УДК 005.22

[1]А.Л. Третьяков, [2]Г.С. Хулап, [3]С.Д. Бушуев

[1]Московский государственный университет, Москва
[2]Російська академія наук, генеральний директор «ТЕХІНТЕЛ», Росія
[3]Киевский национальный университет строительства и архитектуры, Киев

НОВАЯ КОНЦЕПЦИЯ ОРГАНИЗАЦИИ ГЛОБАЛЬНОЙ ВЫЧИСЛИТЕЛЬНОЙ АРХИТЕКТУРЫ

Предложено новую концепция организации вычислительного процесса таким образом, что количество последовательных одновременных тактовых операций (или число векторных операций) не зависит от числа п — *размерности задачи. При этом архитектура вычислительной среды адаптирована под конкретную решаемую задачу и вычисление осуществляется без обмена информацией между элементарными вычислительными устройствами — элементарными процессорами, число которых зависит от* п. *Описан алгоритм реализации данной идеологии на примере решения задачи многоэкстремальной оптимизации (или выбора максимального из* п *заданных чисел), а также алгоритм решения задачи коммивояжера.*

Ключевые слова: концепция, вычеслительная архитектура, модель вычеслительного процесса

НОВА КОНЦЕПЦІЯ ОРГАНІЗАЦІЇ ГЛОБАЛЬНОЇ ОБЧИСЛЮВАЛЬНОЇ АРХІТЕКТУРИ

Пропонується нова концепція організації обчислювального процесу таким чином, що кількість послідовних одночасних тактових операцій (або число векторних операцій) не залежить від числа n - розмірності задачі. При цьому архітектура обчислювального середовища адаптована під конкретну задачу й обчислення здійснюється без обміну інформацією між елементарними обчислювальними пристроями - елементарними процесорами, число яких залежить від n. Описано алгоритм реалізації даної ідеології на прикладі рішення задачі багатоекстремального оптимізації (або вибору максимального з n заданих чисел), а також алгоритм розв'язання задачі комівояжера 11-4].

NEW CONCEPT OF ORGANIZATION GLOBAL COMPUTER ARCHITECTURE

A new concept of organization of the computational process in such a way that the number of concurrent serial clock operations (or the number of vector operations) does not depend on the number n - the dimension of the problem. The architecture of computer ¬ tion environment adapted to the specific problem being solved and the calculation is carried out without the exchange of information between the elementary computing devices - elementary processors, the number of which depends on n The algorithm for the implementation of this ideology in the Multiple-solution problem of optimization (or select the maximum of n given numbers), and the algorithm for solving the traveling salesman problem 04.11].

Введение

В последнее время все больше внимания специалисты различных областей знаний уделяют необходимости решения больших и сложных задач эффективными и надежными средствами. Однако естественно заложенное в проблему “проклятие” последовательной обработки информации резко ограничивает возможности исследователя. Даже использование новых технологий таких, как параллельные вычисления, многопроцессорные системы и т.д., не дают полной универсальности и так или иначе зависят от сложности решаемой задачи, которую в дальнейшем мы будем называть размерностью и обозначать через *n*.

Обработка больших массивов однотипной информации требует существенных вычислительных затрат, связанных с многими причинами, в числе которых одной из главных является необходимость обмена информацией между элементарными процессорами. В данной работе предлагается математическая модель организации вычислительного процесса таким образом, что отсутствует обмен информацией между элементарными процессорами и количество параллельных арифметических операций не зависит от числа *n*. Основная идея подхода излагается на примере решения задачи глобальной максимизации на отрезке, хотя все рассуждения и выводы справедливы и для задачи максимизации на *n*-мерном параллелепипеде. Следует отметить, что идея глобальной обработки информации использовалась ранее в [5; 6] для решения ряда задач теплофизики и телекоммуникаций.

I. Модель глобальной вычислительной архитектуры. Пусть имеется следующая задача оптимизации:

$$\max \varphi(x), \ x \in [a,b], \qquad (1)$$

где функция $\varphi(x)$ задана своими значениями в n точках: $x_1 = a,\ x_2, ..., x_{n-1}, x_n = b$ (рис. 1).

Теперь представим себе, что мы хотим "увидеть" сразу поведение всей поверхности исследуемой функции (на отрезке). Здесь "поверхность" понимается в смысле совокупности точек ее задания - n точек, т.е. мы хотим "увидеть" сразу весь ансамбль $\{\varphi(x_i)\}$ значений функции φ в n точках и, не перебирая последовательно значения функции в точках x_i, сразу охватить единым взглядом всю картину "глобального" поведения многоэкстремальной поверхности с точки зрения выявления нужного нам свойства – в данном случае поиска глобального максимума.

