ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ
1973 год Выпуск 57

УДК 681.326

ОБ ИНФОРМАЦИОННО- И ПРОСТРАНСТВЕННО-ВРЕМЕННЫХ ПРЕОБРАЗОВАНИЯХ АЛГОРИТМОВ

Ю.Г. Косарев

Рассматриваются два вида преобразований алгоритмов, направленные на оптимальное использование:

- объемов запоминающих устройств (информационно-временные преобразования, или ИВ-преобразования);
- процессоров (пространственно-временные преобразования, или ПВ-преобразования).

I. Информационно-временные преобразования.

I.I.I. Поставим в соответствие алгоритму A решения данной задачи объем вычислений N и объем информации $\mathcal{I}$.

$$N = \sum_i r_i g_i , \qquad (1)$$

где r_i - среднее число операций типа i, выполняемое данным алгоритмом при решении данной задачи; g_i - вес операции типа i, пропорциональный среднему времени ее выполнения на заданном вычислительном устройстве;

$$\mathcal{I} = \max_t \mathcal{I}_t , \qquad (2)$$

где $\mathcal{I}_t$ - объем информации, подлежащей хранению при выполнении шага t алгоритма A; $\mathcal{I}_t$ учитывает также затраты на хранение самого алгоритма (точнее, его машинной реализации - программы).

I.I.2. Пусть имеется задача $\mathcal{U}$, то есть задано множество выходных данных и сформулированы условия, которым должны удовлетворять выходные данные как по составу, так и по точности.

Пусть C – множество заданных эквивалентных алгоритмов решения задачи $\mathcal{U}$. Выделенную из множества C последовательность алгоритмов B, в которой алгоритмы упорядочены по убыванию N; любые алгоритмы $A_i(N_i, T_i) \in B$ и $A_j(N_j, T_j) \in B$ несравнимы, то есть

$$N_i > N_j \quad \text{и} \quad T_i < T_j \quad \text{при} \quad i < j, \quad N_i < N_j \quad \text{и} \quad T_i > T_j \quad \text{при} \quad i > j \qquad (3)$$

и ограничена степень близости (ε_N и ε_T) алгоритмов по значениям N и T

$$\left| \frac{N_i - N_{i-1}}{N_i} \right| \geqslant \varepsilon_N, \qquad \left| \frac{T_i - T_{i-1}}{T_i} \right| \geqslant \varepsilon_T, \qquad (4)$$

назовем главной последовательностью эквивалентных алгоритмов (ГП).

I.I.3. Поставим в соответствие ГП информационно-временную функцию (ИВФ)

$$N = f_u(T), \qquad (5)$$

определенную в точках, соответствующих алгоритмам, входящим в ГП.

I.I.4. Из определения ГП и ИВФ непосредственно вытекают следующие их свойства:

– ГП упорядочена как по убыванию N, так и по возрастанию T;

– любой участок ГП и ИВФ $[A_i(N_i, T_i), A_j(N_j, T_j)]$ содержит конечное число k_{ij} алгоритмов:

$$k_{ij} \leqslant k_{ij}^{(max)} \leqslant min\left(entier\left(\frac{\log \frac{N_i}{N_j}}{\log(1-\varepsilon_N)} \right), entier\left(\frac{\log \frac{T_j}{T_i}}{\log(1+\varepsilon_T)} \right) \right), \qquad (6)$$

где $k_{ij}^{(max)}$ – максимальное число точек, которое может быть размещено на данном участке ГП или ИВФ без нарушения условий (4);

– ИВФ – монотонно-убывающая функция, определенная на отрезке $[T_0, T_m]$, границы которого соответствуют алгоритмам A_0 с минимальным объемом информации и A_m с минимальным объемом вычислений;

– при включении в ГП нового алгоритма A_j происходит: либо оптимизация, когда из ГП удаляются алгоритмы A_i, для которых

$$N_j < N_i \quad \text{при} \quad T_j \leq T_i \quad \text{или} \quad N_j \leq N_i \quad \text{при} \quad T_j < T_i; \qquad (7)$$

