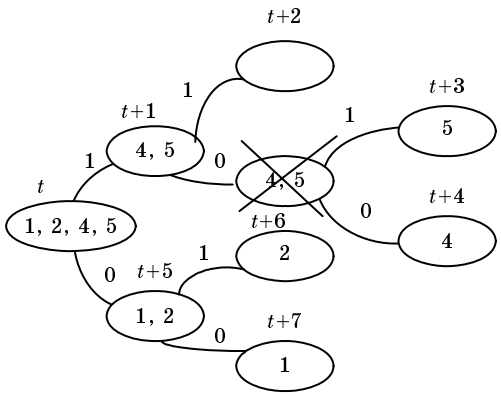
■ ***Рис. 4.*** *Терминальный стек для немодифицированного стек-алгоритма*



■ ***Рис. 5.*** *Дерево разрешения конфликта для немодифицированного стек-алгоритма*

Окно >	t	$t+1$	$t+2$		$t+3$	$t+4$	$t+5$	$t+6$	$t+7$
Ячейка 0	8	4	2	2	1	1	4	2	2
Ячейка 1	0	4	2	1	1	4	0	2	0
Ячейка 2	0	0	4	0	4	0	0	0	0
Ячейка 3	0	0	0	0	0	0	0	0	0

■ ***Рис. 6.*** *Терминальный стек для модифицированного стек-алгоритма*



■ ***Рис. 7.*** *Дерево разрешения конфликта для модифицированного стек-алгоритма*

Скорость алгоритма доступа

Понятие скорости алгоритма доступа для системы с конечным числом абонентов для случая, когда каждый абонент имеет бесконечную очередь, было не явно введено Б. С. Цыбаковым и В. А. Михайловым в работе [8]. Согласно этому определению, скорость – это максимальная интенсивность входного потока, при которой очереди у абонентов конечны.

Для двухбуферной модели понятие скорости было предложено Капетанакисом [4]. Предполагается, что все абоненты системы имеют готовый для передачи пакет. Тогда скорость алгоритма доступа – это отношение числа пакетов, переданных в сеансе кратности M, к длительности сеанса:

$$R = \frac{M}{2M-1} = \frac{2^l}{2^{l+1}-1} = \frac{1}{2-2^{-l}}. \qquad (1)$$

Можно показать, что это определение скорости совпадает с определением, предложенным в работе [8].

Для канала с ложными конфликтами скорость можно вычислить следующим образом:

$$R = \frac{M}{M-1+MT_{1,0}} = \frac{M}{M-1+M\dfrac{1}{1-q_1}} =$$

$$= \frac{2^l}{2^l-1+2^l\dfrac{1}{1-q_1}} = \frac{1}{1-2^{-l}+\dfrac{1}{1-q_1}} =$$

$$= \frac{1-q_1}{2-2^{-l}(1-q_1)-q_1}, \qquad (2)$$

где $T_{1,0}$ – среднее число окон до наступления ситуации «успех» при возникновении в концевой вершине ложного конфликта кратности 1.

Можно показать, что предельное значение скорости при $M \to \infty$ или, по-другому, при $l \to \infty$, для канала без шума $\dfrac{1}{2}$, а для канала с ложными конфликтами $\dfrac{1-q_1}{2-q_1}$.

Расчет средней задержки

Задержкой передачи пакета называется время от момента его поступления в систему до момента его успешной передачи. Занумеруем числовой последовательностью все поступающие в систему пакеты и выделим из этой последовательности пакет с номером i. Этот пакет мы назовем *меченым* и найдем для него среднюю задержку.

Обозначим через δ_i случайную задержку передачи меченого пакета. Определим среднюю стационарную задержку передачи пакета равенством

$$D = \lim_{i\to\infty} M\delta_i. \qquad (3)$$

Методика расчета средней задержки для канала без шума подробно рассмотрена в работе [7]. При вычислении средней задержки некоторые величины зависят от интенсивности входного потока λ, а другие не зависят (средняя длина сеанса, среднее время выхода и распределение вероятностей кратностей сеанса по длинам сеанса). Величины, не зависящие от λ, рассчитываются по рекуррентным формулам. Так как ложные конфликты приводят к модификации дерева разрешения конфликта, то необходимо определить рекуррентные формулы для вычисления этих величин в канале с шумом. Заметим, что все эти величины представляют собой функции вероятностей ложных конфликтов q_0 и q_1. Однако в дальнейшем с целью сокращения записи выражений эти параметры опускаются. Далее, используя методику [7], определяем среднюю задержку пакета для канала с ложными конфликтами.

Средняя длина сеанса

Пусть $T_{k,l}$ – средняя длина сеанса кратности k в вершине, соответствующей 2^l абонентам. Известно, что для канала без шума $T_{0,0} = T_{1,0} = 1$. Для канала с ложными конфликтами в окнах, соответствующих концевым вершинам дерева, может возникнуть ложный конфликт кратности 0 или 1. На рис. 8 рассмотрены возможные ситуации в концевой вершине дерева. Если в концевой вершине дерева имеет место ложный конфликт кратности 0 (рис. 8, *а*), то абоненты, участвующие в первоначальном конфликте, наблюдают за выходом канала до появления пустого окна. В случае, когда в концевой вершине дерева происходит ложный конфликт кратности 1 (рис. 8, *б*), абонент, который передавал пакет в данном окне, продолжает передавать пакет в последующих окнах до тех пор, пока пакет не будет успешно передан. Тогда

$$T_{0,0} = (1-q_0) + 2q_0(1-q_0) + 3q_0^2(1-q_0) + \ldots + sq_0^{s-1}\times$$

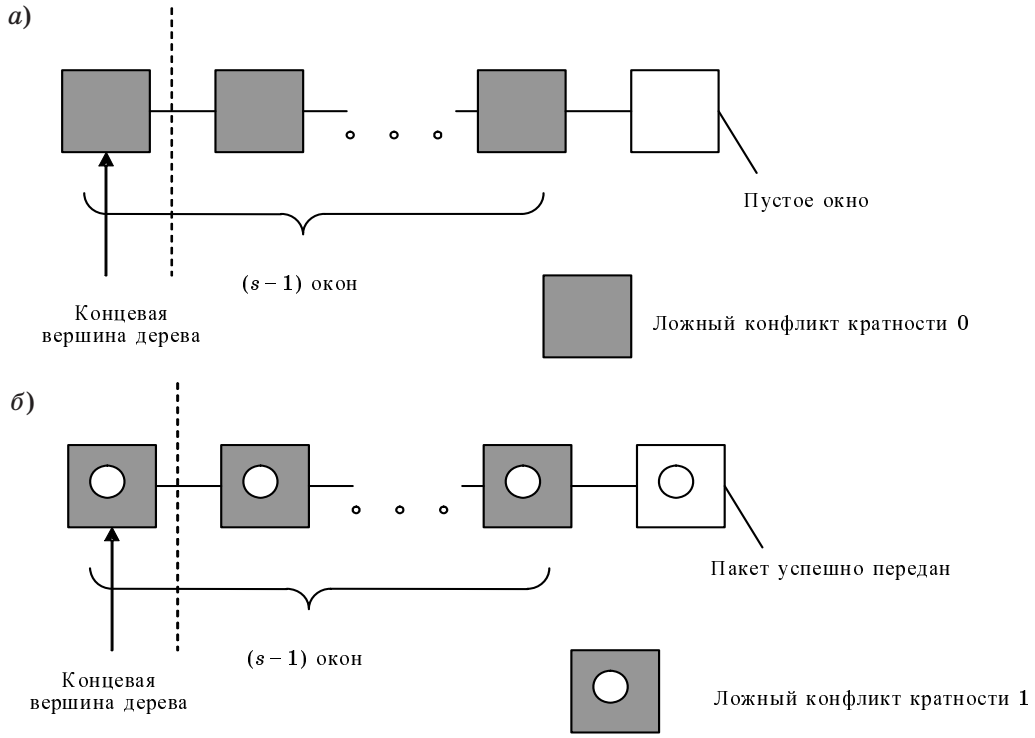
$$\times(1-q_0) = (1-q_0)(1+2q_0+3q_0^2+\ldots+sq_0^{s-1}\} = \frac{1}{1-q_0}.$$

Аналогично, для $T_{1,0}$ имеем

$$T_{1,0} = \frac{1}{1-q_1}.$$

Для блокированного немодифицированного стек-алгоритма определить $T_{k,l}$ можно по следующей рекуррентной формуле:

$$T_{k,l} = 1 + Q_k \sum_{i=\max(0,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i}\left(T_{i,l-1} + T_{k-i,l-1}\right),$$



■ ***Рис. 8.*** *Поведение алгоритма в концевых вершинах дерева: а) в концевой вершине ложный конфликт кратности 0; б) в концевой вершине ложный конфликт кратности 1*

где $\psi_{k,l,i} = \dfrac{C^{i}_{2^{l-1}} C^{k-i}_{2^{l-1}}}{C^{k}_{2^{l}}}$; Q_k – вероятность конфликта, которая определяется как

$$Q_k = \begin{cases} q_0, & \text{если} \quad k = 0 \\ q_1, & \text{если} \quad k = 1 \\ 1, & \text{если} \quad k \geq 2 \end{cases}. \qquad (4)$$

Формулу (4) можно легко получить, преобразовав выражение для $T_{k,l}$ в случае бесшумного канала, учитывая модификацию дерева разрешения конфликта для канала с ложными конфликтами.

Утверждение 5. Величины $T_{k,l}$ для модифицированного блокированного стек-алгоритма определяются по формуле

$$T_{k,l} = 1 + Q_k \sum_{i=\max(1,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i} \left(T_{k,l} + T_{k-i,l-1} \right) +$$
$$+ \psi_{k,l,0} \{ T_{k,l-1} (1 - q_0 + q_0 Q_k) + T_{0,l-1} Q_k - (1 - q_0) \}. \qquad (5)$$

Доказательство. Введем в рассмотрение две величины: $T^{(1)}_{k,l}$ – среднее время разрешения конфликта кратности k в системе из 2^l абонентов при условии, что в первом окне сеанса был ложный конфликт $(k = 0,1)$ или конфликт $(k \geq 2)$; $T^{(2)}_{k,l}$ – среднее время разрешения конфликта при условии, что в первом окне не было конфликта.

$$T^{(1)}_{k,l} = T_{k,l}, \text{ если } k \geq 2;$$
$$T^{(2)}_{k,l} = 1 \text{ при } k = 0,1;$$
$$T^{(2)}_{k,l} = 0 \text{ при } k \geq 2.$$

Тогда можно записать

$$T_{k,l} = Q_k T^{(1)}_{k,l} + (1 - Q_k) T^{(2)}_{k,l},$$

откуда можно выразить $T^{(1)}_{k,l}$:

$$T^{(1)}_{k,l} = \frac{T_{k,l} - (1 - Q_k)}{Q_k}. \qquad (6)$$

Для модифицированного блокированного стек-алгоритма имеем

$$T_{k,l} = 1 + Q_k \sum_{i=\max(1,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i} \left(T_{i,l-1} + T_{k-i,l-1} \right) + \psi_{k,l,0} B_k,$$

где

$$B_k = Q_k \left\{ (1 - q_0) \left(T^{(2)}_{0,l-1} + T^{(1)}_{k,l-1} - 1 \right) + \right.$$
$$\left. + q_0 \left(T^{(1)}_{0,l-1} + T_{k,l-1} \right) \right\}. \qquad (7)$$

Подставляя (6) в (7) и учитывая, что $T^{(2)}_{0,l-1} = 1$, имеем

$$B_k = Q_k\left\{(1-q_0)\frac{T_{k,\,l-1}-(1-Q_k)}{Q_k}+\right.$$
$$\left.+q_0\left(\frac{T_{0,\,l-1}-(1-q_0)}{q_0}+T_{k,\,l-1}\right)\right\}=(1-q_0)T_{k,\,l-1}-$$
$$-(1-q_0)(1-Q_k)+Q_kT_{0,\,l-1}-(1-q_0)Q_k+Q_kq_0T_{k,\,l-1}. \quad (8)$$

Сгруппировав слагаемые в выражении (8) и подставив результирующее выражение для B_k в (7), получим выражение (5). Утверждение доказано.

Введем в рассмотрение дополнительный параметр γ – тип алгоритма. Пусть для немодифицированного алгоритма $\gamma = 0$, а для модифицированного алгоритма $\gamma = 1$. Тогда можно записать общую формулу для $T_{k,l}$

$$T_{k,l}=1+Q_k\sum_{i=\max(\gamma,k-2^{l-1})}^{\min(k,2^{l-1})}\psi_{k,l,i}\left(T_{i,\,l-1}+T_{k-i,\,l-1}\right)+$$
$$+\psi_{k,l,0}\gamma\left\{T_{k,\,l-1}\left(1-q_0+q_0Q_k\right)+T_{0,\,l-1}Q_k-\left(1-q_0\right)\right\}. \quad (9)$$

Пусть $P_l(s\,|\,k)$ – вероятность события (сеанс кратности k длится s окон в вершине, соответствующей 2^l абонентам).

Вычисление $P_l(s\,|\,k)$

Известно, что для канала без шума $P_0(1\,|\,0)=P_0(1\,|\,1)=1$. Для канала с ложными конфликтами, учитывая поведение алгоритма в концевых вершинах дерева (см. рис. 8), можно записать:

$$P_0(s\,|\,0)=(1-q_0)q_0^{s-1},\ s\ge 1;$$

$$P_0(s\,|\,1)=(1-q_1)q_1^{s-1},\ s\ge 1;$$

$$P_0(s\,|\,k)=0,\ s\ge 1,\ k\ge 2.$$

Утверждение 6. Величины $p_l(s\,|\,k)$ для модифицированного блокированного стек-алгоритма определяются следующим образом:

$$p_l(s\,|\,k)=Q_k\sum_{i=\max(0,k-2^{l-1})}^{\min(k,2^{l-1})}\psi_{k,l,i}\sum_{\nu=1}^{s-2}p_{l-1}(\nu\,|\,i)\times$$
$$\times p_{l-1}(s-\nu-1\,|\,k-i)+\left(1-q_0\right)\times$$
$$\times\psi_{k,l,0}\left\{p_{l-1}(s-1\,|\,k)-p_{l-1}(s-2\,|\,k)\right\}. \quad (10)$$

Доказательство. Введем в рассмотрение величину $p_l^{(1)}(s\,|\,k)$ – вероятность события (сеанс кратности k длится s окон в вершине, соответствующей 2^l абонентам, при условии, что в данной вершине имеет место ситуация конфликта).

Тогда можно записать:

$$p_l(s\,|\,k)=Q_k\,p_l^{(1)}(s\,|\,k),$$

$$p_l(s\,|\,k)=p_l^{(1)}(s\,|\,k) \quad (11)$$

при $k\ge 2$.

Для модифицированного блокированного стек-алгоритма имеем

$$p_l(s\,|\,k)=Q_k\sum_{i=\max(1,k-2^{l-1})}^{\min(k,2^{l-1})}\psi_{k,l,i}\sum_{\nu=1}^{s-2}p_{l-1}(\nu\,|\,i)\times$$
$$\times p_{l-1}(s-\nu-1\,|\,k-i)+\psi_{k,l,0}Q_k\left\{\sum_{\nu=2}^{s-2}p_{l-1}(\nu\,|\,0)\times\right.$$
$$\left.\times p_{l-1}(s-\nu-1\,|\,k)+\left(1-q_0\right)p_{l-1}^{(1)}(s-1\,|\,k)\right\}. \quad (12)$$

Преобразуя первую сумму в формуле (12), определив нижний индекс как $i=\max\left(0,2^{l-1}\right)$ и выражая $p_{l-1}^{(1)}(s-1\,|\,k)$ через $p_{l-1}(s-1\,|\,k)$, получим

$$p_l(s\,|\,k)=Q_k\sum_{i=\max(0,k-2^{l-1})}^{\min(k,2^{l-1})}\psi_{k,l,i}\sum_{\nu=1}^{s-2}p_{l-1}(\nu\,|\,i)\times$$
$$\times p_{l-1}(s-\nu-1\,|\,k-i)-Q_k\psi_{k,l,0}\,p_{l-1}(s-2\,|\,k)\times$$
$$\times p_{l-1}(1\,|\,0)+\psi_{k,l,0}\left(1-q_0\right)p_{l-1}\ (s-1\,|\,k). \quad (13)$$

Заменяя в (13) $p_{l-1}(1\,|\,0)$ на $(1-q_0)$ и вынося $\psi_{k,l,0}\left(1-q_0\right)$ за скобки, получим (10). Утверждение доказано.

Отметим, что в выражении (10) второе слагаемое необходимо только для модифицированного алгоритма. Поэтому добавим параметр γ и приведем общую формулу для вычисления $P_l(s\,|\,k)$ для блокированного стек-алгоритма:

$$p_l(s\,|\,k)=Q_k\sum_{i=\max(0,k-2^{l-1})}^{\min(k,2^{l-1})}\psi_{k,l,i}\sum_{\nu=1}^{s-2}p_{l-1}(\nu\,|\,i)\times$$
$$\times p_{l-1}(s-\nu-1\,|\,k-i)+\left(1-q_0\right)\gamma\psi_{k,l,0}\times$$
$$\times\left\{p_{l-1}(s-1\,|\,k)-Q_k\,p_{l-1}(s-2\,|\,k)\right\}. \quad (14)$$

Среднее время выхода

Пусть $d_{k,l}$ – среднее время выхода пакета из конфликта кратности k в вершине, соответствующей 2^l абонентам.

