n-Ядро и распределение памяти компьютера*

В. В. Мазалов[1], Л. И. Трухина[2]
Институт прикладных математических исследований КарНЦ РАН, Петрозаводск, Россия
[2] *Читинский институт Байкальского государственного университета экономики и права, Россия*
e-mail: `vmazalov@krc.karelia.ru`, `lit-79@mail.ru`

Предложен теоретико-игровой подход к определению оптимального способа распределения памяти компьютера, разбитой на n блоков. Информация поступает в один из блоков случайным образом. Критерием эффективности является минимизация среднего числа распределений памяти, которые необходимо производить при полном заполнении одного из участков.

Ключевые слова: стек, распределение памяти, кооперативные игры, n-ядро.

Введение

Наиболее часто используемыми в вычислительной технике и программировании базовыми структурами данных являются стеки и очереди. Для работы с ними применяют различные программные или аппаратные решения. В ряде работ были предложены математические модели и алгоритмы оптимального управления этими структурами данных. Так, в [1] данные моделируются пуассоновскими процессами, в [2] в качестве математических моделей предложены случайные блуждания по целочисленным решеткам в различных областях. В качестве критерия эффективности в этих моделях рассматривается среднее время до переполнения памяти.

В настоящей работе предложен теоретико-игровой подход к определению оптимального способа распределения памяти компьютера, состоящей из последовательно расположенных стеков. Критерием эффективности здесь является минимизация среднего числа распределений памяти, которые необходимо производить при переполнении одного из участков.

1. Оптимальная схема распределения памяти

Память размера M разбита на n частей. Длины разбиений обозначим m_1, m_2, ..., m_n, $m_1+m_2+\dots+m_n = M$ — целые числа. Информация в каждый блок поступает случайным образом с вероятностями $p_1, p_2, \dots, p_n$ (вероятностные характеристики известны). Предполагается, что информация поступает в виде порций единичной длины и заполнение каждого участка осуществляется с одного конца. Работа начинается с пустой структуры данных. Текущие длины занятых участков обозначим k_1, k_2, ..., k_n. Тогда

*Работа выполнена при финансовой поддержке РФФИ (грант № 10-01-00089-а) и Отделения математических наук РАН.

$d_i = m_i - k_i$ — свободная память i-го участка. При переполнении одного из участков необходимо производить распределение свободной памяти. Каким образом это сделать, чтобы число распределений было минимальным?

Пусть свободная память представлена вектором $(d_1, ..., d_n)$, где $d_i \geq 0$, $i = \overline{1, n}$. Обозначим через $T(d_1, ..., d_n)$ среднее время до первого переполнения. Данный критерий удовлетворяет уравнению

$$T(d_1, ..., d_n) = 1 + \sum_{i=1}^{n} p_i T(d_1, ..., d_i - 1, ..., d_n) \tag{1}$$

с граничным условием $T(d_1, ..., d_n) = 0$, если хотя бы для одного i $d_i = 0$ [3]. Тогда оптимальный алгоритм распределения может быть описан следующим образом. Предположим, что функция $T(d_1, ..., d_n)$ вычислена для всех наборов $(d_1, ..., d_n)$. В этом случае при заполнении одного из участков, например $d_1 = 0$, разделяем оставшееся свободное пространство $D = \sum_{i=1}^{n} d_i$ на новые сегменты $(d_1^{'}, ..., d_n^{'})$ так, чтобы значение функции $T(d_1^{'}, ..., d_n^{'})$ было максимальным. При такой схеме распределения среднее время до переполнения памяти каждый раз будет максимальным, а среднее число распределений минимальным.

Это можно осуществить для небольших размеров памяти, однако при больших n и D сложность перебора становится экспоненциальной. Функция T вычисляется по формуле (1) рекуррентным образом через предыдущие значения. Таким образом, на практике данный алгоритм использовать невозможно.

Возникает проблема эффективного распределения компьютерной памяти. Ниже в качестве такой процедуры предлагается использовать n-ядро кооперативной теории игр.

2. n-Ядро в распределении памяти

2.1. Алгоритм n-ядра

В 1985 г. Ауманн совместно с Машлером рассмотрели задачу о банкротстве на основе одного из текстов, представленных в Талмуде. В своей работе они предложили так называемую *коалиционную процедуру*, которая приводит к *устойчивому решению* данной задачи, а устойчивое решение в свою очередь является n-ядром соответствующей игры [4, 5].