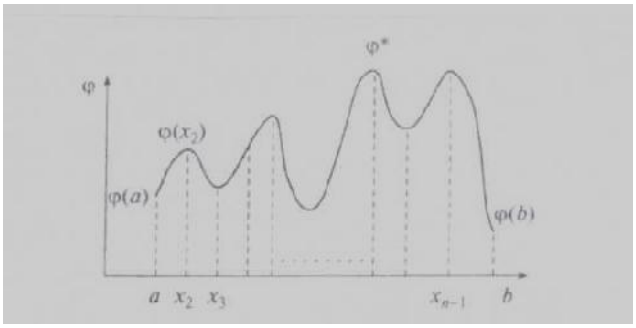

Рис. 1.Ансамбль $\{\varphi(x_i)\}$ значений функции $\varphi(x)$ в n точках x_i

Описываемый подход можно проиллюстрировать на модельном примере с "забором", у которого заменим одну "старую", темную доску на светлую, "новую". Выявление светлой ("новой") доски осуществляется сразу, без перебора всех досок. Чтобы отыскать "новую" доску (глобальный максимум), мы не будем перебирать все "старые", темные доски (сравнивать все значения функции в точках x_i,), а сразу отреагируем — по другому признаку — и укажем "новую" доску [4;7].

Таким образом, задача состоит в построении математической модели вычисления (выбора) максимального элемента из n значений $\{\varphi(x_i)\}$ функции $\varphi(x)$ в точках $x_к$ со сложностью, независящей от n. Обозначим $N_k = \varphi(x_k),\ k = 1,2,...,n$ „ и рассмотрим задачу определения максимального элемента

$$N_p = \max_{k=q,n} N_k. \qquad (2)$$

Здесь n (число величин $N_к$) – размерность данной задачи. Для простоты изложения предположим, что все $N_к$ — целые положительные числа и $N_i \neq N_j,\ i \neq j,\ j = \overline{1,n}$. Отметим, что сложность традиционных алгоритмов решения этой задачи с использованием параллельных вычислений (n процессоров) будет порядка $O(\log_2 n)$. В нашей модели мы также будем использовать n элементарных идентичных процессоров Pr_k для вычисления значений $\varphi(x_k)$ (или чисел N_k), $k = \overline{1,n}$. Считаем, что каждый элементарный процессор Pr_k преобразует значение N_k в световой сигнал соответствующей высоты $H_k = N_k = \varphi(x_k),\ k = \overline{1,n}$ (рис. 2).

Имеется главный процессор ***MP***, который реагирует только на световой сигнал от элементарных процессоров (не важно, от какого) и управляет подвижным блокирующим вертикальным экраном, который может двигаться либо вверх, либо вниз на заданную величину (рис. 2). Если высота экрана больше $H_к$, то свет от источников Pr_i таких, что $H_i \le H_k,\ i \in \{\overline{1,n}\}$, блокируется и не доходит до главного процессора ***MP***.

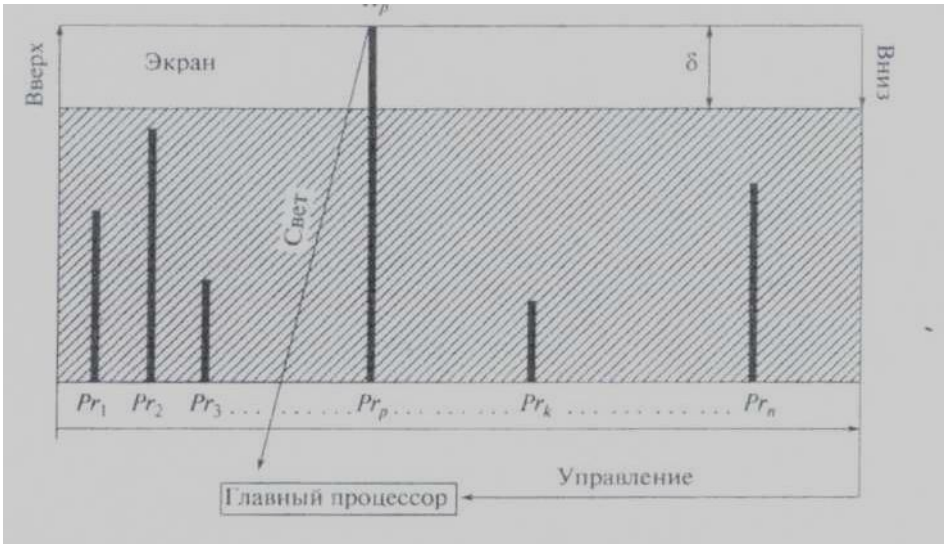

Рис. 2. Модель глобальной вычислительной архитектуры нахождения максимального элемента из n чисел

Модель работает следующим образом. Главный процессор ***MP*** осуществляет поднятие экрана до тех пор, пока есть световой сигнал хотя бы от одного источника Pr_k. Если светового сигнала нет, т.е. все источники блокированы, главный процессор ***MP*** останавливает экран и осуществляет его движение вниз на величину δ (точность получаемого решения). В нашем случае δ = 1/2. Световой сигнал появляется снова, но уже только от одного источника Pr_p, соответствующего максимальному значению N_p (рис. 2).

