УДК 681.142.2+519.2

ОПТИМАЛЬНОЕ УПРАВЛЕНИЕ N FIFO-ОЧЕРЕДЯМИ НА БЕСКОНЕЧНОМ ВРЕМЕНИ

Е. А. Аксенова,
канд. физ.-мат. наук, научный сотрудник
А. В. Соколов,
доктор физ.-мат. наук, ведущий научный сотрудник
Институт прикладных математических исследований Карельского научного центра РАН
А. В. Драц,
студент
Петрозаводский государственный университет

Исследуются методы представления n FIFO-очередей в памяти размером m единиц. Решаются задача оптимального разбиения памяти между очередями в случае последовательного циклического представления очередей и задача анализа связанного представления очередей. В качестве математических моделей предложены случайные блуждания по целочисленной решетке в различных областях n-мерного пространства. Задачи решаются с помощью аппарата регулярных цепей Маркова.

Ключевые слова *— FIFO-очередь, связанный список, случайное блуждание, регулярные цепи Маркова.*

Введение

Во многих приложениях требуется работа с несколькими FIFO-очередями, расположенными в общем пространстве памяти. Для этого применяют различные программные или аппаратные решения [1–3]. В данной статье предлагаются математические модели для последовательного циклического и связанного способов представления очередей [1]. В обоих способах представления для каждой очереди нужны два указателя на начало и конец. В первом способе элементы равных длин располагаются циклически в последовательных адресах участка памяти, выделенного очереди. Во втором каждая очередь представлена в виде односвязанного списка элементов, и переполнение памяти наступает тогда, когда список свободных элементов пуст и требуется включить элемент в какую-либо очередь. В обоих способах операции включения и исключения выполняются за время $O(1)$. В качестве критерия оптимальности рассмотрена минимальная доля потерянных элементов при бесконечном времени работы очередей. Эту величину разумно минимизировать, когда переполнение очереди является не аварийной, а стандартной ситуацией (здесь мы подчеркиваем, что в некоторых приложениях при переполнении очереди работа программы заканчивается, и тогда в качестве критерия оптимальности надо рассматривать максимальное среднее время до переполнения памяти). То есть если очередь занимает всю предоставленную ей память, то все последующие элементы, поступающие в нее, отбрасываются до тех пор, пока не появится свободная память (т. е. до тех пор, пока не произойдет исключение элемента из очереди). Такая схема применяется, например, в работе сетевых маршрутизаторов [3] в том случае, когда по мере увеличения трафика очередь на исходящем интерфейсе маршрутизатора заполняется пакетами. Такое поведение маршрутизатора называется «сбросом хвоста». Потери пакетов приводят к нежелательному результату, поэтому число таких ситуаций необходимо свести к минимуму. Мы в этой работе строим математические модели в виде случайных блужданий по целочисленной решетке. Первоначально такие модели в виде случайного блуждания в треугольнике [4–7] были построены для решения задачи анализа процесса работы с двумя стеками, растущими навстречу друг другу [1]. В этих моделях предполагается, что на каждом шаге дискретного времени с заданными вероятностями происходят некоторые операции со стеками. Время выполнения операций — это не случайная величина, а константа, поэтому фиксированным является и шаг времени. Рассмотрены случаи последовательного представления очередей для $n = 2$ [8] и $n = 3$ [9]. В данной статье рассмотрены последовательный и связанный способы представления FIFO-очередей для случая произвольного n. Необходимо определить, как распределить память между очередями в последовательном способе организации и какой из способов организации очередей является оптимальным.

В работе будем придерживаться следующих обозначений:

m — размер памяти;

n — количество стеков и/или очередей в быстрой памяти;

p_i — вероятность включения элемента в i-ю очередь;

q_i — вероятность извлечения элемента из i-й структуры данных;

r — вероятность того, что не произойдет операции включения или извлечения;

k_i — размер памяти, выделенной для очереди с номером i при последовательном представлении;

x_i — текущая длина структуры данных с номером i;

l — отношение размера узла к размеру указателя (для связанного представления);

P^* — доля времени, которую проводит очередь в состоянии «сброса хвоста».

Одна очередь на бесконечном времени

Рассмотрим одну FIFO-очередь, расположенную в быстрой памяти размером m. В каждый момент дискретного времени может произойти одна из следующих операций: включение элемента с вероятностью p, исключение элемента с вероятностью q, очередь не изменяет своей длины с вероятностью r, где $p + q + r = 1$. Если происходит включение элемента при полностью заполненной памяти, то он считается потерянным. Требуется определить долю времени, в течение которого происходят потери элементов, при бесконечном времени работы.