Сформируем кооперативную игру, связанную с задачей распределения памяти. По аналогии с задачей о банкротстве объем занятой памяти будем считать требованиями, а суммарную свободную память — "состоянием":

$k_1, k_2, \ldots, k_n$ — размеры заполненных участков — требования каждого игрока;

$k_1 + k_2 + ... + k_n = K$ — суммарные требования;

$d_1 + d_2 + ... + d_n = D$ — свободная память — состояние.

Тогда характеристическую функцию для каждой коалиции определим так же, как в задаче о банкротстве:

$$\upsilon(S) = \left(D - \sum_{i \in N \setminus S} k_i \right)^{+},$$

где $a^+ = \max(a, 0)$. Тогда $v(N) = D$ — объем свободной памяти, который нужно перераспределить между n участками.

Дележ, образующий n-ядро, определяется при помощи коалиционной процедуры [4]. Упорядочим участки памяти по возрастанию занятого объема. На первом шаге игроки объединяются в две коалиции: $\{1\}$ — первый участок памяти и $\{2, \ldots, n\}$ — остальные участки. Если $n\frac{k_1}{2} \le D \le K - n\frac{k_1}{2}$, то D делится между $\{1\}$ и $\{2, \ldots, n\}$ по принципу "*contested garment*" ("оспариваемый предмет одежды"), т. е. спорное количество памяти делится поровну и каждый игрок дополнительно получает ее количество, уступленное другим. Тогда получим следующий дележ:

$$x_1 = \frac{D - (D - k_1)^+ - \left(D - \sum\limits_{i=2}^{n} k_i\right)^+}{2} + \left(D - \sum_{i=2}^{n} k_i\right)^+ \text{ —}$$

объем памяти, выделенный первому участку, и

$$x_2 = \frac{D - (D - k_1)^+ - \left(D - \sum\limits_{i=2}^{n} k_i\right)^+}{2} + (D - k_1)^+ \text{ —}$$

объем памяти, выделенный остальным участкам. Если $D \le n\frac{k_1}{2}$, то свободная память делится поровну, если $D \ge K - n\frac{k_1}{2}$, то назначаются равные потери

$$x_1 = k_1 - \frac{K - D}{2}, \quad x_2 = \sum_{i=2}^{n} k_i - \frac{K - D}{2}.$$

На втором шаге коалиция $\{2, \ldots, n\}$ разбивается на две: $\{2\}$ и $\{3, \ldots, n\}$, и повторяется вышеописанная процедура для $(n - 1)$ игрока, и т. д.

Понятно, что для большого числа участков применить данный алгоритм и получить дележ достаточно сложно. Поэтому модифицируем его.

2.2. Модифицированный алгоритм

Продолжим рассмотрение свободной памяти. Участки упорядочим по убыванию свободной памяти. Если $d_1 + d_2 + \ldots + d_n = D \ge \frac{d_1}{2}n$, то каждому участку i $(i = \overline{1, n})$ выделяется половина его свободной памяти, а вторая половина делится поровну между участками с номерами от $i + 1$ до n. Таким образом, имеем

$$x_1 = \frac{d_1}{2},$$

$$x_2 = \frac{d_1}{2(n-1)} + \frac{d_2}{2},$$

$$x_3 = \frac{d_1}{2(n-1)} + \frac{d_2}{2(n-2)} + \frac{d_3}{2},$$

$$\ldots$$

$$x_{n-1} = \frac{d_1}{2(n-1)} + \frac{d_2}{2(n-2)} + ... + \frac{d_{n-2}}{4} + \frac{d_{n-1}}{2},$$

$$x_n = \frac{d_1}{2(n-1)} + \frac{d_2}{2(n-2)} + ... + \frac{d_{n-2}}{4} + \frac{d_{n-1}}{2} + \frac{d_n}{2}.$$

Так как $d_n = 0$, то $x_{n-1} = x_n = \frac{1}{2}\sum\limits_{i=1}^{n-1} \frac{d_i}{n-i}$. Если вышеуказанное условие не выполняется, то применяется алгоритм равного деления.

Преимущество модифицированного алгоритма в том, что он позволяет получить дележ в явном виде для любого числа участков.

