УДК 004.021, 004.032.24, 004.051 **DOI: 10.14529/cmse230X0Z**

ПРИМЕНЕНИЕ МЕТОДА ПРОЕКТИРОВАНИЯ Q-ЭФФЕКТИВНЫХ ПРОГРАММ ДЛЯ АЛГОРИТМА ДЕЙКСТРЫ

© 2023 В.Н. Алеева, П.А. Манатин

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76),
E-mail: aleevavn@susu.ru, manatinpa@ya.ru

Поступила в редакцию: 21.10.2022

Проблема повышения эффективности параллельных вычислений чрезвычайно актуальна. В статье впервые продемонстрировано применение концепции Q-детерминанта для эффективной реализации алгоритма на графах. Концепция Q-детерминанта основана на унифицированном представлении численных алгоритмов в форме Q-детерминанта. Q-детерминант позволяет выразить и оценить внутренний параллелизм алгоритма, а также показать способ его параллельного исполнения. В работе приведены основные понятия концепции Q-детерминанта, необходимые для понимания приведенного исследования. Также описан основанный на концепции Q-детерминанта метод проектирования эффективных программ для численных алгоритмов. Результатом применения метода является программа, полностью использующая ресурс параллелизма алгоритма. Такая программа называется Q-эффективной. В качестве первого применения метода проектирования Q-эффективных программ для алгоритмов на графах описано проектирование программ для реализации алгоритма Дейкстры на параллельных вычислительных системах с общей и распределенной памятью. Приведены также результаты экспериментального исследования разработанных программ, проведенного с помощью суперкомпьютера «Торнадо ЮУрГУ». На основе анализа результатов экспериментального исследования определяются динамические характеристики разработанных программ и выявляются особенности их выполнения. Проведенные в статье исследования дают возможность сделать вывод, что применение концепции Q-детерминанта с целью разработки эффективных программ возможно не только для численных алгоритмов, но и для алгоритмов на графах.

Ключевые слова: повышение эффективности параллельных вычислений, Q-детерминант алгоритма, представление алгоритма в форме Q-детерминанта, Q-эффективная реализация алгоритма, ресурс параллелизма алгоритма, Q-эффективная программа, алгоритм Дейкстры

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Алеева В.Н., Манатин П.А. Применение метода проектирования Q-эффективных программ для алгоритма Дейкстры // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2023. Т. X, № Y. С. Z1–Z2. DOI: 10.14529/cmse230X0Z.

Введение

В настоящее время широко распространены параллельные вычислительные системы (далее — ПВС), к которым относятся, в том числе, персональные компьютеры с многоядерными процессорами. Архитектура ПВС позволяет сократить общее время решения задач с помощью последовательных программ за счет их одновременного выполнения. Однако, подобная организация работы программ во многих случаях не использует все доступные ресурсы вычислительной системы на всем промежутке времени выполнения данных программ. Для дальнейшего ускорения работы необходима разработка параллельных программ, то есть программ, производящих одновременные вычисления на нескольких ядрах или вычислительных узлах, причем эффективных программ, которые имеют наибольшую эффективность по сравнению с программами, выполняющими другие реализации данного алгоритма.

Целью данной работы является разработка эффективных программ для алгоритма Дейкстры [1] с помощью метода проектирования Q-эффективных программ на основе концепции Q-детерминанта. Для достижения цели решаются следующие задачи:

1) построение Q-детерминанта алгоритма Дейкстры;
2) описание Q-эффективной реализации алгоритма Дейкстры;
3) разработка параллельных программ для выполнимой Q-эффективной реализации алгоритма Дейкстры и исследование их динамических характеристик.

Статья относится к направлению исследований, целью которых является разработка эффективных параллельных программ, представленному работами [2–4], и вносит вклад в развитие этого направления. В данной статье описано проектирование эффективных программ, реализующих алгоритм Дейкстры на ПВС с общей и распределенной памятью.

Статья организована следующим образом. Она состоит из введения, четырех разделов и заключения. Раздел 1 содержит обзор работ по теме исследования. В разделе 2 приведены некоторые основные понятия концепции Q-детерминанта, используемые в статье, и изложен метод проектирования Q-эффективных программ [5]. В разделе 3 описано применение метода проектирования Q-эффективных программ для алгоритма Дейкстры. В разделе 4 представлены результаты экспериментального исследования динамических характеристик разработанных Q-эффективных программ, реализующих алгоритм Дейкстры. Заключение содержит краткое изложение полученных результатов и выводы об их применении.

1. Обзор работ по теме исследования

Проблеме эффективной реализации алгоритмов на ПВС посвящено много научных исследований. Приведем краткий обзор некоторых из них.

Очень важным и развитым направлением по исследованию параллельной структуры алгоритмов и программ с целью их реализации на ПВС является направление, основы которого изложены в [6, 7]. Результаты исследований данного направления используются при создании Интернет-энциклопедии AlgoWiki [8, 9]. В энциклопедии описываются свойства, особенности, статические и динамические характеристики конкретных алгоритмов. Это помогает реализовывать описанные алгоритмы эффективно. Однако в рамках данного направления программное исследование ресурса параллелизма алгоритмов не приводится. Кроме того, не предлагается технология создания параллельных программ, использующих весь ресурс параллелизма алгоритмов.

За время развития параллельных вычислений предложены различные подходы к разработке параллельных программ, созданы десятки языков параллельного программирования и множество различных инструментальных средств. Среди таких разработок отметим T-систему [10, 11]. Она представляет среду программирования с поддержкой автоматического динамического распараллеливания программ. Однако нет оснований полагать, что создаваемые с помощью T-системы параллельные программы используют ресурс параллелизма алгоритмов полностью.

Еще одним подходом к созданию параллельных программ является синтез параллельных программ. Метод синтеза параллельных программ заключается в том, чтобы из базы знаний параллельных алгоритмов конструировать новые параллельные алгоритмы для решения более крупных задач. На основе метода синтеза параллельных программ разработана технология фрагментированного программирования и реализующие ее язык и система программирования LuNA. В настоящее время это направление исследований развивает-

ся [12, 13]. Подход является универсальным, но при его применении не исследуется использование ресурса параллелизма алгоритмов.

Для преодоления ресурсных ограничений в статье [14] предлагаются методы построения параллельных архитектурно-независимых программ с использованием функционального языка программирования. Однако отсутствуют исследования касательно использования создаваемыми программами полного ресурса параллелизма алгоритмов.