Для описания процесса работы построим однородную регулярную цепь Маркова [10] с $m + 2$ состояниями, где состояния с номерами $0, \ldots, m$ соответствуют количеству элементов, находящихся в очереди. Состояние $m + 1$ соответствует «сбросу хвоста», т. е. пока процесс находится в этом состоянии, происходит потеря поступающих в очередь элементов.

Пусть процесс находится в состоянии $m + 1$, т. е. очередь заполнила всю выделенную память и произошла попытка включения еще одного элемента. Тогда с вероятностью p процесс остается на месте, так как при попытке включения элемента в переполненную очередь он будет потерян и для процесса блуждания ничего не изменится, с вероятностью q процесс перейдет в состояние $m - 1$, т. е. исключается элемент из переполненной очереди и появляется одна свободная позиция в памяти, с вероятностью r процесс перейдет в состояние m. Переход процесса из состояния x в состояние x' определяется следующими правилами:

$$x \xrightarrow{p} x' \begin{cases} x+1, & 0 \le x \le m; \\ x, & x = m+1; \end{cases} \quad x \xrightarrow{q} x' \begin{cases} x, & x = 0; \\ x-1, & 1 \le x \le m; \\ x-2, & x = m+1; \end{cases}$$

$$x \xrightarrow{r} x' \begin{cases} x, & 0 \le x \le m; \\ x-1, & x = m+1. \end{cases}$$

Построим матрицу переходных состояний **P**. Для данной цепи она имеет вид

$$\mathbf{P} = \begin{pmatrix} q+r & p & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\ q & r & p & 0 & \ldots & 0 & 0 & 0 & 0 \\ 0 & q & r & p & \ldots & 0 & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & q & r & p & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & q & r & p \\ 0 & 0 & 0 & 0 & \ldots & 0 & q & r & p \end{pmatrix}.$$

У построенной цепи Маркова существует предельный вектор $\boldsymbol{\alpha} = (\alpha_0, \alpha_1, \ldots, \alpha_m, \alpha_{m+1})$, который удовлетворяет уравнению $\boldsymbol{\alpha}\mathbf{P} = \boldsymbol{\alpha}$. По закону больших чисел, значение α_i является долей времени, которое процесс проводит в состоянии i. Тогда α_{m+1} является долей времени, которое процесс проводит в состоянии «сброса хвоста» при бесконечном времени работы.

Построим систему уравнений для определения предельных вероятностей:

$$\begin{pmatrix} 1-(q+r) & -q & 0 & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ -p & 1-r & -q & 0 & \ldots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & 1-r & -q & \ldots & 0 & 0 & 0 & 0 & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 0 & \ldots & -p & 1-r & -q & 0 & 0 \\ 0 & 0 & 0 & 0 & \ldots & 0 & -p & 1-r & -q & -q \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & -p & 1-r & -r \\ 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & -p & 1-p \end{pmatrix} \times$$

$$\times \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \ldots \\ \alpha_{m-2} \\ \alpha_{m-1} \\ \alpha_m \\ \alpha_{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \ldots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Вычеркнем первую строчку и добавим условие нормировки $\alpha_0 + \alpha_1 + ... + \alpha_m + \alpha_{m+1} = 1$:

$$\begin{pmatrix} -p & 1-r & -q & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & 1-r & -q & \dots & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & 1-r & -q & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & 1-r & -q & -q \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & 1-r & -r \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p & 1-p \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \dots \\ \alpha_{m-2} \\ \alpha_{m-1} \\ \alpha_m \\ \alpha_{m+1} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

Найдем α_{m+1} по формуле Крамера: $\alpha_{m+1} = \Delta_{m+1} / \Delta$. Разложим определитель Δ_{m+1} по последнему столбцу:

$$\Delta_{m+1} = \begin{vmatrix} -p & 1-r & -q & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & 1-r & -q & \dots & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & 1-r & -q & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & 1-r & -q & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & 1-r & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p & 0 \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} -p & 1-r & -q & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -p & 1-r & -q & \dots & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & 1-r & -q & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & 1-r & -q \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & 1-r \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p \end{vmatrix} = (-p)^{m+1}.$$

Для вычисления Δ прибавим к первой строке строки с номерами 2, ..., m, ко второй — с номерами 3, ..., m и т. д. Учитывая равенство $p + q + r = 1$, получаем

$$\Delta = \begin{vmatrix} -p & 1-r & -q & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & 1-r & -q & \dots & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & 1-r & -q & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & 1-r & -q & -q \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & 1-r & -r \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p & 1-p \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 \end{vmatrix} = \begin{vmatrix} -p & q & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & q & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & q & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & q & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & q & q \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p & 1-p \\ 1 & 1 & 1 & 1 & \dots & 1 & 1 & 1 & 1 & 1 \end{vmatrix}.$$