Известно, что для канала без шума $d_{1,0}=0$. Для канала с ложными конфликтами, учитывая поведение алгоритма в концевых вершинах дерева (см. рис. 8), можно записать

$$d_{1,0}=0(1-q_1)+d_1(1-q_1)2q_1^2+(1-q_1)+\ldots$$

$$+\ sq_1^s\,(1-q_1)=(1-q_1)q_1(1+2q_1+3q_1^2+\ldots+sq_1^{s-1}\}=$$

$$=\frac{q_1}{1-q_1}.$$

Для блокированного немодифицированного стек-алгоритма определить $d_{k,l}$ можно по следующей рекуррентной формуле:

$$d_{k,l} = Q_k \left\{ 1 + \sum_{i=\max(0,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i} \left(\frac{i}{k} d_{i,l-1} + \right. \right.$$
$$\left. \left. + \frac{k-i}{k} \left(d_{k-i,l-1} + T_{i,l-1} \right) \right) \right\}. \qquad (15)$$

Утверждение 7. Величины $d_{k,l}$ для модифицированного блокированного стек-алгоритма определяются по формуле

$$d_{k,l} = Q_k \left\{ 1 + \sum_{i=\max(1,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i} \left(\frac{i}{k} d_{i,l-1} + \right. \right.$$
$$\left. \left. + \frac{k-i}{k} \left(d_{k-i,l-1} + T_{i,l-1} \right) \right) \right\} + \psi_{k,l,0} \left\{ d_{k,l-1} \left(1 - q_0 + q_0 Q_k \right) + \right.$$
$$\left. + T_{0,l-1} Q_k - \left(1 - q_0 \right) Q_k \right\}. \qquad (16)$$

Доказательство утверждения 7 выполняется аналогично доказательству утверждений 5 и 6 и поэтому здесь не приводится.

Объединяя формулы (15) и (16), получим

$$d_{k,l} = Q_k \left\{ 1 + \sum_{i=\max(\gamma,k-2^{l-1})}^{\min(k,2^{l-1})} \psi_{k,l,i} \left(\frac{i}{k} d_{i,l-1} + \right. \right.$$
$$\left. \left. + \frac{k-i}{k} \left(d_{k-i,l-1} + T_{i,l-1} \right) \right) \right\} + \psi_{k,l,0} \gamma \left\{ d_{k,l-1} \left(1 - q_0 + q_0 Q_k \right) + \right.$$
$$\left. + T_{0,l-1} Q_k - \left(1 - q_0 \right) Q_k \right\}. \qquad (17)$$

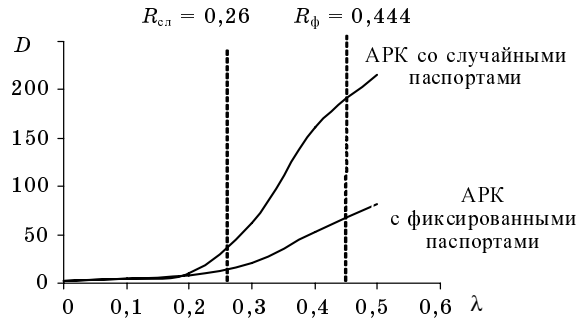
Численные результаты

На основе рассмотренной методики построим зависимости средней задержки от интенсивности входного потока. С помощью этих зависимостей проиллюстрируем, что во влиянии ложных конфликтов на процесс разрешения конфликтов имеется ряд особенностей по сравнению со случаем, когда для разрешения конфликтов не используются адреса абонентов. Результаты для алгоритмов, использующих случайные паспорта для разрешения конфликтов, получены численным путем по методике, изложенной в работе [3].

Средняя задержка пакетов для немодифицированного стек-алгоритма при использовании случайных и фиксированных паспортов абонента для разрешения конфликта в системе приведена в табл. 1. Для случайных паспортов наиболее критичной является вероятность ложного конфликта на пустом окне. Алгоритм со случайными паспортами работает только в случае, когда эта вероятность меньше 1/2. При использовании адресов абонентов

■ *Таблица 1*

λ	Средняя задержка D (M = 64, $q_0 = q_1$ = 0,2)	
	АРК с фиксированными паспортами	АРК со случайными паспортами
0,0	2,662	2,682
0,1	4,161	4,432
0,2	7,683	10,198
0,3	20,408	62,565
0,4	53,072	161,79
0,5	81,947	215,36



■ ***Рис. 9.*** *Зависимость средней задержки от интенсивности входного потока для АРК со случайными и фиксированными паспортами при $q_0 = q_1 = 0{,}2$*

ложные конфликты на пустом окне вообще не оказывают влияния на скорость алгоритма [см. формулу (2)]. На рис. 9 показаны значения скорости алгоритма СМД для алгоритма со случайными ($R_{сл}$) и фиксированными ($R_ф$) паспортами. Выражение для $R_{сл}$ было приведено в работе [9], а $R_ф$ рассчитывают по формулам (1), (2).

При отсутствии ложных конфликтов средняя задержка при использовании адресов абонентов лишь незначительно меньше задержки при использовании случайного выбора адресов. При увеличении вероятностей ложных конфликтов разница в задержке становится более существенной (рис. 10). В табл. 2 приведены средние задержки пакета в системе.

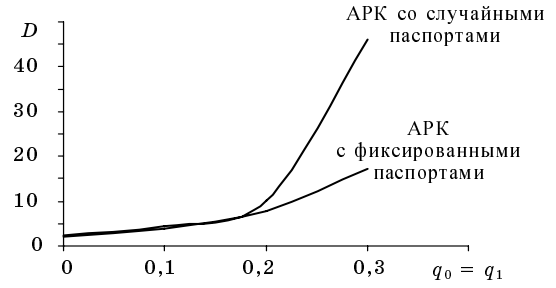
Известно, что при использовании случайных паспортов модифицированный стек-алгоритм не работоспособен в канале с ложными конфликтами [5]. При использовании фиксированных паспортов алгоритм работает в канале с шумом. Из графика зависимости средней задержки от интенсивности входного потока при различных значениях вероятностей ложных конфликтов (рис. 11) видно, что при низких интенсивностях наибольшее влияние на среднюю задержку оказывает ве-

■ *Таблица 2*

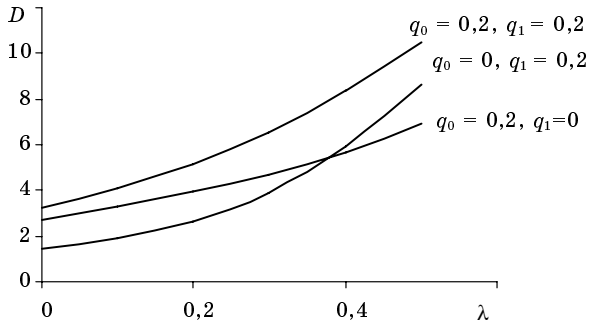
q_0=q_1	Средняя задержка D (M = 64, λ = 0,2)	
	АРК с фиксированными паспортами	АРК со случайными паспортами
0,0	2,13	2,338
0,1	3,86	4,432
0,2	7,683	10,198
0,3	17,163	46,03

■ *Таблица 3*

λ	Средняя задержка D (M = 8)		
	$q_0 = 0,0$ $q_1 = 0,2$	$q_0 = 0,2$ $q_1 = 0,0$	$q_0 = 0,2$ $q_1 = 0,2$
0,0	1,446	2,671	3,209
0,1	1,897	3,312	4,109
0,2	2,651	3,934	5,173
0,3	3,91	4,672	6,559
0,4	5,904	5,64	8,358
0,5	8,608	6,937	10,501



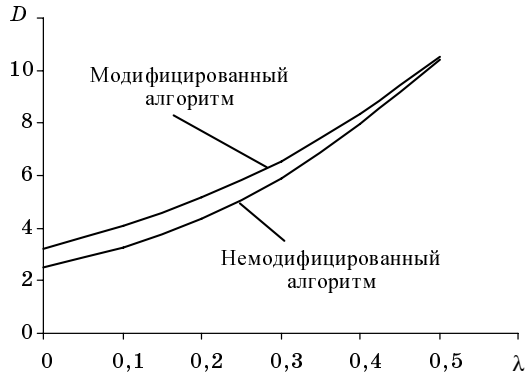
■ ***Рис. 10.*** *Зависимость средней задержки от вероятностей ложных конфликтов для АРК со случайными и фиксированными паспортами при l = 0,2*



■ ***Рис. 11.*** *Зависимость средней задержки от интенсивности входного потока для модифицированного алгоритма*

■ *Таблица 4*

λ	Средняя задержка D (M = 8, $q_0 = q_1 = 0,2$)	
	Немодифицированный стек-алгоритм	Модифицированный стек-алгоритм
0,0	2,496	3,209
0,1	3,292	4,109
0,2	4,367	5,173
0,3	5,898	6,559
0,4	7,967	8,358
0,5	10,4	10,501



■ ***Рис. 12.*** *Зависимость средней задержки от интенсивности входного потока для немодифицированного и модифицированного алгоритмов при $q_0 = q_1 = 0,2$*

роятность q_0. Это объясняется тем, что возникновение ложного конфликта на пустом окне приводит к появлению двух дополнительных окон, если окно не является концевым. При высоких интенсивностях уменьшается вероятность появления пустых окон, и наибольшее влияние на величину задержки оказывает вероятность q_1. Результаты численного расчета средней задержки в системе с шагом 0,1 и различных вероятностях ложных конфликтов представлены в табл. 3.

Если не использовать адреса для разрешения конфликта, то модифицированный алгоритм в бесшумном канале имеет более высокую скорость, но при этом алгоритм не работоспособен при появлении ложных конфликтов на пустом окне. При использовании адресов абонентов модификация не влияет на скорость алгоритма, однако алгоритм при этом становится работоспособен при появлении ложных конфликтов на пустом окне. В табл. 4 сравнение модифицированного и исходного алгоритмов по задержке выполнено с шагом 0,1. Соответствующая графическая зависимость приведена на рис. 12. Можно отметить, что модификация алгоритма позволяет лишь незначительно снизить

задержку при высоких интенсивностях. Это снижение становится значимым лишь для случая, когда ложные конфликты на пустых окнах отсутствуют.

Заключение

Изложена методика расчета точного значения средней задержки пакета для канала с шумом. Реализация предложенных алгоритмов оказывается достаточно простой и сводится к выполнению арифметических операций над тремя и четырьмя переменными для немодифицированного и модифицированного стек-алгоритмов соответственно. Алгоритмы, использующие адреса абонентов для разрешения конфликтов, оказываются более устойчивыми к проявлению ложных конфликтов, чем алгоритмы, основанные на чисто случайном способе разрешения конфликтов. Так, относительный выигрыш алгоритмов с фиксированными паспортами по средней задержке составляет примерно 37% (при $\lambda = 0{,}2$, $q_0 = q_1 = 0{,}2$ и $M = 8$) и резко возрастает при увеличении λ и вероятностей q_0 и q_1.

Литература

1. Giuseppe Bianchi. Performance Analysis of the IEEE 802.11 Distributed Coordination Function // IEEE Journal On Selected Areas In Communications. 2000. Vol. 18. N 3. P. 535–547.
2. IEEE P802.16-REVd/D5-2004. Draft IEEE standard for local and metropolitan area networks, Part 16: Air Interface for Fixed Broadband Wireless Access Systems.
3. **Цыбаков Б. С., Михайлов В. А.** Свободный синхронный доступ пакетов в широковещательный канал с обратной связью // Проблемы передачи информации. 1978. Т. 14. № 4. С. 32–59.
4. **Capetanakis J. L.** Generalized TDMA. The Multi-Accessing Tree Protocol Channels // IEEE Trans. Commun. 1979. Vol. 27. N 10. P. 1476–1483.
5. **Евсеев Г. С., Ермолаев Н. Г.** Оценка характеристик разрешения конфликтов в канале со свободным доступом и шумом // Проблемы передачи информации. 1982. Т.18. № 2. С.101–105.
6. **Черняк Л.** Сети промышленных контроллеров//Открытые системы. 2001. № 5–6.
7. **Тюрликов А. М., Марковский С. Г.** Использование адресов абонентов для организации доступа к высокоскоростному каналу связи // Информационно-управляющие системы. 2003. № 1. С. 32–38.
8. **Цыбаков Б. С., Михайлов В. А.** Эргодичность синхронной системы АЛОХА// Проблемы передачи информации. 1979. Т. 15. № 4. С.73–87.
9. **Евсеев Г. С., Тюрликов А. М.** Анализ пропускной способности одного алгоритма свободного множественного доступа, устойчивого к воздействию шумов // Проблемы передачи информации. 1986. Т. 22. № 2. С.104–109.

УДК 519.872

ПОЛНЫЙ РАСЧЕТ СИСТЕМЫ ОБСЛУЖИВАНИЯ С РАСПРЕДЕЛЕНИЯМИ КОКСА

Ю. И. Рыжиков,
доктор техн. наук, профессор
Военно-космическая академия им. А. Ф. Можайского

Разработан алгоритм полного расчета стационарных характеристик системы обслуживания вида $C_2/C_2/n$: стационарного распределения числа заявок, их распределения перед прибытием очередной заявки, моментов распределения интервалов между обслуженными заявками. Обсуждаются частные случаи и обобщения. Рассмотрены особенности программной реализации алгоритма, приведены результаты тестирования. Алгоритм предлагается использовать при расчете сетей обслуживания методом потокоэквивалентной декомпозиции.

An algorithm is proposed to compute the characteristics of the system $C_2/C_2/n$. This algorithm computes the stationary distribution of the number of demands, its distribution before arriving, and the moments of interdeparting intervals. Particular cases and generalizations, programming realization and testing results are discussed. The algorithm is planned to be used in the flow-equivalent network analysis.

Введение

Известно значительное влияние на результаты расчета сложных систем и сетей обслуживания не только средних интервалов между заявками и длительностей обслуживания, но и их распределений. Универсальным методом получения требуемых числовых характеристик является имитационное моделирование, существенными недостатками которого являются сложная логика программ, большой расход машинного времени и низкая точность определения малых вероятностей. Системы автоматизации имитационного моделирования типа GPSS (включая новейшую модификацию последней – GPSS World) не снимают перечисленных проблем, а лишь упрощают программирование ценой замедления работы моделей (как показали проведенные автором эксперименты – в десятки раз). Таким образом, несмотря на увеличение быстродействия ЭВМ, численные методы расчета СМО сохраняют свою актуальность.

Опубликовано множество методов расчета различных классов немарковских систем обслуживания, наиболее универсальные из которых опираются на подбираемые по методу моментов фазовые аппроксимации исходных распределений показательным, эрланговским, гиперэкспоненциальным, коксовым и общим распределением Ньютса. Для последнего, однако, отсутствуют алгоритмы расчета параметров аппроксимации. Эрланговские распределения из-за требования целочисленности фазового параметра не обеспечивают точного выравнивания даже двух моментов. Поэтому наибольшую практическую ценность представляют гиперэкспоненциальное и коксово распределения. Однако для первого из них распределения Эрланга являются особыми случаями: система последовательных фаз не может быть сведена к тому же числу параллельных. Это гарантирует аварийное завершение программы подбора параметров аппроксимирующего распределения, вынуждает менять тип аппроксимации в зависимости от исходных данных и требует обширного набора программ, включающего все возможные комбинации. От данного недостатка свободно распределение Кокса, исследование свойств и систематическое использование которого в обеих позициях систем вида $GI/G/n$ впервые описываются ниже.

Распределение Кокса

К фазовым распределениям относятся порождаемые системой подлежащих прохождению фаз обслуживания (прибытия заявки) с показательно распределенной длительностью пребывания в каждой из них. При фиксации номера фазы такие распределения приобретают марковское свойство, что и определяет целесообразность их использования. Для практических целей применение двухфазного распределения C_2, обеспечивающего выравнивание трех начальных моментов, представляется вполне достаточным. Диаграмма этого распределения представлена на рис. 1. Параметры μ_1 и μ_2 задают интенсивности потоков пе-

рехода, y – вероятность потребности во второй фазе, $\bar{y}$ – вероятность отсутствия таковой.