Существует много исследований, в которых при разработке параллельных программ учитывается специфика алгоритмов и архитектуры ПВС. В качестве примеров таких исследований можно привести [15–20]. Эти исследования повышают эффективность реализации конкретных алгоритмов или реализации алгоритмов на ПВС конкретной архитектуры. Однако они не предлагают универсального подхода. Важно отметить, что в таких исследованиях нет информации о степени использования ресурса параллелизма алгоритма.

Реализации алгоритма Дейкстры с помощью параллельных программ посвящено несколько работ, например, [21–23]. Существенным отличием этой работы является применение для алгоритма на графах концепции Q-детерминанта, которая позволяет разработать программы, называемые Q-эффективными. Такие программы являются эффективными. В данном исследовании мы применяем для них часто используемые вычислительные инфраструктуры, т.е. условия разработки и выполнения программ. Динамические характеристики программы зависят от ресурса параллелизма реализуемого алгоритма и ее вычислительной инфраструктуры [4]. Так как вычислительных инфраструктур существует потенциально бесконечное множество, то для алгоритма не существует Q-эффективной программы с лучшими динамическими характеристиками. Однако, Q-эффективная программа наиболее эффективна для той вычислительной инфраструктуры, для которой она создавалась [4]. Для представления входных данных при вычислении кратчайших расстояний от выделенной вершины графа до всех остальных вершин в данном исследовании используется матрица смежности. Применение другого представления данных влечет то, что будет другой алгоритм для решения той же задачи. Новый алгоритм будет иметь другой Q-детерминант и, возможно, другой ресурс параллелизма алгоритма. Поэтому для сравнения наших результатов с другими нужно использовать те же алгоритм и вычислительную инфраструктуру, которые используются в нашем случае. Конечно, для данного алгоритма более хорошие динамические характеристики могут получиться при использовании другой инфраструктуры, но это будет лишь отражением выбора инфраструктуры. Следует отметить, что найти аналогичный результат практически невозможно, так как, как правило, работы не содержат описания всех характеристик вычислительной инфраструктуры.

2. Некоторые основные понятия концепции Q-детерминанта

Пусть α – алгоритм для решения алгоритмической проблемы $\bar{y} = F(N, B)$, где $N = \{n_1, n_2, \ldots, n_k\}$ – множество параметров размерности или пустое множество, B – множество входных данных, $\bar{y} = \{y_1, y_2, \ldots, y_k\}$, – множество выходных данных, $\bar{N}$ – вектор $(\bar{n}_1, \bar{n}_2, \ldots, \bar{n}_k)$ где $\bar{n}_i$– некоторое значение параметра n_i, $\{\bar{N}\}$ – множество всевозможных векторов $\bar{N}$, Q – множество операций, используемых алгоритмом α.

Определение 1. Любое однозначное отображение $w\colon \{\bar{N}\} \to V$, где V – множество всех выражений над B и Q, называется *безусловным Q-термом.*

Определение 2. Если при любом $\bar{N} \in \{\bar{N}\}$ и любой интерпретации переменных B $w\left(\bar{N}\right)$ принимает значение логического типа, то w называется *безусловным логическим Q-термом.*

Определение 3. Пусть $u_1, u_2, ..., u_l$ – безусловные логические Q-термы, $w_1, w_2, \ldots, w_l$ – безусловные Q-термы. Множество пар (u_i, w_i), где i=$1, 2, \ldots, l$, обозначается $(\bar{u},\ \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ и называется *условным Q-термом длины* l. Счетное множество пар безусловных Q-термов $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,2,\ldots}$ называется *условным бесконечным Q-термом*, если $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ является условным Q-термом для любого конечного l.

Определение 4. Под *вычислением безусловного Q-терма w при интерпретации B* следует понимать вычисление выражения $w\left(\bar{N}\right)$ при некотором $\bar{N} \in \{\bar{N}\}$. Для вычисления при заданной интерпретации B и $\bar{N} \in \{\bar{N}\}$ условного Q-терма $(\bar{u}, \bar{w}) = \{(u_i, w_i)\}_{i=1,\ldots,l}$ необходимо найти такие $u_{i_0}\left(\bar{N}\right), w_{i_0}\left(\bar{N}\right)$, что $u_{i_0}\left(\bar{N}\right)$ принимает значение *true*, а значение $w_{i_0}\left(\bar{N}\right)$ определено. В качестве значения $(\bar{u}, \bar{w})$ нужно взять $w_{i_0}\left(\bar{N}\right)$. Если установлено, что выражений $u_{i_0}\left(\bar{N}\right)$, $w_{i_0}\left(\bar{N}\right)$ не существует, то значение $(\bar{u}, \bar{w})$ для данной интерпретации B и N не определено. Вычисление условного бесконечного Q-терма определяется аналогично.

Определение 5. Предположим, что I_1, I_2, I_3 – подмножества множества $I = (1, ..., m)$ такие, что: одно или два из множеств $I_i (i = 1, 2, 3)$ могут быть пустыми, $I_1 \cup I_2 \cup I_1 = I, I_i \cap I_j = \varnothing$, где $i \neq j$. Множество Q-термов $\{f_i\}_{i \in I}$ удовлетворяет условиям: $f_{i_1} = w^{i_1} (i_1 \in I_1)$ – безусловный Q-терм, $f_{i_2} = (\bar{u}, \bar{w}) = \{(u_j^{i_2}, w_j^{i_2})\}_{\text{j}=1,\ldots,\text{l}} (i_2 \in I_2)$ – условный Q-терм, $f_{i_3} = (\bar{u}, \bar{w}) = \{(u_j^{i_3}, w_j^{i_3})\}_{i_3=1,2,\ldots} (i_3 \in I_3)$ – условный бесконечный Q-терм. Если алгоритм α состоит в том, что для определения $y_i (i \in I)$ требуется вычислить Q-терм f_i, то множество Q-термов $f_i (i \in I)$ называется *Q-детерминантом алгоритма*, а представление алгоритма в виде $y_i = f_i (i \in I)$ – *представлением в форме Q-детерминанта* [2].