Рассмотрим случай $p \neq q$. Умножим первый столбец на q/p и прибавим ко второму столбцу, затем умножим второй столбец на q/p и прибавим к третьему. Продолжим эту операцию до столбцов с номерами m и $m + 1$. Потом столбец с номером m умножим на q/p и прибавим к столбцу с номером $m + 2$. Тогда

$$\Delta = \begin{vmatrix} -p & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ 0 & -p & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -p & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -p & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & -p & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & -p & 1-p \\ 1 & s_2 & s_3 & s_4 & \dots & s_{m-2} & s_{m-1} & s_m & s_{m+1} & s_{m+1} \end{vmatrix},$$

где s_i — сумма геометрической прогрессии:

$$s_i = 1 + \frac{q}{p} + \left(\frac{q}{p}\right)^2 + \ldots + \left(\frac{q}{p}\right)^{i-1} = \frac{\left(\frac{q}{p}\right)^i - 1}{\frac{q}{p} - 1}.$$

Умножим столбец с номером $m + 1$ на $(1 - p)/p$ и прибавим к столбцу с номером $m + 2$, после чего получим диагональную матрицу, поэтому окончательно определитель

$$\Delta = s_{m+1}\left(1 + \frac{1-p}{p}\right)(-p)^{m+1} =$$

$$= \frac{\left(\frac{q}{p}\right)^{m+1} - 1}{\left(\frac{q}{p} - 1\right)p}(-p)^{m+1} = \frac{\left(\frac{q}{p}\right)^{m+1} - 1}{q - p}(-p)^{m+1}.$$

Тогда

$$\alpha_{m+1} = \frac{\Delta_{m+1}}{\Delta} = \frac{q - p}{\left(\frac{q}{p}\right)^{m+1} - 1}.$$

Если $q = p$, то s_i будет суммой арифметической прогрессии и $s_i = i$. Тогда

$$\alpha_{m+1} = \frac{p}{m+1}.$$

Окончательно получаем

$$\alpha_{m+1} = \begin{cases} \dfrac{q - p}{\left(\frac{q}{p}\right)^{m+1} - 1}, & q \neq p; \\ \dfrac{p}{m+1}, & q = p. \end{cases} \qquad (*)$$

Последовательное представление очередей

Постановка задачи.

Рассмотрим n FIFO-очередей, расположенных в быстрой памяти размером m. Для последовательного представления каждой очереди необходимо выделить k_i единиц памяти, где $k_1 + \ldots + k_n = m$. Если очередь занимает всю предоставленную ей память, то все последующие элементы, поступившие в нее, отбрасываются до тех пор, пока не появится свободная память.

Рассмотрим очередь с номером i. Вероятность того, что ее длина останется прежней, будет

$$r_i = r + \sum_{j=1, j\neq i}^{n} (p_j + q_j) = 1 - p_i - q_i.$$

В этом случае долю времени p_i^*, которое процесс проводит в состоянии «сброса хвоста», для i-й очереди можно вычислить по формуле (*). Тогда общая доля времени, проведенного в состояниях «сброса хвоста»:

$$P^* = \sum_{i=1}^{n} p_i^*.$$

Задача заключается в том, чтобы минимизировать долю потерянных элементов при переполнении какой-либо из очередей. Другими словами, необходимо определить такие значения k_i, $i = 1, 2, \ldots, n$, чтобы доля времени, проведенного в состояниях «сброса хвоста», была минимальной.

Случай равных вероятностей.

Рассмотрим случай, когда $p_i = q_i$. Тогда доля времени, проведенного в состояниях «сброса хвоста», равна

$$\sum_{i=1}^{n} \frac{p_i}{k_i + 1}.$$

Таким образом, необходимо найти

$$\min_{k_1 + \ldots + k_n = m} \sum_{i=1}^{n} \frac{p_i}{k_i + 1}.$$

Рассмотрим функцию

$$F(k_1, k_2, \ldots, k_{n-1}) = \sum_{i=1}^{n-1} \frac{p_i}{k_i + 1} + \frac{p_n}{m - \sum_{i=1}^{n-1} k_i + 1}$$