Получим алгоритм расчета параметров распределения C_2 по его моментам $\{\tilde{f}\}$. Прежде всего заметим, что эти моменты с вероятностью y являются моментами свертки показательно распределенных задержек в фазах, а с дополнительной к ней суть моменты задержки в первой фазе. Таким образом, для расчета параметров C_2 имеем систему уравнений

$$\frac{\bar{y}}{\mu_1}+y\left(\frac{1}{\mu_1}+\frac{1}{\mu_2}\right)=\tilde{f}_1;$$

$$\bar{y}\frac{2}{\mu_1^2}+y\left(\frac{2}{\mu_1^2}+2\frac{1}{\mu_1\mu_2}+\frac{2}{\mu_2^2}\right)=\tilde{f}_2;$$

$$\bar{y}\frac{6}{\mu_1^3}+y\left(\frac{6}{\mu_1^3}+3\frac{2}{\mu_1^2}\frac{1}{\mu_2}+3\frac{1}{\mu_1}\frac{2}{\mu_2^2}+\frac{6}{\mu_2^3}\right)=\tilde{f}_3.$$

Перейдем к обратным величинам $\{x_i=\mu_i^{-1}\}$ и нормированным моментам $\{f_i=\tilde{f}_i/i!\}$. Тогда исходная система может быть переписана как

$$\bar{y}x_1+y(x_1+x_2)=f_1;$$

$$\bar{y}x_1^2+y\left(x_1^2+x_1x_2+x_2^2\right)=f_2;$$

$$\bar{y}x_1^3+y\left(x_1^3+x_1^2x_2+x_1x_2^2+x_2^3\right)=f_3.$$

С учетом $y+\bar{y}=1$ она преобразуется в систему

$$x_1+yx_2=f_1;$$

$$x_1^2+yx_2(x_1+x_2)=f_2;$$

$$x_1^3+yx_2\left(x_1^2+x_1x_2+x_2^2\right)=f_3. \qquad (1)$$

Из первого уравнения системы исключаем

$$yx_2=f_1-x_1 \qquad (2)$$

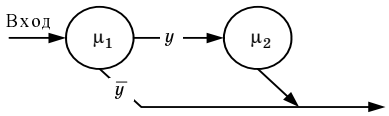
и подставляем в два оставшихся. Первое из них дает

$$x_2=\frac{f_2-x_1^2}{f_1-x_1}-x_1.$$

Конечная форма задачи – квадратное уравнение с решением

$$x_{1,2}=\frac{f_3-f_1f_2\pm\sqrt{4f_1^3f_3+4f_2^3+f_3^2-3f_1^2f_2^2-6f_1f_2f_3}}{2\left(f_2-f_1^2\right)}. \qquad (3)$$

Решение для x_1 берется с плюсом, y определяется из равенства (2). Заметим, что случай $f_2=f_1^2$ соответствует одной экспоненте и должен выявляться на первом этапе алгоритма. Здесь естественно принять $y=0$ и $\mu_1=1/f_1$, а значение μ_2 можно выбрать произвольно (например, равным μ_1).



■ ***Рис. 1.*** *Двухфазное распределение Кокса*

■ ***Таблица 1.*** *Параметры коксовой аппроксимации*

α	y	μ_1	μ_2
0.2	−10.196	268	3.732
0.4	−3.682	.488	3.512
0.6	−1.584	.677	3.323
0.8	−.577	.845	3.155
1.0	.000	1.000	1.000
1.2	.745	2.853	1.147
1.4	.815	2.707	1.293
1.6	.880	2.555	1.445
1.8	.940	2.378	1.622
2.0	1.000	2.000	2.000
2.2	1.061−.011i	2.000+.354i	2.000−.354i
2.4	1.111−.027i	2.000+.485i	2.000−.485i
2.6	1.154−.044i	2.000+.577i	2.000−.577i
2.8	1.190−.062i	2.000+.649i	2.000−.649i
3.0	1.222−.079i	2.000+.707i	2.000−.707i

Подставим в эти формулы моменты гамма-распределения с параметрами α и μ. Подкоренное выражение

$$D=\frac{\alpha^2(\alpha+1)(2-\alpha)(\alpha-1)^2}{18\mu^6}.$$

Таким образом, при $\alpha>2$ аппроксимация должна иметь комплексные параметры. Случай $\alpha=2$ является граничным: он соответствует распределению Эрланга второго порядка с $y=1$ и $\mu_1=\mu_2$.

В табл. 1 приведены параметры C_2-аппроксимации гамма-распределения при единичном первом моменте. Таблица подтверждает сделанный прогноз и снимает опасения относительно возможной отрицательности $\{\mu_j\}$. Отметим парадоксальный, но ожидаемый результат: при $\alpha<1$ вероятность выхода процесса во вторую фазу отрицательна.

■ ***Таблица 2.*** *Распределение числа заявок в системе $M/G/1$*

j	Параметр гамма-распределения							
	0.5		1.5		3.0		1e9	
	MG1	MCN	MG1	MCN	MG1	MCN	MG1	MCN
0	.300e-0	.300e-0	.300e-0	.300e-0	.300e-0	.300e-0	.300e-0	.300e-0
1	.165e-0	.167e-0	.233e-0	.233e-0	.263e-0	.263e-0	.304e-0	.306e-0
2	.120e-0	.118e-0	.159e-0	.159e-0	.174e-0	.173e-0	.190e-0	.187e-0
3	.912e-1	.903e-1	.106e-0	.106e-0	.106e-0	.106e-0	.101e-0	.101e-0
4	.706e-1	.703e-1	.696e-1	.696e-1	.637e-1	.638e-1	.517e-1	.524e-1
5	.550e-1	.550e-1	.457e-1	.456e-1	.379e-1	.380e-1	.263e-1	.267e-1
6	.430e-1	.431e-1	.299e-1	.299e-1	.226e-1	.226e-1	.134e-1	.135e-1
7	.336e-1	.338e-1	.196e-1	.196e-1	.134e-1	.134e-1	.682e-2	.682e-2
8	.264e-1	.265e-1	.128e-1	.128e-1	.798e-2	.798e-2	.347e-2	.344e-2
9	.207e-1	.208e-1	.842e-2	.841e-2	.474e-2	.474e-2	.177e-2	.173e-2
10	.162e-1	.167e-1	.551e-2	.551e-2	.282e-2	.281e-2	.899e-3	.873e-3
11	.127e-1	.128e-1	.361e-2	.361e-2	.168e-2	.167e-2	.457e-3	.440e-3
12	.996e-2	.100e-1	.237e-2	.237e-2	.996e-3	.993e-3	.233e-3	.222e-3
13	.782e-2	.784e-2	.155e-2	.155e-2	.592e-3	.590e-3	.118e-3	.118e-3
14	.613e-2	.614e-2	.101e-2	.101e-2	.352e-3	.350e-3	.603e-4	.562e-4
15	.481e-2	.481e-2	.665e-3	.666e-3	.209e-3	.208e-3	.307e-4	.283e-4
16	.377e-2	.377e-2	.436e-3	.436e-3	.124e-3	.124e-3	.156e-4	.143e-4
17	.296e-2	.296e-2	.285e-3	.286e-3	.740e-4	.735e-4	.794e-5	.719e-5
18	.232e-2	.232e-2	.187e-3	.187e-3	.440e-4	.436e-4	.404e-5	.362e-5

Приведем результаты расчета распределения числа заявок в системе $M/G/1$ (табл. 2). Процедура MG1 использовала общеизвестные соотношения, реализующие метод вложенных цепей Маркова на базе рекуррентного расчета распределения $\{q_i\}$ числа заявок, прибывающих за случайное время обслуживания:

$$q_j = \int_0^\infty \frac{(\lambda t)^j}{j!} e^{-\lambda t}\, dB(t), \quad j = 0, 1, \ldots,$$

для вырожденного и гамма-распределений [1]. Процедура MCN опиралась на C_2-аппроксимацию и решала описанные ниже векторно-матричные уравнения баланса. Индекс j указывает число заявок в системе. Параметр 1e9 фактически определяет вырожденное распределение, т. е. модель $M/D/1$.

Гамма-распределение было выбрано как удобный эталон по двум причинам:

1) параметры его плотности

$$b(t) = \frac{\mu(\mu t)^{\alpha-1}}{\Gamma(\alpha)} e^{-\mu t}$$

легко выражаются через среднее b_1 и дисперсию D согласно

$$\alpha = b_1^2 / D, \quad \mu = \alpha / b_1;$$

2) вышеупомянутые $\{q_j\}$ для него вычисляются до удивления просто [1, с. 82]:

$$q_0 = \left(\frac{\mu}{\lambda+\mu}\right)^\alpha;$$

$$q_j = q_{j-1} \frac{\lambda}{\lambda+\mu} \frac{\alpha+j-1}{j}, \quad j = 1, 2, \ldots .$$

Хорошее согласие прямого расчета модели $M/G/1$ и конечных результатов ее коксовой аппроксимации в широком диапазоне условий, в том числе для комплексных и отрицательных параметров, убедительно свидетельствует о приемлемости обсуждаемой аппроксимации.

Достоинством C_2-аппроксимации является естественное включение в нее как частных случаев M- и E_2-моделей, а недостатками – трудность обобщения на большее число составляющих и усложнение диаграмм переходов.

Многофазное представление системы

Будем представлять состояние системы $C_2/C_2/n$ полным числом заявок в системе, фазой прибытия очередной заявки и расстановкой проходящих обслуживание заявок по его фазам («ключами» соответствующих микросостояний). Например, ключ (21) означает, что две заявки проходят первую фазу обслуживания с интенсивностями μ_1, а одна – вторую с интенсивностью μ_2. Диаграмма состояний разделяется по фазам прибытия на две вертикальные ветви с интенсивностями λ_1 и λ_2 соответственно, а по числу заявок в системе – на горизонтальные ярусы. Совокупности ключей в левой и правой ветвях диаграммы дублируются. На рис. 2 показаны переходы по завершению фазы обслуживания для половины диаграммы (для другой половины они тождественны).

Обозначим через S_j множество всех возможных микросостояний системы, при которых на обслуживании находится ровно j заявок, а через σ_j – количество элементов S_j. Нам потребуются следующие матрицы интенсивностей инфинитезимальных переходов:

$A_j\left[\sigma_j \times \sigma_{j+1}\right]$ – в S_{j+1} (прибытие заявки),

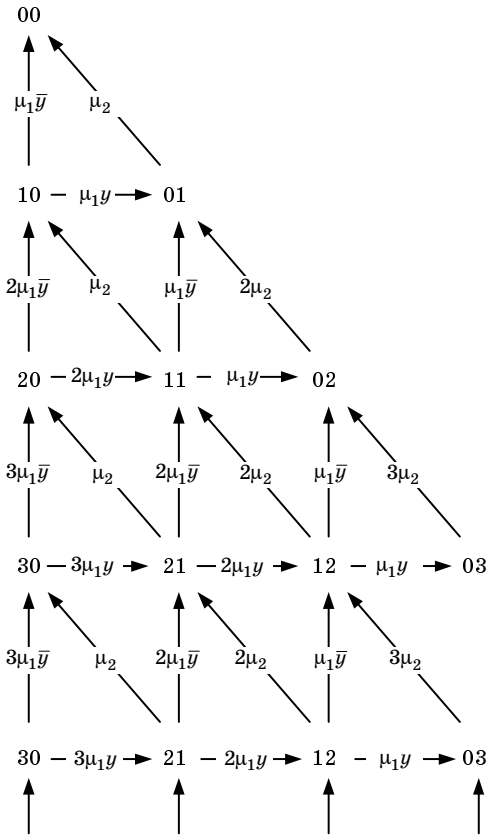
$F_j\left[\sigma_j \times \sigma_j\right]$ – в S_j (конец промежуточной фазы обслуживания или прибытия заявки),

$B_j\left[\sigma_j \times \sigma_{j-1}\right]$ – в S_{j-1} (полное завершение обслуживания заявки),

$D_j\left[\sigma_j \times \sigma_j\right]$ – ухода из состояний яруса j.

В квадратных скобках здесь и далее указывается размер матриц. Элемент (i, k) любой из этих матриц представляет интенсивность перехода из i-го состояния j-го яруса в k-е состояние смежного (по переходам рассматриваемого типа) яруса. Заметим, что при $j > n$ все эти матрицы перестают зависеть от индекса. Ниже будут приведены примеры матриц для второго яруса.

Переходы по завершению фазы обслуживания не меняют фазы прибытия, что позволяет ограничиться изображением для них половины диаграммы (рис. 2). Здесь окончание первой фазы с интенсивностью $y\mu_1$ переводит заявку во вторую фазу обслуживания, а с интенсивностью $\overline{y}\mu_1$ – завершает обслуживание и поднимает изображающую точку на вышележащий ярус. Последнее при завершении второй фазы (интенсивность μ_2) происходит всегда. Интенсивности переходов проставлены с учетом числа заявок, находящихся в каждой фазе исходного микросостояния.



■ ***Рис. 2.*** *Интенсивности переходов по завершению обслуживания в фазе*

Например, для полной диаграммы переходов по завершению обслуживания («удвоенный» рис. 2) имеем клеточно-диагональную матрицу интенсивностей переходов со второго на первый ярус:

$$B_2 = \begin{bmatrix} 2\mu_1\overline{y} & 0 & 0 & 0 \\ \mu_2 & \mu_1\overline{y} & 0 & 0 \\ 0 & 2\mu_2 & 0 & 0 \\ 0 & 0 & 2\mu_1\overline{y} & 0 \\ 0 & 0 & \mu_2 & \mu_1\overline{y} \\ 0 & 0 & 0 & 2\mu_2 \end{bmatrix}.$$

Диаграмма переходов по фазам прибытия заявок в трехканальной системе оказывается весьма запутанной; поэтому мы ограничимся описанием логики ее построения. Если интервалы между заявками подчинены C_2-распределению с параметра-

ми $\{u, \lambda_1, \lambda_2\}$, то все переходы из первой фазы с интенсивностями $u\lambda_1$ приводят в аналогичное микросостояние второй фазы прибытия того же яруса, а с интенсивностями $\bar{u}\lambda_1$ – в первую фазу нижележащего яруса. Переходы по завершению второй (всегда последней) фазы имеют интенсивность λ_2 и также приводят в первую фазу нижележащего яруса. Соответственно интенсивности «окончательных прибытий» описывает матрица

$$A_2 = \begin{bmatrix} \bar{u}\lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \bar{u}\lambda_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \bar{u}\lambda_1 & 0 & 0 & 0 & 0 & 0 \\ \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Переходы в пределах яруса могут быть связаны с выходом из первой во вторую фазу обслуживания (в пределах каждой ветви) либо с прибытием заявки (переход из левой ветви в одноименное микросостояние правой):

$$F_2 = \begin{bmatrix} 0 & 2\mu_1 y & 0 & u\lambda_1 & 0 & 0 \\ 0 & 0 & \mu_1 y & 0 & u\lambda_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & u\lambda_1 \\ 0 & 0 & 0 & 0 & 2\mu_1 y & 0 \\ 0 & 0 & 0 & 0 & 0 & \mu_1 y \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Для диагональной матрицы D_2 суммарных интенсивностей ухода из микросостояний яруса мы приведем только ее диагональ:

$$\operatorname{diag} D_2 = [\lambda_1 + 2\mu_1,\ \ \lambda_1 + \mu_1 + \mu_2,$$
$$\lambda_1 + 2\mu_2,\ \ \lambda_2 + 2\mu_1,\ \ \lambda_2 + \mu_1 + \mu_2,\ \ \lambda_2 + 2\mu_2].$$

Уравнения глобального баланса и их решение

Получим стационарные вероятности микросостояний, представленных на диаграммах предыдущего раздела.

Введем векторы-строки $\gamma_j = \{\gamma_{j,1}, \gamma_{j,2}, \ldots, \gamma_{j,\sigma_j}\}$ нахождения системы в состоянии (j, i), $j = 0, 1, \ldots$. Теперь можно записать векторно-матричные уравнения баланса переходов между состояниями

$$\gamma_0 D_0 = \gamma_0 F_0 + \gamma_1 B_1;$$
$$\gamma_j D_j = \gamma_{j-1} A_{j-1} + \gamma_j F_j + \gamma_{j+1} B_{j+1}, \quad j = 1, 2, \ldots . \quad (4)$$

Система (4), дополненная условием нормировки, даже для моделей с ограниченной очередью характеризуется чрезвычайно высокой размерностью, и стандартные методы решения систем линейных алгебраических уравнений применительно к ней оказываются малоэффективными. Для расчета разомкнутых систем обслуживания можно применить восходящий к Ивэнсу [5] метод матрично-геометрической прогрессии. Этот метод основан на представлении векторов вероятностей микросостояний для $j > n$ формулой

$$\gamma_j = \gamma_n R^{j-n}, \quad j = n, n+1, \ldots, \quad (5)$$

где R – некоторая матрица (знаменатель прогрессии). Перепишем второе из уравнений (4) с учетом (5) для значения j, при котором произошла вышеупомянутая стабилизация матриц, так что их индексы можно опустить:

$$\gamma_{j-1} RD = \gamma_{j-1} A + \gamma_{j-1} RF + \gamma_{j-1} R^2 B. \quad (6)$$

Теперь ясно, что R удовлетворяет матричному квадратному уравнению

$$R^2 B + RE + A = 0, \quad (7)$$

где $E = F - D$. Применим к его решению метод итераций. Будем искать поправку Δ к очередному приближению из уравнения (7), заменив в нем R на $R + \Delta$. Пренебрегая членом, содержащим квадрат поправки, имеем

$$R^2 B + \Delta RB + R\Delta B + RE + \Delta E + A \approx 0$$

или

$$\Delta(RB + E) + R\Delta B \approx -(R^2 B + RE + A).$$

Последнее уравнение приводится к уравнению

$$\Delta G + R\Delta B = H,$$

которое можно расписать как систему линейных алгебраических уравнений относительно компонент матричной поправки Δ и решить любым стандартным методом. В качестве начального приближения можно принять $R = \rho^{2/(\upsilon_A^2 + \upsilon_B^2)} I$, где ρ – коэффициент загрузки системы; υ_A и υ_B – коэффициенты вариации распределений интервалов между смежными заявками и длительности обслуживания соответственно; I – единичная матрица.