либо пополнение ГП и ИВФ, когда удовлетворяются условия (3) и (4);

– по мере пополнения множества с ИВФ перемещается в сторону начала координат и стремится к оптимальной, когда ГП и ИВФ состоят из алгоритмов, которые одновременно минимальны по объему операций при фиксированном $\mathcal{Y}$ и минимальны по объему информации при фиксированном N;

– по мере пополнения множества C каждый участок $[\mathcal{A}_k, \mathcal{A}_\ell]$ ГП и ИВФ приближается к равномерно-представительному (полному), когда для двух любых соседних по номеру алгоритмов $\mathcal{A}_i$ и $\mathcal{A}_{i-1}$

$$(N_i - N_{i-1})/N_i = \varepsilon_N \qquad \text{или} \qquad (\mathcal{Y}_i - \mathcal{Y}_{i-1})/\mathcal{Y}_i = \varepsilon_{\mathcal{Y}}. \qquad (8)$$

I.I.5. Рассмотрим пути нахождения новых эквивалентных алгоритмов, основывающиеся на преобразовании с помощью эффективных процедур*) известных, уже вошедших в ГП алгоритмов.

Задача пополнения ГП заключается в следующем. Имеется алгоритм $\mathcal{A}_i(N_i, \mathcal{Y}_i) \in$ ГП. Требуется эффективно преобразовать его в алгоритм $\mathcal{A}_j(N_j, \mathcal{Y}_j)$, эквивалентный $\mathcal{A}_i$ и удовлетворяющий условиям (4). Если при этом удовлетворяется условие (3), то происходит пополнение ГП, если удовлетворяется (7) – оптимизация.

Условимся писать $\mathcal{A}_j = P(\mathcal{A}_i)$, если алгоритм $\mathcal{A}_j$, эквивалентный $\mathcal{A}_i$, получен из $\mathcal{A}_i$ с помощью эффективной процедуры P.

Множество C_A алгоритмов, эквивалентных по результатам некоторому алгоритму $\mathcal{A}$, каждый из которых может быть получен с помощью эффективной процедуры либо из алгоритма $\mathcal{A}$, либо из любого другого алгоритма из множества C_A назовем с е м е й с т в о м алгоритмов. Участки ИВФ и ГП, состоящие из алгоритмов, принадлежащих к одному семейству, назовем о д н о с е м е й н ы м и.

I.2. Не претендуя на полноту и строгость, попытаемся выявить эффективные процедуры преобразования алгоритмов (далее ИВ-процедуры). При этом будем как правило, основываться на примерах ИВ-преобразований алгоритмов, встретившихся на практике.

---

*) Функции, операторы, процедуры, преобразования, алгоритмы и т.п. будем называть э ф ф е к т и в н ы м и, если объем вычислений при их реализации не превышает заранее заданной величины.

I.2.I. Вынесение операторов из цикла. В цикле могут присутствовать операторы, которые при каждом повторении цикла вычисляют одни и те же значения величин и размещают их в те же участки памяти. Вынесение таких тождественно повторяющихся операторов из цикла сокращает объем вычислений. Если при этом объем информации остается тем же или возрастает менее чем в $(I - \varepsilon_y)$ раз, то происходит оптимизация ГП. Если объем информации растет и удовлетворяются условия (3) и (4), то происходит пополнение ГП.

ПРИМЕР I. Расчет на ЭВМ "Минск-22" атомного реактора на компанию методом прогонки [I]. Внутренние циклы в этом случае можно представить в виде

$$\underbrace{Q_1^m \; \underbrace{Q_2^{mi}}\; P(m)}\; P(i), \tag{9}$$

где Q_1^m вычисляет значения прогоночных коэффициентов для каждого узла сетки m; Q_2^{mi} определяет новые значения функции в узле m на итерации i. Результаты вычислений оператора Q_1^m не зависят от итерации i. Это позволяет вынести его из цикла i:

$$\underbrace{Q_1^m}\; P(m) \quad \underbrace{Q_2^{mi}}\; P(m)\; P(i). \tag{I0}$$

На выполнение оператора Q_1^m в алгоритме (9) затрачивается около 2/3 общего времени счета, поэтому алгоритм (I0) выполняется примерно в 3 раза быстрее, но его осуществление стало возможным при удвоении объема оперативной памяти (ОП).