и найдем ее минимум при условии $k_i > 0$, $i = 1, 2, \ldots, n-1$, $k_1 + \ldots + k_{n-1} < m$. Функция $F(k_1, \ldots, k_{n-1})$ дважды непрерывно-дифференцируема на рассматриваемом множестве:

$$F'_{k_i} = -\frac{p_i}{(k_i + 1)^2} + \frac{p_n}{\left(m - \sum_{l=1}^{n-1} k_l + 1\right)^2};$$

$$F''_{k_i k_i} = \frac{2p_i}{(k_i + 1)^3} + \frac{2p_n}{\left(m - \sum_{l=1}^{n-1} k_l + 1\right)^3};$$

$$F''_{k_i k_j} = \frac{2p_n}{\left(m - \sum_{l=1}^{n-1} k_l + 1\right)^3}, \; i \neq j.$$

Найдем точку, подозрительную на экстремум, из условия $F'_{k_i}(k_1, \ldots, k_{n-1}) = 0$:

$$\frac{p_i}{(k_i + 1)^2} = \frac{p_n}{\left(m - \sum_{l=1}^{n-1} k_l + 1\right)^2}, \; 1 \leq i \leq n-1.$$

Введем новые переменные $y_i = k_i + 1$. Для каждого i извлечем квадратный корень из обеих частей уравнения. Учитывая, что обе части неотрицательные, получаем

$$\frac{\sqrt{p_i}}{y_i} = \frac{\sqrt{p_n}}{m + n - \sum_{l=1}^{n-1} y_l}, \; 1 \leq i \leq n-1.$$

Рассмотрим уравнения при $i = 1$ и при $i = 2$:

$$\begin{cases} \dfrac{\sqrt{p_1}}{y_1} = \dfrac{\sqrt{p_n}}{m+n-\sum_{l=1}^{n-1} y_l}; \\ \dfrac{\sqrt{p_2}}{y_2} = \dfrac{\sqrt{p_n}}{m+n-\sum_{l=1}^{n-1} y_l}. \end{cases}$$

Вычтем второе уравнение из первого:

$$\frac{\sqrt{p_1}}{y_1} - \frac{\sqrt{p_2}}{y_2} = 0.$$

Получаем

$$y_2 = \frac{y_1\sqrt{p_2}}{\sqrt{p_1}}.$$

Аналогично выразим остальные переменные y_i через y_1:

$$y_i = \frac{y_1\sqrt{p_i}}{\sqrt{p_1}},\ 2 \le i \le n-1.$$

Подставим в уравнение при $i = 1$:

$$\frac{\sqrt{p_1}}{y_1} = \frac{\sqrt{p_n}}{m+n-\dfrac{y_1}{\sqrt{p_1}}\sum\limits_{l=1}^{n-1}\sqrt{p_l}}.$$

Найдем y_1:

$$y_1 = \frac{(m+n)\sqrt{p_1}}{\sum_{l=1}^{n}\sqrt{p_l}}.$$

Получаем точку, подозрительную на экстремум:

$$k_i^* = \frac{(m+n)\sqrt{p_i}}{\sum_{l=1}^{n}\sqrt{p_l}} - 1,\ 1 \le i \le n-1.$$

Покажем, что эта точка является точкой минимума. Для этого по критерию Сильвестра нужно показать, что

$$\begin{vmatrix} F''_{k_1k_1}(k_1^*,\ldots,k_{n-1}^*) & F''_{k_1k_2}(k_1^*,\ldots,k_{n-1}^*) & \ldots & F''_{k_1k_{n-1}}(k_1^*,\ldots,k_{n-1}^*) \\ F''_{k_1k_2}(k_1^*,\ldots,k_{n-1}^*) & F''_{k_2k_2}(k_1^*,\ldots,k_{n-1}^*) & \ldots & F''_{k_2k_{n-1}}(k_1^*,\ldots,k_{n-1}^*) \\ \ldots & \ldots & \ldots & \ldots \\ F''_{k_1k_{n-1}}(k_1^*,\ldots,k_{n-1}^*) & F''_{k_2k_{n-1}}(k_1^*,\ldots,k_{n-1}^*) & \ldots & F''_{k_{n-1}k_{n-1}}(k_1^*,\ldots,k_{n-1}^*) \end{vmatrix} > 0, \quad 1 \le i \le n-1.$$

Введем обозначения

$$F''_{k_ik_j} = \frac{2p_n}{\left(m-\sum_{l=1}^{n-1}k_l+1\right)^3} = A,\ i \ne j; \quad F''_{k_ik_i} = \frac{2p_i}{(k_i+1)^3} + \frac{2p_n}{\left(m-\sum_{l=1}^{n-1}k_l+1\right)^3} = A + B_i.$$

Очевидно, что $A > 0$ и $B_i > 0$. Покажем, что определитель $\Delta_i > 0$, где

$$\Delta_i = \begin{vmatrix} A+B_1 & A & \ldots & A \\ A & A+B_2 & \ldots & A \\ \ldots & \ldots & \ldots & \ldots \\ A & A & \ldots & A+B_i \end{vmatrix}, \quad i \ge 1.$$