Определение 6. *Реализацией алгоритма, представленного в форме Q-детерминанта*, называется вычисление Q-термов при заданной интерпретации входных данных. *Реализация алгоритма* называется *Q-эффективной*, если выражения вида: $W\left(\bar{N}\right) = \left\{w^{i_1}\left(i_1 \in I_1\right); u_j^{i_2}\left(\bar{N}\right), w_j^{i_2}\left(\bar{N}\right)\left(i_2 \in I_2; j = 1, 2, \ldots, l_{i_2}\right); u_j^{i_3}\left(\bar{N}\right), w_j^{i_3}\left(\bar{N}\right)\left(i_3 \in I_3; j = 1, 2, \ldots,\right)\right\}$ вычисляются одновременно и операции при этом выполняются по мере готовности. Такая реализация лучше всего использует ресурс параллелизма [3].

Определение 7. *Реализация алгоритма* называется *выполнимой*, если одновременно должно выполняться конечное (непустое) множество операций. Существуют алгоритмы, Q-эффективная реализация которых невыполнима [4].

Существует метод проектирования Q-эффективных программ, то есть программ, выполняющих Q-эффективную реализацию алгоритма. Он основан на концепции Q-детерминанта и использует то, что, во-первых, любой численный алгоритм представим в форме Q-детерминанта, а во-вторых, Q-детерминант содержит Q-эффективную реализацию этого алгоритма [3].

Метод состоит из следующих этапов:

1) построение Q-детерминанта алгоритма;
2) описание Q-эффективной реализации алгоритма;

3) разработка параллельной программы для выполнимой Q-эффективной реализации алгоритма (см. определение 7).

Этот метод возможно применять для разработки Q-эффективных программ как для общей памяти, так и для распределенной памяти с применением модели «Master–Slave».

Применение данного метода подробно описано в следующем разделе.

3. Применение метода проектирования Q-эффективных программ для эффективной реализации алгоритма Дейкстры

Опишем постановку решаемой задачи. В качестве представления графа будем использовать матрицу смежности A размера $n \times n$ простого ориентированного взвешенного графа без дуг отрицательного веса, содержащего n вершин. Для отметки посещения вершины в ходе выполнения алгоритма применяется вектор посещенных вершин $\vec{p}$ размера n. Для хранения результатов промежуточных вычислений и вывода конечного результата применяется вектор расстояний $\vec{r}$ размера n.

Алгоритм реализуется следующим образом.

1. Инициализируется вектор посещенных вершин $\vec{p}$ и вектор расстояний $\vec{r}$.
2. Заполняется вектор расстояний $\vec{r}$ из матрицы смежности A значениями расстояний от начальной вершины до всех остальных вершин.
3. Если есть непосещенные вершины в векторе посещенных вершин $\vec{p}$, то среди них выбирается вершина с наименьшим значением в векторе расстояний $\vec{r}$, иначе алгоритм переходит к шагу 7.
4. Выбранная вершина помечается как посещенная в векторе посещенных вершин $\vec{p}$.
5. Для каждой вершины, посещенной ранее на предыдущих итерациях, значение в векторе расстояний $\vec{r}$ принимает значение расстояния от помеченной вершины до данной, если подобный путь существует и эта сумма меньше значения в векторе расстояний $\vec{r}$.
6. Алгоритм возвращается к шагу 3.
7. Конец алгоритма.

Таким образом, критерием завершения алгоритма является выполнение такого условия, что все компоненты вектора посещенных вершин $\vec{p}$ являются помеченными. Очевидно, что цикл из шагов 3–6 повторяется n раз.

3.1. Представление алгоритма в форме Q-детерминанта

Опишем представление в форме Q-детерминанта для алгоритма Дейкстры. В данном алгоритме используется граф, представленного в виде матрицы смежности A с количеством вершин n, где $A = [a_{ij}]_{i,j=1,\ldots,n}$– матрица смежности простого ориентированного взвешенного графа. Пусть $\vec{p} = (p_1, \ldots, p_n)$ – вектор посещенных вершин, а $\vec{r} = (r_1, \ldots, r_n)$ – вектор расстояний от вершины r_1 до вершины r_i, где $i = 1, \ldots, n$. Процесс работы алгоритма можно представить как определенный процесс отметки вершин в векторе $\vec{p}$, за которым следует процесс заполнения по определенному правилу вектора расстояний $\vec{r}$. Таким образом, получаем алгоритм δ, реализующий алгоритм Дейкстры.

Процесс работы алгоритма δ на k-ом этапе можно представить в виде итераций:

$$r_j^k = \begin{cases} r_j^{k-1}, \text{ если } r_j^{k-1} \le r_i^k + a_{ij} \\ r_i^k + a_{ij}, \text{ если } r_j^{k-1} > r_i^k + a_{ij} \end{cases} \text{, где } j = 1, \ldots, n, j \ne i, a_{ij} \ne 0, \quad (1)$$

$\forall i \in (1, \ldots, n) : r_i^k = \min\left(r_i^{k-1}\right) \wedge p_i = \mathit{false}.$

Критерием завершения алгоритма является выполнение такого условия, что все вершины вектора $\vec{p}$ являются помеченными. Очевидно, что для достижения данного условия необходимо выполнить n этапов, после чего работа алгоритма завершается.

Этап 1. Система уравнений

$$= r_j = \left\{\left(u_1, w_1^j\right), \ldots, \left(u_{g(n)}, w_{g(n)}^j\right)\right\}_{j \in \{1,\ldots,n\}} \quad (2)$$

является представлением алгоритма δ в форме Q-детерминанта, состоящий из n условных Q-термов длины $g(n)$, где $g(n)$ – функция, определяющая длину Q-терма в зависимости от количества вершин n.

Этап 2. Опишем Q-эффективную реализацию алгоритма δ. В рамках реализации алгоритма необходимо поочередно выполнять вычисление r_j^k, $k = 1, \ldots, n$. Когда будет выполнен последний шаг, будет получено решение поставленной задачи. Q-эффективная реализация алгоритма является выполнимой, так как при реализации одновременно необходимо выполнять конечное число операций.

Этап 3. Данную реализацию можно использовать для разработки Q-эффективной программы для общей памяти. Опишем процесс реализации алгоритма δ для распределенной памяти с использованием модели «Master-Slave», при котором используется один вычислительный узел «Master» (обозначается буквой M) и несколько вычислительных узлов «Slave» (обозначаются буквой S). При подобной организации общие данные для узлов S на очередной итерации рассылаются и затем результаты вычислений от них обрабатываются только узлом M, что во многих случаях сокращает частоту обмена данными между узлами.