I.2.2. Сокращение числа тождественных повторений цикла.

ПРИМЕР 2. Задача перекрещивания массивов информации [2]. Соответствующий алгоритм имеет вид:

$$\Lambda_2 \; \underbrace{\Lambda_1 \; \mathcal{A} \; \underbrace{P(i)}_{i \le c_1}}\; \underbrace{P(j)}_{j \le c_2}. \tag{II}$$

Оператор Λ_2 вводит в ОП часть массива M_2, мимо которой прогоняется с помощью оператора Λ_1 часть массива M_1. Оператор $\mathcal{A}$ выполняет некоторое действие над кодами массивов M_1 и M_2, одновременно находящимися в ОП.

Здесь внутренний цикл (по i) тождественно повторяется c_2 раз (для каждого j):

$$c_2 = m_2 / \ell_2 , \qquad (12)$$

где m_2 – размеры массива M_2 ; ℓ_2 – размеры ОП, отведенные для размещения массива M_2 .

Время на перекрещивание массивов (без затрат на оператор A) обратно пропорционально объему оперативной памяти, отводимому под массив M_2 . При $\ell_2 = m_2$ повторные вводы массива M_1 прекращаются.

I.2.3. Применение таблиц [3,4]. Во многих практических задачах встречаются многократные вычисления функции от небольшого числа переменных при сравнительно невысоких требованиях к точности. Такие функции могут быть затабулированы и счет заменен выборкой из таблицы (с последующей интерполяцией, когда это требуется).

В общем виде данный процесс преобразования алгоритма сводится к замене каждого вхождения соответствующего оператора Q_f, вычисляющего функцию f , оператором выборки Q_B и к вынесению перед алгоритмом блока вычисления таблицы

$$\{Q_f\} \Rightarrow \underbrace{Q_T \quad P(\omega)}_{t\ \omega \le W} \{Q_B\} . \qquad (13)$$

Оператор Q_T вычисления табличного значения f обычно проще Q_f благодаря закономерному ходу вычисления табличных значений.

Очевидное условие целесообразности применения таблиц:

$$N(Q_T)\frac{W}{M} + N(Q_B) < N(Q_f) , \qquad (14)$$

где $N(Q)$ – объем вычислений оператора Q ; W – размер таблицы; M – общее число обращений к таблице при всех реализациях алгоритма (и всех других использующих ее алгоритмов).

ПРИМЕР 3. Моделирование взаимодействия электронов с веществом методом Монте-Карло [5,6]. Успех в решении этой сложной задачи был достигнут главным образом на основе применения таблиц. На каждом варианте решения надо промоделировать 10000 траекторий электронов со средним числом шагов порядка 10^2 – 10^3. На каждом шаге акты взаимодействия электронов с веществом разыгрывались по сложным формулам, определяющим длину пробега ΔS и угол рассеяния $\mathcal{P}$. Каждое вычисление ΔS и $\mathcal{P}$ занимало около 1000 операций,

кроме того, примерно по 100 операций приходилось на вычисление $\cos x$ и $\sqrt{1-x^2}$ с помощью стандартных программ.

Для всех четырех функций $\Delta S(S,\gamma)$, $P(S,\gamma)$, $\cos x$ и $\sqrt{1-x^2}$ были составлены таблицы. Первые две, зависящие от длины пути S и случайного числа γ, имели большой объем по 128000 чисел в каждой. Они хранились на магнитной ленте и вызывались в ОП частями. Благодаря монотонному изменению одного из аргументов (S) с помощью данной части таблиц можно было одновременно обрабатывать группу траекторий и тем самым свести удельное время на ввод таблиц к небольшой величине. Для этих таблиц $N(Q_B)\approx 5$; $N(Q_T)\approx 500$.