По методу математической индукции:
1) база индукции $i = 1$: $\Delta_1 = A + B_1 > 0$;
2) пусть верно при $i = j$;
3) докажем при $i = j + 1$:

$$\Delta_{j+1} = \begin{vmatrix} A+B_1 & A & \dots & A & A \\ A & A+B_2 & \dots & A & A \\ \dots & \dots & \dots & \dots & \dots \\ A & A & \dots & A+B_j & A \\ A & A & \dots & A & A+B_{j+1} \end{vmatrix} =$$

$$= \begin{vmatrix} A+B_1 & A & \dots & A & A \\ A & A+B_2 & \dots & A & A \\ \dots & \dots & \dots & \dots & \dots \\ A & A & \dots & A+B_j & A \\ A & A & \dots & A & A \end{vmatrix} +$$

$$+ \begin{vmatrix} A+B_1 & A & \dots & A & 0 \\ A & A+B_2 & \dots & A & 0 \\ \dots & \dots & \dots & \dots & \dots \\ A & A & \dots & A+B_j & 0 \\ A & A & \dots & A & B_{j+1} \end{vmatrix} =$$

$$= \begin{vmatrix} B_1 & 0 & \dots & 0 & A \\ 0 & B_2 & \dots & 0 & A \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & B_j & A \\ 0 & 0 & \dots & 0 & A \end{vmatrix} +$$

$$+ \begin{vmatrix} A+B_1 & A & \dots & A & 0 \\ A & A+B_2 & \dots & A & 0 \\ \dots & \dots & \dots & \dots & \dots \\ A & A & \dots & A+B_j & 0 \\ A & A & \dots & A & B_{j+1} \end{vmatrix} =$$

$$= AB_1 \dots B_j + B_{j+1}\Delta_j > 0.$$

Значит, точка $(k_1^*, \dots, k_{n-1}^*)$ является единственной точкой минимума функции $F(k_1, \dots, k_{n-1})$. Следовательно, в этой точке функция достигает своего наименьшего значения. Поскольку k_i должны быть целыми, то значения $k_1^*, \dots, k_{n-1}^*$ необходимо округлить до ближайшего целого и перебрать значения, удовлетворяющие условию $k_1 + \dots + k_n = m$:

$$k_n^* = m - \sum_{l=1}^{n-1} k_l^* = m + n - 1 -$$

$$- \frac{(m+n)\sum_{l=1}^{n-1}\sqrt{p_l}}{\sum_{l=1}^{n}\sqrt{p_l}} = \frac{(m+n)\sqrt{p_n}}{\sum_{l=1}^{n}\sqrt{p_l}} - 1.$$

Наименьшая доля времени, проведенного в состояниях «сброса хвоста»:

$$P^* = \sum_{i=1}^{n} \frac{p_i}{k_i + 1} = \sum_{i=1}^{n} \left(\frac{1}{m+n} \sqrt{p_i} \sum_{l=1}^{n} \sqrt{p_l} \right) =$$

$$= \frac{\left(\sum_{i=1}^{n} \sqrt{p_i}\right)^2}{m+n}.$$

Общий случай.

Введем обозначение

$$Z_m(k_1, \dots, k_n) = \min_{k_1 + \dots + k_n = m} \left(\sum_{i=1}^{n} p_i^*(k_i) \right),$$

где Z_m — доля времени, которое процесс проводит в состоянии «сброса хвоста», при оптимальном разбиении памяти размером m между очередями; $p_i^*(k_i)$ — доля времени, в течение которого происходит потеря элементов только для i-й очереди при размере памяти k_i.

Рассмотрим рекуррентную формулу

$$Z_m(k_1, \dots, k_n) = Z_{m-1}(k_1, \dots, k_n) -$$

$$- \max_{1 \le i \le n} \left(p_i^*(k_i) - p_i^*(k_i + 1) \right).$$

Начальное значение Z_0 равно [по формуле (*)]

$$Z_0(0, \dots, 0) = \sum_{i=1}^{n} \frac{q_i - p_i}{\left(\frac{q_i}{p_i}\right)^{0+1} - 1} = \sum_{i=1}^{n} p_i,$$

т. е. при отсутствии памяти любая попытка включения элемента в одну из очередей будет приводить к его потере.

