ИМИТАЦИОННЫЕ МОДЕЛИ В НЕКОТОРЫХ ЗАДАЧАХ ОПТИМАЛЬНОГО УПРАВЛЕНИЯ ДИНАМИЧЕСКИМИ СТРУКТУРАМИ ДАННЫХ*

Е. А. Аксенова, Т. В. Афанасьева, А. В. Драц, А. В. Соколов, А. В. Рюгина, А. В. Тарасюк (Петрозаводск)

В докладе рассмотрены имитационные модели в некоторых задачах оптимального управления такими структурами данных, как LIFO-стеки, FIFO-очереди и приоритетные очереди. Ранее [1–5] были предложены математические модели в виде случайных блужданий по целочисленной решетке в различных областях и алгоритмы оптимального управления этими структурами данных. В настоящем докладе предложены имитационные модели, с помощью которых сравниваются результаты решения поставленных оптимизационных задач методами имитационного моделирования и с помощью теории цепей Маркова. Для построения имитационных моделей реализованы новые датчики случайных чисел на языках JAVA и C++, которые работали лучше, чем встроенные датчики.

Первая задача – это задача оптимального управления двумя параллельными стеками в двухуровневой памяти. Пусть в памяти объема m расположены два стека, растущие навстречу друг другу. В этом случае место их встречи можно рассматривать как случайную величину. Пусть известно, что на каждом шаге с вероятностью p может произойти исключение элемента из одного из стеков и с вероятностью $1\text{-}p$ – включение информации в один из стеков. Обозначим через $M(m,p)$ математическое ожидание случайной величины $\max(k1,k2)$, где $k1$ и $k2$ – длины стеков при встрече. Д. Кнут [6,2.2.2,упр. 13] поставил задачу разработать математическую модель этого процесса и установить вид функции $M(m,p)$.

В работах [7–11] построена математическая модель процесса в виде двумерного случайного блуждания в треугольной области с двумя отражающими экранами и одним поглощающим. В [3] рассмотрена задача оптимального управления двумя параллельными стеками в двухуровневой памяти, обобщающая задачу Д. Кнута. Предполагается, что вершины двух стеков растут навстречу друг другу в быстрой памяти, к которой разрешен доступ нескольких параллельных процессоров. После переполнения или опустошения вершин стеков в быстрой памяти нужно привести обе вершины стека в оптимальное состояние. Будем считать, что при перераспределении мы всегда приходим к некоторому заданному состоянию памяти, после чего начинается следующий этап работы. Ставится задача: в какое состояние, в зависимости от заданных вероятностей включения элементов в стеки или их исключения, следует переходить после обращения к памяти второго уровня, чтобы среднее время работы до следующего перераспределения памяти было максимально, т. е. чтобы минимизировать среднее число обменов между уровнями.

Обозначим через x_1 текущую длину первого стека, а через x_2 – второго. Тогда в качестве математической модели процесса получаем случайное блуждание по целочисленной решетке в области $x_1 \ge 0, x_2 \ge 0,$ где $x_1 + x_2 \le m$ (не превосходит объема выделенной памяти). Параллельное выполнение операций подразумевает возможность одновременной работы с двумя стеками. В каждый момент времени могут произойти следующие операции над стеками: включение элемента в первый стек с вероятностью p_1, исключение элемента из первого стека с вероятностью q_1, включение элемента во второй стек с вероятностью p_2, исключение элемента из второго стека с вероятностью q_2,

* Работа выполнена при финансовой поддержке РФФИ (проект № 06-01-00303).

включение элементов в два стека одновременно с вероятностью p_{12}, исключение элементов из двух стеков одновременно с вероятностью q_{12}, включение в первый и одновременно исключение из второго с вероятностью pq_{12}, исключение из первого и одновременно включение во второй с вероятностью qp_{12}, с вероятностью r стеки не изменяют своей длины (только чтение или отсутствие операции), где $p_1 + q_1 + p_2 + q_2 + p_{12} + q_{12} + pq_{12} + qp_{12} = 1$.

Рис. 1. Область блуждания

Задача состоит в том, чтобы определить такое начальное состояние блуждания внутри треугольной области, при выходе из которого среднее время блуждания до поглощения на прямых $x_1 = -1, x_2 = -1$, и ломаной, которая задается соотношениями $x_1 + x_2 = m+1$ и $x_1 + x_2 = m+2$, было максимальным. Если пронумеровать сначала поглощающие состояния, начиная с точки *(m,0)* (против часовой стрелки), а затем невозвратные (снизу вверх, справа налево), то получим конечную однородную поглощающую цепь Маркова [10, 12]. Геометрически блуждание по решетке можно представить так: включение элемента в первый стек – сдвиг вправо по оси x_1, исключение элемента из первого стека – сдвиг влево по оси x_1, включение элемента во второй стек – сдвиг вверх по оси x_2, исключение элемента из второго стека – сдвиг вниз по оси x_2, включение элементов в два стека одновременно – сдвиг вправо вверх по диагонали, исключение элементов из двух стеков одновременно – сдвиг влево вниз по диагонали, включение в первый и исключение из второго одновременно – сдвиг вправо вниз по диагонали, исключение из первого и включение во второй одновременно – сдвиг влево вверх по диагонали. В [3] был предложен алгоритм решения данной задачи и приведены результаты численных экспериментов.

