EESTI NSV TEADUSTE AKADEEMIA TOIMETISED. 24. KÖIDE
FÜÜSIKA * MATEMAATIKA. 1975, NR. 2

ИЗВЕСТИЯ АКАДЕМИИ НАУК ЭСТОНСКОЙ ССР. ТОМ 24
ФИЗИКА * МАТЕМАТИКА. 1975, № 2

https://doi.org/10.3176/phys.math.1975.2.05

УДК 519.95

В. АЛАДЬЕВ, О. ОСИПОВ

О СЛОЖНОСТИ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ, ОПРЕДЕЛЯЕМЫХ ОДНОРОДНЫМИ СТРУКТУРАМИ

В последние годы важную теоретическую и практическую значимость, в частности для определения сложности программ для ЭВМ [1], приобрела проблема оценки сложности алгоритмов. В связи с этим внимание исследователей направлено на изучение однородных структур (ОС), так как они нашли широкое применение в вычислительной технике [2], особенно в свете новой технологической базы [3], и наилучшим образом отвечают новому подходу к организации вычислительного процесса. В исследованиях по созданию вычислительной техники на базе ОС наметились два направления: решение вопросов технического конструирования логических и вычислительных устройств и создание теории параллельных алгоритмов. Впервые на реальную возможность распараллеливания вычислительного процесса посредством определения параллельных алгоритмов было указано в [2]. Настоящее сообщение посвящено оценке сложности параллельных алгоритмов, определяемых одномерными ОС.

В теории алгоритмов понятие «алгоритм» обычно уточняется посредством описания математической модели вычислительного устройства. Здесь возможны два подхода: оценка сложности или самого алгоритма (устройства), или вычислительного процесса, протекающего в соответствии с данным алгоритмом.

В качестве меры сложности алгоритмов рассмотрим функционал, относящий каждому алгоритму α некоторое число $M(\alpha)$, характеризующее в некотором смысле его громоздкость. Здесь под сложностью будем понимать сложность в смысле Нагорного [4], а временную оценку сложности проведем относительно машин Тьюринга (МТ). Н. М. Нагорный показал, что любая словарная частично рекурсивная функция, заданная в алфавите A, вычислима с помощью нормального алгоритма уже в алфавите $\{a_0, A\}$, отличающемся от A лишь одной дополнительной буквой a_0 $(a_0 \notin A)$. Зададимся целью сопоставить по сложности параллельные алгоритмы, определяемые одномерными ОС, с нормальными алгоритмами Маркова, как внешне наиболее подобными. Для цельности изложения введем на содержательном уровне необходимый минимум понятий.

Одномерная ОС есть упорядоченная четверка $\langle A_a, Z, G, L^G \rangle$, где $A_a = \{0, 1, \ldots, a-1\}$ — множество состояний единичного автомата. Множество A_a содержит состояние покоя, которое обозначается через 0. Z есть множество целых чисел числовой оси, называемое пространством. G есть упорядоченное множество $G = \{g_1, g_2, \ldots, g_p\}$ точек из Z, называемое индексом соседства.