Далее усеченная система (4) для $j = 0, n$, в последнем из уравнений которой вектор γ_{n+1} заменен на $\gamma_n R$, расписывается относительно компонент векторов $\{\gamma_j\}$, $j = 0, n$. Затем одно из уравнений заменяется на условие баланса заявок

$$\sum_{j=0}^{n-1} (n - j)\gamma_j \mathbf{1}_j = n - b/a,$$

где $\mathbf{1}_j$ – вектор-столбец из σ_j единиц; a и b – средние интервал между заявками и длительность обслуживания. Наконец, находится решение системы для компонент начальных векторов.

Последующие векторы вероятностей микросостояний вычисляются согласно формуле (5).

Для решения системы (4) можно также применить менее быстродействующий, но более универсальный итерационный метод, впервые предложенный Такахаси и Таками [7] и описанный в работах [1, 2] в более общей форме, с детальной проработкой расчетных зависимостей и с частными вариантами.

Расчет выходящего потока

В сетях обслуживания потоки на входе узлов формируются из выходящих потоков других узлов. В общем случае выходящий поток сохраняет среднюю интенсивность (и средний интервал между заявками) входящего, но меняет вид распределения интервалов между заявками. Последнее удобно характеризовать коэффициентами немарковости

$$\xi_i = d_i / d_1^i - i!, \quad i = 2,\ 3,\ \ldots,$$

где $\{d_j\}$ – моменты распределения интервалов между заявками выходящего потока, который предполагается рекуррентным. Для простейшего потока упомянутые коэффициенты согласно данному определению равны нулю.

Ключом к проблеме преобразования потока в коксовом узле обслуживания является расчет марковской системы.

Марковские системы. В системе $M/M/n$, загруженной полностью, частное преобразование Лапласа–Стилтьеса (ПЛС) интервала до очередного обслуживания при наличии в системе $j \ge n$ заявок

$$\delta_j(s) = n\mu / \left(n\mu + s\right), \quad j = n,\ n+1,\ \ldots .$$

При $j < n$ интенсивность обслуживания меняется с прибытием каждой очередной заявки. Будем интерпретировать s как параметр простейшего потока «катастроф». Тогда $\delta_j(s)$ можно считать вероятностью отсутствия катастроф за интервал от ухода заявки, оставившей в системе ровно j заявок, до следующего завершения обслуживания. Вероятности появления событий каждого из рассматриваемых простейших потоков (катастроф, завершений обслуживания и прибытия новых заявок) пропорциональны их интенсивностям, т. е. s, $j\mu$ и λ соответственно. Теперь ясно, что

$$\delta_j(s) = \frac{j\mu}{j\mu + \lambda + s} + \frac{\lambda}{j\mu + \lambda + s}\delta_{j+1}(s),$$
$$j = n-1,\ n-2,\ \ldots,\ 0.$$

Здесь первое слагаемое соответствует завершению обслуживания, а второе – прибытию новой заявки (отчего число заявок в системе увеличилось на единицу) и отсутствию катастроф до следующего завершения обслуживания при новом начальном условии.

Результирующее распределение интервалов между заявками выходящего потока задается ПЛС

$$D(s) = \sum_{j=0}^{n-1} \pi_j \delta_j(s) + \left(1 - \sum_{j=0}^{n-1} \pi_j\right)\delta_n(s),$$

где финальные вероятности $\{\pi_j\}$ наличия в системе сразу после завершения обслуживания ровно j заявок при $t \to \infty$ подсчитываются согласно уравнению

$$\pi_j = \mu_{j+1} p_{j+1} \Big/ \sum_{i=1}^{\infty} \mu_i p_i, \quad j = 0,\ 1,\ \ldots . \qquad (8)$$

Здесь $\{p_j\}$ – стационарное распределение числа заявок в системе, а μ_j равно $j\mu$ при $j<n$ и $n\mu$ – в противном случае. Знаменатель формулы (8) сводится к равенству

$$\sum_{i=1}^{\infty} \mu_i p_i = \mu\left[\sum_{i=1}^{n-1} i p_i + n p_n / (1-\rho)\right].$$

Моменты искомого распределения можно получить многократным численным дифференцированием в нуле таблично заданной функции $D(s)$, $s = 0$, h, $2h$, ... с последующей сменой знаков у нечетных производных. Однако предпочтительнее вычислять их непосредственно. Обозначим $E(x)$ набор моментов показательного распределения с параметром x, имеющий компоненты

$$E_i(x) = i!/x^i, \quad i = 1,\ 2, \ldots$$

Символом * будем обозначать свертку распределений в моментах. Для наборов моментов распределений между выходящими заявками сохраним индексацию прежних обозначений, заменив лишь $\delta(s)$ на d. В модели $M/M/n$ при числе заявок в системе $j \ge n$

$$d_j = E(n\mu), \quad j = n,\ n+1,\ \ldots .$$

Перепишем формулу для $\delta_j(s)$ при $j < n$ в виде

$$\delta_j(s) = \frac{j\mu + \lambda}{j\mu + \lambda + s}\left(\frac{j\mu}{j\mu + \lambda}\cdot 1 + \frac{\lambda}{j\mu + \lambda}\delta_{j+1}(s)\right) =$$
$$= \frac{j\mu}{j\mu + \lambda}\frac{j\mu + \lambda}{j\mu + \lambda + s} + \frac{\lambda}{j\mu + \lambda}\frac{j\mu + \lambda}{j\mu + \lambda + s}\delta_{j+1}(s).$$

Заменяя преобразование Лапласа показательных плотностей их моментами и произведение ПЛС – сверткой, убеждаемся, что

$$d_j = \frac{j\mu}{j\mu + \lambda} E(j\mu + \lambda) + \frac{\lambda}{j\mu + \lambda} E(j\mu + \lambda) * d_{j+1},$$
$$j = n-1,\ n-2,\ \ldots,\ 0,$$

в частности:

$$d_0 = E(\lambda) * d_1.$$

Заметим, что свертка в моментах распределений A и B может быть выполнена на основе символического разложения

$$c^n = (a+b)^n \qquad (9)$$

путем перевода показателей степени в индексы соответствующих моментов. Для выполнения сверток полезно составить автономную процедуру.

Результаты практически совпали с моментами распределения интервалов между входящими заявками, а коэффициенты немарковости с точностью до пяти знаков оказались нулевыми (выходящий поток – простейший). Это согласуется с классическим результатом Берке, позволяет считать основную идею расчета правильной и распространить ее на более сложные немарковские системы, для которых теория расчета выходящего потока до сих пор отсутствовала (в частности, на интересующую нас «дважды коксову»).

Коксова система. Введем обозначения для j-го яруса:

$M_{j,i}$ – суммарная интенсивность обслуживания в i-м микросостоянии каждой ветви;

$\widehat{M}_{j,i}$ – интенсивность переходов вверх по завершению обслуживания;

$\overrightarrow{M}_{j,i}$ – интенсивность переходов вправо по завершению фазы обслуживания (сумма двух последних интенсивностей равна $M_{j,i}$);

$d_{j,i}^{(1)}$, $d_{j,i}^{(2)}$ – наборы моментов распределения времени до ближайшего обслуживания при соответствующем исходном состоянии (верхние индексы указывают левую и правую ветвь диаграммы соответственно);

σ_j – количество микросостояний в одной ветви диаграммы.

В этих обозначениях для $j \ge n$ и правой ветви диаграммы имеем

$$d_{n,i}^{(2)} = \frac{\widehat{M}_{n,i}}{M_{n,i}+\lambda_2} E(M_{n,i}+\lambda_2) +$$
$$+E(M_{n,i}+\lambda_2)*\left[\frac{\overrightarrow{M}_{n,i}}{M_{n,i}+\lambda_2} d_{n,i}^{(2)} + \frac{\lambda_2}{M_{n,i}+\lambda_2} d_{n,i}^{(1)}\right],$$
$$i = \sigma_n,\ \sigma_n - 1,\ \ldots,\ 1.$$

Для тех же значений j и левой ветви диаграммы надо дополнительно учитывать возможные переходы в правую ветвь по завершению фазы прибытия заявки:

$$d_{n,i}^{(1)} = \frac{\widehat{M}_{n,i}}{M_{n,i}+\lambda_1} E(M_{n,i}+\lambda_1) + E(M_{n,i}+\lambda_1)*$$
$$*\left[\frac{\overrightarrow{M}_{n,i}}{M_{n,i}+\lambda_1} d_{n,i}^{(1)} + \frac{u\lambda_1}{M_{n,i}+\lambda_1} d_{n,i}^{(2)} + \frac{\bar{u}\lambda_1}{M_{n,i}+\lambda_1} d_{n,i}^{(1)}\right],$$
$$i = \sigma_n,\ \sigma_n - 1,\ \ldots,\ 1.$$

После выполнения сверток в моментах согласно формуле (9) замечаем, что для каждого i искомые $\{d_{n,i}\}$ входят в обе части приведенных уравнений, причем «с перекрестом». Поэтому для их определения приходится последовательно решать σ_n систем из шести линейных алгебраических уравнений (по три момента в обеих ветвях на каждое микросостояние).

Для «ненасыщенных» ярусов $j = n-1, n-2, \ldots, 1$ в правой ветви диаграммы прибытие заявки приводит в одноименные состояния левой ветви нижележащего яруса. Соответственно имеем рекуррентные формулы

$$d_{j,i}^{(2)} = \frac{\widehat{M}_{j,i}}{M_{j,i}+\lambda_2} E(M_{j,i}+\lambda_2) +$$
$$+E(M_{j,i}+\lambda_2)*\left[\frac{\overrightarrow{M}_{j,i}}{M_{j,i}+\lambda_2} d_{j,i+1}^{(2)} + \frac{\lambda_2}{M_{j,i}+\lambda_2} d_{j+1,i}^{(1)}\right],$$
$$i = \sigma_j,\ \sigma_j - 1,\ \ldots,\ 1.$$

Для левой ветви зоны ненасыщенных состояний завершение фазы прибытия заявки может привести как в правую ветвь диаграммы, так и в нижележащий ярус:

$$d_{j,i}^{(1)} = \frac{\widehat{M}_{j,i}}{M_{j,i}+\lambda_1} E(M_{j,i}+\lambda_1) + E(M_{j,i}+\lambda_1)*$$
$$*\left[\frac{\overrightarrow{M}_{j,i}}{M_{j,i}+\lambda_1} d_{j,i+1}^{(1)} + \frac{u\lambda_1}{M_{j,i}+\lambda_1} d_{j,i}^{(2)} + \frac{\bar{u}\lambda_1}{M_{j,i}+\lambda_1} d_{j+1,i}^{(1)}\right],$$
$$i = \sigma_j,\ \sigma_j - 1,\ \ldots,\ 1.$$

Здесь первое слагаемое соответствует непосредственному завершению обслуживания, а последующие – переходам по завершению первой фазы обслуживания и фазы прибытия заявки (окончательной или с переходом во вторую фазу ожидания прибытия) с рекуррентным учетом моментов распределения времени до ближайшего обслуживания во вновь возникшем состоянии.

Наконец, в случае полного освобождения системы до очередного завершения обслуживания сначала придется дождаться прибытия хотя бы одной заявки. Здесь

$$d_{0,1}^{(2)} = E(\lambda_2) * d_{1,1}^{(1)};$$
$$d_{0,1}^{(1)} = E(\lambda_1) * \left(u d_{0,1}^{(2)} + \bar{u} d_{1,1}^{(1)}\right) d_{1,1}^{(1)}.$$

В «ненасыщенной» зоне искомые наборы моментов определяются последовательно через ранее вычисленные: для каждого i сначала вычисляются «правые», а затем «левые» моменты.

Распределение времени ожидания

Интересующее нас распределение можно задать его первым моментом $\tilde{w}_1$ и набором коэффициентов немарковости. Тогда высшие моменты можно вычислить по формуле

$$\tilde{w}_i = \tilde{w}_1^i(\xi_i + i!), \quad i = 2, 3, \ldots . \qquad (10)$$

Известно, что условное распределение времени ожидания заявки в системе $GI/M/n$ подчинено показательному закону. Это обстоятельство позволяет считать, что вид распределения времени ожидания не зависит от распределения интервалов между смежными заявками и числа каналов и определяется только распределением времени обслуживания. В работе [4] предложен конструктивный метод решения этой задачи, основанный на сохранении коэффициентов немарковости системы $M/G/1$.

Проблемы программной реализации

Зависимость структуры матриц переходов от типа и порядка аппроксимирующих фазовых распределений ставит ценность программной реализации расчетной схемы в прямую зависимость от возможности автоматического построения матриц переходов. Эту проблему в общем случае можно решить следующим образом:

– для каждого j-го яруса системы, $j = \overline{0, n}$, автоматически сгенерировать последовательность ключей микросостояний;

– формировать матрицы переходов, сопоставляя «ключи-источники» j-го яруса и совместимые с ними по выбранному типу переходов ключи-«преемники» для матрицы B_j на $(j-1)$-м ярусе, для F_j – на j-м, для A_j – на $(j+1)$-м.

В обсуждаемом случае C_2-аппроксимации структура матриц и правила их формирования относительно очевидны. Однако программная реализация метода оказалась значительно сложнее, чем это представляется при ознакомлении с его идеей. Дело в том, что метод предполагает работу с переменным числом матриц перехода изменяемой размерности, определяемой в процессе счета. Соответственно при программировании на Фортране пришлось упаковывать матрицы каждого вида (и их половинки для треугольных матриц) в линейные массивы, выделять под эти объекты динамическую ([allocatable]) память, фиксировать отдельно их размеры и координаты в линейных структурах, явно переадресовывать операнды матричных операций.

На эти проблемы наложились порождаемые специфической структурой упомянутых матриц: A – блочная с ненулевыми блоками в виде диагональных матриц, B – блочно-диагональная с одинаковыми блоками, F – верхняя треугольная, D – диагональная.

Учет этих особенностей при программировании требуемых матричных операций оказался нетривиальным и весьма поучительным.

Тестирование метода

Для проверки корректности метода и его программной реализации оценивалась средняя длина очереди (показатель компактный, наглядный, чувствительный к входным данным и инвариантный к масштабу времени) при коэффициенте загрузки системы $\rho = 0{,}7$ для различных видов исходных распределений и числа каналов. Для моделей (в обозначениях Кендалла) $M/M/1$, $M/E_3/1$, $M/D/1$, $M/\Gamma_{1,5}/1$, $E_4/M/1$, $D/M/1$, $\Gamma_{1,5}/M/1$, $M/M/2$, $M/D/2$, $M/E_3,/2$, $E_4/D/2$ были получены соответственно значения 1,633, 1,089, 0,817, 1,361, 0,861, 0,599, 1,292, 1,345, 0,692, 0,911, 0,113. Эти результаты практически совпали с полученными ранее посредством других численных методов, в том числе не использующих фазовые аппроксимации (например, метода Кроммелина для систем $M/D/n$, Быкадорова для $E_k/D/n$, Такача для $GI/M/n$).

Правильность расчета выходящего потока подтверждается:

– равенством средних интервалов между заявками входящего и выходящего потоков;

– получением практически нулевых коэффициентов немарковости выходящего потока для моделей вида $M/M/n$, обработанных по общей схеме;

– приближением этих коэффициентов к таковым для распределения времени обслуживания – в случае одноканальных систем при устремлении коэффициента загрузки к единице;

– совпадением результатов обсчета вышеперечисленных моделей с полученными посредством менее универсальной аппроксимации $H_2/H_2/n$.

Частные случаи и обобщения

Самостоятельную ценность имеет наиболее типичный частный случай рассмотренной модели – с простейшим входящим потоком. В этом случае диаграмма переходов сводится к одной ветви и программная реализация сильно упрощается.

Как обобщение алгоритма можно рассматривать его вариант для системы с ограниченной очередью. В этом случае в микросостояния нижнего яруса диаграммы нет переходов снизу. Соответственно для этого яруса корректируется матрица $D-F$ и отпадает надобность в расчете одного из вспомогательных векторов итерационного метода (β_j''). Для определения вероятности свободного состояния на заключительном этапе итерационного метода здесь условие баланса заявок заменяется условием нормировки вероятностей. Дальнейшим обобщением является расчет замкнутой системы с интенсивностью входящего (простейшего) потока, зависящей от числа заявок в системе – в типичном случае линейно убывающей до нуля. Здесь дополнительно модифицируются числовые значения матриц.

Заключение

Конкретные результаты данной статьи сводятся к следующим.