Эта же задача решалась при помощи имитационного моделирования. Сначала на языке Java была реализована имитационная модель поведения двух стеков в быстрой памяти. Ее можно использовать для визуализации случайного блуждания внутри треугольника и для решения оптимизационной задачи. Для нахождения оптимальной точки проводился полный перебор всех возможных точек: для каждой из них определенное количество раз запускалась имитация блуждания внутри треугольника до достижения переполнения или опустошения одного из стеков. Вычислялось среднее время блуждания для каждой точки и из всех времен выбиралось наибольшее. Результаты, полученные с помощью программы на Java, довольно сильно отличались от полученных с помощью марковской модели, потому что для каждой точки проводилось 10 000 имитаций блуждания. Для увеличения точности вычислений был также разработан вариант программы, использующий многопроцессорный суперкомпьютер IBM pSeries 690(Regatta), установленный на ВМК МГУ. Доступ к ней осуществлялся с помощью удаленного терминала по протоколу ssh. С использованием языка C++ и техно-

логии параллельного программирования OpenMP был разработан параллельный вариант имитационной модели. Поскольку Regatta – значительно более мощная ЭВМ, то для каждой точки использовался 1 000 000 повторений, вследствие чего результаты стали гораздо ближе к результатам, полученным с помощью марковской модели.

Вторая задача заключается в сравнении эффективности разных методов представления в памяти одного уровня двух FIFO-очередей. В [2] были построены математическая модель последовательного циклического метода представления каждой очереди в виде случайного блуждания по целочисленной решетке в прямоугольной области и модель метода представления двух очередей в виде двух связных списков в виде случайного блуждания по целочисленной решетке в треугольной области. В качестве критерия оптимальности рассмотрено среднее время до переполнения памяти. В [2] решалась задача оптимального представления двух последовательных FIFO-очередей, когда в качестве критерия оптимальности рассмотрена доля потерянных при переполнении элементов очередей на бесконечном промежутке времени. Такой критерий уместен, например, в сетевых маршрутизаторах, когда пакеты, которые помещаются в переполненную очередь, просто теряются («сброс хвоста») [13]. В [1] был предложен новый способ работы с несколькими последовательными очередями. В этом методе память заранее не делится между очередями, а очереди двигаются друг за другом по кругу, начиная с некоторой заданной точки.

Авторами предлагается имитационная модель, разработанная для оценки эффективности описанных методов представления двух FIFO-очередей с точки зрения рассмотренных критериев оптимальности. Модель реализована на языке Java с использованием своего датчика случайных чисел. Также разработана имитационная модель для оптимального управления двухприоритетной очередью [4]. Эта модель реализована на языке C++ с использованием своего датчика случайных чисел, основанного на линейном конгруэнтном методе [14]. Сравнение результатов численных экспериментов на математической и имитационной моделях для описанных задач показало, что они близки. Построенные имитационные модели могут быть применены и для решения более сложных задач оптимального управления структурами данных, когда математические модели построить невозможно. Кроме того, эксперименты показали, что вычисления с помощью имитационного моделирования достаточно легко можно распараллелить.

Литература

1. **Соколов А. В.** Математические модели и алгоритмы оптимального управления динамическими структурами данных. Петрозаводск: ПГУ, 2002.
2. **Соколов А. В., Тарасюк А. В.** Об оптимальном управлении циклическими FIFO-очередями//Системы управления и информационные технологии. 2005. N 3(20). С. 29–33.
3. **Аксенова Е. А., Соколов А. В., Тарасюк А. В.** Об оптимальном управлении двумя FIFO-очередями в конечной области памяти//Системы управления и информационные технологии. 2006. N 3. С. 62–68.
4. **Аксенова Е. А., Соколов А. В.** Оптимальное управление двумя параллельными стеками в двухуровневой памяти//Дискретная математика. 2007. Т. 19. Вып. 1. С. 67–75.
5. **Афанасьева Т. В., Соколов А. В.** Оптимальное управление параллельной двухприоритетной очередью//Материалы IX Международного семинара «Дискретная математика и ее приложения», посвященного 75-летию со дня рождения академика О. Б. Лупанова. М.: Изд-во механико-математического факультета МГУ, 2007. С. 203–206.
6. **Кнут Д. Э.** Искусство программирования для ЭВМ. М.: Мир, 1976. Т. 1.

7. **Соколов А. В.** О распределении памяти для двух стеков. Автоматизация эксперимента и обработки данных. Петрозаводск, 1980. С. 65–71.
8. **Yao A. C.** An analysis of a memory allocation scheme for implementating stacks. SIAM J. Computing. 1981. P. 398–403.
9. **Flajolet P.** The evolution of two stacks in bounded space and random walks in a triangle//Lec. Notes Computer Sci. 1986. Vol. 23. P. 325–340.
10. **Louchard G**. Random walks, heat equation and distributed algorithms//G. Louchard, R. Schott, M. Tolley, P. Zimmermann . J. Comput. Appl. Math. 1994. P. 243–274.
11. **Maier R. S.** Colliding Stacks: A Large Deviations Analysis . Random Structures and Algorithms. 1991. P. 379–421.
12. **Кемени Дж., Снелл Дж.** Конечные цепи Маркова. М.: Наука, 1970.
13. **Боллапрагада В., Мэрфи К., Уайт Р.** Структура операционной системы Cisco IOS//М.: издательский дом «Вильямс». 2002.
14. **Кнут Д. Э.** Искусство программирования. Получисленные алгоритмы М.: издательский дом «Вильямс», 2005. Т. 2.