Он определяет позиции относительно автомата i из Z тех автоматов, которые непосредственно связаны с автоматом i. Шаблон соседства автомата i есть множество автоматов, непосредственно связанных с автоматом i. Если G содержит точку 0, то каждый автомат содержится в своем собственном шаблоне соседства. В дальнейшем это условие будем считать выполненным. Первые три компоненты ОС образуют статическую часть, описывающую ее физическую структуру. Конфигурацией (КФ) пространства Z (словом) будем называть любое отображение $c_f : Z \to A_a$. КФ будем считать конечной, если она содержит только конечное число элементов из множества $A_a \backslash \{0\}$. Множество таких КФ (слов) обозначим через $\bar{C}_a$. Функционирование ОС будет определяться тогда локальной функцией перехода L^G, которая задает состояние единичного автомата i в момент T в зависимости от состояний автоматов его шаблона соседства в момент времени $T-1$. L^G есть любое отображение $L^G : A_a^p \to A_a$. Если L^G — локальная функция перехода, то состояние автомата i в момент T определяется применением функции L^G к множеству состояний автоматов из шаблона соседства i в момент $T-1$, т. е. $S(i,T) = L^G(x_1, \ldots, x_p, T-1)$ или $x_1 \ldots x_p \to S(i,T)$, где $S(i,T)$ есть состояние автомата i в момент T и $x_k \in A_a$ $(k = \overline{1,p})$. Будем рассматривать ОС, удовлетворяющие условию конечной скорости передачи информации между автоматами среды: $L^G(0, \ldots, 0, T) = 0$. Одновременное применение функции L^G к шаблонам соседства всех автоматов в ОС определяет глобальную функцию перехода $\tau : \bar{C}_a \to \bar{C}_a$. Функция τ, примененная к любой КФ из $\bar{C}_a$, снова порождает КФ из множества $\bar{C}_a$, т. е. $\bar{C}_a$ замкнуто относительно операции τ. Если $c_0 \in \bar{C}_a$ есть начальная КФ Z в момент $T = 0$, то КФ Z в момент $T = n$ есть $c_0\tau^n \in \bar{C}_a$. Будем говорить, что ОС генерирует из $c_0 \in \bar{C}_a$ последовательность КФ (слов) $W_{c_0} = \{c_0, c_1, c_2, \ldots, c_m, \ldots\}$, если для каждого $j \geqslant 0$ $c_j\tau = c_{j+1}$. Таким образом, глобальная функция τ в алфавите A_a определяет параллельный алгоритм переработки слов, определяемый одномерной ОС. Причем, под концом переработки любого слова c_0 таким алгоритмом естественно понимать слово $c_m = c^*$, для которого имеет место соотношение $c^*\tau = c^*$. Нетрудно заметить, что параллельный алгоритм на самом общем уровне описывает функционирование одномерных ОС из идентичных конечных алгоритмов. Универсальной КФ (УКФ) будем называть такую КФ $c_0 \in \bar{C}_a$, для которой имеет место соотношение $W_{c_0} = \bar{C}_a \backslash \{\bar{0}\}$, где $\bar{0}$ есть нулевая КФ Z. Ограниченную ОС можно легко представить в виде двух конечных смежных и связных областей автоматов из $Z - V$ и M, где V определяет само тело среды длиной l_V, а M — границу длиной $l_M = p - 1$. Область M состоит из шаблона соседства правого крайнего автомата из V, исключая его самого. Под КФ такой ОС понимается КФ области V, а под КФ области M — граничные условия. Понятие УКФ для ограниченных ОС вводится аналогичным образом, но соотношение $W_{c_0} = \bar{C}_a \backslash \{\bar{0}\}$ заменяется на соотношение $W_{c_0} = \bar{C}_{l_V}$, где $\bar{C}_{l_V}$ есть множество всевозможных слов длиной l_V в алфавите A_a. Не нарушая общности, шаблоны соседства будем полагать связными.

Перейдем к изложению полученных результатов. В [5] показано, что ограниченные ОС могут иметь УКФ, если шаблон соседства меньше длины тела структуры только на единицу и граничные условия постоянны. Проведенный нами на ЕС-1050 обширный анализ ограниченных ОС с различными типами граничных условий дает возможность сформулировать следующую гипотезу: для любого G: 1) не суще-

ствует ограниченной ОС при периодических граничных условиях с периодом $\leqslant a^{l_V}$, имеющей УКФ для достаточно больших l_V; 2) для любого $a \geqslant 2$ существует ограниченная ОС и граничные условия такие, что для них существует УКФ при любых l_V.

Для ограниченных итеративных систем проблема существования УКФ имеет положительное решение при следующих соображениях. Очевидно, что любая конечная КФ длиной l в алфавите A_a может быть однозначно представлена числом в a-основании

$$\Pi_{(a)} = x_{l-1}a^{l-1} + x_{l-2}a^{l-2} + \ldots + x_1a^1 + x_0a^0 = \sum_{k=0}^{l-1} x_k a^k.$$

Значит, в системе счисления с основанием a получаем множество $\Pi^*_{(a)}$ всех чисел в интервале $0 \div \overbrace{y \ldots . y}^{l}$ $(y = a-1)$, если $x_k \in A_a$ $(k = \overline{0,\ l-1})$. Тогда, если производить итеративную операцию

$$\begin{cases} (y_{l-1}\ y_{l-2}\ \ldots\ y_1\ y_0)_{T+1} = (y_{l-1}\ y_{l-2}\ \ldots\ y_1\ y_0)_T + 1 \pmod a, \\ (y_{l-1}\ y_{l-2}\ \ldots\ y_1\ y_0)_1 = \overbrace{0\ 0\ \ldots\ 0}^{l} \quad (T = 1,\ 2,\ 3,\ \ldots), \end{cases}$$

то с ее помощью можно получать, естественно, все числа из интервала $0 \div \overbrace{y \ldots . y}^{l}$ $(y = a-1)$. Таким образом, алгоритм функционирования ограниченной итеративной структуры, реализуемый согласно приведенным выше соображениям, позволяет получать из нулевой начальной КФ все КФ длиной l в алфавите A_a. Эти соображения могут быть использованы при моделировании ОС на ЭВМ.