Такая модель вычислительной архитектуры позволяет одновременно передавать информацию о номере p и максимальном значении N_p вместе со световым сигналом и базируется на двух аналоговых операциях:

1. – поднятие экрана до тех пор, пока световой сигнал есть;
2. – остановка поднятия экрана при отсутствии светового сигнала и опускание экрана на величину δ.

В дальнейшем будем за одну базовую операцию в данной модели считать:

- поднятие (опускание) экрана до момента детекции света (либо свет есть, либо света нет);
- поднятие (или опускание) экрана на заданную величину.

Число операций (сложность) в данной модели не зависит от величины n. Общее время вычислений зависит от величины максимального элемента N_p и будет определяться числом операций поднятия экрана для достижения точки $H_p = N_p$.

2. Алгоритм глобального типа определения максимального элемента из массива.

Пусть имеется массив n натуральны x S из n чисел $a_i,...a_n$, где $a_i < 0,\ i = \overline{1,n}$. Требуется определить l^* — максимальное значение a_p из массива S, т.е.

$$l^* = \max_{i=1,n} a_i = a_p.$$

Будем использовать модель глобальной вычислительной архитектуры, описанную выше, с использованием световой детекции.

Обозначим

$\delta = \min\left\{\left|a_i - a_j\right| \middle| i, j \in \{1,...,n\}, a_i \neq a_j\right\}$. Алгоритм вычисления максимального элемента из массива на основе глобальной вычислительной архитектуры будет следующим [1;4; 8].

Алгоритм 1 (Метод удвоений [9, 10]).

Шаг 1. Полагаем $l_0 = 0, l_1 > 0,$, где l_0, l_1 — высоты экрана, выбранные таким образом, что при высоте l_1 свет есть, а при высоте экрана $2l_1$, нет детекции света. Предположим, что при $\tau = \overline{1,k}$ свет есть при высоте l_m и света нет при высоте $l_m + (l_m - l_{m-1})$. Вычисляем следующее l_{k+1}, $m = k+1$.

Шаг к + 1. Проверяем, выполнено ли неравенство

$$l_k - l_{k-1} < \delta / 2. \qquad (3)$$

Если неравенство выполнено, то полагаем $K = k + 1$ и STOP. В противном случае полагаем $s_k = 1$ и идем на А2.

А1. Вычисляем

$$l_{s_k} = l_k + \frac{l_1 - l_{k-1}}{2^{s_k}}.$$

А2. Проверяем, есть ли световой сигнал для высоты экрана l_{s_k}.

Если света нет, то $s_k = s_k + 1$ и идем на **А1**.

Если свет есть, то $l_{k+1} = l_{s_k}, k = k1$ и идем на шаг $k + 1$.

Пусть K - число операций, необходимых описанному выше алгоритму для достижения максимального элемента a_p, т.е. выполнено условие (3) $\left|l_{k+1} - l_k\right| \delta / 2$, гарантирующее, что максимальный элемент $l^* = a_p$ локализован за $K = k + 1$ итераций (шагов). Для оценки асимптотической сложности K алгоритма будет верна следующая теорема.

Теорема 1. Пусть $A = 2l^*$. Тогда число операций K в алгоритме 1 будет

$$K \sim \log_2 \frac{A}{\delta}$$

Замечание 1. Число операций K не зависит от n.

Доказательство. Без ограничения общности рассмотрим случай S_k=1 для любых k. Общий случай рассматривается аналогично. В соответствии с алгоритмом при высоте экрана l_k есть свет, а при $l_k + (l_k - l_{k-1})$ — нет. По построению мы имеем

$$l_{k+1} - l_k \le \frac{l_k - l_{k-1}}{2} \text{ и } l_k < l^* \le l_k + (l_k - l_{k-1}).$$

Аналогично

$$l_{k+1} - l_k \le \frac{l_1 - l_0}{2^k} \le \frac{A}{2^k}.$$

Ближайшее a_1, отличается от l^* на величину, превосходящую или равную δ, поэтому номер шага K для определения l^* должен удовлетворять соотношению

$$\frac{A}{2^K} \le \frac{\delta}{2}. \quad (4)$$

Учитывая целочисленность K, находим, что число шагов (операций) для выполнения неравенства (4) может быть равным

$$K = \left[\log \frac{2A}{\delta} \right] + 1,$$

$$\text{т.е. } K \sim \log_2 \frac{A}{\delta}.$$

Замечание 2. В реальной ситуации значение δ больше, чем машинно-возможное представимое число в компьютерной архитектуре, т.е. “машинный ноль” 0_M. В нашей модели мы можем взять $l^*/\delta < M$, где M соответствует числу $1/0_M$. Поэтому общее число операций K будет не более

$$\sim \log_2 \frac{2A}{\delta} \le \log_2 \frac{2AM}{l^*} = \log 4M.$$

Заметим, что число *M* не зависит от *n*.