Был реализован эффективный алгоритм решения задачи на основе предложенной модели динамического программирования, который за время $O(mn)$ вычисляет оптимальное разбиение памяти и долю времени, проведенного в состояниях «сброса хвоста». Также была построена математическая модель этого процесса в виде случайного блуждания по целочисленному n-мерному параллелепипеду с вершиной в начале координат, ребрами, параллельными осям координат, и длинами ребер $k_1 + 1, \dots, k_n + 1$. Гиперплоскости соответствуют состояниям «сброса хвоста». Была предложена нумерация состояний и на ее основе разработан алгоритм генерации соответствующей цепи Маркова и решения задачи с использованием результатов теории регулярных цепей Маркова. Этот метод решения задачи будет подробно изложен далее на примере анализа связанного представления очередей. Метод динамического программирования в данном случае приводит к более эффективному алгоритму.

Связанное представление очередей

При связанном представлении каждая очередь хранится в виде связанного списка, в кото-

ром $1/l$-я часть памяти тратится на хранение указателей. Пусть $M = m(1 - 1/l)$. В качестве математической модели рассмотрим блуждание по целочисленной n-мерной пирамиде с ребрами $0 \le x_1 \le M$, $0 \le x_2 \le M$, ..., $0 \le x_n \le M$ и основанием $x_1 + x_2 + ... + x_n = M$. Для каждого состояния $(x_1, x_2, ..., x_n)$ на плоскости $x_1 + x_2 + ... + x_n = M$, т. е. когда вся память уже занята, введем состояние $(\overline{x_1}, \overline{x_2}, ..., \overline{x_n})$, соответствующее «сбросу хвоста». В это состояние можно попасть в случае попытки включить элемент в любую из очередей, когда вся память занята. Переход процесса из состояния $(x_1, x_2, ..., x_n)$ определяется по следующим правилам:

$$(..., x_i, ..., x_j, ...) \xrightarrow{p_i}$$

$$\xrightarrow{p_i} \begin{cases} (..., x_i + 1, ..., x_j, ...), & 0 \le x_1 + ... + x_n < M; \\ (..., \overline{x_i}, ..., \overline{x_j}, ...), & x_1 + ... + x_n = M; \end{cases}$$

$$(..., x_i, ..., x_j, ...) \xrightarrow{q_i}$$

$$\xrightarrow{q_i} \begin{cases} (..., x_i - 1, ..., x_j, ...), & x_i < 0; \\ (..., x_i, ..., x_j, ...), & x_i = 0; \end{cases}$$

$$(..., x_i, ..., x_j, ...) \xrightarrow{r} (..., x_i, ..., x_j, ...);$$

$$(..., \overline{x_i}, ..., \overline{x_j}, ...) \xrightarrow{p_i} (..., \overline{x_i}, ..., \overline{x_j}, ...);$$

$$(..., \overline{x_i}, ..., \overline{x_j}, ...) \xrightarrow{q_i}$$

$$\xrightarrow{q_i} \begin{cases} (..., x_i - 1, ..., x_j, ...), & x_i > 0; \\ (..., x_i, ..., x_j, ...), & x_i = 0; \end{cases}$$

$$(..., \overline{x_i}, ..., \overline{x_j}, ...) \xrightarrow{r} (..., x_i, ..., x_j, ...).$$

На плоскости $x_1 + x_2 + ... + x_n = M$ количество состояний «сброса хвоста»

$$C_{n+M-1}^{M} = \frac{(M+n-1)!}{(n-1)!M!}.$$

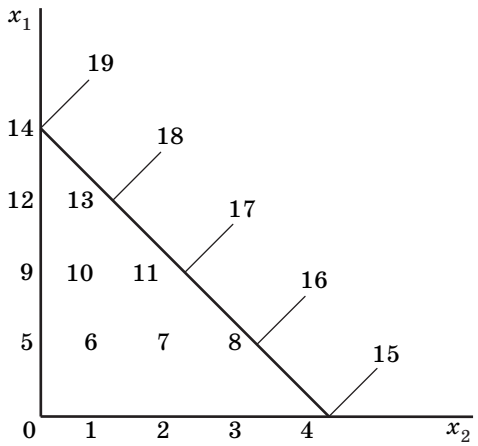
Перечислим все состояния области блуждания:
(0, 0, 0, ..., 0, 0), (1, 0, 0, ..., 0, 0), ..., (M – 1, 0, 0, ..., 0, 0), (M, 0, 0, ..., 0, 0);
(0, 1, 0, ..., 0, 0), (1, 1, 0, ..., 0, 0), ..., (M – 2, 1, 0, ..., 0, 0), (M – 1, 1, 0, ..., 0, 0);
...
(0, M – 1, 0, ..., 0, 0), (1, M – 1, 0, ..., 0, 0);
(0, M, 0, ..., 0, 0);
(0, 0, 1, ..., 0, 0), (1, 0, 1, ..., 0, 0), ..., (M – 2, 0, 1, ..., 0, 0), (M – 1, 0, 1, ..., 0, 0);
(0, 1, 1, ..., 0, 0), (1, 1, 1, ..., 0, 0), ..., (M – 3, 1, 1, ..., 0, 0), (M – 1, 1, 1, ..., 0, 0);
...
(0, 0, M – 1, ..., 0, 0), (0, 0, M, ..., 0, 0);
...
(0, 0, 0, ..., 0, 1), (1, 0, 0, ..., 0, 1), ..., (M – 2, 0, 0, ..., 0, 1), (M – 1, 0, 0, ..., 0, 1);
(0, 1, 0, ..., 0, 1), (1, 1, 0, ..., 0, 1), ..., (M – 3, 1, 0, ..., 0, 1), (M – 2, 1, 0, ..., 0, 1);
...
(0, M – 2, 0, ..., 0, 1), (1, M – 2, 0, ..., 0, 1);
(0, M – 1, 0, ..., 0, 1);
...
(0, 0, 0, ..., 0, M).