Таблицы для $\cos x$ и $\sqrt{1-x^2}$ с равномерным шагом объемом по 1024 числа каждая полностью разместились в ОП. Для них $N(Q_B)=2$. Эти таблицы насчитывались один раз для всех вариантов, поэтому удельными затратами на их счет можно пренебречь.

1.2.4. Расписывание цикла. Рассмотрим какой-либо цикл. В операторной форме записи он выглядит как

$$\underbrace{A^i \quad P(i)}_{i\le n}. \tag{15}$$

Расписать цикл означает записать этот же алгоритм без оператора цикла в виде последовательности $A_1, A_2, \dots, A_n$. В этом случае увеличивается объем записи алгоритма, но зато не нужно выполнять оператор $P(i)$. Естественно, что это имеет смысл делать, когда $N(P)$ сравнимо с $N(A)$. Обычно полное расписывание цикла излишне. Достаточно уменьшить повторяемость оператора P, применяя частичное расписывание цикла, при котором число повторений цикла n уменьшается до

$$z = \text{entier}\left(\frac{n}{k}\right), \qquad k = 4, 8, 16, \dots \tag{16}$$

Тогда цикл выглядит так

$$\underbrace{P(k-\Delta)\, A_1 \dots A_k\, P(i)}_{i<z}. \tag{17}$$

Здесь оператор $P(k-\Delta)$ учитывает некратность n и k; Δ – остаток от деления n на k.

Применение частичного расписывания цикла для счета парных произведений в задачах линейной алгебры дало выигрыш во времени

примерно на 15-20% [7]. Несколько большее сокращение N (до $\approx 40\%$) достигается в циклах из одной операции, например, при пересылке с помощью цикла группы кодов, очистке памяти и т.п.

1.2.5. Перераспределение объемов памяти между блоками. Пусть алгоритм $A(N, \mathcal{J})$ состоит из двух блоков $A_1(N_1, \mathcal{J}_1)$ и $A_2(N_2, \mathcal{J}_2)$, для каждого из которых имеются свои $\Gamma\Pi_1$ и $\Gamma\Pi_2$. В общем случае, когда A_1 и A_2 зависимые (то есть имеют общие подблоки или массивы) получение новых точек ГП алгоритма A может быть сведено к перебору (декартову произведению $\Gamma\Pi_1$ и $\Gamma\Pi_2$) и последующей проверке совместимости и удовлетворению условиям (4), (3) или (7). В некоторых случаях точки ГП можно находить аналитически, как уже рассматривалось в примере 2.

ПРИМЕР 4. Оптимальное распределение памяти в задаче перекрещивания массивов. Пусть, как и в примере 2, массив M_1 объемом m_1 кодов вводится во внутреннем цикле, а массив M_2, объемом m_2 кодов – во внешнем. Обозначим через b_1 и b_2 число кодов, отведенных в ОП соотвественно для частей массивов M_1 и M_2 ($b_1 + b_2 = \mathcal{J}$). Обозначим

$$\frac{b_2}{b_1} = \gamma. \qquad (18)$$

Общий объем вычислений

$$N = m_2 t_2 + \frac{m_1 m_2}{b_2} t_1 + \frac{m_2}{b_2} \tau_2 + \frac{m_1 m_2}{b_1 b_2} \tau_1, \qquad (19)$$

где t_1 и t_2 – затраты на ввод одного кода массивов M_1 и M_2 (учитывается возможность размещения массивов на разных типах вспомогательных памятей), τ_1 и τ_2 – единоразовые затраты при вводе каждой части кодов масеивов M_1 и M_2. Заменяя b_1 и b_2 через γ и $\mathcal{J}$, имеем

$$N = m_2 t_2 + \frac{m_1 m_2}{\mathcal{J}} t_1 \cdot \frac{\gamma + 1}{\gamma} + \frac{m_2}{\mathcal{J}} \tau_2 \cdot \frac{\gamma + 1}{\gamma} + \frac{m_1 m_2}{\mathcal{J}^2} \tau_1 \cdot \frac{(\gamma + 1)^2}{\gamma}. \qquad (20)$$