Замечание 3. Для проблемы Кука мы имеем булеву функцию *f(x)* от переменной $x = (x_1, ..., x_n)$, принимающую только на одном наборе $x^* = (x_1, ..., x_n)$ значение $1 : f(x^*) = (x^*_1, ..., x_n^*)$ и значение 0 на остальных, т.е. $f(x) \neq 0$, если $x \neq x^*$. Проблема состоит в отыскании числа x^*. Если предположить, что мы имеем $n!$ элементарных процессоров $\Pr_k$, каждый из которых вычисляет $f(x_1^{(k)}, ..., x_n^{(k)})$, то δ = 1 и при использовании метода удвоения нам потребуется всего лишь одна операция, т.е. $\log_2$ 2 = 1. Однако для этого необходимо $n!$ элементарных процессоров.

В свою очередь, если предположить, что мы имеем компьютер с бесконечной лентой ячеек, в каждой из которых находится элементарный процессор, аналогично машине Тьюринга с бесконечной лентой ячеек из нулей и единиц, то проблема будет решена за одну операцию.

3. Применение алгоритма глобального типа для решения задачи коммивояжера (построение минимального маршрута). Рассмотрим теперь более сложную оптимизационную задачу. Покажем, как применить алгоритм глобального типа для решения задачи коммивояжера — главного представителя задач экспоненциальной сложности, так называемых NP-полных (non polynomial) проблем.

Пусть имеется граф с $n+1$ вершинами $a_0, a_1, \ldots, a_n$, координаты которых $a_0(x_0, y_0)$, $a_1(x_1, y_1), ..., a_n(x_n, y_n)$ соответственно и где a_0 - фиксированная начальная точка, a_n - фиксированная конечная точка. Необходимо найти минимум функции $f(i_1, i_2, ..., i_{n-1})$, такой, что $f^*(i^*_1, i^*_2, ..., i^*_{n-1})$ – наименьшее расстояние между вершинами $a_0, a_i, ..., a_{o_{n-1}} a_n$, где $i_j \neq i_k$, если $j \neq k$, $i_j, i \in \{1, ..., n-1\}, j, k \in \{1, ..., n-1\}$. Определим f следующим образом:

$$f(i_1, ..., i_{n-1}) = dist(a_0, a_{i_1}) + dist(a_{i_1}, a_{i_2}) + ... dist(a_{i_{n-1}}, a_n) =$$
$$= \sqrt{(x_{i_1} - x_0)^2 + (y_{i_1} - y_0)^2} + ... + \sqrt{(x_n - x_{i_{n-1}})^2 + (y_n - y_{i_{n-1}})^2}. \quad (5)$$

Необходимо вычислить

$$\min_{\substack{i_k \in \{1,...,n\} \\ i_l \neq i_j, l \neq j}} f(i_1, ..., i_{n-1}) = f(i^*_1, ..., i^*_{n-1}).$$

Иными словами, надо найти минимум из $n!$ возможных значений $f(i_1, ..., i_{n-1})$.

Сейчас мы опишем алгоритм нахождения минимального маршрута (пули), проходящего через вершины $a_0, a_1, ..., a_n$ с фиксированным началом a_0 и фиксированным концом a_n.

Алгоритм 2 (построение кратчайшего пути [1]).

Шаг 1. По формуле (5) вычисляются все возможные расстояния между двумя вершинами dist $(a_i, a_j) i, j \in \{0, 1, ..., n\}$. Заметим, что мы должны подсчитать расстояние между всеми возможными парами, например между $\{a_i, a_j\}$ и $\{a_j, a_i\}$.

Шаг 2. Помечаем первую вершину в паре как a_i^- и вторую — как $a_i^+, i \in \{1, ..., n-1\}$. Это нужно для идентификации входа и выхода конструируемого пути в соответствующей паре $\{a_i, a_j\}$.