2.3. Алгоритм равного деления

Суммарная свободная память делится на n равных частей. Тогда каждому участку выделяется память размера

$$x_i = \frac{\sum\limits_{i=1}^{n} d_i}{n} = \frac{D}{n}, \quad i = \overline{1, n}.$$

3. Численные эксперименты

Для сравнения представленных алгоритмов проведем два вычислительных эксперимента. В первом из них будем менять размер памяти при постоянном числе участков, во втором — число участков при постоянном размере памяти. В каждом из экспериментов используем три набора вероятностных распределений:

$$p_i = 1/n,$$

$$p_i = 1/2^i, \quad i = \overline{1, n-1}, \quad p_n = 1 - \sum_{i=1}^{n-1} p_i,$$

$$p_i = 1/3^i, \quad i = \overline{1, n-1}, \quad p_n = 1 - \sum_{i=1}^{n-1} p_i.$$

На рисунке представлены результаты экспериментов, в которых размер памяти M менялся от 100 до 200 при постоянном числе участков $n = 4$. Естественно, для всех наборов вероятностей минимальное число распределений показывает оптимальный алгоритм. Для случая равных вероятностей остальные алгоритмы дают примерно одинаковое число распределений. Для вероятностей вида $p_i = 1/2^i$ преимущество показывает алгоритм n-ядра. Для вероятностей вида $p_i = 1/3^i$ результаты модифицированного алгоритма и алгоритма равного деления примерно одинаковы, а алгоритм n-ядра дает худшие результаты. Вместе с тем с ростом объема памяти разница между числом распределений уменьшается.

В таблице приведены результаты эксперимента, в котором менялось число участков при постоянном размере памяти $M = 10\,000$. В этом случае сравнивались только два алгоритма — равного деления и модифицированный, так как выше отмечалось, что при большом числе участков ($n > 4$) оптимальный алгоритм и алгоритм n-ядра применить сложно. Из данных таблицы видно, что для двух наборов вероятностей среднее число