Можно видеть, что $N = N_{min}$ при

$$\gamma = \gamma_{opt} = \sqrt{\mathcal{J}\left(\frac{t_1}{\tau_1} + \frac{\tau_2}{\tau_1} \cdot \frac{1}{m_1}\right)} + 1, \qquad (21)$$

$$N_{min} = m_2 t_2 + \frac{m_1 m_2}{\mathcal{J}^2} \tau_1 \left(1 + \gamma_{opt}\right)^2. \qquad (22)$$

Эти выражения позволяют получить полную ГП аналитическими средствами.

I.2.6. Преобразование массивов информации к виду, облегчающему последующую обработку.

ПРИМЕР 5. Расчет подвесок автомобилей [8]. В алгоритме решения этой задачи на ЭВМ "Минск-22" основное время счета занимало обращение к таблицам функций, зависящим от микропрофиля дороги. Эти таблицы имеют неравномерный шаг и без интерполяции не дают необходимой точности. При удвоении объема памяти удалось ввести равномерный и достаточно мелкий шаг. Это позволило исключить интерполяцию и свести обращение к таблице всего к 2 операциям вместо 44.

I.2.7. Сжатие массивов. Существует довольно много способов сжатия информации. Укажем некоторые из них:

- пропуск нулевых элементов. В этом случае вместо позиционной записи вводится непозиционная с указанием адресов нулевых элементов, которые указываются либо непосредственно вместе с элементами, либо отдельно в виде шкал;

- вынесение общих заголовков. Одинаковые части рядом находящихся элементов выносятся из этих элементов в виде отдельного заголовка, который снабжается также указанием, на сколько элементов он распространяется [2];

- перекодировка. Элементам или их частям придаются новые коды. Например, буквенные названия заменяются более компактными числовыми кодами. При этом нередко работа с кодами оказывается более удобной, то есть такая замена может привести к оптимизации.

I.2.8. Преобразование формул. Преобразование формул, вычисление по которым составляет задачу данного алгоритма, представляет собой, по-видимому, наиболее сложный вид преобразования алгоритмов. Здесь приходится иметь дело не только с формализацией аналитических преобразований, но и учитывать их машинную реализацию. Так, например, тождественное алгебраическое преобразование $\left(\frac{x_1}{z} - \frac{x_2}{z}\right) = \frac{x_1 - x_2}{z}$ может оказаться недопустимым из-за потери точности при счете на машине.

Этот обширный вид преобразований требует отдельного рассмотрения вне рамок данной работы.

2. Пространственно-временные преобразования. Эти преобразования алгоритмов подробно описывались нами ранее [9,10], и им посвящена обзорная работа Н.Н. Миренкова [11]. Поэтому ограничимся перечислением основных результатов.

2.1. Каждый параллельный алгоритм (р-алгоритм) можно характеризовать числом ветвей L и объемом вычислений N в самой большой по числу операций ветви.

Аналогично главной последовательности информационно-временной функции для ИВ-преобразований можно построить главную последовательность параллельных алгоритмов

$$A_1(N_1, L_1),\ A_2(N_2, L_2),\ \ldots,\ A_m(N_m, L_m)$$

и пространственно-временную функцию ПВФ $N = f_n(L)$, которая совпадает с характеристической функцией р-алгоритмов [10,12]. (Точнее, ГП (N, L) и ПВФ соответствуют точкам характеристической функции, в которых р-алгоритм одновременно минимален по числу ветвей при заданном N и минимален по объему вычислений при заданном L). ГП (N, L) и ПВФ обладают всеми свойствами, указанными для ГП (N, L) и ПВФ.

Более того, доказано существование преобразования, с помощью которого из р-алгоритма $A_\ell(N_\ell, L_\ell)$ может быть получен участок ГП (N, L) и ПВФ $[A_\ell(N_\ell, L_\ell), A_o(N_o, 1)]$, где A_o – последовательный алгоритм, эквивалентный алгоритму A_ℓ. Важное свойство этого преобразования – сохранение (или даже увеличение) эффективности (малости доли затрат на взаимодействие между параллельными ветвями).