1. Предложен алгоритм расчета параметров распределения Кокса C_2, выравнивающий три заданных момента.

2. Проведено исследование алгоритма; выявлена область входных данных, порождающая «патологические» вероятности переходов (комплексные и отрицательные); показано, что при обратном переходе от вероятностей микросостояний к укрупненным вероятностям патология исчезает и конечные результаты согласуются с полученными существенно иными методами. Указаны преимущества C_2-аппроксимации в сравнении с другими фазовыми распределениями.

3. Впервые предложены диаграммы переходов для «дважды коксовой» системы, на основе которой получены матрицы интенсивностей переходов.

4. Разработан способ расчета распределения интервалов между заявками выходящего потока, в частности – непосредственно моментов этого распределения.

5. Даны рекомендации по программной реализации основных этапов алгоритма.

Эти результаты вместе с методами решения уравнений баланса и расчета временных характеристик позволяют, наконец, сравнительно просто и достаточно точно проводить полный расчет многоканальных систем с произвольными распределениями интервалов между заявками и времени обслуживания.

В сравнении с имитационным моделированием метод существенно выигрывает по объему и точности вычислений. Он избавляет от изучения и использования других фазовых аппроксимаций и (если не говорить о подготовке будущих теоретиков) – вообще всех прочих методов расчета немарковских систем обслуживания. Метод может использоваться как составная часть процедур расчета сетей обслуживания методом потокоэквивалентной декомпозиции [1].

Литература

1. **Рыжиков Ю. И.** Теория очередей и управление запасами: Учебник. СПб.: Питер, 2001. 384 с.
2. **Рыжиков Ю. И.** Алгоритм расчета многоканальной системы с эрланговским обслуживанием //Автоматика и телемеханика. 1980. № 5. С. 30–37.
3. **Рыжиков Ю. И., Хомоненко А. Д.** Итеративный метод расчета многоканальных систем с произвольным распределением времени обслуживания // Проблемы управления и теории информации. 1980. № 3. С. 32–38.
4. **Рыжиков Ю. И.** Три метода расчета временных характеристик разомкнутых систем массового обслуживания // Автоматика и телемеханика. 1993. № 2. С. 127–133.
5. **Evans R. D.** Geometric Distribution in Some Two Dimensional Queuing Systems // Operat. Res. 1967. Vol. 15. N 5. P. 830–846.
6. **Khomonenko A. D., Bubnov V. P.** A Use of Coxian Distribution for Iterative Solution of M/G/n/R≤∞\le \infty queuing systems // Problems of Control and Information Theory. 1985. Vol. 14. N 2. P. 143–153.
7. **Takahashi Y., Takami Y.** A Numerical Method for the Steady-state Probabilities of a GI/G/c Queuing System in a General Class // J. of the Operat. res. soc. of Japan. 1976. Vol. 19. N 2. P. 147–157.

УДК 007: 57 + 007:573

ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ИССЛЕДОВАНИЙ И КОРРЕКЦИЙ В ГЕРНИОЛОГИИ

П. И. Бегун,
доктор техн. наук, профессор
Санкт-Петербургский государственный электротехнический университет
С. М. Лазарев,
доктор мед. наук, профессор
Санкт-Петербургская государственная медицинская академия им. И. И. Мечникова
Е. А. Лебедева,
аспирант
Санкт-Петербургский государственный электротехнический университет

Построены расчетные схемы и компьютерные модели для вычисления напряжений и перемещений в герниосистемах. Проведены исследования зависимости напряжений и перемещений при развитии патологического образования в белой линии живота, при ущемлении грыжевых ворот и после проведения операции.

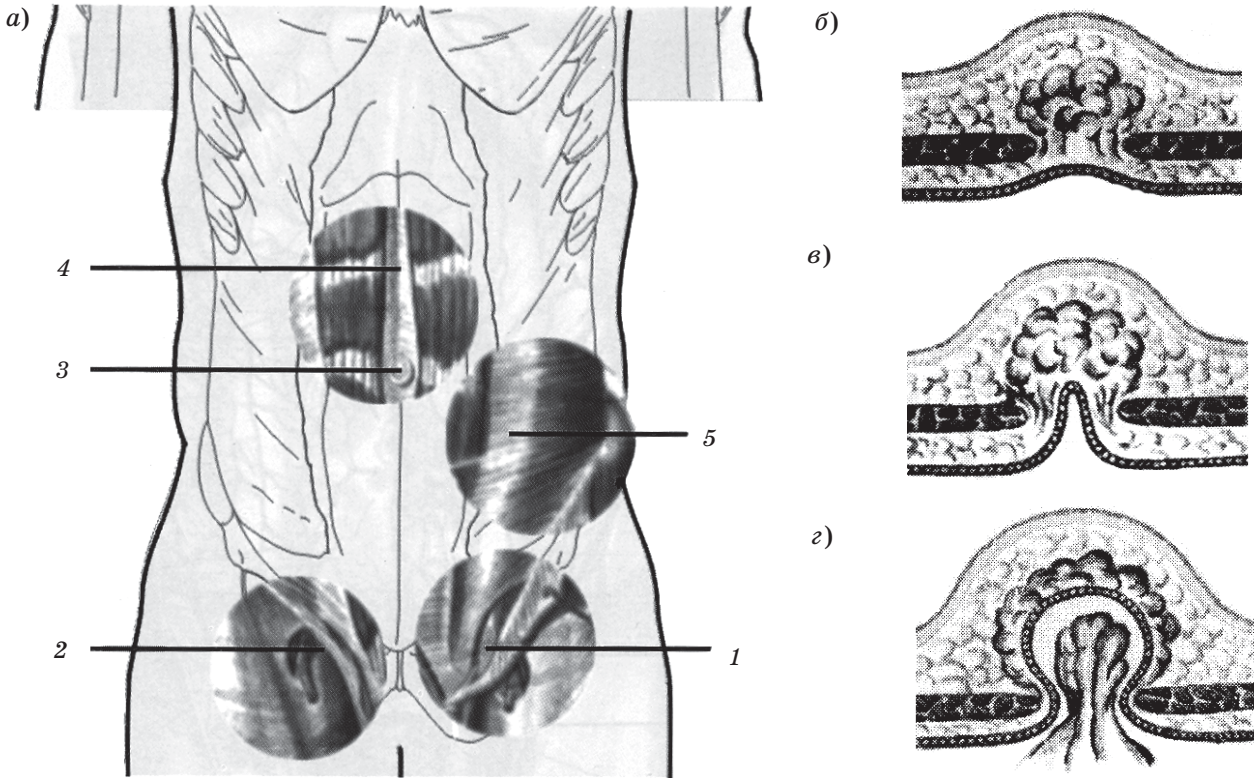
Theoretical schemes and computer-based models of calculation of the voltage and displacements in herniasystems are constructed. The dependence of voltage and displacement are studied during the development of pathological forming in the white line of a stomach and the infringement of the hernial gate.

Герниология – раздел хирургии, изучающий этиологию, патогенез и локализацию грыж живота и разрабатывающий методы их лечения и профилактики. Грыжи живота – выхождение органов брюшной полости за ее пределы под кожу или в другие ткани и полости. Грыжи передней брюшной стенки – одно из самых распространенных заболеваний человека, они наблюдаются у 30% населения планеты. Основные признаки этой патологии: 1) грыжевые ворота – расширение естественных щелей в брюшной стенке или образование новых (искусственных); 2) грыжевый мешок – выпячивание пристеночной брюшины через грыжевые ворота с выхождением в грыжевой мешок части внутренностей живота.

Появление грыжи обусловливается причинами общего и местного характера (возраст, пол, особенности телосложения; повышение внутрибрюшного давления вследствие запоров, кашля; ослабление брюшной стенки в результате ее растяжения и истончения при повторных беременностях, травмах; снижение мышечного тонуса при параличах, в старческом возрасте и т. д.). По месту выхода (рис. 1, *а*) грыжи разделяют на паховые, пупочные, грыжи белой линии и сухожильных перемычек прямых мышц живота, поясничные, запирательные, седалищные, промежностные.

Со временем грыжа становится больше в размерах, увеличивая риск опасных для жизни осложнений – ущемления в грыжевых воротах внутренних органов. Грыжи живота требуют обязательного хирургического лечения и являются самыми распространенными случаями, требующими хирургического вмешательства (10–21% всех оперативных вмешательств). Для хирургического лечения различных грыж живота в настоящее время уже предложено свыше 300 оперативных способов и модификаций [1]. С целью устранения грыжевого дефекта разработаны многочисленные методики – от простых аутопластических способов за счет собственных тканей больного до сложных реконструктивных операций с использованием биологических и искусственных материалов. Однако, как показывает клинический опыт, ни один из предложенных способов не дает гарантии от рецидивов грыж [2, 3].

Проблема заключается в выборе обоснованных методов лечения при различных видах грыж, с одной стороны, доступных для населения, с другой стороны, порождающих минималь-



■ ***Рис. 1.*** *Схемы локализации, образования и развития грыж: а – места выхода грыж (1 – паховые, 2 – бедренные; 3 – пупочные; 4 – белой линии; 5 – спигелиевой линии); б – г –образование грыжи белой линии живота (б – предбрюшинная липома; в –образование грыжевого мешка; г – сформировавшаяся грыжа)*

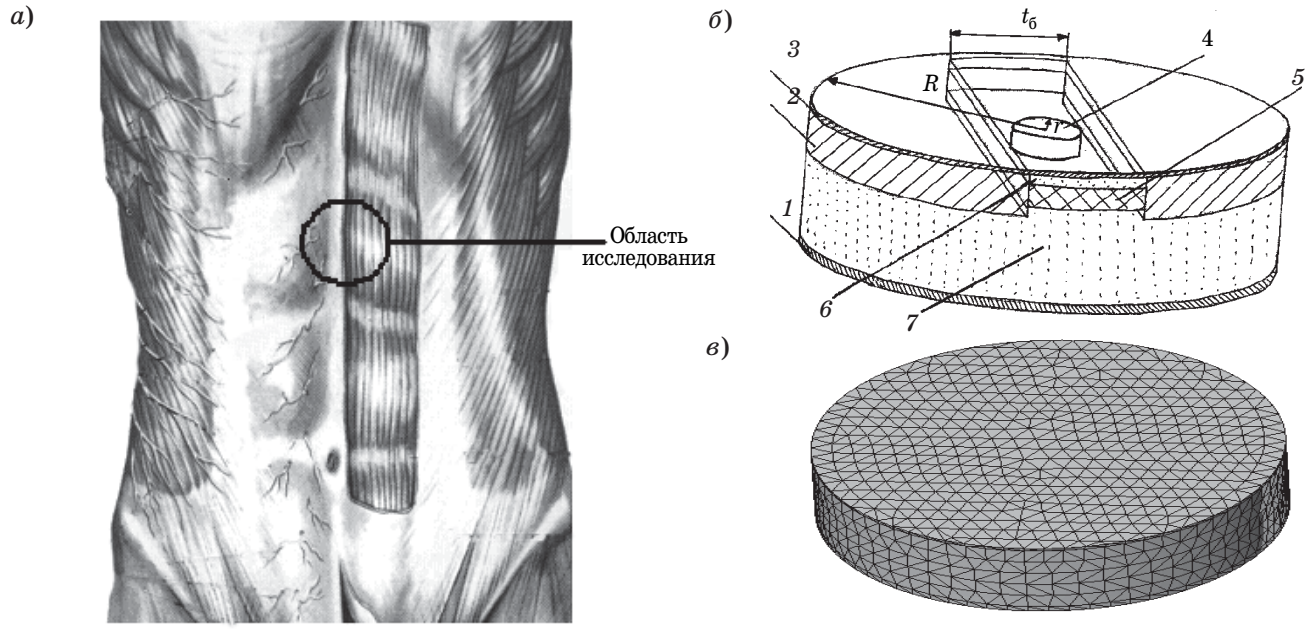
ную частоту рецидивов. Создание новых надежных оперативных методик, выбор технологии хирургического вмешательства в каждом конкретном случае невозможно осуществить без применения методов предоперационной диагностики результатов операций.

Решение проблемы – предоперационный анализ напряженно-деформированного состояния структур тканей в герниосистемах и структур, задействованных в герниопластике при различных способах и методах операций. Для проведения анализа необходимо построить расчетные схемы для биологических структур, сопрягаемых при операциях, и применяемого шовного материала, построить математические и компьютерные модели, учитывающие реальную геометрию и механические характеристики структур, и провести вычисления.

На примерах патологий белой линии живота проиллюстрируем возможности компьютерного моделирования при: 1) анализе условий развития грыжи; 2) определении критического состояния при развитии грыжи; 3) сравнительном предоперационном анализе результатов герниопластик, выполненных по разным технологиям.

Белая линия представляет собой фиброзную пластинку, простирающуюся по передней срединной линии от мечевидного отростка до лобкового симфиза. Белая линия отличается большой прочностью. Она содержит лишь тонкие ветви кровеносных сосудов, поэтому при проведении разрезов вдоль белой линии во время операции почти не отмечается кровотечения. Этим пользуются хирурги, когда необходимо создать широкий доступ к органам брюшной полости и таза.

Первопричиной образования срединных грыж является расширение белой линии живота. При этом в белой линии может возникать «анатомическое неустройство» в виде щелей, образующихся перекрещивающимися волокнами апоневрозов. Эти щели обычно заполнены жировой тканью. Вначале грыжи не имеют грыжевого мешка и представляют собой выпячивания предбрюшинного жира, которые называют предбрюшинными липомами (рис. 1, *б*). Такие грыжи называют скрытыми. В дальнейшем в грыжевые ворота вместе с жировой тканью втягивается в виде конуса прилежащий отдел париетальной брюшины (рис. 1, *в*). Если грыжа продолжает увеличиваться, то образуется уже настоящий гры-



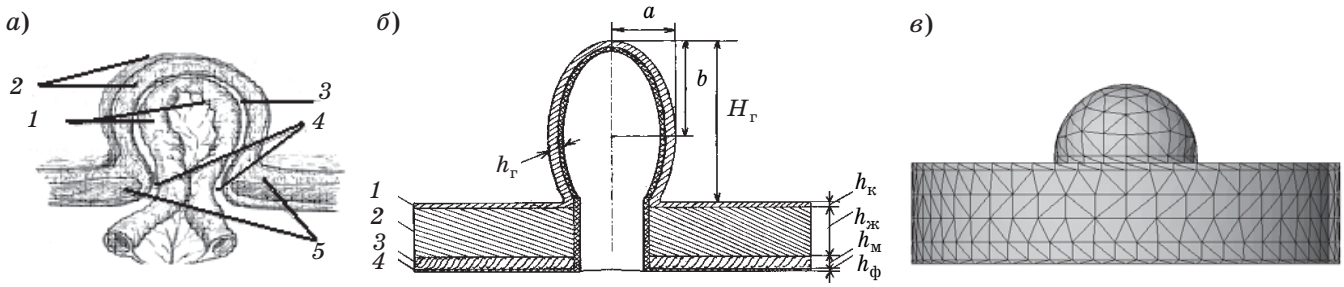
■ ***Рис. 2.*** *Моделирование и исследование развития патологического образования в белой линии живота: а – строение передней брюшной стенки (ОИ – область исследования); б – расчетная схема (1 – кожа; 2 – мышцы; 3 – брюшина; 4 – патологическое отверстие; 5 – белая линия живота; 6 – предбрюшинно-жировая клетчатка; 7 – подкожно-жировая клетчатка; R – радиус рассматриваемой области; $t_б$ – ширина белой линии); в – конечно-элементная модель*

жевой мешок (рис. 1, *г*) [4]. Диагностика при скрытых грыжах белой линии живота затруднительна, так как грыжевое выпячивание не определяется, и правильный диагноз может быть установлен лишь во время оперативного вмешательства [2].