1. Каждая компонента вектора $\vec{r}$ вычисляется на отдельном узле S. Если количество узлов S меньше n, то узлы S должны выполнять вычисления для нескольких компонент вектора. Перед вычислением r_i^0, где $i \in \{1, 2, \ldots, n\}$, узел S получает от узла M строку матрицы A с номером i
2. Полученные на узлах S результаты, являющиеся компонентами вектора $\vec{r}^0$, передаются на узел M для вычисления поэлементных минимумов вектора $\vec{r}^0$
3. Аналогично шагу 1, производится вычисление $\vec{r}^1$ на узлах S
4. Полученные на узлах S результаты, являющиеся компонентами вектора $\vec{r}^1$, передаются на узел M для вычисления поэлементных минимумов вектора $\vec{r}^1$
5. Если компоненты вектора $\vec{r}^0$ не равны компонентам вектора $\vec{r}^1$, итерации алгоритма δ продолжаются аналогично шагам 3–5.

4. Разработка и экспериментальное исследование Q-эффективных программ

В нашем исследовании при разработке Q-эффективных программ используются:

1) язык программирования $C{+}{+}11$;

2) компилятор *GCC* версии 10.1.0 с поддержкой стандарта *OpenMP* без использования опций оптимизации;
3) библиотека *OpenMPI* версии 2.1.0, реализующая технологию *MPI* для ПВС с распределенной памятью.

Технология *OpenMP* позволяет распределить итерации циклов между нитями при инициализации входных данных и обновлении значений промежуточных данных на очередном этапе выполнения алгоритма. Технология *MPI* используется для распределения элементов данных по вычислительным узлам, решения на данных узлах локальных задач и использования полученных результатов для решения исходной задачи по некоторым правилам.

Для выполнения Q-эффективной реализации алгоритма δ были разработаны Q-эффективные программы для общей памяти и для распределенной памяти с применением модели «Master-Slave», проектирование которых описано в разделе 3.

Для всех разработанных программ было проведено экспериментальное исследование динамических характеристик. Оно проводилось на суперкомпьютере «Торнадо» Южно-Уральского государственного университета. Для программы для общей памяти был использован один вычислительный узел, для распределенной памяти — несколько вычислительных узлов. В экспериментах у каждого из использованных вычислительных узлов были задействованы все имеющиеся два центральных процессора Intel Xeon X5680 с частотой 3.33 ГГц, каждый из которых имеет 6 ядер и поддерживает 12 нитей [24]. Таким образом, на каждом узле может одновременно выполняться до 24 нитей.

В экспериментах исследуются динамические характеристики программы в зависимости от количества нитей и количества вычислительных узлов (только в случае программы для распределенной памяти) на графах с разным количеством вершин. В ходе экспериментов на каждом вычислительном узле выполняется только один процесс, соответствующий рассматриваемой программе. Кроме того, в программе для общей памяти приведены результаты для разного числа нитей, исполняемых на каждом вычислительном узле. В программе для распределенной памяти на каждом вычислительном узле всегда выполняются 24 нити, что соответствует максимально возможному числу нитей, которые могут выполняться одновременно на узле. При экспериментальном исследовании находится вектор расстояний $\vec{r}$. Компоненты матрицы A формировались путем генерации случайных чисел. Измерения для каждой конфигурации проводились несколько раз.

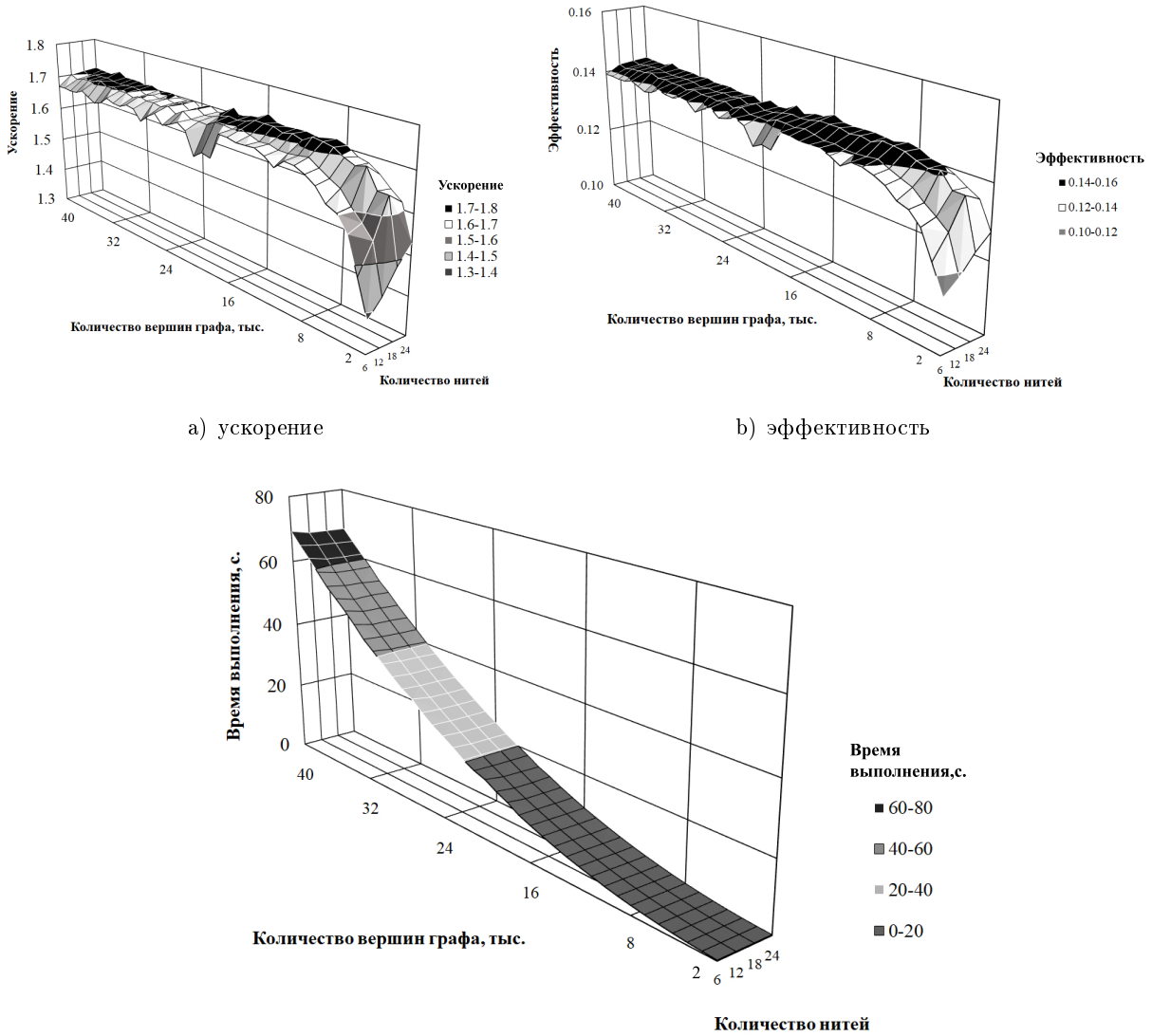
На основании анализа полученных рядов измерений времени выполнения для определенных сочетаний конфигурации и размерности задачи можно сделать вывод об отсутствии оснований для опровержения гипотезы о подчинении этих рядов нормальному закону распределения по критерию Пирсона. На подчинение рядов нормальному закону распределения косвенно указывает тот факт, что в данных рядах выборочная средняя почти равна выборочной медиане, а мода отсутствует, так как значения в рядах не повторяются.