Особенно просто это преобразование выполняется при наличии независимого цикла (или совокупности независимых циклов), охватывающего весь или почти весь алгоритм. В этом случае построение р-алгоритма $A_i(N_i, L_i)$ из алгоритма $A_\ell(N_\ell, L_\ell)$ $(N_i > N_\ell, L_i < L_\ell)$ сводится к перераспределению числа повторений этого цикла и соответствующих частей массива между L_i ветвями.

2.2. Проблема преобразования последовательного алгоритма в эквивалентный ему параллельный также решается эффективно, когда имеются (и явно указаны) циклы с необходимыми свойствами (независимость повторений, охват всех основных операторов, достаточно

большое число повторений). Предпринятый нами анализ большого числа разнообразных классов задач с большим объемом вычислений показал, что для всех них существуют алгоритмы с циклами, обладающими указанными выше свойствами. Более сложной в общем случае оказалась проблема формализации обнаружения таких циклов непосредственно по алгоритму или программе.

3. Как можно видеть из рассмотренных выше примеров, ИВ - и ПВ-преобразования алгоритмов тесно связаны со свойствами циклов. Формализация выявления свойств циклов, необходимых для того или иного преобразования, представляет собой весьма сложную проблему. Для ее успешного решения необходимо прежде всего ввести форму представления алгоритмов, которая облегчала бы обнаружение свойств циклов и позволила бы создать соответствующий формальный аппарат для преобразований алгоритмов.

Существующие алгоритмические языки, как правило, имеют специальные средства для записи циклических процессов. Однако эти средства направлены именно на запись повторяющегося процесса, а не на его преобразования. К тому же обычно не запрещается организовывать цикл другими средствами, например, с помощью условных переходов, переключателей и т.п., а также с помощью рекурсивных конструкций, что нередко позволяет лаконично записать алгоритм, но может в сильной мере усложнить его анализ.

3.1. Циклические алгоритмы (*C* -алгоритмы). Выскажем соображения о форме представления алгоритмов, которая может оказаться удобной для целей анализа и преобразования алгоритмов.

Представляет интерес попытаться построить алгоритм в виде совокупности циклов, каждый из которых рассматривается как самостоятельный блок. Цикл имеет имя, описание массивов и переменных. При обращении к циклу задаются значения фактических параметров, по которым специальный оператор настройки (Н) настраивает цикл на реализацию повторяющегося процесса.

Основную часть цикла составляет его изменяющаяся часть - тело (Т), которое состоит из ядра (Я) и организатора цикла (О). Ядро может быть оператором, циклом или последовательностью циклов. Организатор цикла определяет функционирование тела цикла: при каждом повторении изменяет адреса переменных, используемых в теле цикла, вносит какие-либо изменения в тело цикла, а также проверяет условия выхода из цикла.

С каждым циклом связана также функция перехода (f), которой передает управление организатор при выполнении условия выхода из цикла. Функция определяет, какой цикл должен быть выполнен вслед за данным. Вообще говоря, функция f зависит от результатов, полученных телом цикла, и может изменяться организатором цикла.

Кроме полных циклов, содержащих все указанные компоненты, могут быть неполные циклы, состоящие из оператора и функции перехода. Оператор предполагается далее неразложимым на циклы.

3.2. Построение ИВ и ПВ-функций преобразования алгоритмов тесно связано с созданием универсальных алгоритмов и программ, настраивающихся на заданные параметры технических средств. Построение универсальных программ, настраивающихся на число параллельных ветвей, уже нашло применение на практике [9,10]. Эту задачу можно считать в основном решенной.

Задачу построения универсальных программ, настраивающихся на заданные объемы памяти, еще находится в стадии осмысления. Одна из попыток такого рода представлена в данной работе.

Следующим этапом представляется задача построения универсальных программ, настраивающихся одновременно и на объем памяти и на число процессов. В этом случае алгоритм одновременно характеризуется тремя величинами $A(N, T, L)$, связанными между собой двумерной функцией преобразования $N = f_{ИП}(T, L)$.