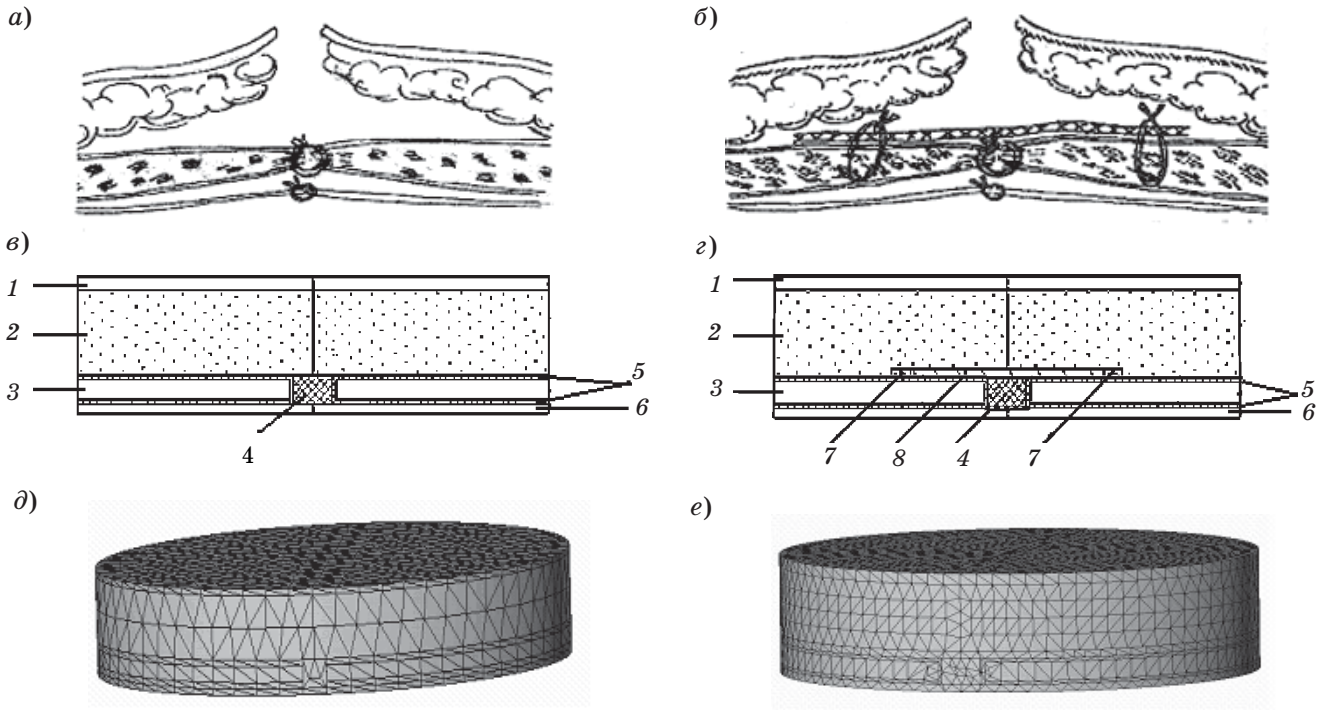
Расчетные схемы для исследования напряженно-деформированного состояния в патологически измененном участке белой линии живота построены при следующих допущениях: 1) материалы мышечно-апоневротических структур и имплантатов – однородны и изотропны; 2) среда сплошная; 3) начальные напряжения в структурах и имплантатах отсутствуют; 4) белая линия жестко закреплена по торцам; 5) усилия прикладываются к мышцам; 6) размеры исследуемой области выбраны такими, что условия закрепления не оказывают влияние на напряженно-деформированное состояние структур в области патологии. Математическая модель строится в рамках механики трехмерного тела при использовании численного метода – метода конечных элементов (КЭ). Для анализа используются линейные тетраэдальные элементы. Компьютерные модели биологических структур в герниосистемах и при герниопластике реализованы в среде геометрической программы SolidWorks и программы Cosmos для расчета напряжений в телах сложной формы. При расчете биологической структуры как трехмерного тела рассматривается неоднородное напряженное состояние. При оценке предельных состояний используем гипотезу энергоформообразования (критерий Мизеса).

Расчетная схема для исследования развития грыжи белой линии живота (рис. 2, *а*) приведена на рис. 2, *б*. Приняты следующие обозначения: r – радиус патологического образования; $h_к$ – толщина кожи; $h_ж$ – толщина подкожно-жировой клетчатки; $h_м$ – толщина мышц; $h_б$ – толщина белой линии; $h_ф$ – толщина фасции. Число конечных элементов (рис. 2, *в*) оказывает влияние на результаты вычислений до значения 35 тыс. элементов, далее влияние на точность не значительно, а время расчета существенно увеличивается. Поэтому для достижения максимальной точности при минимальных затратах времени на расчеты было выбрано разбиение на 35 тыс. тетраэдальных элементов.

По результатам вычислений построены зависимости экстремальных значений напряжений по Мизесу и перемещений в окрестности патологического образования (R = 60 мм; $h_к$ = 3 мм; $h_ж$ = 15 мм; $h_м$ = 5 мм; $h_ф$ = 1 мм; $d_п$ = 10 мм; $t_б$ = 20 мм; $h_б$ = 5 мм; $E_б$ = 0,8 МПа; $E_к$ = 25 МПа; $E_ж = 10^3$ Па; $E_м$ = 0,5 МПа; $E_ф$ = 3 МПа ; коэффициенты Пуассона материала биологических структур $\nu_б$ = 0,4) при изменении величин модуля нормальной упругости белой линии и растягивающих мышечных усилий на развитие грыжи. Из полученных зависимостей видно, что при увеличении модуля нормальной упругости белой линии в 2 раза значения напряжений и перемещений увеличиваются в 1,5



■ ***Рис. 3.*** *Исследование состояния грыжи белой линии живота: а – схема грыжи живота; (1 – грыжевое содержимое; 2 – наружные грыжевые оболочки: кожа, подкожно-жировая клетчатка, фасция; 3 – грыжевый мешок; 4 – грыжевые ворота; 5 – мышцы); б – расчетная схема; в – конечно-элементная модель*

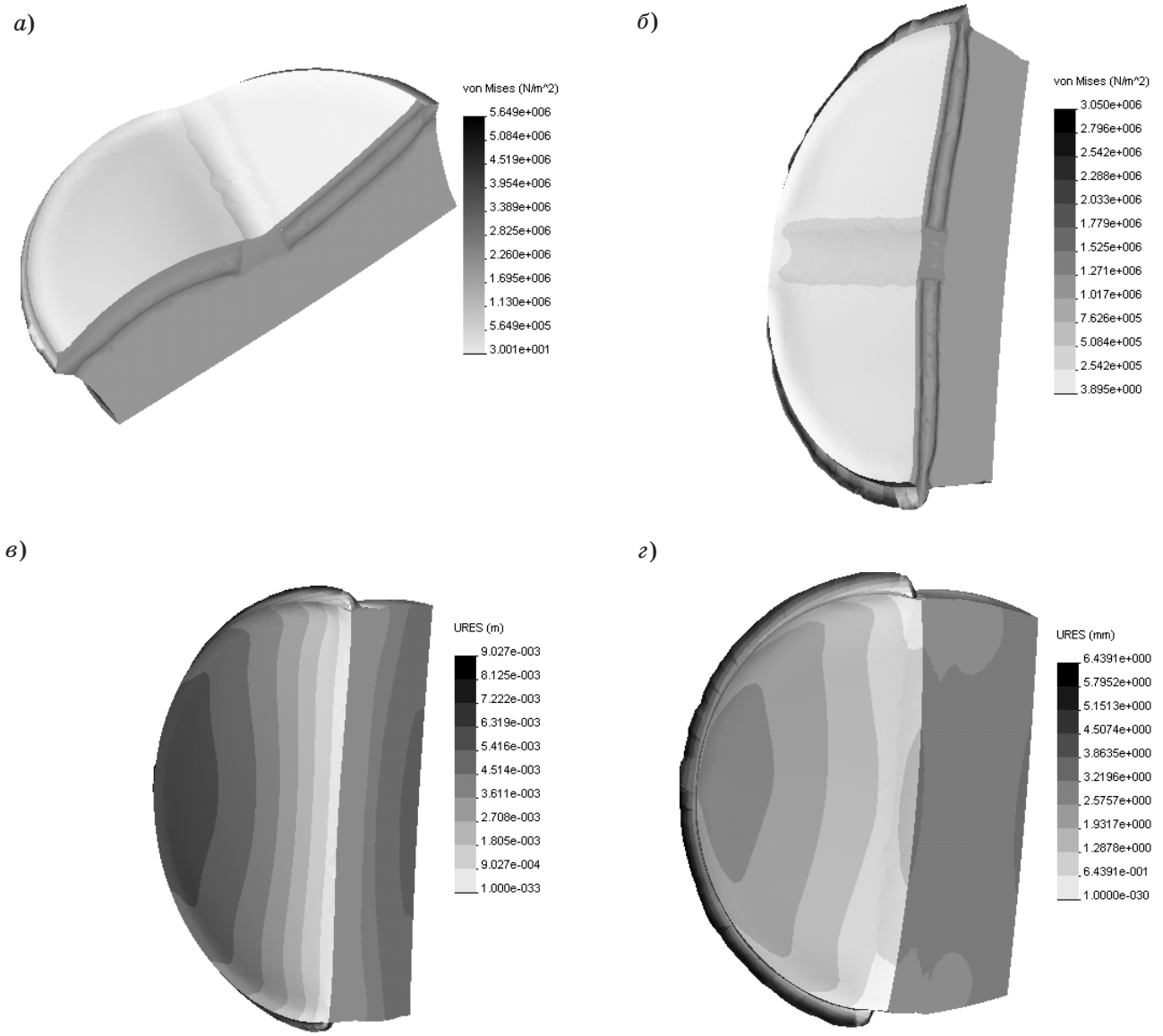


■ ***Рис. 4.*** *Герниопластика с натяжением: а – местными тканями; б – местными тканями с применением сетчатого имплантата; в, г – расчетные схемы (1 – кожа; 2 – подкожно-жировая клетчатка; 3 – мышцы; 4 – шов белой линии; 5 – апоневротические футляры мышц; 6 – брюшина; 7 – швы сетчатого имплантата; 8 – сетчатый имплантат); д, е – конечноэлементные модели*

раза. Подобные вычисления, учитывающие индивидуальные особенности пациента (геометрические параметры и механические свойства биологических структур) позволяют прогнозировать характер развития конкретного патологического образования.

Число пациентов с ущемленными грыжами достигает 15–18% от общего количества грыженосителей. Послеоперационная летальность при этом неотложном состоянии составляет 3–8%, а для больных старше 60 лет она возрастает до 16–20% [1, 2]. В дополнение к схематизации, принятой для расчетной схемы, отражающей раскрытие грыжи (см. рис. 2, *б*), для исследования критического состояния при ущемлении грыжи (рис. 3, *а*) введены следующие допущения: 1) купол грыжи – эллипсоид с полуосями a и b; 2) толщина стенки грыжевого мешка $h_г$, высота $H_г$ (рис. 3, *б*). Разбиение проводилось на 50 тыс. конечных элементов.

Построены зависимости экстремальных значений напряжений по Мизесу и перемещений в окрестности ущемляемой грыжи при a = 20 мм; b = 40 мм; $h_г$ = 5 мм; $H_г$ = 80 мм. Из полученных результатов видно, что при увеличении модуля нормальной упругости белой линии в 2 раза значения напряжений и перемещений уменьшаются на 10%.

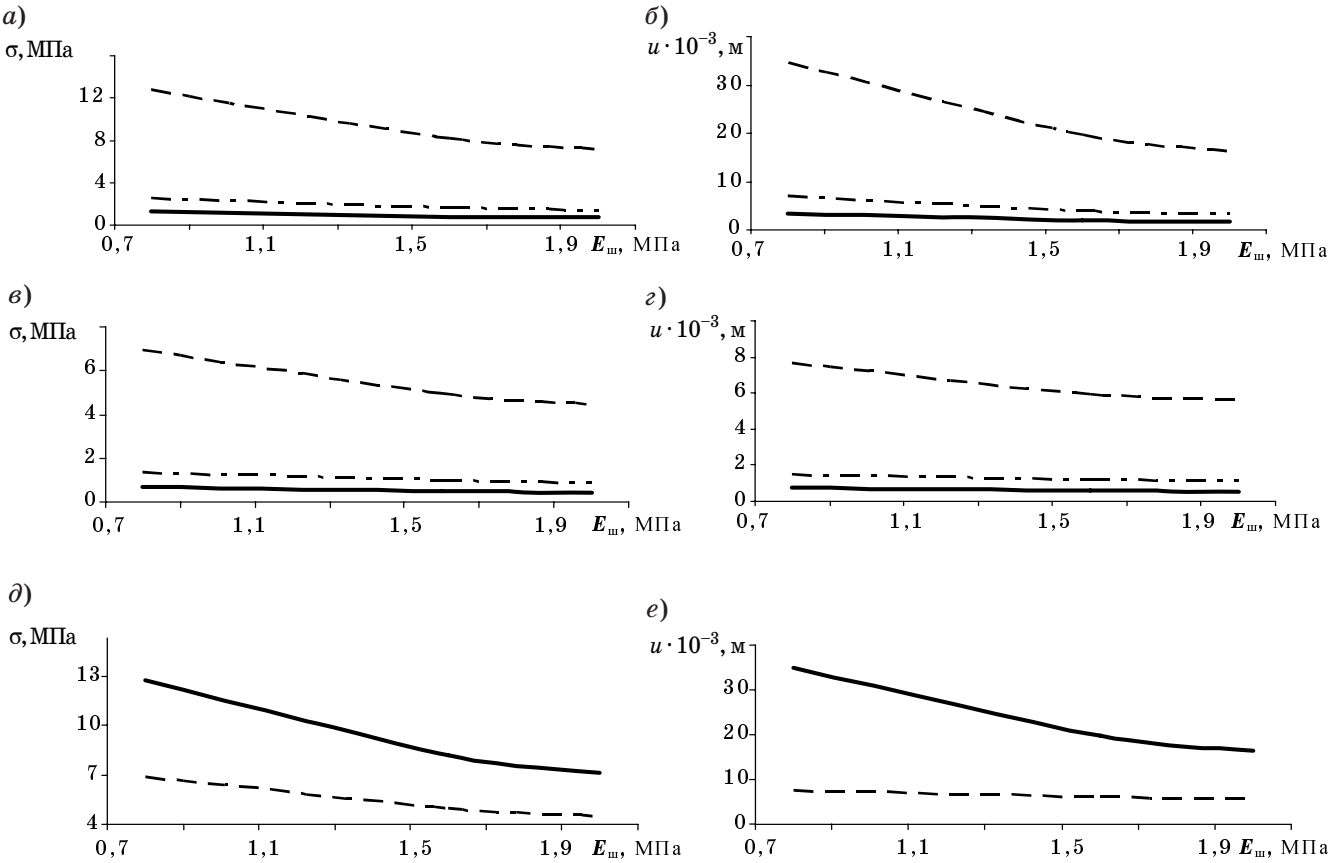


■ ***Рис. 5.*** *Эпюры напряжений (а, б) и перемещений (в, г) в биологических структурах при герниопластике с натяжением тканей (а, в) и при герниопластике с натяжением тканей при дополнительном усилении синтетическим протезом (б, г)*

Все методы хирургического лечения грыж могут быть разделены на два типа. При операциях первого типа соединение тканей происходит с их натяжением –стягиванием краев отверстия; при втором типе операций натяжение тканей отсутствует – отверстие закрывают имплантатом сверху или снизу. Разработано несколько методов герниопластики с натяжением тканей и множество модификаций этих методов. Пластика сопровождается довольно значительным натяжением сшитых тканей. В дальнейшем это может привести к растяжению рубцовой ткани и образованию рецидива грыжи.

Приведены схемы герниопластик с натяжением (рис. 4, *а–е*) и эпюры напряжений и перемещений биологических структур для этих методов (рис. 5, *а–г*).

В дополнение к ранее рассмотренным расчетным схемам (см. рис. 2, 3) введены следующие допущения: 1) материалы шва и сетчатого синтетического имплантата – изотропные и однородные с конструктивными модулями нормальной упругости соответственно $E_{ш} = 1{,}6$ МПА; $E_{и} = 1$ МПа; 2) коэффициенты Пуассона материала шва и синтетического имплантата соответственно $\nu_{ш} = 0{,}4$; $\nu_{и} = 0{,}35$; 3) толщина и ширина прямоугольной сетки соответственно $h_{с} = 2$ мм; $b_{с} = 10$ мм; ширина шва $b_{ш} = 10$ мм; 4) усилие натяжения трансплантата $p_{ат} = 5 \cdot 10^{5}$ Па.



■ ***Рис. 6.*** *Зависимости экстремальных значений напряжений σ (а, в, д) и перемещений и (б, г, е) в коррегируемых биологических структурах белой линии живота от модуля нормальной упругости шва $E_ш$ для двух методов пластики при заданных мышечных усилиях (а–г – ——— – при $p = 5 \cdot 10^4$ Па; — — – при $p = 5 \cdot 10^5$ Па; – – – при $p = 10^5$ Па) и сравнение этих методов пластики при $p = 5 \cdot 10^5$ Па (д, е – ——— – встык; – – – onlay)*

Зависимости экстремальных значений напряжений и перемещений в корригируемых биологических структурах белой линии живота от модуля нормальной упругости шва, построенные для двух методов пластики (рис. 6), наглядно иллюстрируют значительное снижение напряженно-деформированного состояния в области шва при пластике с использованием сетчатого синтетического имплантата.

Следовательно, проведение герниопластики с применением сетчатых синтетических имплантатов снижает не только травматичность операции, но и вероятность возникновения рецидива, тем самым повышая ее надежность.

Литература

1. **Тоскин К. Д., Жебровский В. В.** Грыжи брюшной стенки. М.: Медицина, 1990. 245 с.
2. Ненатяжная герниопластика / Под ред. В. Н. Егиева. М.: Медпрактика-М, 2002. 146 с.
3. Основные грыжесечения / В. И. Ороховский, И. Гастингер, И. Хорнтрих, Ш. Шваниц. Ганновер; Донецк; Коттбус: МУНЦЭХ, КИТИС, 2000. 328 с.
4. Послеоперационные вентральные грыжи / А. И. Мариев, Н. Д. Ушаков, В. А. Шорников, А. М. Иванова / ПетрГУ. Петрозаводск, 2003. 282 с.