На основании полученных измерений времени выполнения определяются следующие динамические характеристики программ [7].

1. Ускорение — отношение времени выполнения программы для общей памяти, выполняемой на одном ядре одного вычислительного узла с использованием одной нити (назовем ее последовательной) ко времени выполнения параллельной программы.
2. Эффективность — отношение ускорения к количеству используемых параллельной программой вычислителей. Количество вычислителей, используемых программой для об-

щей памяти, соответствует количеству нитей, а используемых программой для распределенной памяти — количеству вычислительных узлов.

На рисунке 1 показаны графики времени выполнения, ускорения и эффективности Q-эффективной программы для общей памяти. Рисунок 2 содержит графики времени выполнения, ускорения и эффективности Q-эффективной программы для распределенной памяти. На данных графиках в качестве значений указаны соответствующие значения выборочных средних.

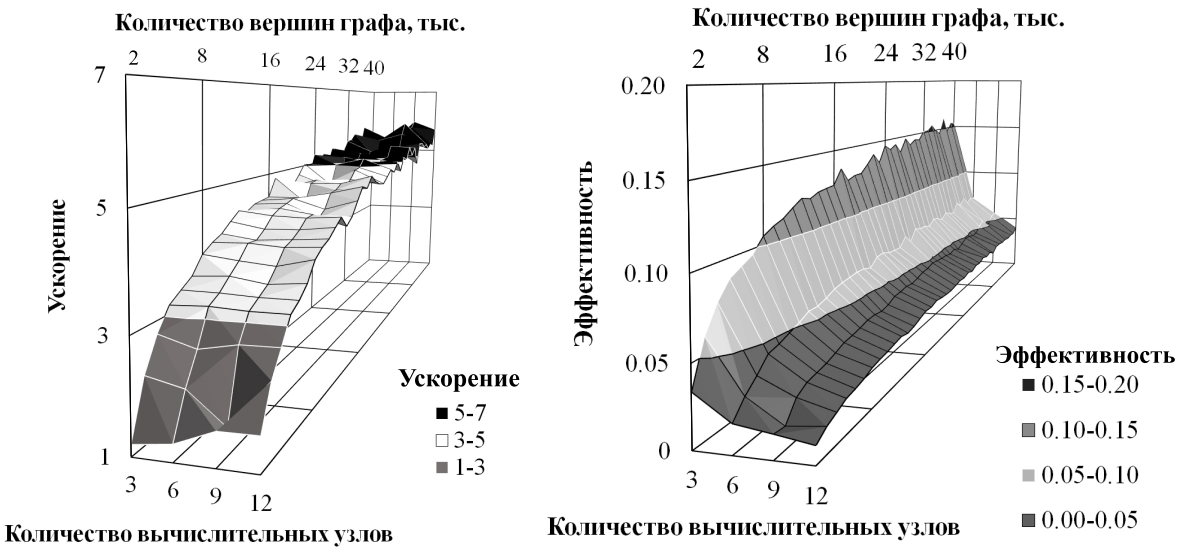


a) ускорение

b) эффективность

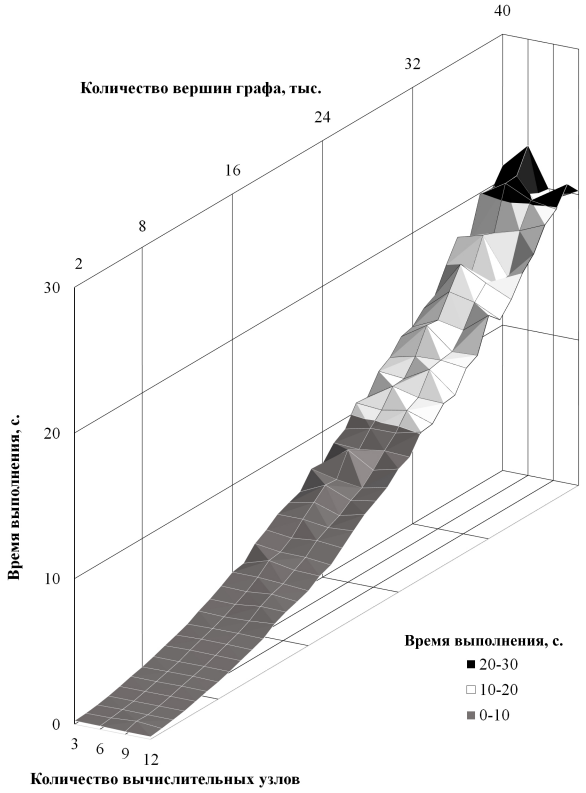
c) время выполнения

Рис. 1. Динамические характеристики программы для общей памяти



a) ускорение

b) эффективность



c) время выполнения

Рис. 2. Динамические характеристики Q-эффективной программы для распределенной памяти

Охарактеризуем результаты экспериментального исследования.

1. Величина ускорения Q-эффективной программы для общей памяти в зависимости от количества вычислителей и вершин графа варьируется в диапазоне 1.3–1.8. Минимальное ускорение достигнуто при минимальном числе вершин графа (2 тыс.) и минимальном количестве вычислителей (6 нитей), также при данном сочетании достигнута наи-

меньшая разница во времени выполнения последовательной и параллельной программ. Ускорение возрастает по мере роста числа вершин графа, затем достигает максимального значения, равного около 1.8, при работе 18 вычислителей и наличии 9 тыс. вершин графа. При таком сочетании параметров достигается наибольший эффект от использования многопоточности. Далее значение ускорения плавно уменьшается до значения около 1.6 по мере дальнейшего роста числа вершин графа, что указывает на ухудшение локальности данных алгоритма.