В [5] доказано, что бесконечные ОС не могут иметь даже конечного множества УКФ. Поэтому возникает вопрос о минимальном расширении алфавита A_a до алфавита $A_{a+V} \subset A_a$ с тем, чтобы над алфавитом A_a вопрос существования УКФ разрешался положительно. Оказывается, что одного вспомогательного символа вполне достаточно для реализации любого алгоритма. Идея доказательства состоит в следующем. Под КФ МТ будем понимать совокупность

$$a_{j_1}\ a_{j_2}\ \ldots\ q_i a_{j_k}\ \ldots\ a_{j_r}, \qquad (1)$$

образованную символами a_{j_k} $(k = \overline{1,r})$ состояний всех непустых ячеек ленты, состоянием внутренней памяти q_i и номером K сканируемой ячейки a_{j_k}. Работа МТ состоит в переработке КФ вида (1) с помощью программы, состоящей из команд вида

$$q_i a_j \longrightarrow a'_j q'_i \mu \qquad (\mu = -1,\ 0,\ 1).$$

По этой команде, если внутренняя память находится в состоянии q_i и символ a_j сканируется на ленте, то в следующий момент символ a_j заменяется на символ a'_j, внутренняя память переходит в состояние q'_i и сканирующая головка сдвигается на одну ячейку ленты (при $\mu = -1, 0, 1$ соответственно влево, на месте, вправо). Как известно [6], для каждой частично рекурсивной словарной функции $G(S)$, определенной в алфавите A_a, существует МТ с символами $\{\Lambda\} \vee A_a$ на ленте и подходящими внутренними состояниями q_i $(i = \overline{0,d})$, которая правильно вычисляет функцию G. Будем говорить, что МТ правильно вычисляет функцию $G(S)$, если

$$\Lambda q_1 \Lambda S \Lambda \longrightarrow \Lambda q_0 \Lambda G(S) \Lambda$$

верно для любой системы слов в алфавите A_a, принадлежащей к области определения функции G и если в случае, когда $G(S)$ не определена,

МТ, начав работать с КФ $\Lambda q_1 \Lambda S \Lambda$, никогда не остановится, т. е. не придет во внутреннее состояние (заключительное) q_0, и никогда в процессе работы не будет надстраивать ленту слева. В случае ОС алгоритмы, определенные ими, заданы на любом слове S в алфавите A_a. Следовательно, начав работать с состояния q_1, МТ, реализующая такой алгоритм, всегда приходит в состояние q_0, если функция $G(S)$ определена для данного S. Поэтому вводя кодировку

$$q_t \equiv b^{t+1}0^{d-t+2},$$

$$u \equiv b^{d+2}0^1 \quad (t=\overline{0,\ d})\ \ u \text{ — маркер}, \quad b \notin A_a,$$

и полагая начальную КФ ОС в виде

$$\overline{0}\ в^{d+2}\ 0^1\ в^2\ 0^{d+1}\ a_{j_1}\ a_{j_2}\ \ldots\ a_{j_r}\ в^{d+2}\ \overline{0}, \tag{2}$$

через ряд шагов приходим к результату

$$\overline{0}\ в^{d+2}\ 0^1\ в^1\ 0^{d+2}\ G\ (\overline{0}\ a_{j_1}\ a_{j_2}\ \ldots\ a_{j_r}\ \overline{0})\ в^{d+2}\ \overline{0}. \tag{3}$$

Затем, выбрав ОС с алфавитом $\{в\} \vee A_a$ и шаблоном соседства размером $m = 2(d+4)$, легко определить локальную функцию перехода для нее так, что она будет 1-моделировать [5] нашу МТ. Следовательно, для реализации произвольного алгоритма в алфавите A_a можно ограничиться только одним вспомогательным символом, т. е. имеет место следующее

Предложение 1. *В смысле Нагорного параллельные алгоритмы, определяемые одномерными ОС, по сложности эквивалентны нормальным алгоритмам.*

Однако возможность обойтись только одним вспомогательным символом получена благодаря увеличению размера шаблона соседства ОС. С точки зрения оценки сложности параллельных алгоритмов представляет интерес следующий вопрос: какова связь между величинами M (т. е. числом вспомогательных символов) и N (размером шаблона соседства) при реализации в одномерной ОС произвольного алгоритма в алфавите A_a?