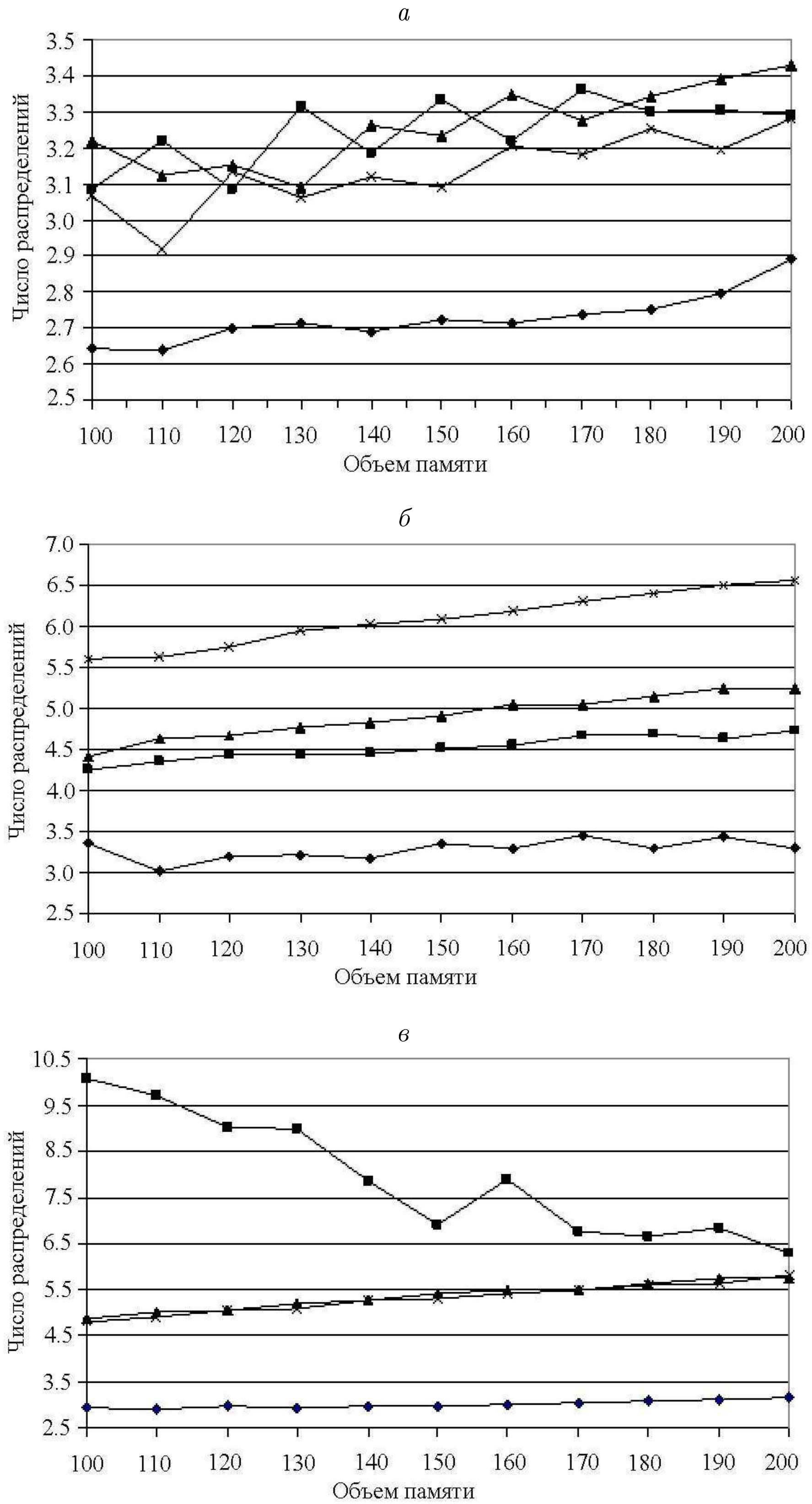

Рис. 1. Среднее число распределений для набора вероятностей вида $p_i = 1/n$ (*а*), $p_i = 1/2^i$ (*б*), $p_i = 1/3^i$ (*в*); ♦ — оптимальный алгоритм, ■ — алгоритм n-ядра, ▲ — модифицированный алгоритм, × — алгоритм равного деления

Среднее число распределений памяти для $M = 10\,000$

Число участков	Вероятность	Среднее число распределений	
		Алгоритм равного деления	Модифицированный алгоритм
10	$p_i = 1/n$	10.9247	11.4310
	$p_i = 1/2^i$	37.5626	28.4988
	$p_i = 1/3^i$	29.8461	25.9096
15	$p_i = 1/n$	17.2411	17.5756
	$p_i = 1/2^i$	56.6942	42.4482
	$p_i = 1/3^i$	42.7845	36.3179
20	$p_i = 1/n$	21.6191	21.9891
	$p_i = 1/2^i$	74.8886	56.0917
	$p_i = 1/3^i$	54.5431	46.1052

распределений при использовании модифицированного алгоритма значительно меньше, чем в случае алгоритма равного деления, причем с ростом числа участков разность между числом распределений увеличивается. Для равных вероятностей преимущество показывает алгоритм равного деления, однако отличия от модифицированного алгоритма несущественные.

Заключение

В работе предложены алгоритмы распределения памяти компьютера, состоящей из последовательно расположенных стеков. Результаты вычислительного эксперимента показали, что для ряда наборов вероятностных распределений поступления информации алгоритм n-ядра и модифицированный алгоритм дают меньшее число распределений памяти по сравнению с алгоритмом равного деления.

Список литературы

[1] Мазалов В.В., Тамаки М., Винниченко С.В. Оптимальное распределение памяти компьютера для пуассоновских потоков // Автоматика и телемеханика. 2008. № 9. С. 69–75.

[2] Аксенова Е.А., Соколов А.В. Оптимальное управление двумя параллельными стеками // Дискретная математика. 2007. Т. 19, вып. 1. С. 67–75.

[3] Мазалов В.В. Об одном методе динамического распределения нестраничной памяти // Программирование. 1995. № 4. С. 183–185.

[4] Aumann R., Maschler M. Game-teoretic analysis of a bankruptcy problem from the Tulmud // J. of Economic Theory. 1985. No. 36. P. 195–213.

[5] Мазалов В.В. Математическая теория игр и приложения: Уч. пособие. 1-е изд. СПб.: Лань, 2010.

Поступила в редакцию 11 февраля 2011 г.