2. Q-эффективная программа для распределенной памяти при большинстве сочетаний параметров может продемонстрировать большее ускорение относительно Q-эффективной программы для общей памяти. Величина ускорения относительно последовательной программы в зависимости от количества вычислителей и числа вершин графа варьируется в диапазоне 1.2–6.0. Зависимость ускорения Q-эффективной программы для распределенной памяти от изучаемых факторов имеет сложный характер. Описание этой зависимости приведены в следующих пунктах.
3. Наименьшее ускорение (меньше 2) достигается в области малых по числу вершин графов. Это означает, что на малых задачах организация работы нитей отнимает большее время, чем межпроцессное взаимодействие и косвенно указывает на хорошую локальность данных алгоритма, реализованного для систем с распределенной памятью [7].
4. Наибольшее ускорение (больше 5) достигается в области больших по числу вершин графов. Так как рассматриваемый алгоритм использует матрицу смежности, то объем входных данных зависит от квадрата количества вершин графа. В таком случае возрастающее ускорение косвенно указывает на факт того, что при существенном увеличении объема входных данных накладные расходы увеличиваются незначительно, и тогда их доля в общем времени выполнения уменьшается.
5. Ускорение программы для распределенной памяти, равно как и программы для общей памяти, не зависит линейно от количества вычислителей. Это связано с тем, что по мере увеличения количества вычислителей растут и накладные расходы. В случае программы для общей памяти они связаны с организацией работы нитей, а в случае программы для распределенной памяти, кроме того — с увеличением в общем времени выполнения доли задержек в коммуникационной среде по мере роста общего числа передаваемых узлами сообщений, хоть и становящихся меньшими по объему.
6. Наиболее эффективными с точки зрения распараллеливания являются конфигурации Q-эффективной программы для распределенной памяти, использующие небольшое число узлов и большое количество нитей, т.к. при несущественном росте количества вычислителей и использовании многопоточности достигаются наименьшие накладные расходы и, как следствие, максимальный прирост ускорения.

Заключение

В статье впервые показано применение метода проектирования Q-эффективных программ, основанного на концепции Q-детерминанта, для эффективной реализации алгоритма на графах, сводящегося к численному алгоритму, на примере алгоритма Дейкстры. Для этого были решены следующие задачи:

1) построение Q-детерминанта алгоритма Дейкстры;
2) описание Q-эффективной реализации алгоритма Дейкстры;

3) разработка Q-эффективных программ для алгоритма Дейкстры, предназначенных для ПВС с общей и распределенной памятью, и их экспериментальное исследование.

Данное исследование показывает, что описание эффективной реализации возможно не только для численных алгоритмов, но и для алгоритмов других разновидностей, в частности, для алгоритмов на графах. Подобная реализация существует, если алгоритмы можно описать как численные.

Проведенное исследование пополнило коллекцию алгоритмов, для которых разработаны Q-эффективные программы и оценены их динамические характеристики. Применение метода проектирования Q-эффективных программ решает проблему наиболее полного использования ресурса параллелизма численных алгоритмов и тем самым делает возможной эффективную реализацию численных алгоритмов на ПВС. Программная Q-система для исследования ресурса параллелизма численных алгоритмов [25, 26] метод проектирования Q-эффективных программ и технология Q-эффективного программирования [5] в комплексе являются одним из решений проблемы повышения эффективности параллельных вычислений, использующих численные алгоритмы. Их могут применять разработчики программного обеспечения для проектирования эффективных программ, предназначенных для любых ПВС — от персональных компьютеров с многоядерными процессорами до суперкомпьютеров. Это приведет к уменьшению времени выполнения программного обеспечения и более полному использованию ресурсов вычислительных систем.

Литература

1. Dijkstra E.W. A note on two problems in connexion with graphs // Numerische Mathematik. 1959. Vol. 1, no. 1. P. 269–271. DOI: 10.1007/BF01386390.

2. Алеева В.Н., Алеев Р.Ж. Применение Q-детерминанта численных алгоритмов для параллельных вычислений // Параллельные вычислительные технологии – XIII международная конференция, ПаВТ'2019, г. Калининград, 2–4 апреля 2019 г. Короткие статьи и описания плакатов. Челябинск: Издательский центр ЮУрГУ, 2019. С. 133–145.

3. Алеева В.Н. Основные положения технологии Q-эффективного программирования // Наука ЮУрГУ. Секции технических наук: материалы 71-й научной конференции (Челябинск, апрель 2019 г.). Челябинск: Издательский центр ЮУрГУ, 2019. С. 334–342.

4. Алеева В.Н., Шатов М.Б. Применение концепции Q-детерминанта для эффективной реализации численных алгоритмов на примере метода сопряженных градиентов для решения систем линейных уравнений // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 56–71. DOI: 10.14529/cmse210304.

5. Aleeva V.N. Improving Parallel Computing Efficiency // Proceedings – 2020 Global SmartIndustry Conference, GloSIC 2020. IEEE. Chelyabinsk, Russia. November 17–19. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

6. Voevodin V.V., Voevodin V.V. The V-Ray technology of optimizing programs to parallel computers. In: Vulkov L.G., Yalamov P., Wasniewski J. (eds.) WNAA 1996. LNCS, Vol. 1196, P. 546–556. Springer, Heidelberg (1997). DOI: 10.1007/3-540-62598-4_136.

7. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с.

8. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations, Vol. 2, no. 1. 2015. P. 4–18. DOI: 10.14529/jsfi150101.

9. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki Project as an Extension of the Top500 Methodology. Supercomputing Frontiers and Innovations, Vol. 5, no. 1. 2018. P. 4–10. DOI: 10.14529/jsfi180101.

10. Абрамов С.М., Адамович А.И., Коваленко М. Р. Т-система – среда программирования с поддержкой автоматического динамического распараллеливания программ. Пример реализации алгоритма построения изображений методом трассировки лучей // Программирование. 1999. 25 (2). С. 100–107.

11. Абрамов С.М., Васенин В.А., Мамчиц Е.Е., Роганов В.А., Слепухин А.Ф. Динамическое распараллеливание программ на базе параллельной редукции графов. Архитектура программного обеспечения новой версии Т-системы // Научная сессия МИФИ–2001, 22–26 января 2001 г.: Сборник научных трудов. Т. 2. 2001. С. 234.