В последнее время часто обсуждается вопрос о преимуществах параллельных алгоритмов. По-видимому, они наиболее эффективны там, где сама сущность процесса допускает достаточную степень распараллеливания. Рассмотрим тривиальный пример: словарная функция G определена на множестве W всех конечных слов в алфавите $A_a \setminus \{0\}$ и состоит в замене буквы 1 на букву 2. Такая функция легко реализуется параллельным алгоритмом вида

$$\begin{aligned} xyz &\to y, \quad && x, y, z \in A_a; \\ x1z &\to 2, && y \neq 1. \end{aligned}$$

На вычисление функции $G(S)$ при любом $S \in W$ требуется только один шаг параллельного алгоритма, тогда как для МТ, реализующей аналогичный алгоритм, не менее $l(S)$ шагов. Рассмотренный случай дает нам пример максимального распараллеливания вычислительного процесса.

При другой степени распараллеливания параллельный алгоритм будет затрачивать бóльшее число шагов. Естественно предположить, что при минимальной степени распараллеливания для вычисления функции $G(S)$ нужно иметь информацию о всем слове S и о всех промежуточных словах. Учитывая это, можно показать, что тогда слово S длиной d параллельный алгоритм, определяемый ОС с шаблоном соседства размером n, переводит в слово $G(S)$ не менее чем за $(d-n)/(n-1)$ шагов, т. е.

$$\text{ОС}(S) \geqslant (d-n)/(n-1). \tag{4}$$

Оценим теперь число шагов, затрачиваемых МТ для моделирования ОС (S) шагов ОС, если начальная КФ ОС имеет размер d. Не нарушая общности, индекс соседства выбираем в виде $\{-(n-1), \ldots, -2, -1, 0\}$ [5]. Тогда, если $c_0 = \overline{0}x_1x_2\ldots x_d\overline{0}$ и $c_0\tau = \overline{0}\,x'_1x'_2\ldots x'_d\ldots x'_{d+m}\overline{0}$ $(x_i, x'_j \in A_a;\ i = \overline{2,\ d-1};\ j = \overline{2,\ d+m-1};\ (x_1, x_d, x'_1,\ x'_{d+m} \neq 0);$ $m \leqslant n-1)$, то МТ КФ $\Lambda q_1 \Lambda c_0 \Lambda$ может перевести в КФ $\Lambda q_0 \Lambda c_0\tau \Lambda$ не более чем за $2(d+n-1)$ шагов. А так как ОС затрачивает по предположению ОС (S) шагов и при каждом шаге длина КФ S не может увеличиваться более чем на $n-1$, то на эту же работу МТ может затратить не более чем

$$\mathrm{MT}(S) \leqslant 2\{[d+(n-1)]+[d+2(n-1)]+\ldots+[d+\mathrm{OC}(S)(n-1)]\}\times$$
$$\times \mathrm{OC}(S) = (2d+(n-1)[\mathrm{OC}(S)+1])\mathrm{OC}(S) \qquad (5)$$

шагов. Используя теперь соотношение (4) и условие ОС $(S) \geqslant 1$, из неравенства (5) получаем

Предложение 2. *Для любой ОС с шаблоном соседства размером n, реализующей параллельный алгоритм θ, существует МТ, реализующая тот же алгоритм θ при выполнении условия*

$$(\forall S)\,(\mathrm{MT}(S) \leqslant 2(3n-2)\mathrm{OC}^2(S)),$$

где S — любая начальная КФ ОС.

Таким образом, любую функцию, вычисляемую ОС за время T, МТ может вычислять не хуже чем за αT^2 $(\alpha = \mathrm{const})$. Этот результат получен методом моделирования ОС посредством МТ. Были предприняты исследования обратного моделирования: смоделировать МТ посредством ОС с сокращением времени работы ОС такого же порядка. Однако из этого ничего не получилось — результат дал сокращение времени линейного порядка, которое зависело практически только от размера шаблона соседства ОС. Итак, можно сделать вывод, что такой подход и не может дать никакого эффекта, ибо мы моделируем сугубо последовательный алгоритм параллельным, не используя при этом возможностей распараллеливания. Следовательно, МТ имеет временной проигрыш квадратного порядка относительно ОС, а время работы самой ОС зависит уже от степени распараллеливания алгоритма, реализуемого ОС. И значительный временной выигрыш при реализации алгоритмов на ОС мы получим только тогда, когда сможем эффективно распараллеливать реальные алгоритмы и классифицировать их согласно этому критерию.