Шаг 3. Конструируем каждый возможный путь длины n таким способом, чтобы вершина a_0 входила в него только один раз и была выходом, т.е. a_0 и вершина a_n входила только один раз и была входом, т.е. a_i^+. Остальные вершины $a_i, i \in \{1,...,n-1\}$, присутствуют в пути ровно 2 раза, как вход и выход соответственно a_i^+ и a_i^-,т.е.

$$\left\{(a_0, a_{i_1}^+), (a_{i_1}^-, a_{i_2}^+), ..., (a_{i_{n-1}}^-, a_n\right\}.$$

Обозначим через $IJ(1, n-1)$ множество всевозможных цепочек индексов $\{i_1,...,i_{n-1}\}$, построенных в соответствии с шагами 1 —3.

Шаг 4. Вычисляем длину каждого пути

$$f(i_1,\ldots,i_{n-1}) = l(a_0, a_{i_1},\ldots, a_{i_{n-1}}, a_n) = dist(a_0, a_{i_1}^+) + \\ + dist(a_{i_1}^-, a_{i_2}^+) + \ldots + dist(a_{i_{n-1}}^-, a_n).$$

Шаг 5. Определяем минимальный путь, построенный в соответствии с алгоритмом, решая задачу минимизации

$$l^* = l(a_0, a^*_{i_1},..., a^*_{i_{n-1}}, a_n = \min_{\{i_1,...,i_{n-1}\} \in IJ(1,n-1)} l(a_0, a_{i_1},..., a_n).$$

Обоснованием описанного выше алгоритма является следующая теорема [1].

Теорема 2. Алгоритм 2 реализует все возможные пути с началом в точке a_0 и концом в точке a_n и проходящие через вершины $a_i, i = \overline{1, n-1}$, ровно один раз. Набор $\left\{a_0, a^*_{i_1},..., a^*_{i_{n-1}}, a_n\right\}$ является решением задачи, т.е.

$$l(a_0, a^*_{i_1},..., a^*_{i_{n-1}}, a_n = \min_{\substack{i \in (\overline{1,n-1}) \\ j=\overline{1,n-1} \\ i_j \neq i_k, j \neq k}} l(a_0, a_{i_1},...a_{i_{n-1}}, a_n).$$

Доказательство. Очевидно, что достаточно доказать первое утверждение теоремы, т.е. что алгоритм 2 реализует все маршруты (пути), проходящие через точки $a_1,...,a_{n-1}$, начинаясь в точке a_0 и заканчиваясь в точке a_n. Для этого рассмотрим фиксированный путь, проходящий через точки $a_0, a_{i_1},..., a_{i_{n-1}}, a_n$. Вершины a_0 и a_n входят ровно один раз в каждый путь в соответствии с алгоритмом 2. Так как каждая $a_{i_k}, k = \overline{1, n-1}$, включена в (3) ровно 2 раза (с индексом "-" и "+"), то путь $\left\{(a_0, a_{i_1}^+), (a_{i_1}^-, a_{i_2}^+), ..., (a_{i_{n-1}}^-, a_n\right\}.$, построенный в соответствии с алгоритмом 2, удовлетворяет условию прохождения только один раз через каждую вершину, пройдя через все вершины $a_i,...,a_{n-1}$.

Далее доказательство теоремы будет закончено, если мы покажем, что реализуются все маршруты, проходящие через точки $a_1,...,a_{n-1}$, ровно один раз с началом в точке a_0 и концом в точке a_n. Докажем это от противного. Пусть наш алгоритм 2 не реализует некоторый маршрут $\left\{a_0, a_{k_1},..., a_{k_{n-1}, a_n}\right\}$. Но это означает, что не подсчитано по крайней мере одно значение из набора $dist(a_0, a_{k_1}),..., dist(a_{k_{n-1}}, a_n)$, а это противоречит тому, что реализуются (считаются) все возможные расстояния $dist(a_{i_1}, a_{i_2}), i_1, i_2 \in \{0,1...,n\}$, в алгоритме 2, причем с соответствующим порядком следования, т.е. либо $\left\{a_{i_1}, a_{i_1}\right\}$, либо $\left\{a_{i_2}, a_{i_1}\right\}$ (соответственно $(a_{i_1}^-, a_{i_2}^+)$ и $(a_{i_2}^-, a_{i_1}^+)$. Теорема доказана.

Теперь покажем, как может быть реализован алгоритм 2 с использованием глобальной вычислительной архитектуры.

4. Модель многоуровневой, объектно-ориентированной на решение задачи коммивояжера глобальной вычислительной архитектуры.