12. Malyshkin V.E., Perepelkin V.A., Schukin G.F. Distributed Algorithm of Data Allocation in the Fragmented Programming // Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31–September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS, Vol. 9251. P. 80–85. DOI: 10.1007/978-3-319-21909-7_8.

13. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. Control Flow Usage to Improse Performanse of Fragmented // Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS, Vol. 9251. P. 86–90. DOI: 10.1007/978-3-319-21909-7_9.

14. Легалов А.И. Функциональный язык для создания архитектурно-независимых параллельных программ // Вычислительные технологии. 2005. № 1 (10). С. 71–89.

15. Gurieva Y.L., Il'in V.P. On Parallel Computational Technologies of Augmented Domain Decomposition Methods// Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31–September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland, 2015. LNCS, Vol. 9251. P. 35–46. DOI: 10.1007/978-3-319-21909-7_4.

16. Schlueter M., Munetomo M. Parallelization strategies for evolutionary algorithms for MINLP // IEEE Congress on Evolutionary Computation. Cancun, Mexico. June 23–25. 2013. P. 635–641. DOI: 10.1109/CEC.2013.6557628.

17. Wang Q., Liu J., Tang X., *et al.* Accelerating embarrassingly parallel algorithm on Intel MIC // IEEE International Conference on Progress in Informatics and Computing. Shanghai, China. May 16–18. 2014. P. 213–218. DOI: 10.1109/PIC.2014.6972327.

18. Li Y., Dou W., Yang K., *et al.* Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis // 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. Xi'an, China. November 24–27. 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

19. Prifti V., Bala R., Tafa I., *et al.* The time profit obtained by parallelization of quicksort algorithm used for numerical sorting // Science and Information Conference (SAI). London, UK. July 28–30. 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

20. Rajashri A. Parallelization of shortest path algorithm using OpenMP and MPI // International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). Palladam, India. February 10–11. 2017. P. 304–309. DOI: 10.1109/I-SMAC.2017.8058360.

21. Fazio M., Buzachis A., Galletta A., *et al.* A Map-Reduce Approach for the Dijkstra Algorithm in SDN Over Osmotic Computing Systems // International Journal of Parallel Programming. Mar. 2021. Vol. 49, no. 3. P. 347–375. DOI: 10.1007/s10766-021-00693-3.

22. Zhang W., Zhang L., Chen Y. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor // International Conference on Algorithms and Architectures for Parallel Processing. 2018. P. 337–357. DOI: 10.1007/978-3-030-05051-1_24.

23. Jasika N., Alispahic N., Elma A.,*et al.* Dijkstra's shortest path algorithm serial and parallel execution performance analysis // 2012 Proceedings of the 35th International Convention MIPRO. Opatija, Croatia. May 21–25. 2012. P. 1811–1815. URL: `https://ieeexplore.ieee.org/document/6240942/` (дата обращения: 14.03.2023 г.).

24. Биленко Р. В., Долганина Н.Ю., Иванова Е. В., Рекачинский А. И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университета // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2022. Т. 11, № 1. С. 15–30. DOI: 10.14529/cmse220102.

25. Алеева В.Н., Зотова П.С., Склезнев Д.С. Расширение возможностей исследования ресурса параллелизма численных алгоритмов с помощью программной Q-системы // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 2. С. 66–81. DOI: 10.14529/cmse210205.

26. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism // Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. Moscow, Russia. September 23–24. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.

Алеева Валентина Николаевна, к.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Манатин Павел Андреевич, студент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse230X0Z

APPLICATION OF THE DESIGN METHOD FOR Q-EFFICIENT PROGRAMS IMPLEMENTING DIJKSTRA'S ALGORITHM

© 2023 V.N. Aleeva, P.A. Manatin

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: aleevavn@susu.ru, manatinpa@ya.ru

Received: 21.10.2022

The problem of improving the efficiency of parallel computing is very topical. The article demonstrates the initial application of the concept of Q-determinant for the effective implementation of graph algorithms. The concept of the Q-determinant is based on a unified representation of numerical algorithms in the form of the Q-determinant. The Q-determinant allows you to express and evaluate the internal parallelism of the algorithm, as well as to show the method of its parallel execution. The article gives the main notions of the Q-determinant concept necessary for better understanding of our research. Also, we describe a method of designing effective programs for numerical algorithms on the base of the concept of the Q-determinant. As a result, we obtain the program which uses the parallelism resource of the algorithm completely, and this program is called Q-effective. As the initial application of the method for design of Q-effective programs implementing graph algorithms, we describe the designing programs for Dijkstra's algorithm implementation on parallel computing systems with shared and distributed memory. Finally, for developed programs we present the results of experiments on a supercomputer «Tornado SUSU». Analyzing the results of the experimental study, we determine the effectiveness of the developed programs and identify the features of their execution. The research described in the article leads to the conclusion that the application of the Q-determinant concept for the development of effective programs is possible not only for numerical algorithms, but also for graph algorithms.

Keywords: improving parallel computing efficiency, Q-determinant of algorithm, representation of algorithm in the form of Q-determinant, Q-effective implementation of algorithm, parallelism resource of algorithm, Q-effective program, Dijkstra's algorithm.

FOR CITATION

Application of the design method for Q-effective programs implementing Dijkstra's algorithm. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2023. Vol. X, no. Y. P. Z1–Z2. (in Russian) DOI: 10.14529/cmse230X0Z.

References

1. Dijkstra E.W. A note on two problems in connexion with graphs. Numerische Mathematik. 1959. Vol. 1, no. 1. P. 269–271. DOI: 10.1007/BF01386390.

2. Aleeva V.N., Aleev R.Zh. Application of the Q-determinant of numerical algorithms for parallel computing. Parallel computing Technologies – XIII International Conference, PaVT'2019, Kaliningrad, April 2–4, 2019. Short articles and poster descriptions. Chelyabinsk: SUSU Publishing Center, 2019. P. 133–145. (in Russian)

3. Aleeva V.N. The fundamentals of Q-effective programming technology. Science of SUSU. Sections of Technical Sciences: materials of the 71st Scientific Conference (Chelyabinsk,

April 2019). Chelyabinsk: SUSU Publishing Center, 2019. P. 334–342. (in Russian)

4. Aleeva V.N., Shatov M.B. Application of the Q-determinant Concept for Efficient Implementation of Numerical Algorithms by the Example of the Conjugate Gradient Method for Solving Systems of Linear Equations. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 56–71. (in Russian) DOI: 10.14529/cmse210304.