Из предложения 2 ни в коем случае не следует, что за счет применения ОС нельзя сократить время вычислений более существенно, чем порядка корня квадратного относительно времени, затрачиваемого самой оптимальной МТ. Это утверждение следует из вышеприведенных рассуждений. Более того, на наш взгляд, существует целая шкала распараллеливания алгоритмов, на противоположных концах которой и располагаются рассмотренные нами примеры алгоритмов, реализуемых на ОС. Это еще один довод в пользу необходимости исследований в области оценки степени допустимости распараллеливания алгоритмов для выяснения целесообразности реализации их на ОС, хотя эта проблема и представляется нам чрезвычайно сложной.

В исследовании принципиально новых архитектур ЭВМ одним из наиболее интересных вопросов является разработка параллельных процессоров или однородных вычислительных сред. Может показаться, что такие ЭВМ найдут ограниченное применение, но исследования в этой области позволят, по-видимому, глубже разобраться в задачах, которые могут решаться параллельно, а не последовательно. Параллельная

обработка имеет к этому прямое отношение, хотя и связана скорее с теорией параллельных алгоритмов. И здесь в первую очередь предполагается целесообразной именно разработка теории параллельных алгоритмов с тем, чтобы вначале выяснить целесообразность распараллеливания той или иной задачи, а затем уже строить параллельные процессоры для решения таких задач, ибо может случиться так, что построив ЭВМ высокопараллельного действия, мы будем искать как задачи для нее, так и способ их наилучшего представления. В противном же случае эффективности по сравнению с последовательными ЭВМ или мультипроцессорными вычислительными системами можно и не добиться. Например, уже для такой ЭВМ высокопараллельного действия, как машина Холланда [7], остро встает вопрос о классе задач, наиболее эффективно решаемых на ней. Более того, изучение параллельных алгоритмов, определяемых ОС, приводит к мысли, что и на понятие «параллельный алгоритм» пока не существует единой точки зрения. Очень трудно отождествлять некоторые параллельные алгоритмы, определяемые ОС, и алгоритмы, весь параллелизм которых состоит в наличии нескольких независимых ветвей (именно такой тип алгоритмов эффективно реализуется на мультипроцессорных вычислительных системах [8]). Суть же параллелизма, вероятно, намного сложнее. Прогресс в этом направлении поможет нам глубже взглянуть на вычислительный процесс в целом.

ЛИТЕРАТУРА

1. Алферова З. В., Математическое обеспечение экономических расчетов с использованием теории графов, М., 1974.
2. Евреинов Э. В., Косарев Ю. Г., Однородные вычислительные системы высокой производительности, Новосибирск, 1966.
3. Прангишвили И. В. и др., Микроэлектроника и однородные структуры для построения логических и вычислительных устройств, М., 1967.
4. Нагорный Н. М., К усилению теоремы приведения теории нормальных алгоритмов, ДАН СССР, **90**, № 3 (1953).
5. Аладьев В. З., К теории однородных структур, Таллин, 1972.
6. Мальцев А. И., Алгоритмы и рекурсивные функции, М., 1965.
7. Essays on Cellular Automata (edited by A. W. Burks), University of Illinois Press, 1970.
8. Мультипроцессорные вычислительные системы, под ред. Я. А. Хетагурова, М., 1971.

Эстонский филиал ВГПТИ ЦСУ СССР

Поступила в редакцию
17/VII 1974

V. ALADJEV, O. OSIPOV

HOMOGEENSETE STRUKTUURIDEGA MÄÄRATUD PARALLEELSETE ALGORITMIDE KOMPLITSEERITUSEST

Artiklis vaadeldakse ühedimensiooniliste homogeensete struktuuridega määratud paralleelsete algoritmide komplitseerituse hinnangut. Tõestatakse, et sellised algoritmid on Markov-Nagornõi mõttes ekvivalentsed normaalalgoritmidega; veelgi enam: kui sellise algoritmiga kulub mingi osaliselt rekursiivse funktsiooni arvutamiseks T sammu, siis vastavalt Turingi masinal kulub selleks mitte rohkem kui αT^2 sammu ($\alpha = \text{const}$).

V. ALADYEV, O. OSIPOV

ABOUT THE COMPLEXITY OF PARALLEL ALGORITHMS DEFINED BY HOMOGENEOUS STRUCTURES

In this paper the problem of estimation of complexity of parallel algorithms defined by one-dimensional homogeneous structures is discussed. It is proved that such algorithms in Markov-Nagorny's sense are equivalent to normal algorithms. Furthermore, if such an algorithm computes some partial recursive function in T steps, then, correspondengly, the Turing machine can compute this function in no more than αT^2 steps ($\alpha = \text{const}$).