Мы могли бы смоделировать вычислительную архитектуру решения задачи коммивояжера на основе метода удвоения, подобную поиску максимального элемента, описанную в разд. 1, 2. Однако в этом случае нам потребуется *n!* элементарных процессоров, где *n* — число точек обхода при построении маршрута. Более того, если *n!* процессоров параллельно сравнивают пары длин маршрутов, мы получим минимальный маршрут за порядка $\log_2 n! \sim n \log_2 n$ операций. Однако с использованием глобальной вычислительной архитектуры мы можем организовать вычисление при помощи только *n(n*-1) элементарных процессоров, вычисляющих длины пар различных вершин, и сохранить алгоритмическую сложность решения проблемы коммивояжера, линейно-логарифмически зависящей от размерности задачи *n,* т.е. ~*n* $\log_2$ *n.*

Пусть все элементарные процессоры $\Pr_k, k = \overline{1, n(n-1)}$, получают одновременно информацию о координатах $(x_{i_1}, y_{i_1}), i_1 \in \left\{\overline{o, n}\right\}$, от главного процессора *MP* и каждый конкретный процессор $\Pr_k$ выделяет соответствующую ему пару (x_{i_1}, y_{i_1}), (x_{i_2}, y_{i_2}) и вычисляет соответствующее расстояние между точками a_{i_1}, a_{i_2}. Это занимает две операции у каждого элементарного процессора. При этом соответствие между номером процессора *к* и

координатами i_1, i_2 можно определять различными способами, например $k = i_1 n + i_2$.

Итак, пусть все элементарные процессоры вычисляют расстояние $\mathrm{dist}(a_i, a_j)$ между всевозможными вершинами a_i, и a_j, из множества вершин $\{a_0, a_1, ..., a_n\}$, где каждая вершина помечена либо a_i^-, либо a_i^+, в зависимости от позиции a_i в паре:

- для пары (a_i, a_j) будет a_i^-
- для пары (a_h, a_i) будет a_i^+.

Предположим, что каждый элементарный процессор снабжен физическим устройством, испускающим электромагнитную волну с частотой $V_{i_{k+1}}$ соответствующей длине $dist(a_{i_k}, a_{i_{k+1}})$, которую он вычислил, т.е.

$$V_{i_{k+1}} \sim dist(a_{i_k}, a_{i_{k+1}}),\ k = \overline{0, n}.$$

При этом происходит одновременное испускание электромагнитных волн от всех Pr_k элементарных процессоров, частоты которых при взаимодействии (интерференции) соответственно суммируются, причем информация о числах i_k и i_{k+1} может одновременно передаваться при наложении волн от всех $n(n\text{-}1)$ передатчиков (процессоров) (см. рис. 3) [4].

Такой способ позволяет получить все возможные комбинации волн различных частот, соответствующие возможным маршрутам, проходящим через точки a_0 и a_n, в форме суммы частот

$V_{i_{n-1}} = V_{i_0} + V_{i_1} + ... + V_{i_{n-1}}$, где минимальному маршруту соответствует минимальная частота. Однако нам надо выделить только подходящие волны, т.е. проходящие через точки $a_1, ..., a_{n-1}$ только один раз с началом в точке a_n и концом в точке $a_{,,}$. Для этого в модель вводится фильтрующий буфер, пропускающий только электромагнитные волны, которые содержат не менее *n* пар точек, причем точки, повторяющиеся не более двух раз и с различной индикацией. Точки a_0 и a_n должны входить в пропускаемую волну только один раз. Оставшиеся сигналы (волны) соответствуют маршрутам, построенным в алгоритме 2.

Теперь надо выбрать минимальный путь из всех построенных. Согласно описанной модели, кратчайшему маршруту длины $l(a_0, a^*_{i_1}, ..., a^*_{i_{n-1}}, a_n)$ соответствует волна с минимальной частотой

$$V^*_{i_{n-1}} = V^*_{i_0} + V^*_{i_1} + ... + V^*_{i_{n-1}},$$

Последняя операция осуществляется приемным устройством, которое преобразует каждую волну с частотой представляющую собой длину маршрута $\{a_0, a_{i_1}, ..., a_{i_{n-1}}, a_n\}$, в адекватный световой сигнал высоты $H_{i_1,...,i_{n-1}}$ таким образом, что $H_{i_1,...,i_{n-1}} \sim (-f(i_1, ..., i_{n-1}))$, т.е. пропорционально $-f(i_1, ..., i_{n-1})$. После этого можно применить метод удвоения нахождения максимального элемента, описанный в разд. 2, для выделения столбца $H_{i^*_1,...,i^*_{n-1}}$ реализующего

$$\min_{\substack{i_k \in (\overline{1,n-1}) \\ j_l \neq 1, l \neq 1}} f(i_1, ..., i_{n-1}) = f^*(i^*_1, ..., i^*_{n-1}),$$

т.е. решению задачи коммивояжера.