УДК 681.513:541.13

ПАРАМЕТРИЧЕСКАЯ ИДЕНТИФИКАЦИЯ ЭЛЕКТРОХИМИЧЕСКОГО ПРОЦЕССА НА ОСНОВЕ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

А. М. Мендельсон,
аспирант
Е. Н. Бендерская,
канд. техн. наук
Санкт-Петербургский государственный политехнический университет
Р. А. Тенно,
доктор техн. наук
Хельсинкский технологический университет

Рассматривается задача идентификации параметров двухсторонней модели электрохимического процесса. Использован подход, основанный на оценке параметров линеаризованной модели с последующим пересчетом линейных параметров в нелинейные. Для учета нелинейных ограничений при оценивании предлагается использовать генетические алгоритмы.

The problem of identifying the parameters of an electrochemical process is considered. The solution is based on genetic optimisation of linear coefficients subject to nonlinear constraints and recalculation of linear parameters to nonlinear parameters. The simulation results show the advantages of this method.

Введение

В задаче электроосаждения методом химического восстановления модель процесса [1], состоящего из четырех реакций, задается в виде

$$\begin{cases} i_1 = i_{01}\left[e^{k_1\alpha_{a1}(\Phi-U_1)} - e^{-k_1(1-\alpha_{a1})(\Phi-U_1)}\right] \\ i_2 = i_{02}\left[e^{k_2\alpha_{a2}(\Phi-U_2)} - e^{-k_2(1-\alpha_{a2})(\Phi-U_2)}\right] \\ i_3 = i_{03}\left[e^{k_3\alpha_{a3}(\Phi-U_3)} - e^{-k_3(1-\alpha_{a3})(\Phi-U_3)}\right] \\ i_4 = i_{04}\left[e^{k_4\alpha_{a4}(\Phi-U_4)} - e^{-k_4(1-\alpha_{a4})(\Phi-U_4)}\right] \end{cases}, \quad (1)$$

где $i_j, j=\overline{1,\ 4}$ – токи, соответствующие каждой реакции; $i_{0j}, j=\overline{1,\ 4}$ – плотности токов, соответствующие каждой реакции; $k_j, j=\overline{1,\ 4}$ – известные электрохимические константы; $\alpha_{aj}, j=\overline{1,\ 4}$ – коэффициенты наклона, соответствующие каждой реакции; $U_j, j=\overline{1,\ 4}$ – электродные потенциалы реакций; Φ – смешанный потенциал.

Математическая модель (1) основана на уравнении электрохимической кинетики Батлера–Фольмера [2]. Электродные потенциалы определяются на основе уравнений Нернста [2, 3]. Значения токов и электродных потенциалов предполагаются наблюдаемыми. Цель идентификации – определение плотностей токов и коэффициентов наклона. При наличии соответствующих измерений смешанного потенциала данная задача может быть решена обычным нелинейным методом наименьших квадратов [4]. К сожалению, практически такие измерения являются достаточно неточными и дорогостоящими. Поэтому актуальной является разработка алгоритма идентификации неизвестных параметров модели (1) при наличии, максимум, знаний о диапазоне изменения смешанного потенциала.

Линеаризация модели

Для идентификации будем использовать подход, основанный на сведении исходной задачи к задаче квадратичного программирования [5]. Введем следующее обозначение:

$$\eta_j = \Phi - U_j,\ j=\overline{1,\ 4}.$$

Разложение в ряд Тейлора каждого уравнения модели (1) ведет к следующей линейной зависимости токов реакций от потенциалов:

$$i_j(\Phi, U_j) = i_j(\Phi_0, U_{0j}) + i'_j(\Phi_0, U_{0j})(\eta_j - \eta_{0j}) = \\ = a_j + b_j(\eta_j - \eta_{0j}),\ j = \overline{1,\ 4}.$$

Линеаризованная модель процесса выглядит следующим образом:

$$\begin{cases} i_1 = a_1 + b_1((\Phi - \Phi_0) - (U_1 - U_{01})) \\ i_2 = a_2 + b_2((\Phi - \Phi_0) - (U_2 - U_{01})) \\ i_3 = a_3 + b_3((\Phi - \Phi_0) - (U_3 - U_{01})) \\ i_4 = a_4 + b_4((\Phi - \Phi_0) - (U_4 - U_{01})) \end{cases}. \qquad (2)$$

Смешанный потенциал удовлетворяет уравнению консервации $i_1 + i_2 + i_3 + i_4 = 0$. Этот факт позволяет привести модель (2) к следующему виду:

$$\begin{cases} i_1 = \gamma_1 + \beta_{21}\Delta U_{21} + \beta_{31}\Delta U_{31} + \beta_{41}\Delta U_{41} \\ i_2 = \gamma_2 + \beta_{12}\Delta U_{12} + \beta_{32}\Delta U_{32} + \beta_{42}\Delta U_{42} \\ i_3 = \gamma_3 + \beta_{13}\Delta U_{13} + \beta_{23}\Delta U_{23} + \beta_{44}\Delta U_{43} \\ i_4 = \gamma_4 + \beta_{14}\Delta U_{14} + \beta_{24}\Delta U_{24} + \beta_{34}\Delta U_{34} \end{cases},$$

$$\Delta U_{ij} = U_i - U_j. \qquad (3)$$

Параметры γ_i, β_i системы (3) очевидным образом связаны с параметрами a_i, b_i системы (2). На коэффициенты системы (3), исходя из (2), накладываются следующие линейные и нелинейные ограничения:

$$\sum_{i=1}^{4}\gamma_i = 0,\ \beta_{ij} = \beta_{ji}, \qquad (4)$$

$$\beta_{ij}\beta_{kl} = \beta_{ik}\beta_{jl}. \qquad (5)$$

Заметим, что при отсутствии нелинейных ограничений система (3) может быть идентифицирована обычным методом наименьших квадратов [5]. Линейное оценивание не позволяет учесть нелинейные ограничения (5). В хорошо обусловленном случае и при малой ошибке линеаризации это не принципиально, так как ограничения (5) в итоге выполняются. Однако в общем случае ошибка линеаризации и ошибки измерений могут вести к сильным разбросам параметров, что не позволяет однозначно определить коэффициенты модели (3). Для того чтобы решить указанную проблему, далее предлагается использовать генетические алгоритмы. Здесь же заметим, что обратное преобразование осуществляется в два этапа. Сначала по оценкам параметров γ_i, β_i рассчитываются также линейные параметры a_i, b_i. Затем преобразование линейных параметров в нелинейные осуществляется с использованием формул

$$\alpha_{aj}^{\text{opt}} = \frac{b_j}{a_j k_j} - \frac{e^{-k_j\eta_{0j}}}{1 - e^{-k_j\eta_{0j}}},$$

$$i_j^{\text{opt}} = \frac{a_j}{e^{k_j\alpha_{aj}^{\text{opt}}\eta_{0j}}\left(1 - e^{-k_j\eta_{0j}}\right)},\ j = \overline{1,\ 4}. \qquad (6)$$

Разрешимость уравнений (6) требует знания точки линеаризации. Для этого необходимо знание средних значений электродных потенциалов, легко вычисляемых на основе измерений, и среднего значения смешанного потенциала, предполагаемого известным. Очевидно, предложенный алгоритм дает адекватную параметрическую оценку только в случае малых отклонений потенциалов от точки разложения. В связи с тем, что целью управления в задаче электроосаждения [1] является поддержание постоянной толщины и постоянного состава сплава, что равносильно поддержке примерно постоянных токов и смешанного потенциала, предположение о малых отклонениях является допустимым. Однако в общем случае для получения достоверных результатов идентификации необходима предварительная кластеризация данных с последующим применением алгоритма для каждого кластера.

Идентификация на основе генетических алгоритмов

Необходимо определить коэффициенты модели (3) при наличии линейных (4) и нелинейных (5) ограничений. Целевая функция, подлежащая минимизации, имеет следующий вид:

$$J = \min\left\{\left(\mathbf{i}_1 - \mathbf{X}_1\boldsymbol{\theta}_1^T\right)^2 + \left(\mathbf{i}_2 - \mathbf{X}_2\boldsymbol{\theta}_2^T\right)^2 + \right. \\ \left. + \left(\mathbf{i}_3 - \mathbf{X}_3\boldsymbol{\theta}_3^T\right)^2 + \left(\mathbf{i}_4 - \mathbf{X}_4\boldsymbol{\theta}_4^T\right)^2\right\}, \qquad (7)$$

где $\mathbf{X}$ и $\boldsymbol{\theta}$ – регрессоры и векторы параметров из модели (3).

Алгоритм идентификации системы с нелинейными ограничениями с использованием генетических алгоритмов состоит в следующем. Каждая хромосома состоит из четырех параметров, при этом каждый параметр, в свою очередь, кодируется бинарным словом. При этом заранее учтено, что коэффициенты могут быть только положительными, что означает отсутствие гена знака. Возможный вариант задания хромосомы – коэффициенты β_{12}, β_{13}, β_{14}, β_{23}. Два других параметра β_{24}, β_{34} могут быть получены использованием нелинейных ограничений (5). Оставшиеся

■ ***Таблица 1.*** *Результаты идентификации по полному набору данных*

Исходные параметры		Число обусловленности $5.1 \cdot 10^6$		Число обусловленности $4.7 \cdot 10^4$	
i_0, мА	α	i_0, мА	α	i_0, мА	α
17.7	0.53	17.1	0.61	17.7	0.51
0.50	0.38	0.48	0.36	0.50	0.38
2.51	0.41	2.11	0.36	2.6	0.43
1.61	0.54	1.56	0.53	1.6	0.54

■ ***Таблица 2.*** *Результаты идентификации по двум кластерам*

Первый кластер		Второй кластер	
i_0, мА	α	i_0, мА	α
17.7	0.53	17.7	0.53
0.50	0.38	0.50	0.38
2.50	0.41	2.50	0.41
1.61	0.54	1.61	0.54

три параметра γ_1, γ_2, γ_3 могут быть определены стандартным методом наименьших квадратов с линейными ограничениями [6]:

$$\boldsymbol{\theta}_{\text{constr}} = \boldsymbol{\theta} - \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{A}\left(\mathbf{A}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{A}^T\right)^{-1}\left(\mathbf{A}\boldsymbol{\theta} - \mathbf{b}\right).$$

Полный алгоритм идентификации на основе генетических алгоритмов следующий:

1) генерация начальной популяции;

2) вычисление функции предпочтения для каждой хромосомы на основе целевой функции (7);

3) скрещивание и мутация;

4) формирование новой популяции;

5) если количество генераций больше заданного или ошибка меньше порога – конец, иначе – переход к шагу 2.

Результатом идентификации является лучшая хромосома последней популяции.

Таким образом, при указанном определении хромосом поиск всегда ведется в допустимой области ограничений на параметры. По сравнению с классическими методами оптимизации, подход на основе генетических алгоритмов позволяет исключить такую величину, как невязка ограничений, которая определяет, удовлетворяют ли параметры ограничениям. В рассматриваемом случае эта невязка всегда равна нулю.

Пример идентификации

Рассмотрим электрохимический процесс из четырех реакций, плотности токов и коэффициенты наклона которого приведены в табл. 1. Электродные потенциалы моделировались на основе уравнений Нернста. В генетическом алгоритме были использованы следующие параметры: количество популяций 100, размер популяции 1000, вероятность мутации 0.05. Каждая переменная была закодирована 16-битным словом. Тестирование алгоритма идентификации проводилось в двух режимах: в первом случае электродные потенциалы вычислялись строго по уравнениям Нернста, а во втором – для повышения степени обусловленности к ним добавлялся малый псевдослучайный сигнал. Диапазоны изменения электродных потенциалов составляют [–895, –865], [–483, –477], [–365, –333], [–288.74, –288.73] мВ соответственно. Диапазон изменения смешанного потенциала [–752, –728] мВ. В соответствии с этим линеаризация производилась в следующей точке: $U_{01} = -880$ мВ, $U_{02} = -480$ мВ, $U_{03} = -349$ мВ, $U_{04} = -288.735$ мВ, $\Phi_0 = -740$ мВ. Размерность окна данных при оценивании выбиралась в соответствии с теорией планирования научного эксперимента и составила 1000 измерений. Результаты идентификации представлены в табл. 1. На однопроцессорной ЭВМ AMD Duron 800 MHz, 512 RAM расчет с указанными выше размером популяции, количеством популяций и размером окна данных, определяющих быстродействие генетического алгоритма, занимает в среднем порядка 3 мин. В табл. 2 представлены результаты идентификации с предварительным разбиением данных на два кластера. Сравнение табл. 1 и 2 показывает, что кластеризация улучшает качество параметрических оценок. Это объясняется уменьшением ошибки разложения.

Заключение

В работе был предложен алгоритм идентификации нелинейных параметров двухсторонней модели электрохимического процесса на основе линеаризации с использованием генетических алгоритмов. Основными преимуществами алгоритма являются простота и однозначность получения оценок даже в плохообусловленном случае.

Литература

1. **Paunovic M., Sclesinger M.** Fundamentals of electrochemical deposition. John Wiley & Sons, 1998. 301 p.
2. **Newman J.** Electrochemical systems. Wiley-Interscience, 2004. 672 p.
3. **Дамаскин Б. Б., Петрий О. А., Цирлина Г. А.** Электрохимия. М.: Химия, 2001. 624 с.
4. http://mathworld.wolfram.com/NonlinearLeastSquaresFitting.html.
5. **Мендельсон А. М.** Идентификация частично наблюдаемого электрохимического процесса на основе линеаризации // Идентификация систем и задачи управления: Тр. V Междунар. конф. М., 2006. С. 846–848.
6. **Soderstom T., Stoica P.** System Identification. New York: Prentice Hall, 1989. 612 p.

ПАМЯТКА ДЛЯ АВТОРОВ

Поступающие в редакцию статьи проходят обязательное рецензирование.

При наличии положительной рецензии статья редактируется и рассматривается редакционной коллегией. Принятая в печать статья направляется автору для согласования редакторских правок. После согласования автор представляет в редакцию окончательный вариант текста статьи.

Процедуры согласования текста статьи могут осуществляться как непосредственно в редакции, так и по e-mail (80x@mail.ru).

При отклонении статьи редакция представляет автору мотивированное заключение и рецензию. При необходимости доработать статью – рецензию.

Редакция журнала напоминает, что ответственность за подбор, достоверность и точность фактов, экономико-статистических и технических показателей, собственных имен и прочих сведений, а также за то, что в материалах не содержится сведений, не подлежащих открытой публикации, несут рекламодатели.

Федеральное космическое агентство
Федеральное агентство по промышленности
Федеральное агентство по образованию
Российская академия наук
Российская академия космонавтики им. К.Э.Циолковского
Российская академия авиации и воздухоплавания
Московский авиационный институт

V МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ «АВИАЦИЯ И КОСМОНАВТИКА-2006»
23–26 октября 2006 года

Место проведения конференции: Московский авиационный институт
Адрес: 125993, г. Москва, А-80, ГСП-3, Волоколамское шоссе, д. 4

Организатор конференции

Московский авиационный институт

Общая информация

Международная конференция «Авиация и космонавтика» проводится ежегодно. В 2005 году конференция проходила в рамках совместных мероприятий с авиасалоном МАКС-2005. В конференции участвовало более 200 российских и зарубежных предприятий, организаций и институтов. Во время конференции заслушивается около 500 докладов, посвященных современным технологиям и новым возможностям в широком спектре аэрокосмической тематики, включающей в себя как прикладные, так и фундаментальные вопросы. Число участников конференции превышает 1100 человек. Конференция «Авиация и космонавтика» позволяет полнее использовать научно-технический и производственно-технологический потенциал российских предприятий и их зарубежных коллег.

Направления работы конференции

Авиационная безопасность и безопасность полетов
Анализ и синтез сложных систем
Аэродинамика
Аэрокосмическое образование
Баллистика, динамика и управление движением
Беспилотные летательные аппараты
Бортовые радиоэлектронные комплексы
Вертолетостроение
Высокоточное позиционирование с помощью GNSS
Динамика и прочность конструкций ЛА
Интернет-технологии и средства виртуальной реальности в науке и образовании
Информационные технологии радиоэлектронных средств и их менеджмент
Исследование и разработка университетских нано- и пико-спутников Земли
Компьютерный дизайн
История развития и сотрудничества отечественной и зарубежной авиации и космонавтики
Компьютерные и информационные технологии на транспорте
Конкурентоспособность российских фирм
Материаловедение
Многосредные аппараты и системы
Прикладные и математические методы
Прогнозирование, экономика, конкурентоспособность авиационной техники
Проектирование, технологии и производство
Ракетно-космические аппараты и системы
Робототехнические системы
Самолетостроение
Системы жизнеобеспечения
Системы управления движением, ориентации и навигации
Тепловые двигательные установки
Тепловые режимы, теплозащита, терморегулирование
Философское и социально-гуманитарное обоснование развития авиации и космонавтики
Управление качеством, экология, экономика, коммерциализация и маркетинг
Электроракетные двигатели и энергоустановки
Электроэнергетические, электромеханические и биотехнические системы
Студенческая и школьная секция

Контрольные сроки

Заявки принимаются до 15 июля 2006 г.