Введем нумерацию этих состояний, начиная с 0. Для того чтобы построить матрицу переходных вероятностей, построим функцию $F(X) = I$, $X = (x_1, x_2, ..., x_n)$, где $x_1, ..., x_n$ — текущие длины очередей; I — номер состояния. Будем искать ее в виде

$$F(x_1, ..., x_n) = F(0, ..., 0, x_n) + (F(0, ..., x_{n-1}, x_n) - F(0, ..., 0, x_n)) + ... + (F(0, x_2, x_3, ..., x_{n-1}, x_n) - F(0, 0, x_3, ..., x_{n-1}, x_n)) + (F(x_1, x_2, x_3, ..., x_{n-1}, x_n) - F(0, x_2, x_3, ..., x_{n-1}, x_n)),$$

т. е. будем увеличивать значения аргументов, начиная с последнего, и вычислять, на сколько увеличится значение функции. Увеличение значения разности функций $F(0, 0, ..., 0, x_i, x_{i+1}, ..., x_n) - F(0, 0, ..., 0, 0, x_{i+1}, ..., x_n)$ зависит от номера i, от значения x_i и от суммы $x_{i+1} + ... + x_n$, т. е. от количества уже занятых ячеек памяти. Теперь пронумеруем состояния «сброса хвоста» таким образом, чтобы состоянию на плоскости $x_1 + x_2 + ... + x_n = M$ с меньшим номером соответствовало состояние «сброса хвоста» с меньшим номером.

На рисунке показан пример нумерации состояний при $n = 2$, $M = 4$. Для нахождения доли времени, проведенного в состояниях «сброса хвоста», необходимо найти предельный вектор $\boldsymbol{\alpha}$ и просуммировать его компоненты с номерами от $\frac{(M+n)!}{n!M!}$ до $\frac{(M+n-1)!}{n!M!} + \frac{(M+n-1)!}{(n-1)!M!}$. В приведенном примере необходимо найти сумму $\alpha_{15} + ... + \alpha_{19}$.



■ *Нумерация состояний при n = 2, M = 4*

■ ***Таблица 1.*** *$M = 16$, $n = 5$*

r	p_1	p_2	p_3	p_4	p_5	q_1	q_2	q_3	q_4	q_5	P^*	l_1	l_2	l_4	l_8
0	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.15	0.2273	0.1923	0.1786	0.1667
0	0.25	0.15	0.05	0.03	0.02	0.25	0.15	0.05	0.03	0.02	0.1208	0.2273	0.1923	0.1786	0.1667
0	0.3	0.03	0.15	0.1	0.05	0.05	0.03	0.05	0.04	0.03	0.61	0.6989	0.6923	0.6902	0.6886
0	0.05	0.2	0.05	0.04	0.03	0.3	0.2	0.15	0.1	0.05	0.01	0.012	0.0045	0.0027	0.0017
0	0.05	0.05	0.05	0.05	0.05	0.15	0.15	0.15	0.15	0.15	0.005413	0.0002506	0.000559	0.000113	0.000021
0	0.07	0.07	0.07	0.07	0.07	0.13	0.13	0.13	0.13	0.13	0.0249	0.0322	0.0168	0.0083	0.00395