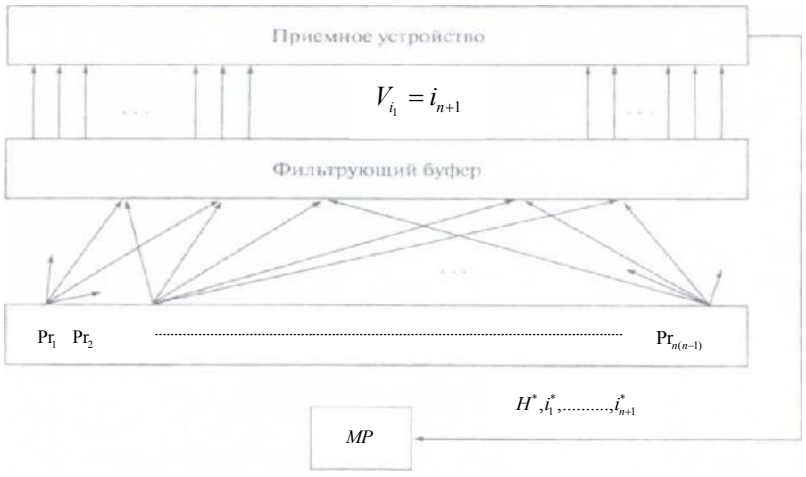

Рис. 3. Модель многоуровневой глобальной вычислительной архитектуры с фильтрующим буфером и принимающим устройством

5. Алгоритм глобальной вычислительной архитектуры для решения задачи коммивояжера.

Сделаем следующие предположения. Пусть

$$f(i_1,...,i_{n-1}) = dist(a_0, a_i, ..., a_{i_{n-1}}, a_n) \tag{6}$$

$$\left| f(i_1,...,i_{n-1}) - f(j_1,..., j_{n-1}) \right| \geq \delta > 0,$$

Если

$$f(i_1,...,i_{n-1}) \neq f(j_1,..., j_{n-1}), i_k \neq i_k,\ j_k \neq j_l,\ k \neq l.$$

Условие (6) вполне естественно для задачи коммивояжера.

Так как число разных $dist(a_0, a_{u_1}, ..., a_{i_{n-1}}, a_n)$ равно

$$n!, \text{ то } dist(a_0, a_{u_1}, ..., a_{i_{n-1}}, a_n) \leq cn!,$$

где

$$c = \max_{i \neq j, i\ j \in \overline{\{0,n\}}} dist(a_i, a_j),$$

— не зависящая от n константа. В качестве величины $H_{i^*_1,...,i^*_{n-1}}$ можно взять, например

$$H_{i^*_1,...,i^*_{n-1}} = M - f(i_1,...,i_{n-1}),$$

где $\boldsymbol{M = 2c.}$ Минимальный путь $l^*(a_0, a^*_{i_1}, ..., a^*_{i_{n-1}}, a_n)$ соответствует максимуму $H^* = H_{i^*_1,...,i^*_{n-1}}$, причем $\left| H_{i_1,...,i_{n-1}} - H_{j_1,...,j_{n-1}} \right| \geq \delta$, если $H_{i_1,...,i_{n-1}} \neq H_{j_1,...,j_{n-1}}, i_k \neq i_l,\ j_k \neq j_l$, при $k \neq l$. Это означает, что точность δ гарантирует однозначное определение максимума H^* (а значит, и l^*) с использованием метода удвоения, представленного в алгоритме I. В данной ситуации можно считать δ =1.

Алгоритм 3 (метод удвоения для задачи коммивояжера).

Шаг 1. Полагаем δ =1; $H_к$ — высота экрана на шаге *к*. Пусть H_0=0, H_1 >0, такие, что мы имеем детекцию света при высоте экрана H_1, а при высоте экрана 2 H_1, света нет.

Пусть для всех $m = \overline{1,k}$ для высот экрана H_m есть свет, а при высоте экрана H_m +(H_m-$\mathrm{H_{т\text{-}1}}$) нет светового сигнала. Опишем построение элемента $H_{m+1}, m = k+1$..

Шаг к+*1*. Проверяем $H_k - H_{k-1} < 1/2$. Если это условие выполнено, полагаем *К* = *к* + 1 и STOP. В противном случае полагаем $S_k = 1$.

A1:

$$H_{s_k} = H_k + \frac{H_k - H_{k-1}}{2^{S_k}}.$$

Проверяем наличие света при H_{S_k} .

Если “нет”, то $s_k = s_k + 1$ и идем на **A1**. Если “да”, то $H_{k+1} = H_{S_k}$, $k = k+1$ и идем на шаг *к* + 1. Покажем, что описанный выше алгоритм определяет максимальное значение *H** (или, соответственно, минимальный путь *l** за *К* шагов, причем *К* пропорционально $n\log_2 n (K \sim n\log_2 n)$. Условие $H_{k+1} - H_k \leq 1/2$ гарантирует однозначное определение значения максимального элемента *H**, даже если оно достигается при нескольких значениях различных аргументов $\{i_1,...,i_{n-1}\}$.