Дополнительная информация и справки

Тел.: (495) 195-94-83, (906) 717-83-91
E-mail: aviacosmos_2006@mai.ru
http://www.mai.ru/conf/aerospace/

SIMILAR NoE
INTAS
ELSNET
ISCA
Санкт-Петербургский институт информатики и автоматизации РАН

XI МЕЖДУНАРОДНАЯ КОНФЕРЕНЦИЯ «РЕЧЬ И КОМПЬЮТЕР» SPECOM-2006 26–29 июня 2006

Место проведения конференции: Россия, Санкт-Петербург,
Санкт-Петербургский институт информатики и автоматизации РАН
Адрес: Россия, СПИИРАН, Санкт-Петербург, 14 линия 39, 199178

Организатор

Санкт-Петербургский институт информатики и автоматизации РАН (СПИИРАН)

Цель конференции

Обсуждение проблем и достижений в области естественного взаимодействия человека с компьютером

Направления работы конференции

Многомодальный анализ и синтез
Распознавание и понимание речи
Обработка естественного языка
Цифровая обработка сигналов
Идентификация диктора и языка
Синтез речи
Восприятие речи и нарушения в речи
Ресурсы языка и речи
Прикладные системы для человеко-машинного взаимодействия и др.

Контрольные сроки

Заявка на специальную сессию до 1 февраля 2006
Представление текста доклада до 1 марта 2006
Подтверждение принятия текста до 31 марта 2006
Ранняя регистрация 15 апреля 2006

Культурная программа конференции

В этот раз конференция проводится в Санкт-Петербурге во время белых ночей, когда город особенно привлекателен. Летом круглосуточно организуются водные экскурсии по каналам и рекам города. Вы сможете наблюдать разведенные мосты через Неву и дворцы в ночном освещении, а днем побывать в янтарной комнате Царского села и полюбоваться фонтанами в Петродворце.

Дополнительная информация

Организационный Комитет SPECOM-2006, СПИИРАН, Санкт-Петербург, 14 линия 39, 199178
Тел.: +7 (812) 328 70 81
Факс: + 7(812) 3284450
E-mail: specom@iias.spb.su
Web: http://www.specom.nw.ru

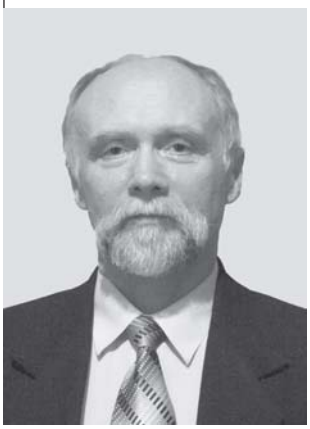
БЕГУН
Петр
Иосифович

Профессор, заместитель заведующего кафедрой прикладной механики и инженерной графики Санкт-Петербургского государственного электротехнического университета, академик Академии медико-технических наук России.
В 1962 году окончил Ленинградский военно-механический институт.
В 2003 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором 250 научных публикаций.
Область научных интересов –

БЕНДЕРСКАЯ
Елена
Николаевна

Доцент кафедры автоматики и вычислительной техники Санкт-Петербургского государственного политехнического университета.
В 1993 году окончила Санкт-Петербургский государственный технический университет.
В 1996 году защитила диссертацию на соискание ученой степени кандидата технических наук.
Является автором 30 научных публикаций и соавтором трех учебных пособий.
Область научных интересов – системы искусственного интеллекта, распознавание образов, интеллектуализация систем функционального диагностирования, системный анализ и теория принятия решений.

ГОРДЕЕВ
Александр
Владимирович



Профессор, заведующий кафедрой компьютерных систем проектирования Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1975 году окончил Ленинградский институт авиационного приборостроения.
В 1998 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 80 научных публикаций.
Область научных интересов – моделирование параллельных вычислительных систем, распределенные вычисления, вы-

ИЦЫКСОН
Владимир
Михайлович

Доцент, заведующий лабораторией программно-аппаратных разработок, руководитель телекоммуникационного центра факультета технической кибернетики Санкт-Петербургского государственного политехнического университета.
В 1996 году окончил Санкт-Петербургский государственный технический университет.
В 2000 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 50 научных публикаций.
Область научных интересов – технологии разработки программного обеспечения, гетерогенные многопроцессорные вычислительные комплексы, технологии компьютерных сетей, защита информации.

КУЗИНА
Нина
Владимировна

Инженер-программист Военно-морской академии им. Н. Г. Кузнецова.
В 1976 году окончила Ленинградский политехнический институт им. М. И. Калинина.
Является автором семи научных публикаций.
Область научных интересов – информатика и имитационное моделирование.

ЛАЗАРЕВ
Сергей
Михайлович

Профессор, заведующий кафедрой и директор клиники хирургических болезней № 2, основатель и руководитель Центра пластической и реконструктивной хирургии Санкт-Петербургской государственной медицинской академии им. И. И. Мечникова.
В 1973 году окончил 1-й Ленинградский медицинский институт им. И. П. Павлова.
В 1996 году защитил диссертацию на соискание ученой степени доктора медицинских наук.
Является автором более 200 научных публикаций, в том числе соавтором двух книг.
Область научных интересов – реконструктивная хирургия внутренних органов и пластическая хирургии грудной и брюшной стенок.

ЛЕБЕДЕВА Елена Александровна

Аспирант Санкт-Петербургского государственного электротехнического университета.
В 2004 году окончила Санкт-Петербургский государственный электротехнический университет.
Является автором 18 научных публикаций.
Область научных интересов – биомеханика.

МЕНДЕЛЬСОН Александр Маркович

Аспирант кафедры автоматики и вычислительной техники Санкт-Петербургского государственного политехнического университета.
В 2002 году окончил с отличием факультет технической кибернетики Санкт-Петербургского государственного политехнического университета по специальности «Информатика и управление в технических системах».
Является автором 12 научных публикаций.
Область научных интересов – современная теория управления, теория идентификации нелинейных систем, системы искусственного интеллекта, распознавание образов, цифровая обработка сигналов.

РЫЖИКОВ Юрий Иванович

Профессор кафедры математического обеспечения ЭВМ Военно-космической академии им. А. Ф. Можайского.
Заслуженный деятель науки РФ.
В 1958 году окончил Черноморское высшее военно-морское училище им. П. С. Нахимова.
В 1969 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 200 научных публикаций, в том числе 40 учебных пособий и монографий.
Область научных интересов – теория очередей, имитационное моделирование, вычислительные методы и прикладное программирование, управление запасами, науковедение и педагогика высшей школы.

МАРКОВСКИЙ Станислав Георгиевич

Старший преподаватель кафедры вычислительных машин и комплексов Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1986 году окончил Ленинградский институт авиационного приборостроения по специальности «Электронные вычислительные машины».
Является автором 26 научных и научно-методических публикаций.
Область научных интересов – протоколы случайного множественного доступа, нетрадиционные архитектуры компьютеров.

РОЗОВ Алексей Константинович

Старший научный сотрудник Военно-морской академии им. Н. Г. Кузнецова.
В 1947 году окончил Высшее инженерно-техническое училище Военно-морского флота по специальности «Инженер-электромеханик».
В 1968 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором 27 научных публикаций.
Область научных интересов – применение статистических методов в задачах обнаружения, классификации и оценивания сигналов.

СТРУЧКОВ Игорь Вячеславович

Аспирант кафедры автоматики и вычислительной техники Санкт-Петербургского государственного политехнического университета.
В 2002 году окончил магистратуру Санкт-Петербургского государственного политехнического университета по специальности «Сети ЭВМ и телекоммуникации».
Является автором семи научных публикаций.
Область научных интересов – распределенные вычисления, высокопроизводительные вычислительные системы, технологии компьютерных сетей.

СЫРЦЕВ Алексей Николаевич

Преподаватель Военно-морской академии им. Н. Г. Кузнецова.
В 1986 году окончил Черноморское высшее военно-морское училище им. П. С. Нахимова, в 1995 году – Военно-морскую академию им. Н. Г. Кузнецова.
В 1998 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором более 30 научных публикаций.
Область научных интересов – информационно-управляющие системы и комплексы, помехозащищенность информационных каналов систем управления.

ТЮРЛИКОВ Андрей Михайлович

Доцент кафедры информационных систем Санкт-Петербургского государственного университета аэрокосмического приборостроения.
В 1980 году окончил Ленинградский институт авиационного приборостроения по специальности «Информационные системы управления».
В 1986 году защитил диссертацию на соискание ученой степени кандидата технических наук.
Является автором 60 научных публикаций.
Область научных интересов – многоабонентные системы связи, системы дистанционного обучения, протоколы передачи данных в реальном масштабе времени.

ТЕННО Роберт Альфредович

Доцент лаборатории разработки систем управления Хельсинкского технологического университета (HUT).
В 1973 году окончил Таллинский технический университет по специальности «Автоматика и телемеханика».
В 1989 году защитил диссертацию на соискание ученой степени доктора технических наук.
Является автором более 70 научных публикаций и одной монографии.
Область научных интересов – идентификация и управление частично наблюдаемыми стохастическими системами, приложения методов оценивания и управления в микроэлектронике, электрохимии и

УДК 681.516.7.015

Оптимальное правило остановки наблюдений – способ достижения наивысшей вероятности обнаружения сигналов

Розов А. К., Сырцев А. Н., Кузина Н. В. Информационно-управляющие системы, 2006. № 2. С. 2–7.

Излагается процедура применения оптимального правила остановки наблюдений для достижения наивысшей вероятности обнаружения слабых сигналов.

Список лит.: 3 назв.

UDK 681.516.7.015

An optimal rule for stopping the observations in order to maximize the probability of detecting the signals

Rozov A. K., Syrtsev A. N., Kuzina N. V. IUS, 2006. N 2. P. 2–7.

We discuss an optimal rule of stopping the observations to reach the maximal probability of finding weak signals.

Refs: 3 titles.

УДК 004.415.2.032.24

Формализм для описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени

Стручков И. В., Ицыксон В. М. Информационно-управляющие системы, 2006. № 2. С. 8–13.

Рассматриваются вопросы формализованного описания программных систем и вычислительных процессов циклической параллельной обработки данных реального времени для специализированных информационно-вычислительных комплексов. Для указанного класса систем предлагается новый способ формализованного описания – граф генераторов и преобразователей данных, который может использоваться для выделения функциональных модулей-задач, имитационного моделирования, статического и динамического распределения задач по процессорам в многопроцессорной системе.

Список лит.: 5 назв.

UDK 004.415.2.032.24

A formal approach to description of cyclic real-time parallel computational processes and software systems

Struchkov I. V., Itsykson V. M. IUS, 2006. N 2. P. 8–13.

The article addresses a formal description of cyclic real-time parallel computational processes and corresponding software systems for specialized computing complexes. A new formal approach for describing the specified class of systems is introduced: the data generator-transformer graph. This graph can be utilized for the decomposition of task modules, simulation, static and dynamic task mapping in a multiprocessor system.

Refs: 5 titles.

УДК 65.012.122

Субоптимальный алгоритм построения расписаний для иерархических вычислительных систем

Колесов Н. В., Толмачева М. В. Информационно-управляющие системы, 2006. № 2. С. 14–20.

Рассматривается субоптимальный алгоритм составления расписаний в иерархических вычислительных системах, практически не требующий перебора вариантов. Приводятся результаты исследований его эффективности на основе случайного генерирования примеров.

Список лит.: 4 назв.

UDK 65.012.122

A suboptimal scheduling algorithm for hierarchical commuting systems

Kolesov N. V., Tolmacheva M. V. IUS, 2006. N 2. P. 14–20.

We propose a suboptimal scheduling algorithm for hierarchical computing systems which requires almost no case study of schedule versions. It is based on several optimal no-case-study scheduling algorithms. Each of the algorithms is intended for its own, rather narrow basic class of systems.

Refs: 4 titles.

УДК 681.324:681.3.001.57

Виртуальные машины и сети

Гордеев А. В. Информационно-управляющие системы, 2006. № 2. С. 21–26.

В статье кратко описываются наиболее распространенные средства моделирования персональных компьютеров и локальных сетей на их основе, известные как виртуальные машины. В результате сравнительного анализа средств моделирования излагаются основные их возможности, достоинства и недостатки. Приведены примеры использования виртуальных машин и сетей как в учебном процессе, так и в реальной практике построения вычислительных сетей.

Список лит.: 2 назв.

UDK 681.324:681.3.001.57

Virtual machines and networks

Gordeev A. V. IUS, 2006. N 2. P. 21–26.

We describe most popular tools of emulations of PCs and networks, known as virtual machines. There are very few publications on this topic in Russian. As a result of comparison of virtual machines, the author talks about their basic features, advantages and disadvantages. The article narrates about examples of using virtual PCs and virtual networks both in the educational aspect and in the case of real-life networks.

Refs: 2 titles.

УДК 621.391

Использование адресов абонентов для разрешения конфликтов в канале с шумом

Марковский С. Г., Тюрликов А. М. Информационно-управляющие системы, 2006. № 2. С. 27–37.

Рассматриваются алгоритмы случайного множественного доступа, использующие адреса абонентов для разрешения конфликтов в системе с конечным числом абонентов. Показывается, что при определенной модификации данные алгоритмы работоспособны в канале с шумом, приводящим к возникновению ложных конфликтов. Приведена методика определения скорости и средней задержки передачи сообщения в канале с ложными конфликтами.

Список лит.: 9 назв.

UDK 621.391

Using the adresses of subscribers for collision resolution in a channel with noise

Markovski S. G., Tyurlikov A. M. IUS, 2006. N 2. P. 27–37.

The paper considers random multiple access algorithms that use subscribers addresses for collisions resolution in finite system. With some modifications, these algorithms are able to work in the channel with noise, false collisions appearing. A method of algorithm rate and average delay definition in the channel with false collision is presented.

Refs: 9 titles.

УДК 519.872

Полный расчет системы обслуживания с распределениями Кокса

Рыжиков Ю. И. Информационно-управляющие системы, 2006. № 2. С. 38–46.

Разработан алгоритм полного расчета стационарных характеристик системы обслуживания вида C2/C2/n: стационарного распределения числа заявок, их распределения перед прибытием очередной заявки, моментов распределения интервалов между обслуженными заявками. Обсуждаются частные случаи и обобщения. Рассмотрены особенности программной реализации алгоритма, приведены результаты тестирования. Алгоритм предлагается использовать при расчете сетей обслуживания методом потокоэквивалентной декомпозиции.

Список лит.: 7 назв.

UDK 519.872

A complete calculation of the service system with Coxian distribution

Ryzhikov Yu. I. IUS, 2006. N 2. P. 38–46.

An algorithm is proposed to compute the characteristics of the system C2/C2/n. This algorithm computes the stationary distribution of the number of demands, its distribution before arriving, and the moments of interdeparting intervals. Particular cases and generalizations, programming realization and testing results are discussed. The algorithm is planned to be used in the flow-equivalent network analysis.

Refs: 7 titles.

УДК 007: 57 + 007:573

Информационное обеспечение исследований и коррекций в герниологии

Бегун П. И., Лазарев С. М., Лебедева Е. А. Информационно-управляющие системы, 2006. № 2. С. 47–52.

Построены расчетные схемы и компьютерные модели для вычисления напряжений и перемещений в герниосистемах. Проведены исследования зависимости напряжений и перемещений при развитии патологического образования в белой линии живота, при ущемлении грыжевых ворот и после проведения операции.

Список лит.: 4 назв.

УДК 681.513:541.13

Параметрическая идентификация электрохимического процесса на основе генетических алгоритмов

Мендельсон А. М., Бендерская Е. Н., Тенно Р. А. Информационно-управляющие системы, 2006. № 2. С. 53–56.

Рассматривается задача идентификации параметров двухсторонней модели электрохимического процесса. Использован подход, основанный на оценке параметров линеаризованной модели с последующим пересчетом линейных параметров в нелинейные. Для учета нелинейных ограничений при оценивании предлагается использовать генетические алгоритмы.

Список лит.: 6 назв.

UDK 007: 57 + 007:573

Algorithms and software for research in herniology

Begun P. I., Lazarev S. M., Lebedeva E. A. IUS, 2006. N 2. P. 47–52.

Theoretical schemes and computer-based models of calculation of the voltage and displacements in herniasystems are constructed. The dependence of voltage and displacement are studied during the development of pathological forming in the white line of a stomach and the infringement of the hernial gate.

Refs: 4 titles.

UDK 681.513:541.13

Genetic based identification of electochemical processes

Mendelsson A. M., Benderskaya E. N., Tenno R. A. IUS, 2006. N 2. P. 53–56.

The problem of identifying the parameters of an electrochemical process is considered. The solution is based on genetic optimisation of linear coefficients subject to nonlinear constraints and recalculation of linear parameters to nonlinear parameters. The simulation results show the advantages of this method.

Refs: 6 titles.