При определении объема информации в ИВ-преобразованиях учитываются все виды информации, хранящиеся в оперативной памяти, а не только программа как в большинстве преобразований [13].

В данной работе преследовалась цель обратить внимание на преобразования алгоритмов решения фиксированной задачи, которые ориентированы на наилучшее использование возможностей, предоставляемых техническими средствами. В отличие от других известных преобразований алгоритмов (например, обсужденных в [13]), которые ориентированы, как правило, на оптимизацию качества алгоритма по какому-либо параметру (объему вычислений, длине программы, числу используемых рабочих ячеек и т.п.), предлагаемые ИВ- и ПВ-преобразования выполняются над частично-упорядоченным подмножеством оптимальных алгоритмов.

ИВ- и ПВ-преобразования предполагают возможность измерения как объема вычислений, так и объема информации. В связи с этим счи-

таются фиксированными как сама задача, так и типовые наборы данных. В этих случаях указанные параметры могут быть получены либо непосредственно по алгоритму и исходным данным, либо определены с помощью специальных программ [I4].

Литература

I. КОСАРЕВ Ю.Г., ВЕЛЕСЬКО А.А., ЮРЕВИЧ Р.В. Многогрупповой расчет двумерного реактора в диффузионном приближении на системе "Минск-222".-"Вычислительные системы", Новосибирск, "Наука" Сиб. отд., I968, вып. 30, с. I5-22.

2. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г., УСТИНОВ В.А. Применение ЭВМ в исследовании письменности древних Майя. Т.4. Новосибирск, "Наука", I969.

3. КОСАРЕВ Ю.Г. Примеры использования таблиц для сокращения времени счета. -"Вычислительные системы", Новосибирск, "Наука" Сиб.отд., I968, вып. 30, с. 46-54.

4. ГОЛОВЯШКИНА Л.В., КОСАРЕВ Ю.Г. О построении таблиц с быстрой выборкой. Настоящий сборник, с. I25-I37.

5. КОСАРЕВ Ю.Г. Моделирование неупругого рассеяния электронов методом Монте-Карло. -"Вычислительные системы", Новосибирск, "Наука" Сиб.отд., I968, вып. 30, с. 34-4I.

6. ГОЛОВЯШКИНА Л.В. О методике прямого моделирования случайных процессов. Настоящий сборник, с. I38-I48.

7. КОСАРЕВ Ю.Г., ГОЛОВЯШКИНА Л.В. Решение системы линейных уравнений на "Минске-222". -"Вычислительные системы",Новосибирск, "Наука" Сиб.отд., I967, вып. 24, с. 55-75.

8. КОСАРЕВ Ю.Г., ПЕТРОВИЧ А.И. Исследование колебательных процессов автопоездов на системе "Минск-222". -"Вычислительные системы" Новосибирск, "Наука" Сиб.отд., I968, вып.30, с. I2-I5.

9. КОСАРЕВ Ю.Г. Распараллеливание по циклам. -"Вычислительные системы", Новосибирск, "Наука" Сиб.отд., I967, вып.24,с.3-20.

I0. ЕВРЕИНОВ Э.В., КОСАРЕВ Ю.Г. Однородные универсальные вычислительные системы высокой производительности.Новосибирск,"Наука", I966.

II. МИРЕНКОВ Н.Н. Параллельные алгоритмы для решения задач на однородных вычислительных системах. Настоящий сборник,с.3-32.

I2. КОСАРЕВ Ю.Г. О методике решения задач на универсальных вычислительных системах.-"Вычислительные системы", Новосибирск, I965, вып.I7, с. 6I-99.

I3. Системное и теоретическое программирование.Сборник трудов ВЦ СО АН СССР,Новосибирск , I972.

I4. КОЛОСОВА Ю.И. Комплекс средств производства параллельных программ. Настоящий сборник, с.98-II4.

Поступила в ред.-изд.отд.
7 августа I973 года