Теорема 3. Число шагов (операций) *К* в описанном алгоритме 3, основанном на модели многоуровневой глобальной вычислительной архитектуры для определения кратчайшего маршрута в задаче коммивояжера с n+1-й вершиной пропорционально $n\log_2 n$ (т.е. $L \sim n\log_2 n$).

Доказательство. Необходимо показать, что число операций (шагов) *К* для достижения точности $\left| H_{K+1} - H_K \right| \leq 1/2$ пропорционально $n\log_2 n$. Предположим, что на каждом шаге *к* число $s_k = 1$. Общий случай рассматривается аналогично. Для каждого k имеем

$$H_{K+1} - H_K = \frac{1}{2}\left| H_k - H_{k-1} \right| = \frac{1}{2}\left| H_1 - H_0 \right| \leq \frac{2cn!}{2^{k+1}} = \frac{cn!}{2^k}.$$

Необходимо найти такое K, для которого выполняется неравенство

$$\frac{cn!}{2^k} \leq \frac{1}{2}. \tag{7}$$

Из формулы Стирлинга [3] имеем

$$n! \sim \frac{dn^{n+1/2}}{e^n},\ d = \sqrt{2\pi} < 3.$$

Очевидно, что $n! < 3\left(\frac{n}{e}\right)^n n^{1/2}$. Это означает, что если

$$\frac{1}{2^K} \le \frac{1}{6c\left(\frac{n}{e}\right)^n n^{1/2}},$$

то выполнено условие

$$\frac{cn!}{2^K} \le \frac{1}{2}.$$

Используем (8) для определения *K*. С этой целью решим следующее уравнение:

$$\frac{1}{2^x} = \frac{1}{6c\left(\frac{n}{e}\right)^n n^{1/2}}, \quad (8)$$

откуда

$$x = \log_2 6c\left(\frac{n}{e}\right)^n n^{1/2} = n\log_2 6c\left(\frac{n}{e}\right) + \frac{1}{2}\log_2 n.$$

Если взять

$$K = \left[n\log_2 6c\left(\frac{n}{e}\right) + \frac{1}{2}\log_2 n \right] + 1,$$

то условие (7) будет заведомо выполнено, и мы однозначно определим *H**. Учитывая, что $n \ge 1$, $c \le 1$ и c — независимая от *n* константа, получаем $K \sim n\log_2 n$, т.е. $K \sim n\log_2 n$.

Виводы

В данной статье приводится модель решения задачи оптимизации, поиска максимального элемента на основе глобальной вычислительной архитектуры. Данный подход позволяет решить задачу про число итераций, не зависящее от числа заданных элементов *n*. Применение описанной идеологии для решения задачи коммивояжера обеспечивает линейно-логарифмическую сложность вычислений с использованием порядка n^2 элементарных процессоров. Основой предложенной модели является глобальная вычислительная архитектура, использующая традиционный метод удвоения. Изложенный подход дает возможность создания вычислительных систем. построенных на глобальной объектно-ориентированной архитектуре.

Список литературы

1. *Rarczak A., Tret'yakov A., Zakrzewski L. NP-complete Problems and Global Compulations // Proc. Artificial Intelligence Studie. 2006. V. 3 (26).*

2. *Галкина В.А. Дискретная математика: комбинаторная оптимизация на графах– М.: Гели ос ЛРБ, 2003.*

3. *Кнут Д. Искусство программирования. Т. 1—3. М.: "Вильямс", 2008.*

4. *Tret'yakov A., Zakrzewski L. Hybrid intelligent techniques for solving nonlinear problems // Scientific Bulletin of Chelm. Section of Mathematics and Computer Science. Chelm, 2006. № 2.*

5. *Хулап Г.С., Логинов В.И. Применение непараметрических критериев для оценки качества уравнений связи // Изв. вузов. Геодезия и аэрофотосъемка, 1974. –№ 1*

6. *Попов Ю.А., Карпельсон А.Е., Хулап Г.С. Возможности активного теплового контроля сотовых конструкций //Дефектоскопия. 1977. № 4.*

7. *Tret'yakov A. The global calculus and the synthesis of the new knowledge // The global calculus and the future of computer science. Biala Podlaska, 2004.*

8. *Tret'yakov A., Zakrzewski l,, The Algorithm of the Global Optimization Based on t he Multilevel Computational Architecture // Proc. Artifical Intelligence Studies, 2006.–V. 3 (26).*

9. *Ортега Д., Рейнболдт В. Итерационные методы решения нелинейных систем уравнений со многими переменными.– М.: Мир, 1975.*

10. *Карманов В.Г. Математическое программирование. –М.: Наука, 1986.*

Статья поступила в редколлегию 5.05.2012

Рецензент: д-р техн. наук, проф., С.В. Цюцюра, Киевский национальный университет строительства и архитектуры, Киев.