5. Aleeva V.N. Improving Parallel Computing Efficiency. Proceedings – 2020 Global SmartIndustry Conference, GloSIC 2020. IEEE. Chelyabinsk, Russia. November 17–19. 2020. P. 113–120. Article number 9267828. DOI: 10.1109/GloSIC50886.2020.9267828.

6. Voevodin V.V., Voevodin V.V. The V-Ray technology of optimizing programs to parallel computers. In: Vulkov L.G., Yalamov P., Wasniewski J. (eds.) WNAA 1996. LNCS, Vol. 1196, P. 546–556. Springer, Heidelberg (1997). DOI: 10.1007/3-540-62598-4_136.

7. Voevodin V.V., Voevodin Vl.V. Parallel Computing. St. Petersburg, BHV-Petersburg, 2002. 608 p. (in Russian)

8. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki: an Open Encyclopedia of Parallel Algorithmic Features. Supercomputing Frontiers and Innovations, Vol. 2, no. 1. 2015. P. 4–18. DOI: 10.14529/jsfi150101.

9. Voevodin Vl., Antonov A., Dongarra J. AlgoWiki Project as an Extension of the Top500 Methodology. Supercomputing Frontiers and Innovations, Vol. 5, no. 1. 2018. P. 4–10. DOI: 10.14529/jsfi180101.

10. Abramov S.M., Adamovich A.I., Kovalenko M. R. T-system programming environment with support for automatic dynamic parallelization of programs. An example of the implementation of an algorithm for constructing images by ray tracing. Programming. 1999. 25 (2). pp. 100–107. (in Russian)

11. Abramov S.M., Vasenin V.A., Mamchits E.E., Roganov V.A., Slepukhin A.F. Dynamic parallelization of programs based on parallel graph reduction. Software architecture of the new version of the T-system. Scientific session of MEPhI–2001, January 22–26, 2001: Collection of scientific papers. Vol. 2. 2001. p. 234. (in Russian)

12. Malyshkin V.E., Perepelkin V.A., Schukin G.F. Distributed Algorithm of Data Allocation in the Fragmented Programming. Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31–September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS, Vol. 9251. P. 80–85. DOI: 10.1007/978-3-319-21909-7_8.

13. Malyshkin V.E., Perepelkin V.A., Tkacheva A.A. Control Flow Usage to Improse Performanse of Fragmented. Parallel Computing Technologies: 13th International Conference, PaCT 2015, Petrozavodsk, Russia, August 31–September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland. 2015. LNCS, Vol. 9251. P. 86–90. DOI: 10.1007/978-3-319-21909-7_9.

14. Legalov A.I. Functional language for architecturally independent parallel programs creating. Computational Technologies. 2005. No. 1 (10). pp. 71–89 (in Russian)

15. Gurieva Y.L., Il'in V.P. On Parallel Computational Technologies of Augmented Domain Decomposition Methods. Parallel Computing Technologies: 13th International Conference,

PaCT 2015, Petrozavodsk, Russia, August 31-September 4, 2015, Proceedings. V. Malyshkin (Ed.). Springer International Publishing Switzerland, 2015. LNCS, Vol. 9251. P. 35–46. DOI: 10.1007/978-3-319-21909-7_4.

16. Schlueter M., Munetomo M. Parallelization strategies for evolutionary algorithms for MINLP. IEEE Congress on Evolutionary Computation. Cancun, Mexico. June 23–25. 2013. P. 635–641. DOI: 10.1109/CEC.2013.6557628.

17. Wang Q., Liu J., Tang X., *et al.* Accelerating embarrassingly parallel algorithm on Intel MIC. IEEE International Conference on Progress in Informatics and Computing. Shanghai, China. May 16–18. 2014. P. 213–218. DOI: 10.1109/PIC.2014.6972327.

18. Li Y., Dou W., Yang K., *et al.* Optimized Data I/O Strategy of the Algorithm of Parallel Digital Terrain Analysis. 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science. Xi'an, China. November 24–27. 2014. P. 34–37. DOI: 10.1109/DCABES.2014.10.

19. Prifti V., Bala R., Tafa I., *et al.* The time profit obtained by parallelization of quicksort algorithm used for numerical sorting. Science and Information Conference (SAI). London, UK. July 28–30. 2015. P. 897–901. DOI: 10.1109/SAI.2015.7237248.

20. Rajashri A. Parallelization of shortest path algorithm using OpenMP and MPI. International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC). Palladam, India. February 10–11. 2017. P. 304–309. DOI: 10.1109/I-SMAC.2017.8058360.

21. Fazio M., Buzachis A., Galletta A., *et al.* A Map-Reduce Approach for the Dijkstra Algorithm in SDN Over Osmotic Computing Systems. International Journal of Parallel Programming. Mar. 2021. Vol. 49, no. 3. P. 347–375. DOI: 10.1007/s10766-021-00693-3.

22. Zhang W., Zhang L., Chen Y. Asynchronous Parallel Dijkstra's Algorithm on Intel Xeon Phi Processor International Conference on Algorithms and Architectures for Parallel Processing. 2018. P. 337–357. DOI: 10.1007/978-3-030-05051-1_24.

23. Jasika N., Alispahic N., Elma A., *et al.* Dijkstra's shortest path algorithm serial and parallel execution performance analysis. 2012 Proceedings of the 35th International Convention MIPRO. Opatija, Croatia. May 21–25. 2012. P. 1811–1815. URL: `https://ieeexplore.ieee.org/document/6240942/` (accessed: 14.03.2023 г.).

24. R.V. Bilenko, N.Yu. Dolganina, E.V. Ivanova, A.I. Rekachinsky. High-performance computing resources of South Ural State University. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2022. Vol. 11, no. 1. P. 15–30. DOI: 10.14529/cmse220102.

25. Aleeva V.N., Zotova P.S., Skleznev D.S. Advancement of research for the parallelism resourceof numerical algorithms with help of software Q-system. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 2. P. 66–81. (in Russian) DOI: 10.14529/cmse210205.

26. Aleeva V., Bogatyreva E., Skleznev A., *et al.* Software Q-system for the Research of the Resource of Numerical Algorithms Parallelism. Supercomputing. RuSCDays 2019. Communications in Computer and Information Science. Moscow, Russia. September 23–24. 2019. Vol. 1129. P. 641–652. DOI: 10.1007/978-3-030-36592-9_52.