■ ***Таблица 2.*** *$M = 16$, $n = 4$*

r	p_1	p_2	p_3	p_4	q_1	q_2	q_3	q_4	P_1^*	P_2^*	l_1	l_2	l_4	l_8
0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.08	0.08	0.1333	0.1143	0.1	0.089
0	0.44	0.02	0.02	0.02	0.44	0.02	0.02	0.02	0.1	0.06	0.1667	0.1429	0.125	0.1111
0	0.5	0.07	0.07	0.07	0.08	0.07	0.07	0.07	0.462	0.459	0.5964	0.5964	0.5964	0.5964
0	0.08	0.05	0.05	0.05	0.5	0.07	0.07	0.07	0.042	0.039	0.08	0.0677	0.0589	0.0517
0	0.15	0.08	0.05	0.02	0.35	0.2	0.1	0.05	0.0049	0.0061	0.0067	0.0022	0.00068	0.0000198

Численные результаты

Был разработан комплекс программ для ЭВМ, который реализует вышеописанные алгоритмы нахождения доли времени, проведенного в состояниях «сброса хвоста». Приведем некоторые численные результаты.

Сравним последовательное и связанное представление очередей с точки зрения доли времени, в течение которого происходят потери пакетов (табл. 1). В строке P^* указана минимальная доля времени, проведенного в состояниях «сброса хвоста», для последовательного представления. В строке l_1 указана доля времени, проведенного в состояниях «сброса хвоста», для связанного представления, когда на связи тратится 1/2 часть памяти (размер информационной части равен размеру указателя), в строке l_2 — когда на связи тратится 1/3 часть памяти (размер указателя равен 1/2 размера информационной части), в строке l_4 — когда на связи тратится 1/5 часть памяти (размер указателя равен 1/4 информационной части), в строке l_8 — когда на связи тратится 1/9 часть памяти (размер указателя равен 1/8 информационной части).

На практике вероятности включения и исключения, которые считались известными, не всегда могут быть вычислены. В этом случае будет логичным разделить память поровну между всеми структурами данных в случае последовательного представления. В табл. 2 сравнивается связанное и последовательное представление в том случае, если память разделена поровну между очередями. В строке P_1^* указана доля времени, проведенного в состояниях «сброса хвоста», когда память разделена поровну между очередями, в строке P_2^* указана минимальная доля времени, проведенного в состояниях «сброса хвоста», когда память разделена оптимально. Строки l_1, l_2, l_4, l_8 такие же, как в табл. 1.

Заключение

Из приведенных таблиц видно, что связанное представление предпочтительнее использовать, если вероятности включения элемента в очереди меньше, чем вероятности исключения, и на связи тратится 1/3 часть памяти или меньше. В остальных случаях лучше использовать последовательное представление, даже если вероятностные характеристики очередей заранее неизвестны и разбиение памяти может быть неоптимальным.

Работа выполнена при финансовой поддержке РФФИ, грант № 09-01-00330-а.

Литература

1. **Кнут Д.** Искусство программирования для ЭВМ. Т. 1. — М.: Вильямс, 2001. — 736 с.
2. **Седжвик Р.** Фундаментальные алгоритмы на C++. — Киев: Диасофт, 2001. — 688 с.
3. **Боллапрагада В., Мэрфи К., Расс У.** Структура операционной системы Cisco IOS. — М.: Вильямс, 2002. — 208 с.

4. **Соколов А. В.** О распределении памяти для двух стеков // Автоматизация эксперимента и обработки данных: Сб. ст. / Карельский филиал АН СССР. Петрозаводск, 1980. С. 65–71.
5. **Flajolet P.** The evolution of two stacks in bounded space and random walks in a triangle // Lecture Notes in Computer Science. 1986. Vol. 223. P. 325–340.
6. **Louchard G., Schott R., Tolley M., Zimmermann P.** Random walks, heat equation and distributed algorithms // Journal of Computational and Applied Mathematics. 1994. N 53. P. 243–274.
7. **Аксенова Е. А., Соколов А. В.** Оптимальное управление двумя параллельными стеками // Дискретная математика. 2007. № 1. С. 67–75.
8. **Соколов А. В., Тарасюк А. В.** Об оптимальном управлении циклическими **FIFO-очередями // Си**стемы управления и информационные технологии. 2005. № 3 (20). С. 29–33.
9. **Аксенова Е. А.** Оптимальное управление FIFO-очередями на бесконечном времени // Стохастическая оптимизация в информатике: Межвуз. сб. СПб.: СПбГУ, 2006. Вып. 2. С. 71–76.

УВАЖАЕМЫЕ АВТОРЫ!

С сентября 2009 г. основные элементы статей, размещенные на платформе РУНЭБ, индексируются в крупнейшей поисковой системе Интернета Google.

На сайте РУНЭБ (http://www.elibrary.ru) доступна новая услуга — «обсуждение статьи». Авторы и читатели теперь могут вступить в диалог и ответить на вопросы и комментарии друг друга.